

# PROPOSAL TESIS



**Telkom  
University**

IAN AGUNG PRAKOSO – 2101191002  
PROGRAM STUDI MAGISTER TEKNIK ELEKTRO  
UNIVERSITAS TELKOM UNIVERSITY  
2021

## Table of Contents

<b>Abstrak:</b> .....	3
<b>BAB I PENDAHULUAN</b> .....	4
1.1    Background.....	4
1.2    Rumusan Masalah.....	5
1.3    Tujuan Penelitian.....	5
1.4    Hypothesis.....	6
1.5    Metodologi .....	6
1.6    Metode Penelitian.....	7
<b>BAB II STUDI PUSTAKA.</b> .....	8
2.1    Arsitektur Software Define Networks (SDN).....	8
2.1.1    Application Plane .....	9
2.1.2    Control Plane.....	9
2.1.3    Data Plane .....	9
2.1.4    NourthBound Interface .....	9
2.1.5    SouthBound Interface.....	9
2.1.6    Kelebihan SDN .....	10
2.2    Protokol <i>OpenFlow</i> .....	10
2.3    SDN Controller.....	12
2.3.1    SDN Applications .....	12
2.3.2    RYU .....	13
2.4    Mininet .....	14
2.5    Fuzzy Logic .....	14
2.5.1    Kelebihan Logika Fuzzy .....	15
2.5.2    Himpunan Fuzzy .....	15
2.5.3    Fungsi Keanggotaan.....	17
2.5.4    Representasi Linear .....	17
2.6    Teknik Load Balancing.....	19
2.6.1    Jenis Teknik <i>Load Balancing</i> .....	20
2.7    Teknik LBBSRT – Load Balancing Based Server Response Time .....	21
2.7.1    Pengukuran <i>Real-time</i> untuk mendapatkan <i>server response time</i> [2].....	21
2.7.2    Pemrosesan <i>User Request</i> [2].....	22
<b>BAB III DESAIN SISTEM DAN SKENARIO SIMULASI</b> .....	23
3.1.    Model Sistem.....	23
3.2.    Skenario Simulasi.....	29

PROPOSAL THESIS

3.3.	Skenario Pengambilan Data.....	29
3.4.	Skenario Analisis.....	32
	<i>Bibliography</i> .....	33
	<i>Lampiran 1</i> .....	34
	<i>Lampiran 2</i> .....	39
	<i>Lampiran 3</i> .....	45

**Abstrak:**

Research Project Title: Analysis of Fuzzy Logic Algorithm for Load Balancing in SDN

Submitted by: ian Agung Prakoso

NIM: 2101191002

Concentration: Network Engineering – Electrical Engineering

E-mail address: ianagung@student.telkomuniversity.ac.id

Supervisor (I): Mrs. Sofia Naning Hertiana

Supervisor (II): Mr. Favian Dewanta

Submitted Date:

Topic Summary (Abstract): (max 300 words)

Perkembangan teknologi pada bidang telekomunikasi belakangan ini berkembang sangat pesat, dimana perkembangannya dapat memudahkan dalam hal pembangunan, *monitoring* dan *maintenance*. Seiring berjalanannya waktu, perkembangan teknologi pada bidang telekomunikasi khususnya pada jaringan memunculkan sebuah konsep baru yaitu *Software Defined Network* (SDN). SDN adalah arsitektur jaringan baru yang dapat memisahkan bidang kontrol dan data serta menyediakan kemampuan yang bisa diprogram dan terstandarisasi. SDN memungkinkan tampilan jaringan global dan memberikan kemampuan untuk menggunakan sumber daya jaringan secara efisien. SDN juga mengurangi beban manajemen jaringan dan meningkatkan fleksibilitas jaringan. Kelebihan SDN dalam mengontrol jaringan dapat dimanfaatkan dengan berbagai strategi *load balancing* yang digunakan untuk mendistribusikan beban trafik guna meningkatkan kinerja sistem secara keseluruhan.

Pada Jaringan SDN, permasalahan *load imbalance* masih bisa terjadi. Jika proses pendistribusian beban server tidak rata, akan mempengaruhi performansi dan efisiensi keseluruhan jaringan. Dengan menggunakan strategi *Server Load Balancing*, maka beban trafik dapat dialokasikan ke server yang berbeda sehingga menghindari terjadinya *link congestion* serta meningkatkan performansi jaringan. Beban yang seimbang dalam jaringan dapat memaksimalkan *throughput*, meminimalkan *response time*, dan menghindari kelebihan beban pada jaringan. *Response time* adalah faktor terpenting yang menentukan *user experience* dalam model penyediaan layanan yang melibatkan cluster server. Namun, skema *load balancing* cluster server tradisional tidak menggunakan data kondisi perangkat dan tidak dapat sepenuhnya memanfaatkan waktu respons server untuk *load balancing*.

Dalam penelitian ini, kami mengusulkan metode Server *Load Balancing* berbasis SDN dengan menggunakan metode logika fuzzy. Dengan menggunakan waktu *respons* dan kondisi perangkat dari setiap server yang diukur, peneliti memproses permintaan pengguna agar mendapatkan beban server yang seimbang. Hasil yang diharapkan yaitu server mampu memproses permintaan dengan waktu respons server rata-rata minimum dan adanya peningkatan kualitas pada parameter *throughput* dan *delay*, serta menggambarkan tidak terjadinya overload pada server ketika lalu lintas padat.

## BAB I

### PENDAHULUAN

#### 1.1 Background

Perkembangan teknologi pada bidang telekomunikasi belakangan ini berkembang sangat pesat, dimana perkembangannya dapat memudahkan dalam hal pembangunan, *monitoring* dan *maintenance*. Seiring berjalanannya waktu, perkembangan teknologi pada bidang telekomunikasi khususnya pada jaringan memunculkan sebuah konsep baru yaitu *Software Defined Network* (SDN). SDN adalah salah satu arsitektur baru pada jaringan yang bersifat *dynamic*, *manageable*, *cost-effective*, dan *adaptable*. Arsitektur SDN bertujuan untuk membuat jaringan menjadi lebih fleksibel dan mempermudah dalam mengontrol jaringan. Kemampuan yang dimiliki oleh jaringan SDN juga dapat digunakan untuk mengubah perilaku jaringan serta dapat melakukan perubahan secara otomatis seperti memaksimalkan pembagian beban trafik dengan menggunakan *load balancing*.

Pada Jaringan SDN permasalahan *Load Imbalance* masih bisa terjadi. Jika proses pen-distribusian beban jaringan tidak rata, akan mempengaruhi performansi dan efisiensi keseluruhan jaringan. Secara umum, beban trafik yang seimbang dalam jaringan membantu mengoptimalkan pemanfaatan *resource* yang tersedia dengan cara memaksimalkan *throughput*, meminimalkan waktu respon, dan menghindari *overloading* beban pada jaringan. Oleh karena itu, teknologi *Load Balancing* dipakai untuk membagi sumber daya jaringan secara merata agar performansi jaringan meningkat. Teknologi *Load Balancing* di SDN bisa diklasifikasikan menjadi *Server Load Balancing* dan *Link Load Balancing*. Ketika terjadi *link congestion* yang disebabkan oleh gangguan pada *Traffic Scheduling Management* di jaringan maka performansi jaringan menjadi turun dan delay transmisi menjadi naik. Dengan menggunakan strategi *Server Load Balancing* maka kita bisa meng-alokasikan beban trafik ke server yang berbeda sehingga menghindari terjadinya *link congestion* [1].

Metode LBBSRT memberikan solusi *Load Balancing* dengan menggunakan metode *Load Balancing Based Server Response Time* (LBBSRT). LBBSRT menggunakan SDN kontroler untuk mendapatkan nilai *response time* dari setiap server dengan tujuan untuk memilih sebuah server dengan nilai *response time*. SDN kontroler mengirim *multiple Packet-out messages* ke *switch* dengan *time interval t* dan menyimpan hasilnya berupa waktu transmisi. Metode LBBSRT tidak mempertimbangkan sisi konsumsi energi dan *resource* pada server [2].

Peneliti mengusulkan solusi baru dalam menyelesaikan permasalahan tersebut dengan sistem *Load Balancing* berbasis SDN dengan menggunakan metode Algoritma Logika Fuzzy serta melakukan penelitian terkait efisiensi energi algoritma tersebut.

## 1.2 Rumusan Masalah

- a. Pendistribusian beban *Server* yang belum seimbang / *balanced* sehingga memerlukan metode *Load Balancing*.
- b. Metode *Load Balancing* pada penelitian sebelumnya belum optimal karena belum memperhatikan permasalahan penggunaan energi di server.
- c. Nilai *response time*, *cpu* dan *RAM usage* yang harus dihasilkan oleh setiap server dan digunakan bersamaan sebagai inputan sistem Load Balancing yang lebih baik.
- d. Pemilihan nilai parameter metode Algoritma Logika Fuzzy yang tepat untuk memilih server dengan beban lebih kecil.
- e. Pemilihan SDN controller yang tepat untuk sistem Load Balancing
- f. Apakah metode Sistem Load Balancing yang baru dapat menghasilkan perbaikan jika dibandingkan dengan metode yang lama
- g. Pada Jaringan SDN permasalahan *Load Imbalance* masih bisa terjadi. Jika proses pendistribusian beban jaringan tidak rata, akan mempengaruhi performansi dan efisiensi keseluruhan jaringan. Beban trafik yang seimbang dalam jaringan membantu mengoptimalkan pemanfaatan *resource* yang tersedia dengan cara memaksimalkan *throughput*, meminimalkan waktu respon, dan menghindari *overloading* beban pada jaringan.
- h. Berdasarkan penelitian [2], *load balancing* pada SDN dengan menggunakan algoritma LBBSRT tidak mempertimbangkan dari segi konsumsi energi dan *resource* pada server. Sehingga perlu dilakukan penelitian *load balancing* pada SDN dengan mempertimbangkan masalah penggunaan energi di server.

## 1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk

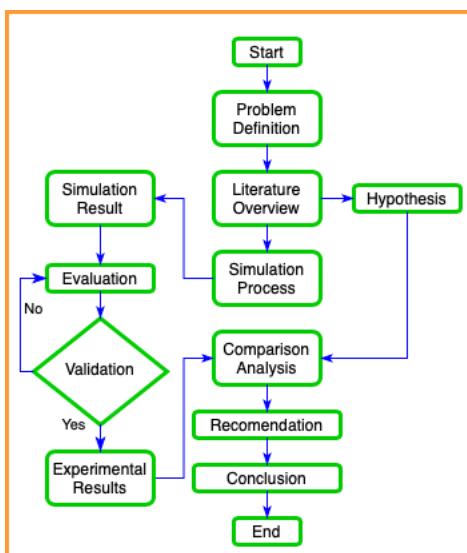
- a) Mendapatkan solusi Metode Server Load Balancing berbasis Arsitektur SDN dengan metode Algoritma Logika Fuzzy
- b) Mendapatkan solusi Metode Server Load Balancing berbasis Arsitektur SDN, dimana Nilai *response time*, *cpu* dan *RAM usage* dari setiap server diolah oleh SDN Kontroler.
- c) Mengimplementasikan Metode Server Load Balancing dengan menggunakan modul Ryu sebagai SDN Kontroler.

- d) Membuktikan ke-efektifan metode Server Load Balancing yang baru dengan cara membandingkan nilai Nilai *response time*, *cpu* dan *RAM usage* dengan metode yang sebelumnya.

### 1.4 Hypothesis

Penelitian ini diharapkan menyelesaikan permasalahan *Load Balancing* pada jaringan SDN dengan menggunakan metode Algoritma Logika Fuzzy, sehingga beban kerja Server dapat seimbang.

### 1.5 Metodologi



Gambar 1. Metodologi Penelitian

Metodologi penelitian dijelaskan pada gambar **Error! Reference source not found.** diatas. Kami menjelaskan Definisi masalah pada bab 1. Tinjauan pustaka dari penelitian sebelumnya tentang karakteristik Jaringan SDN, Teknik LBBSRT (*Load Balancing Based Server Response Time*), Protokol OpenFlow disajikan pada bab 2. Dari literatur dan teori dasar, dengan menggunakan metode Logika Fuzzy dan Nilai *response time*, *cpu* dan *RAM usage* dari setiap server dapat ditentukan Server dengan beban yang kecil dalam proses Load Balancing, dan diharapkan memberikan hasil yang lebih baik dibandingkan jika hanya menggunakan nilai *Server Response Time* saja, sehingga ini menjadi hipotesis. Karena untuk membuktikan hipotesis ini, kami melakukan simulasi dan menetapkan Nilai *response time*, *cpu* dan *RAM usage* sebagai kontrol variabel, kemudian hasil percobaan dibandingkan dengan hipotesis. Rekomendasi kami berdasarkan hasil Analisa dan perbandingan dengan metode. Akhirnya, kesimpulan dari makalah yang sederhana ini akan ada di bab terakhir

### **1.6 Metode Penelitian**

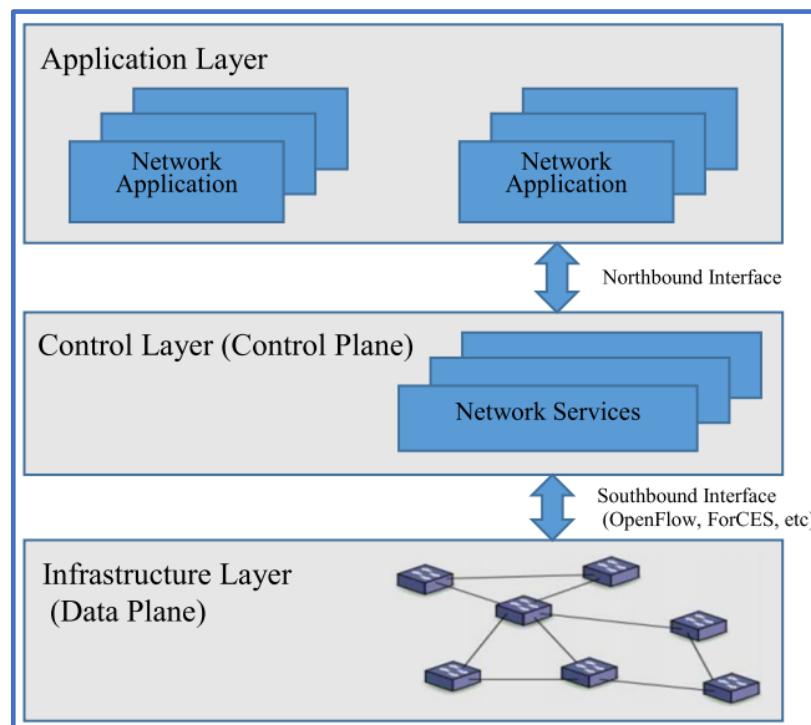
Pada penelitian ini kami mencoba mencari solusi yang lebih baik untuk permasalahan *Server Load Balancing* di Jaringan SDN dengan menggunakan Algoritma Logika Fuzzy, Nilai *response time*, *cpu* dan *RAM usage* digunakan sebagai variable input untuk algoritma tersebut. Hasil percobaan nantinya akan dibandingkan dengan Sistem *Load Balancing* dengan metode LBBSRT, Round Robin dan Random untuk mem-validasi *hipotesis*.

## BAB II

### STUDI PUSTAKA

#### 2.1 Arsitektur Software Define Networks (SDN).

Software Define Networks (SDN) arsitekturenya memisahkan *Data plane* dengan *Control plane* [3]. Pada *Control plane* ini meliputi *network operating system* dan *applications*, sedangkan untuk *Data plane* berisi gabungan standar protocol untuk mendukung perangkat hardware di jaringan. Arsitektur SDN dapat dikategorikan menjadi sistem *three-layer* [4], yaitu aplikasi, control dan data. Dibandingkan dengan network arsitektur jaringan tradisional, structure three-layer pada SDN merupakan sebuah inovasi yang sangat efektif untuk mengatasi masalah kerentanan pada jaringan tradisional [5]. Selain itu SDN mendukung pemrograman yang independent pada *network control system* pada mode management dan dapat diupgrade secara instan oleh *network application interface*. Layer aplikasi pada arsitektur SDN menyediakan user berbagai macam API interface yang dapat digunakan untuk membangun modul sesuai fungsi dan kebutuhan usernya [6].



Gambar 2. arsitektur SDN dengan data plane, control plane dan Application plane

Dari Gambar 2 mempunyai keterangan sebagai berikut :

### 2.1.1 Application Plane

Application Plane: berada pada lapisan teratas, berupa aplikasi yang dapat secara langsung dan eksplisit mendefinisikan Network Requirement dan Network Behavior yang diinginkan. Layer ini berkomunikasi dengan Control Layer melalui NorthBound Interface (NBI).

### 2.1.2 Control Plane

Controller Plane: yaitu entitas kontrol yang memiliki tugas yaitu mentranslasikan Network Requirement yang telah didefinisikan oleh Application Layer menjadi instruksi-instruksi yang sesuai untuk Infrastructure Layer, dan memberikan abstract view yang dibutuhkan bagi Application Layer (abstract view meliputi informasi statistik dan event yang terjadi di jaringan).

### 2.1.3 Data Plane

Data Plane: terdiri dari elemen jaringan yang dapat menerima instruksi dari Control Layer. Interface antara Controller Plane dan Data Plane disebut SouthBound Interface (SBI), atau Control-To-Data-Plane Interface (CDPI). Controller Plane merupakan otak dari jaringan SDN, dapat dijalankan secara terpisah dari Data Plane. Sedangkan Data Plane merupakan perangkat-perangkat keras jaringan yang terprogram secara khusus dan dikendalikan penuh oleh Control Plane. Pada SDN tersedia Open Interface yang memungkinkan sebuah entitas Software atau aplikasi untuk mengendalikan konektivitas dari sumber daya jaringan, mengendalikan aliran trafik, serta melakukan inspeksi atau memodifikasi trafik tersebut.

### 2.1.4 NourthBound Interface

NourthBound Interface bagian yang terdapat di antara Application Plane dan Control Plane yaitu sebuah bagian komponen jaringan digunakan untuk saling berkomunikasi dengan komponen jaringan yang lain dengan level komponen yang lebih tinggi.

### 2.1.5 SouthBound Interface

SouthBound Interface bagian yang terdapat di antara Control Plane dan Data Plane yaitu sebuah bagian komponen jaringan digunakan untuk saling berkomunikasi dengan komponen jaringan yang lain dengan level komponen yang lebih rendah.

### 2.1.6 Kelebihan SDN

Kelebihan jaringan SDN adalah

#### 1) *Directly programmable*

Dapat diprogram secara langsung karena control plane terpisah dari forwarding plane (data plane).

#### 2) *Agile*

Pemisahan antara Control Plane dengan Forwarding Plane menyebabkan administrator jaringan dapat secara dinamis menyesuaikan Flow Traffic Network sesuai kebutuhan organisasi/institusi.

#### 3) *Centrally Managed*

Network Intelligence berada terpusat di SDN Controller, dan mengelola satu gambaran umum (global view) yang utuh mengenai jaringan, sehingga tampak bagi pengguna hanya terdiri atas satu Logical Switch saja.

#### 4) *Programmatically Configured*

SDN memungkinkan Network Administrator untuk mengelola sumberdaya jaringan dengan sangat cepat, melalui program SDN yang terotomatisasi dan dapat ditulis sendiri karena program tersebut tidak bergantung pada Proprietary Software atau Device.

#### 5) *Open standards-based and Vendor-neutral*

Format instruksi Controller plan didefinisikan menggunakan open standard dan tidak tergantung pada Vendor-specific Device atau protokol tertentu, sehingga dapat diimplementasikan pada jaringan apapun tanpa terikat oleh vendor, dan pada akhirnya akan mempermudah inovasi di bidang jaringan

## 2.2 Protokol *OpenFlow*

*Interface protocol OpenFlow* merupakan elemen dasar yang dibutuhkan untuk membangun SDN [7]. *OpenFlow* adalah standar komunikasi pertama yang digunakan antara layer kontrol dengan layer infrastruktur pada arsitektur SDN [8] [9]. *OpenFlow* menggunakan konsep aliran untuk mengidentifikasi *network traffic* berdasarkan 4 aturan yang bisa deprogram secara statis ataupun dinamis oleh *SDN control software* [8]. Setiap paket yang datang, diatur oleh switch dan diupdate di *flow table entry*. Menggunakan entries ada *flow table*, switch akan memforward paket tanpa perlu membangun atau memodifikasi *flow table*. SDN controller akan membuat dan menginstall aturan pada flow table switch secara bersamaan. Arsitekture SDN dengan *OpenFlow-based* memberikan hak control secara terperinci dengan cara membuat

network dapat bereaksi secara real-time pada setiap perubahan yang terjadi baik di aplikasi maupun di *service user* [10]. Teknologi pada SDN OpenFlow-based dapat meningkatkan kemampuan bandwidth secara efektif untuk menangani aplikasi secara dinamis dan secara signifikan dapat mengurangi kerumitan operation dan managemen [11]. Fitur yang kas dan inovasi dari SDN adalah dapat mendukung pengembangan dan pengujian untuk strategi *novel forwarding* dan *network protocol* secara efektif. Ada tiga tipe *message* pada *OpenFlow* yaitu *Controller-to-Switch*, *Asynchronous and Symmetric*. Teknologi *OpenFlow-based* pada SDN dapat meningkatkan kemampuan dan secara efektif menangani aplikasi yang dinamis dan secara signifikan dapat mengurangi *operation* dan *management* yang kompleks [2].

Pertukaran pesan pada komunikasi *OpenFlow* dibagi dengan tiga jenis pesan [12],

- 1) *Controller-to-Switch*

Jenis pesan ini dimulai oleh *controller* dan digunakan untuk mengelola *logical state* dari *switch* seperti konfigurasi, entri tabel aliran, pesan *Packet\_out*.

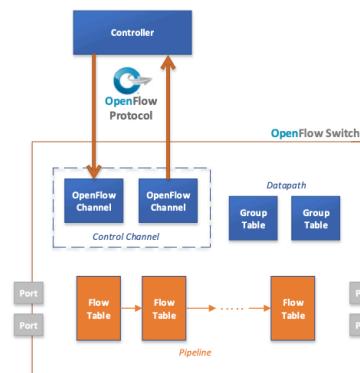
- 2) Asinkron

Jenis pesan ini dikirim ke *controller* tanpa diminta, seperti pesan status, pesan *Packet\_in*. *Packet\_in* ini digunakan ketika tidak ada tabel aliran yang cocok untuk aliran tertentu kemudian mengirimkannya ke controller.

- 3) Simetris

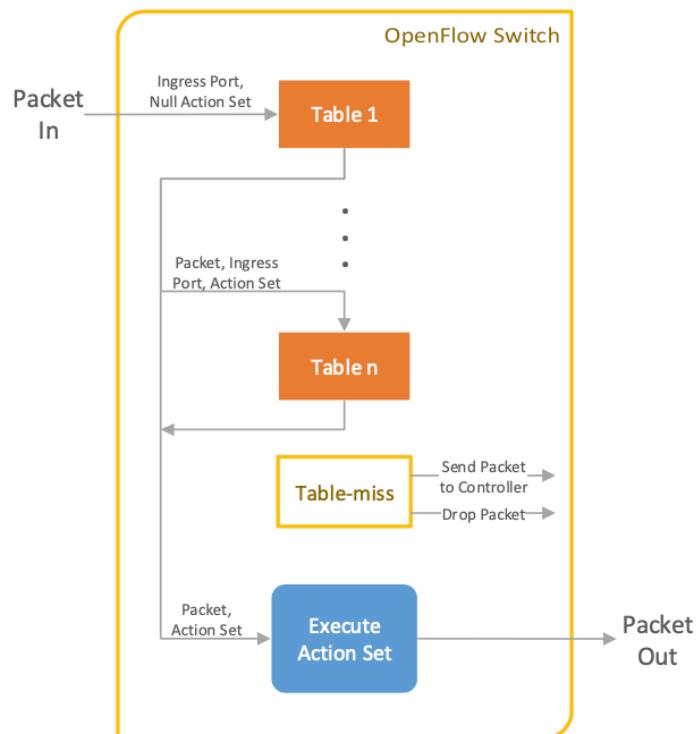
Jenis pesan ini dikirim tanpa diminta dari controller dan switch dan digunakan untuk pembuatan koneksi.

Controller SDN berkomunikasi dengan switch yang mendukung OpenFlow menggunakan protokol OpenFlow. Switch yang mendukung OpenFlow mempunyai satu atau lebih tabel aliran, yang pencarian dan penerusan paket dan juga mungkin menyertakan entri aliran tabel-miss, yang menentukan proses yang tidak cocok dengan tabel aliran.



Gambar 3. Struktur dasar dari lingkungan OpenFlow

Untuk setiap paket masuk, switch membandingkan bidang header terhadap setiap aturan pada setiap tabel aliran yang ada secara berurutan. Keluaran dari setiap tabel alir meliputi set tindakan yang menjadi masukan untuk tabel alur berikutnya. Semua set tindakan akan dieksekusi setelah semua tabel alur diperiksa.



Gambar 4. Tabel aliran pada OpenFlow Switch

### 2.3 SDN Controller

Pengontrol SDN adalah entitas logis yang menerima instruksi atau persyaratan dari lapisan Aplikasi SDN dan menyampaikannya ke komponen jaringan. Pengontrol juga mengekstrak informasi tentang jaringan dari perangkat keras dan berkomunikasi kembali ke Aplikasi SDN dengan tampilan abstrak jaringan, termasuk statistik dan peristiwa tentang apa yang terjadi.

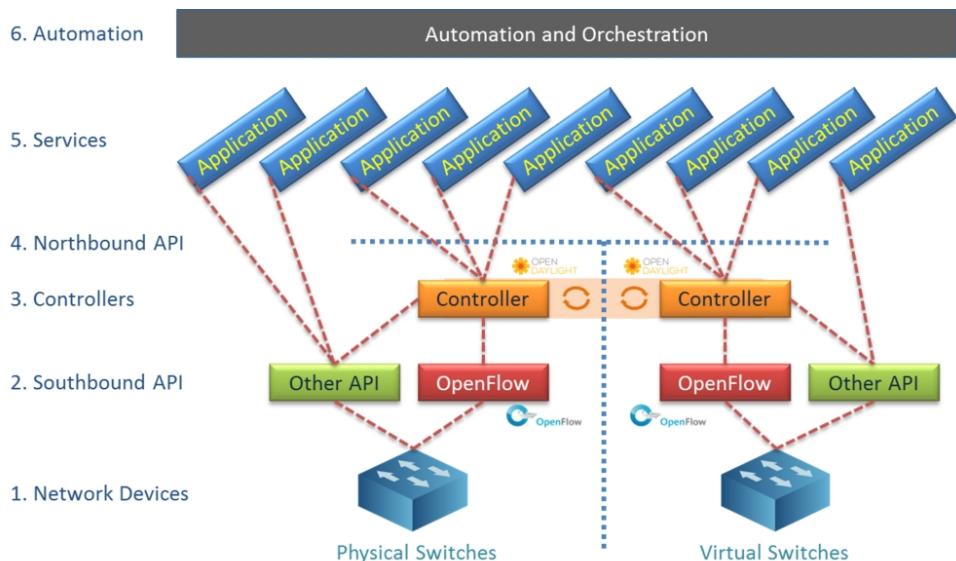
Contoh:

RYU, OpenDayLight, ONOS, FloodLight, dll

#### 2.3.1 SDN Applications

Aplikasi SDN adalah program yang mengkomunikasikan perilaku dan sumber daya yang dibutuhkan dengan Pengontrol SDN melalui antarmuka pemrograman aplikasi (API). Aplikasi ini dapat mencakup manajemen jaringan, analitik, atau aplikasi bisnis yang

digunakan untuk menjalankan pusat data besar. Ini adalah tempat untuk penelitian, inovasi, ide-ide baru dll. *NorthBound API* didefinisikan sebagai koneksi antara pengontrol dan aplikasi.



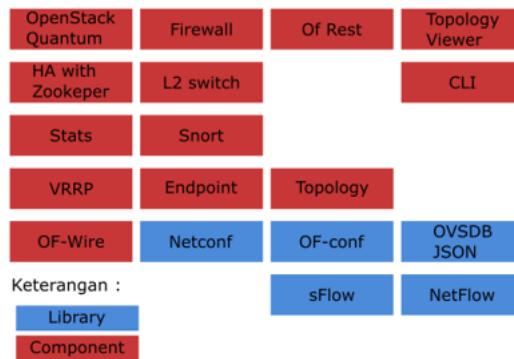
Gambar 5. SDN Applications and NorthBound API

### 2.3.2 RYU

RYU merupakan salah satu *controller* dalam *software defined network* yang dirancang untuk meningkatkan kemampuan dalam jaringan yang bermanfaat untuk mempermudah dan mengatur. Secara umum *controller* merupakan fungsi otak dari SDN. RYU merupakan *open source* yang dikembangkan oleh NTT. Dalam RYU *application program interface* (API) sudah didefinisikan dengan sangat baik yang berarti dapat melakukan pengembangan dengan mudah untuk membuat suatu network management yang baru. RYU merupakan *controller* yang menggunakan bahasa pemrograman python yang mudah dalam pemakaiannya serta memiliki dokumentasi yang banyak sehingga lebih mudah menemukan solusi jika terdapat permasalahan. *Controller* RYU ini mendukung beberapa protocol dalam software defined network diantaranya OpenFlow, Netconf, OF-config serta lainnya.

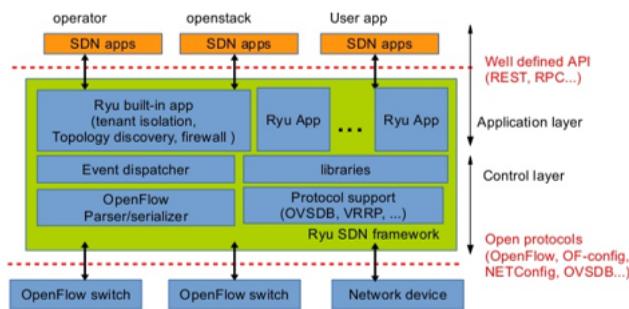
Keunggulan dibanding dengan controller lain diantaranya.

- 1) RYU menyediakan banyak komponen yang berguna untuk aplikasi software defined network.
- 2) Komponen lama dalam RYU dapat dimodifikasi sesuai dengan kebutuhan dan menerapkan pada komponen yang baru.
- 3) Menggabungkan komponen untuk membangun aplikasi.



Gambar 6. Komponen dan Library yang ada di RYU

Framework RYU berada pada Control Layer pada Arsitektur SDN, dan beberapa aplikasi pada RYU berada pada Application Layer guna untuk berkomunikasi dengan Aplikasi SDN lain menggunakan API seperti REST, RPC, dan sebagainya.



Gambar 7. Arsitektur RYU

## 2.4 Mininet

Mininet adalah sebuah emulator untuk membuat Prototype jaringan berskala besar secara cepat pada sumberdaya yang terbatas (seperti pada single komputer atau laptop maupun Virtual Machine). Mininet diciptakan dengan tujuan untuk mendukung riset di bidang SDN dan OpenFlow. Emulator Mininet memungkinkan kita untuk menjalankan sebuah kode secara interaktif di atas laptop atau di atas virtual Hardware, tanpa harus memodifikasi kode tersebut. Artinya kode simulasi sama persis dengan kode pada Real Network Environment.

## 2.5 Fuzzy Logic

[13] Logika Fuzzy pertama kali dikembangkan oleh Lotfi A. Zadeh, seorang ilmuwan Amerika Serikat berkebangsaan Iran dari Universitas California di Barkeley, melalui tulisannya pada tahun 1965.

Menurut Prof. Lotfi A. Zadeh: “*Pada hampir semua kasus kita dapat menghasilkan suatu produk tanpa menggunakan logika Fuzzy, namun menggunakan Fuzzy akan lebih cepat dan murah.*”

Dalam paper yang dibuat oleh Lofti A Zadeh, Zadeh memperkenalkan teori yang memiliki objek-objek dari himpunan *Fuzzy* yang memiliki batasan yang tidak presisi dan keanggotaan dalam himpunan *Fuzzy*, dan bukan dalam bentuk logika benar (*true*) atau salah (*false*), tapi dinyatakan dalam derajat (*degree*). Konsep seperti ini disebut dengan *Fuzziness* dan teorinya dinamakan *Fuzzy Set Theory*.

*Fuzziness* dapat didefinisikan sebagai logika kabur berkenaan dengan semantik dari suatu kejadian, fenomena atau pernyataan itu sendiri. Seringkali ditemui dalam pernyataan yang dibuat oleh seseorang, evaluasi dan suatu pengambilan keputusan. Logika fuzzy adalah suatu cara yang tepat untuk memetakan suatu ruang input ke dalam suatu ruang output.

### 2.5.1 Kelebihan Logika Fuzzy

Berikut beberapa alasan mengapa menggunakan Fuzzy:

1. Konsep logika fuzzy mudah dimengerti. Karena logika fuzzy menggunakan dasar teori himpunan, maka konsep matematis yang mendasari penalaran fuzzy tersebut cukup mudah untuk dimengerti.
2. Logika fuzzy sangat fleksibel, artinya mampu beradaptasi dengan perubahan-perubahan, dan ketidakpastian yang menyertai permasalahan.
3. Logika fuzzy memiliki toleransi terhadap data yang tidak tepat. Jika diberikan sekelompok data yang cukup homogen, dan kemudian ada beberapa data yang “ekslusif”, maka logika fuzzy memiliki kemampuan untuk menangani data ekslusif tersebut.
4. Logika fuzzy mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.
5. Logika fuzzy dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan. Dalam hal ini, sering dikenal dengan nama Fuzzy Expert System menjadi bagian terpenting.
6. Logika fuzzy dapat bekerjasama dengan teknik-teknik kendali secara konvensional. Hal ini umumnya terjadi pada aplikasi di bidang teknik mesin maupun teknik elektro.
7. Logika Fuzzy umumnya diterapkan pada masalah-masalah yang mengandung **unsur ketidakpastian (*uncertainty*)**.

### 2.5.2 Himpunan Fuzzy

Himpunan Fuzzy dapat digunakan untuk mengantisipasi kelemahan dari himpunan crisp. Seseorang dapat masuk dalam dua himpunan yang berbeda, misal MUDA dan

## PROPOSAL THESIS

PAROBAYA, PAROBAYA dan TUA, dsb. Seberapa besar eksistensi dalam himpunan tersebut dapat dilihat pada derajat keanggotaannya.

**Definisi Himpunan Fuzzy adalah** Diberikan  $X$  himpunan semesta. Himpunan bagian Fuzzy  $A$  dari  $X$  adalah himpunan bagian dari  $X$  yang keanggotaannya didefinisikan melalui fungsi keanggotaan (membership function) sebagai berikut:

$$\mu_A : X \rightarrow [0,1]$$

yang menghubungkan setiap  $x \in X$  ke bilangan real  $\mu_A(x)$  di dalam interval  $[0,1]$  dengan nilai  $\mu_A(x)$  di  $x$  menunjukkan derajat keanggotaan  $x$  dalam  $A$ . Himpunan Fuzzy  $A$  ditulis sebagai berikut:

$$A = \{(x, \mu_A(x)) \mid x \in X\},$$

dengan  $(x, \mu_A(x))$  menyatakan elemen  $x$  mempunyai derajat keanggotaan  $\mu_A(x)$ .

**Derajat Keanggotaan.** Arti derajat keanggotaan adalah sebagai berikut:

- a. Jika  $\mu_A(x) = 1$ , maka  $x$  adalah anggota penuh dari himpunan  $A$ .
- b. Jika  $\mu_A(x) = 0$ , maka  $x$  bukan anggota himpunan  $A$
- c. Jika  $\mu_A(x) = \mu$  dengan  $0 < \mu < 1$ , maka  $x$  adalah anggota himpunan  $A$  dengan derajat keanggotaan sebesar  $\mu$ .

Beberapa hal yang perlu diperlukan diperhatikan mengenai derajat keanggotaan:

- a. Pada himpunan Fuzzy derajat keanggotaan terletak pada rentang 0 sampai 1.
- b. Persamaan dengan probabilitas:  
Keduanya memiliki nilai pada interval  $[0,1]$ , namun interpretasi nilainya sangat berbeda antara kedua kasus tersebut.
- c. Perbedaan dengan probabilitas:
  - 1) Derajat keanggotaan Fuzzy memberikan suatu ukuran terhadap pendapat atau keputusan, sedangkan probabilitas mengindikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang.
  - 2) Misalnya, jika derajat keanggotaan suatu himpunan MUDA adalah 0.9, maka tidak perlu dipermasalahkan berapa seringnya nilai itu diulang secara individual untuk mengharapkan suatu hasil yang hampir pasti muda. Di lain pihak, nilai probabilitas 0.9 muda berarti 10% dari himpunan tersebut diharapkan tidak muda.

### 2.5.3 Fungsi Keanggotaan

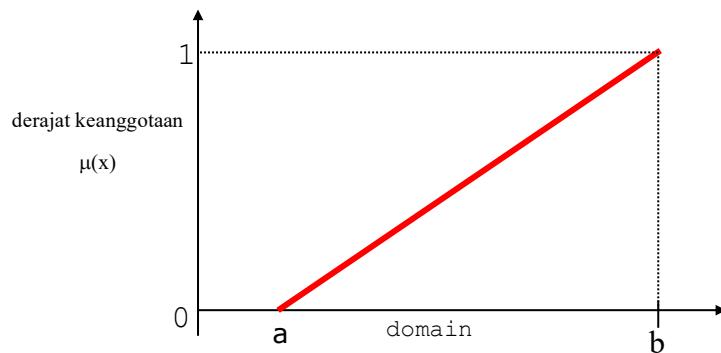
Fungsi Keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam derajat keanggotanya (sering juga disebut dengan nilai keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan derajat keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan, yaitu :

### 2.5.4 Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan fuzzy yang linear, yaitu :

#### a. Representasi Linear Naik

Kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi.



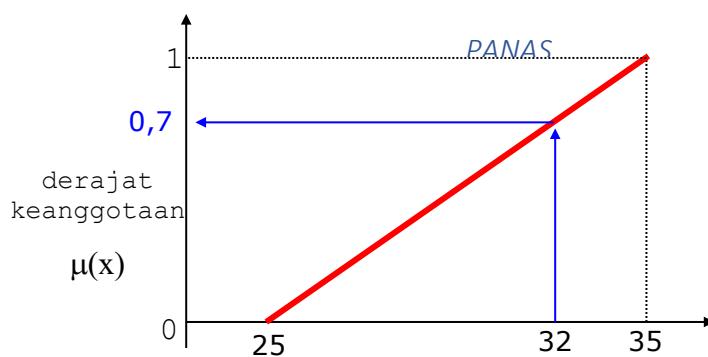
Gambar 8. Representasi Linear Naik.

#### Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} 0; & x \leq a \\ (x-a)/(b-a); & a \leq x \leq b \\ 1; & x \geq b \end{cases}$$

Fungsi keanggotaan untuk himpunan PANAS pada variabel temperatur ruangan seperti terlihat pada Gambar 2.2.

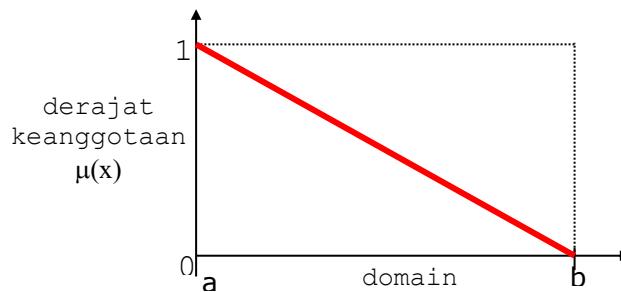
$$\begin{aligned} \mu_{\text{PANAS}}(32) &= (32-25)/(35-25) \\ &= 7/10 = 0,7 \end{aligned}$$



Gambar 9. Fungsi keanggotaan himpunan fuzzy PANAS

### b. Representasi Linear Turun

Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah.



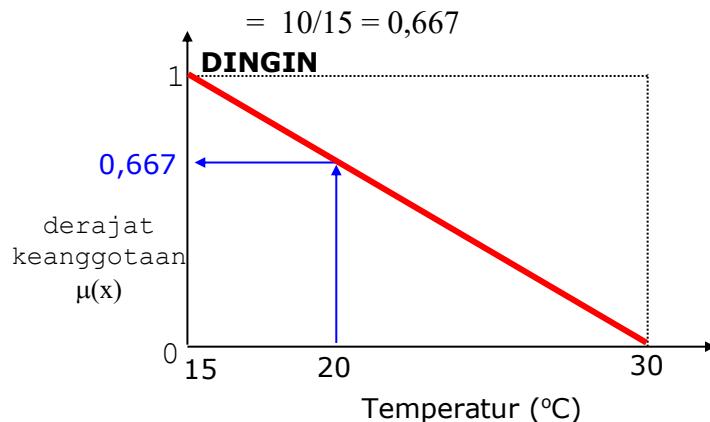
Gambar 10. Representasi Linear Turun

#### Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} (b-x)/(b-a); & a \leq x \leq b \\ 0; & x \geq b \end{cases}$$

Fungsi keanggotaan untuk himpunan DINGIN pada variabel temperatur ruangan seperti terlihat pada Gambar 2.4.

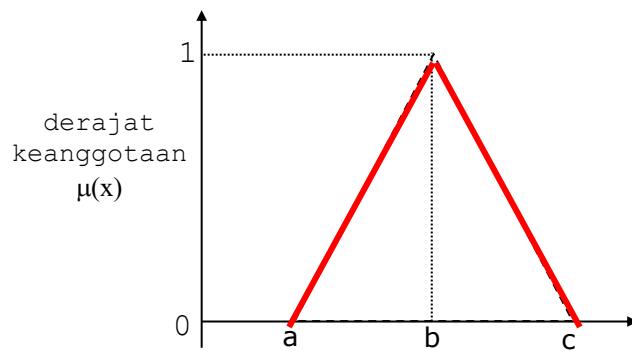
$$\mu_{\text{DINGIN}}(20) = (30-20)/(30-15)$$



Gambar 11. Fungsi keanggotaan himpunan Fuzzy DINGIN

## 2. Representasi Kurva Segitiga

Kurva Segitiga pada dasarnya merupakan gabungan antara 2 garis (linear).



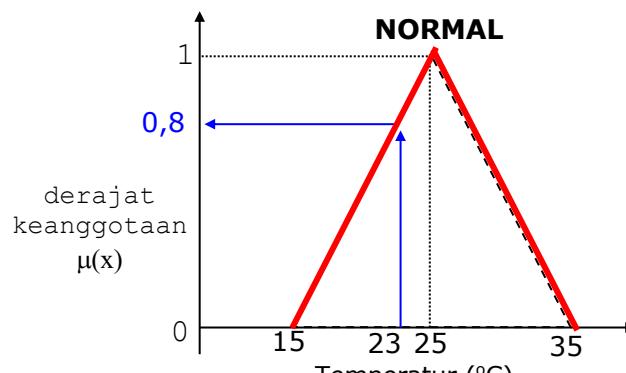
Gambar 12. Kurva Segitiga

### Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} 0; & x \leq a \text{ atau } x \geq c \\ (x - a)/(b - a); & a \leq x \leq b \\ (c - x)/(c - b); & b \leq x \leq c \end{cases}$$

Fungsi keanggotaan untuk himpunan NORMAL pada variabel temperatur ruangan seperti terlihat pada Gambar 2.6.

$$\begin{aligned} \mu_{\text{NORMAL}}(23) &= (23-15)/(25-15) \\ &= 8/10 = 0,8 \end{aligned}$$



Gambar 13. Fungsi keanggotaan himpunan Fuzzy NORMAL.

## 2.6 Teknik Load Balancing

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. 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 ditambahkan 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 daftar *Real server*, dan jika *Real server* tersebut kembali aktif maka akan dimasukkan ke dalam daftar *Real server*. *Load balancer* juga dapat dikonfigurasi redundant dengan *Load Balancer* yang lain.

Beberapa komponen yang ada dalam arsitektur *Load Balancer*

- 1) *Client*, merupakan pengguna yang menggunakan *service* pada *server*
- 2) *Load Balancer*, Perangkat yang bertugas membagi distribusi beban trafik
- 3) *Server*, bertugas untuk menyediakan layanan terhadap *user* yaitu sebuah *web server*

#### 2.6.1 Jenis Teknik *Load Balancing*

Secara umum, algoritma-algoritma pembagian beban yang banyak digunakan saat ini adalah:

- a. Round Robin. Algoritma Round Robin merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat load balancing. Algoritma ini membagi beban secara bergiliran dan berurutan dari satu server ke server lain sehingga membentuk putaran.
- b. Ratio. Ratio (rasio) sebenarnya merupakan sebuah parameter yang diberikan untuk masing-masing server yang akan dimasukkan kedalam sistem load balancing. Dari parameter Ratio ini, akan dilakukan pembagian beban terhadap server-server yang

diberi rasio. Server dengan rasio terbesar diberi beban besar, begitu juga dengan server dengan rasio kecil akan lebih sedikit diberi beban.

- c. Fastest. Algoritma yang satu ini melakukan pembagian beban dengan mengutamakan server-server yang memiliki respon yang paling cepat. Server di dalam jaringan yang memiliki respon paling cepat merupakan server yang akan mengambil beban pada saat permintaan masuk.
- d. Least Connection. Algoritma Least connection akan melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah server. Server dengan pelayanan koneksi yang paling sedikit akan diberikan beban yang berikutnya akan masuk.

## 2.7 Teknik LBBSRT – Load Balancing Based Server Response Time

Merupakan Teknik *Load Balancing* yang menggunakan data *Server Response Time*, sebagai inputan dimana Teknik ini membagi beban server dengan cara mengarahkan beban *traffic* ke server dengan *Response Time* paling kecil. Ada 2 proses dalam LBBSRT yaitu menghitung *Server Response Time* dan Pemrosesan *User Request*.

### 2.7.1 Pengukuran *Real-time* untuk mendapatkan *server response time* [2]

Langkah – Langkah :

**Langkah 1:** Kirim pesan Packet\_out ke switch. Setelah sistem dimulai, SDN Kontroler mengirimkan beberapa pesan Packet\_out ke switch dengan interval waktu t dan mencatat waktu transmisi. Jumlah pesan yang dikirim sama dengan jumlah server di kumpulan server. Setiap pesan Packet\_out membawa paket data yang alamat sumbernya adalah alamat IP SDN Kontroler, sedangkan alamat tujuan ditetapkan dengan setiap IP server.

**Langkah 2:** Switch menangani pesan Packet\_out. Ketika switch OpenFlow menerima pesan Packet\_out yang dikirim oleh SDN Kontroler, switch akan mengurai paket data dan mengirimkan paket data ini ke setiap server.

**Langkah 3:** Server mengirim pesan balasan, dari mana SDN Kontroler memperoleh waktu respons server. Setelah menerima paket data yang dikirim dari SDN Kontroler, setiap server menjalankan simulasi permintaan klien dan kemudian mengirimkan paket data dengan alamat sumber ditetapkan sebagai IP server dan alamat tujuan ditetapkan sebagai IP SDN Kontroler. Karena ini adalah acara baru di tabel alur, switch perlu mengirim pesan Packet\_in ke SDN Kontroler. Sekarang

SDN Kontroler mendapatkan waktu kedatangan setiap paket data server dengan mengurai pesan Packet\_in. Hasilnya, SDN Kontroler memperoleh waktu respons dari setiap server, dan memperbarui database yang sesuai.

Setelah waktu respons server diperoleh oleh SDN Kontroler lalu algoritma Load Balancing dijalankan berdasarkan *user request*.

### 2.7.2 Pemrosesan *User Request* [2]

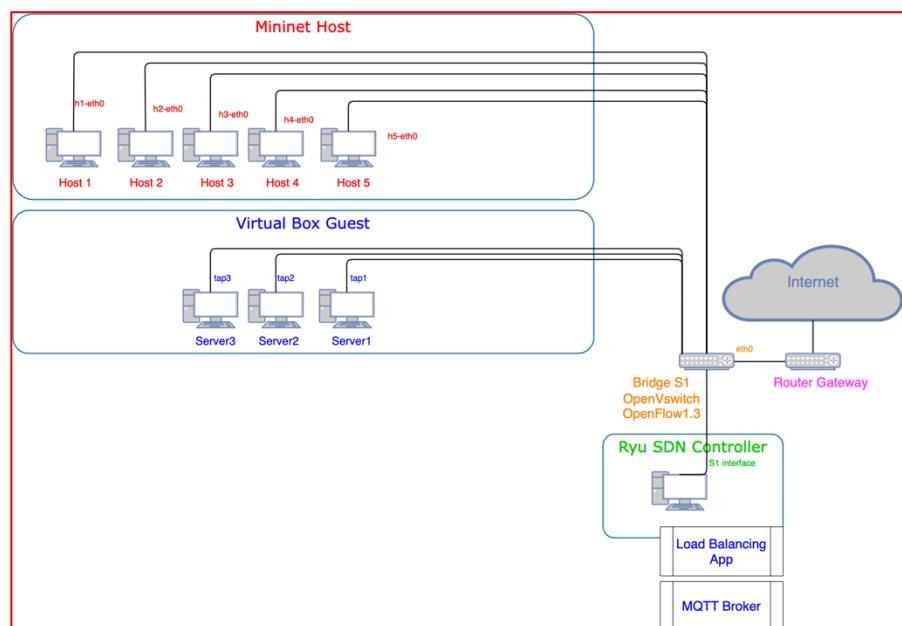
Langkah – Langkah :

**Langkah 1:** SDN Kontroler menangani pesan ARP. Pengguna akan mengirim pesan ARP\_broadcast ke switch pada akses pertama karena switch tidak berisi tabel aliran untuk memproses ARP dan mengirim pesan Packet\_in ke SDN Kontroler. Kontroler kemudian akan membangun alamat MAC virtual, berdasarkan itu, ia mengirim pesan Packet\_out ke switch, dan switch mengirimkan paket balasan ARP ke terminal pengguna.

**Langkah 2:** SDN Kontroler menangani permintaan pengguna. Setelah pengguna menerima paket balasan ARP, mereka memulai permintaan layanan ke server, dan proses permintaan serupa dengan Langkah1. SDN Kontroler juga akan menerima paket layanan permintaan pengguna tersebut, dan memilih server dengan waktu respons minimum atau stabil sesuai dengan data server yang diperoleh.

**BAB III****DESAIN SISTEM DAN SKENARIO SIMULASI****3.1. Model Sistem****3.1.1. Perancangan Jaringan SDN dengan Mininet**

Permasalahan yang ingin diatasi adalah belum dimaksimalkan Penggunaan Data *Resource Utilization* dari Server sebagai input sistem dalam *Load Balancing* mengatasi Beban Server yang tidak seimbang. Topologi Jaringan SDN berikut dibuat untuk mensimulasikan Teknik Load Balancing



Gambar 14. Topologi Jaringan SDN Dengan Sistem Load Balancing

Sistem ini dibagi menjadi beberapa devices, yaitu :

- 1) *Client*, merupakan pengguna/client yang menggunakan layanan yang terdapat pada masing-masing *server*.
- 2) *OpenFlow VSwitch*, perangkat ini bertujuan untuk meneruskan paket dari *controller*.
- 3) *Controller*, perangkat ini bertugas untuk mengatur *OpenFlow Switch* yang di berikan jaringan.
- 4) *Switch*, perangkat ini menghubungkan antara *Client* dengan *OpenFlow Switch*
- 5) *Server*, bertugas untuk menyediakan layanan terhadap *user* yaitu sebuah web *server*. terdapat 3 buah *server* yang merupakan replika.

### 3.1.2. Asumsi dan Batasan

- a) Proses Load Balancing di jaringan SDN.
- b) Proses Load Balancing menggunakan Mininet Simulator , OpenVSwitch dan Ryu SDN Controller.
- c) Metode Load Balancing yang digunakan adalah Logika Fuzzy sedangkan benchmark memakai metode LBBSRT, Round Robin, Random.

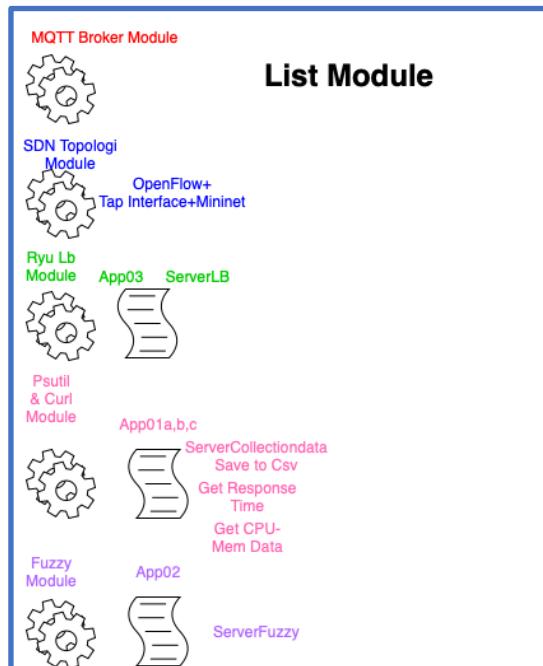
### 3.1.3. Desain Sistem Load Balancing

Sistem Server Load Balancing ini menggunakan inputan dari nilai *Response Time*, *CPU* dan *Memory Utilization* dari *Server*. Setiap Server mempunyai satu nilai *Load Window*. Nilai *Load window* tersebut digunakan untuk membatasi beban trafik yang akan dikirimkan ke server. Dengan menggunakan data tadi sebagai inputan yang dikirim dari Server ke SDN Kontroller maka algoritma Fuzzy akan menentukan nilai Load Window dari sebuah Server. Jika nilai Load Window semakin besar maka server tersebut semakin berkurang beban kerjanya. Lalu metode Load Balancing akan memilih server dengan beban kerja paling kecil untuk memproses *user request*.

Kontroler fuzzy menggunakan nilai cpu  $\hat{c}_{\tau+1}^k$  dan memory utilization  $\hat{m}_{\tau+1}^k$  serta server response time sebagai input dan nilai laju perubahan Load Window  $\omega_{\tau+1}^k$  sebagai output. Sehingga nilai Load Window dihitung dengan rumus

$$W_{\tau+1}^k = W_{\tau}^k + (\omega_{\tau+1}^k \times W_{\tau}^k)$$

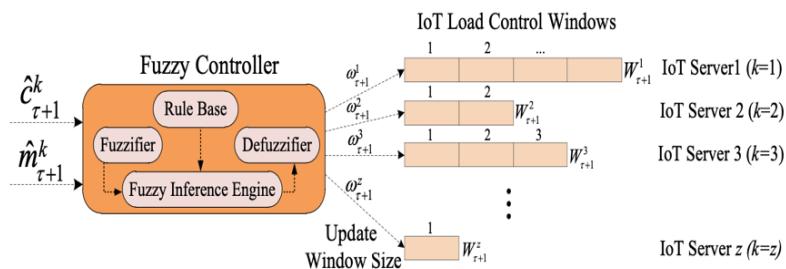
Modul aplikasi yang dipakai adalah sebagai berikut



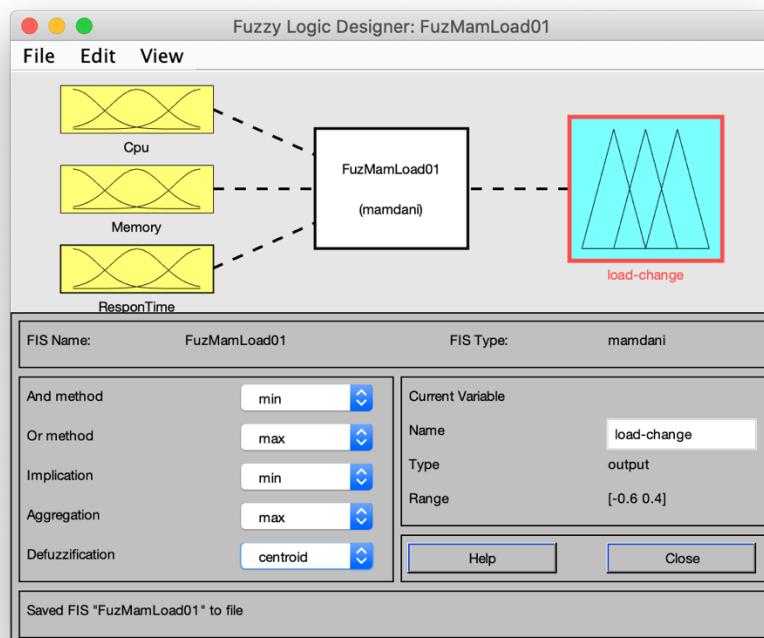
Gambar 15. List Modul dan Script yang digunakan

### 3.1.4. Desain Metode Fuzzy Logic

**Fuzzy inference method** yang dipakai adalah Mamdani methode, sedangkan metode Center of Gravity (CoG) dipakai untuk defuzzification. Penentuan nilai membership Function menggunakan *Gaussian Membership Function* dan *Triangle Membership Function*

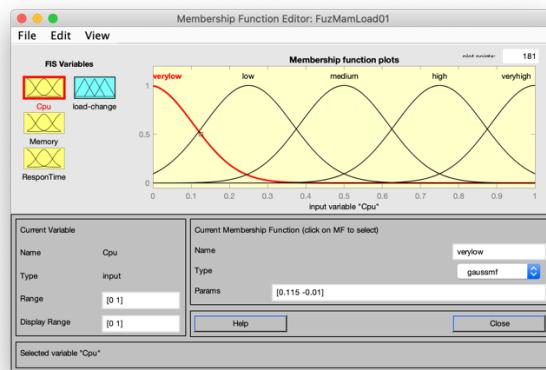


Gambar 16. Diagram Sistem Fuzzy Logic

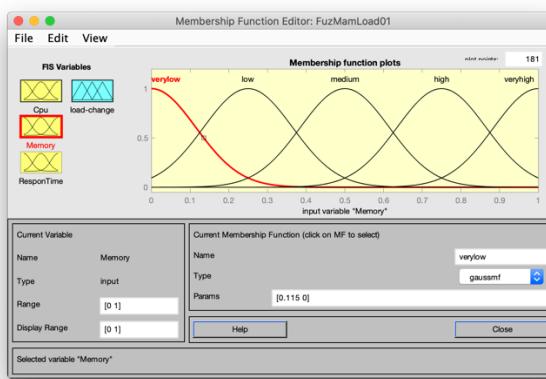


Gambar 17. Tampilan Fuzzy Logic Designer di Matlab

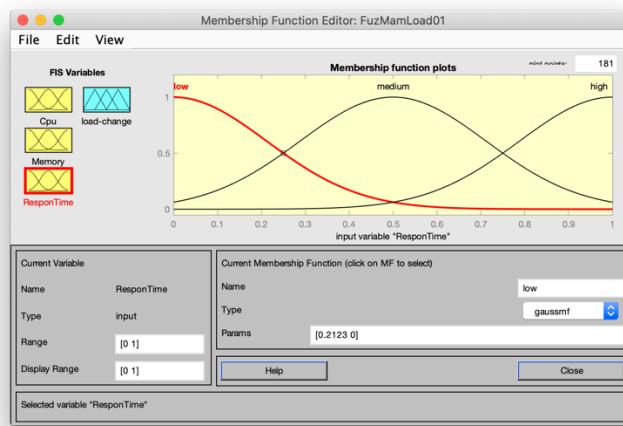
# PROPOSAL THESIS



Gambar 18. Penentuan membership Function – input CPU Usage di Matlab

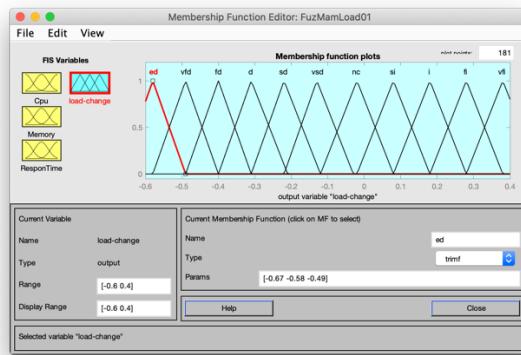


Gambar 19. Penentuan membership Function – input Memory Usage di Matlab

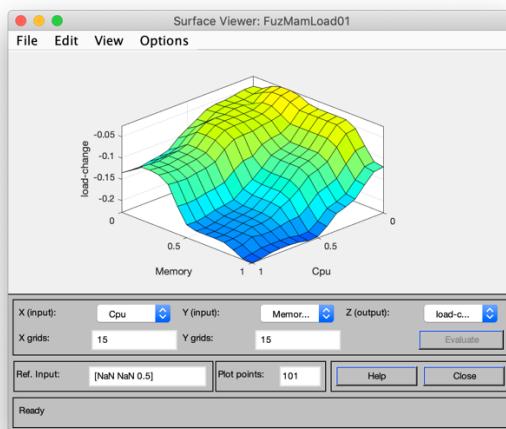


Gambar 20. Penentuan membership Function – input Respon Time Usage di Matlab

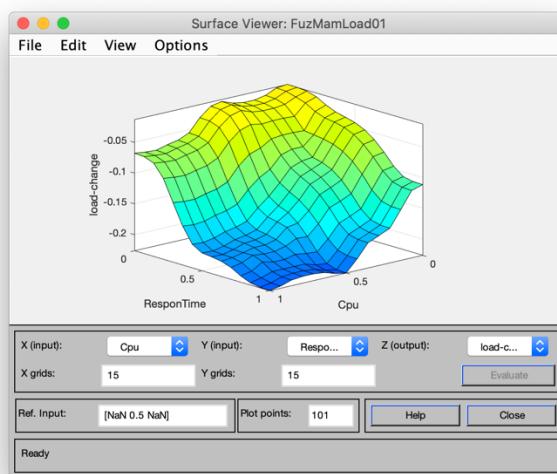
# PROPOSAL THESIS



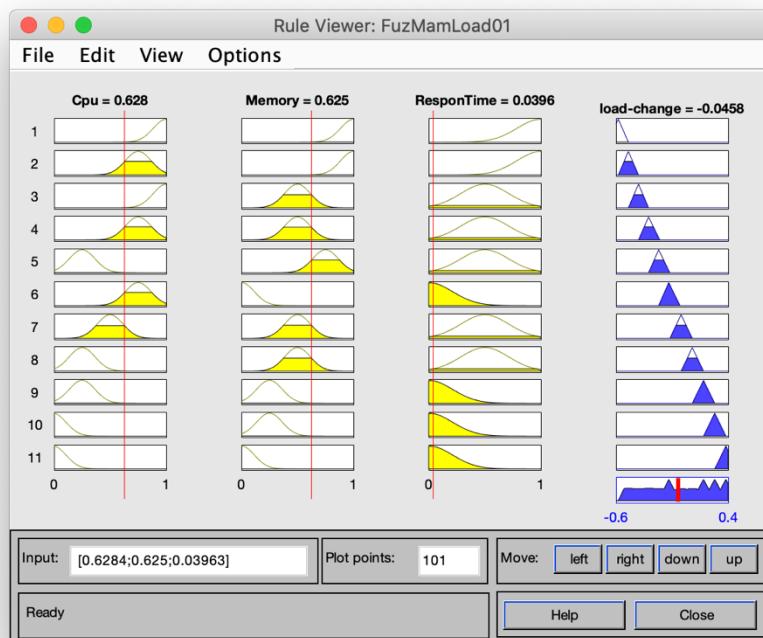
Gambar 21. Penentuan membership Function – output Load Change di Matlab



Gambar 22. Tampilan Surface Viewer CPU-Memory Usage vs Load Change



Gambar 23. Tampilan Surface Viewer CPU Usage- Response Time vs Load Change



Gambar 24. Fuzzy Rule Viewer di Matlab [14]

**Fuzzy Rule Base**. Rule dibuat untuk menunjukkan bahwa sumber daya yang lebih besar menyebabkan beban kerja yang lebih tinggi; Oleh karena itu, ukuran jendela harus lebih kecil [14].

Table 1. Fuzzy rule base

S	Mem	RspTmStd	RspTmMean	WindowChange
verylow	verylow	low	low	<b>very fast increase (vfi)</b>
verylow	low	low	low	<b>fast increase (fi)</b>
verylow	medium	medium	low	<b>increase (i)</b>
verylow	high	medium	low	<b>very slow decrease (vsd)</b>
verylow	veryhigh	high	low	<b>slow decrease (sd)</b>
low	verylow	low	medium	<b>fast increase (fi)</b>
low	low	low	medium	<b>increase (i)</b>
low	medium	medium	medium	<b>slow increase (si)</b>
low	high	medium	medium	<b>slow decrease (sd)</b>
low	veryhigh	high	medium	<b>decrease (d)</b>
medium	verylow	low	medium	<b>increase (i)</b>
medium	medium	medium	medium	<b>slow increase (si)</b>
medium	high	medium	medium	<b>no change (nc)</b>
medium	veryhigh	high	medium	<b>decrease (d)</b>
high	verylow	low	high	<b>fast decrease (fd)</b>
high	low	low	high	<b>very slow decrease (vsd)</b>
high	medium	medium	high	<b>slow decrease (sd)</b>
high	high	medium	high	<b>decrease (d)</b>
high	veryhigh	high	high	<b>fast decrease (fd)</b>
veryhigh	verylow	low	high	<b>very fast decrease (vfd)</b>
veryhigh	low	low	high	<b>slow decrease (sd)</b>
veryhigh	medium	medium	high	<b>fast decrease (fd)</b>
veryhigh	high	medium	high	<b>very fast decrease (vfd)</b>
veryhigh	veryhigh	high	high	<b>extremely decrease (ed)</b>

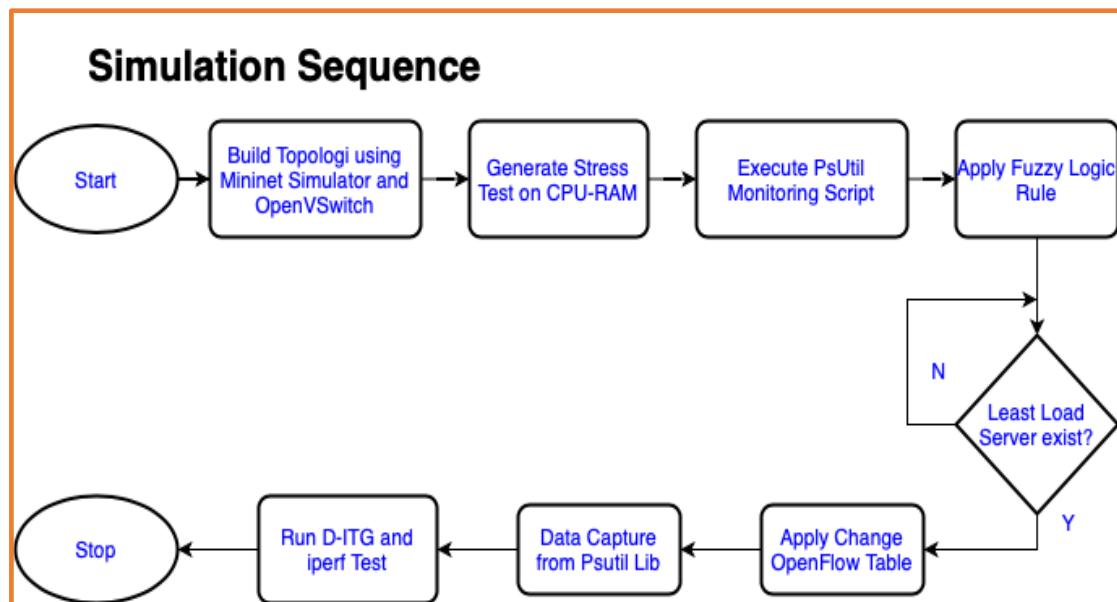
Dengan menggunakan Metode Fuzzy maka akan diharapkan bisa mengatur besar Load Window . Server dengan nilai Load Window terbesar dipilih dan diberi beban

beban trafik tambahan karena dianggap sebagai server yang sedang tidak sibuk. Hal tersebut dilakukan agar kondisi Server Load Balancing tercapai dan beban trafik yang diarahkan ke server – server menjadi seimbang.

### 3.2. Skenario Simulasi

Simulasi Teknik Load Balancing dijalankan dengan menggunakan Protokol OpenFlow, 1 buah Ryu SDN Controller, 1 buah OpenvSwitch, beberapa server dan client.

Berikut ini Flowchart dari Simulasi Sistem Load Balancing. Pada penelitian ini SDN Controller dipakai untuk mengumpulkan data *Server Resource* dan *response time* lalu pada akhirnya kontroler SDN akan membuat dan meng-*install flow rules* pada OpenvSwitch menggunakan Protokol OpenFlow.



Gambar 25. Skema Simulasi

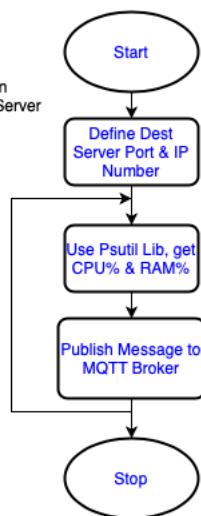
### 3.3. Skenario Pengambilan Data

#### 3.3.1. Pengambilan Data

Pengambilan data dilakukan untuk mendapatkan nilai *server response time*, *cpu* dan *memory utilization* dari masing – masing server.

**Flowchart01**

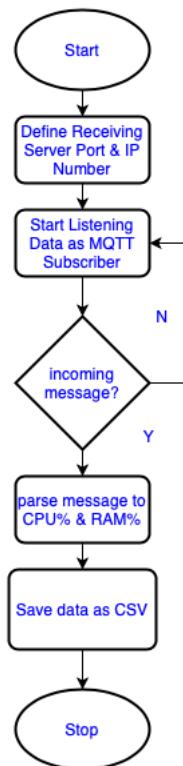
Pengambilan Data Cpu dan Memory dengan PsUtil di Server



Gambar 26. Flowchart Pengambilan Data Cpu-RAM dg Library Psutil

**Flowchart02**

Penerimaan Data Cpu dan Memory dengan PsUtil di Server



Gambar 27. Flowchart Penerimaan data Cpu-RAM yang dikirim berupa paket UDP ke Server Load-Balancing

Berikut pseudo code Python untuk mengukur CPU dan *Memory Usage*

```

#!/usr/bin/env python3
import paho.mqtt.client as mqtt
import multitimer
import time
from psutil import cpu_percent, cpu_count, virtual_memory

#MQTT connection
broker_url = "localhost"
broker_port = 1883
client = mqtt.Client()
# edit code for passwords
print("setting password")
  
```

```

client.username_pw_set(username="user01",password="mqtt")
client.connect(broker_url, broker_port)
time.sleep( 5 )

def job():

# message you send to server
msg = str(cpu_percent()) + ';' + str(virtual_memory().percent)
print("Publish Cpu/Ram data", msg)
client.publish(topic="sdn/cpumem01", payload=msg, qos=1, retain=False)
client.publish(topic="sdn/cpumem01", payload=msg, qos=0, retain=False)

# This timer will run job() five times, one second apart
timer = multitimer.MultiTimer(interval=1, function=job, count=-1)
# Also, this timer would run indefinitely...
timer.start()

```

### 3.3.2. Validasi Sistem

a) **Pengujian ke-1,**

Dalam percobaan ini, kami menggunakan 10 klien untuk mengakses ke server. Jadi kami menyiapkan dua frekuensi akses yang berbeda, yang pertama adalah setiap klien mengirimkan permintaan HTTP terus-menerus ke server, dan yang lainnya adalah setiap klien mengirimkan permintaan setiap dua detik. SimpleHTTPServer digunakan untuk membangun aplikasi web di tiga server. Kami menggunakan 10 klien untuk mengakses server dalam tiga situasi berbeda: (1) 4 klien mengirim permintaan layanan ke server secara terus menerus, sementara 6 klien mengirim permintaan secara terputus-putus; (2) 5 klien mengirim permintaan layanan ke server secara terus menerus, sedangkan 5 klien mengirim permintaan secara terputus-putus; (3) 6 klien mengirim permintaan layanan ke server secara terus menerus, sedangkan 4 klien mengirim permintaan secara terputus-putus. Di sisi SDN controller, kami menambahkan dua modul, satu digunakan untuk mengukur waktu respons server, dan yang lainnya digunakan untuk memproses *user request* dan mengatur *flow table*

b) **Pengujian ke-2,**

Pengujian ini bertujuan mendapatkan grafik perubahan nilai *Throughput*, *average delay*, dan *Server Response Time* untuk masing – masing algoritma dalam kondisi diberi traffic dari salah satu client dengan jumlah traffic meningkat dari 50 – 650 mps. Host / client masing – masing menjalankan proses *inject traffic* dengan nilai 50 – 650 mps (*message per second*) menuju ke server. Lalu akan dilihat sampai batas mana server bisa menangani beban traffic tanpa menggunakan metode Load Balancing setelah itu beban trafik dinaikkan lagi (*overloading*) sampai 1000 mps. Dengan menggunakan metode Load Balancing

yang berbeda maka akan didapatkan pergerakan nilai *Throughput*, *average delay*, dan *Server Response Time* untuk masing – masing algoritma.

### c) Pengujian ke-3

Pengujian ini bertujuan mendapatkan grafik perubahan CPU Utilization dan Memory Utilization dengan menggunakan bermacam algoritma. Mekanismenya sama dengan pengujian ke-1

### 3.4. Skenario Analisis

Analisa grafik untuk percobaan 1 nantinya mengilustrasikan waktu respons server untuk jangka waktu setelah akses bersamaan mencapai nilai maksimum dalam tiga situasi masing-masing. Kami memilih waktu respons rata-rata server sebagai waktu respons keseluruhan. Tujuan utama dari load balancing adalah untuk menghindari penyimpangan beban sistem yang signifikan dalam waktu yang lama sehingga dapat meningkatkan efisiensi sistem dan mencapai pengalaman pengguna yang lebih baik. Jelas, efek dari load balancing tergantung pada beban dan waktu respon server. Jadi kami memilih dua parameter ini untuk dibandingkan dengan skema lain. Dalam penelitian ini, pertama-tama kami mengevaluasi efisiensi skema Load Balancing dengan metode Fuzzy Logic terhadap skema round-robin, skema *random* dan skema LBBSRT.

Analisa Grafik untuk percobaan untuk percobaan 3 nantinya mengilustrasikan CPU dan *Memory Usage* untuk setiap server ketika diberi beban sistem. Lalu kami akan membandingkan hasil yg didapat untuk setiap algoritma

Hasil Sistem *Server Load Balancing* akan dianalisa nilai performansi seperti *Throughput*, *average delay* dan nilai *resource* setiap server seperti *Server Response Time*, *CPU Utilization* dan *Memory Utilization*, lalu divalidasi sesuai hipotesis awal dengan cara membandingkan hasil yang didapat dengan hasil metode sebelumnya yaitu *Load Balancing* menggunakan metode LBBSRT, Random dan Round Robin.

## Bibliography

- [1] e. Lin Li, "Load balancing researches in SDN: A survey.," *IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*., vol. 7, 2017.
- [2] Y. F. J. C. H. Zhong, "LBBSRT: An efficient SDN load balancing scheme based on server response time," *Future Generation Computer Systems*, 2017.
- [3] S. M. K, N. K. H and K. H. J, "Software-defined networking (SDN): A reference architecture and open APIs," in *ICT Convergence (ICTC) IEEE*, 2012.
- [4] G. D. D. L. e. a. alishevskiy A, "OpenFlow-Based Network Management with Visualization of Managed Elements," in *IEEE Research and Educational Experiment Workshop (GREE)*, 2014.
- [5] G. J. C. K. L. Huang S, "Network Hypervisors: Enhancing SDN Infrastructure," in *Computer Communications*, 2014.
- [6] K. A. R. B. Lara A, "Network innovation using openflow: A survey," in *Communications Surveys & Tutorials, IEEE*, 2014.
- [7] A. T. B. H. e. a. McKeown N, "OpenFlow: enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*, 2008.
- [8] S. N. U. S. e. a. Rotsos C, "OFLOPS: An open framework for OpenFlow switch evaluation," in *Passive and Active Measurement. Springer*, Berlin, 2012.
- [9] S. S. P. G. e. a. Kobayashi M, "Maturing of OpenFlow and Software-defined Networking through deployments," in *Computer Networks*, 2014.
- [10] Z. T. X. H. Yin H, "Defining Data Flow Paths in Software-Defined Networks with Application-Layer Traffic Optimization," in *U.S. Patent Application*, 2013.
- [11] P. M. T. W. L. e. a. Pakzad F, "Efficient topology discovery in OpenFlow-based Software Defined Networks," in *Computer Communications*, 2016.
- [12] O. N. Foundation, "Software-Defined Networking, ONF White Paper," 2012.
- [13] S. Kusumadewi and H. Purnomo, *Aplikasi logika fuzzy untuk pendukung keputusan*, Yogyakarta: Graha Ilmu, 2010.
- [14] M. H. Y. Ahmadreza Montazerolghaem, "Load-balanced and QoS-aware Software-defined Internet of Things," *IEEE Internet of Things Journal*, January 2020.

## Lampiran 1.

Python Code utk Script mencari data Respon Time dan membentuk Data Frame utk semua Data

```

#!/usr/bin/env python3
import paho.mqtt.client as mqtt
import multitimer
import time
import pandas as pd

broker_url = "localhost"
broker_port = 1883
cpu01 = 10
cpu02 = 10
cpu03 = 10
mem01 = 10
mem02 = 10
mem03 = 10
time01 = 10
time02 = 10
time03 = 10
add01 = 'http://192.168.8.1'
add02 = 'http://192.168.8.2'
add03 = 'http://192.168.8.3'
rsp_tm_s1 = 1
rsp_tm_s2 = 1
rsp_tm_s3 = 1
rsp_std_s1 = 1
rsp_std_s2 = 1
rsp_std_s3 = 1

#create initial dataframe
# intialise data of lists.
dataadf = {'id':['server1','server2','server3'],'cpu':[cpu01,cpu02,cpu03],
'mem':[mem01,mem02,mem03],'rsp_tm':[time01,time02,time03]}
# Create DataFrame
df1 = pd.DataFrame(dataadf)

def on_connect(client, userdata, flags, rc):
    print("Connected With Result Code " ,rc)
    if rc==0:
        print("connected OK Returned code=",rc)
    else:
        print("Bad connection Returned code=",rc)

def on_disconnect(client, userdata, rc):
    print("Client Got Disconnected")

#Start get Cpu Mem Data from here
def on_message_from_cpumem01(client, userdata, message):
    global cpu01
    global mem01
    cpumem = str(message.payload.decode("utf-8"))
    print("Message Recieved from Server01: "+ cpumem)
    x = cpumem.split(";")
    cpu01 = float(x[1])
    mem01 = float(x[2])

def on_message_from_cpumem02(client, userdata, message):
    global cpu02

```

## PROPOSAL THESIS

```

global mem02
cpumem = str(message.payload.decode("utf-8"))
print("Message Recieved from Server02: "+ cpumem)
x = cpumem.split(";")
cpu02 = float(x[1])
mem02 = float(x[2])

def on_message_from_cpumem03(client, userdata, message):
    global cpu03
    global mem03
    cpumem = str(message.payload.decode("utf-8"))
    print("Message Recieved from Server03: "+ cpumem)
    x = cpumem.split(";")
    cpu03 = float(x[1])
    mem03 = float(x[2])

def on_message(client, userdata, message):
    print("Message Recieved from Others: "+message.payload.decode())

client = mqtt.Client()
client.on_connect = on_connect
client.on_disconnect = on_disconnect
#To Process Every Other Message
client.on_message = on_message
# edit code for passwords
print("setting password")
client.username_pw_set(username="user01",password="mqtt")

client.connect(broker_url, broker_port)

client.subscribe("sdn/cpumem01", qos=0)

client.message_callback_add("sdn/cpumem01", on_message_from_cpumem01)

client.subscribe("sdn/cpumem02", qos=0)

client.message_callback_add("sdn/cpumem02", on_message_from_cpumem02)

client.subscribe("sdn/cpumem03", qos=0)

client.message_callback_add("sdn/cpumem03", on_message_from_cpumem03)

#Start getting Server Response From Here
def job1():
    global time01
    global add01
    c = pycurl.Curl()
    buffer = BytesIO()
    #c.setopt(c.URL, 'http://pycurl.io/')
    now1 = datetime.datetime.now() # time object
    print("now =", now1)
    print("do curl")
    c.setopt(c.URL, add01)
    c.setopt(c.WRITEDATA, buffer)
    c.perform()
    c.close()
    now2 = datetime.datetime.now() # time object
    selisih = now2 - now1
    #print("selisih :", selisih)

```

## PROPOSAL THESIS

```

#hasil dalam seconds
hasil = selisih.seconds + (selisih.microseconds/1000000)
print("Server1 Respons time =", hasil)
time01 = float(hasil)

# This timer will run job() five times, one second apart
timer1 = multitimer.MultiTimer(interval=10, function=job1, count=-1)
# Also, this timer would run indefinitely...
timer1.start()

#Start getting Server Response From Here
def job2():
    global time02
    global add02
    c = pycurl.Curl()
    buffer = BytesIO()
    #c.setopt(c.URL, 'http://pycurl.io/')
    now1 = datetime.datetime.now() # time object
    print("now =", now1)
    print("do curl")
    c.setopt(c.URL, add02)
    c.setopt(c.WRITEDATA, buffer)
    c.perform()
    c.close()
    now2 = datetime.datetime.now() # time object
    selisih = now2 - now1
    #print("selisih :", selisih)
    #hasil dalam seconds
    hasil = selisih.seconds + (selisih.microseconds/1000000)
    print("Server2 Respons time =", hasil)
    time012 = float(hasil)

# This timer will run job() five times, one second apart
timer2 = multitimer.MultiTimer(interval=10, function=job2, count=-1)
# Also, this timer would run indefinitely...
timer2.start()

#Start getting Server Response From Here
def job3():
    global time03
    global add03
    c = pycurl.Curl()
    buffer = BytesIO()
    #c.setopt(c.URL, 'http://pycurl.io/')
    now1 = datetime.datetime.now() # time object
    print("now =", now1)
    print("do curl")
    c.setopt(c.URL, add03)
    c.setopt(c.WRITEDATA, buffer)
    c.perform()
    c.close()
    now2 = datetime.datetime.now() # time object
    selisih = now2 - now1
    #print("selisih :", selisih)
    #hasil dalam seconds
    hasil = selisih.seconds + (selisih.microseconds/1000000)
    print("Server3 Respons time =", hasil)
    time03 = float(hasil)

# This timer will run job() five times, one second apart
timer3 = multitimer.MultiTimer(interval=10, function=job3, count=-1)

```

## PROPOSAL THESIS



```
# Also, this timer would run indefinitely...
timer3.start()

time.sleep( 2 )
#Start collect all data
#save to dataframe
#ready to print to csv
#publish data clean to MQTT broker
def job4():
    global df1
    global rsp_tm_s1
    global rsp_tm_s2
    global rsp_tm_s3
    global rsp_std_s1
    global rsp_std_s2
    global rsp_std_s3
    global cpu01
    global cpu02
    global cpu03
    global mem01
    global mem02
    global mem03
    global time01
    global time02
    global time03
    global client
    # Appending a Data Frame
    s =
pd.DataFrame({'id':['server1','server2','server3'],'cpu':[cpu01,cpu02,cpu03],
    'mem':[mem01,mem02,mem03],'rsp_tm':[time01,time02,time03]})

    # makes index continuous
    df1= df1.append(s, ignore_index = True)
    print(df1, "\n")

    #select column dataframe for rsp_tm only
    # Set 'Name' column as index
    # on a Dataframe
    df = df1
    df.set_index('id', inplace = True)

    # Using the operator .loc[] to
    # select multiple rows with some
    # particular columns
    df_s1 = df.loc[['server1'],
        ['rsp_tm']]
    rsp_tm_s1 = df_s1['rsp_tm'].tail().mean()
    rsp_std_s1 = df_s1['rsp_tm'].std()
    # Show the dataframe
    print("s1 respon time mean(last 5) = ", rsp_tm_s1)
    print("s1 respon time std = ", rsp_std_s1)
    print(df_s1, "\n")

    df_s2 = df.loc[['server2'],
        ['rsp_tm']]
    rsp_tm_s2 = df_s2['rsp_tm'].tail().mean()
    rsp_std_s2 = df_s2['rsp_tm'].std()
    # Show the dataframe
    print("s2 respon time mean(last 5) = ", rsp_tm_s2)
    print("s2 respon time std = ", rsp_std_s2)
```

```

print(df_s2, "\n")

df_s3 = df.loc[['server3'],
                 ['rsp_tm']]
rsp_tm_s3 = df_s3['rsp_tm'].tail().mean()
rsp_std_s3 = df_s3['rsp_tm'].std()
# Show the dataframe
print("s3 respon time mean(last 5) = ", rsp_tm_s3)
print("s3 respon time std = ", rsp_std_s3)
print(df_s3, "\n")

#Publish data to MQTT Broker
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/rsptm01", payload=rsptm_s1, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/rsptm02", payload=rsptm_s2, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/rsptm03", payload=rsptm_s3, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/rspstd01", payload=rsp_std_s1, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/rspstd02", payload=rsp_std_s2, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/rspstd03", payload=rsp_std_s3, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/cpu01", payload=cpu01, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/cpu02", payload=cpu02, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/cpu03", payload=cpu03, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/mem01", payload=mem01, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/mem02", payload=mem02, qos=0,
retain=False)
#client.publish(topic="sdn/cpumem01", payload=msg, qos=1,
retain=False)
#client.publish(topic="sdn/mem03", payload=mem03, qos=0,
retain=False)

```

## PROPOSAL THESIS

```

# This timer will run job() five times, one second apart
timer4 = multitimer.MultiTimer(interval=1, function=job4, count=5)
# Also, this timer would run indefinitely...
timer4.start()

#Start Lopp MQTT
client.loop_forever()

print("Do Something else")

```

## Lampiran 2.

Python Code utk aplikasi Fuzzy Logic

```

import
numpy
as np

import skfuzzy as fuzz
from matplotlib import pyplot as plt

def fuzzy_mamdani(cpu_val, mem_val, rsp_time_val)
    # Problem: from service quality and food quality to tip amount
    x_cpu = np.arange(0, 1.01, 0.1)
    x_mem = np.arange(0, 1.01, 0.1)
    x_rsptmstddev = np.arange(0, 1.01, 0.1)
    x_load = np.arange(-0.6, 0.4, 0.1)

    # Membership functions
    cpu_verylow = fuzz.trapmf(x_cpu, [-0.2, -0.1, 0.1, 0.4 ])
    cpu_low = fuzz.trapmf(x_cpu, [-0.1, 0.2, 0.3, 0.6 ])
    cpu_medium = fuzz.trapmf(x_cpu, [0.1, 0.45, 0.55, 0.9 ])
    cpu_high = fuzz.trapmf(x_cpu, [0.4, 0.7, 0.8, 1.1 ])
    cpu_veryhigh = fuzz.trapmf(x_cpu, [0.6, 0.95, 1.05, 1.4 ])

    mem_verylow = fuzz.trapmf(x_mem, [-0.2, -0.1, 0.1, 0.4 ])
    mem_low = fuzz.trapmf(x_mem, [-0.1, 0.2, 0.3, 0.6 ])
    mem_medium = fuzz.trapmf(x_mem, [0.1, 0.45, 0.55, 0.9 ])
    mem_high = fuzz.trapmf(x_mem, [0.4, 0.7, 0.8, 1.1 ])
    mem_veryhigh = fuzz.trapmf(x_mem, [0.6, 0.95, 1.05, 1.4 ])

    rsptmstddev_low = fuzz.trapmf(x_rsptmstddev, [-0.1, -0.01, 0.2, 0.6 ])
    rsptmstddev_medium = fuzz.trapmf(x_rsptmstddev, [0.1, 0.4, 0.6, 1 ])
    rsptmstddev_high = fuzz.trapmf(x_rsptmstddev, [0.5, 0.8, 1, 1.3 ])

    load_extdec = fuzz.trapmf(x_load, [-0.7, -0.65, 0.55, 0.5 ])
    load_veryfastdec = fuzz.trimf(x_load, [ -0.6, -0.5, -0.4 ])
    load_fastdec = fuzz.trimf(x_load, [ -0.5, -0.4, -0.3 ])

```

## PROPOSAL THESIS

```

load_dec = fuzz.trimf(x_load, [ -0.4, -0.3, -0.2 ])
load_smalldec = fuzz.trimf(x_load, [ -0.3, -0.2, -0.1 ])
load_verysmalldec = fuzz.trimf(x_load, [ -0.2, -0.1, 0 ])
load_nochange = fuzz.trimf(x_load, [ -0.1, 0, 0.1 ])
load_smallincrease = fuzz.trimf(x_load, [ 0, 0.1, 0.2 ])
load_increase = fuzz.trimf(x_load, [ 0.1, 0.2, 0.3 ])
load_fastincrease = fuzz.trimf(x_load, [ 0.2, 0.3, 0.4 ])
load_veryfastincrease = fuzz.trapmf(x_load, [0.3, 0.35, 0.45, 0.5 ])

# Input: service score and food score
cpu_score = cpu_val / 100
mem_score = mem_val / 100
rspetime_score = rsp_time_val

cpu_verylow_degree = fuzz.interp_membership(
    x_cpu, cpu_verylow, cpu_score)
cpu_low_degree = fuzz.interp_membership(
    x_cpu, cpu_low, cpu_score)
cpu_medium_degree = fuzz.interp_membership(
    x_cpu, cpu_medium, cpu_score)
cpu_high_degree = fuzz.interp_membership(
    x_cpu, cpu_high, cpu_score)
cpu_veryhigh_degree = fuzz.interp_membership(
    x_cpu, cpu_veryhigh, cpu_score)

mem_verylow_degree = fuzz.interp_membership(
    x_cpu, mem_verylow, cpu_score)
mem_low_degree = fuzz.interp_membership(
    x_cpu, mem_low, cpu_score)
mem_medium_degree = fuzz.interp_membership(
    x_cpu, mem_medium, cpu_score)
mem_high_degree = fuzz.interp_membership(
    x_cpu, mem_high, cpu_score)
mem_veryhigh_degree = fuzz.interp_membership(
    x_cpu, mem_veryhigh, cpu_score)

rsptmstddev_low_degree = fuzz.interp_membership(
    x_rsptmstddev, rsptmstddev_low, cpu_score)
rsptmstddev_medium_degree = fuzz.interp_membership(
    x_rsptmstddev, rsptmstddev_medium, cpu_score)
rsptmstddev_high_degree = fuzz.interp_membership(
    x_rsptmstddev, rsptmstddev_high, cpu_score)

# Whole config
fig_scale_x = 2.0

```

## PROPOSAL THESIS

```

fig_scale_y = 1.5
fig = plt.figure(figsize=(6.4 * fig_scale_x, 6.4 * fig_scale_y))
row = 3
col = 3

plt.subplot(row, col, 1)
plt.title("CPU Usage")
plt.plot(x_cpu, cpu_verylow, label="verylow", marker=".")
plt.plot(x_cpu, cpu_low, label="low", marker=".")
plt.plot(x_cpu, cpu_medium, label="medium", marker=".")
plt.plot(x_cpu, cpu_high, label="high", marker=".")
plt.plot(x_cpu, cpu_veryhigh, label="veryhigh", marker=".")
plt.plot(cpu_score, 0.0, label="cpu_score", marker="D")
plt.plot(cpu_score, cpu_verylow_degree,
         label="verylow degree", marker="o")
plt.plot(cpu_score, cpu_low_degree,
         label="low degree", marker="o")
plt.plot(cpu_score, cpu_medium_degree,
         label="medium degree", marker="o")
plt.plot(cpu_score, cpu_high_degree,
         label="high degree", marker="o")
plt.plot(cpu_score, cpu_veryhigh_degree,
         label="veryhigh degree", marker="o")
plt.legend(loc="upper left")

plt.subplot(row, col, 2)
plt.title("Memory Usage")
plt.plot(x_mem, mem_verylow, label="verylow", marker=".")
plt.plot(x_mem, mem_low, label="low", marker=".")
plt.plot(x_mem, mem_medium, label="medium", marker=".")
plt.plot(x_mem, mem_high, label="high", marker=".")
plt.plot(x_mem, mem_veryhigh, label="veryhigh", marker=".")
plt.plot(mem_score, 0.0, label="memory_score", marker="D")
plt.plot(mem_score, mem_verylow_degree,
         label="verylow degree", marker="o")
plt.plot(mem_score, mem_low_degree,
         label="low degree", marker="o")
plt.plot(mem_score, mem_medium_degree,
         label="medium degree", marker="o")
plt.plot(mem_score, mem_high_degree,
         label="high degree", marker="o")
plt.plot(mem_score, mem_veryhigh_degree,
         label="veryhigh degree", marker="o")
plt.legend(loc="upper left")

plt.subplot(row, col, 3)

```

## PROPOSAL THESIS



```
plt.title("Respon Time Standar Deviation")
plt.plot(x_rsptmstddev, rsptmstddev_low, label="low", marker=".")
plt.plot(x_rsptmstddev, rsptmstddev_medium, label="medium", marker=".")
plt.plot(x_rsptmstddev, rsptmstddev_high, label="high", marker=".")
plt.plot(rsptime_score, 0.0, label="respontime_score", marker="D")
plt.plot(rsptime_score, rsptmstddev_low_degree,
         label="low degree", marker="o")
plt.plot(rsptime_score, rsptmstddev_medium_degree,
         label="medium degree", marker="o")
plt.plot(rsptime_score, rsptmstddev_high_degree,
         label="high degree", marker="o")
plt.legend(loc="upper left")

plt.subplot(row, col, 4)
plt.title("Change Load Window")
plt.plot(x_load, load_extdec, label="extdecrease", marker=".")
plt.plot(x_load, load_veryfastdec, label="veryfastdecrease", marker=".")
plt.plot(x_load, load_fastdec, label="fastdecrease", marker=".")
plt.plot(x_load, load_dec, label="decrease", marker=".")
plt.plot(x_load, load_smalldec, label="smalldecrease", marker=".")
plt.plot(x_load, load_verysmalldec, label="verysmalldecrease", marker=".")
plt.plot(x_load, load_nochange, label="nochange", marker=".")
plt.plot(x_load, load_smallincrease, label="smallincrease", marker=".")
plt.plot(x_load, load_increase, label="increase", marker=".")
plt.plot(x_load, load_fastincrease, label="fastincrease", marker=".")
plt.plot(x_load, load_veryfastincrease, label="veryfastincrease",
marker=".")
plt.legend(loc="upper left")

# =====
# Mamdani (max-min) inference method:
# * min because of logic 'and' connective.
# 1) ed_degree <-> loadchange_ed
# 2) vfd_degree <-> loadchange_vfd
# 3) fd_degree <-> loadchange_fd
# 4) dec_degree <-> loadchange_dec
# 5) sd_degree <-> loadchange_sd
# 6) vsd_degree <-> loadchange_vsd
# 7) nc_degree <-> loadchange_nc
# 8) si_degree <-> loadchange_si
# 9) inc_degree <-> loadchange_inc
# 10) fi_degree <-> loadchange_fi
# 11) vfi_degree <-> loadchange_vfi

#extremely decrease load window value change
```

## PROPOSAL THESIS



```
ed_degree = np.fmax(cpu_veryhigh_degree,np.fmax(mem_veryhigh_degree,
rsptmstddev_high_degree))

#very fast decrease load window value change
vfd_degree = np.fmax(cpu_high_degree,np.fmax(mem_veryhigh_degree,
rsptmstddev_high_degree))

#fast decrease load window value change
fd_degree = np.fmax(cpu_veryhigh_degree,np.fmax(mem_medium,
rsptmstddev_medium))

#decrease load window value change
dec_degree = np.fmax(cpu_high_degree,np.fmax(mem_medium_degree,
rsptmstddev_medium_degree))

#small decrease load window value change
sd_degree = np.fmax(cpu_low_degree,np.fmax(mem_high_degree,
rsptmstddev_medium_degree))

#very small decrease load window value change
vsd_degree = np.fmax(cpu_high_degree,np.fmax(mem_verylow_degree,
rsptmstddev_low_degree))

#no change load window value change
nc_degree = np.fmax(cpu_medium_degree,np.fmax(mem_medium_degree,
rsptmstddev_medium_degree))

#small increase load window value change
si_degree = np.fmax(cpu_low_degree,np.fmax(mem_medium_degree,
rsptmstddev_medium_degree))

#increase load window value change
inc_degree = np.fmax(cpu_low_degree,np.fmax(mem_low_degree,
rsptmstddev_low_degree))

#fast increase load window value change
fi_degree = np.fmax(cpu_verylow_degree,np.fmax(mem_low_degree,
rsptmstddev_low_degree))

#very fast increase load window value change
vfi_degree = np.fmax(cpu_verylow_degree,np.fmax(mem_verylow_degree,
rsptmstddev_low_degree))

plt.subplot(row, col, 5)
plt.title("Some Fuzzy Rules")
t = ( "ed_degree <-> loadchange_ed\n"
      "vfd_degree <-> loadchange_vfd\n"
      "fd_degree <-> loadchange_fd\n"
      "dec_degree <-> loadchange_dec\n"
      "sd_degree <-> loadchange_sd\n"
      "vsd_degree <-> loadchange_vsd\n"
      "nc_degree <-> loadchange_nc\n"
      "si_degree <-> loadchange_si\n"
      "inc_degree <-> loadchange_inc\n"
      "fi_degree <-> loadchange_fi\n"
      "vfi_degree <-> loadchange_vfi")
```

# PROPOSAL THESIS



```
plt.text(0.1, 0.5, t)

activation_extdec = np.fmin(ed_degree, load_extdec)
activation_veryfastdec = np.fmin(vfd_degree, load_veryfastdec)
activation_fastdec = np.fmin(fd_degree, load_fastdec)
activation_dec = np.fmin(dec_degree, load_dec)
activation_smalldec = np.fmin(sd_degree, load_smalldec)
activation_verysmalldec = np.fmin(vsd_degree, load_verysmalldec)
activation_nochange = np.fmin(nc_degree, load_nochange)
activation_smallinc = np.fmin(si_degree, load_smallincrease)
activation_increase = np.fmin(inc_degree, load_increase)
activation_fastinc = np.fmin(fi_degree, load_fastincrease)
activation_veryfastinc = np.fmin(vfi_degree, load_veryfastincrease)

plt.subplot(row, col, 6)
plt.title("Tip Activation: Mamdani Inference Method")

plt.plot(x_load, activation_extdec, label="load change ext decrease",
marker=".")
plt.plot(x_load, activation_veryfastdec, label="load change very fast ext
decrease", marker=".")
plt.plot(x_load, activation_fastdec, label="load change fast decrease",
marker=".")
plt.plot(x_load, activation_dec, label="load change decrease", marker=".")
plt.plot(x_load, activation_smalldec, label="load change small decrease",
marker=".")
plt.plot(x_load, activation_verysmalldec, label="load change very small
decrease", marker=".")
plt.plot(x_load, activation_nochange, label="load change nochange",
marker=".")
plt.plot(x_load, activation_smallinc, label="load change small increase",
marker=".")
plt.plot(x_load, activation_increase, label="load change increase",
marker=".")
plt.plot(x_load, activation_fastinc, label="load change fast increase",
marker=".")
plt.plot(x_load, activation_veryfastinc, label="load change very fast
increase", marker=".")
plt.legend(loc="upper left")

# Apply the rules:
# * max for aggregation, like or the cases
aggregated1 = np.fmax(
    activation_extdec,
    np.fmax(activation_veryfastdec, activation_fastdec))
aggregated2 = np.fmax(
```

```

activation_dec,
np.fmax(activation_smalldec, activation_verysmalldec))
aggregated3 = np.fmax(
    activation_nochange,
    np.fmax(activation_smallinc, activation_increase))
aggregated4 = np.fmax(
    activation_fastinc,
    np.fmax(activation_veryfastinc, aggregated1))
aggregated5 = np.fmax(
    aggregated2,
    np.fmax(aggregated3, aggregated4))

# Defuzzification
tip_centroid = fuzz.defuzz(x_load, aggregated5, 'centroid')
tip_bisector = fuzz.defuzz(x_load, aggregated5, 'bisector')
tip_mom = fuzz.defuzz(x_load, aggregated5, "mom")
tip_som = fuzz.defuzz(x_load, aggregated5, "som")
tip_lom = fuzz.defuzz(x_load, aggregated5, "lom")

print(tip_centroid)
print(tip_bisector)
print(tip_mom)
print(tip_som)
print(tip_lom)

plt.subplot(row, col, 6)
plt.title("Aggregation and Defuzzification")
plt.plot(x_load, aggregated5, label="fuzzy result", marker=".")
plt.plot(tip_centroid, 0.0, label="centroid", marker="o")
plt.plot(tip_bisector, 0.0, label="bisector", marker="o")
plt.plot(tip_mom, 0.0, label="mom", marker="o")
plt.plot(tip_som, 0.0, label="som", marker="o")
plt.plot(tip_lom, 0.0, label="lom", marker="o")
plt.legend(loc="upper left")

plt.savefig("7-tipping-problem-mamdani.png")
plt.show()

```

### Lampiran 3.

Python Script untuk Aplikasi Load Balancing dengan API Ryu Controller

```

# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0

```

## PROPOSAL THESIS



```
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or  
# implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
  
import logging  
import json  
import random  
  
from ryu.base import app_manager  
from ryu.controller import ofp_event  
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER  
from ryu.controller.handler import set_ev_cls  
from ryu.ofproto import ofproto_v1_3  
from ryu.ofproto import inet  
from ryu.lib.packet import packet  
from ryu.lib.packet import ethernet  
from ryu.lib.packet import ether_types  
from ryu.lib.packet import ipv4  
from ryu.lib.packet import tcp  
from ryu.lib.packet import arp  
from ryu.lib.packet import in_proto  
from ryu.lib import mac as mac_lib  
from ryu.lib import ip as ip_lib  
from ryu.lib import dpid as dpid_lib  
from ryu.ofproto import ether, inet  
from ryu.ofproto import ofproto_v1_0, ofproto_v1_3  
  
UINT32_MAX = 0xffffffff  
  
class SimpleSwitch13Lb(app_manager.RyuApp):  
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]  
  
    def __init__(self, *args, **kwargs):  
        super(SimpleSwitch13Lb, self).__init__(*args, **kwargs)  
        self.mac_to_port = {}  
        self.server_index = 0  
        self.servers = []  
        self.rewrite_ip_header = True  
  
        self.virtual_ip = None  
        self.virtual_ip = "10.0.0.5"  
        self.virtual_mac = "00:00:00:00:00:05" # Pick something dummy and  
                                                # probably not valid  
  
        self.servers.append({'ip':'10.0.0.2', 'mac':'00:00:00:00:00:02'})  
        self.servers.append({'ip':'10.0.0.3', 'mac':'00:00:00:00:00:03'})  
        self.servers.append({'ip':'10.0.0.4', 'mac':'00:00:00:00:00:04'})  
  
        #server = {}  
        #server[0] = {'ip':IPAddr("10.0.0.2"),  
        #'mac':EthAddr("00:00:00:00:00:02"), 'outport': 2}  
        #server[1] = {'ip':IPAddr("10.0.0.3"),  
        #'mac':EthAddr("00:00:00:00:00:03"), 'outport': 3}  
        #server[2] = {'ip':IPAddr("10.0.0.4"),  
        #'mac':EthAddr("00:00:00:00:00:04"), 'outport': 4}  
        #total_servers = len(server)
```

## PROPOSAL THESIS



```
def get_attachment_port(self, dpid, mac):
    if dpid in self.mac_to_port:
        table = self.mac_to_port[dpid]
        if mac in table:
            return table[mac]
    return None

@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    # install table-miss flow entry
    #
    # We specify NO BUFFER to max_len of the output action due to
    # OVS bug. At this moment, if we specify a lesser number, e.g.,
    # 128, OVS will send Packet-In with invalid buffer_id and
    # truncated packet data. In that case, we cannot output packets
    # correctly. The bug has been fixed in OVS v2.1.0.
    match = parser.OFPMatch()
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                      ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 0, match, actions)

def add_flow(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPIInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                          actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                               priority=priority, match=match,
                               instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                               match=match, instructions=inst)
    datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    # If you hit this you might want to increase
    # the "miss_send_length" of your switch
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ether.ethernet)[0]

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        # ignore lldp packet
        return
    dst = eth.dst
    src = eth.src
```

## PROPOSAL THESIS



```
dpid = format(datapath.id, "d").zfill(16)
self.mac_to_port.setdefault(dpid, {})

self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)

# Only handle IPv4 traffic going forward
if eth.ethertype == ether_types.ETH_TYPE_IP:
    #iphdr = pkt.get_protocols(ipv4.ipv4)[0]
    iphdr = pkt.get_protocols(ipv4.ipv4)[0]
    # Only handle traffic destined to virtual IP
    if iphdr.dst == self.virtual_ip:
        # Only handle TCP traffic
        if iphdr.proto == in_proto.IPPROTO_TCP:
            tcphdr = pkt.get_protocols(tcp.tcp)[0]
            print("Load Balancing Start")

        valid_servers = []
        for server in self.servers:
            outport = self.get_attachment_port(dpid,
server['mac'])
            if outport != None:
                server['outport'] = outport
                valid_servers.append(server)

        total_servers = len(valid_servers)

        # If we there are no servers with location known, then
skip
        if total_servers == 0:
            return

        # Round robin selection of servers
        index = self.server_index % total_servers
        selected_server_ip = valid_servers[index]['ip']
        selected_server_mac = valid_servers[index]['mac']
        selected_server_outport =
valid_servers[index]['outport']
        self.server_index += 1
        print "Selected server", selected_server_ip

##### Setup route to server
match = parser.OFPMatch(in_port=in_port,
                        eth_type=eth.ethertype, eth_src=eth.src,
eth_dst=eth.dst,
                        ip_proto=iphdr.proto, ipv4_src=iphdr.src,
ipv4_dst=iphdr.dst,
                        tcp_src=tcp hdr.src_port,
tcp_dst=tcp hdr.dst_port)

        if self.rewrite_ip_header:
            actions =
[parser.OFPActionSetField(eth_dst=selected_server_mac),
parser.OFPActionSetField(ipv4_dst=selected_server_ip),
parser.OFPActionOutput(selected_server_outport) ]
else:
```

## PROPOSAL THESIS



```
        actions =
[parser.OFPActionSetField(eth_dst=selected_server_mac),
 parser.OFPActionOutput(selected_server_outport) ]

        inst =
[parser.OFPIstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]

        cookie = random.randint(0, 0xffffffffffffffff)

        mod = parser.OFPFlowMod(datapath=datapath, match=match,
idle_timeout=10,
                           instructions=inst, buffer_id = msg.buffer_id,
cookie=cookie)
        datapath.send_msg(mod)

##### Setup reverse route from server
match =
parser.OFPMatch(in_port=selected_server_outport,
                 eth_type=eth.ethertype,
eth_src=selected_server_mac, eth_dst=eth.src,
                     ip_proto=iphdr.proto,
ipv4_src=selected_server_ip, ipv4_dst=iphdr.src,
                     tcp_src=tcp hdr.dst_port,
tcp_dst=tcp hdr.src_port)

        if self.rewrite_ip_header:
            actions =
([parser.OFPActionSetField(eth_src=self.virtual_mac),
parser.OFPActionSetField(ipv4_src=self.virtual_ip),
                     parser.OFPActionOutput(in_port) ])
        else:
            actions =
([parser.OFPActionSetField(eth_src=self.virtual_mac),
                     parser.OFPActionOutput(in_port) ])

        inst =
[parser.OFPIstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]

        cookie = random.randint(0, 0xffffffff)

        mod = parser.OFPFlowMod(datapath=datapath, match=match,
idle_timeout=10,
                           instructions=inst, cookie=cookie)
        datapath.send_msg(mod)

# learn a mac address to avoid FLOOD next time.
self.mac_to_port[dpid][src] = in_port

if dst in self.mac_to_port[dpid]:
    out_port = self.mac_to_port[dpid][dst]
else:
    out_port = ofproto.OFPP_FLOOD

actions = [parser.OFPActionOutput(out_port)]

# install a flow to avoid packet_in next time
if out_port != ofproto.OFPP_FLOOD:
    match = parser.OFPMatch(in_port=in_port, eth_dst=dst,
eth_src=src)
```

## PROPOSAL THESIS



```
# verify if we have a valid buffer_id, if yes avoid to send
both
# flow_mod & packet_out
if msg.buffer_id != ofproto.OFP_NO_BUFFER:
    self.add_flow(datapath, 1, match, actions, msg.buffer_id)
    return
else:
    self.add_flow(datapath, 1, match, actions)
data = None
if msg.buffer_id == ofproto.OFP_NO_BUFFER:
    data = msg.data

out = parser.OFPPacketOut(datapath=datapath,
buffer_id=msg.buffer_id,
                           in_port=in_port, actions=actions,
data=data)
datapath.send_msg(out)
```