ANALISA KINERJA VLAN BERBASIS *SOFTWARE DEFINED NETWORK* DENGAN *RYU CONTROLLER*

(Skripsi)

Oleh

M. CHAIRUL ANAM
1715061023

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG

2023

ABSTRAK

ANALISA KINERJA VLAN BERBASIS *SOFTWARE DEFINED NETWORK* DENGAN *RYU CONTROLLER*

Oleh

M. CHAIRUL ANAM

Software Defined Network adalah arsitektur jaringan yang memisahkan antara control plane dan data plane dalam perangkat jaringan yang kemudian membuat control plane menjadi independen dan dapat diprogram. Tugas *control plane* tersebut diambil controller dimana controller dapat mengatur semua aturan yang ada dalam seluruh perangkat jaringan, salah satu jenis controller tersebut adalah *ryu controller*. VLAN merupakan konfigurasi jaringan yang mendistribusikan beberapa segmen yang berbeda kepada perangkat yang terhubung pada jaringan yang dapat meningkatkan perfoma jaringan. Penelitian ini bertujuan untuk mengetahui bagaimana kinerja VLAN pada *Software Defined Network* yang menggunakan *Ryu Controller* dengan menggunakan metode simulasi dengan 2 skenario, yaitu topologi linear 3 OpenFlow Switch 30 host dan topologi linear 3 OpenFlow Switch 60 host yang diberi *traffic* 100, 200, 300, dan 400 Mb. Pengujian dilakukan menggunakan *app rest_router* pada *ryu controller* yang dapat menjalankan static routing tanpa VLAN dan dengan VLAN supaya terlihat perbedaan antara yang menggunakan VLAN dan tidak. Hasil pengujian menunjukkan bahwa topologi linear 3 switch baik 30 host atau 60 host dapat bekerja dengan baik. Berdasarkan pengujian parameter Qos *Throughput, Packet Loss,* dan *Jitter* keseluruhan topologi jaringan dengan VLAN kurang baik dibandingkan tanpa VLAN dan mendapat hasil 222,9 Mbps, 0%, 0,0000636 s, dan 0,0000674 ms. Namun seiring bertambahnya host, VLAN lebih dapat diandalkan.

Kata Kunci: *Software Defined Netwok, Ryu Controller,* Mininet, VLAN, *QoS*

ABSTRACT

PERFORMANCE ANALYSIS OF VLAN BASED ON *SOFTWARE DEFINED NETWORK* WITH *RYU CONTROLLER*

By

M. CHAIRUL ANAM

Software Defined Network is a network architecture that separates the control plane and data plane in network devices, which then makes the control plane independent and programmable. The control plane task is taken by the controller where the controller can control all the rules that exist in all network devices, one of the types of controllers is the ryu controller. VLAN is a network configuration that distributes several different segments to devices connected to the network that can improve network performance. This study aims to determine how the performance of VLAN on Software Defined Network that uses Ryu Controller using simulation methods with 2 scenarios, namely a linear topology of 3 OpenFlow Switch 30 hosts and a linear topology of 3 OpenFlow Switch 60 hosts that are given traffic of 100, 200, 300, and 400 Mb. Testing is done using the rest_router app on the ryu controller which can run static routing without VLAN and with VLAN so that the difference between using VLAN and not can be seen. The test results show that the linear topology of 3 switches, both 30 hosts or 60 hosts, can work well. Based on the Qos Throughput, Packet Loss, and Jitter parameter testing, the overall network topology with VLAN is worse than without VLAN and gets the results of 222.9 Mbps, 0%, 0.0000636 s, and 0.0000674 ms. However, as the number of hosts increases, VLAN is more reliable.

Keywords: Software Defined Netwok, Ryu Controller, Mininet, VLAN, QoS

ANALISA KINERJA VLAN BERBASIS *SOFTWARE DEFINED NETWORK* DENGAN *RYU CONTROLLER*

Oleh
M. Chairul Anam

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Program Studi Teknik Informatika
Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
2023

Judul Skripsi	: **ANALISA KINERJA VLAN BERBASIS *SOFTWARE DEFINED NETWORK* DENGAN *RYU CONTROLLER***
Nama Mahasiswa	: M. Chairul Anam
Nomor Pokok Mahasiswa	: 1715061023
Program Studi	: Teknik Informatika
Jurusan	: Teknik Elektro
Fakultas	: Teknik

MENYETUJUI

1. Komisi Pembimbing

Ir. Ing. Hery Dian Septama, S.T.	**Rio Ariestia P, S.Kom., M.T.I.**
NIP 19850915 200812 1 001	NIP 19860323 201903 1 013

2. Mengetahui

Ketua Jurusan Teknik Elektro	Ketua Program Studi Teknik Informatika
Herlinawati, S.T., M.T.	**Mona Arif Muda, S.T., M.T.**
NIP 19710314 199903 2 001	NIP 19711112 200003 1 002

MENGESAHKAN

1. Tim Penguji

Ketua : **Ir. Ing. Hery Dian Septama, S.T.** .....

Sekretaris : **Rio Ariestia P, S.Kom., M.T.I.** .....

Penguji : **Ir. Gigih Forda Nama, S.T., M.T.I., IPM.** .....

2. Dekan Fakultas Teknik

Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.
NIP 19750928 200112 1 002

Tanggal Lulus Ujian Skripsi : **3 Agustus 2023**

SURAT PERNYATAAN

Saya yang bertanda tangan dibawah ini, menyatakan bahwa skripsi saya yang berjudul "Analisa Kinerja VLAN berbasis Software Defined Network dengan Ryu Controller" dengan ini menyatakan bahwa skripsi saya dibuat oleh saya sendiri. Semua hasil yang tertuang dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung.

Apabila dikemudian hari terbukti bahwa skripsi ini merupakan salinan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai dengan ketentuan hukum atau akademik yang berlaku.

Bandar Lampung, 6 Oktober 2023
Pembuat pernyataan,

M. Chairul Anam
NPM. 1715061023

RIWAYAT HIDUP

Penulis dilahirkan di Poncowarno pada tanggal 16 September 1999. Penulis merupakan anak pertama dari 2 bersaudara dari pasangan Bapak Imam Bajuri dan Ibu Sa'adah.

Penulis memulai jenjang pendidikan dari TK Al-Hidayah Kalirejo, SD Negeri 2 Kalirejo dan lulus pada tahun 2011, SMP Negeri 1 Kalirejo dan lulus pada tahun 2014, SMK Al-Hikmah Kalirejo dan lulus pada tahun 2017 dan ditahun yang sama diterima pada Program Studi Teknik Informatika Universitas Lampung melalui jalur SBMPTN.

Selama menjalani proses perkuliahan secara aktif, penulis sempat mengikuti Himpunan Mahasiswa Teknik Elektro (HIMATRO) sebagai anggota Divisi Penelitian dan Pengembangan pada Periode 2018. Pada organisasi luar kampus, penulis juga mengikuti komunitas Gerakan Digital Ekosistem Nusantara (GRADIEN) yang menambah wawasan penulis di bidang teknologi informasi. Selain proses perkuliahan, penulis juga pernah melaksanakan kerja praktik di Dinas Komunikasi Informatika dan Statistik Provinsi Lampung.

Prestasi yang pernah didapat penulis antara lain adalah sebagai penerima Digitalent Scholarship yang diselenggarakan oleh Kominfo dengan mitra yaitu, Cloud Computing Alibaba Cloud tahun 2021, Cloud Computing AWS tahun 2022, IT Support Google tahun 2022, dan Cyber Security Technician EC-Council. Dari beasiswa pelatihan tersebut penulis mendapat sertifikasi ACA & ACP dari Alibaba Cloud, IT Support dari Google, dan CCT dari EC-Council. Penulis memiliki minat di bidang Teknologi Informasi seperti *Network* dan *Security* juga *backend developer*.

MOTTO

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya. Dia mendapat (pahala) dari (kebajikan) yang dikerjakannya dan dia mendapat (siksa) dari (kejahatan) yang diperbuatnya."

(Al-Baqarah: 286)

"KEEP GOING"
(Diri Sendiri)

"MMR IS JUST A NUMBER"
(Danylo Ishutin)

"HIDUP INI TIDAK ADA ADIL KAWAN, JADI BIASAKANLAH DIRIMU"
(PATRICK STAR)

Sujud Syukur kupersembahkan kepada Allah سُبْحَانَهُ وَ تَعَالَى , Tuhan Yang Maha Esa dan Maha Besar. Berkat limpahan rahmat-Mu saya bisa menjadi pribadi yang bertaqwa, beriman, dan berilmu. Semoga dengan keberhasilan yang telah dicapai ini saya dapat menuju masa depan yang lebih baik dan dapat menggapai cita-cita serta selalu berada di jalan-Mu.

KUPERSEMBAHKAN KARYA ILMIAH INI
TERUNTUK:

“Ibunda Sa’adah dan Ayahanda Imam Bajuri atas dukungan dan kasih sayang yang diberikan mulai dari saya ada di dunia ini sampai saya sudah besar seperti sekarang ini. Terima kasih kepada Ibu dan Ayah atas doa yang tak henti-hentinya dipanjatkan serta pengorbanan yang tak terhitung nilainya. Semoga dengan ilmu dan cita-cita yang saya dapatkan kelak akan menjadi amal jariyah bagi Ibu dan Ayah”

“Terima Kasih untuk adikku Azizatun Nissa yang selalu menemani ibu saat aku tidak dirumah. Semoga kau kelak menjadi pribadi yang lebih sukses dari kakak-kakakmu”

“Terima Kasih kepada dosen pembimbing saya pak hery dan pakri karena telah dengan sabar memberikan arahan dan membimbing saya dalam mengerjakan penelitian ini. Terima Kasih kepada semua pihak yang turut serta mendukung dan membantu penelitian ini sampai akhir.”

“Terima Kasih kepada teman-teman Teknik Informatika 2017 yang telah menemani dan membantu saya selama perkuliahan di kampus tercinta Universitas Lampung. Terima Kasih atas kenangan yang kalian berikan selama perkuliahan mulai dari proses pengenalan kampus sampai dengan akhir semester. Semoga kelak kita semua akan menjadi orang-orang yang sukses.”

“Terima Kasih kepada seseorang yang selalu mendoakan saya dimanapun kamu berada. Semoga Allah selalu memberikan kita kebahagiaan di apapun jalan yang kita pilih kedepannya. Dan bertemu di waktu yang tepat.”

SANWACANA

Puji syukur penulis panjatkan kepada Allah سُبْحَانَهُ وَ تَعَالَى, yang telah memberikan karunia serta ridho-Nya sehingga penulis dapat melaksanakan dan menyelesaikan penelitian ini yang berjudul "Analisa Kinerja VLAN Berbasis *Software Defined Network* dengan *RYU Controller*" Penelitian ini merupakan salah satu sarat untuk menyelesaikan kurikulum mata kuliah penelitian skripsi pada Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung.

Pelaksanaan penelitian ini penulis mendapatkan bantuan, bimbingan serta pengarahan dari berbagai pihak. Maka dari itu, penulis mengucapkan terima kasih sebanyak-banyaknya kepada:

1. Allah سُبْحَانَهُ وَ تَعَالَى yang senantiasa memberikan kemudahan dan kelancaran kepada penulis serta Rasulullah Muhammad صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ yang telah menjadi suri tauladan selama penelitian berlangsung;
2. Ibu dan Ayah serta keluarga penulis yang selalu memberikan motivasi dan dukungan kepada penulis;
3. Bapak Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik Universitas Lampung;
4. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Lampung;
5. Bapak Mona Arif Muda, S.T., M.T. selaku Ketua Program Studi Teknik Informatika Universitas Lampung dan telah membantu proses kelancaran pengerjaan penelitian selaku Pembimbing Akademik yang telah memberikan bimbingan selama perkuliahan disetiap semester;

6. Bapak Ing. Hery Dian Septama, S.T., selaku Pembimbing Utama penelitian yang selalu meluangkan waktunya untuk memberikan bimbingan dan dukungan;
7. Bapak Rio Ariestia P, S.Kom., M.T.I selaku Pembimbing Pendamping penelitian yang selalu memberikan motivasi dan memberikan bimbingan kepada penulis untuk menjadi lebih baik;
8. Mbak Rika selaku Admin Program Studi Teknik Informatika yang telah memberikan bantuan dalam proses administrasi penelitian;
9. GRADIEN selaku komunitas sekaligus wadah penulis dalam melakukan pembelajaran diluar perkuliahan;
10. Teman-teman Teknik Informatika 2017 yang selalu mendukung penulis;
11. Semua pihak yang turut serta dalam membantu menyelesaikan penelitian dan tidak mungkin penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penulisan laporan penelitian ini masih bisa disempurnakan kembali. Oleh karena itu, penulis mengharapkan saran dan kritik yang bersifat membangun dari para pembaca. Penulis berharap laporan skripsi ini dapat bermanfaat bagi banyak pihak.

Bandar Lampung, 6 Oktober 2023

Penulis,

M. Chairul Anam

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

I. PENDAHULUAN

1.1 Latar Belakang

Dengan adanya peningkatan kebutuhan untuk jaringan komputer dan perangkat yang dibutuhkan dalam berbagai bidang, yang kita ketahui juga pertumbuhannya menjadi semakin meningkat pesat. Hal tersebut telah mempengaruhi kebutuhan akan kinerja jaringan yang lebih baik dari sebelumnya, dikarenakan tujuan utama dan juga penambahan konfigurasi jaringan yang menjadi semakin besar. Konfigurasi tersebut juga menjadi lebih kompleks dan melakukan kontrol dalam sebuah jaringan pun menjadi semakin sulit. Hal tersebut menyebabkan jaringan tidak fleksibel dan sulit untuk diatur [1].

Software Defined Network (SDN) merupakan sebuah konsep baru yang digunakan untuk melakukan perancangan, implementasi dan juga manajemen kebutuhan dalam jaringan komputer. Perangkat jaringan tradisional seperti switch dan router dapat dibagi menjadi 3 logical plane yang berbeda, yaitu data plane, *control plane*, dan *management plane*. *Data plane* tersebut megacu pada perangkat keras yang meneruskan paket, sedangkan *control plane* mengacu kepada bagian yang mengimplementasikan protokol dalam proses routing. Biasanya, dalam perangkat jaringan *control plane* akan diimplementasikan pada *proprietary firmware* yang dikembangkan oleh vendor perlengkapan. *Management plane* akan digunakan untuk mengawasi dan mengontrol fungsi.

SDN merupakan sebuah arsitektur jaringan yang sedang berkembang yang memisahkan antara *data plane* dan *control plane* dalam perangkat jaringan yang kemudian membuat *control plane* menjadi independent dan dapat diprogram. Pemisahan yang dilakukan anatara *data plane* dan *control plane* menjadikannya abstraksi dari infrastruktur jaringan dan aplikasi yang membuat jaringan menjadi sebuah keberadaan virtual [2].

Controller merupakan aplikasi yang mengatur *flow* yang terjadi pada jaringan SDN, yang dapat melakukan proses *intelligence networking*. *Controller* sendiri biasanya dijalankan pada protocol seperti *OpenFlow*. Kemudian, paket yang telah dikirimkan dapat diketahui tujuannya. Lalu lintas data yang terjadi dalam jaringan SDN seperti penerusan paket data, tindakan yang diperlukan untuk paket, dan bahkan mengirim aturan baru yang disesuaikan supaya dapat menangani paket data dikemudian hari juga diatur oleh controller[3].

Salah satu *controller* dengan sumber terbuka yang paling sering digunakan adalah *ryu controller*. Controller ryu juga memiliki built-in function yang sudah berada dalam *package* saat diinstal pada sistem operasi *linux* seperti *layer 2 switch, layer 3 switch, router,* dll. Pada penelitian ini akan dilakukan analisa kinerja VLAN pada *Ryu controller* dengan menggunakan simulasi dalam emulator mininet menggunakan topologi dari jaringan yang telah ditentukan untuk mengetahui pengaruh VLAN dalam penerapannya pada *Software Defined Network* dan kemudian performa tersebut akan dinilai berdasarkan standar TIPHON.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah bagaimana merancang dan juga mengimplementasikan static routing tanpa VLAN dan juga dengan VLAN pada topologi yang dibuat pada *Software Defined Network* menggunakan *RYU controller*. Kemudian, melakukan analisis kinerja pada kedua skenario dan mengevaluasi hasil dari skenario tersebut berdasarkan standar TIPHON.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Melakukan analisis terhadap kinerja dari penggunaan VLAN pada *RYU controller* berdasarkan standar TIPHON.

2. Merancang dan mengimplementasikan static routing tanpa VLAN dan dengan VLAN pada *Software Defined Network* menggunakan simulator mininet yang divisualisasikan oleh MiniNAM.

1.4 Manfaat Penelitian

Dalam penelitian ini, diharapkan dapat memberikan manfaat dan pandangan baru mengenai *Software Defined Network* dan pengaruh penerapan VLAN didalamnya yang juga bisa dijadikan juga sebagai salah satu pertimbangan untuk melakukan perubahan infrastruktur jaringan yang semula konvensional menjadi infrastruktur yang berbasis *Software Defined Network*.

1.5 Batasan Masalah

Pada penelitian ini memiliki batasan seperti sebagai berikut:

1. Infrastruktur SDN yang digunakan menggunakan *RYU controller*.
2. Topologi yang digunakan adalah topologi linear dengan 3 OVSSwitch dengan 30 host dan juga 60 host.
3. Protokol pengujian yang digunakan adalah UDP.

1.6 Sistematika Penulisan

Sistematika penulisan yang diterapkan dalam penulisan skripsi ini terbagi menjadi beberapa bab, diantaranya:

BAB I: PENDAHULUAN

Dalam bab ini membahas secara umum mengenai latar belakang, tujuan, manfaat, batasan, dan rumusan masalah dalam penelitian yang dilakukan terkait dengan kinerja dari *RYU controller*.

BAB II: TINJAUAN PUSTAKA

Dalam bab ini berisi mengenai teori-teori yang menjadi dasar dalam penelitian dan juga beberapa sumber referensi yang berasal dari penelitian yang sudah ada agar lebih mudah dalam memahami analisis kinerja *Software Defined Network*.

BAB III: METODOLOGI PENELITIAN

Dalam bab ini membahas mengenai metode penelitian yang digunakan dalam melakukan analisis kinerja *Software Defined Network*.

BAB IV: HASIL DAN PEMBAHASAN

Dalam bab ini berisi tentang hasil dan juga pembahasan yang didapatkan melalui penelitian.

DAFTAR PUSTAKA

LAMPIRAN

II. TINJAUAN PUSTAKA

2.1 Jaringan Komputer

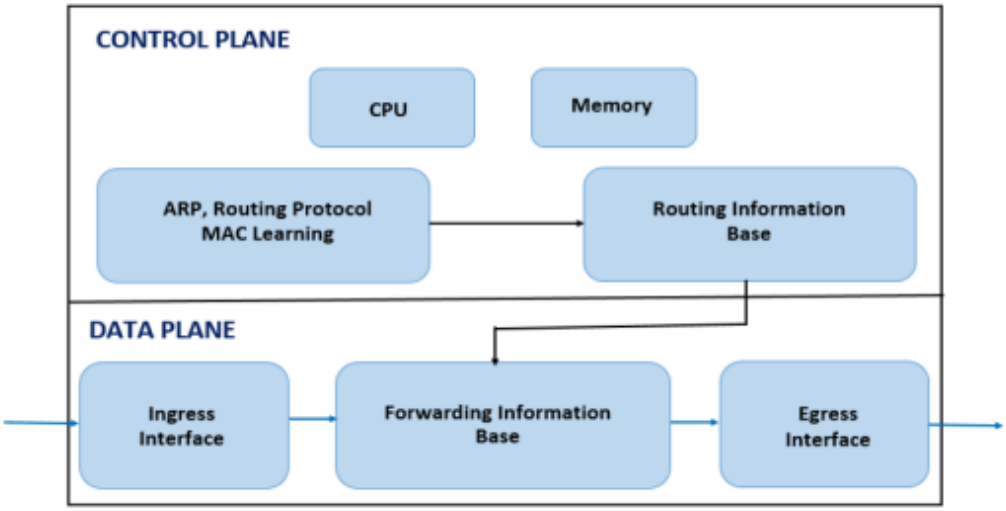
Jaringan komputer merupakan interkoneksi yang terjadi antara 2 buah komputer *autonomous* atau lebih yang dihubungkan melalui media transmisi kabel ataupun nirkabel. Autonomous sendiri merupakan hal dimana komputer tidak memiliki wewenang untuk dapat mengontrol komputer lain dengan akses penuh, yang kemudian dapat membuat komputer lainnya, *restart, shutdown*, kehilangan file atau data maupun kerusakan dalam sistem [4].

2.2 *Software Defined Network*

Software Defined Network (SDN) merupakan sebuah konsep baru yang digunakan dalam merancang, implementasi dan juga manajemen kebutuhan dalam jaringan komputer. Perangkat jaringan tradisional seperti switch dan router dapat dibagi menjadi 3 logical plane yang berbeda, yaitu data plane, *control plane*, dan *management plane*. *Data plane* tersebut megacu pada perangkat keras yang meneruskan paket, sedangkan *control plane* mengacu kepada bagian yang mengimplementasikan protokol dalam proses routing. Biasanya, dalam perangkat jaringan *control plane* akan diimplementasikan pada *proprietary firmware* yang dikembangkan oleh vendor perlengkapan. *Management plane* akan digunakan untuk mengawasi dan mengontrol fungsi.

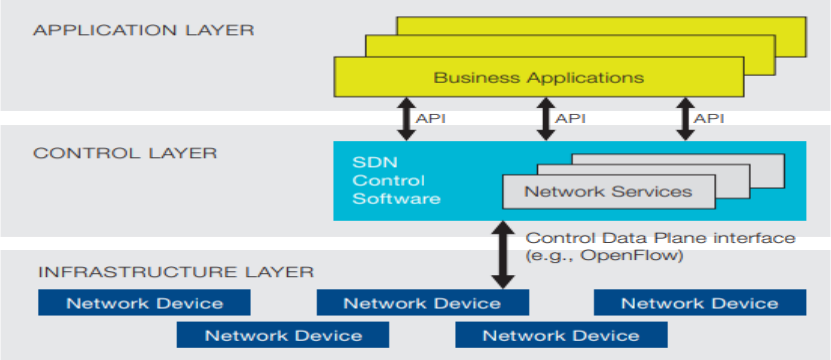
SDN merupakan sebuah arsitektur jaringan yang sedang berkembang yang memisahkan antara data plane dan control plane dalam perangkat jaringan yang

kemudian membuat control plane menjadi independent dan dapat diprogram. Pemisahan yang dilakukan anatara data plane dan control plane menjadikannya abstraksi dari infrastruktur jaringan dan aplikasi yang membuat jaringan menjadi sebuah keberadaan virtual. Berikut merupakan bagaimana data plane dan control plane dalam perangkat jaringan tradisional [1].



Gambar 2.1 Data dan control plane tradisional

Menurut jurnal Open Network Fondation (ONF) yang berjudul "*Software Defined Network*ing: *The New Norm For Networks*", SDN didefinisikan sebagai sebuah arsitektur jaringan yang memisahkan fungsi untuk mengontrol dan meneruskan paket yang dapat diprogram secara langsung, jadi operator maupun administrator jaringan dapat mengkonfigurasi jaringan dengan lebih mudah dan terpusat meskipun antara ribuan perangkat [5]

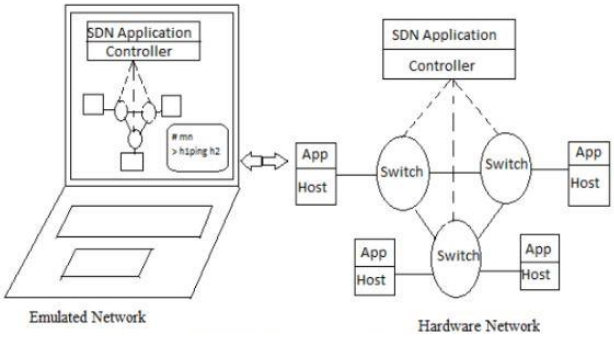


Gambar 2.2 Arsitektur SDN [6]

2.3 Mininet

Mininet merupakan sebuah simulator jaringan yang dapat membuat jaringan dari virtual host, switch, *controller*, dan link. Mininet berjalan pada standar dari perangkat lunak jaringan linux, dan setiap switch mendukung untuk penggunaan openflow dengan tingkat fleksibilitas yang tinggi untuk kustom routing dan *software defined network* (SDN). Adapun keuntungan dalam menggunakan mininet adalah sebagai berikut:

1. Penyediaan textbed yang sederhana dan tidak terlalu mahal dalam pengembangan aplikasi openflow.
2. Memungkinkan multiple concurrent developer untuk bekerja secara independent dalam topologi yang sama.
3. Mendukung system-level regression test, yang mudah diulang dan dikemas.
4. Memungkinkan complex topology testing tanpa perlu menghubungkan dengan jaringan fisik.
5. Menggunakan CLI yang termasuk dalam kebutuhan topologi dan openflow untuk melakukan debug ataupun menjalakan tes jaringan dengan skala besar.
6. Mendukung arbitrary custom topology dan termasuk set dasar pada parametrized topology.
7. Dapat digunakan tanpa pemrograman, tapi
8. Juga menyediakan kemudahan dan skalabilitas python API dalam eksperimen ataupun pembuatan jaringan [7].



Gambar 2.3 Emulasi perangkat jaringan menggunakan mininet [8]

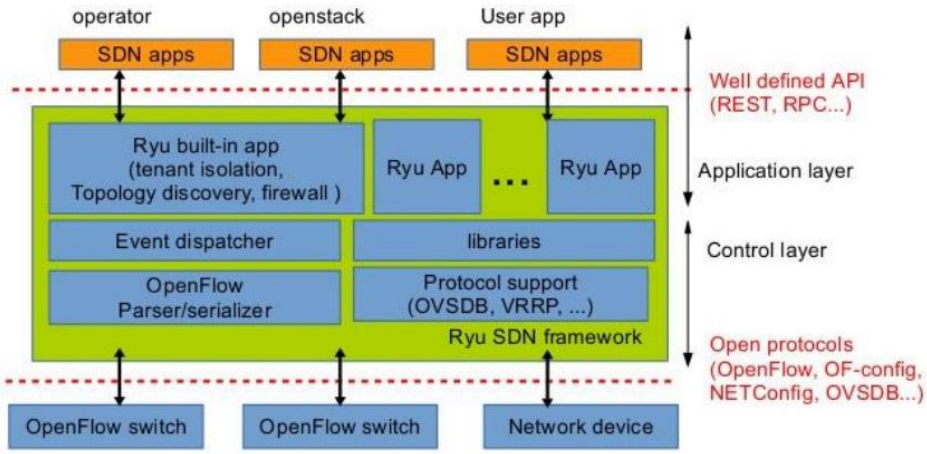
Mininet merupakan kombinasi dari fitur-fitur terbaik yang ada pada emulator, perangkat keras testbed, dan simulator-simulator yang telah ada sebelumnya. Jaringan berbasis mininet tidak dapat (untuk saat ini) melebihi CPU atau bandwidth yang tersedia di satu server. Mininet tidak dapat (untuk saat ini) menjalankan switch atau aplikasi OpenFlow yang tidak kompatibel dengan Linux namun saat ini hal itu bukanlah sebuah kebutuhan yang penting.

2.4 *RYU Controller*

RYU merupakan salah satu komponen dasar dari *software defined network* (SDN). RYU meneyediakan komponen perangkat lunak dengan API yang terdefinisi dengan baik yang mudah dikembangkan dalam membuat sebuah manajemen jaringan dan pengontrolan aplikasi. RYU mendukung berbagai macam protocol yang dapat digunakan untuk memanajemen perangkat jaringan, beberapa contohnya adalah OpenFlow, Netconf, OF-config, dll. Ryu mendukung OpenFlow dengan versi 1.0, 1.2, 1.4, 1.5 dan nicira extension [9].

Ryu merupakan controller opensource yang dinaungi oleh Apache 2.0 license, yang dtulis secara utuh menggunakan python. Ryu didukung dan dikerahkan oleh NTT cloud data centers. Source code utamanya dapat ditemukan di github, disediakan dan juga didukung oleh Ryu community. Juga mendukung NETCONF dan OF-config network management protocol yang sebaik openflow. Mengingat kompatibilitasnya, OpenFlow switch, Hewlett Packard, IBM, dan NEC

telah diuji dan disertifikasi dengan *Ryu controller*. Ryu juga mendukung protokol OpenFlow hingga versi terbaru yaitu 1.5 [10].



Gambar 2.4 Arsitektur Ryu SDN controller

Ryu juga membuat OpenFlow packet, memanajemen event terkait dengan paket yang masuk dan keluar sama seperti SDN *controller* lainnya. Ryu memiliki library yang melimpah yang mendukung operasi pemrosesan paket. Mengenai dukungan untuk *southbound protocol*, Ryu bekerja sama dengan protokol seperti XFlow (Netflow dan Sflow), OF-Config, NETCONF, Open vSwitch Database Management Protocol (OVSDB)dll. VLAN, GRE dan VLAN, dll juga didukung oleh *Ryu Packet Libraries.* Didalam ryu juga terdapat beberapa *app* yang dapat langsung diterapkan pada arsitektur SDN seperti berikut ini.

Tabel 2.1 Modul/App yang ada pada *RYU Controller*

Nama	Fungsi
bmpstation.py	BMP
cbench.py	Openflow 1.0 Responder
example_switch_13.py	Menerapkan simple switch
ofctl_rest.py	OpenFlow Controller menggunakan REST
rest_conf_switch.py	Konfigurasi REST Switch
rest_firewall.py	Menerapkan firewall
rest_qos.py	Mengatur qos
rest_router.py	Menerapkan simple router
rest_topology.py	Topologi REST
rest_vtep.py	Menerapkan VTEP untuk EVPN VXLA

Nama	Fungsi
simple_monitor_13.py	Memonitor jaringan
simple_switch.py	Simple switch menggunakan OpenFlow 1.0
simple_switch_12.py	Simple switch menggunakan OpenFlow 1.2
simple_switch_13.py	Simple switch menggunakan OpenFlow 1.3
simple_switch_14.py	Simple switch menggunakan OpenFlow 1.4
simple_switch_15.py	Simple switch menggunakan OpenFlow 1.5
simple_switch_igmp.py	Simple switch IGMP OpenFlow 1.0
simple_switch_igmp_13.py	Simple switch IGMP OpenFlow 1.3
simple_switch_lacp.py	Simple switch IACP OpenFlow 1.0
simple_switch_lacp_13.py	Simple switch IACP OpenFlow 1.3
simple_switch_rest_13.py	Simple switch REST OpenFlow 1.3
simple_switch_snort.py	Simple Switch snort
simple_switch_stp.py	Simple switch spanning tree OpenFlow 1.0
simple_switch_stp_13.py	Simple switch spanning tree OpenFlow 1.3
simple_switch_websocket_13.py	Simple switch dengan web socket OF 1.3
ws_topology.py	Topologi Web Socket

2.5 MiniNAM

MiniNAM merupakan tools berbasis GUI yang ditulis menggunakan Python Tkinter. MiniNAM menyediakan *real-time animation* dari jaringan apapun yang dibuat oleh emulator mininet. MiniNAM juga memiliki komponen yang yang diperlukan untuk memulai, memvisualisasikan, dan memodifikasi alur dari jaringan mininet secara *real-time* [10][11].

2.6 Topologi

Topologi merupakan sebuah langkah-langkah dalam penghubungan antar komputer yang dapat disesuaikan dengan kebutuhan kedalam sebuah jaringan. Topologi memiliki berbagai macam bentuk, seperti sebagai berikut[12]:

1. Topologi Ring

   Pada topologi ring, tiap client akan dihubungkan dengan 2 client lainnya yang berdekatan. Kemudian, paket akan dikirimkan dengan alamat tujuan dan

dikirim kedalam cincin. Paket akan dikirimkan berputar didalam cincin hingga client tujuan ditemukan, dan paket diterima oleh client bersangkutan.

2. Topologi Star

   Topologi ring menghubungkan antara client dan server dalam sebuah pusat yang biasanya pusat tersebut adalah hub. Tiap komputer yang melakukan komunikasi akan melewatkan paket data melalui hub. Jadi, kinerja jaringan tergantung dengan bagaiman kinerja dari hub tersebut. Jika hub mengalami error maka keseluruhan jaringan akan menjadi terhambat.

3. Topologi Bus

   Setiap komputer akan terhubung kedalam jalur utama. Topologi ini memiliki keunggulan yaitu tiap komputer/client tambahan bisa langsung terhubung ke jaringan tanpa melakukan konfigurasi ulang dalam jaringan. Topologi ini biasanya digunakan dalam jaringan skala dengan relative pendek namun memiliki banyak komputer.

4. Topologi Tree

   Topologi ini merupakan gabungan antara topologis star dan bus, pada topologi ini backbone-cablenya akan membentuk topologi bus yang menghubungkan node-node jaringan kecil lainnya yang menggunakan topologi star.

5. Topologi Mesh

   Dalam topologi mesh, menerapkan komunikasi data dengan central secara penuh. Jumlah saluran yang dimiliki harus cukup untuk membentuk jaringan mesh dengan anturan jumlah central dikurangi 1 (**n-1, n=jumlah central**). Tingkat kompleksitas jaringan akan meningkat mengikuti jumlah central yang terhubung. Karena hal tersebut jaringan ini dikategorikan kurang ekonomis dan relative mahal dalam pengoperasiannya. Jika dilihat dari bentuk, topologi mesh ini sebenarnya merupakan gabungan antara topologi ring dan star [12].

2.7 *IP address*

IP address atau yang biasa dikenal dengan kode pengenal pada komputer dalam sebuah jaringan/internet merupakan salah satu komponen vital. Setiap perangkat yang terhubung dalam sebuah jaringan/ internet harus memiliki IP pada setiap interfacenya dan memliki keunikan sendiri yang tidak memperboleh lebih dari satu perangkat memiliki IP yang sama. IP sendiri terdiri dari bilangan biner dengan Panjang 32bit sebagai bentuk identifikasi host dalam sebuah jaringan. Paket yang berisi data akan dibawa oleh IP dari pengirim data, dan IP ke perangkat yang dituju, kemudia dikirim ke dalam jaringan tersebut [12].

2.8 Ubuntu

Ubuntu adalah sistem operasi Linux yang lengkap, dapat digunakan dan dimodifikasi secara bebas dukungan komunitas dan profesional. Komunitas Ubuntu dibangun berdasarkan ide-ide yang diabadikan dalam Manifesto Ubuntu, bahwa perangkat lunak harus tersedia secara gratis, perangkat lunak harus dapat digunakan oleh orang-orang dalam bahasa lokal mereka dan terlepas dari kecacatannya, dan orang-orang harus mempunyai kebebasan untuk menyesuaikan dan mengubah perangkat lunak mereka dengan cara apa pun yang mereka inginkan [13].

- Ubuntu akan selalu gratis, tidak ada biaya tambahan untuk *"enterprise edition"*, ubuntu menyediakan karya terbaiknya untuk semua orang dengan persyaratan Gratis yang sama.
- Ubuntu menyertakan yang terbaik dalam terjemahan dan infrastruktur aksesibilitas yang ditawarkan oleh komunitas Perangkat Lunak Bebas, untuk membuat Ubuntu dapat digunakan oleh sebanyak mungkin orang.
- Ubuntu dikirimkan dalam siklus rilis yang stabil dan teratur; rilis baru akan dikirimkan setiap enam bulan. Setiap dua tahun genap rilis Ubuntu long term support (LTS) akan tersedia, yang didukung selama 5 tahun. Rilis Ubuntu di antaranya (dikenal sebagai rilis pengembangan atau non-LTS) masing-masing didukung selama 9 bulan.

- Ubuntu sepenuhnya berkomitmen pada prinsip-prinsip pengembangan perangkat lunak sumber terbuka; kami mendorong orang untuk menggunakan perangkat lunak sumber terbuka, meningkatkannya, dan menyebarkannya.

Ubuntu cocok untuk penggunaan desktop dan server. Rilis Ubuntu saat ini mendukung Intel x86 (PC yang kompatibel dengan IBM), AMD64 (x86-64), ARMv7, ARMv8 (ARM64), IBM POWER8/POWER9 (ppc64el), IBM Z zEC12/zEC13/z14 dan IBM LinuxONE Rockhopper I+II / Kaisar I+II (s390x). Ubuntu sendiri menyediakan ribuan perangkat lunak, dimulai dengan kernel Linux versi 5.4 dan GNOME 3.28, dan mencakup setiap aplikasi desktop standar seperti aplikasi pengolah kata dan spreadsheet hingga aplikasi akses internet, perangkat lunak server web, perangkat lunak email, bahasa dan alat pemrograman, dan juga permainan [13].

2.9 VMware

VMware adalah salah satu perangkat lunak yang digunakan sebagai *virtual machine*. Kegunaan dari VMware sendiri adalah menjalankan lebih dari satu sistem operasi dalam sebuah pernagkat keras dan juga menjalankan aplikasi yang digunakan pada sistem operasi yang lain. VMware dapat menjalankan lebih dari satu sistem operasi bersamaan dalam satu perangkat PC secara Bersama. Hal tersebut bisa dilakukan tanpa adanya pemartisian dan boot ulang. VMWare sendiri terdiri dari 3 jenis, yaitu:

1). VMWare Workstation merupakan sebuah perangkat lunak *virtual machine* yang kompatibel dengan komputer Intel x86. Perangkat lunak ini dapat membuat pemakai membuat satu atau lebih *virtual machine* dan mejalankannya secara bersamaan. Setiap *virtual machine* pun dapat menjalankan guest operating systemnya sendiri seperti Linux, Windows, BSD dan lain-lain. Tetapi perangkat lunak tersebut tidak bisa menjalankan *virtual machine* yang dibuat oleh produk *virtual machine* lain yang sejenis.

2). VMWare Server memiliki sistem kerja yang kurang lebih sama dengan VMWare Workstation. Namun, VMWare Server mempunyai kelebihan yaitu dapat menjalankan *virtual machine* yang dibuat oleh produk *virtual machine* lain. VMWare Server juga bisa mengoperasikan *virtual machine* buatan dari Microsoft Virtual PC.

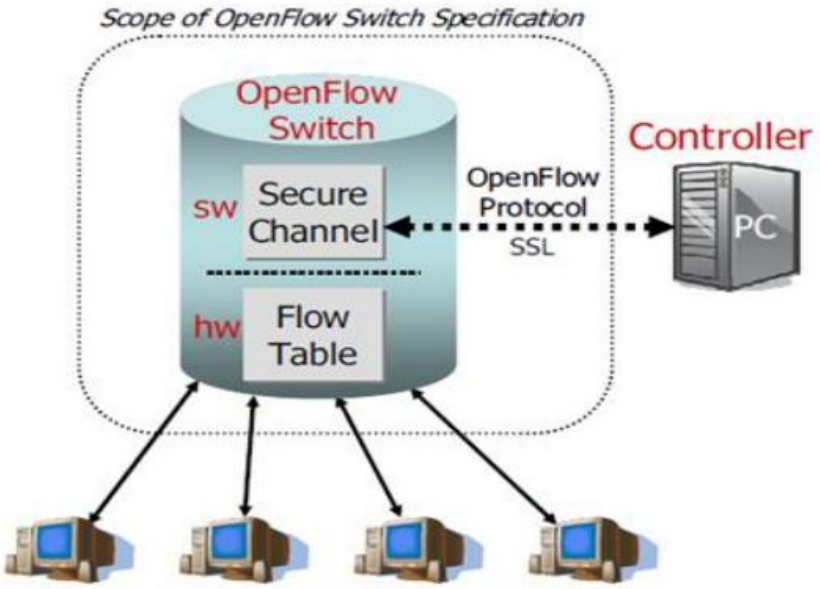
3). VMWare Player merupakan perangkat lunak yang digunakan untuk menjalankan *virtual machine* yang dibuat oleh produk *virtual machine* lainnya. Tetapi perangkat lunak ini tidak bisa membuat *virtual machine*nya sendiri [14].

2.10 *Openflow Protocol*

OpenFlow protocol merupakan protocol yang paling popular dan diterima secara luas untuk penggunaan dalam *southbound Application Programming Interface. OpenFlow protocol* bermaksud untuj menyediakan akses ke data plane dari switch. Ini dilakukan dengan menentukan bahasa yang dapat dikenali dan digunakan oleh switch untuk memperbarui *forwarding tables*. OpenFlow merupakan bahasa yang digunakan untuk mendefinisikan secara umum karakteristik dari arus lalu lintas tertentu dan rangkaian tindakan yang akan dilakukan ketika switch menemukan paket yang cocok dengan karakteristik tersebut.

Mekanisme aktual yang digunakan untuk memprogram aliran kedalam switch bervariasi tergantung pada vendor perangkat kerasnya. Sebagai gantinya, OpenFlow menyediakan cara untuk menggambarkan keadaan aliran yang diinginkan dalam agen yang berjalan secara lokal pada perangkat forwading. Semua switch yang mengaktifkan OpenFlow akan memiliki Agen OpenFlow yang akan menginterpretasikan perintah OpenFlow. Spesisifikasi OpenFlow juga mencakup penggunaan untuk *OpenFlow controller* yang dapat dikendalikan dan terletak pada control plane untuk memodifikasi pada informasi. OpenFlow agent dilengkapi dengan informasi aliran yang diprogram kedalamnya oleh *controller* yang bertindak bertindak seperti control plane pada switch tradisional. Satu satunya perbedaan adalah OpenFlow tidak diharuskan menjalankan protokol routing, ataupun

membuat keputusan secara lokal. Semua keputusan akan dibuat oleh OpenFlow *controller* jarak jauh dan OpenFlow agent akan menyimpan entri OpenFlow ini dan memasukkannya ke dalam flow table di perangkat keras. Gambar berikut menunjukkan OpenFlow switch yang ideal dimana tabel aliran dikendalikan oleh OpenFlow *controller* jarak jauh [15].



Gambar 2.5 OpenFlow switch ideal

2.11 *Quality of Service (Qos)*

Quality of service merupakan salah satu metode pendukung dalam sebuah penilaian ataupun evaluasi dalam menunjukkan kinerja dari suatu service yang didalamnya terdapat atribut kerja yang berfungsi sebagai dasar penilaian atau evaluasi *service*. Kemudian, dalam *Quality of Service (Qos)* juga memiliki parameter diantaranya sebagai berikut [16]:

a. Bandwidth

Bandwidth merupakan luas atau lebar dari cakupan frekuensi yang digunakan oleh sinyal dalam medium transmisi. Bandwidth juga sering bisa dikatakan sebagai kecepatan transfer data atau *transfer rate* yaitu jumlah data dibawa dari sebuah titik menuju je titik yang lain dalam satu waktu .(biasanya dalam satuan detik).

b. *Throughput*

Throughput merupakan kemampuan jaringan untuk dapat mengirim data. *Throughput* berkaitan erat dengan bandwidth, Jika bandwidth bersifat teta sedangkan *throughput* bersifat dinamis tergantung dari trafik yang sedang terjadi dalam suatu jaringan.

Throughput dan juga bandwidth sendiri juga dipengaruhi oleh pernagkat jaringan, tipe transfer data, jumlah pengguna, topologi yang digunakan, spesifikasi komputer dari client, spesifikasi server, dan lain sebagainya. *Throughput* juga bisa disebut kecepatan transfer data yang efektif dalam ukuran bps (*bit per second*). *Throughput* adalah jumlah dari kedatangan paket yang sukses diawasi pada destinasi dalam interval waktu tertenty kemudian dibagi dengan durasi interval waktu. Rumus dari *throughput* adalah sebagai berikut:

$$\textbf{Throughput :} \quad \frac{\textit{Packed received (kb)}}{\textit{Time transmitted (s)}}$$

Gambar 2.6 Rumus *throughput*

Adapun standar *Throughput* menurut TIPHON:

Tabel 2.2 Standar *throughput* TIPHON

Kategori *Throughput*	***Throughput***	***Indeks***
Bad	0 – 338 kbps	0
Poor	338 – 700 kbps	1
Fair	700 – 1200 kbps	2
Good	1200 kbps – 2,1 Mbps	3
Excelent	>2,1 Mbps	4

c. *Jitter*

Jitter merupakan variasi atau perubahan latency dari *delay* ataupun perbedaan kedatangan dari suatu paket. *Jitter* dapat dianggap juga sebagai gangguan yang disebabkan oleh adanya perubahan sinyal dalam referensi posisi

waktu. *Jitter* juga dapat menyebabkan kehilangan data, terlebih pada pengiriman data dengan kecepatan tinggi. Hal – hal yang dapat menyebabkan *jitter* antaa lain:

- Antrian pengolahan data yang Panjang.
- Peningkatan trafik mendadak yang mengakibatkan bandwidth menyempit dan menjadi antrian dan,
- Kemampuan tiap node dalam menerima dan mengirim paket juga dapat mengakibatkan *jitter*.

Jitter merupakan parameter yang mewakili QoS audio, atau ukuran variasi penundaan paket berturut-turut dalam arus lalu lintas. Dengan diketahuinya *jitter* yang dihasilkan dalam proses akses internet, maka dapat diketahui kualitas dari suatu perangkat yang digunakan dalam menghitung nilai *jitter* rata-rata yang dihasilkan. Berikut merupakan standar *jitter* menurut TIPHON:

$$\textbf{\textit{Jitter}}: \quad \frac{Total\ Variasi\ Delay}{(Jumlah\ Paket - 1)}$$

Gambar 2.7 Rumus *Jitter*

Tabel 2.3 Standar *Jitter* TIPHON

Kategori *Jitter*	***Jitter***	**Indeks**
Poor	125 – 225 ms	1
Fair	75 – 125 ms	2
Good	0 – 75 ms	3
Excelent	0 ms	4

d. *Packet Loss*

Packet loss menggambarakn situasi yang menampilkan jumlah paket yang tidak sampai atau hilang. Paket tersebut hilang disebabkan oleh adanya collision dan juga congestion dalam jaringan. *Packet loss* juga sebagai parameter yang menunjukkan kegagalan transmisi data yang dikirim ketujuan. Ada beberapa kemungkinan yang menyebabkan kegagalan tersbut antara lain:

- Lalu lintas data yang overload.

- Congestion.
- Error yang terjadi pada perangkat keras.
- Penerima gagal menerima karena adanya overflow pada buffer.

Rumus dari *packet loss* adalah sebagai berikut:

$$\textbf{Packet loss} = \frac{(Packet\ transmitted - Packet\ received)}{Packet\ transmitted} X\ 100\%$$

Gambar 2.8 Rumus *packet loss*

Adapun standar *packet loss* menurut TIPHON adalah sebagai berikut:

Tabel 2.4 Standar *Packet Loss* TIPHON

Kategori *Packet Loss*	***Packet loss***	**Indeks**
Poor	>25%	1
Fair	12 – 24%	2
Good	3 – 14%	3
Excelent	0 – 2%	4

e. Latency

Latency merupakan jumlah waktu tunda dari suatu paket yang disebakan oleh proses tranmsisi data dari titik satu ke titik lainnya. Dalam jaringan sendiri, *delay* terdiri dari *delay* packetization, *delay* serialization, *delay* processing, *delay jitter* buffer dan *delay* network. Berikut merupakan standar latency menurut TIPHON antara lain:

Tabel 2.5 Standar Latency TIPHON

Kategori *Latency*	***Latency***	**Indeks**
Poor	>450 s	1

Kategori *Latency*	***Latency***	**Indeks**
Fair	300 - 450 s	2
Good	150 – 300 s	3
Excelent	< 150 s	4

Kemudian berikut ini merupakan rumus untuk mencari nilai *latency(delay)*:

$$\textbf{Rata-rata delay:} \quad \frac{Total\ Delay}{Jumlah\ Paket}$$

Gambar 2.9 Rumus *latency (delay)*

2.12 Wireshark

Wireshark merupakan salah satu aplikasi opensource yang dapat menganalisis paket data dalam jaringan. Perangkat ini juga digunakan untuk memecahkan permasalahan dalam jaringan, analisis, pengembangan protokol komuniaksi, dan edukasi. Dari banyaknya network analyzer yang digunakan oleh Network administrator, wireshark paling sering digunakan dalam proses analisis kinerja dari jaringan dan juga mengontrol lalu lintas data yang terjadi didalamnya dengan menangkap paket data yang lewat dan juga informasi dalam berbagai jenis protokol.

Wireshark sendiri memiliki kekurangan, yaitu kesulitan dalam menangani pendeteksian yang bersifat nirkabel daripada pendeteksian drive jaringan kabel. Wireshark hanya dapat mengenali driver wwireless buatan Microsoft ataupun hanya protokol dari sebuah WLAN. Namun protokol jenis 802.11 tetap dapat dibaca dan djuga dijalankan[17].

2.13 Metode Simulasi

Dalam proses *lifecycle* terdapat berbagai jenis proses pemodelan dan simulasi. Pada metode simulasi ini memiliki hal – hal dasar yang perlu diperhatikan dalam proses simulasi. Namun, dalam proses simulasi dapat dilewatkan untuk bagian problem formulation dan mulai dati tahap berikutnya sampai tahap akhir. Berikut tahapan dalam metode simulasi [18]:

a. *Problem formulation*

Proses simulasi dimulai dengan masalah praktis yang membutuhkan pemecahan atau pemahaman. Mungkin kasus perusahaan kargo yang mencoba mengembangkan strategi baru untuk pengiriman truk atau astronom yang mencoba memahami bagaimana nebula terbentuk. Pada tahap ini, kami harus memahami perilaku sistem bunga (yang dapat berupa alam atau buatan) sistem, ada atau tidak), mengatur operasi sistem sebagai objek dan aktivitas di dalam kerangka eksperimental yang menarik. Kemudian kita perlu menganalisis berbagai alternatif solusi dengan menyelidiki hasil lain yang sudah ada sebelumnya untuk masalah serupa. Solusi yang paling dapat diterima harus dipilih (menghilangkan tahap ini dapat menyebabkan pemilihan yang mahal atau solusi yang salah). Kita juga harus mengidentifikasi variabel input/output dan mengklasifikasikannya menjadi variabel keputusan (dapat dikendalikan) atau parameter (tidak dapat dikendalikan). Jika masalahnya melibatkan analisis kinerja, ini adalah titik di mana kita juga dapat mendefinisikan metrik kinerja (berdasarkan pada variabel keluaran) dan fungsi tujuan (yaitu, kombinasi dari beberapa metrik). Pada tahap ini, kita juga dapat melakukan analisis risiko dan memutuskan apakah akan mengikuti atau membuang proyek

b. *Conceptual Model*

Langkah ini terdiri dari membangun deskripsi tingkat tinggi tentang struktur dan perilaku sistem dan mengidentifikasi semua objek dengan atribut dan antarmukanya. Kita juga harus mendefinisikan apa variabel keadaan, bagaimana mereka terkait, dan mana yang penting untuk belajar. Pada langkah ini, aspek kunci dari persyaratan diungkapkan (jika mungkin, menggunakan formalisme, yang

memperkenalkan tingkat presisi yang lebih tinggi). Selama definisi konseptual model, kita perlu mengungkapkan fitur yang sangat penting (misalnya, kemungkinan ketidakstabilan, kebuntuan, atau kelaparan). Kita juga harus mendokumentasikan informasi yang tidak berfungsi misalnya, kemungkinan perubahan di masa depan, perilaku non-intuitif (atau non-formal), dan hubungannya dengan lingkungan.

c. *Analysis of input/output data*

Pada fase ini, kita harus mempelajari sistem untuk mendapatkan data input/output. Untuk melakukannya, kita harus mengamati dan mengumpulkan atribut-atribut yang dipilih pada fase sebelumnya. Ketika entitas sistem adalah dipelajari, kami mencoba mengaitkannya dengan nilai waktu. Masalah penting lainnya selama ini fase adalah pemilihan ukuran sampel yang valid secara statistik dan format data yang dapat diproses dengan komputer. Akhirnya, kita harus memutuskan atribut mana yang stokastik dan mana yang bersifat deterministik. Dalam beberapa kasus, tidak ada sumber data untuk dikumpulkan (misalnya, untuk sistem yang tidak ada). Dalam kasus tersebut, kita perlu mencoba untuk mendapatkan kumpulan data dari sistem yang serupa (jika: tersedia). Pilihan lainnya adalah dengan menggunakan pendekatan stokastik untuk menyediakan data yang dibutuhkan melalui generasi nomor acak.

d. *Modeling phase*

Dalam fase pemodelan, kita harus membangun representasi rinci dari sistem berdasarkan: model konseptual dan data I/O yang dikumpulkan. Model dibangun dengan mendefinisikan objek, atribut, dan metode menggunakan paradigma yang dipilih. Pada titik ini, model spesifikasi dibuat, termasuk himpunan persamaan yang mendefinisikan perilaku dan strukturnya. Setelah menyelesaikan definisi ini, kita harus mencoba membangun struktur awal model (mungkin menghubungkan sistem variabel dan metrik kinerja), dengan hati-hati menjelaskan asumsi dan penyederhanaan apa pun dan mengumpulkannya ke dalam EF model.

e. *Simulation phase*

Selama tahap simulasi, kita harus memilih mekanisme untuk mengimplementasikan model (dalam kebanyakan kasus menggunakan komputer dan bahasa pemrograman yang memadai dan alat), dan simulasi model dibangun. Selama langkah ini, mungkin perlu untuk mendefinisikan algoritma simulasi dan menerjemahkannya ke dalam program komputer. Dalam fase ini, kita juga harus membangun model dari EF untuk simulasi.

f. *Verification and validation*

Selama langkah sebelumnya, tiga model berbeda dibangun: model konseptual (spesifikasi), model sistem (desain), dan model simulasi (program yang dapat dieksekusi). Kita butuh untuk memverifikasi dan memvalidasi model ini. Verifikasi terkait dengan konsistensi internal antara ketiga model (apakah model diimplementasikan dengan benar?). Validasi difokuskan pada korespondensi antara model dan kenyataan: apakah hasil simulasi konsisten dengan sistem? sedang dianalisis? Apakah kita membangun model yang tepat? Berdasarkan hasil yang diperoleh selama ini fase, model dan implementasinya mungkin perlu perbaikan. Seperti yang akan kita bahas di bagian selanjutnya, proses V&V bukan merupakan fase tertentu dari siklus hidup, tetapi bagian yang tidak terpisahkan darinya. Proses ini harus formal dan harus didokumentasikan dengan benar karena versi model yang lebih baru akan membutuhkan putaran V&V lain, yang sebenarnya merupakan salah satu dari fase paling mahal dalam siklus.

g. *Experimentation*

Kita harus menjalankan model simulasi, mengikuti tujuan yang dinyatakan dalam model konseptual. Selama fase ini, kita harus mengevaluasi output dari simulator, menggunakan korelasi statistic untuk menentukan tingkat presisi untuk metrik kinerja. Fase ini dimulai dengan desain percobaan, menggunakan teknik yang berbeda. Beberapa teknik ini termasuk sensitivitas analisis, optimasi, pengurangan varians (untuk mengoptimalkan hasil dari titik statistik tampilan), dan peringkat dan seleksi (perbandingan dengan sistem alternatif).

h. *Output analysis phase*

Pada fase analisis keluaran, keluaran simulasi dianalisis untuk memahami perilaku sistem. Keluaran ini digunakan untuk memperoleh tanggapan tentang perilaku aslinya sistem. Pada tahap ini, alat visualisasi dapat digunakan untuk membantu proses tersebut. Tujuan dari visualisasi adalah untuk memberikan pemahaman yang lebih dalam tentang sistem nyata yang sedang diselidiki dan untuk membantu dalam mengeksplorasi set besar data numerik yang dihasilkan oleh simulasi.

2.14 *Virtual Local Area Network (*VLAN*)*

2.14.1 Pengertian VLAN

VLAN adalah sebuah *subnetwork* yang dapat mengelompokkan kumpulan perangkat pada jaringan fisik yang terpisah. LAN merupakan sekelompok komputer dan perangkat yang saling berbagi jalur komunikasi melalui koneksi kabel maupun nirkabel dalam letak geografis yang sama. VLAN memudahkan *network administrator* untuk membagi jaringan untuk menyesuaikan dengan kebutuhan fungsional dan juga persyaratan keamanan pada sistem tanpa harus membuat kabel baru atau membuat perubahan besar terhadap infrastruktur jaringannya. VLAN sering digunakan oleh perusahaan besar untuk mempartisi ulang perangkatnya untuk manajemen lalu lintas data yang lebih baik [19].

2.14.2 Tipe VLAN

Beberapa tipe VLAN adalah sebagai berikut[19]:

a. Protokol VLAN

   Protokol VLAN menangani lalu lintas data berdasarkan protokol. Switch akan memisahkan atau meneruskan paket berdasarkan protokol yang keluar dan masuk.

b. Static VLAN

Static VLAN juga bisa disebut dengan *Port-Based* VLAN. VLAN ini memerlukan *network administrator* untuk menetapkan port pada jaringan ke jaringan virtual

c. Dynamic VLAN

VLAN ini mengijinkan *network administrator* untuk mendefinisikan keanggotaan dalam jaringan berdasarkan karakterisik dari perangkat tersebut, contohnya lokasi dari port switch.

2.15 Penelitian Terkait

Berikut merupakan beberapa penelitian terkait yang dijadikan sebagai perbandingan dan juga rujukan mengenai metode yang dipakai didalam penelitian ini.

Penelitian oleh Sajjad A. Madani et al yang berjudul "*Wireless sensor networks: modeling and simulation*" membahas tentang daftar model dan alat simulasi/emulasi yang tersedia untuk jaringan sensor nirkabel. Selanjutnya membahas tentang pengantar dan perspektif sejarah ke bidang *Wireless Sensor Network* dan beberapa aplikasinya. Kemudian, membahas mengenai pemodelan peristiwa diskrit yang berbeda dan metodologi simulasi setelah itu langkah/fase utama dalam studi *Modeling* dan *Simulation* digariskan dengan identifikasi tonggak utama dan diakhiri dengan daftar berbagai model, alat simulasi dan emulasi. Metode simulasi digunakan agar sebelum dilakukan penerapan pada dunia nyata akan dilakukan improvisasi dan pengetesan yang bervariasi yang memungkinkan untuk diterapkan. Metode paling akurat dan diandalkan adalah implementasi secara langsung. Namun, terkadang hal itu tidak mungkin atau membuatnya menjadi lebih sulit. Maka dari itu, metode yang terpilih adalah *modeling* dan simulasi [18]. Sedangkan pada penelitian ini akan menggunakan metode simulasi sebagai metode terpilih untuk membahas mengenai VLAN pada *software defined network* berbasis *ryu controller*.

Penelitian yang berjudul "*Performance evaluation of campus network involving VLAN and broadband multimedia wireless networks using OPNET modeler*" yang dilakukan oleh Dhurgham Abdulridha Jawad Al-Khaffaf dan Mohammed G. Al-Hamiri memiliki 2 model sistem yaitu menggunakan teknologi VLAN dan LAN (tidak menggunakan VLAN). Performa yang akan dievealuasi akan menggunakan server kabel dan server tanpa kabel. Dalam penggunaan VLAN, VLAN tersebut dibagi menjadi 2 bagian yaitu VLAN staff dan VLAN students. Hasil dari penelitian ini membuktikan bahwa arsitektur jaringan yang menggunakan VLAN memiliki tingkat *throughput* yang lebih kecil daripada tanpa VLAN [20]. Kemudian, VLAN juga memiliki tingkat *delay* yang lebih rendah daripada teknologi tanpa VLAN. Sedangkan pada penelitian yang akan dilakukan akan menggunakan arsitektur SDN berbasis *ryu controller* untuk mengetahui performa VLAN berdasarkan parameter *Qos* yaitu, *throughput*, *delay*, *packet loss*, latency.

Penelitian oleh Saleh Asadollahi et al dari *Computer Science* Saurashtra University dan Christ University yang memiliki judul "*Ryu Controller's Scalability Experiment on Software Defined Networks*" bertujuan untuk mengetahui skalabilitas dari *ryu controller* karena setiap harinya jumlah node yang terhubung dengan jaringan global internet. Penelitian ini berguna untuk melakukan pengujian terhadap teknologi baru sebelum melakukan pengujian dengan keaadan yang kedepannya akan selalu berubah-ubah. Sementara banyak penelitian yang dilakukan untuk memberikan solusi berupa SDN untuk mengatasi keterbatasan jaringan tradisional, hal tersebut menyebabkan komunitas riset ingin menguji penerapannya dan kaliber terhadap toleransi kesalahan pada SDN controller. Hasil penelitian ini adalah peneliti membuktikan bahwa setelah mengimplentasikan berbagai skenario yang berbeda dalam lingkungan eksperimen simulasi, peneliti memberikan gagasan yang jelas bagaimana membuat tes eksperimental test bed bersamaan dengan analisis hasil statistik yang diperoleh membuat kinerja *throughput* sebagai fokus utama. Peneliti menyimpulkan makalah tersebut dengan memberikan tanda negatif untuk maju ke peneliti yang mencari implementasi ide mereka atas *Ryu Controller* di domain Jaringan yang Ditetapkan Perangkat Lunak tanpa keraguan [10]. Sedangkan pada penelitian yang akan dilakukan akan

menggunakan *Ryu Controller* sebagai basis dari arsitektur SDN yang akan diterapkan VLAN.

Penelitian yang dilakukan oleh Rohmat Tulloh dkk dari Universitas Telkom melakukan Simulasi VLAN berbasis *Software Defined Network* yang menggunakan POX Controller. Penelitian ini memiliki fokus untuk mengevaluasi performa forwarding VLAN yang memanfaatkan *Openflow* sebagai *control plane* dapat berfungsi dengan baik. Hasil penelitian ini mengusulkan penerapan karakteristik teknologi VLAN pada SDN karena telah berjalan dengan benar sesuai hasil pengujian konektifitas, verifikasi dan keamanan. Kemudian hasil pengujian lanjutan untuk melihat pengaruh SDN dengan skenario penambahan jumlah VLAN ID didapatkan bahwa set-up time akan bertambah seiring meningkatnya jumlah host dan dengan menggunakan protokol OpenFlow, latency yang terjadi di jaringan dapat dipantau dengan parameter round trip time (RTT) yang stabil direntang 0,2 sampai 6 second walaupun jumlah vlan_id dan *background traffic* bertambah [21]. Sedangkan pada penelitian ini juga akan melakukan evaluasi performa VLAN dengan parameter *Qos* yaitu *throughput, jitter, delay, dan packet loss* yang menggunakan *Ryu Controller* dengan bandwidth yang telah ditentukan.

Penelitian yang dilakukan oleh Krisna Bayu Aditya Nurcahyo dan Agus Prihanto dari Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya bertujuan untuk menganalisis performansi jaringan sebelum dan sesudah menerapkan Virtual Local Area Network (VLAN). Skenario jaringan VLAN pada penelitian ini dibangun pada software GNS3 dan dilakukan 5 kali pengukuran parameter QoS yaitu bandwidth, *throughput*, packet loss, *delay*, dan *jitter* menggunakan Wireshark Network Analizer Tools. Hasil pengukuran bandwidth setelah menerapkan VLAN mendapatkan kecepatan download dan upload lebih kecil daripada tanpa menggunakan VLAN [22]. Sedangkan pada penelitian ini juga akan menggunakan standar parameter *Qos* yang sama namun akan diterapkan pada VLAN berbasis SDN.

Penelitian yang dilakukan Walaa Amayreh yang berjudul Analysis of the Vlan Network *Delay* Performance to Improve Quality of Services (QOS) meneliti tentang bagaimana meningkatkan performa QOS *delay* pada jaringan menggunakan *Virtual Local Area Network* (VLAN). Hasil dari penelitian ini pembangunan jaringan dan pengaturan jaringan VLAN adalah cara yang efektif untuk mengatasi masalah *delay*. Penerjemahan organisasi dan departemen adalah salah satu hal paling penting untuk meningkatkan performa jaringan. VLAN menerapkan gagasan tersebut karena tertarik pada divisi jaringan LAN untuk departemen. VLAN mengatur lalu lintas data dari dan ke jaringan sehingga teknologi VLAN menjadi efektif, murah dan sempurna untuk mengatasi *delay* dalam banyak kasus pada jaringan [23]. Sedangkan pada penelitian ini akan melakukan analisa performa VLAN pada *Software Defined Network* dengan parameter QOS *Throughput*, *Delay*, *Jitter*, dan *Packet Loss*.

Penelitian yang dilakukan oleh Romi Afan dkk yang berjudul "Analisis Efek Penggunaan Kontroler Ryu Dan Pox Pada Performansi Jaringan Sdn" yang meneliti tentang pengaruh penggunaan kontroler ryu dan pox terhadap performa jaringan SDN. Hasil dari penelitian ini adalah *resource utilization* pada kontroler ryu memiliki tingkat penggunaan memori yang lebih besar dari pada pox dengan nilai penggunaan ryu sebesar 1,7% dan pox sebesar 0,5% memory dari 4GB memori *virtual machine* dan penambahan jumlah switch terhadap topologi yang telah dirancang tidak mempengaruhi pemakaian jumlah memori baik kontroler ryu maupun pox [24].

III. METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Waktu dan tempat pelaksanaan penelitian dilakukan pada:

1. Waktu penelitian : 16 Juni 2022 – 16 Agustus 2022
2. Tempat penelitian : Universitas Lampung

3.2 Alat Penelitian

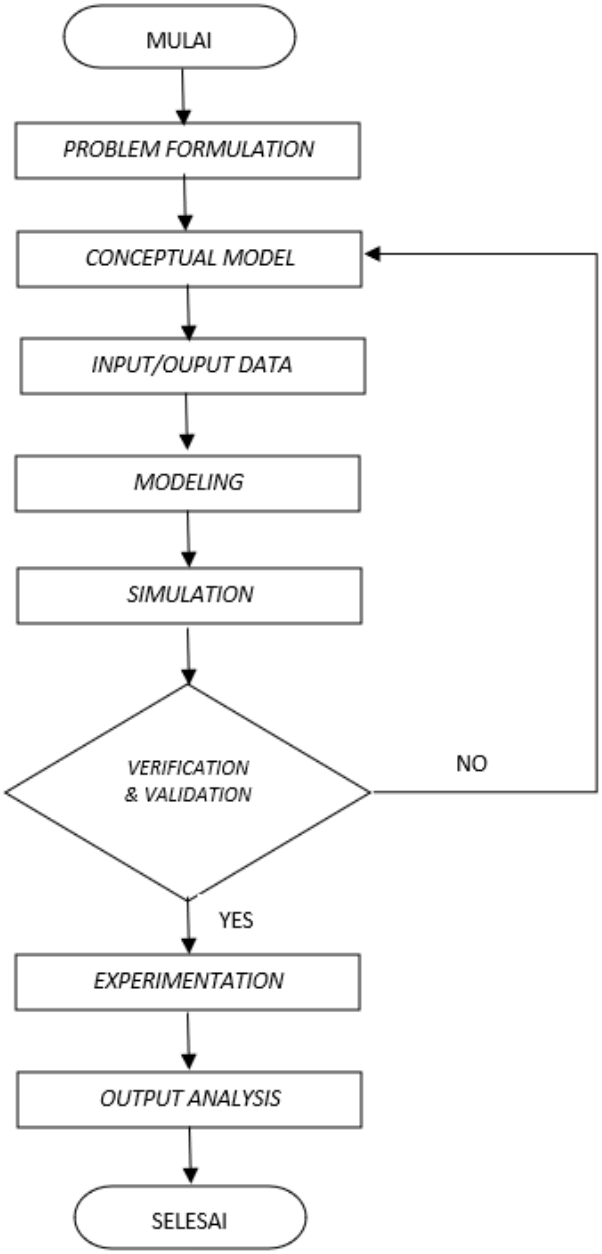
Alat-alat yang digunakan pada penelitian ini adalah sebagai berikut:

Tabel 3.1 Alat Penelitian

No	Nama	Spesifikasi	Fungsi
1	Laptop	AMD Ryzen 5 3500u, RAM 8GB, dengan sistem operasi Windows 10 Home Single Language	Perangkat perancangan dan pengujian
2	VMware	Workstation 16 player	*Virtual machine* yang digunakan sebagai perangkat untuk SDN
3	Ubuntu	Versi 20.04	Sistem operasi *virtual machine*
4	Mininet	Versi 2.3.0	Mininet sebagai simulator SDN
5	*RYU Controller*	Versi 4.30	RYU berperan sebagai *controller* dari SDN
6	MiniNAM	Versi 1.0.1	MiniNAM sebagai alat yang berfungsi memvisualiasi topologi dalam SDN
7	Wireshark	Versi 3.2.3	Wireshark berperan untuk mengcapture paket

			yang lewat dalam jaringan
8	Iperf		Iperf digunakan untuk menghasilkan *traffic* pada skenario yang dilakukan

3.3 Alur Penelitian



Gambar 3.1 Alur penelitian

3.4 Tahapan Penelitian

Pada penelitian ini digunakan metode simulasi untuk melakukan Analisa Kinerja *RYU Controller* Pada Jaringan *Software Defined Network*. Dalam metode simulasi memiliki beberapa tahapan seperti sebagai berikut:

3.4.1 *Problem Formulation*

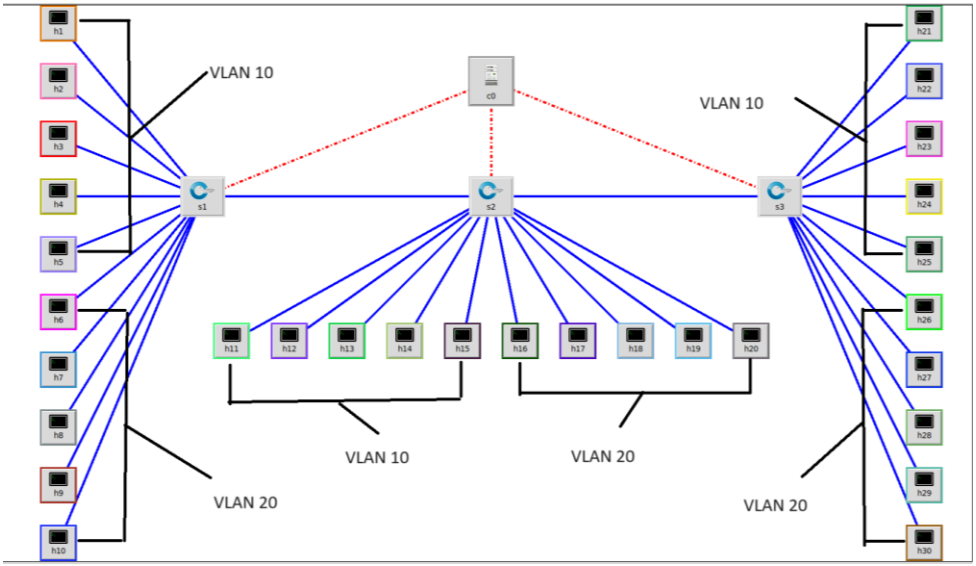
Pada tahap ini, dilakukan pencarian terhadap referensi-referensi yang berhubungan dengan topik dan metode yang akan digunakan dalam penelitian. Pencarian referensi tersebut dilakukan melalui buku, jurnal, dan juga penelitian-penelitian terdahulu yang dapat diakses secara online melalui internet. Informasi yang telah didapatkan kemudian dijadikan landasan teori, metode, dan cara Analisa Kinerja Jaringan *Software Defined Network* berdasarkan standar TIPHON. Kemudian didapatkan permasalahan yaitu bagaimana pengaruh VLAN atau pembagian segmen pada jaringan jika diterapkan pada *Software Defined Network* yang menggunakan *RYU Controller*. Penggunaan modul/app yang akan digunakan juga ditemukan dalam tahap ini yaitu, modul/app yang bernama rest_router yang dapat menjadikan Openflow Switch atau openflow switch bertindak seperti router sederhana. Modul ini dapat menjalankan static routing dan juga dapat melakukan pembagian segmen menggunakan VLAN. Modul/app ini juga memiliki dasaran kode yang sama namun yang membedakan adalah saat penerapan dapat dilakukan konfigurasi penambahan nomor VLAN. Kemudian untuk menjawab permasalaha tersebut didapatkan permasalahan utama analisa jaringan berbasis ryu controller yang menggunakan static routing dan juga VLAN.

3.4.2 *Conceptual Model*

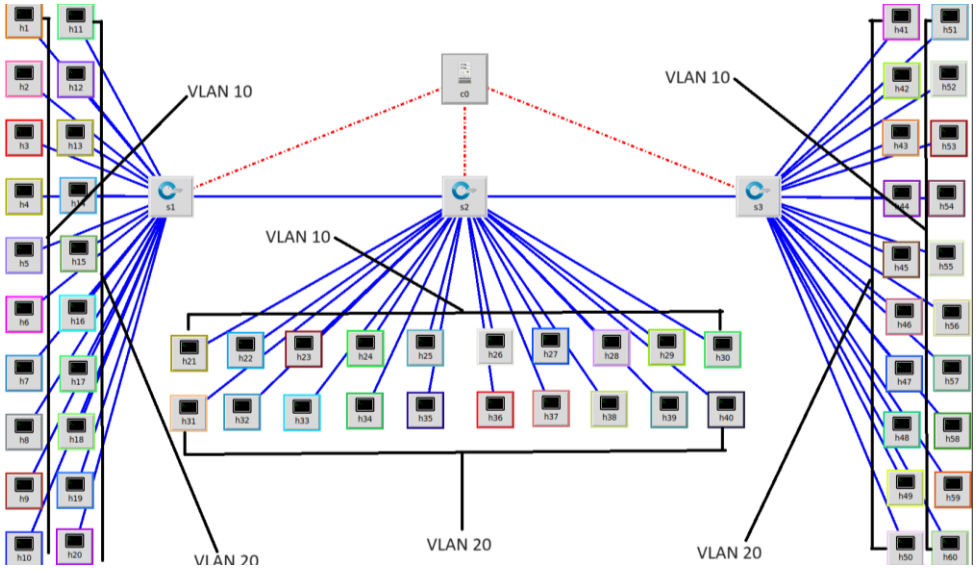
Conceptual Model merupakan penggambaran konsep model simulasi virtual, sebelum melakukan proses simulasi terlebih dahulu melakukan pemodelan konseptual dengan merancang topologi yang akan digunakan pada penelitian ini

beserta alamat IP dari masing- masing perangkat yang ada pada tiap topologi. Pada penelitian ini menggunakan mininet sebagai emulator yang berjalan pada ubuntu 20.04. Pada tahapan ini akan dilakukan perancangan topologi topologi jaringan dengan menggunakan 3 Openflow Switch yang terdiri dari 30 host dengan 10 host tiap Openflow Switch dan 3 Openflow Switch yang terdiri dari 60 host dengan 20 host pada tiap Openflow Switch yang kemudian dihubungkan dengan *RYU Controller* sebagai pengontrol dari jaringan yang dibuat. App/Modul yang akan dijalankan pada *RYU Controller* adalah rest_router yang dapat menjalankan fungsi simple router pada topologi yang terhubung dengan *RYU Controller*. Dengan adanya App tersebut maka dapat diterapkan fungsi static routing dan juga segmentasi jaringan dengan menggunakan VLAN. Langkah terakhir yang dilakukan yaitu melakukan Analisa dengan menggunakan parameter QoS.

Langkah awal pada tahap ini dilakukan dengan cara menentukan parameter (*delay*, *jitter*, *packet loss* dan *throughput* pada protokol UDP) dan karakteristik yang digunakan selama simulasi, yang dinamakan dengan variable. Pada tahap ini dilakukan pembuatan skenario-skenario yang akan digunakan untuk proses simulasi pada topologi yang telah ditentukan beserta variasi traffic. Berikut merupakan topologi yang akan digunakan dalam penelitian ini:



Gambar 3.2 Topologi 3 Openflow Switch dan 30 Host



Gambar 3.3 Topologi 3 Openflow Switch dan 60 Host

Adapun pengalamatan IP tersebut sebagai berikut:

Tabel 3.2 Pengalamatan IP 30 host

Device	IP Address	Subnet mask
C0	127.0.0.1	255.255.255.255
S1	172.16.20.1 172.16.30.30	255.255.255.0 255.255.255.0
S2	172.16.10.1 172.16.30.1 192.168.10.1	255.255.255.0 255.255.255.0 255.255.255.0
S3	192.168.30.1 192.168.10.1	255.255.255.0 255.255.255.0
h1	172.16.20.11	255.255.255.0
h2	172.16.20.12	255.255.255.0
h3	172.16.20.13	255.255.255.0
h4	172.16.20.14	255.255.255.0
h5	172.16.20.15	255.255.255.0
h6	172.16.20.16	255.255.255.0
h7	172.16.20.17	255.255.255.0
h8	172.16.20.18	255.255.255.0
h9	172.16.20.19	255.255.255.0

Device	IP Address	Subnet mask
h10	172.16.20.20	255.255.255.0
h11	172.16.10.11	255.255.255.0
h12	172.16.10.12	255.255.255.0
h13	172.16.10.13	255.255.255.0
h14	172.16.10.14	255.255.255.0
h15	172.16.10.15	255.255.255.0
h16	172.16.10.16	255.255.255.0
h17	172.16.10.17	255.255.255.0
h18	172.16.10.18	255.255.255.0
h19	172.16.10.19	255.255.255.0
h20	172.16.10.20	255.255.255.0
h21	192.168.30.11	255.255.255.0
h22	192.168.30.12	255.255.255.0
h23	192.168.30.13	255.255.255.0
h24	192.168.30.14	255.255.255.0
h25	192.168.30.15	255.255.255.0
h26	192.168.30.16	255.255.255.0
h27	192.168.30.17	255.255.255.0
h28	192.168.30.18	255.255.255.0
h29	192.168.30.19	255.255.255.0
h30	192.168.30.20	255.255.255.0

Kemudian, topologi tersebut memiliki konfigurasi yang kedua yaitu penambahan VLAN untuk pemisahan host dalam jaringan tersebut.

Tabel 3.3 Pengalamatan IP 30 host dengan VLAN

Device	IP Address	Subnet mask	VLAN
C0	127.0.0.1	255.255.255.255	-
S1	10.0.0.1	255.255.255.0	-
S2	10.0.0.2	255.255.255.0	-
S3	10.0.0.3	255.255.255.0	-
h1	172.16.20.11	255.255.255.0	10
h2	172.16.20.12	255.255.255.0	10

Device	IP Address	Subnet mask	VLAN
h3	172.16.20.13	255.255.255.0	10
h4	172.16.20.14	255.255.255.0	10
h5	172.16.20.15	255.255.255.0	10
h6	172.16.20.16	255.255.255.0	20
h7	172.16.20.17	255.255.255.0	20
h8	172.16.20.18	255.255.255.0	20
h9	172.16.20.19	255.255.255.0	20
h10	172.16.20.20	255.255.255.0	20
h11	172.16.10.11	255.255.255.0	10
h12	172.16.10.12	255.255.255.0	10
h13	172.16.10.13	255.255.255.0	10
h14	172.16.10.14	255.255.255.0	10
h15	172.16.10.15	255.255.255.0	10
h16	172.16.10.16	255.255.255.0	20
h17	172.16.10.17	255.255.255.0	20
h18	172.16.10.18	255.255.255.0	20
h19	172.16.10.19	255.255.255.0	20
h20	172.16.10.20	255.255.255.0	20
h21	192.168.30.11	255.255.255.0	10
h22	192.168.30.12	255.255.255.0	10
h23	192.168.30.13	255.255.255.0	10
h24	192.168.30.14	255.255.255.0	10
h25	192.168.30.15	255.255.255.0	10
h26	192.168.30.16	255.255.255.0	20
h27	192.168.30.17	255.255.255.0	20
h28	192.168.30.18	255.255.255.0	20
h29	192.168.30.19	255.255.255.0	20
h30	192.168.30.20	255.255.255.0	20

Tabel 3.4 Pengalamatan IP 60 host

Device	IP Address	Subnet mask
C0	127.0.0.1	255.255.255.255
S1	172.16.20.1	255.255.255.0

Device	IP Address	Subnet mask
	172.16.30.30	255.255.255.0
S2	172.16.10.1	255.255.255.0
	172.16.30.1	255.255.255.0
	192.168.10.1	255.255.255.0
S3	192.168.30.1	255.255.255.0
	192.168.10.1	255.255.255.0
h1	172.16.20.11	255.255.255.0
h2	172.16.20.12	255.255.255.0
h3	172.16.20.13	255.255.255.0
h4	172.16.20.14	255.255.255.0
h5	172.16.20.15	255.255.255.0
h6	172.16.20.16	255.255.255.0
h7	172.16.20.17	255.255.255.0
h8	172.16.20.18	255.255.255.0
h9	172.16.20.19	255.255.255.0
h10	172.16.20.20	255.255.255.0
h11	172.16.20.21	255.255.255.0
h12	172.16.20.22	255.255.255.0
h13	172.16.20.23	255.255.255.0
h14	172.16.20.24	255.255.255.0
h15	172.16.20.25	255.255.255.0
h16	172.16.20.26	255.255.255.0
h17	172.16.20.27	255.255.255.0
h18	172.16.20.28	255.255.255.0
h19	172.16.20.29	255.255.255.0
h20	172.16.20.30	255.255.255.0
h21	172.16.10.11	255.255.255.0
h22	172.16.10.12	255.255.255.0
h23	172.16.10.13	255.255.255.0
h24	172.16.10.14	255.255.255.0
h25	172.16.10.15	255.255.255.0
h26	172.16.10.16	255.255.255.0
h27	172.16.10.17	255.255.255.0
h28	172.16.10.18	255.255.255.0
h29	172.16.10.19	255.255.255.0
h30	172.16.10.20	255.255.255.0

Device	IP Address	Subnet mask
h31	172.16.10.21	255.255.255.0
h32	172.16.10.22	255.255.255.0
h33	172.16.10.23	255.255.255.0
h34	172.16.10.24	255.255.255.0
h35	172.16.10.25	255.255.255.0
h36	172.16.10.26	255.255.255.0
h37	172.16.10.27	255.255.255.0
h38	172.16.10.28	255.255.255.0
h39	172.16.10.29	255.255.255.0
h40	172.16.10.30	255.255.255.0
h41	192.168.30.11	255.255.255.0
h42	192.168.30.12	255.255.255.0
h43	192.168.30.13	255.255.255.0
h44	192.168.30.14	255.255.255.0
h45	192.168.30.15	255.255.255.0
h46	192.168.30.16	255.255.255.0
h47	192.168.30.17	255.255.255.0
h48	192.168.30.18	255.255.255.0
h49	192.168.30.19	255.255.255.0
h50	192.168.30.20	255.255.255.0
h51	192.168.30.21	255.255.255.0
h52	192.168.30.22	255.255.255.0
h53	192.168.30.23	255.255.255.0
h54	192.168.30.24	255.255.255.0
h55	192.168.30.25	255.255.255.0
h56	192.168.30.26	255.255.255.0
h57	192.168.30.27	255.255.255.0
h58	192.168.30.28	255.255.255.0
h59	192.168.30.29	255.255.255.0
h60	192.168.30.30	255.255.255.0

Kemudian, topologi tersebut memiliki konfigurasi yang kedua yaitu penambahan VLAN untuk pemisahan host dalam jaringan tersebut.

Tabel 3.5 Pengalamatan IP 60 host dengan VLAN

Device	IP Address	Subnet mask	VLAN
C0	127.0.0.1	255.255.255.255	-
S1	10.0.0.1	255.255.255.0	-
S2	10.0.0.2	255.255.255.0	-
S3	10.0.0.3	255.255.255.0	-
h1	172.16.20.11	255.255.255.0	10
h2	172.16.20.12	255.255.255.0	10
h3	172.16.20.13	255.255.255.0	10
h4	172.16.20.14	255.255.255.0	10
h5	172.16.20.15	255.255.255.0	10
h6	172.16.20.16	255.255.255.0	10
h7	172.16.20.17	255.255.255.0	10
h8	172.16.20.18	255.255.255.0	10
h9	172.16.20.19	255.255.255.0	10
h10	172.16.20.20	255.255.255.0	10
h11	172.16.20.21	255.255.255.0	20
h12	172.16.20.22	255.255.255.0	20
h13	172.16.20.23	255.255.255.0	20
h14	172.16.20.24	255.255.255.0	20
h15	172.16.20.25	255.255.255.0	20
h16	172.16.20.26	255.255.255.0	20
h17	172.16.20.27	255.255.255.0	20
h18	172.16.20.28	255.255.255.0	20
h19	172.16.20.29	255.255.255.0	20
h20	172.16.20.30	255.255.255.0	20
h21	172.16.10.11	255.255.255.0	10
h22	172.16.10.12	255.255.255.0	10
h23	172.16.10.13	255.255.255.0	10
h24	172.16.10.14	255.255.255.0	10
h25	172.16.10.15	255.255.255.0	10
h26	172.16.10.16	255.255.255.0	10
h27	172.16.10.17	255.255.255.0	10
h28	172.16.10.18	255.255.255.0	10
h29	172.16.10.19	255.255.255.0	10
h30	172.16.10.20	255.255.255.0	10
h31	172.16.10.21	255.255.255.0	20

Device	IP Address	Subnet mask	VLAN
h32	172.16.10.22	255.255.255.0	20
h33	172.16.10.23	255.255.255.0	20
h34	172.16.10.24	255.255.255.0	20
h35	172.16.10.25	255.255.255.0	20
h36	172.16.10.26	255.255.255.0	20
h37	172.16.10.27	255.255.255.0	20
h38	172.16.10.28	255.255.255.0	20
h39	172.16.10.29	255.255.255.0	20
h40	172.16.10.30	255.255.255.0	20
h41	192.168.30.11	255.255.255.0	10
h42	192.168.30.12	255.255.255.0	10
h43	192.168.30.13	255.255.255.0	10
h44	192.168.30.14	255.255.255.0	10
h45	192.168.30.15	255.255.255.0	10
h46	192.168.30.16	255.255.255.0	10
h47	192.168.30.17	255.255.255.0	10
h48	192.168.30.18	255.255.255.0	10
h49	192.168.30.19	255.255.255.0	10
h50	192.168.30.20	255.255.255.0	10
h51	192.168.30.21	255.255.255.0	20
h52	192.168.30.22	255.255.255.0	20
h53	192.168.30.23	255.255.255.0	20
h54	192.168.30.24	255.255.255.0	20
h55	192.168.30.25	255.255.255.0	20
h56	192.168.30.26	255.255.255.0	20
h57	192.168.30.27	255.255.255.0	20
h58	192.168.30.28	255.255.255.0	20
h59	192.168.30.29	255.255.255.0	20
h60	192.168.30.30	255.255.255.0	20

3.4.3 *Input/Output Data*

a. *Input*

Input adalah salah satu hal yang akan digunakan yang berfungsi sebagai atribut, diantaranya:

1. ***Node***

   Node adalah perangkat yang berada pada topologi yang dibuat dalam jaringan *Software Defined Network*. Node disini berada dalam topologi linear dengan jumlah host sebanyak 30 dan 60 buah serta OVS Switch sebanyak 3 buah dengan tiap Openflow Switch yang dihubungkan ke *RYU controller*.

2. ***Background Traffic***

   Background Traffic merupakan jumlah paket yang dikirimkan dan diterima dalam sebuah lalu lintas data pada jaringan. Dalam penelitian ini, *Background Traffic* yang akan digunakan adalah 100, 200, 300, 400 Mb.

b. *Output*

1. Throughput

Throughput berfungsi sebagai pengukur kecepatan pengiriman paket pada Analisa *RYU controller.*

2. Delay

Delay, digunakan sebagai pengukur waktu yang dibutuhkan oleh data untuk sampai ke alamat tujuan.

3. Jitter

Jitter bisa disebut juga dengan variasi yang disebabkan oleh antrian yang panjang dalam pengiriman data, waktu pengolahan data, dan waktu pengumpulan paket ke tujuan.

4. *Packet Loss*

Packet Loss ialah jumlah dari keseluruhan paket yang hilang atau tidak sampai ke tujuan yang dapat disebakan oleh *collision ataupun congestion* didalam jaringan. *Packet Loss* sendiri biasanya ditampilkan dalam jumlah persentase.

3.4.4 Modelling

Pada tahap ini dilakukan pembuatan skenario yang nantinya akan digunakan untuk mendapatkan parameter yang dibutuhkan (*Throughput, Packet loss, dan Jitter*) termasuk dengan hal-hal yang perlu dilakukan selama simulasi seperti command-command yang akan digunakan. Dibawah ini merupakan skenario pengujian yang telah dibuat.

3.4.4.1 Skenario pengujian konektivitas static routing 30 Host

Skenario pertama yang akan dilakukan adalah pengujian konektivitas static routing. Topologi yang akan digunakan memiliki 30 host dan 3 OVS Switch. Pada skenario ini program yang akan digunakan adalah static routing. Hal pertama yang akan dilakukan adalah dengan melakukan perintah pingall. Perintah tersebut digunakan untuk melakukan pengiriman paket kepada seluruh perangkat yang terhubung dalam jaringan yang bertujuan untuk menguji konektivitas setiap perangkat atau host yang terhubung. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server. Node yang akan digunakan pada skenario ini adalah node h1, h20, dan h30. Node-node tersebut akan bergantian untuk menjadi client dan juga server dengan catatan tiap node akan mengisi posisi client sebanyak 2 kali dan server sebanyak 2 kali seperti yang ada pada tabel berikut ini:

Tabel 3.6 Skenario Client Server tanpa VLAN 30 host

No	Client	Server
1	H1	H20
2	H1	H30
3	H20	H1
4	H20	H30
5	H30	H1
6	H30	H20

Setelah itu, akan dilakukan pengujian dengan beberapa variasi *traffic* yaitu 100, 200, 300 dan 400 mb dengan durasi waktu 10 detik dengan interval *report* setiap 1 detik. Command untuk sisi client adalah “iperf -u -c (ip server) -i 1 -b (bandwidth)” sedangkan untuk sisi server menggunakan command “iperf -u -s -i 1 yang nantinya ip server dan *traffic* yang dihasilkan akan diubah sesuai skenario yang dijalankan. Kemudian, paket yang dikirimkan tersebut akan di*capture* menggunakan wireshark. Setelah paket dicapure akan dilakukan pengolahan dan

penghitungan dari hasi tersebut untuk mendapatkan parameter *Qos* yang berupa *Throughput*, *Jitter*, *Delay*, dan *Packet Loss*.

3.4.4.2 Skenario pengujian konektivitas VLAN 10 30 host

Pada skenario kedua, akan dilakukan pengujian konektivitas jaringan menggunakan VLAN. Namun, untuk spesifik akan dilakukan pengujian hanya untuk VLAN 10. Topologi yang digunakan juga masih sama dengan 30 host dan 3 OVS Switch. Pada skenario ini program yang akan digunakan adalah static routing dengan VLAN. Hal pertama yang akan dilakukan adalah dengan melakukan perintah pingall. Perintah tersebut digunakan untuk melakukan pengiriman paket kepada seluruh perangkat yang terhubung dalam jaringan yang bertujuan untuk menguji konektivitas setiap perangkat atau host yang terhubung. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server seperti tabel dibawah ini:

Tabel 3.7 Skenario Client server VLAN 10 30 host

No	Client	Server
1	H1	H11
2	H1	H21
3	H11	H1
4	H11	H21
5	H21	H1
6	H21	H11

Selama pengujian tersebut, akan dilakukan pengujian dengan beberapa variasi *traffic* yaitu 100, 200, 300 dan 400 mb dengan durasi waktu 10 detik dengan interval *report* setiap 1 detik. Command untuk sisi client adalah “iperf -u -c (ip server) -i 1 -b (bandwidth)” sedangkan untuk sisi server menggunakan command

“iperf -u -s -i 1 yang nantinya ip server dan *traffic* yang dihasilkan akan diubah sesuai skenario yang dijalankan. Kemudian, paket yang dikirimkan tersebut akan di*capture* menggunakan wireshark. Dan setelah itu akan dilakukan perhitungan parameter *Qos* yang berupa *Throughput*, *Jitter*, *Delay*, dan *Packet Loss*.

3.4.4.3 Skenario pengujian konektivitas VLAN 20 30 host

Pada skenario terakhir, akan dilakukan pengujian konektivitas jaringan menggunakan VLAN. Namun, untuk spesifik akan dilakukan pengujian hanya untuk VLAN 10. Topologi yang digunakan juga masih sama dengan 30 host dan 3 OVS Switch. Pada skenario ini program yang akan digunakan adalah static routing dengan VLAN. Hal pertama yang akan dilakukan adalah dengan melakukan perintah pingall. Perintah tersebut digunakan untuk melakukan pengiriman paket kepada seluruh perangkat yang terhubung dalam jaringan yang bertujuan untuk menguji konektivitas setiap perangkat atau host yang terhubung. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server seperti tabel dibawah ini:

Tabel 3.8 Skenario Client server VLAN 20 30 host

No	Client	Server
1	H6	H16
2	H6	H26
3	H16	H6
4	H16	H26
5	H26	H6
6	H26	H16

Selama pengujian tersebut, akan dilakukan pengujian dengan beberapa variasi *traffic* yaitu 100, 200, 300 dan 400 mb dengan durasi waktu 10 detik dengan

interval *report* setiap 1 detik. Command untuk sisi client adalah “iperf -u -c (ip server) -i 1 -b (bandwidth)” sedangkan untuk sisi server menggunakan command “iperf -u -s -i 1 yang nantinya ip server dan *traffic* yang dihasilakan akan diubah sesuai skenario yang dijalankan. Kemudian, paket yang dikirimkan tersebut akan di*capture* menggunakan wireshark. Dan setelah itu akan dilakukan perhitungan parameter *Qos* yang berupa *Throughput*, *Jitter*, *Delay*, dan *Packet Loss*.

3.4.4.4 Skenario pengujian konektivitas static routing 60 Host

Skenario pertama yang akan dilakukan adalah pengujian konektivitas static routing. Topologi yang akan digunakan memiliki 30 host dan 3 OVS Switch. Pada skenario ini program yang akan digunakan adalah static routing. Hal pertama yang akan dilakukan adalah dengan melakukan perintah pingall. Perintah tersebut digunakan untuk melakukan pengiriman paket kepada seluruh perangkat yang terhubung dalam jaringan yang bertujuan untuk menguji konektivitas setiap perangkat atau host yang terhubung. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server. Node yang akan digunakan pada skenario ini adalah node h1, h20, dan h30. Node-node tersebut akan bergantian untuk menjadi client dan juga server dengan catatan tiap node akan mengisi posisi client sebanyak 2 kali dan server sebanyak 2 kali seperti yang ada pada tabel berikut ini

Tabel 3.9 Skenario Client Server tanpa VLAN 60 host

No	Client	Server
1	H1	H21
2	H1	H41
3	H21	H1
4	H21	H41
5	H41	H1
6	H41	H21

Setelah itu, akan dilakukan pengujian dengan beberapa variasi *traffic* yaitu 100, 200, 300 dan 400 mb dengan durasi waktu 10 detik dengan interval *report* setiap 1 detik. Command untuk sisi client adalah “iperf -u -c (ip server) -i 1 -b (bandwidth)” sedangkan untuk sisi server menggunakan command “iperf -u -s -i 1 yang nantinya ip server dan *traffic* yang dihasilkan akan diubah sesuai skenario yang dijalankan. Kemudian, paket yang dikirimkan tersebut akan di*capture* menggunakan wireshark. Setelah paket dicapure akan dilakukan pengolahan dan penghitungan dari hasi tersebut untuk mendapatkan parameter *Qos* yang berupa *Throughput*, *Jitter*, *Delay*, dan *Packet Loss*.

3.4.4.5 Skenario pengujian konektivitas VLAN 10 60 host

Pada skenario kedua, akan dilakukan pengujian konektivitas jaringan menggunakan VLAN. Namun, untuk spesifik akan dilakukan pengujian hanya untuk VLAN 10. Topologi yang digunakan juga masih sama dengan 30 host dan 3 OVS Switch. Pada skenario ini program yang akan digunakan adalah static routing dengan VLAN. Hal pertama yang akan dilakukan adalah dengan melakukan perintah pingall. Perintah tersebut digunakan untuk melakukan pengiriman paket kepada seluruh perangkat yang terhubung dalam jaringan yang bertujuan untuk menguji konektivitas setiap perangkat atau host yang terhubung. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server seperti tabel dibawah ini:

Tabel 3.10 Skenario Client server VLAN 10 60 host

No	Client	Server
1	H1	H21
2	H1	H41
3	H21	H1

No	Client	Server
4	H21	H41
5	H41	H1
6	H41	H21

Selama pengujian tersebut, akan dilakukan pengujian dengan beberapa variasi *traffic* yaitu 100, 200, 300 dan 400 mb dengan durasi waktu 10 detik dengan interval *report* setiap 1 detik. Command untuk sisi client adalah “iperf -u -c (ip server) -i 1 -b (bandwidth)” sedangkan untuk sisi server menggunakan command “iperf -u -s -i 1 yang nantinya ip server dan *traffic* yang dihasilkan akan diubah sesuai skenario yang dijalankan. Kemudian, paket yang dikirimkan tersebut akan di*capture* menggunakan wireshark. Dan setelah itu akan dilakukan perhitungan parameter *Qos* yang berupa *Throughput*, *Jitter*, *Delay*, dan *Packet Loss*.

3.4.4.6 Skenario pengujian konektivitas VLAN 20 60 host

Pada skenario terakhir, akan dilakukan pengujian konektivitas jaringan menggunakan VLAN. Namun, untuk spesifik akan dilakukan pengujian hanya untuk VLAN 10. Topologi yang digunakan juga masih sama dengan 30 host dan 3 OVS Switch. Pada skenario ini program yang akan digunakan adalah static routing dengan VLAN. Hal pertama yang akan dilakukan adalah dengan melakukan perintah pingall. Perintah tersebut digunakan untuk melakukan pengiriman paket kepada seluruh perangkat yang terhubung dalam jaringan yang bertujuan untuk menguji konektivitas setiap perangkat atau host yang terhubung. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server. Kemudian akan dilakukan pengiriman *background traffic* dengan menggunakan iperf, untuk melakukan pengujian akan digunakan 2 buah node yang nantinya akan berperan sebagai client dan juga server seperti tabel dibawah ini:

Tabel 3.11 Skenario Client server VLAN 20 60 host

No	Client	Server
1	H11	H31
2	H11	H51
3	H31	H11
4	H31	H51
5	H51	H11
6	H51	H31

Selama pengujian tersebut, akan dilakukan pengujian dengan beberapa variasi *traffic* yaitu 100, 200, 300 dan 400 mb dengan durasi waktu 10 detik dengan interval *report* setiap 1 detik. Command untuk sisi client adalah "iperf -u -c (ip server) -i 1 -b (bandwidth)" sedangkan untuk sisi server menggunakan command "iperf -u -s -i 1 yang nantinya ip server dan *traffic* yang dihasilakan akan diubah sesuai skenario yang dijalankan. Kemudian, paket yang dikirimkan tersebut akan di*capture* menggunakan wireshark. Dan setelah itu akan dilakukan perhitungan parameter *Qos* yang berupa *Throughput*, *Jitter*, *Delay*, dan *Packet Loss*.

3.4.5 *Simulation*

Pada fase simulasi akan dilakukan penerapan model yang dihasilkan pada tahapan sebelumnya. Pada penelitian ini implementasi akan disimulasikan dengan variable atau parameter- parameter yang sudah ditentukan. Setelah proses simulasi dilakukan maka hasil rekaman komunikasi data tersebut diproses sesuai dengan kebutuhan yang diperlukan yang akan menghasilkan sebuah informasi untuk proses analisis kinerja.

a. Instalasi *Mininet*

Dalam prosesnya, instalasi dari mininet ini akan dijalankan pada sistem operasi ubuntu 20.04. Berikut merupakan langkah-langkah untuk melakukan instalasi:

1. Sebelum melakukan proses penginstalan, dilakukan update ubuntu yang digunakan dengan *command* $ sudo apt-get update

2. Selanjutnya, lakukan proses penginstalan git menggunakan *command* $ sudo apt-get install git
3. Setelah penginstalan selesai, dilakukan *clone repository* dari github milik mininet dengan *command* $ git clone https://github.com/mininet/mininet
4. Setelah *clone repository* selesai dilakukan. Gunakan *command* $cd mininet untuk masuk kedalam direktori. Gunakan *command* $ git tag untuk memeriksa versi yang tersedia pada mininet. Terakhir, gunakan comman $ git checkout untuk menentukan versi mininet yang ingin diinstal. Pada penelitian ini penulis menggunakan versi 2.2.2, maka *command* yang digunakan adalah $ git checkout -b 2.2.2 2.2.2
5. Pada proses terakhir, gunakan *command* $mininet/util/install.sh -a. *Command* ini akan menginstall keseluruhan *package* yang dibutuhkan oleh mininet termasuk *Open Vswitch*, *wireshark*

b. Instalasi MiniNAM

MiniNAM merupakan aplikasi yang akan memvisualisasikan topologi yang telah dibuat. Dalam penerapannya, untuk menjalankan MiniNAM tetap dibutuhkan untuk instalasi mininet. Untuk melakukan instalasi MiniNAM, cukup dengan mengetik ***sudo apt-get install python-imaging-tk -y*** pada terminal dan tunggu proses intalasi selesai.

c. Instalasi *RYU Controller*

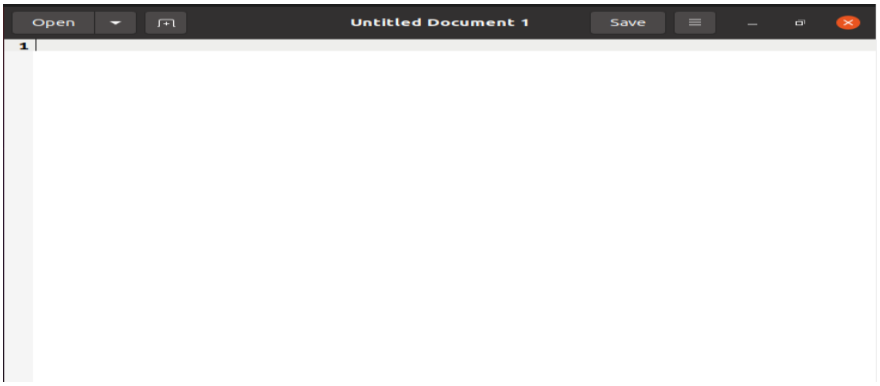
Dalam prosesnya, instalasi *RYU controller* akan dijalankan pada sistem operasi Ubuntu 20.04. Berikut merupakan langkah-langkah dalam melakukan instalasi:

1. Lakukan update pada ubuntu menggunakan *command* $ sudo apt-get update
2. Selanjutnya, lakukan clone repository dari github milik Ryu dengan *command* $ git clone git://github.com/osrg/Ryu.git
3. Terakhir, masuk kedalam repository Ryu dengan menggunakan *command* $cd Ryu dan lakukan proses instalasi menggunakan *command* $ pip install Ryu.

d. Membuat dan menjalankan topologi di MiniNAM

Pembuatan topologi di mininet dilakukan seperti sebagai berikut :

1. Masuk kedalam text editor



Gambar 3.4 Text editor ubuntu

2. Kemudian buat topologi menggunakan Bahasa pemrograman python seperti berikut ini:

```
"""
skripsi.py
Custom opology creation for routing example.
"""
from mininet.topo import Topo

class MyTopo( Topo ):

    def __init__( self, **oopts ):

        "Create custom topo."

        #Initialize topology
        Topo.__init__( self, **oopts  )

        # Add hosts and switches
        s1 = self.addSwitch('s1', protocols="OpenFlow13")
        s2 = self.addSwitch('s2', protocols="OpenFlow13")
        s3 = self.addSwitch('s3', protocols="OpenFlow13")

        h1 = self.addHost('h1', ip='172.16.20.11/24', defaultRoute='via 172.16.20.1')
        h2 = self.addHost('h2', ip='172.16.20.12/24', defaultRoute='via 172.16.20.1')
        h3 = self.addHost('h3', ip='172.16.20.13/24', defaultRoute='via 172.16.20.1')
        h4 = self.addHost('h4', ip='172.16.20.14/24', defaultRoute='via 172.16.20.1')
        h5 = self.addHost('h5', ip='172.16.20.15/24', defaultRoute='via 172.16.20.1')
        h6 = self.addHost('h6', ip='172.16.20.16/24', defaultRoute='via 172.16.20.1')
        h7 = self.addHost('h7', ip='172.16.20.17/24', defaultRoute='via 172.16.20.1')
        h8 = self.addHost('h8', ip='172.16.20.18/24', defaultRoute='via 172.16.20.1')
        h9 = self.addHost('h9', ip='172.16.20.19/24', defaultRoute='via 172.16.20.1')
        h10 = self.addHost('h10', ip='172.16.20.20/24', defaultRoute='via 172.16.20.1')
        h11 = self.addHost('h11', ip='172.16.10.11/24', defaultRoute='via 172.16.10.1')
        h12 = self.addHost('h12', ip='172.16.10.12/24', defaultRoute='via 172.16.10.1')
        h13 = self.addHost('h13', ip='172.16.10.13/24', defaultRoute='via 172.16.10.1')
        h14 = self.addHost('h14', ip='172.16.10.14/24', defaultRoute='via 172.16.10.1')
        h15 = self.addHost('h15', ip='172.16.10.15/24', defaultRoute='via 172.16.10.1')
        h16 = self.addHost('h16', ip='172.16.10.16/24', defaultRoute='via 172.16.10.1')
        h17 = self.addHost('h17', ip='172.16.10.17/24', defaultRoute='via 172.16.10.1')
        h18 = self.addHost('h18', ip='172.16.10.18/24', defaultRoute='via 172.16.10.1')
        h19 = self.addHost('h19', ip='172.16.10.19/24', defaultRoute='via 172.16.10.1')
        h20 = self.addHost('h20', ip='172.16.10.20/24', defaultRoute='via 172.16.10.1')
        h21 = self.addHost('h21', ip='192.168.30.11/24', defaultRoute='via 192.168.30.1')
        h22 = self.addHost('h22', ip='192.168.30.12/24', defaultRoute='via 192.168.30.1')
        h23 = self.addHost('h23', ip='192.168.30.13/24', defaultRoute='via 192.168.30.1')
        h24 = self.addHost('h24', ip='192.168.30.14/24', defaultRoute='via 192.168.30.1')
        h25 = self.addHost('h25', ip='192.168.30.15/24', defaultRoute='via 192.168.30.1')
        h26 = self.addHost('h26', ip='192.168.30.16/24', defaultRoute='via 192.168.30.1')
        h27 = self.addHost('h27', ip='192.168.30.17/24', defaultRoute='via 192.168.30.1')
        h28 = self.addHost('h28', ip='192.168.30.18/24', defaultRoute='via 192.168.30.1')
        h29 = self.addHost('h29', ip='192.168.30.19/24', defaultRoute='via 192.168.30.1')
        h30 = self.addHost('h30', ip='192.168.30.20/24', defaultRoute='via 192.168.30.1')

```

Gambar 3.5 Source code Node

```
52          #Add links
53          self.addLink(s1, s2)
54          self.addLink(s2, s3)
55          self.addLink(s1, h1)
56          self.addLink(s1, h2)
57          self.addLink(s1, h3)
58          self.addLink(s1, h4)
59          self.addLink(s1, h5)
60          self.addLink(s1, h6)
61          self.addLink(s1, h7)
62          self.addLink(s1, h8)
63          self.addLink(s1, h9)
64          self.addLink(s1, h10)
65          self.addLink(s2, h11)
66          self.addLink(s2, h12)
67          self.addLink(s2, h13)
68          self.addLink(s2, h14)
69          self.addLink(s2, h15)
70          self.addLink(s2, h16)
71          self.addLink(s2, h17)
72          self.addLink(s2, h18)
73          self.addLink(s2, h19)
74          self.addLink(s2, h20)
75          self.addLink(s3, h21)
76          self.addLink(s3, h22)
77          self.addLink(s3, h23)
78          self.addLink(s3, h24)
79          self.addLink(s3, h25)
80          self.addLink(s3, h26)
81          self.addLink(s3, h27)
82          self.addLink(s3, h28)
83          self.addLink(s3, h29)
84          self.addLink(s3, h30)
85
86 topos = { 'mytopo': ( lambda: MyTopo() ) }
87
```

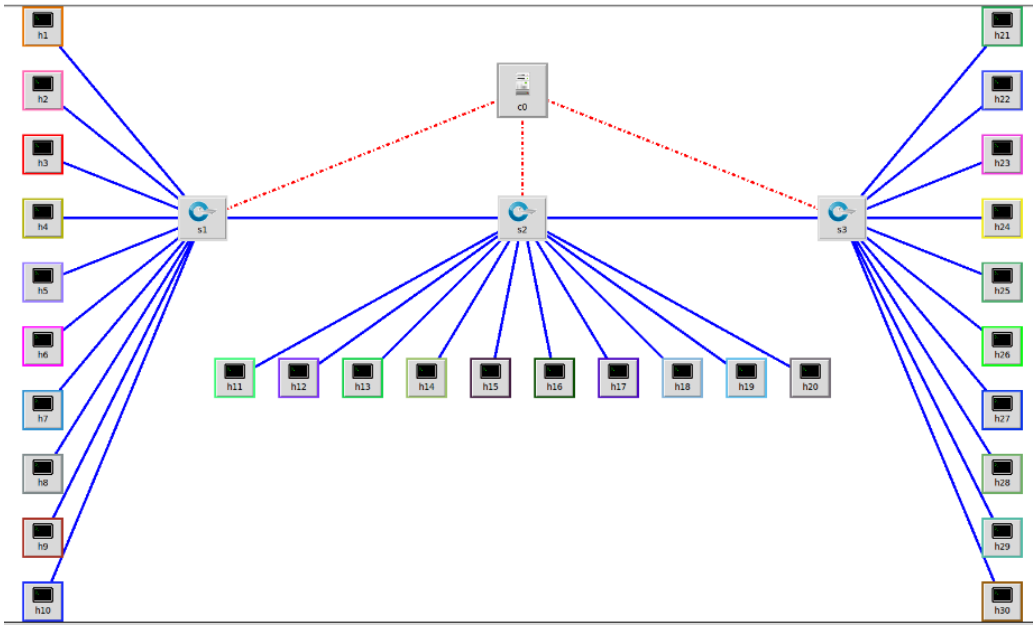
Gambar 3.6 Source code link antar node

3. Selanjutnya, menjalankan topologi menggunakan MiniNAM. dengan cara masuk ke direktori MiniNAM dengan *command* cd MiniNAM lalu gunakan perintah $ sudo python MiniNAM.py --custom skripsi.py --topo mytopo --controller remote untuk menjalankan topologi hingga menampilkan tampilan seperti dibawah ini:

```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h
23 h24 h25 h26 h27 h28 h29 h30
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h1) (s1, h2) (s1, h3) (s1, h4) (s1, h5) (s1, h6) (s1, h7) (s1, h8) (s1, h9)
 (s1, h10) (s1, s2) (s2, h11) (s2, h12) (s2, h13) (s2, h14) (s2, h15) (s2, h16)
(s2, h17) (s2, h18) (s2, h19) (s2, h20) (s2, s3) (s3, h21) (s3, h22) (s3, h23) (
s3, h24) (s3, h25) (s3, h26) (s3, h27) (s3, h28) (s3, h29) (s3, h30)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h
23 h24 h25 h26 h27 h28 h29 h30
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Gambar 3.7 Tampilan MiniNAM pada terminal

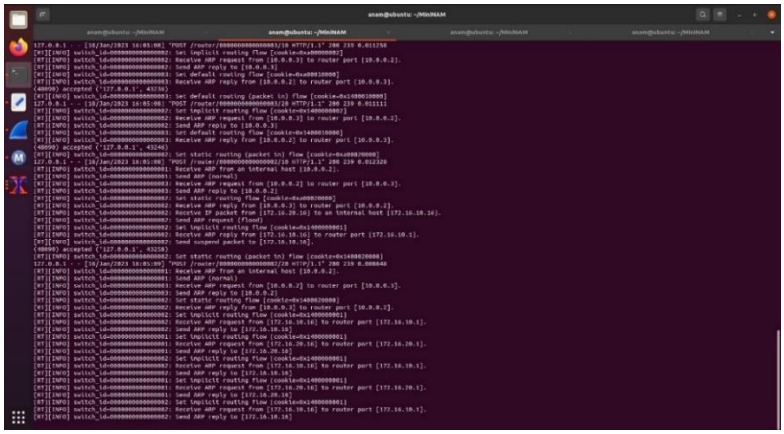
Kemudian akan menampilkan tampilan topologi seperti berikut ini:



Gambar 3.8 Tampilan topologi pada MiniNAM

e. Menjalankan *RYU Controller*

Untuk dapat menjalankan app/modul yang akan digunakan pada arsitektur SDN digunakan *command* ryu-manager yang diikuti dengan ryu.app nama app yang akan dijalankan. Pada penelitian ini akan menggunakan *app* rest_router yang sudah ada didalam instalasi ryu. Rest_router memiliki fungsi untuk dapat menjalankan fungsi *simple router* pada arsitektur *Software Defined Network.* Setelah *command* dijalankan maka *simple router* tersebut akan diterapkan pada akan muncul tampilan seperti sebagai berikut :



Gambar 3.9 *RYU Controller* berjalan

Setelah controller berjalan, tambahkan tab baru pada terminal untuk melakukan konfigurasi IP Address dan juga gateway yang akan digunakan oleh Openflow Switch yang nantinya akan memiliki fungsi seperti router setelah menjalankan *app* yang ada di ryu yaitu rest_router. Berikut adalah daftar command curl yang digunakan untuk melakukan POST atau memberikan IP Address dan gateway kepada Openflow Switch sesuai dengan dpid yang dimiliki.

```
curl -X POST -d '{"address":"172.16.20.1/24"}' http://localhost:8080/router/0000000000000001
curl -X POST -d '{"address": "172.16.30.30/24"}' http://localhost:8080/router/0000000000000001
curl -X POST -d '{"address":"172.16.10.1/24"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"address": "172.16.30.1/24"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"address": "192.168.10.1/24"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"address": "192.168.30.1/24"}' http://localhost:8080/router/0000000000000003
curl -X POST -d '{"address": "192.168.10.20/24"}' http://localhost:8080/router/0000000000000003
curl -X POST -d '{"gateway": "172.16.30.1"}' http://localhost:8080/router/0000000000000001
curl -X POST -d '{"gateway": "172.16.30.30"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"gateway": "192.168.10.1"}' http://localhost:8080/router/0000000000000003
curl -X POST -d '{"destination": "192.168.30.0/24", "gateway": "192.168.10.20"}' http://localhost:8080/router/0000000000000002
```

Gambar 3.10 Command setting IP dan gateway static routing

```
curl -X POST -d '{"address":"172.16.20.1/24"}' http://localhost:8080/router/0000000000000001/10
curl -X POST -d '{"address":"172.16.10.1/24"}' http://localhost:8080/router/0000000000000002/10
curl -X POST -d '{"address": "192.168.30.1/24"}' http://localhost:8080/router/0000000000000003/10
curl -X POST -d '{"address":"172.16.20.1/24"}' http://localhost:8080/router/0000000000000001/20
curl -X POST -d '{"address":"172.16.10.1/24"}' http://localhost:8080/router/0000000000000002/20
curl -X POST -d '{"address": "192.168.30.1/24"}' http://localhost:8080/router/0000000000000003/20
curl -X POST -d '{"address": "10.0.0.1/24"}' http://localhost:8080/router/0000000000000001/10
curl -X POST -d '{"address": "10.0.0.1/24"}' http://localhost:8080/router/0000000000000001/20
curl -X POST -d '{"address": "10.0.0.2/24"}' http://localhost:8080/router/0000000000000002/10
curl -X POST -d '{"address": "10.0.0.2/24"}' http://localhost:8080/router/0000000000000002/20
curl -X POST -d '{"address": "10.0.0.3/24"}' http://localhost:8080/router/0000000000000003/10
curl -X POST -d '{"address": "10.0.0.3/24"}' http://localhost:8080/router/0000000000000003/20
curl -X POST -d '{"gateway": "10.0.0.2"}' http://localhost:8080/router/0000000000000001/10
curl -X POST -d '{"gateway": "10.0.0.2"}' http://localhost:8080/router/0000000000000001/20
curl -X POST -d '{"gateway": "10.0.0.1"}' http://localhost:8080/router/0000000000000002/10
curl -X POST -d '{"gateway": "10.0.0.1"}' http://localhost:8080/router/0000000000000002/20
curl -X POST -d '{"gateway": "10.0.0.2"}' http://localhost:8080/router/0000000000000003/10
curl -X POST -d '{"gateway": "10.0.0.2"}' http://localhost:8080/router/0000000000000003/20
curl -X POST -d '{"destination": "192.168.30.0/24", "gateway": "10.0.0.3"}' http://localhost:8080/router/0000000000000002/10
curl -X POST -d '{"destination": "192.168.30.0/24", "gateway": "10.0.0.3"}' http://localhost:8080/router/0000000000000002/20
```

Gambar 3.11 Command setting IP dan gateway VLAN 10 dan 20

f. Mengatur segmentasi VLAN pada host

Pada skenario yang membutuhkan pembagian nomor atau segmentasi VLAN perlu dilakukan pengkonfigurasian pada host-host yang ada sesuai dengan topologi pada proses modeling dengan menggunakan xterm atau eksternal terminal yang menggunakan command seperti contoh berikut ini:

1. **ip link add link h1-eth0 name h1-eth0.10 type vlan id 10**

   Command tersebut digunakan untuk menambahkan interface baru yang akan digunakan untuk VLAN 10 dengan nama interface h1-eth0.

2. **ip addr add 172.16.20.11 dev h1-eth0.10**

   Command diatas digunakan untuk menambahkan ip address baru pada interface VLAN yang telah dibuat sebelumnnya.

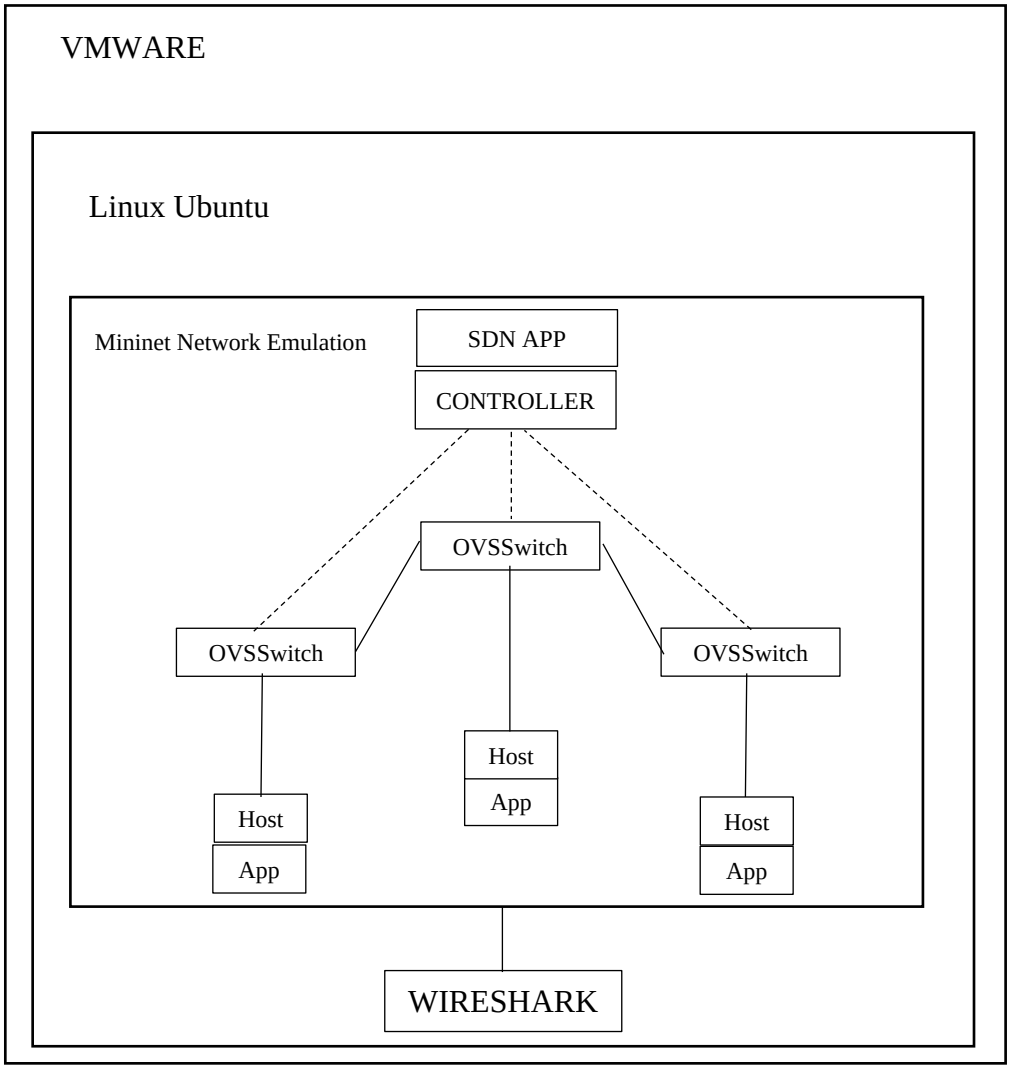
3. **ip link set dev h1-eth0.10 up**

   Setelah melakukan penambahan ip address dengan menggunakan command sebelumnya, command diatas digunakan untuk menghidupkan interface VLAN tersebut.

4. **ip route add default via 172.16.20.1**

   Command diatas merupakan command terakhir yang digunakan yang berfungsi untuk mendefinisikan route pada host tersebut.

3.5 Desain Sistem



Gambar 3.12 Desain sistem

V. SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan penelitian ini diperoleh kesimpulan bahwa:

1. VLAN yang diterapkan pada static routing dan juga tanpa VLAN dapat diimplementasikan dengan baik pada *Software Defined Network* dengan *Ryu Controller* yang diberi *background traffic* 100, 200, 300, dan 400 Mb baik dengan 30 host maupun 60 host.
2. Nilai rata-rata parameter throughput terbaik adalah tanpa VLAN dengan nilai 242,52 Mbps dan masuk kategori *excellent* dengan indeks 4 berdasarkan TIPHON. Nilai rata-rata parameter *packet loss* baik tanpa VLAN atau menggunakan VLAN sama baiknya dan masuk kedalam kategori *excellent* dengan indeks 4 berdasarkan TIPHON. Nilai parameter *delay* terbaik adalah tanpa VLAN dengan nilai rata-rata 0,0000615 s dan masuk kategori *perfect* dengan indeks 4. Nilai jitter terbaik adalah pada tanpa VLAN dengan nilai rata-rata 0,0000593 ms dan masuk kategori *Good* dengan indeks 3.
3. Nilai parameter Qos *Throughput, Packet Loss, delay* dan *Jitter* keseluruhan VLAN kurang baik dibandingkan tanpa VLAN dengan nilai berturut-turut 222,9 Mbps, 0%, 0,0000636 s, dan 0,0000675 ms. Namun seiring bertambahnya host, VLAN lebih dapat diandalkan.

5.2 Saran

Saran terkait penelitian ini adalah:

1. Penggunan *traffic generator* selain iperf seperti D-ITG.
2. Penggunaan simulator lain seperti Omnet++ atau GNS3.
3. Melakukan penelitian menggunakan perangkat dengan spesifikasi yang lebih baik untuk simulator seperti alokasi RAM 4GB atau lebih dan storage lebih besar dari 10GB.
4. Penggunaan topologi jaringan yang lebih kompleks atau berdasarkan jaringan riil.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[1] K. Ahmed, J. Blech, M. Gregory, and H. (Heinz) Schmidt, "Software Defined Networks in Industrial Automation," *Journal of Sensor and Actuator Networks*, vol. 7, p. 33, Aug. 2018, doi: 10.3390/jsan7030033.

[2] K. N. Sharma, "A Scalable and Fault Tolerant OpenFlow Controller," (Thesis, Master of Science (MSc)), University of Waikato, Hamilton, New Zealand, 2015.

[3] M. H. Hidayat, N. R. Rosyid, Y. Sekip, U. Iv, and Y. Indonesia, "Analisis Kinerja dan Karakteristik Arsitektur Software-Defined Network Berbasis OpenDaylight Controller," *Citee, 2085*, vol. 6350, pp. 194–200, 2017.

[4] S. Wongkar, A. A. E. Sinsuw, and X. Najoan, "Analisa Implementasi Jaringan Internet Dengan Menggabungkan Jaringan LAN Dan WLAN Di Desa Kawangkoan Bawah Wilayah Amurang II," *Jurnal Teknik Elektro dan Komputer*, vol. 4, no. Vol. 4 No. 6 (2015): Jurnal Teknik Elektro dan Komputer, Nov. 2015.

[5] H. Nugroho, M. Irfan, and A. Faruq, "Software Defined Networks: a Comparative Study and Quality of Services Evaluation," *Scientific Journal of Informatics*, vol. 6, pp. 181–192, Dec. 2019, doi: 10.15294/sji.v6i2.20585.

[6] ONF, "Software-Defined Networking: The New Norm for Networks." Accessed: Apr. 11, 2022. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[7] Mininet, "Mininet Overview." Accessed: Apr. 10, 2022. [Online]. Available: http://mininet.org/overview/

[8] M. Sood and K. K. Sharma, "Mininet as a Container Based Emulator for Software Defined Networks," Dec. 2014.

[9] Ryu SDN Framework Community, “COMPONENT-BASED SOFTWARE DEFINED NETWORKING FRAMEWORK:Build SDN Agilely.” Accessed: Apr. 10, 2022. [Online]. Available: https://ryu-sdn.org/

[10] S. Asadollahi, B. Goswami, and M. Sameer, “Ryu controller’s scalability experiment on software defined networks,” in *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, 2018, pp. 1–5. doi: 10.1109/ICCTAC.2018.8370397.

[11] A. Khalid, J. J. Quinlan, and C. J. Sreenan, “MiniNAM: A network animator for visualizing real-time packet flows in Mininet,” in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, pp. 229–231. doi: 10.1109/ICIN.2017.7899417.

[12] Mufadhol, *Networking & Internet*. Semarang University Press, 2008.

[13] Ubuntu, “What is Ubuntu?” Accessed: Apr. 10, 2022. [Online]. Available: https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html

[14] S. Ramdan, “PENGARUH MEDIA PEMBELAJARAN BERBASIS VMWARE WORKSTATION DAN MOTIVASI BELAJAR TERHADAP PRESTASI BELAJAR MATA PELAJARAN ADMINISTRASI SERVER KELAS XI TEKNIK KOMPUTER JARINGAN DI SMK SE-KABUPATEN KULON PROGO,” Skripsi, Universitas Negeri Yogyakarta, Yogyakarta, 2015.

[15] N. McKeown *et al.*, “OpenFlow: Enabling innovation in campus networks,” *Computer Communication Review*, vol. 38, pp. 69–74, Apr. 2008, doi: 10.1145/1355734.1355746.

[16] I. Sofana, *Teori dan modul praktikum jaringan komputer*. Bandung: Modula, 2011.

[17] I. Gunawan, “Analisis Keamanan Wifi Menggunakan Wireshark,” *JES (Jurnal Elektro Smart)*, vol. 1, no. 1, Aug. 2021, [Online]. Available: https://www.sttrcepu.ac.id/jurnal/index.php/jes/article/view/158

[18] S. Mahlknecht, S. A. Madani, and J. Kazmi, “Wireless Sensor Networks: Modelling and Simulation,” in *Discrete Event Simulations*, A. Goti, Ed., Rijeka: IntechOpen, 2010, p. Ch. 13. doi: 10.5772/9902.

[19] M. Kabir, *Design a VLAN (Virtual Local Area Network) Based Network*. 2020. doi: 10.13140/RG.2.2.29163.57120.

[20] M. Al-Hamiri and D. Al-Khaffaf, "Performance evaluation of campus network involving VLAN and broadband multimedia wireless networks using OPNET modeler," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 19, pp. 1490–1497, Oct. 2021, doi: 10.12928/TELKOMNIKA.v19i5.18531.

[21] R. Tulloh, R. Negara, and A. Nur Hidayat, "SIMULASI VIRTUAL LOCAL AREA NETWORK (VLAN) BERBASIS SOFTWARE DEFINED NETWORK (SDN) MENGGUNAKAN POX CONTROLLER," *Jurnal Informatika,Telekomunikasi dan Elektronika*, vol. 7, Nov. 2015, doi: 10.20895/infotel.v7i2.134.

[22] K. Bayu, A. Nurcahyo, and A. Prihanto, "ANALISIS QUALITY OF SERVICE (QOS) PADA JARINGAN VLAN (VIRTUAL LOCAL AREA NETWORK)," *Journal of Informatics and Computer Science*, vol. 03, 2021.

[23] W. Amayreh, N. Alqahtani, and B. Al-Balawi, "Analysis of the Vlan Network Delay Performance to Improve Quality of Services (QOS)," *Communications on Applied Electronics*, vol. 5, pp. 51–54, Sep. 2016, doi: 10.5120/cae2016652378.

[24] R. Afan, I. Agus Virgono, and I. R. Rumani M, "ANALISIS EFEK PENGGUNAAN KONTROLER RYU DAN POX PADA PERFORMANSI JARINGAN SDN ANALYISIS OF EFFECT OF CONTROLLER RYU AND POX ON SDN NETWORK PERFORMANCE."