BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini telah terjadi perubahan paradigma jaringan komputer. Jaringan dikontrol oleh perangkat lunak, yang dikenal dengan *software definition network* (SDN). SDN memisahkan antara bagian kontrol (*control plane*) dan bagian data (*data plane*) dalam jaringan agar ebih optimal dan lebih efisien.

SDN adalah paradigma jaringan yang menyederhanakan tugas manajemen jaringan. Selain itu, SDN membuka pintu untuk inovasi jaringan melalui antarmuka fleksibel yang diprogam untuk mengendalikan seluruh jaringan. (Mousa, Bahaa-Eldin, & Sobb, 2016)

Konsep SDN ini memudahkan operator dan administrator dalam mengelola jaringan. SDN juga mampu memberikan solusi untuk permasalahan jaringan yang ada sekarang ini, seperti sulitnya mengintegrasikan teknologi baru karena masalah perbedaan platform perangkat keras, kinerja yang buruk karena ada beberapa operasi yang berlebihan pada protokol dan kesulitan menyediakan layanan-layanan baru. Konsep SDN mempermudah dan mempercepat inovasi pada jaringan. Ide-ide baru yang lebih baik dan dapat dengan cepat diimplementasikan pada lingkungan jaringan muncul seiring dengan diimplementasikannya SDN.

SDN erat kaitannya dengan OpenFlow sehingga banyak yang beranggapan bahwa SDN adalah OpenFlow. OpenFlow adalah sebuah elemen dalam arsitektur SDN. *OpenFlow* merupakan standar terbuka dari komunikasi protokol yang mampu melakukan pemisahan, serta mampu menciptakan komunikasi yang sangat baik antara *control plane* dan *data plane*. OpenFlow bukanlah satu-satunya protokol yang dapat digunakan untuk pengembangan SDN. Protokol lain diantaranya: BGP, XMPP, OVSDB, dan MPLS-TP.(Kaur, 2014).

Ada banyak jenis *controller* SDN antara lain: POX, RYU, ONOS, Floodlight, OpenDaylight, dan lain sebagainya. Setiap *controller* memiliki keunggulan dan dasar pemrograman yang berbeda. Perbedaan tersebut menghasilkan performa *controller* yang juga berbeda diukur dari *throughput, delay, latency* dan lain-lain. Penelitian ini bertujuan untuk membandingkan performa jenis *controller* ONOS dan RYU dimana ONOS menggunakan dasar pemrograman java sedangkan RYU menggunakan dasar

pemrograman Python. Berdasarkan perbedaan tersebut, peneliti menganalisa performa kedua *controller* pada parameter besar *throughput* dan tingginya *latency*.

ONOS (*Open Network Operating System*) adalah *controller* jaringan SDN bersifat *open source*. ONOS diprakarsai dan dibuat oleh ON.Lab dengan kerja-sama lintas sektor, baik operator, vendor maupun universitas. ONOS menggunakan bahasa pemrograman Java.

Ryu merupakan *controller* jaringan SDN yang bersifat terbuka, yang dirancang dengan tujuan untuk meningkatkan kemampuan jaringan yang mudah dikelola dan menyesuaikan cara penanganan lalu lintas jaringan. Ryu yang berarti "flow" berasal dari bahasa jepang. Ryu menggunakan bahasa Python.

1.2 Identifikasi Masalah

Dari latar belakang yang tertulis di atas, peneliti mengidentifikasi masalah yang akan dijadikan bahan penelitian sebagai berikut:

1. Performa kerja dari RYU dan ONOS SDN *Controller.*
2. Perbandingan *throughput* dan *latency* jaringan SDN menggunakan RYU dan ONOS.
3. Perbandingan performa kinerja dari topologi *tree* dan *star* dalam SDN.

1.3 Rumusan Masalah

Berdasarkan latar belakang dan identifikasi yang tertulis di atas berikut adalah rumusan masalah:

1. Bagaimana performa *latency* dari RYU dan ONOS SDN *controller*?
2. Bagaimana performa *throughput* dari RYU dan ONOS SDN *controller*?
3. Bagaimana perbandingan kinerja SDN dengan topologi *tree* dan *star*?

1.4 Batasan Masalah

Mengingat luasnya jangkauan dari permasalahan ini maka perlu adanya batasan masalah yang jelas mengenai apa yang dibuat dan diselesaikan dalam tugas akhir ini. Adapun batasan masalah pada penelitian ini sebagai berikut:

1. Pembuatan simulasi topologi menggunakan mininet.
2. *Controller* yang dibandingkan hanya 2, RYU dan ONOS.

1.5 Tujuan Penelitian

Berdasarkan latar belakang yang tertulis di atas, maka tujuan skripsi ini adalah sebagai berikut:

1. Mengetahui performa *controller* RYU dan ONOS.
2. Mengetahui performa SDN dengan topologi yang berbeda.
3. Mengetahui pemanfaatan *controller* RYU dan ONOS dalam tinjauan agama Islam.

1.6 Manfaat Penelitian

Diharapkan, skripsi ini memberi manfaat sebagai berikut:

1. Mengetahui hasil perbandingan performa SDN menggunakan RYU dan ONOS dalam *throughput* dan *latency.*
2. Memberikan informasi tentang performa *controller* mana yang lebih baik terkait skalabilitas jaringan.

BAB 2

TINJAUAN PUSTAKA

2.1 Topologi

Topologi jaringan adalah suatu cara atau konsep yang digunakan untuk menghubungkan dua komputer atau lebih, berdasarkan hubungan geometris antara unsur-unsur dasar penyusun jaringan, yaitu *node*, *link*, dan *station* (Supriyadi & Gartina, 2007).

1. Topologi *Tree*

Topologi *tree* merupakan kombinasi antara topologi bintang dan bus. Topologi *tree* adalah rangkaian dari topologi bintang yang disambungkan ke dalam satu topologi bus yang digunakan sebagai *backbone* (Ramadhan 2017).

Keuntungan topologi *tree*:

a. Pemasangan kabel dalam topologi *tree* mudah digunakan
b. Mendukung penerapan jaringan komputer dengan skala besar.
c. Topologi *tree* memungkinkan untuk mengaktifkan fungsi *repeater* yang terdapat dalam sebuah *hub*.
d. Topologi *tree* memungkinkan untuk memiliki jaringan *point to point.*

Kekurangan topologi *tree*:

a. Cara kerja jaringan yang diterapkan pada topologi *tree* cenderung lebih lamban.
b. Adanya kabel yang berperan sebagai *backbone* yang berada di posisi bawah akan berpengaruh pada pusat dari topologi *tree* yang dipasang.
c. Proses pendeteksian kesalahan yang terjadi pada jaringan tergolong sedikit.

2. Topologi *Star*

Topologi *star* atau yang disebut juga topologi bintang merupakan bentuk topologi jaringan yang berupa konvergensi dari *node* atau pengguna. Topologi ini memerlukan biaya yang cukup mahal. (Ronny 2018)

Karakteristik topologi *star*:

a. Setiap *node* berkomunikasi langsung dengan konsentrator (*hub*).
b. Bila setiap paket yang masuk ke *hub*, di-*broadcast* ke seluruh *node* yang terhubung, maka kinerja akan semakin menurun.

c. Jika salah satu *ethernet card* rusak atau salah satu kabel putus, maka seluruh jaringan masih tetap bisa berkomunikasi atau tidak terjadi *down* pada jaringan keseluruhan.
d. Tipe kabel yang digunakan ialah kabel UTP.

Keuntungan topologi *star*:

a. Cukup mudah untuk mengubah dan menambah komputer ke dalam jaringan dengan topologi *star* tanpa adanya gangguan saat koneksi sedang berlangsung.
b. Apabila salah satu komputer mengalami gangguan pada jaringan, maka seluruh jaringan tidak akan *down*.
c. Dapat menggunakan beberapa tipe kabel yang berbeda dalam jaringan.

Kerugian topologi *star*:

a. Memiliki satu titik kesalahan, yaitu terletak pada *hub*. Jika *hub* pusat mengalami gangguan, maka seluruh jaringan akan gagal beroperasi.
b. Membutuhkan lebih banyak kabel daripada topologi jaringan yang lain.
c. Jumlah terminal yang terbatas, tergantung pada *port* yang ada pada *hub*.
d. Lalu lintas data yang padat dapat menyebabkan jaringan menjadi lambat.

2.2 SDN

SDN terdiri dari 3 layer, yaitu *data plane layer*, *control plane layer*, dan *application layer*. *Forwarding plane* (*data plane*) merupakan lapisan pertama dari arsitektur SDN. *Forwarding plane* terdiri dari *router*, OpenFlow *switch*, dan peralatan jaringan lainnya. Sedangkan *control plane layer* merupakan lapisan kedua arsitektur SDN yang terdiri dari *controller* yang bertugas sebagai pengendali SDN (OpenDayLight, Floodlight, RYU, ONOS, dan lain sebagainya). *Application layer* merupakan lapisan paling atas dari arsitektur SDN, terdiri dari *traffic engineering* (rekayasa trafik), virtualisasi jaringan, QoS (*quality of service*), *monitoring, routing, security, access control,* dan manajemen *bandwidth.* (Asadollahi, 2017)

2.3 RYU *Controller*

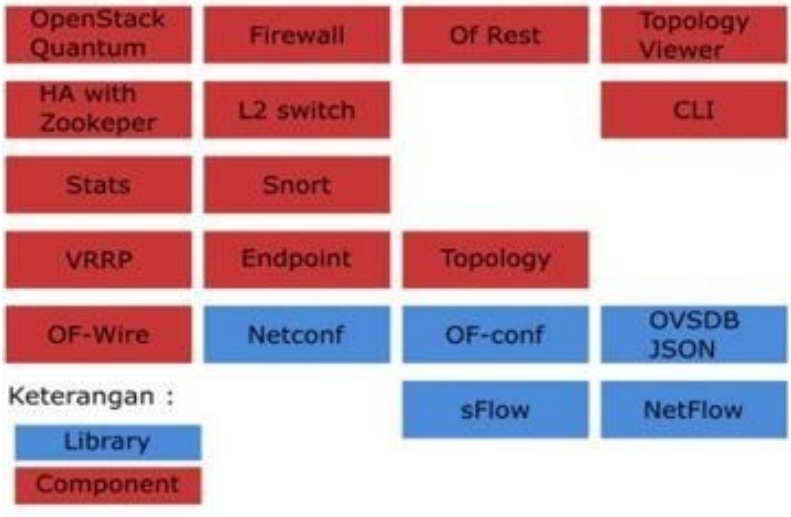
Nama RYU diambil dari Bahasa jepang yang mempunyai arti “aliran”. *Framework* ini bersift *open source* di bawah Apache 2.0 dan didukung dengan API yang memudahkan *developer* untuk membangun sebuah jaringan SDN yang mempunyai kemampuan

network management dan *control application*. (Nuruzzamanirridha, Dyah, & Hariyani, 2016)

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 *software defined network.* RYU merupakan aplikasi *open source* yang dikembangan 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 pemgrograman Python yang mudah dalam pemakaiannya serta memiliki dokumentasi yang lengkap sehingga lebih mudah menemukan solusi jika terdapat permasalahan. *Controller* RYU mendukung beberapa protokol dalam *software defined network* diantaranya OpenFlow, Netconf, OF-config serta lainnya. (Afan, 2018)

Keunggulan *controller* RYU dibandingkan *controller* lain diantaranya:

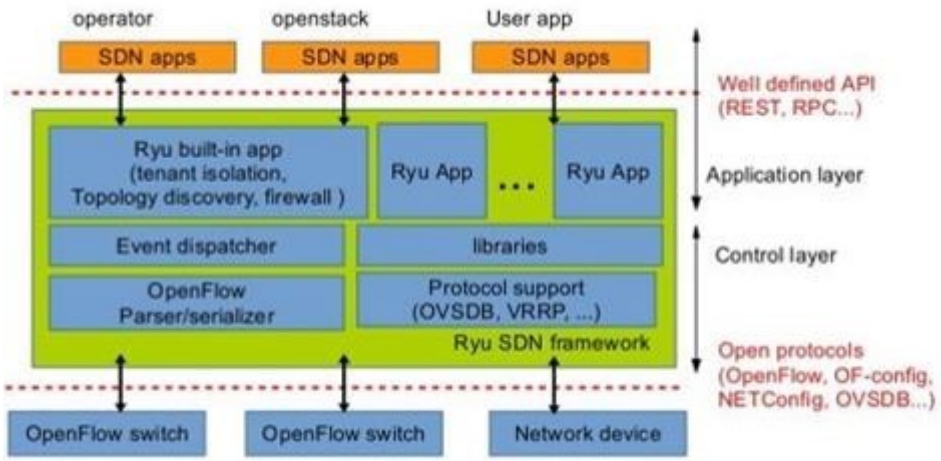
a. RYU menyediakan banyak komponen yang berguna untuk aplikasi *software defined network*.
b. Komponen lama dalam RYU dapat dimodifikasi sesuai dengan kebutuhan dan diterapkan pada komponen yang baru.
c. Menggabungkan komponen untuk membangun aplikasi.



Gambar 1. *Framework* RYU

Framework RYU berada pada *control layer* pada arsitektur SDN, dan beberapa aplikasi pada RYU berada pada *application layer* untuk berkomunikasi dengan Aplikasi

SDN lain menggunakan API seperti REST, RPC, dan sebagainya seperti terlihat pada Gambar 1 dan Gambar 2.



Gambar 2. Arsitektur RYU

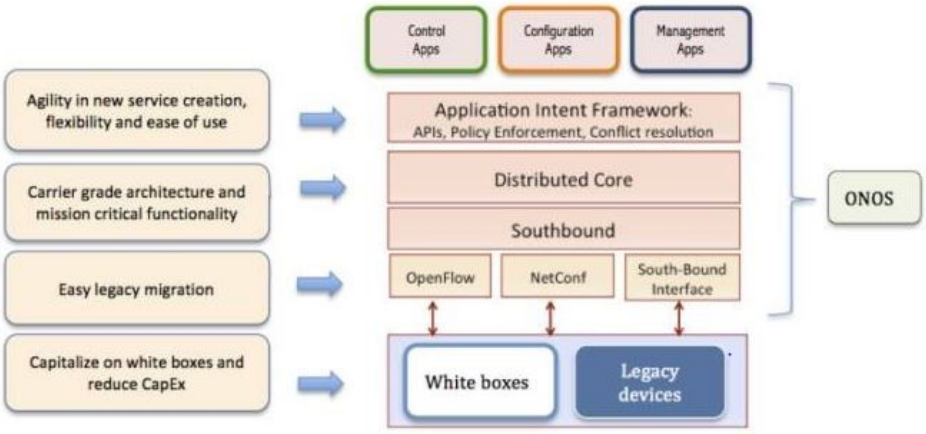
2.4 ONOS *Controller*

ONOS (*open network operating system*) adalah sebuah sistem operasi yang dirancang untuk membantu penyedia layanan jaringan, membangun jaringan berbasis *carrier-grade* yang dirancang untuk skalabilitas, ketersediaan dan kinerja tinggi. Walaupun dirancang khusus untuk memenuhi kebutuhan penyedia layanan, ONOS dapat bertindak sebagai *controller* SDN untuk jaringan area local (LAN) dan jaringan pusat data. (Putra, Pramukantoro, & Yahya, 2018)

ONOS merupakan sistem operasi jaringan *open source* teknologi SDN yg berorientasi pada jaringan operator (*carrier-grade service provider*). ONOS didisain dengan filosofi HA (*high availability*), kinerja tinggi (HP – *high performance*) dan kemampuan *scale-out* yg baik.(Ramadhan 2017)

ONOS diprakarsai dan dibuat oleh ON.Lab, dengan kerja-sama lintas sektor, baik operator, vendor maupun universitas. ONOS versi pertama (1.0.0) dirilis tanggal 5 Desember 2014. Rilis setiap kuartal, saat ini (01.2018) sudah dalam versi 1.12.0 (stabil versi git*hub* dan docker *hub*).

ONOS ditulis dalam bahasa Java dan menggunakan OSGi untuk manajemen fungsionalitas. Setiap fitur dalam ONOS diaktifkan melalui Apache Karaf (*OSGi runtime*)(Gambar 3).



Gambar 3. ONOS *Controller*

ONOS diperkenalkan sebagai sistem operasi jaringan SDN yg multi-entitas dan terdistribusi, serta dirancang untuk HA, kinerja tinggi, skalabilitas dan dengan abstraksi NB (*Northbound*), SB (*Southbound*) yg konsisten (dan semakin baik).

Atribut kunci yg membuat ONOS sesuai untuk jaringan operator, adalah:

- **NB**: Menyediakan AIF (*application intent framework*) sebagai NBI untuk aplikasi. AIF adalah framework (*policy-driven*) yg dapat digunakan oleh devs/ops sebagai bahasa *high-level* tanpa harus memikirkan bagaimana implementasinya di jaringan
- **NB**: Menyediakan informasi struktur (*graph*) seluruh jaringan sebagai bagian dari abstraksi NB
- **Core**: Sistem terdistribusi dengan implikasi HA, HP dan *scale-out*
- **SB**: Menyediakan abstraksi SB untuk *discovery*, konfigurasi dan *programmability*; mendukung OpenFlow dan sejumlah antar-muka atau *device* lainnya.

2.5 Mininet

Mininet merupakan emulator jaringan yang menciptakan jaringan virtual *host*, *switch*, *controller*, dan *link* pada kernel Linux. Mininet mendukung eksperimen yang lengkap pada komputer untuk penelitian, pengembangan, pembelajaran, pembuatan prototip, pengujian, *debugging*, dan lainnya. (Ummah, 2016)

Tabel 1. Perintah untuk menjalankan Mininet

Commands	**Deskripsi**
mn	Menjalankan Mininet
--custom	Menjalankan Mininet dengan menggunakan topologi buatan sendiri
--*controller* remote	Menjalankan *controller* secara *remote*
--*switch* ovsk	Menggunakan *switch* tipe ovsk
--ip	IP untuk yang akan digunakan oleh *controller*
*node*s	Untuk menampilkan *node* yang ada pada topologi
link	Untuk menampilkan *link* yang terjadi pada topologi (contoh; h1-s1, h2-s2, dan sebagainya)
pingall	Untuk melakukan *ping* pada semua *node* yang ada pada topologi
pingpair	Untuk melakukan *ping* secara berpasangan (contoh; pingpair h1 h2)
pingallfull	Melakukan *ping* terhadap semua *node*
pingpairfull	Melakukan *ping* terhadap semua *node* dengan memasangkan sepasang *node*
xterm	Menampilkan terminal dari *node*

2.6 QoS(*Quality of Service*)

Quality of service merupakan parameter tentang seberapa baik jaringan dan merupakan suatu usaha yang mendefinisikan karakteristik dan sifat dari satu layanan. QoS digunakan untuk mengukur sekumpulan atribut kinerja yang telah dispesifikasikan dan diasosiasikan dengan suatu layanan.

Komponen *monitoring* QoS terdiri dari aplikasi *monitoring*, QoS *monitoring, monitor*, dan objek yang dimonitor.

1. Aplikasi *monitoring*

   Merupakan sebuah antar muka bagi administrator jaringan, komponen ini berfungsi mengambil informasi lalu lintas paket data, menganalisisnya dan mengirimkan hasil analisis kepada pengguna. Berdasarkan hasil analisis tersebut, seorang administrator jaringan dapat melakukan operasi-operasi yang lain.

2. QoS *Monitoring*

Menyediakan mekanisme *monitoring* QoS dengan mengambil informasi nilai-nilai parameter QoS dari lalu lintas paket data.

3. Monitor

Mengumpulkan dan merekam informasi lalu lintas paket data yang selanjutnya akan dikirimkan kepada *monitoring application*. Monitor melakukan pengukuran aliran paket data secara *realtime* dan melaporkan hasilnya kepada *monitoring application*.

4. Objek yang dimonitor

Merupakan informasi seperti atribut dan aktifitas yang dimonitor di dalam jaringan. Di dalam konteks *monitoring* QoS, informasi-informasi tersebut merupakan aliran-aliran paket data yang dimonitor secara *realtime*. Tipe aliran paket data tersebut dapat diketahui dari alamat sumber (*source*), tujuan (*destination*) di *layer* IP, *port* yang dipergunakan misalnya UDP (*user datagram protocol*) atau TCP (*transmission control protokol*), dan parameter di dalam paket RTP (*real time transport protocol*).

Parameter *Quality of Service* yang diteliti pada skripsi ini meliputi:

1. *Throughput*

Throughput yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam bps (*bit per second*). *Throughput* adalah jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. Kategori *throughput* dibagi menjadi empat seperti yang terdapat pada Tabel 2.

Tabel 2. Kategori *Throughput*

Kategori *Throughput*	Indeks
Sangat Bagus	>450 mbps
Bagus	300 s/d 450 mbps
Sedang	150 s/d 300 mbps
Buruk	<150 mbps

2. Delay (*Latency*)

Latency merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Latency* (*delay*) dapat dipengaruhi oleh jarak, media fisik, *congestion* atau juga waktu proses yang lama. Kategori *latency* terbagi menjadi 4 (empat), selengkapnya pada Tabel 3.

Tabel 3. Katagori *Delay*

Kategori *Latency*	Besar *Delay* (ms)
Sangat Bagus	<150 ms
Bagus	150 ms s/d 300 ms
Sedang	300 ms s/d 450 ms
Jelek	>450 ms

Menurut Ashton dan Metzler (2013), ada 10 hal yang dapat dipertimbangkan dalam memilih jenis SDN *controller* yang akan digunakan, yaitu:

1. Dukungan OpenFlow
2. Virtualisasi jaringan
3. Fungsi jaringan
4. Skalabilitas
5. Performa kerja
6. Kemampuan pemrograman jaringan
7. Keandalan
8. Keamanan jaringan
9. Pemantauan terpusat dan virtualisasi
10. *Vendor controller* SDN

Dari pertimbangan ini, penelitian ini memilih uji performa. Uji ini terdiri dari 2 QoS (*quality of service*) yaitu *latency* dan *throughput*.

2.7 Wireshark

Wireshark merupakan salah satu tools atau aplikasi "*network analyzer*" atau penganalisa jaringan. Penganalisaan kinerja jaringan itu melingkupi berbagai hal, mulai dari proses menangkap paket-paket data atau informasi yang lewat jaringan, sampai pada digunakan pula untuk *sniffing* (memperoleh informasi penting seperti *password email*, dll). Wireshark merupakan *free tools* untuk *network analyzer* yang ada saat ini. Tampilan dari Wireshark sangat membantu *user* karena menggunakan tampilan grafis atau GUI (*graphical user interface*)..

Fungsi Kinerja Wireshark:

1. Menganalisa jaringan.

2. Menangkap paket data atau informasi yang berkeliaran dalam jaringan.
3. Penganalisaan informasi yang dapat dilakukan dengan *sniffing*, dengan begitu dapat diperoleh informasi penting seperti *password*, dll.
4. Membaca data secara langsung dari Ethernet, Token-Ring, FDDI, serial (PPP dan SLIP), 802.11 wireless LAN, dan koneksi ATM.
5. Dapat mengetahui IP seseorang melalui *capture packet* yang masuk.
6. Menganalisis transmisi paket data dalam jaringan, proses koneksi, dan transmisi data antar komputer, dan lainnya.

2.8 Penelitian Sebelumnya

Penelitian yang dilakukan oleh Sizka L. Hanifa dan Rikie Kartadie dengan judul *"Uji Performa Kontroler Software Defined Network Floodlight vs ONOS"*, Floodlight dan ONOS dibuat dengan menggunakan bahasa pemrograman yang sama yaitu java untuk pengembangan yang lebih cepat. Hasil dan penelitiannya ONOS memiliki fitur-fitur yang hebat dengan *user interface* yang mudah dipahami. Perbedaan antara Floodlight dan ONOS terletak pada fitur dan *user interface*. Pengujian *controller* Floodlight dan ONOS dijalankan pada VirtualBox dengan skenario topologi *single*, 5 *host*. Pengujian diletakkan dalam 1 *host*/*personal computer* dan dijalankan pada sumber daya yang sama. Pengujian dengan mencari nilai *throughput* dan *latency* dengan hasil pengujian terbaik.

Penelitian lainnya dilakukan oleh Moh Wahyudi Putra, Eko Sakti Pramukantoro, Widhi Yahya dengan judul *"Analisis Perbandingan Performansi Kontroler Floodlight, Maestro, RYU, POX dan ONOS dalam Arsitektur Software Defined Network"* pada penelitian ini dilakukan dengan membandingkan kinerja dari *controller* dalaam parameter *throughput* dan *latency.* Dalam hal ini *throughput* adalah besaran jumlah *flow* pada tiap detik yang dapat ditangani. Parameter *latency* adalah jumlah respon yang dapat diberikan oleh *controller* dalam tiap detiknya. Kedua parameter tersebut dihasilkan dari perangkat lunak Cbench. Dalam hal ini penulis bertujuan untuk mengetahui *controller* yang menghasilkan kinerja lebih unggul.

Penelitian yang lain dilakukan oleh Romi Affan, Ir. Agus Virguno., M.T., Drs. Ir. R Rumani M., BC.TT., M.Sc. dengan judul *"Analisis Efek penggunaan Kontroler RYU dan POX pada Performansi Jaringan SDN"* dalam penelitian ini membandingkan nilai

QoS jaringan yang dibangun menggunakan *controller* RYU dan POX. Kedua *controller* ini diuji pada topologi *full-mesh* dengan jumlah *switc*h 6, 8 dan 10 dan masing-masing *switch* mempunyai 2 *host* yang terhubung.

Penelitian yang dilakukan oleh Safida Reynita Sari, Dr. Ir Rendy Munadi,M.T., Danu Dwi Sanjoyo,S.T.,M.T dengan judul *"Analisis Performansi Segment Routing Pada Software Defined Network Menggunakan Kontroler ONOS"*. Pada penelitian ini disimulasikan penggunaan *segment routing* dan membandingkannya dengan performansi jaringan tanpa menggunakan aturan *segment routing* pada SDN dengan menggunakan *controller* ONOS. Topologi yang digunakan adalah *leaf spine*. Parameter yang diukur adalah *resource utilization* dan QoS. Hasil penelitian ini penggunaan memori pada segment routing sebesar 84% dan 50% untuk tanpa *segment routing*.

Berdasarkan penelitian-penelitian di atas peneliti melakukan penelitian dengan subjek dan topologi yang berbeda. Penelitian-penelitian sebelumnya membandingkan atau menggunakan SDN *controller* ONOS dan RYU namun belum ada yang melakukan penelitian perbandingan ONOS dan RYU menggunakan topologi yang berbeda. Peneliti ingin melakukan pengujian dengan topologi *tree* dan *star* dengan masing-masing 9 (sembilan) *switch* dan 27 (dua pulu tujuh) *host* untuk parameter *throughput* dan *latency*.