SKRIPSI

IMPLEMENTASI *FAULT TOLERANT* PADA *CONTROLLER* POX DAN RYU DALAM JARINGAN *SOFTWARE DEFINED NETWORK*

Disusun dan diajukan oleh:

MUHAMMAD KAISAR AGUNG DAENG MARAJA

D121 17 1703

DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA

2024

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI *FAULT TOLERANT* PADA *CONTROLLER* POX DAN RYU DALAM JARINGAN *SOFTWARE DEFINED NETWORK*

Disusun dan diajukan oleh

MUHAMMAD KAISAR AGUNG DAENG MARAJA
D121171703

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian
Studi Program Sarjana Program Studi
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 24 Juli 2024
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Eng. Muhammad Niswar, S.T., M.IT.	Dr.Eng. Zulkifli Tahir, S.T., M.Sc.
NIP 197309221999031001	NIP 198404032010121004

Ketua Program Studi,

Prof. Dr. Ir. Indrabayu, ST., MT., M.Bus.Sys., IPM, ASEAN. Eng.
NIP 197507162002121004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini ;

Nama	: Muhammad Kaisar Agung Daeng Maraja
NIM	: D121171703
Program Studi	: Teknik Informatika
Jenjang	: S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

Implementasi *Fault Tolerant* Pada *Controller* Pox Dan Ryu Dalam Jaringan *Software Defined Network*

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan 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 penerbitannya. 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 diklarifikasi 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, 24 Juli 2024

Yang Menyatakan

Muhammad Kaisar Agung Daeng Maraja

ABSTRAK

MUHAMMAD KAISAR AGUNG DAENG MARAJA. *Implementasi Fault Tolerant Pada Controller Pox Dan Ryu Dalam Jaringan Software Defined Network* (dibimbing oleh Muhammad Niswar dan Zulkifli Tahir)

Arsitektur *Software Defined Network* (SDN) tradisional didasarkan pada pengontrol tunggal di *Control Plane.* Oleh karena itu, fungsi jaringan menjadi sangat bergantung pada kinerja pengontrol tunggal di *Control Plane.* Beberapa *Controller* yang telah berkembang diantaranya *OpenDayLight* (ODL), POX, Nox, Beacon, RYU, *Open Network Operating System* (ONOS), Floodlight, Maestro, Onix dan masih ada beberapa lainnya yang terus dikembangkan. Setiap *controller* mempunyai kelebihan dan kekurangan sehingga, dapat menentukan kegunaan masing-masing pada *controller* tersebut dan pengaruhnya terhadap performansi jaringan

Berdasarkan latar belakang yang telah dipaparkan, maka tujuan penelitian ini dilakukan untuk menganalisis *Fault Tolerant* pada *controller Software Defined Network* (SDN), POX dan RYU *controller*, menggunakan topologi *tree* dengan parameter *Quality of Service* (QoS) seperti *Throughput, Packet Loss, Delay* dan *Jitter* serta menggunakan variasi *background traffic* dalam pengujiannya untuk mengetahui informasi mengenai perbandingan *Fault Tolerant* pada setiap *controller.*

Pengujian dilakukan dengan Perancangan dua arsitektur *Multiple Distributed Controller* terdiri dari dua bagian yaitu perancangan topologi dan perancangan *software*. Perancangan topologi pada penelitian ini menggunakan topologi *tree* yang terdiri dari 2 *controller* dan 4 buah *Open Flow Switch*.Setelah dilakukan proses pengujian kemudian akan dilakukan analisis dengan parameter *Quality of Service* (QoS) *throughput, delay, jitter, dan packet loss* pada kedua *controller* POX dan RYU pada saat kedua *controller* menjadi *controller slave.*

Perbandingan *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network* dilakukan menggunakan parameter *Quality of Service* (QoS) *throughput, delay, jitter, dan packet loss. Controller* RYU secara konsisten menunjukkan tingkat *packet loss* yang rendah pada berbagai tingkat *background traffic,* sementara POX memiliki tingkat *packet loss* yang tinggi mulai dari skala yang rendah sampai tinggi. Dalam hal ini, RYU dapat dianggap lebih *Fault Tolerant* dibandingkan POX karena kemampuannya dalam mengatasi *packet loss* yang lebih baik. Kemudian baik POX maupun RYU menunjukkan kinerja yang baik dalam parameter *throughput, delay, dan jitter* pada berbagai skenario uji coba. Pengujian dilakukan pada *background traffic* 50 MB, 100 MB, 150 MB, dan 200 MB dan menggunakan *Intent Departure Time* (IDT) atau paket konstan sebesar 1000 pps.

Keywords: Fault Tolerant, Controller, POX, RYU

ABSTRACT

MUHAMMAD KAISAR AGUNG DAENG MARAJA. *Implementation of Fault Tolerance on POX and Ryu Controllers in Software Defined Networks* (Supervised by Muhammad Niswar and Zulkifli Tahir)

The traditional *Software Defined Network* (SDN) architecture is based on a single controller in the Control Plane. Consequently, network functions heavily rely on the performance of this single controller in the Control Plane. Several controllers have emerged over time, including OpenDayLight (ODL), POX, Nox, Beacon, RYU, Open Network Operating System (ONOS), Floodlight, Maestro, Onix, and others that continue to be developed. Each controller has its strengths and weaknesses, influencing their usability and impact on network performance.
Based on the aforementioned background, this research aims to analyze fault tolerance in SDN controllers, specifically POX and RYU controllers. This will be done using a tree topology and Quality of Service (QoS) parameters such as Throughput, Packet Loss, Delay, and Jitter, with varying background traffic to gather information on the fault tolerance comparison between these controllers.
The testing involves designing two architectures of Multiple Distributed Controllers, consisting of two parts: topology design and software design. The topology design in this research uses a tree topology that includes 2 controllers and 4 Open Flow Switches. After the testing process, an analysis will be conducted using QoS parameters—throughput, delay, jitter, and packet loss—on both POX and RYU controllers when they function as slave controllers.
The comparison of fault tolerance between POX and RYU controllers in an SDN network is carried out using QoS parameters—throughput, delay, jitter, and packet loss. The RYU controller consistently shows low packet loss across various levels of background traffic, whereas the POX controller exhibits high packet loss from low to high scales. Thus, RYU can be considered more fault-tolerant than POX due to its better handling of packet loss. Both POX and RYU demonstrate good performance in throughput, delay, and jitter across different test scenarios. The testing was conducted with background traffic levels of 50 MB, 100 MB, 150 MB, and 200 MB, using a constant packet rate of 1000 pps Intent Departure Time, (IDT).

Keywords: Fault Tolerant, Controller, POX, RYU

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN DAN ARTI SIMBOL

Lambang/Singkatan	Arti dan Keterangan
SDN	*Software Defined Network*
ODL	*OpenDayLight*
ONOS	*Open Network Operating System*
QoS	*Quality of Service*
IDT	*Intent Departure Time*
API	*Application Programming Interface*
NTT	*Nippon Telegraph and Telephone*
C1	*Controller Master*
C2	*Controller Slave*
BPS	*Bit Per Second*
MBITS/SEC	*Throughput*
%	*Packet loss*
MS	*Delay*
MS	*Jetter*

DAFTAR LAMPIRAN

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan atas kehadirat Allah subhanahu wa ta'ala yang tiada batas memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul "Implementasi *Fault Tolerant* Pada *Controller* Pox Dan Ryu Dalam Jaringan *Software Defined Network*" sebagai syarat dalam menyelesaikan studi jenjang Strata-1 di Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin.

Penulis menyadari bahwa dalam penyusunan dan penulisan laporan penelitian ini tidak terlepas dari doa, usaha, dan bantuan dari berbagai pihak, mulai masa perkuliahan hingga masa penyusunan tugas akhir ini. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih dengan bangga kepada:

1. Keluarga tercinta terkhusus kepada kedua orang tua penulis, Bapak Yusuf Djunaid, S.E., M.Si. dan Ibu Hasfiati yang selalu memberikan pikiran, dukungan, doa, dan bantuannya yang tak terbatas.
2. Pembimbing utama, Bapak Dr. Eng. Muhammad Niswar, S.T., M.IT. dan pembimbing pendamping, Bapak Dr.Eng. Zulkifli Tahir, S.T., M.Sc. atas waktu, pikiran, perhatian, tenaga, arahan dan bantuannya yang luar biasa hingga saat ini penulis dapat menyelesaikan penelitian ini.
3. Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin, Bapak Prof. Dr. Ir. Indrabayu, S.T., M.T., M.Bus.Sys., IPM, ASEAN. Eng. atas ilmu, wawasan, dan pengalaman yang diberikan kepada penulis.
4. Segenap Dosen dan Staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah banyak membantu penulis.
5. Lab UBICON, yang selalu memberikan ruang diskusi dan memberikan banyak masukan kepada penulis.
6. Rekan tim penelitian SDN *(Software Difined Network)*, saudara Ghazi dan Sanji yang banyak memberikan dukungan dan pemikiran pemikiran dalam penelitian penulis.
7. DOMI09, sebagai rumah kedua dan tempat cerita penulis.

8. Saudara saya Fmzr, Ody, Bojel, Hery, dan saudari Priska yang hadir memberikan motivasi dan semangat kepada penulis selama menyusun tugas akhir.
9. Orang-orang hebat dan spesial sebagai sosok yang telah hadir dalam kehidupan saat senang dan sulit penulis.
10. Last but no least, Teknik Informatika angkatan 2017 (*RECOG17NER*), saudara seperjuangan yang menyimpan banyak kenangan dan pengalaman tak terlupakan karena takkan ada kisah tanpa ada kamu, aku, dan kalian.

Harapan penulis, semoga Tugas Akhir ini dapat memberikan sumbangsih dan manfaat besar bagi kepentingan bersama dan semoga Allah SWT membalas segala bantuan yang diberikan.

Wassalamualaikum Warahmatullahi Wabarakatuh

Gowa, 13 Juni 2024

Penulis,

Muhammad Kaisar Agung Daeng Maraja

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Teknologi jaringan internet khususnya bidang infrastruktur jaringan mengalami peningkatan yang sangat pesat pada beberapa tahun belakangan ini. Menurut data yang dirilis oleh APJII, bahwa jumlah pengguna internet di Indonesia hingga kuartal II tahun 2020 mencapai 196,7 juta pengguna dari jumlah populasi penduduk Indonesia yaitu 266,9 juta. Hal ini juga berpengaruh pada layanan internet yang berkembang dengan berbagai kompleksitas, desain dan manajemen jaringan itu sendiri, sehingga jaringan harus mampu menampung jumlah pengguna yang besar dan memberikan layanan data yang optimal (APJII, 2020).

Arsitektur *Software Defined Network* (SDN) tradisional didasarkan pada pengontrol tunggal di *Control Plane*. Oleh karena itu, fungsi jaringan menjadi sangat bergantung pada kinerja pengontrol tunggal di *Control Plane,* yang tidak diinginkan untuk aplikasi andal apa pun. Meskipun banyak keuntungan dari SDN, penerapannya di bidang praktis dibatasi karena kemampuan sistem yang handal dan toleransi kesalahan tidak memuaskan. Untuk mengatasi kesulitan SDN ini, arsitektur (FT- SDN) telah diusulkan. Arsitektur yang diusulkan terdiri dari *Control Plane* terdistribusi yang sederhana dan efektif dengan banyak pengontrol. FT-SDN menggunakan mekanisme tersinkronisasi untuk secara berkala memperbarui status pengontrol di dalamnya. Jika terjadi kegagalan, FT-SDN memilikikemampuan untuk memilih pengontrol kerja lain berdasarkan jarak dan penundaan diantara entitas jaringan yang berbeda. Kinerja arsitektur FT-SDN diperiksa sehubungan dengan spesifikasi yang berbeda dengan adanya kesalahan (Rohit Kumar Das dkk, 2020).

Pendekatan yang digunakan untuk mengatasi permasalahan tersebut yaitu melalui konsep *Software Defined Network* (SDN). *Software Defined Network* (SDN) adalah sebuah konsep pendekatan baru untuk mendesain, membangun dan mengelola jaringan komputer yang memisahkan antara sistem kontrol (*control plane*) dan sistem forwarding (*data plane)* pada perangkat jaringan (Shaikh et al., 2018).

Arsitektur SDN memungkinkan suatu jaringan dapat secara dinamis menyesuaikan lingkungan untuk kebutuhan aplikasi maupun kebutuhan pengguna, menyederhanakan manajemen dan meningkatkan skalabilitas jaringan yang diwujudkan melalui implementasi sederhana dari penambahan komponen dan layanan jaringan (M. H. Hidayat dkk, 2017).

Komponen yang paling penting dalam SDN adalah *controller* karena secara langsung melakukan kendali terhadap konfigurasi perangkat jaringan itu sendiri. Pada dasarnya sebuah *controller* memusatkan kecerdasan jaringan, sedangkan jaringan mempertahankan *data plane* yang di distribusikan *Switch OpenFlow*. Oleh karena itu sebuah *controller* menyediakan antarmuka untuk mendesain, membangun dan mengelola tabel *flow* pada *switch* (Pramudita dkk, 2020).

Beberapa *Controller* yang telah berkembang diantaranya *OpenDayLight* (ODL), POX, Nox, Beacon, RYU, *Open Network Operating System* (ONOS), Floodlight, Maestro, Onix dan masih ada beberapa lainnya yang terus dikembangkan. Setiap *controller* mempunyai kelebihan dan kekurangan sehingga, dapat menentukan kegunaan masing-masing pada *controller* tersebut dan pengaruhnya terhadap performansi jaringan (Zhu et al., 2019).

Berdasarkan latar belakang yang telah dipaparkan, maka perlu dilakukan penelitian untuk menganalisis *Fault Tolerant* pada *controller Software Defined Network* (SDN), POX dan RYU *controller*, menggunakan topologi *tree* dengan *parameter Quality of Service* (QoS) seperti *Throughput, Packet Loss, Delay* dan *Jitter* serta menggunakan variasi *background traffic* dalam pengujiannya. Adapun judul pada penelitian ini yaitu **"Implementasi *Fault Tolerant* Pada *Controller* POX dan RYU Dalam Jaringan *Software Defined Network*"**.

1.2 Rumusan Masalah

1. Bagaimana mengimplementasikan *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network* ?
2. Bagaimana perbandingan *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network* ?

1.3 Tujuan Penelitian

1. Untuk mengimplementasikan *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network.*

2. Untuk mengetahui *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network.*

1.4 Manfaat Penelitian

1. Memberikan informasi mengenai cara implementasi *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network.*
2. Memberikan informasi mengenai perbandingan *Fault Tolerant* pada *controller* POX dan RYU dalam jaringan *Software Defined Network.*

1.5 Ruang Lingkup

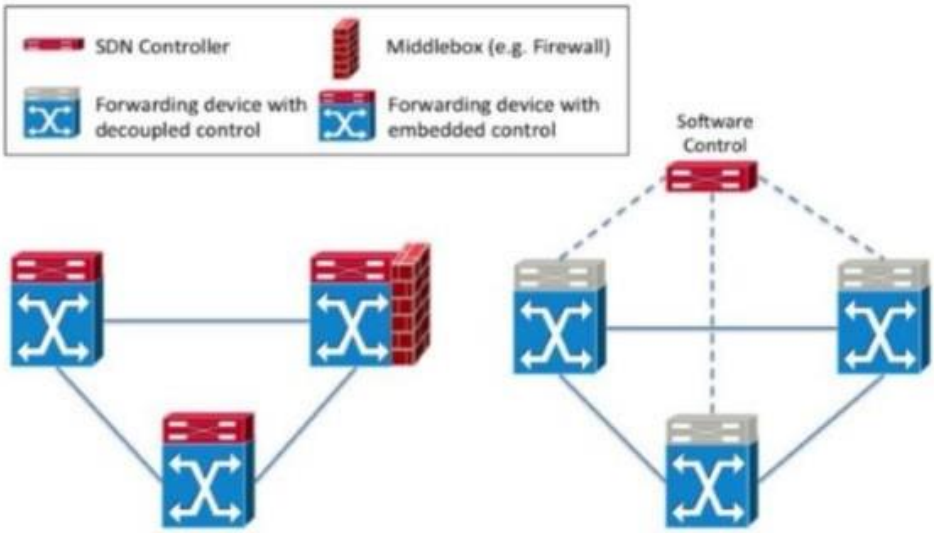
- Emulasi jaringan *Software Defined Network* dilakukan dengan emulator Mininet.
- Topologi yang digunakan Topologi Tree.
- *Controller* yang digunakan adalah *controller* POX dan RYU.
- Menggunakan *virtual box* untuk menjalankan *controller* dan emulator.
- Sistem operasi yang digunakan adalah Ubuntu 22.04 LTS.
- Mitigasi *controller* POX dan RYU.

BAB II
TINJAUAN PUSTAKA

2.1 *Software Defined Networking*

Software-Defined Networking (SDN) adalah sebuah paradigma arsitektur baru dalam bidang jaringan komputer, yang memiliki karakteristik dinamis, *manageable*, *cost-effective*, dan *adaptable*, sehingga sangat ideal untuk kebutuhan aplikasi saat ini yang bersifat dinamis dan *high-bandwidth*. Arsitektur ini memisahkan antara *network control* dan fungsi *forwarding*, sehingga *network control* tersebut menjadi *directly programmable* (dapat diprogram secara langsung), sedangkan infrastruktur yang mendasarinya dapat diabstraksikan untuk *layer* aplikasi dan *network services*. (thirdquarter, 2019).

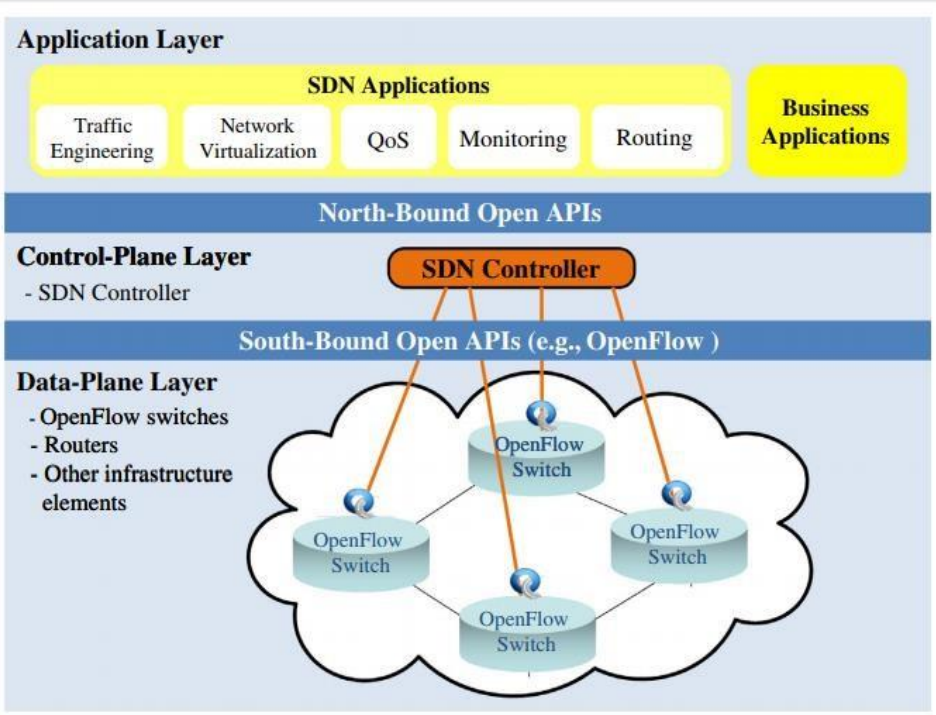
*Software Defined Network*ing (SDN) diusulkan untuk mengatasi kelemahan jaringan tradisional. Sebagai paradigma jaringan baru, SDN merevolusi teknologi jaringan dengan mematahkan ide dasar jaringan tradisional (Hu Tao, 2017).



Gambar 1 Perbedaan Jaringan Tradisional (kiri) dan *Software Defined Network* (kanan) (Hu Tao, 2017).

Perbedaan paling mendasar antara jaringan tradisional dan *Software Defined Network* adalah penempatan fungsi *control* dan *forward*. Pada jaringan tradisional fungsi *control* dan *forward* ditempatkan pada *device* yang sama yaitu *router*. Sedangkan pada jaringan SDN fungsi kontrol ditempatkan pada *software* terpusat (*controller*) dan fungsi *forward* ditempatkan pada suatu perangkat kosong berupa *switch (forwarding decive)* (ONF, 2012).

SDN lebih murah karena perangkat *switching* pengalihan data universal yang mengikuti standar tertentu, dan memberikan kontrol lebih besar atas arus lalu lintas jaringan dibandingkan dengan perangkat jaringan konvensional (Hu Fei, 2014)



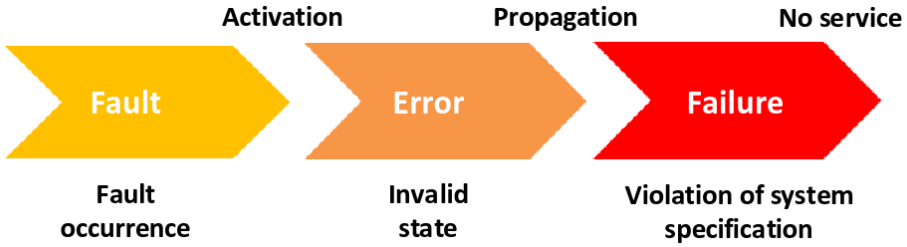
Gambar 2 Arsitektur *Software defined network* (Ahyoung, 2014)

Pada gambar 2 di atas adalah contoh bentuk *interface* pada SDN. *Forwarding Plane (Data plane) Layer* merupakan lapisan pertama dari arsitektur SDN. *Forwarding Plane* terdiri dari *router,* Openflow *switch,* dan *device* jaringan lainnya. Sedangkan *Control plane layer* merupakan lapisan kedua arsitektur SDN yang terdiri dari *controller*, dimana bertugas sebagai pengendali SDN (dapat berupa Opendaylight *Controller*, Floodlight *Controller*, dan lain sebagainya). *Application Layer* merupakan lapisan paling atas dari arsitektur SDN, terdiri dari *Traffic Engineering* (rekayasa trafik), virtualisasi jaringan, QoS (*Quality of Service), Monitoring, Routing, Security, Access control*, dan manajemen *bandwidth* (Ahyoung, 2014).

Data plane layer dapat berkomunikasi dengan *control plane layer*, membutuhkan protokol yang dapat menjembatani komunikasi antara kedua *layer* tersebut. Protokol yang dimaksud adalah Openflow. Sedangkan yang menjembatani komunikasi antara *Control plane Layer* dengan *Application Layer* adalah API (Application Programming Interface).

2.2 *Fault Tolerant*

Fault tolerant adalah hasil dari proses desain membangun sistem yang andal dari komponen yang tidak dapat diandalkan. Ada beberapa penyebab kesalahan misalnya bug perangkat lunak, kesalahan buatan manusia, atau perangkat keras kegagalan daya. Hubungan antara kesalahan, kesalahan, dan kegagalan digambarkan pada Gambar 3 (AU Rehman dkk, 2019).

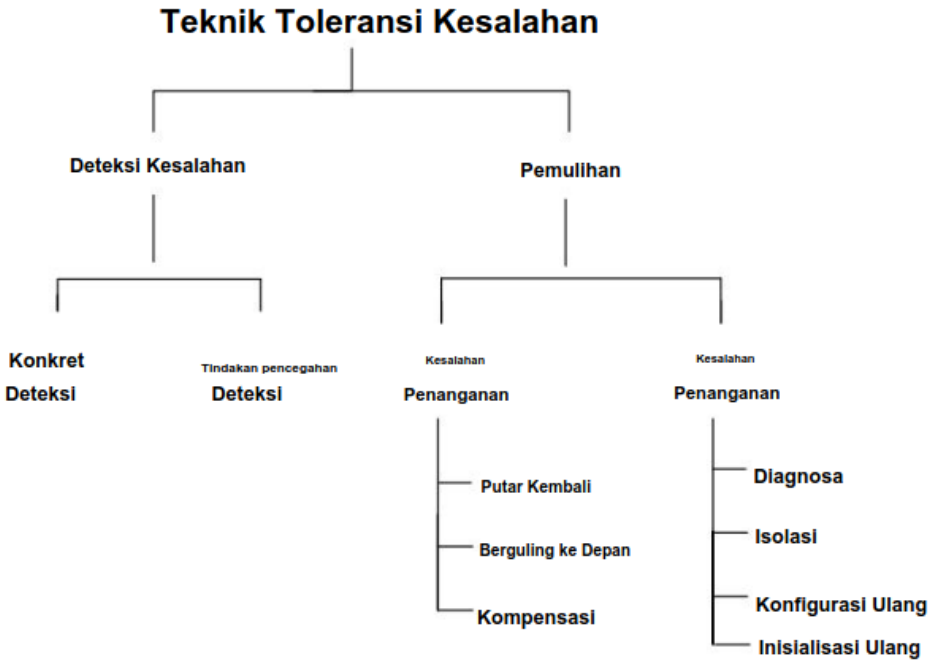


Gambar 3 Hubungan *Fault, Error, and Failure* (AU Rehman dkk, 2019)

Fault Tolerant terdapat pada arsitektur FT-SDN yang memiliki fitur toleransi kesalahan melalui manajemen *Control Plane. Fault Tolerant* menggunakan beberapa *controller* sumber terbuka dari kategori berbeda untuk mengatasi masalah heterogenitas. Sistem dapat berfungsi dengan baik meskipun satu atau lebih *controller* tidak berfungsi. *Fault tolerant* pada Arsitektur FT-SDN memiliki kemampuan instalasi ulang yang efisien dari *controller* yang gagal. Oleh karena itu, ia akan mampu memberikan ketahanan yang baik dalam menghadapi kegagalan beberapa *controller* dan serangan eksternal dengan jumlah paket yang dikirim di jaringan paling sedikit. (Rohit Kumar Das dkk, 2020).

Beberapa teknik *fault tolerant* digunakan untuk menghindarinya kegagalan layanan dengan adanya kesalahan. *Fault tolerant* dilakukan melalui deteksi kesalahan dan pemulihan sistem, atau mekanisme deteksi dan pemulihan. Deteksi kesalahan mengidentifikasi adanya kesalahan, sementara "pemulihan mengubah keadaan sistem yang mengandung satu atau lebih kesalahan dan (mungkin) kesalahan dalam keadaan tanpa kesalahan yang terdeteksi dan kesalahan yang dapat diaktifkan kembali". Pemulihan teknik dapat diklasifikasikan lebih lanjut menjadi dua kategori pertama pemulihan dengan penanganan kesalahan yang menghilangkan kesalahan dari keadaan sistem dan kedua pemulihan dengan penanganan kesalahan yang mencegah kesalahan diaktifkan kembali. Pilihan dari deteksi kesalahan dan teknik pemulihan sedang diadopsi berdasarkan asumsi

kesalahan yang mendasar. Dalam konteksnya SDN, teknik toleransi kesalahan ini harus dieksplorasi untuk meningkatkan toleransi kesalahan di lingkungan SDN.



Gambar 4 Teknik *Fault Tolerant* (AU Rehman dkk, 2019)

2.3 *Controller*

Controller dalam konteks SDN adalah perangkat lunak pengendali yang mengelola protokol OpenFlow. Terdapat dua jenis *controller* yang umumnya digunakan yaitu :

1. *Controller* statis adalah perangkat yang mampu menambahkan atau menghapus aliran data *(flow)* dalam *flow table* secara statis.
2. *Controller* dinamis sebaliknya, dapat secara dinamis memodifikasi isi dari *flow table*, sehingga lebih cocok untuk berbagai konfigurasi yang berubah.

Menurut penelitian oleh Vishnoi, dkk. (2013), *controller* memiliki beberapa tanggung jawab utama, yaitu:

a) Mengelola Koneksi dan Interaksi dengan *Switch* OpenFlow: *Controller* bertanggung jawab menyediakan mekanisme untuk menghubungkan dan berinteraksi dengan *switch* OpenFlow.
b) Menginterpretasikan Pesan dari *Switch* OpenFlow: *Controller* harus mampu memahami pesan yang dikirim oleh *switch* OpenFlow untuk mengatur operasi jaringan dengan benar.

c) Menyediakan Instruksi Sesuai Spesifikasi: *Controller* harus menyediakan semua instruksi yang diperlukan sesuai dengan spesifikasi yang berlaku, baik yang sudah ada, tambahan, atau gabungan keduanya, agar dapat diprogram ke dalam *switch.*
d) Mengelola Informasi dari Umum ke Detail: *Controller* harus memberikan mekanisme kepada *switch* untuk mendapatkan informasi dari level umum hingga yang paling detail. Selain itu, *controller* juga harus mampu menginterpretasikan respon yang sesuai dari *switch.*

2.4 *POX*

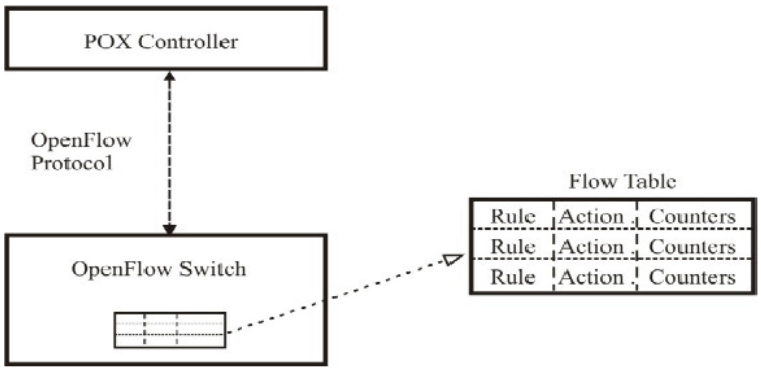
POX merupakan *platform* pengembangan SDN *Controller* yang menggunakan bahasa pemrograman Python sebagai dasarnya. *Platform* ini memiliki beberapa komponen yang dapat dikombinasikan sesuai kebutuhan pengguna (Prayoga et al., 2017).

Dalam konteks POX, penggunaan *controller* menjadi lebih efisien dalam mengimplementasikan protokol OpenFlow, yang merupakan standar komunikasi antara *controller* dan *switch*. Dengan menggunakan POX, pengguna dapat menjalankan berbagai aplikasi, termasuk fungsi-fungsi seperti *hub, switch, load balancer,* dan *firewall*. Selain itu, alat paket *capture* dapat dimanfaatkan untuk memantau aliran paket antara *Controller* POX dan perangkat OpenFlow (Kaur dkk, 2014).

POX sendiri merupakan versi berbasis Python dari NOX (NOX diimplementasikan dalam bahasa C++). Perkembangan POX dilatarbelakangi oleh keinginan untuk mengembangkan NOX ke dalam bahasa C++ sambil menyediakan platform terpisah dalam bahasa Python (Python 2.7). Beberapa keuntungan yang ditawarkan POX dibandingkan dengan NOX pada publikasi Thomas D. Nadeau, (2013) yaitu:

a) POX mempunyai antarmuka Python untuk OpenFlow.
b) POX memiliki sejumlah komponen contoh yang dapat digunakan kembali untuk tugas seperti pemilihan jalur, penemuan topologi, dan lain sebagainya.
c) POX dapat dijalankan di berbagai sistem operasi termasuk Linux, Windows, dan Mac OS.
d) POX mendukung GUI dan visualisasi, serupa dengan NOX.

e) Dalam hal kinerja, POX memiliki performa yang lebih baik dibandingkan dengan NOX.

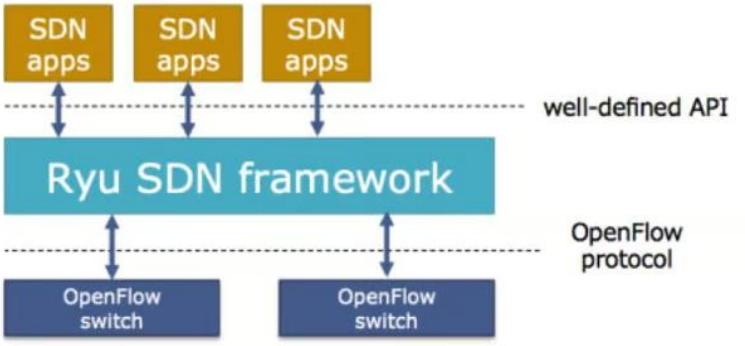


Gambar 5 Arsitektur POX *Controller* (Kaur et al., 2014)

2.5 *RYU*

RYU adalah sebuah kerangka kerja *(framework)* SDN yang dikembangkan oleh *Nippon Telegraph and Telephone* (NTT). RYU adalah pengontrol SDN yang sepenuhnya diimplementasikan menggunakan bahasa pemrograman Python. Nama 'RYU' berasal dari kata dalam bahasa Jepang yang berarti 'mengalir,' yang mengacu pada konsep mengatur 'aliran' untuk mengaktifkan kecerdasan dalam jaringan. *Controller* RYU menyediakan komponen perangkat lunak dengan antarmuka program yang terdefinisi dengan baik (API), yang memudahkan pengembang dalam menciptakan manajemen jaringan baru dan aplikasi pengendalian yang terstruktur (Islam et al., 2020)

RYU menyediakan komponen perangkat lunak dengan antarmuka program yang terdefinisi dengan baik (API), yang memudahkan pengembang untuk membuat manajemen jaringan baru dan aplikasi pengendalian. RYU mendukung berbagai protokol untuk mengelola perangkat jaringan, termasuk OpenFlow, Netconf, OF-config, dan lainnya. Terkait dengan protokol OpenFlow, RYU mendukung sepenuhnya versi 1.0, 1.2, 1.3, 1.4, 1.5, dan juga ekstensi Nicira. Seluruh kode sumber RYU tersedia secara bebas di bawah lisensi Apache 2.0. RYU dikembangkan menggunakan bahasa pemrograman Python dan bersifat *open source* (Edgar dkk, 2019).



Gambar 6 Perintah Mininet (Sumber Rahmawan et al., 2020)

2.6 *Mininet*

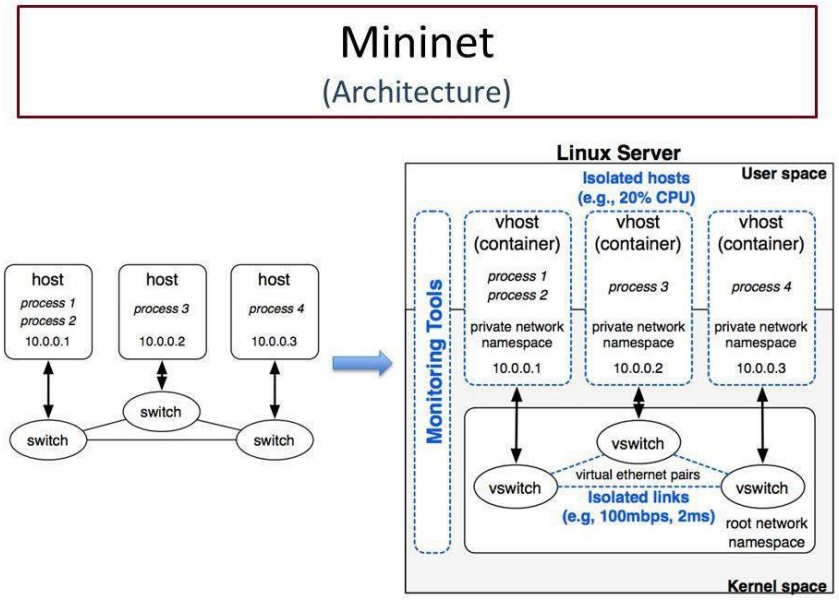
Mininet adalah sebuah emulator untuk membuat *prototype* jaringan berskala besar secara cepat pada sumberdaya yang terbatas (seperti pada single komputer atau laptop maupun Virtual Machine). Mininet diciptakan dengan tujuan untuk mendukung riset di bidang SDN dan *OpenFlow*. Emulator Mininet memungkinkan kita untuk menjalankan sebuah kode secara interaktif di atas laptop atau di atas virtual *hardware*, tanpa harus memodifikasi kode tersebut. Artinya kode simulasi sama persis dengan kode pada *real network environment* (Sembiring, 2018).

Pada mininet ini dilakukan perancangan jaringan dengan topologi yang diinginkan. Secara sederhana mininet ini berfungsi untuk emulasi pada bagian data path untuk mengetes konfigurasi jaringan SDN. Sedangkan untuk melakukan testing pada mininet dapat dilakukan dengan *command* "sudo mn". Dengan *command* ini mininet akan mengemulasikan konfigurasi jaringan SDN yang terdiri dari 1 *controller*, 1 *switch* dan 2 *host* (Sudiyatmoko dkk, 2016).

Gambar 7 Arsitektur Ryu *Controller* (Sudiyatmoko dkk, 2016)

Mininet adalah solusi yang dianggap paling unggul dalam hal kemudahan penggunaan, performansi, akurasi, dan skalabilitas. Ia mampu menyediakan lingkungan yang realistis dan nyaman *(convenience)* dengan harga yang murah *(low cost)*. Kita dapat menggunakan alternatif lain seperti *hardware test-bed* untuk simulasi jaringan, yang mana dapat berjalan cukup kencang dan akurat, namun

harganya mahal dan harus di-*shared* dengan pengguna lain. Begitu pula, kita dapat menggunakan simulator yang harganya murah, namun seringkali kode simulasi akan harus dimodifikasi lagi bila akan dijalankan di *real network environment* (Fadli, 2018).



Gambar 8 Arsitektur Mininet (Fadlli 2018)

BAB III
METODE PENELITIAN

3.1 Waktu dan Lokasi Penelitian

Penelitian ini dimulai pada bulan Desember 2022. Penelitian ini dilakukan di Laboratiorum UBICON, Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin.