SKRIPSI

IMPLEMENTASI *NETWORK SLICING* MENGGUNAKAN RYU *CONTROLLER* UNTUK MENGONTROL *TRAFFIC DATA PACKET* PADA *SOFTWARE DEFINED NETWORK* (SDN)

Disusun dan Diajukan Oleh:

Sony Akbar Manarang
D041 19 1113

PROGRAM STUDI SARJANA TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2024

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI *NETWORK SLICING* MENGGUNAKAN RYU *CONTROLLER* UNTUK MENGONTROL *TRAFFIC DATA PACKET* PADA *SOFTWARE DEFINED NETWORK* (SDN)

Disusun dan diajukan oleh

Sony Akbar Manarang
D041 19 1113

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Elektro Fakultas Teknik Universitas Hasanuddin
Pada Tanggal 16 Oktober 2024
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr.Eng. Wardi, S.T., M.Eng	Azran Budi Arief, S.T., M.T
NIP. 19720828 199903 1 003	NIP. 19890201 201903 1 007

Ketua Program Studi,

Prof. Dr.-Ing. Ir. Faizal A. Samman, IPU, ACPE, APEC Eng.
NIP. 19750605 200212 1 004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : Sony Akbar Manarang

NIM : D041191113

Program Studi : Teknik Elektro

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul:

IMPLEMENTASI *NETWORK SLICING* MENGGUNAKAN RYU *CONTROLLER* UNTUK MENGONTROL *TRAFFIC DATA PACKET* PADA *SOFTWARE DEFINED NETWORK* (SDN)

Adalah karya tulisan saya sendiri dan bukan merupakan pengambil alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitnya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklasrifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 10 Oktober 2024

Yang Menyatakan

Sony Akbar Manarang

KATA PENGANTAR

Puji syukur terpanjatkan kehadirat Allah subhanahu wata'ala atas limpahan rahmat dan karunia-Nya sehingga penyusunan skripsi tugas akhir ini dapat terselesaikan. Shalawat serta salam tak lupa tercurahkan kepada baginda Rasulullah sallallahu 'alaihi wasallam. Penyelesaian skripsi ini merupakan upaya penulis dalam memenuhi salah satu syarat guna memperoleh gelar Sarjana Teknik di Departemen Teknik Elektro Fakultas Teknik Universitas Hasanuddin.

Skripsi ini penulis persembahkan kepada seluruh pembaca dan pengembang riset selanjutnya sebagai literatur studi Pustaka. Kepada yang terkhusus untuk kedua orang tua dan kakak penulis yang telah mendidik dan senantiasa selalu mendoakan dan mendukung penulis selama ini. Semoga ini menjadi salah satu bentuk wasilah birrul walidayn kepada orang tua dan seuruh keluarga penulis.

Skripsi ini berjudul Implementasi *Network Slicing* Menggunakan Ryu *Controller* untuk Mengontrol *Traffic* Data *Packet* pada *Software Defined Network* (SDN). Pelaksanaan rangkaian penelitian dan penyusunan skripsi ini juga tak terlepas dari seluruh bantuan, bimbingan, nasehat dan doa dari berbagai pihak. Sehubungan dengan hal tersebut, maka penulis hendak mengucapkan terima kasih kepada:

1. Orang tua, saudara serta keluarga penulis yang telah memberikan dukungan baik itu secara material, moral serta doa yang tiada hentinya kepada penulis selama proses pengerjaan tugas akhir serta perkuliahan hingga selesai. Terima kasih atas semangat dan motivasi yang diberikan untuk penulis dapat menyelesaikan pendidikan dibangku perkuliahan.
2. Bapak Prof. Dr.-Ing. Ir. Faizal A. Samman, IPU, ACPE, APEC Eng., selaku Ketua Departemen Teknik Elektro Fakultas Teknik Universitas Hasanuddin
3. Bapak Dr. Eng. Wardi, S.T., M.Eng., selaku Pembimbing 1 dengan kerendahan hati mempercayakan penulis sebagai mahasiswa bimbingan penyelesaian tugas akhir ini, serta bapak Azran Budi Arief, S.T., M.T. selaku Pembimbing II atas bimbingan dan segala bentuk perhatian dan dorongan kepada penulis.
4. Para dosen penguji, Prof. Dr. Ir. Andani Achmad, M.T., dan bapak Ir. Samuel Panggalo, M.T. atas segala saran dan masukan kepada penulis dalam menyelesaikan tugas akhir ini.

5. Seluruh tenaga pendidik dan sivitas akademika Departemen Teknik Elektro dan Fakultas Teknik Unhas atas segala bentuk ketulusan pelayanan selama penulis menempuh perkuliahan.
6. Teman seperjuangan TR19GER, Teknik Elektro Angkatan 2019 yang telah membersamai penulis selama menempuh perkuliahan. Serta para senior dan junior yang tak dapat penulis sebutkan satu-persatu. Barakallahu fiikum.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam skripsi ini, oleh karennya segala bentuk saran dan masukan yang membangun dari semua pihak. Akhir kata penulis berharap semoga skripsi dapat diterima sebagai aktualisasi tri dharma penulis yang dapat memberikan manfaat bagi penulis dan pembacanya.

Gowa, 10 Oktober 2024

Penulis

Sony Akbar Manarang

ABSTRAK

SONY AKBAR MANARANG. *Implementasi Network Slicing Menggunakan Ryu Controller untuk Mengontrol Traffic Data Packet pada Software Defined Network (SDN)* (dibimbing oleh Wardi dan Azran Budi Arief)

Penelitian ini bertujuan untuk mengimplementasikan *Network Slicing* menggunakan Ryu *Controller* pada jaringan *Software Defined Network* (SDN) dan menganalisis pengaruhnya terhadap kinerja jaringan. *Network Slicing* adalah teknik yang membagi jaringan fisik menjadi beberapa jaringan virtual yang dioptimalkan untuk kebutuhan layanan tertentu. Implementasi ini melibatkan pembuatan empat topologi jaringan dengan jumlah *switch* yang berbeda dan empat *host*, menggunakan Mininet dan alat pengukuran seperti Wireshark. Pengukuran kinerja mencakup *throughput*, *delay*, *jitter*, dan keandalan pada setiap segmen jaringan. Hasil penelitian menunjukkan bahwa *Network Slicing* meningkatkan *throughput* pada *host* dengan prioritas tinggi seperti layanan *streaming* video, meskipun terdapat penurunan *throughput* yang tidak signifikan pada *host* lain. Selain itu, *Network Slicing* cenderung meningkatkan *delay* pada beberapa *host*, terutama pada layanan VoIP dan *streaming* video, namun peningkatan ini masih dalam batas yang dapat diterima. Penggunaan *Network Slicing* juga meningkatkan *jitter* pada beberapa *host*, terutama layanan VoIP, namun dalam beberapa kasus justru menurunkan *jitter* pada layanan *streaming* video. Secara keseluruhan, *Network Slicing* memungkinkan alokasi sumber daya yang lebih efisien dan efektif sesuai kebutuhan layanan, meningkatkan kualitas layanan dengan memprioritaskan QoS yang lebih tinggi. Implementasi *Network Slicing* pada jaringan SDN memberikan dampak positif terhadap performa jaringan, terutama dalam hal *throughput* dan *jitter*, dengan peningkatan *delay* yang masih dalam batas yang dapat diterima. Hasil ini menunjukkan potensi besar *Network Slicing* dalam mengoptimalkan kinerja jaringan SDN.

Kata Kunci: *Software Defined Network*, *Network Slicing*, Ryu *Controller*, Mininet.

ABSTRACT

SONY AKBAR MANARANG. *Implementation of Network Slicing Using Ryu Controller to Manage Data Packet Traffic in Software Defined Networks (SDN)* (*Supervised by Wardi and Azran Budi Arief)*

This study aims to implement Network Slicing using the Ryu Controller on a Software Defined Network (SDN) and analyze its impact on network performance. Network Slicing is a technique that divides a physical network into multiple virtual networks optimized for specific service requirements. The implementation involves creating four network topologies with different numbers of switches and four hosts, using Mininet and measurement tools such as Wireshark. Performance metrics include throughput, delay, jitter, and reliability in each network segment. The results show that Network Slicing improves throughput for high-priority hosts, such as video streaming services, although there is a negligible decrease in throughput for other hosts. Additionally, Network Slicing tends to increase delay for some hosts, especially VoIP and video streaming services, but this increase remains within acceptable limits. The use of Network Slicing also increases jitter for some hosts, particularly VoIP services, but in some cases, it reduces jitter for video streaming services. Overall, Network Slicing allows for more efficient and effective resource allocation according to service needs, enhancing service quality by prioritizing higher QoS requirements. The implementation of Network Slicing in SDN networks positively impacts network performance, especially in terms of throughput and jitter, with an increase in delay that remains within acceptable limits. These findings highlight the significant potential of Network Slicing in optimizing SDN network performance.

Keywords: Software Defined Network, Network Slicing, Ryu Controler, Mininet.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Software Defined Network (SDN) adalah konsep pendekatan jaringan baru dalam mendesain, mengelola dan mengimplementasi jaringan terutama untuk mendukung kebutuhan dan inovasi dalam jaringan komputer semakin kompleks. Arsitektur SDN memberikan kemudahan kepada pengguna dalam mengembangkan aplikasi pengontrol jaringan dengan memisahkan fungsi *control plane* dan *data plane*. Konsep utama pada SDN adalah pengaturan pengontrolan jaringan berada pada *control plane*, hal ini memudahkan administrator jaringan dalam mengelola dan mengontrol secara langsung lalu lintas jaringan (Caroline, 2023).

Peningkatan lalu lintas jaringan disebabkan oleh semakin banyaknya aplikasi dan layanan berbasis teknologi informasi dalam jaringan komputer. Layanan ini dapat diakses dan digunakan oleh pengguna melalui perangkat desktop dan mobile. Dari perspektif jaringan komputer, lapisan jaringan menjadi tulang punggung yang mendukung kelancaran lalu lintas paket data dan lalu lintas jaringan yang semakin padat.

Peningkatan pesat dalam lalu lintas jaringan menyebabkan jaringan komputer menjadi semakin padat. Kondisi ini dapat menimbulkan berbagai masalah dalam penyediaan aplikasi dan layanan online berbasis teknologi informasi yang tersedia di jaringan komputer, terutama jika masih menggunakan model jaringan konvensional tanpa adanya pengukuran kualitas layanan. Oleh karena itu, diperlukan implementasi teknologi terbaru dalam jaringan komputer untuk mengatasi masalah ini.

Pada SDN perangkat keras jaringan tidak lagi mengatur jalur data secara manual tetapi tugas tersebut diberikan kepada *control plane* menggunakan *controller*. *Controller* berperan sebagai penyedia otomasi terpusat, yang dapat diprogram untuk mengelola, mengkonfigurasi, memantau dan memecahkan masalah infrastruktur jaringan virtual.

Network slicing adalah teknik yang memungkinkan operator jaringan untuk mengisolasi dan mengendalikan lalu lintas data untuk berbagai layanan atau pengguna dalam satu infrastruktur fisik. Dengan *network slicing* satu jaringan memungkinkan pembagian menjadi beberapa potongan virtual yang dapat diatur secara terpisah. Hal ini memungkinkan jaringan untuk mendukung sejumlah aplikasi dan layanan dengan persyaratan yang berbeda-beda, termasuk QoS yang beragam (Barakabitze, 2020). Ryu *Controller* adalah komponen penting dalam arsitektur SDN yang menyediakan antarmuka program aplikasi (API) untuk pengembangan aplikasi manajemen jaringan. Penggunaan Ryu *Controller* dalam implementasi *Network Slicing* memberikan fleksibilitas dan kemudahan dalam pengaturan prioritas layanan dan alokasi *bandwidth*. Keunggulan Ryu *Controller* seperti kemudahan penggunaan dan fleksibilitas menjadikannya alat yang ideal untuk penelitian ini.

Penelitian ini akan membahas implementasi *Network Slicing* menggunakan Ryu *Controller* untuk mengontrol *traffic data packet* pada *Software Defined Network* (SDN). Mininet digunakan sebagai alat simulasi jaringan untuk menguji dan mengevaluasi implementasi *Network Slicing*. Dengan Mininet, topologi jaringan dapat dibuat dan diuji secara *virtual*, memungkinkan pengujian berbagai skenario jaringan secara efisien dan realistis. Tujuan dari penelitian ini adalah meningkatkan efisiensi pengelolaan *traffic data*, meningkatkan alokasi sumber daya jaringan, dan menyediakan pengalaman pengguna yang lebih baik. Penelitian ini diharapkan dapat memberikan kontribusi pada pengembangan teknologi SDN dan penerapan *network slicing* dalam pengelolaan *traffic data* pada jaringan. Dengan demikian, akan memungkinkan operator jaringan untuk lebih baik mengelola lalu lintas data dan memberikan layanan yang lebih baik kepada pengguna akhir.

1.2 Rumusan Masalah

1. Bagaimana caranya mengimplementasikan *Network Slicing* dengan Ryu *Controller* dalam Jaringan SDN?
2. Bagaimana pengaruh implementasi *Network Slicing* terhadap kinerja dan efisiensi jaringan SDN?

1.3 Tujuan Penelitian

1. Melakukan implementasi *Network Slicing* menggunakan Ryu *Controller* dalam Jaringan SDN.
2. Menganalisis pengaruh implementasi *Network Slicing* terhadap kinerja jaringan SDN.

1.4 Manfaat Penelitian

1. Memberikan pemahaman yang lebih mendalam tentang implementasi *Network Slicing* dalam SDN.
2. Memberikan pemahaman yang lebih baik tentang penggunaan Ryu *Controller* dalam implementasi *Network Slicing*.

1.5 Ruang Lingkup

1. Penelitian ini difokuskan pada implementasi *Network Slicing* dengan menggunakan Ryu *Controller* dalam lingkup jaringan SDN tertentu.
2. Evaluasi kinerja Ryu *Controller* akan difokuskan pada parameter *throughput, delay, jitter,* dan keandalan dalam konteks *Network Slicing.*
3. Topologi Jaringan terbatas pada model tertentu.
4. Segmentasi layanan pada tingkat aplikasi seperti layanan VoIP, *streaming video* dan lalu lintas data.

1.6 Sistematika Penulisan

BAB I Pendahuluan

Bab ini berisi tentang penguraian secara singkat Latar Belakang, Rumusan Masalah, Tujuan Penelitian, Manfaat Penelitian, Ruang Lingkup dan Sistematika Penulisan.

BAB II Tinjauan Pustaka

Bab ini berisi penjelasan tentang teori penunjang yang relevan untuk bahan penelitian yang diperoleh dari sumber referensi untuk menyusun laporan tugas akhir ini.

BAB III Metodelogi Penelitian

Bab ini membahas mengenai rancangan penelitian, waktu dan tempat penelitian, diagram alir perencanaan, alat dan bahan yang digunakan, metode penelitian dan analisis data serta skenario penelitian pada tugas akhir ini.

BAB IV Hasil dan Pembahasan

Bab ini membahas mengenai hasil dan pembahasan analisis penelitian yang dlikakukan pada tugas akhir ini.

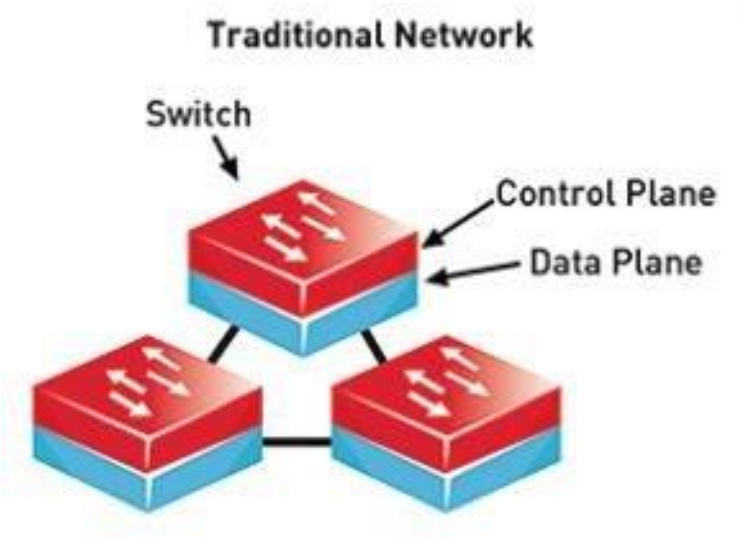
BAB V Kesimpulan dan Saran

Bab ini merupakan penutup yang berisi kesimpulan dari penelitian yang dilakukan serta saran untuk pengembangan yang berguna pada studi lanjut tugas akhir berikutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Konsep Dasar *Software Defined Network* (SDN)

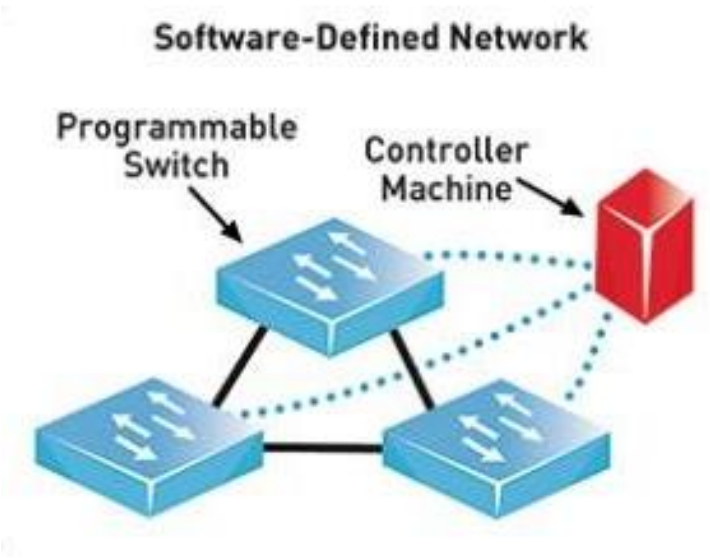
Konsep dalam *network layer* merupakan contoh dari abstraksi jaringan sebelum adanya konsep *Software Defined Networking*. Konsep *layer* ini juga hanya terbatas untuk mengabstraksi-kan data *plane*, tidak ada konsep yang mewakili *control plane*. Sebenarnya tidak ada yang salah dengan ini, namun akan menjadi tugas tambahan apabila nanti jaringan sudah semakin kompleks seperti pada Gambar 1.



Gambar 1 Konsep jaringan tradisional

Dengan menggunakan metode ini, seiring dengan berjalannya waktu, nantinya akan semakin banyak mekanisme (protokol) yang menyebabkan jaringan berkembang semakin kompleks dan semakin sulit untuk dikonfigurasi atau dikelola, bahkan akan menghalangi dan memperlambat inovasi teknologi karena begitu kompleksnya mekanisme seperti Gambar 2. *Software Defined Networking* bekerja dalam sebuah *controller* (point of management) yang sangat memungkinkan sebuah entitas aplikasi dapat mengendalikan seluruh proses komunikasi dalam beberapa resources jaringan, mengendalikan *traffic* yang

melewati perangkat jaringan, serta melakukan suatu inspeksi khusus atau modifikasi terhadap jaringan yang melewatinya.



Gambar 2 Konsep jaringan modern (SDN)

Konsep Sofware-*Defined Network* pertama kali diperkenalkan oleh Martin Casado di Universitas Stanford pada tahun 2007 dengan tulisan pada jurnalnya berjudul "Ethane: Taking *Control* of the Enterprise". Pada jurnal tersebut dijelaskan bahwa ethane merupakan suatu arsitektur baru untuk suatu perusahaan yang mengijinkan manajer mendefinisikan luasan dari sebuah jaringan, kebijakan didalamnya, serta menjalankan hal-hal tersebut secara langsung. Ethane adalah sebuah konsep penyederhanaan yang cukup ekstrim dari ethernet *switch* dengan sebuah *control*er terpusat yang dapat mengatur hak masuknya data dan aliran *routing*.

Software-Defined Networking (SDN) adalah sebuah konsep baru yang memungkinkan pengelola jaringan untuk mengelola router dan *switch* secara fleksibel menggunakan *software* yang berjalan pada *server* eksternal. Open *Network* Foundation, mendefinisikan SDN adalah sebuah sebuah arsitektur jaringan baru di mana kontrol jaringan dipisahkan dari *forwarding plane* dan dapat diprogram secara langsung.

Programmable switch adalah jenis *switch* jaringan yang dapat dikontrol dan dikonfigurasi secara dinamis melalui *controller* menggunakan protokol standar

seperti OpenFlow. Tidak seperti *switch* tradisional yang menggunakan *firmware* atau *hardware* yang sudah tetap untuk menentukan bagaimana paket data ditangani, *programmable switch* memberikan fleksibilitas untuk mengubah aturan *forwarding*, *routing*, pengelolaan trafik, dan QoS secara *real-time* berdasarkan kebijakan jaringan yang ditentukan oleh *controller* SDN.

Software Defined Network merupakan jaringan pintar yang memiliki sentralisasi dalam hal *logical software* (*software-base*), selain itu ONF menyatakan bahwa dengan SDN tidak lagi membutuhkan protokol standar, tetapi cukup hanya menerima instruksi dari sebuah SDN *controller*. (Thomas, 2018).

Fungsi dan Kegunaan SDN

- Manajemen Terpusat: SDN memungkinkan pengelolaan jaringan dari satu titik kontrol terpusat, sehingga memudahkan administrasi dan pengawasan jaringan.
- Fleksibilitas dan Skalabilitas: SDN memungkinkan penyesuaian cepat terhadap perubahan kebutuhan jaringan, seperti menambah atau mengurangi kapasitas jaringan, tanpa perlu mengubah perangkat keras secara fisik.
- Otomatisasi: Dengan SDN, banyak proses jaringan yang sebelumnya manual dapat diotomatisasi, seperti pengaturan jalur lalu lintas, pemantauan kinerja, dan penerapan kebijakan keamanan.
- Penghematan Biaya: SDN mengurangi kebutuhan akan perangkat keras yang mahal dan memungkinkan penggunaan perangkat yang lebih sederhana dan terstandarisasi.
- Peningkatan Keamanan: SDN memungkinkan penerapan kebijakan keamanan yang lebih canggih dan terpusat, serta pemantauan yang lebih efektif terhadap ancaman keamanan jaringan.

Kelebihan dan Kekurangan SDN

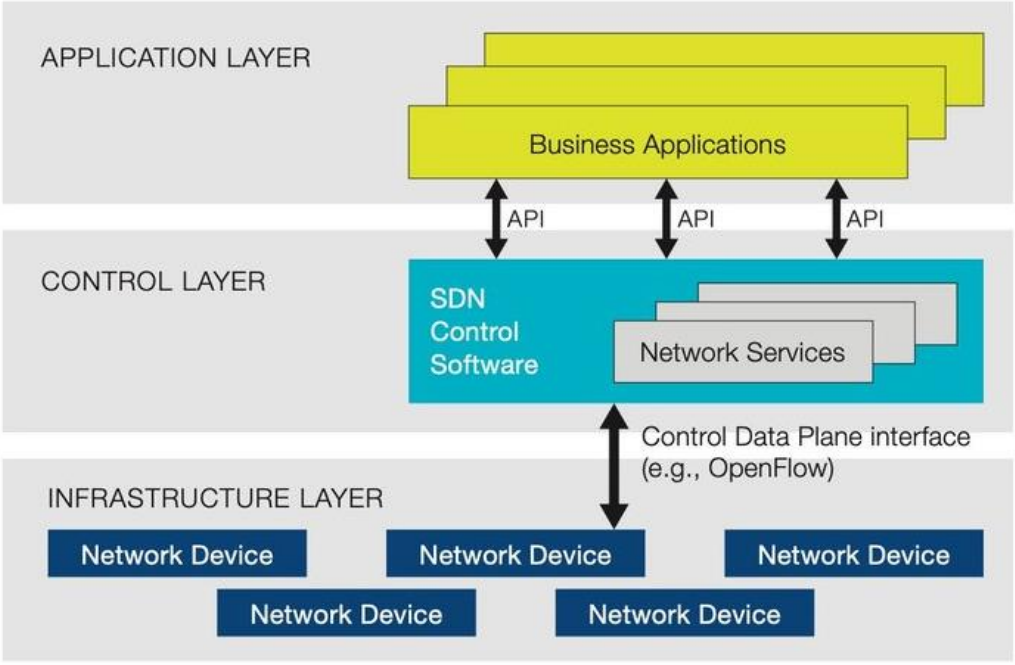
Kelebihan:

- Peningkatan Efisiensi: Memungkinkan pemanfaatan sumber daya jaringan yang lebih efisien dan responsif terhadap perubahan kebutuhan.
- Pengembangan Aplikasi: Mempermudah pengembangan dan pengujian aplikasi jaringan baru dengan memisahkan lapisan kontrol dan data.

- Pengendalian Terpusat: Memungkinkan pengelolaan yang lebih baik dan lebih terintegrasi dari keseluruhan infrastruktur jaringan.

  Kekurangan:
- Kompleksitas Implementasi: Memerlukan pengetahuan khusus dan perangkat lunak tambahan untuk mengelola dan mengimplementasikan SDN.
- Kebergantungan pada *Controller*: Kegagalan pada SDN *controller* dapat menyebabkan gangguan besar pada jaringan jika tidak ada mekanisme redundansi yang memadai.
- Keamanan: Meskipun menawarkan peningkatan keamanan, SDN juga menghadirkan tantangan baru dalam hal keamanan, seperti risiko serangan pada *controller* pusat.

Arsitektur SDN memiliki 3 *layer* yaitu: *Application layer*, *Control layer*, dan *Infrastructure layer*. Ketiga *layer* tersebut mempunyai fungsi yang berbeda-beda. Berikut gambar arsitektur secara umum *Software Defined Network*



Gambar 3 Arsitektur *software defined network*

a. *Application Layer*: *Application layer* adalah lapisan tertinggi dalam arsitektur SDN yang memungkinkan interaksi langsung antara administrator jaringan atau aplikasi pengguna dengan jaringan itu sendiri. Lapisan ini biasanya terdiri dari berbagai aplikasi yang digunakan untuk mengelola dan mengoptimalkan jaringan,

seperti *traffic engineering*, pengelolaan QoS (*Quality of service*), dan keamanan jaringan.

Pada layer ini, aplikasi berkomunikasi dengan control layer melalui API (*Application Programming Interface*) standar yang memungkinkan pengelola jaringan untuk berinteraksi dengan *controller* SDN tanpa perlu memahami detail infrastruktur jaringan yang mendasarinya. Aplikasi di *application layer* memungkinkan administrator untuk memantau jaringan, melakukan penyesuaian otomatis, serta memberikan kebijakan yang ditetapkan pada *control layer* untuk diterapkan ke lapisan infrastruktur.

b. *Control Layer*: *Control Layer* adalah lapisan menengah yang bertindak sebagai otak dari SDN. Lapisan ini mengontrol seluruh aktivitas jaringan dengan mengelola dan mengkoordinasi bagaimana data diproses dan dialirkan melalui jaringan. Pada layer ini, *controller* SDN (seperti Ryu, ONOS, atau OpenDaylight) mengimplementasikan *control plane*, yang bertanggung jawab untuk membuat keputusan terkait routing, *forwarding*, pengaturan sumber daya jaringan, serta penerapan kebijakan QoS (Kreutz, 2015).

   Fungsi *control plane* di *control layer* juga memungkinkan jaringan untuk bersifat dinamis dan dapat diprogram, berbeda dengan jaringan tradisional yang sifatnya statis. *control layer* bertindak sebagai perantara antara *application layer* dan *infrastructure layer*, mengubah kebijakan yang diberikan oleh aplikasi menjadi perintah yang dapat dipahami oleh perangkat jaringan di lapisan infrastruktur.

c. *Infrastructure Layer*: *Infrastructure layer* merupakan lapisan terbawah dalam arsitektur SDN, yang terdiri dari perangkat keras fisik seperti *switch*, *router*, dan server yang melakukan *forwarding* data berdasarkan instruksi dari *controller* SDN. Perangkat-perangkat di lapisan ini hanya menjalankan fungsi data *plane*, yang artinya mereka hanya bertanggung jawab untuk memproses dan mengirimkan paket data, sedangkan semua logika pengendalian dikelola oleh *controller* di *control layer*.

   Dalam jaringan tradisional, perangkat di lapisan infrastruktur ini juga melakukan fungsi *control plane* (misalnya, ***routing*** dan *forwarding*). Namun, dalam arsitektur SDN, fungsi-fungsi tersebut dipindahkan ke lapisan *control layer* sehingga perangkat di *infrastructure layer* hanya bertindak sebagai "*forwarding elements*"

yang sederhana. OpenFlow adalah salah satu protokol yang memungkinkan komunikasi antara *controller* dan perangkat infrastruktur seperti *switch* dan *router*, memastikan bahwa perintah yang diberikan *controller* dilaksanakan oleh perangkat keras tersebut (Friwansya, 2018).

2.2 *Network Slicing*

Teknologi canggih diharapkan menghadapi lalu lintas jaringan yang padat ketika sejumlah besar pengguna meminta layanan dan aplikasi berbeda pada waktu yang berbeda sesuai keinginan pengguna. Untuk meningkatkan situasi ini, konsep *network slicing* dan ketersediaan potongan jaringan menyediakan dukungan yang diperlukan dengan teknologi SDN terkini. *Network slicing* merupakan konsep yang digunakan untuk menyiapkan beberapa jaringan *logic* (*network slice*s) dengan tujuan menawarkan satu set layanan pada jaringan fisik yang sama. Infrastruktur jaringan dipartisi menjadi beberapa jaringan virtual, di mana setiap *slice* dapat memiliki arsitektur, aplikasi, paket, dan kapasitas pemrosesan sinyal sendiri, pada satu jaringan yang sama. Hal ini memungkinkan setiap *network slice*s dirancang untuk menjamin tingkat kinerja yang berbeda. Namun, secara pasti menjamin kinerja dari *network slice*s yang berbagi jaringan fisik yang sama bukanlah hal yang mudah. Dalam konteks *Software Defined Network* (SDN), *network slicing* menjadi lebih fleksibel. SDN memisahkan lapisan kontrol dan data dalam jaringan, memungkinkan pengaturan dan konfigurasi jaringan yang dinamis. Setiap potongan jaringan (*network slice*) dapat memiliki kebijakan, karakteristik, dan fungsi yang berbeda, yang dikendalikan melalui kontrol pusat (SDN *Controller*).

2.3 *Traffic Data Packet*

Traffic data packet adalah unit fundamental dari data yang dikirim melalui jaringan komputer. Paket data ini terdiri dari informasi yang dibagi menjadi bagian-bagian kecil yang dapat diangkut melalui jaringan dan kemudian digabungkan kembali di tujuan akhir. Setiap paket data biasanya terdiri dari dua bagian utama: header dan payload. Header berisi informasi kontrol seperti alamat sumber dan tujuan, sedangkan payload adalah data aktual yang dikirimkan.

Pengontrolan *traffic* data *packet* pada SDN melibatkan manipulasi aliran lalu lintas melalui *controller* SDN. Konsep ini memungkinkan pengaturan prioritas, segmentasi layanan, dan penerapan kebijakan QoS secara dinamis. Pengendalian lalu lintas ini terjadi pada tingkat *controller*, memungkinkan pengguna untuk menyesuaikan alur lalu lintas sesuai dengan kebutuhan spesifik mereka. Jenis-Jenis *Traffic* Data *Packet*

- TCP (Transmission *Control* Protocol): Menggunakan koneksi yang andal untuk memastikan pengiriman paket yang tepat. TCP memastikan paket diterima dan disusun dalam urutan yang benar, mengirim ulang paket yang hilang.
- UDP (User Datagram Protocol): Tidak memerlukan koneksi dan mengirimkan paket secara independen tanpa memastikan penerimaan atau urutan yang benar. Lebih cepat tetapi kurang andal dibandingkan TCP.

2.4 *Quality of service* (QoS)

*Quality of service (*QoS) adalah metode untuk mengukur seberapa baik suatu jaringan berfungsi dan merupakan usaha untuk mendefinisikan karakteristik serta sifat dari sebuah layanan. QoS digunakan untuk mengukur berbagai atribut kinerja yang telah ditentukan dan diasosiasikan dengan suatu layanan. Model Monitoring QoS terdiri dari beberapa komponen, yaitu aplikasi monitoring, monitoring QoS, monitor, dan objek yang dimonitor. Aplikasi monitoring berfungsi sebagai antarmuka bagi administrator jaringan. Komponen ini mengumpulkan informasi lalu lintas paket data dari monitor, menganalisisnya, dan mengirimkan hasil analisis kepada pengguna. QoS Monitoring menyediakan mekanisme untuk memantau QoS dengan mengambil nilai-nilai parameter QoS dari lalu lintas paket data. Monitor melakukan pengukuran aliran paket data secara real-time dan melaporkan hasilnya kepada aplikasi monitoring.

Kemampuan QoS mengacu pada tingkat kecepatan dan keandalan penyampaian berbagai jenis beban data dalam suatu komunikasi. Kemampuan ini merupakan kumpulan dari beberapa parameter teknis, yaitu:

- *Throughput*

Throughput adalah kecepatan transfer data yang efektif, diukur dalam bps (bit per second). *Throughput* adalah jumlah total paket yang berhasil diterima di tujuan selama interval waktu tertentu, dibagi dengan durasi interval waktu tersebut. Berikut merupakan persamaan perhitungan *throughput.*

$$Throughput = \frac{paket\ data\ diterima}{lama\ pengamatan} \quad (1)$$

- *Delay*

Delay adalah waktu yang dibutuhkan data untuk berpindah dari asal ke tujuan. *Delay* dapat dipengaruhi oleh faktor seperti jarak, media fisik, kemacetan, atau waktu proses yang lama. Persamaan perhitungan *delay.*

$$Delay = \frac{waktu\ pengiriman\ paket}{total\ paket\ yang\ diterima} \quad (2)$$

- *Jitter*

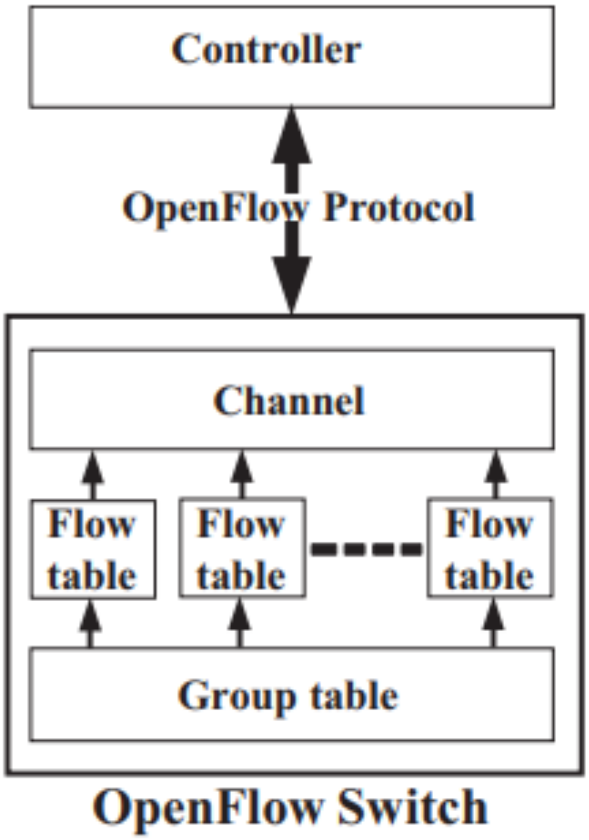
Jitter adalah variasi dalam waktu kedatangan paket. *Jitter* disebabkan oleh variasi dalam panjang antrian, waktu pemrosesan data, dan waktu pengumpulan ulang paket di akhir perjalanan. *Jitter* biasanya disebut sebagai variasi *delay* dan berkaitan erat dengan latensi, yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan. Berikut merupakan persamaan perhitungan *jitter.*

$$Jitter = \frac{total\ variasi\ delay}{total\ paket\ yang\ diterima - 1} \quad (3)$$

2.5 OpenFlow

OpenFlow adalah protokol komunikasi yang memungkinkan perangkat jaringan untuk mengirimkan dan menerima informasi dari *controller* SDN. Protokol ini memungkinkan pengguna untuk mengontrol alur lalu lintas jaringan dan mengatur perangkat jaringan yang terhubung ke dalam jaringan SDN. OpenFlow memiliki beberapa platform pendukung yang memungkinkan pengguna untuk mengembangkan dan mengujicobakan sistem jaringan yang besar, seperti Mininet. OpenFlow memungkinkan pengguna untuk mengontrol alur lalu lintas jaringan dan mengatur perangkat jaringan yang terhubung ke dalam jaringan SDN.

Sebagai protokol SDN standar, OpenFlow diusulkan oleh Stanford. Mengenai pengujian OpenFlow, banyak desain telah diusulkan untuk protokol OpenFlow. Mereka menggunakan kode *open source* untuk mengontrol pengendali dan *switch* SDN universal. Mengenai *switch*, *OpenVSwitch* (OVS) adalah salah satu *switch* OpenFlow berbasis perangkat lunak yang paling populer. OpenFlow adalah protokol yang berorientasi pada aliran dan memiliki abstraksi *switch* dan *port* untuk mengontrol aliran. Dalam SDN, terdapat perangkat lunak bernama *controller* yang mengelola kumpulan *switch* untuk pengendalian lalu lintas. *Controller* berkomunikasi dengan *switch* OpenFlow dan mengelola *switch* melalui protokol OpenFlow. *Switch* OpenFlow dapat memiliki beberapa tabel aliran, tabel grup, dan *port* OpenFlow. Pada awalnya jalur data dari perangkat perutean OpenFlow memiliki tabel perutean kosong dengan beberapa bidang (seperti alamat IP sumber, jenis QoS, dll.).



Gambar 4 OpenFlow model

Tabel ini berisi beberapa bidang paket seperti tujuan port yang berbeda (penerimaan atau transmisi), serta bidang tindakan yang berisi kode untuk tindakan yang berbeda, seperti penerusan atau penerimaan paket, dll. Tabel ini dapat diisi berdasarkan paket data yang masuk. Ketika sebuah paket baru diterima yang tidak memiliki entri yang cocok dalam tabel aliran data, paket tersebut akan diteruskan ke pengontrol untuk diproses. Pengontrol bertanggung jawab atas keputusan penanganan paket, misalnya, sebuah paket dibuang, atau entri baru ditambahkan ke dalam tabel aliran data tentang bagaimana menangani paket ini dan paket serupa yang diterima di masa depan. (Hu, 2014)

Setiap tabel aliran berisi entri aliran dan berkomunikasi dengan *controller*, dan tabel grup dapat mengonfigurasi entri aliran. *Switch* OpenFlow terhubung satu sama lain melalui port OpenFlow. Berikut adalah cara kerja OpenFlow:

- *Switch*: *Switch* dalam jaringan SDN diatur untuk mengirimkan paket ke *controller* SDN ketika paket tersebut tidak dapat diidentifikasi atau tidak sesuai dengan aturan yang ada di dalam *switch*.
- *Controller*: *Controller* SDN menerima paket dari *switch* dan mengirimkan instruksi ke *switch* untuk mengatur alur lalu lintas jaringan. *Controller* SDN juga dapat mengatur alur lalu lintas jaringan dengan mengirimkan instruksi ke *switch* untuk mengatur alur lalu lintas jaringan.
- Flow table: Setiap *switch* dalam jaringan SDN memiliki flow table yang berisi aturan-aturan yang digunakan untuk mengatur alur lalu lintas jaringan. Aturan-aturan ini dapat diatur oleh *controller* SDN.
- Paket: Ketika paket datang ke *switch*, *switch* akan mencocokkan paket dengan aturan yang ada di dalam flow table. Jika tidak ada aturan yang cocok, *switch* akan mengirimkan paket ke *controller* SDN untuk diatur.

OpenFlow juga memungkinkan pengguna untuk menyesuaikan pola jaringan dengan mudah dan menyesuaikan konfigurasi jaringan sesuai dengan perubahan dan kebutuhan.

2.6 Ryu *Controller*

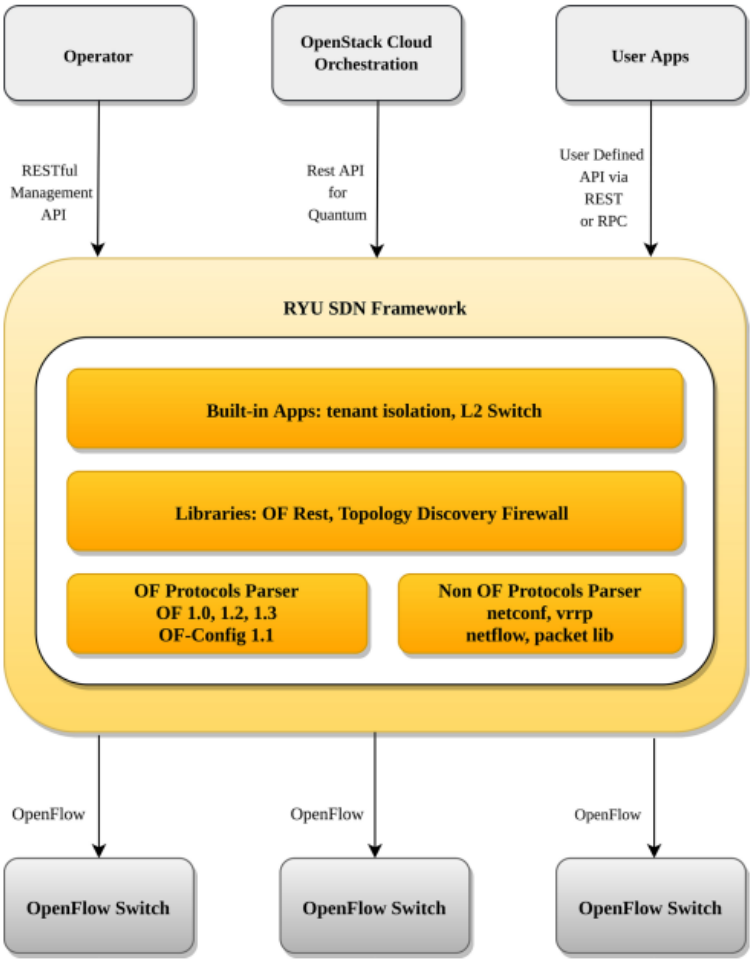
Dalam model SDN, platform *controller* merupakan pilar penting dari arsitektur, dan dengan demikian, upaya dikhususkan untuk mengubah *controller* SDN

menjadi perangkat lunak yang berkinerja tinggi, dapat diskalakan, terdistribusi, modular, dan sangat mudah digunakan oleh programmer. Platform pengontrol terdistribusi, khususnya, harus mengatasi berbagai tantangan. Yang perlu dipertimbangkan secara khusus adalah latensi antara perangkat penerusan dan contoh pengontrol, toleransi kesalahan, penyeimbangan beban, konsistensi, dan sinkronisasi, di antara isu-isu lainnya. Operator juga harus dapat mengamati dan memahami bagaimana kombinasi fungsi dan modul yang berbeda dapat berdampak pada jaringan mereka. Ketika komunitas SDN belajar dari pengalaman pengembangan dan operasional dengan pengontrol OpenFlow (misalnya, Beacon), kemajuan lebih lanjut diharapkan dalam hal kinerja mentah implementasi pengontrol, termasuk eksploitasi desain hierarki dan ukuran buffer yang dioptimalkan. (Kreutz, 2015)

RYU adalah kerangka kerja SDN berbasis komponen yang dikembangkan oleh Nippon Telegraph and Telephone (NTT), RYU adalah pengontrol SDN yang menggunakan bahasa pyhton. *Controller* Ryu menyediakan komponen perangkat lunak dengan API yang terdefinisi dengan baik yang memudahkan pengembang untuk membuat aplikasi manajemen dan kontrol jaringan baru. Ryu *controller* memiliki berbagai fitur yang berguna untuk pengembangan dan pengujian jaringan SDN, termasuk kemampuan untuk membuat aplikasi kontrol jaringan yang disesuaikan, dukungan untuk protokol OpenFlow, dan integrasi dengan berbagai perangkat jaringan. *Controller* ini juga memiliki dokumentasi yang lengkap dan aktif dikembangkan oleh komunitas pengguna dan pengembang. Versi yang digunakan dalam penelitian ini adalah Ryu *Controller* 4.34. Versi ini merupakan rilis stabil yang telah terbukti handal dalam berbagai implementasi jaringan SDN. Dalam penelitian ini, Ryu *Controller* digunakan untuk mengimplementasikan *network slicing* pada jaringan SDN. Dengan menggunakan API yang disediakan oleh Ryu, penelitian ini dapat mengatur prioritas dan alokasi *bandwidth* untuk berbagai jenis lalu lintas data seperti VoIP, *streaming video*, dan data umum.

Pengontrol SDN RYU memiliki tiga lapisan. Lapisan teratas terdiri dari aplikasi logika bisnis dan jaringan yang dikenal sebagai lapisan aplikasi. Lapisan tengah terdiri dari layanan jaringan yang dikenal sebagai lapisan kontrol atau

kerangka kerja SDN, dan lapisan bawah terdiri dari perangkat fisik dan virtual yang dikenal sebagai lapisan infrastruktur. Lapisan tengah menjadi tuan rumah API *northbound* dan API *southbound*. Pengontrol mengekspos API *northbound* terbuka seperti API manajemen *Restful*, REST, API untuk *Quantum*, API yang dideklarasikan pengguna melalui REST atau RPC, yang digunakan oleh aplikasi. RYU menyediakan sekelompok komponen seperti *OpenStack Quantum*, *Firewall*, OFREST, dll. yang berguna untuk aplikasi SDN. Tujuan dari aplikasi-aplikasi ini adalah untuk mengumpulkan intelijen jaringan dengan menggunakan pengontrol, melakukan analisis dengan menjalankan algoritma, dan kemudian mengatur aturan baru dengan menggunakan pengontrol.



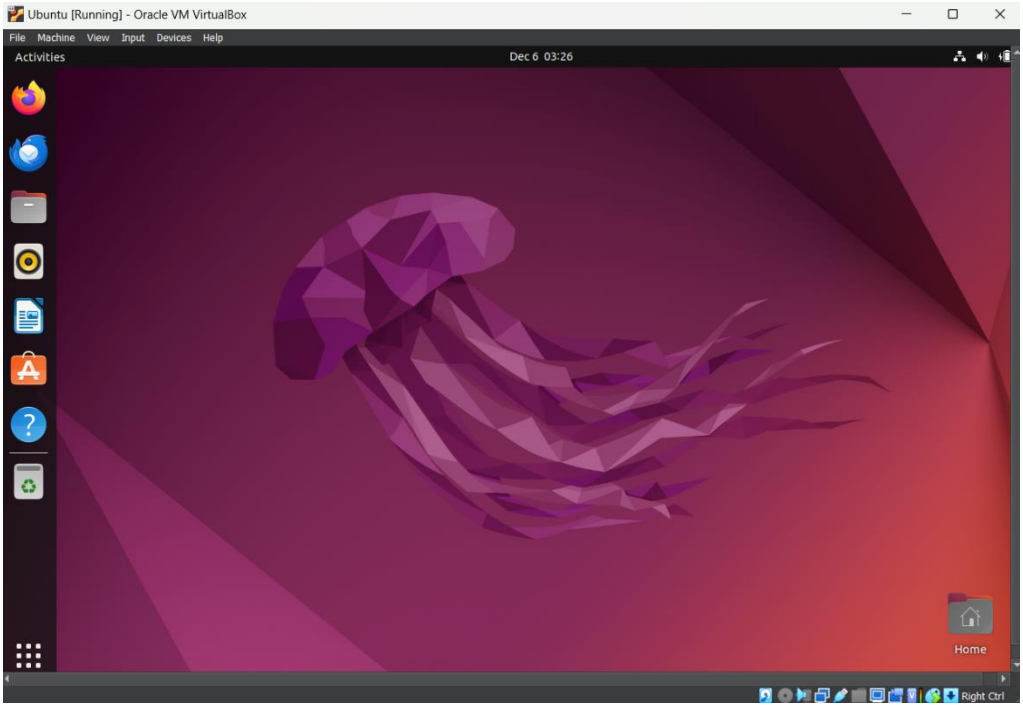
Gambar 5 Arsitektur ryu SDN *controller*

Antarmuka southbound mampu mendukung beberapa protokol seperti OpenFlow, Netconf, OF-confg, dll. RYU menggunakan OpenFlow untuk

berinteraksi dengan bidang penerusan (sakelar dan router) untuk memodifikasi bagaimana jaringan akan menangani aliran trafik. Ini telah diuji dan disertifikasi untuk bekerja dengan beberapa *switch* OpenFlow, termasuk Openv*Switch* dan penawaran dari Centec, Hewlett Packard, IBM, dan NEC

2.7 Ubuntu

Ubuntu adalah sistem operasi berbasis Linux yang bersifat open source. Linux Ubuntu OS didasarkan pada distribusi Debian Linux dan dikembangkan oleh Canonical Ltd, perusahaan yang didirikan oleh entrepreneur Afrika Selatan Mark Shuttleworth. Sistem operasi ini pertama kali dirilis pada tahun 2004 dan sejak itu telah menjadi salah satu distribusi Linux paling populer di dunia. Versi yang digunakan dalam penelitian ini adalah Ubuntu 22.04, dengan kode nama "Jammy Jellyfish". Ubuntu 22.04 digunakan sebagai sistem operasi utama untuk menjalankan semua alat dan perangkat lunak yang diperlukan untuk implementasi dan pengujian *network slicing* menggunakan Ryu *Controller* dalam jaringan SDN. Kestabilan dan dukungan perangkat lunak yang luas dari Ubuntu 22.04 memastikan bahwa lingkungan pengujian dapat dijalankan dengan efisien dan handal.



Gambar 6 Sistem operasi linux ubuntu

2.8 VirtualBox

Oracle VM VirtualBox adalah perangkat lunak virtualisasi open source, cross platform yang memungkinkan beberapa sistem operasi berjalan secara bersamaan pada satu perangkat. VirtualBox memungkinakan pengguna menjalankan berbagai sistem operasi dalam virtual machine, seperti Windows dan Linux di Mac, atau Linux dan Solaris di Windows. VirtualBox adalah perangkat lunak virtualisasi open-source yang dikembangkan oleh Oracle Corporation yang tersedia di bawah GNU General Public License (GPL) versi 3. VirtualBox memiliki berbagai fungsi dan kegunaan, termasuk:

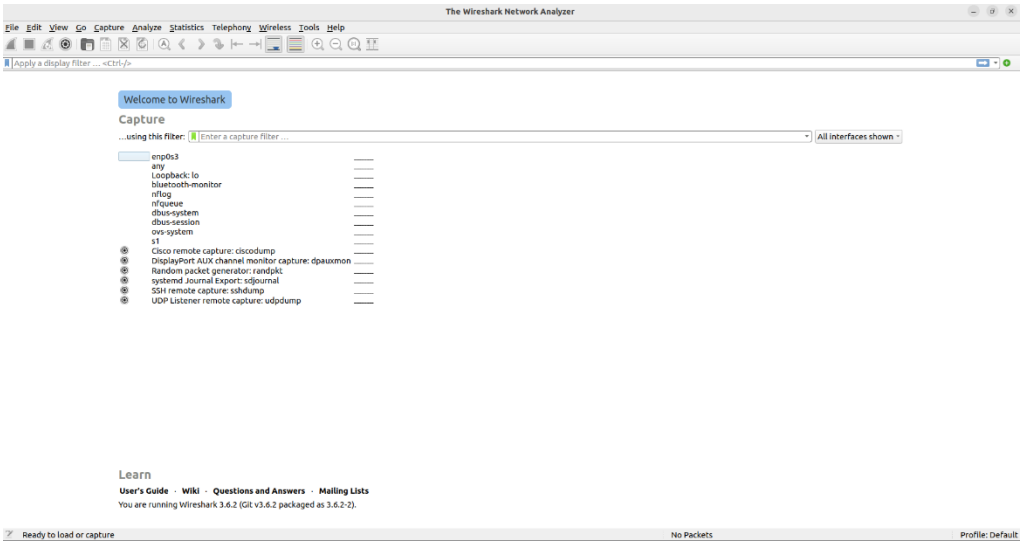
- Pengembangan dan Pengujian Aplikasi: Pengembang dapat menggunakan VirtualBox untuk menguji aplikasi mereka pada berbagai sistem operasi tanpa memerlukan perangkat keras tambahan.
- Lingkungan Uji Coba yang Aman: Dengan menjalankan sistem operasi dalam mesin virtual, pengguna dapat menguji perangkat lunak atau konfigurasi sistem tanpa risiko merusak sistem utama.
- Dukungan Multi-Platform: VirtualBox mendukung berbagai sistem operasi, termasuk Windows, Linux, macOS, dan Solaris, yang membuatnya sangat fleksibel untuk berbagai kebutuhan pengujian dan pengembangan.

2.9 Mininet

Mininet adalah sebuah emulator bersifat *open source* yang mendukung protokol OpenFlow untuk arsitektur SDN. Mininet salah satu alat yang populer yang digunakan untuk riset SDN oleh komunitas. Mininet menggunakan pendekatan virtualisasi untuk membuat sebuah jaringan yang terdiri dari *host*s, *switches*, *controller*, dan *links.* Jaringan realitas secara virtual. Sebuah sistem operasi yang memvirtual sebuah sumber daya dengan proses abstraksi, Mininet menggunakan *process-based* virtualisasi untuk mengemulasikan entitas pada *kernel* satu sistem operasi dengan menjalankan kode secara nyata, termasuk standar aplikasi jaringan, *kernel* nyata dari sistem operasi, dan tumpukan jaringan. Oleh karena itu, desain yang berkerja dengan baik memungkinkan untuk langsung diterapkan pada perangkat keras yang sebenarnya.

2.10 Wireshark

Merupakan aplikasi *network protocol analyzer* terkemuka yang banyak digunakan di dunia. Fungsi utama dari *network protocol analyzer* adalah untuk memeriksa rincian komunikasi dalam jaringan dengan cara menangkap setiap paket yang dikirim dari dan ke komputer, kemudian menyajikannya dalam format yang dapat dibaca oleh manusia. Wireshark tersedia secara open source dan dirilis di bawah GNU General Public License versi 2. Dalam penelitian ini, Wireshark 3.6.2 digunakan untuk memonitor dan menganalisis lalu lintas data pada jaringan SDN yang telah diimplementasikan dengan *network slicing* menggunakan Ryu *Controller*. Alat ini memungkinkan peneliti untuk mendapatkan data rinci tentang performa jaringan dan memastikan bahwa berbagai layanan diprioritaskan dan dialokasikan *bandwidth* sesuai dengan desain *network slicing* yang diterapkan.



Gambar 7 Tampilan wireshark

2.11 *State of Art*

Tabel 1 *State of art*

Deskripsi Judul	Pembahasan
Judul: Analisis QoS Pengaplikasian *Network Slicing* pada Topologi Abilane Jaringan SDN	Penelitian ini membahas penerapan metode *network slicing* pada *Software Defined Networking* (SDN) menggunakan FlowVisor. Metode ini bertujuan untuk menjaga kualitas

Menggunakan FlowVisor dan POX *Controller* Tahun: 2023 Penulis: Fadly Firmansyah Putra, Favian Dewata, Sovia Naning Hertiana	layanan dengan membagi jaringan secara virtual. Evaluasi kualitas layanan (QoS) dilakukan pada SDN dengan dan tanpa *network slicing* menggunakan topologi Abilene. Hasilnya menunjukkan bahwa FlowVisor berhasil mengisolasi flow space setiap *slice* berdasarkan TCP port dalam komunikasi antar *host*. Meskipun ada perbedaan nilai QoS yang tidak signifikan, jaringan SDN tanpa *network slicing* menunjukkan keunggulan sedikit. Pada akhirnya, penelitian ini mengonfirmasi keberhasilan penerapan metode *network slicing* menggunakan FlowVisor pada SDN, dengan fokus pada isolasi berdasarkan TCP port.
Judul: Implementasi Dan Analisis *Network Slicing* Berbasis *Software Defined Network* Tahun: 2022 Penulis: Rachel Caroline, Basuki Rahmat, Favia Dewata	Penelitian ini menggambarkan dampak pesatnya perkembangan teknologi terhadap jaringan komputer. Dalam merancang infrastruktur jaringan yang optimal, *Software-Defined Network* (SDN) dihadirkan sebagai arsitektur baru yang dapat mengatasi tantangan jaringan tradisional. Konsep *Network Slicing*, yang didefinisikan sebagai jaringan logis end-to-end (E2E) dengan kontrol dan manajemen independen, dipergunakan untuk meningkatkan efisiensi layanan dan aplikasi. Emulator Mininet digunakan untuk mensimulasikan teknologi SDN, dengan hasil bahwa jumlah *host* dan topologi berpengaruh pada pengiriman data. Penelitian ini mengaplikasikan satu *controller*, lima *switch*,

	dan empat *host*. Metode evaluasi menggunakan pendekatan kuantitatif, dan hasilnya menunjukkan kualitas layanan (QoS) dengan *Packet* Loss 0%, *Bandwidth* pada *Host* 81,375 Gbps, pada *Switch* 135,76 Gbps, serta *Throughput* pada *Host* 85,45 Gbps dan pada *Switch* 135,72 Gbps.
Judul: *Network Slicing Using FlowVisor for Enforcement of Bandwith Isolation in SDN Virtual Network* Tahun: 2023 Penulis: Vivi Monita, Wisnu Wedanto, Endang Anggiratih	Penelitian ini mengulas popularitas *Software-Defined Networking* (SDN) yang semakin meningkat karena berbagai fitur seperti kontrol pemrograman, pemantauan terintegrasi, kontrol yang sangat halus, fleksibilitas, dukungan untuk banyak penyewa, dan skalabilitas. Permasalahan pada desain sebelumnya, yang dikenal sebagai jaringan konvensional, melibatkan konfigurasi setiap perangkat jaringan secara individu, kontrol terdesentralisasi, dan masalah yang berkelanjutan dengan penegakan penyewa untuk mendukung multi-tenancy. Penelitian ini fokus pada penggunaan *network slicing* dalam SDN untuk memastikan isolasi penyewa menggunakan FlowVisor dan *controller* SDN. Flowspace, bagian dari FlowVisor yang mampu mengimplementasikan isolasi jaringan, digunakan untuk tujuan isolasi dalam penelitian ini. Multi-tenancy didukung dalam SDN melalui teknik *network slicing*. Penelitian ini melibatkan dua jenis penyewa, dan dua prosedur pengujian, yaitu keterhubungan dan fungsionalitas, dijalankan untuk mencapai tujuan penelitian. Temuan

	penelitian mencakup fakta bahwa semua *host* terhubung dengan benar, dan koneksi berhasil dilakukan tanpa mengaktifkan FlowVisor. Fungsi *host* hanya dapat mengirim dan menerima data dari *host* dengan penyewa yang sama. Hasil penelitian menunjukkan bahwa FlowVisor dapat diterapkan untuk penegakan isolasi. Dengan menggunakan FlowVisor untuk membangun Flowspace, penelitian ini menemukan bahwa dapat memisahkan jaringan untuk mendukung multi-tenancy, sehingga setiap penyewa dapat menggunakan potongan jaringannya sendiri tanpa gangguan dari potongan lainnya. Studi lebih lanjut dan pengujian di lingkungan dunia nyata dengan memperluas jumlah potongan mungkin dilakukan.
Judul: Analisis Efek Penggunaan Kontroller RYU dan POX pada Performansi Jaringan SDN Tahun: 2018 Penulis: Romi Afnan	Penelitian ini fokus pada pengembangan jaringan komputer dengan menggunakan *Software Defined Network* (SDN), di mana SDN memisahkan sistem pengontrol arus data dari perangkat kerasnya. Penerapan SDN memungkinkan kontrol jaringan yang terpusat melalui *controller*, dan dalam penelitian ini, dua *controller*, yaitu Ryu dan POX, dibandingkan dalam hal *Quality of service* (QoS). Uji coba dilakukan pada topologi full-mesh dengan jumlah *switch* 6, 8, dan 10, di mana setiap *switch* terhubung dengan 2 *host*. Hasil evaluasi menunjukkan bahwa nilai QoS yang diperoleh oleh kedua *controller* masih berada dalam standar ITU-T G.1010, dengan

	delay yang kecil untuk data, VoIP, dan video. Penelitian ini bertujuan untuk menganalisis efek penggunaan *controller* Ryu dan POX pada performansi jaringan SDN.