BAB II
KERANGKA TEORITIS

A. Landasan Teori

1. Jaringan Komputer

Jaringan komputer adalah sebuah sistem yang terdiri atas komputer-komputer yang didesain untuk dapat berbagi sumber daya (*printer, CPU)*, berkomunikasi (surel, pesan instan) dan dapat mengakses informasi (peramban *web*). Tujuan dari jaringan komputer adalah agar dapat mencapai tujuannya, setiap bagian dari jaringan komputer dapat meminta dan memberikan *service*. Pihak yang meminta/menerima layanan disebut *client* dan yang memberikan/mengirim layanan disebut *server*. Desain ini disebut dengan sistem *client-server*, dan digunakan pada hampir seluruh aplikasi jaringan komputer (jafar noor yulianto, 2007)**.**

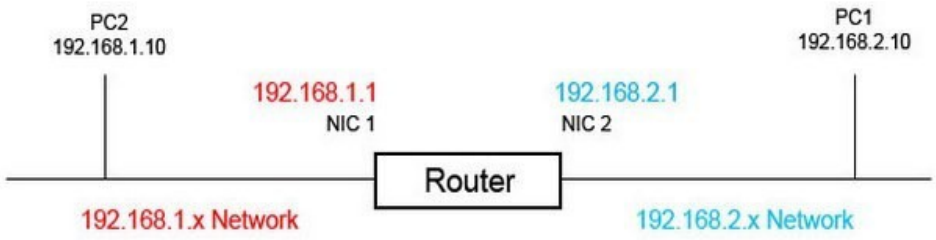
(Herlambang dan L*,* 2008) menjelaskan adapun sejumlah potensi jaringan komputer, yaitu antara lain:

a. Mengintegrasikan dan berbagi-bagi peralatan
   Memungkinkan pengguna bersama perlatan komputer berbagai merek, yang semula tersebar diberbagai ruangan, unit, dan departemen sehingga meningkatkan efektifitas dari penggunaan sumber daya tersebut.
b. Komunikasi
   Jaringan komputer memungkinkan terjadinya komunikasi antara pemakai komputer. Selain itu, tersedia aplikasi *teleconference* yang memungkinkan dilakukannya rapat atau pertemuan tanpa harus meninggalkan meja kerja.
c. Mengintegrasikan data
   Jaringan komputer diperlukan untuk menginegrasikan data antar komputer-komputer *client* sehingga dapat diperoleh suatu data yang relevan
d. Perlindungan data dan informasi
   Jaringan komputer memudahkan upaya perlindungan data yang terpusat pada server, melalui pengaturan hak akses dari pemakai serta penerapan sistem *password*
e. Sistem Terdistribusi
   Jaringan komputer dimanfaatkan pula untuk mendistribusikan proses dan aplikasi sehingga dapat mengurangi terjadinya bottleneck atau penumpukan pekerjaan pada satu bagian.
f. Keteraturan aliran informasi
   Jaringan komputer mampu mengalirkan data-data komputer *client* dengan cepat untuk diintegrasikan dalam komputer server. Selain itu, jaringan mampu

untuk mendistribusikan informasi secara berkelanjutan kepada pihak-pihak terkait yang membutuhkannya.

2. *Router*

Router merupakan perangkat keras jaringan komputer yang dapat digunakan untuk menghubungkan beberapa jaringan yang sama atau berbeda. *Router* adalah sebuah alat untuk mengirimkan paket data melalui jaringan atau internet untuk dapat menuju tujuannya, proses tersebut dinamakan *routing*. Proses *routing* itu sendiri terjadi pada lapisan 3 dari *stack* protokol tujuh-lapis *Open System Interconection (*OSI*). Router* terkadang digunakan untuk mengoneksikan 2 buah jaringan yang menggunakan media berbeda. fungsi *Router* adalah untuk menghubungkan beberapa jaringan dan memfasilitasi transmisi antar jaringan tersebut. Dalam hal ini, *router* membutuhkan setidaknya dua kartu jaringan atau *Network Interface Card* (NIC) yang dipasang pada setiap jaringan.



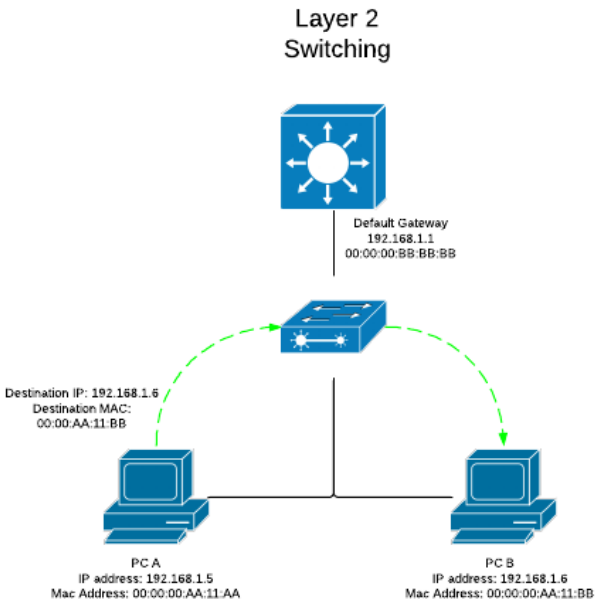
Gambar 2.1 Router

Pada gambar 2.1 router di atas terdapat dua jaringan dengan satu r*outer* yang sangat sederhana dan mudah untuk dikonfigurasi. Untuk jaringan yang besar dan kompleks tentunya pengaturan akan berbeda dan lebih rumit.

3. *Switch*

Switch pada jaringan di gunakan untuk menghubungkan komputer atau penghalang pada sebuah area yang terbatas, s*witch* juga bekerja di lapisan *data link* pada OSI *layer.* Adapun sistem kerja *switch* mirip pada *bridge*, namun *switch* memiliki beberapa port menjadikan sering disebut dengan *multi port bridge*.

Terdapat 2 jenis *switch* yaitu *switch* berdasarkan model OSI dimana terdapat s*witch layer* dua dan *layer* tiga s*witch layer* 2 yang beroperasi pada *data link layer* yang terdapat pada lapisan model *OSI*, yang mana s*witch* akan meneruskan paket dengan melihat *MAC address* tujuan, di lain hal s*witch* dapat melakukan fungsi *bridge* antara segmen-segmen *Local Area Network (LAN)* s*witch* mengirimkan paket-paket data dengan cara melihat alamat yang ditujunya tanpa mengetahui protokol jaringan yang dipakai.



Gambar 2.2 Switching

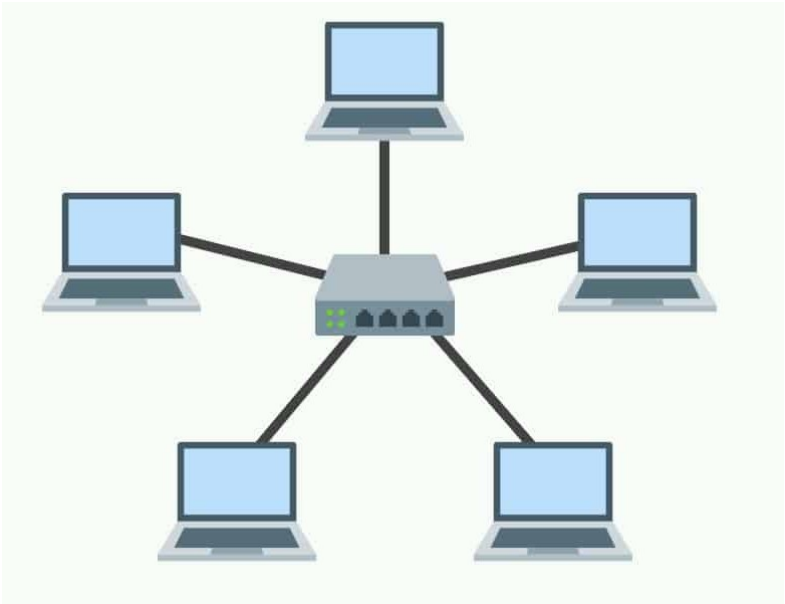
Switch Layer 3 dapat dijumpai pada *network layer* lapisan OSI Variasi *switch* satu ini bisa digunakan untuk melanjutkan paket data dengan memakai *Internet Protocol (IP) address* pada perangkat tertentu. *Switch* satu ini juga kerap kali disebut sebagai *switch multilayer* atau *switch routing*. *Switch layer* 3 bekerja dengan kemampuan *fast forwarding via hardware*. Nantinya *IP forwarding* dapat melibatkan *route lookup*, pengurangan hitungan *Time to Live (TTL)* serta penghitungan ulang *checksum*. Di samping itu, *switch* satu ini juga bisa meneruskan *frame* dengan *MAC header* sesuai *output port* yang semestinya.

4. Topologi Jaringan

Topologi jaringan adalah suatu cara untuk membuat sejumlah komputer saling berhubungan satu sama lain, baik menggunakan kabel maupun yang nirkabel. Biasanya tujuan topologi jaringan adalah demi kemudahan pertukaran informasi. Salah satu topologi yang sering digunakan yaitu topologi star. Topologi jaringan berbentuk star atau bintang adalah jaringan dari beberapa komputer yang memiliki koneksi dengan node yang berada di jaringan pusat. Jadi, masing-masing perangkat memiliki koneksi dengan *node* yang berada di tengah sistem jaringan.

Segala pertukaran data dilakukan dengan melewati jaringan di pusat. Istilah yang sering digunakan dalam teknologi informasi, yakni jaringan pusat disebut stasiun primer dan *node* yang terkoneksi disebut sebagai stasiun sekunder. Cara kerja topologi jaringan star adalah dengan merancang beberapa

jaringan komputer untuk terkoneksi dengan pusat yang disebut *hub* atau *switch*. Kemudian jaringan yang berada di pusat akan menjadi semacam *server* sentral.



Gambar 2. 3 Topologi Star

Switch, sebagai salah satu perangkat di server pusat akan menyimpan semua aliran data dari node sebagai daftar *CAM* (*Content Addressable Memory*) pada memori yang tersedia. *CAM* berfungsi untuk menyimpan semua alamat perangkat yang terhubung dengan s*witch*.

B. Pemahaman Teoritis

1. *Software Defined Network*

SDN adalah sebuah konsep pendekatan jaringan komputer dimana sistem pengontrol dari arus data dipisahkan dari perangkat kerasnya (Open Network Foundation, 2016). Umumnya sistem pembuat keputusan ke mana arus data dikirimkan dibuat menyatu dengan perangkat kerasnya. Sebuah konfigurasi SDN dapat menciptakan jaringan dimana perangkat keras pengontrol lalu lintas data secara fisik dipisahkan dari perangkat keras *data forwarding plane.* Konsep ini dikembangkan di UC Berkeley and Stanford University sekitar tahun 2008. Penemu dan penyedia sistem ini mengklaim dapat menyederhanakan jaringan komputer.

SDN memerlukan beberapa mekanisme agar *control plane* untuk berkomunikasi dengan data plane. Salah satu mekanisme tersebut adalah *Openflow* yang sering disalahpahami setara dengan SDN. *The Open Networking* merupakan yayasan yang didirikan untuk mempromosikan SDN dan *Openflow*

serta mempromosikan istilah *cloud computing* sehingga menjadi popular (Open Networking Foundation, 2022).

SDN merupakan salah satu tahap evolusi *"programmable and active networking".* Salah satu pengaplikasian *SDN* adalah *infrastructure as a service (IaaS).* Jaringan virtual *SDN* dikombinasikan dengan *virtual machine (VM)* dan *virtual storage* dapat menirukan pengalokasian sumber daya pusat data yang dinamis. *SDN* memungkinkan *network administrator* untuk memprogram pusat kontrol jaringan melalui sebuah *controller* tanpa akses fisik ke *switch* (Mulyana,). Beberapa aspek penting dari *Software Defined Network (SDN)* itu sendiri, adalah:

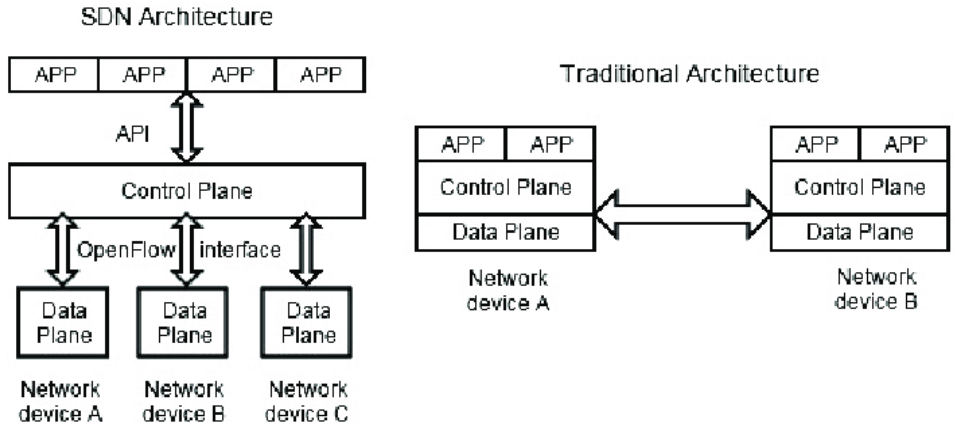
a. Adanya pemisahan secara fisik/eksplisit antara *forwarding/data plane* dan *control plane.*
b. *interface* standar (*vendor-agnostic*) untuk memprogram perangkat jaringan.
c. *Control-plane* yang terpusat secara logika atau adanya sistem operasi jaringan yang mampu membentuk *logical map* dari seluruh jaringan, kemudian memperesentasikannya melalui *API* (*Application Programming Interface).*
d. Virtualisasi dimana beberapa sistem operasi jaringan dapat mengkontrol bagian-bagian (*splices atau substrates*) dari perangkat yang sama.

Dalam hal ini terkait dengan kebutuhan inovasi untuk bidang jaringan yang semakin kompleks. Termasuk diantaranya adalah fakta-fakta dan kebutuhan seperti berikut:

a. Virtualisasi dan *Cloud* adalah komponen dan entitas jaringan *hybrid* antara fisik *bare metal* dan *visual*
b. *Orchestration* and *Scalibility* adalah kemampuan untuk mengatur dan mengelola ribuan perangkat melalui sebuah *point of management*.
c. *Programmability* dan *Automation* adalah kemampuan untuk mengubah *behaviour* (perilaku) jaringan serta untuk dapat melakukan perubahan tersebut secara otomatis (contoh: kemampuan *troubleshooting*, perubahan *policy* dan lain-lain).
d. *Visibility* adalah kemampuan untuk dapat memonitor jaringan, baik dari sisi sumber daya, konektivitas dan lain-lain.
e. Kinerja adalah kemampuan untuk memaksimalkan penggunaan perangkat jaringan, misalnya *bandwidth*, *load balancing*, *traffic engineering* dimana berhubungan dengan *programmability* dan *scability*.

2. Arsitektur Software Defined Network

Pada jaringan dengan arsitektur konvensional atau non-SDN fungsi dari control plane dan data plane berada dalam masing-masing perangkat atau satu perangkat yang sama. *Control Plane* memiliki fungsi untuk mengatur atau mengontrol perangkat jaringan, sedangkan *Forwarding Plane* memiliki fungsi dari pengiriman paket-paket informasinya. Mengutip istilah dari *SDxCentral Control Plane* ini dapat kita ibaratkan sebagai sebuah "otak" yang mengatur lalu lintas data dalam ajringan, sedangkan *Forwarding Plane* bisa kita ibaratkan seperti "otot" yang hanya berfungsi untuk mengirimkan paket ke tujuan sesuai dengan permintaan otak (*control plane*). Perbedaan dari arsitektur jaringan konvensional dengan jaringan SDN dapat kita lihat pada gambar 2.1 *SDN Architecture* dan *Traditional Architecture.*

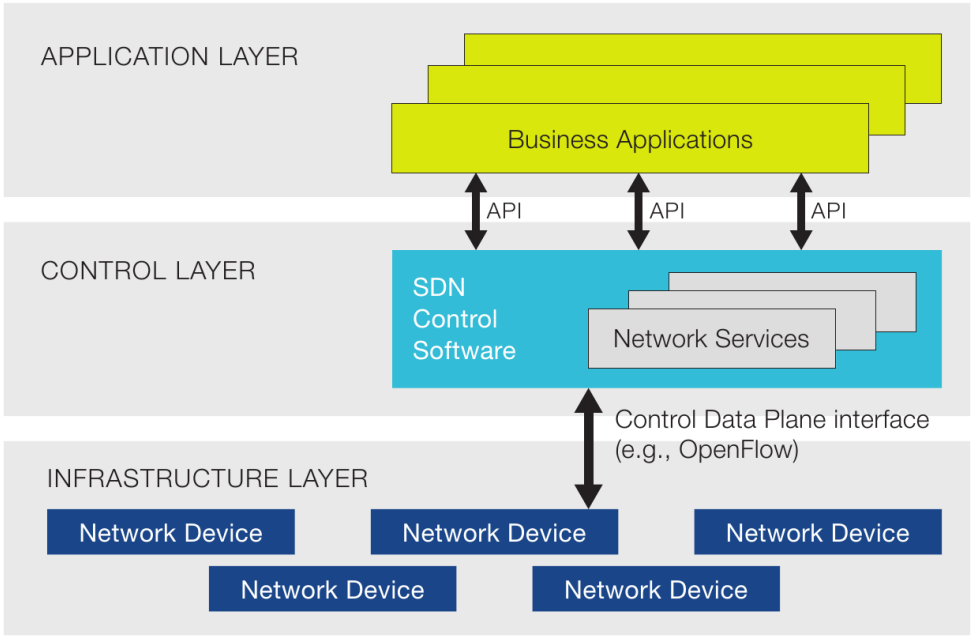


Gambar 2.4 SDN Architechture & Traditional Architechture

Dalam arsitektur jaringan konvensional ketika kita ingin membangun sebuah jaringan (besar) diperlukan konfigurasi yang cukup lama dan cukup sulit. Dimana kita harus mendatangi perangkat yang akan dikonfigurasi satu persatu. Selain itu terdapat banyak kemungkinan *human error* pada konfigurasi yang dilakukan oleh sang *network engineer* atau *administrator*, cara ini dianggap kurang efektif dan efisien. Hal itu diperkuat dengan besarnya kebutuhan *bandwidth* saat ini yang membuat operator telekomunikasi harus sering melakukan *shifting* (penggantian) perangkat lama ke perangkat yang lebih baru.

Pada arsitektur *SDN*, pengaturan atau *control plane* dibuat terpusat. Artinya, satu buah *controller* (c*ontrol plane platform*) dapat melakukan konfigurasi atau manajemen terhadap lebih dari dua buah *data plane* (*forwarding plane*) secara langsung. Jadi, hal ini dapat mengatasi permasalahan implementasi dalam jaringan. Dari yang sebelumnya kita melakukan konfigurasi kepada setiap perangkat secara langsung satu persatu, menjadi konfigurasi terpusat yang dapat

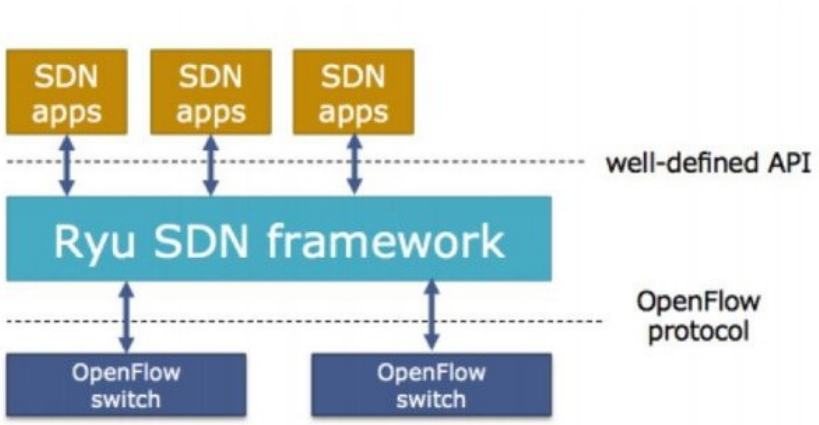
dilakukan lebih cepat dan efisien sehingga dapat menghemat waktu serta pekerjaan menjadi lebih efektif.



Gambar 2.5 SDN Architechture

3. Ryu *Controller*

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



Gambar 2.6 Arsitektur SDN Menggunakan Ryu Controller

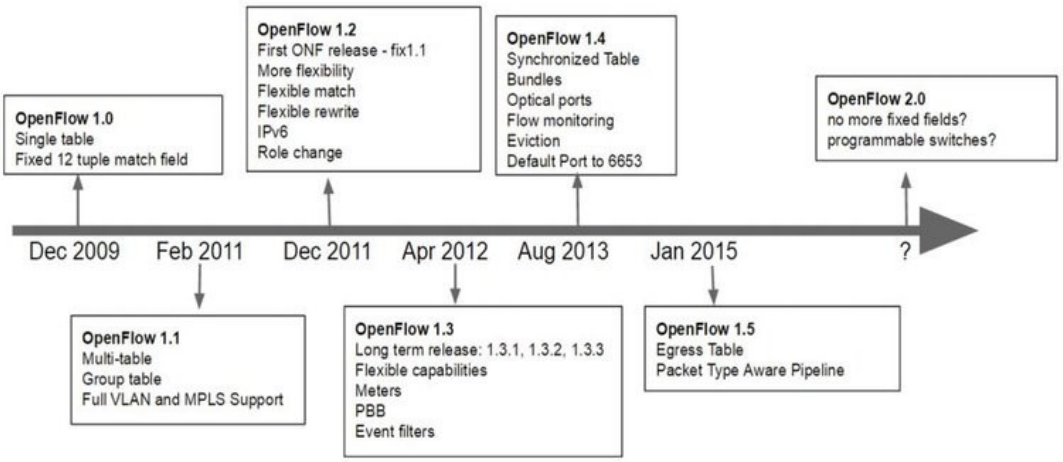
Ryu *controller* sendiri merupakan *framework controller* yang berada pada *control layer* pada arsitektur SDN dan aplikasi pada Ryu berada pada *application layer* guna untuk berkomunikasi dengan aplikasi SDN lainnya menggunakan API seperti REST, RPC dan lainnya.

Ryu banyak digunakan karena mempunyai beberapa keunggulan dibanding dengan *controller* lain diantaranya.

- Ryu menyediakan banyak komponen yang berguna untuk aplikasi SDN.
- Komponen lama Ryu dapat dimodifikasi sesuai dengan kebutuhan dan menerapkan pada komponen yang baru.
- Menggabungkan komponen untuk membangun aplikasi.

4. *Openflow*

Openflow adalah *protocol* terbilang relatif baru yang dirancang dan diimplementasikan di Stanford University pada tahun 2008. *Protocol openflow* telah berkembang sejak dimulainya proses standarisasi oleh *open network foundation (ONF),* yaitu sebuah organisasi yang berfocus dalam pengembangan protocol Openflow, dari versi 1.0 – 1.5*. openflow* terus berkembang dengan menambahkan fitur-fitur baru, dibawah ini adalah *roadmap* perkembangan openflow dari versi 1.0 – 1.5 (Patterson, 2017).



Gambar 2.7 Openflow Version

Pengertian dari *openflow* adalah salah satu jenis API dalam jaringan yang digunakan untuk mengontrol atau mengatur *traffic flows* pada s*witch* dalam sebuah jaringan. singkatnya *control plane* berkomunikasi dengan *data plane* melalui *OpenFlow*. *OpenFlow* dapat bekerja pada *switch* dari berbagai *vendor*. *OpenFlow* sendiri adalah *southbound interface*, dimana *interface* yang memungkinkan terjadinya komunikasi antara *layer controller* dan *data plane* pada arsitektur SDN.

5. 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*.

6. *Quality of Service*

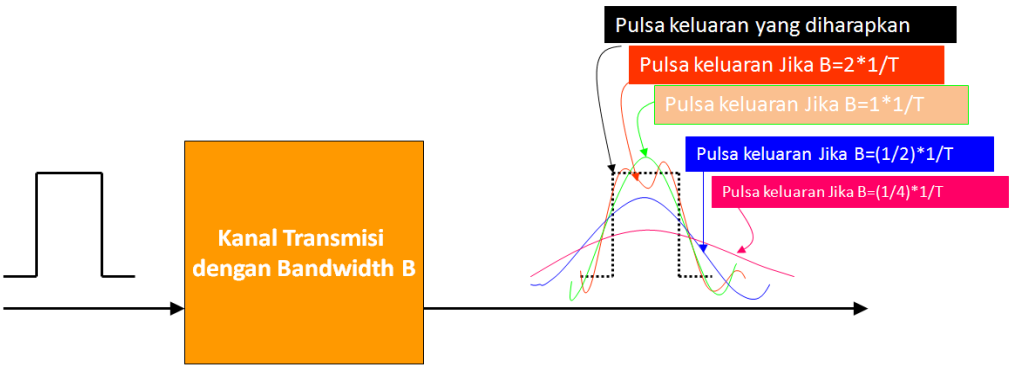
Quality of Service (QoS) adalah kemampuan suatu jaringan untuk menyediakan layanan yang baik dengan menyediakan *bandwith*, mengatasi *jitter* dan *delay*. Parameter *QoS* adalah *latency*, *jitter, packet loss, throughput, MOS, echo cancellation* dan *PDD*. *QoS* sangat ditentukan oleh kualitas jaringan yang digunakan. Terdapat beberapa factor yang dapat menurunkan nilai *QoS*, seperti:

a. Redaman

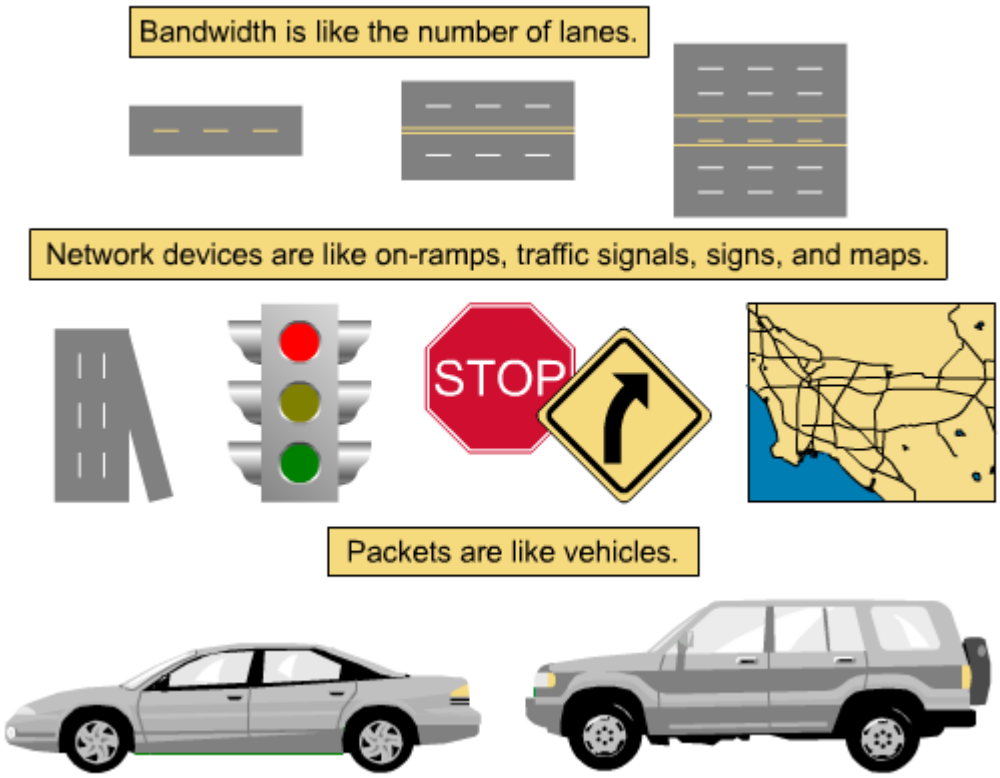
Redaman, yaitu jatuhnya kuat sinyal karena pertambahan jarak dan tebalnya dinding penghalang. Setiap media transmisi memiliki redaman yang berbeda-beda, tergantung dari jenis dan bahan yang digunakan. Kekuatan sinyal yang ditransmisikan bisa mengalami pelemahan karena jarak yang jauh dan medium penghalang dalam bentuk apapun.

b. Distorsi

yaitu fenomena atau kejadian yang disebabkan bervariasinya kecepatan propagasi karena perbedaan *bandwidth*. Hal ini bisa terjadi akibat kecepatan sinyal yang berbeda dalam hal ini medium sinyal frekuensi yang di lalui pada seluruh jaringan *hotspot*, sehingga data atau *packet* tiba pada penerima dalam waktu yang berbeda. Untuk mengurangi nilai distorsi, maka dibutuhkan *bandwidth* transmisi yang memadai dan dianjurkan digunakan pemakaian *bandwidth* yang seragam, sehingga distorsi dapat dikurangi.



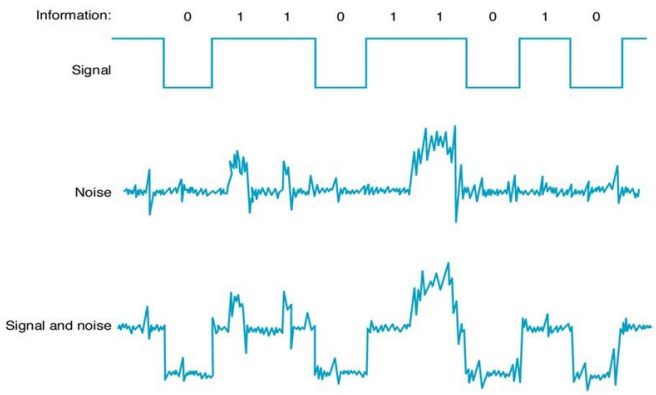
Gambar 2.8 Distorsi



Gambar 2.9 SImulasi Distorsi

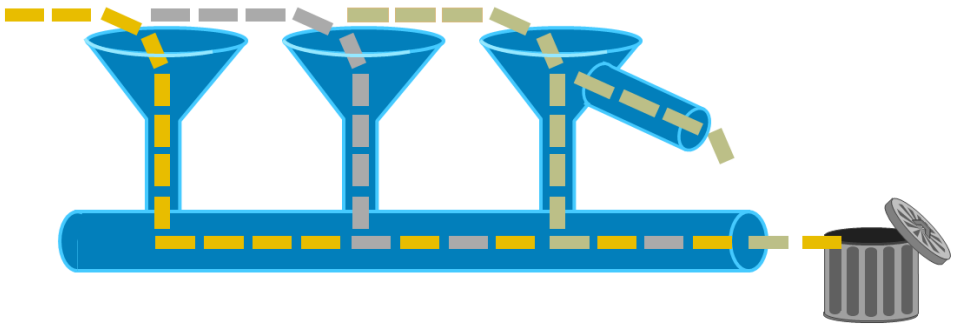
c. *Noise*

Noise adalah tambahan sinyal yang tidak dikehendaki atau berdekatan (*interferensi Co-Channel*) yang masuk di manapun di antara transmisi pengirim dan penerima pada saat pengukuran parameter *QoS*.



Gambar 2.10 Noise

QoS *"the collective effect of service performance which determines the degree of satisfaction of a user of the service"*. *International Telecommunication Union (ITU). Quality of Service (QoS)* didesain untuk membantu *end user (client)* menjadi lebih produktif dengan memastikan bahwa user mendapatkan performansi yang handal dari aplikasi-aplikasi berbasis jaringan.



Gambar 2.11 Quality of Services

Ada beberapa alasan mengapa kita memerlukan QoS, yaitu:

a. Untuk memberikan prioritas untuk aplikasi-aplikasi yang kritis pada jaringan.
b. Untuk memaksimalkan penggunaan investasi jaringan yang sudah ada.
c. Untuk meningkatkan performansi untuk aplikasi-aplikasi yang sensitif terhadap *delay*, seperti v*oice* dan *video*.
d. Untuk merespon terhadap adanya perubahan-perubahan pada aliran *traffic* di jaringan.

Terdapat 3 tingkat *QoS* yang umum dipakai, yaitu *best-effort service*, *integrated service* dan *differentiated service*. Ketiga level tersebut akan diuraikan lebih detail dibawah ini.

a. *Best-Effort Service*

Best-effort service digunakan untuk melakukan semua usaha agar dapat mengirimkan sebuah paket ke suatu tujuan. Penggunakan *best-effort service* tidak akan memberikan jaminan agar paket dapat sampai ke tujuan yang dikehendaki. Sebuah aplikasi dapat mengirimkan data dengan besar yang bebas kapan saja tanpa harus meminta ijin atau mengirimkan pemberitahuan ke jaringan. Beberapa aplikasi dapat menggunakan *best-effort service*, sebagai contohnya *File Transfer Potocol (FTP)* dan *Hyper Text Transfer Protocol (HTTP)* yang dapat mendukung *best-effort service* tanpa mengalami permasalahan.

b. *Integrated Service*

Model integrated service menyediakan aplikasi dengan tingkat jaminan layanan melalui negosiasi parameter-parameter jaringan secara *end-to-end*. Aplikasi-aplikasi akan meminta tingkat layanan yang dibutuhkan untuk dapat beroperasi dan bergantung pada mekanisme *Quality of Service (QoS)* untuk menyediakan sumber daya jaringan yang dimulai sejak permulaan transmisi dari aplikasi-aplikasi tersebut. Aplikasi tidak akan mengirimkan *trafict*, sebelum menerima tanda bahwa jaringan mampu menerima beban yang akan dikirimkan aplikasi dan juga mampu menyediakan *Quality of Service (QoS)* yang diminta secara *end-to-end*.

c. *Differentiated Service*

Differentiated service menyediakan suatu set perangkat klasifikasi dan mekanisme antrian terhadap protokol-protokol atau aplikasi-aplikasi dengan prioritas tertentu di atas jaringan yang berbeda. *Differentiated service* bergantung pada kemampuan *edge router* untuk memberikan klasifikasi dari paket-paket yang berbeda tipenya yang melewati jaringan. *Trafict* jaringan dapat diklasifikasikan berdasarkan alamat jaringan, *protocol* dan *port*, *ingress interface*, atau klasifikasi lainnya selama masih didukung oleh *standard access list* atau *extended access list*.

Performansi mengacu ke tingkat kecepatan dan kehandalan penyampaian berbagai jenis beban data di dalam suatu komunikasi. Performansi merupakan kumpulan dari beberapa parameter besaran teknis, parameter *Quality of Service (QOS)* diantaranya:

a. *Throughput*

Throughput yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam *byte per second (bps)*. *Troughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada destination selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut.

Tabel 2.1 Kategori Throughput

Kategori *Throughput*	***Throughput***	**Indeks**
Bad	0 – 338 kbps	0
Poor	338 – 700 kbps	1
Fair	700 – 1.200 kbps	2
Good	1.200 kbps – 2,1 Mbps	3
Excelent	> 2,1 Mbps	4

b. *Packet Loss*

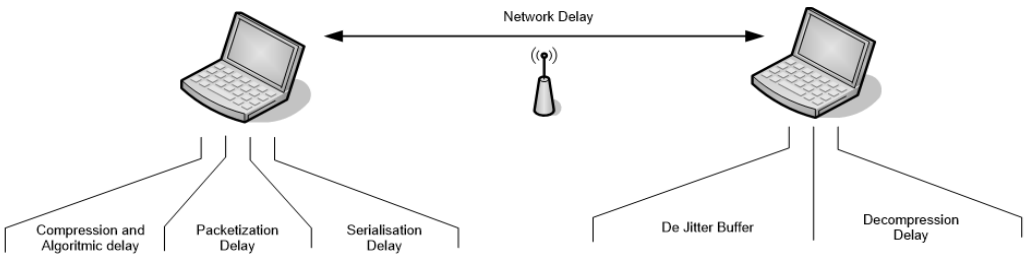
Merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi karena retransmisi akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi-aplikasi tersebut. Umumnya perangkat jaringan memiliki *buffer* untuk menampung data yang diterima. Jika terjadi kongesti yang cukup lama, *buffer* akan penuh, dan data baru tidak akan diterima.

Tabel 2.2 Kategori Degradasi

Kategori Paket *Loss*	*Packet Loss*	Indeks
Poor	> 25 %	1
Medium	12 – 24 %	2
Good	3 – 14 %	3
Perfect	0 – 2%	4

c. *Delay (latency)*

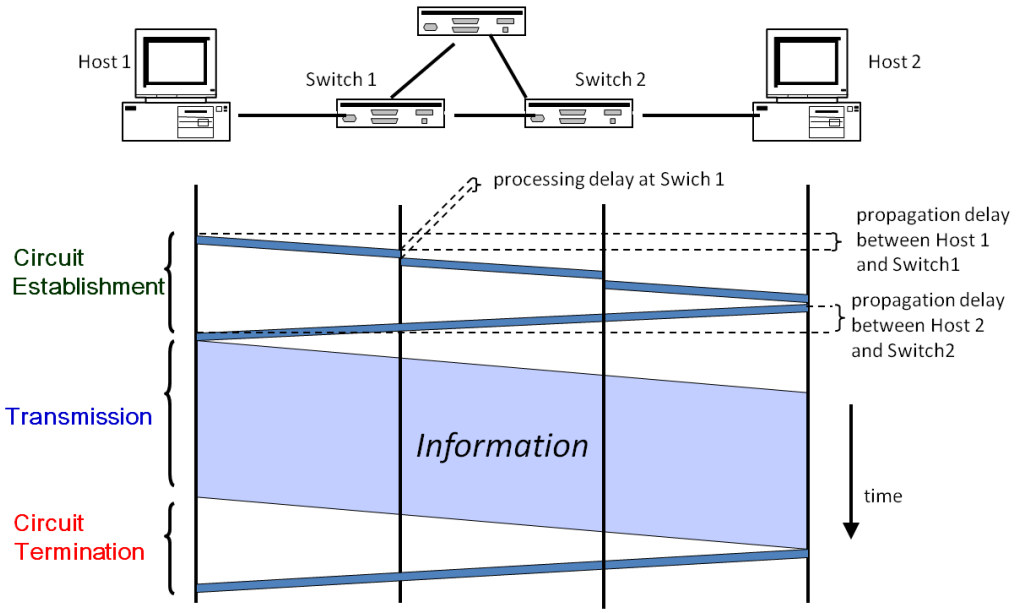
Delay (latency) adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama.



Gambar 2.12 Delay Latency

Tabel 2.3 Jenis Delay

Jenis Delay	Keterangan
Algoritmic Delay	*Delay* ini disebabkan oleh *standar codec* yang digunakan.
Packetization delay	*Delay* yang disebabkan oleh peng-akumulasian *bit voice sample* ke *frame*. Seperti contohnya, standard G.711 untuk *payload 160 bytes* memakan waktu 20ms
Serialization delay	*Delay* ini terjadi karena adanya waktu yang dibutuhkan untuk pentransmisian paket ip dari sisi *originating* (pengirim)
Propagation delay	*Delay* ini terjadi karena perambatan atau perjalanan. Paket ip di media transmisi ke alamat tujuan contohnya delay propagasi di dalam kabel akan memakan waktu 4 sampau 6 ms per kilometernya
Coder (processing) Delay	Waktu yang diperlukan oleh *digital signal processing (DSP)* untuk mengkompres sebuah *block PCM*, nilainya bervariasi tergantung dari *codec* dan kecepatan *processor.*



Gambar 2.13 Simulasi Delay Latency

Tabel 2.4 Kategori Latensi

KATEGORI LATENSI	BESAR *DELAY*
Excellent	< 150 ms
Good	150 s/d 300 ms
Poor	300 s/d 450 ms
Unacceptable	➢ 450 ms

d. Jitter

Jitter, atau variasi kedatangan paket, hal ini diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket di akhir perjalanan *jitter*. *Jitter* lazimnya disebut variasi *delay*, berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada taransmisi data di jaringan. *Delay* antrian pada *router* dan *switch* dapat menyebabkan *jitter*.

Tabel 2.5 Kategori Jitter

KATEGORI Jitter	*PEAK JITTER*
Sangat bagus	0 ms
Bagus	0 s/d 75 ms
Sedang	76 s/d 125 ms
Jelek	125 s/d 225 ms

C. Manajemen Jaringan

Manajemen jaringan dapat didefinisikan sebagai operasional, administrasi, pemeliharaan, dan penyedia jaringan serta layanan jaringan. Jenis pengoperasian berkaitan dengan kegiatan sehari-hari dalam menyediakan layanan jaringan (Arif Budiman Santoso, n.d.) Manajemen jaringan adalah sebuah pekerjaan untuk memeliharaseluruh sumber jaringan dalam keadaan baik. Sistem manajemen jaringan adalah sekumpulan perangkat untuk memantau dan mengontrol jaringan. Sistem manajemen jaringan terdiri dari tambahan perangkat keras dan piranti lunak yang diimplementasikan di antara komponen–komponen jaringan yang sudah ada. Dalam implementasinya, para pengelola jaringan sering mengandalkan berbagai jenis peralatan.

Dalam sebuah model konseptual yang didefinisikan oleh *The International Organization for Standardization (ISO)* fungsi manajemen jaringan yang dapat dijelaskan sebagai berikut:

1. *Fault Management* (Manajemen Kesalahan)
   Menyediakan fasilitas yang memungkinkan administrator jaringan untuk mengetahui kesalahan (*fault*) pada perangkat yang dikelola, jaringan, dan operasi jaringan, agar dapat segera menentukan apa penyebabnya dan dapat segera mengambil tindakan (perbaikan).
2. *Configuration Management* (Manajemen Konfigurasi)
   Memonitor informasi konfigurasi jaringan sehingga dampak dari perangkat keras atau pun perangkat lunak tertentu dapat dikelola dengan baik.
3. *Accounting* (Pelaporan)
   Mengukur utilisasi jaringan dari pengguna atau grup tertentu untuk membantu dalam menjaga performa jaringan pada level tertentu yang dapat diterima.
4. *Performance Management* (Manajemen Performa)
   Mengukur berbagai aspek dari performa jaringan termasuk pengumpulan dan analisis dari data statistik sistem sehingga dapat dikelola dan dipertahankan pada level tertentu yang dapat diterima.
5. *Security Management* (manajemen Keamanan)
   mengatur akses ke sumber daya jaringan sehingga informasi tidak dapat diperoleh tanpa izin.

Arsitektur dari manajemen jaringan terdiri dari

1. *Network Management System*
   Network Management Station (NMS), menjalankan aplikasi manajemen jaringan yang mampu mengumpulkan informasi mengenai perangkat yang dikelola dari agen manajemen yang terletak dalam perangkat. Aplikasi manajemen jaringan harus memproses data dalam jumlah yang besar, bereaksi terhadap peristiwa tertentu (*event*), dan mempersiapkan informasi yang relevan untuk ditampilkan. NMS biasanya memiliki *console* kendali dengan sebuah antarmuka *Graphical User Interface* (*GUI*) yang memungkinkan pengguna untuk melihat representasi grafis dari jaringan, mengontrol perangkat dalam jaringan yang dikelola, dan memprogram aplikasi manajemen jaringan. Beberapa aplikasi manajemen jaringan dapat diprogram untuk bereaksi terhadap informasi yang didapat dari agen manajemen atau mengatur nilai ambang (*threshold*).

2. Perangkat yang dikelola

   Perangkat yang dikelola, berupa semua jenis perangkat yang berada dalam jaringan, seperti komputer, *printer*, atau pun *router*.

3. *Management Agent*

   Management agent, memberikan informasi mengenai perangkat yang dikelola kepada NMS dan dapat juga menerima informasi kendali/kontrol.

4. *Protocol Management* Jaringan

   Protokol manajemen jaringan, digunakan oleh NMS dan agen manajemen untuk bertukar informasi.

5. Information Management

   Informasi manajemen, merupakan informasi yang dipertukarkan antara *NMS* dan agen manajemen yang memungkinkan proses monitor dan kontrol dari perangkat.

D. Tinjauan Pustaka

Sebelum penelitian ini sudah dilakukan beberapa penelitian pada kasus yang berbeda dengan metode yang sama, sebagai bahan pertimbangan pada penelitian ini dan untuk mengetahui perbedaan penelitian sebelumnya dengan penelitian akan dilakukan. Berikut adalah penelitian yang telah dilakukan sebelumnya:

1. **Dalam Penelitian Estu Rizky Huddiniah "Optimasi Rute Untuk *Software Defined Networking-Wide Area Network* (*SDN-WAN*) Dengan *Openflow Protokol*". (1 Februari 2018)**(Huddiniah et al., 2018)

   Pada penelitian tersebut permasalahan yang dibahas adalah jaringan computer konvensional dengan jaringan berbasis *Software Defined Network – wide area network* (*SDN-WAN*) dengan menggabungkan 2 buah proses *Routing* Pada implementasi *SDN-WAN* berbasis *openFlow*, terdapat pendekatan *path label Routing* untuk mengoptimalkan proses *Routing*. Pendekatan tersebut merupakan penggabungan *path label Routing* pada *WAN* berbasis *SDN-WAN* dengan *Routing* konvensional menggunakan *interface* dari *OpenFlow.*

   Perbedaan dengan yang diteliti yaitu: implementasi *Software Defined Network* untuk mengoptimalkan jaringan komputer dengan *openflow* di area local kemudian di bandingkan kinerjanya dengan jaringan yang masih menggunakan arsitektur konvensional.

2. **Dalam penelitian Izzatul Ummah "Perancangan Simulasi Jaringan Virtual Berbasis *Software-Define Networking*" (1 Maret 2016)**(Ummah, 2016)

   Pada penelitian tersebut permasalahan yang dibahas adalah pengujian dilakukan untuk mengetahui perilaku jaringan virtual *SDN* serta performansinya, diukur dari *delay*, *jitter*, dan *throughput*, Kinerja jaringan virtual berbasis *SDN*

memiliki nilai *delay* yang tetap meskipun ada beberapa peningkatan sesuai dengan jumlah *Switch* yang semakin meningkat, nilai *jitter* cenderung meningkat sesuai dengan jumlah *Switch*, namun nilai *delay dan jitter* masih memenuhi standard rekomendasi *ITU-T*. Sedangkan untuk nilai *throughput* cenderung stabil baik untuk *port TCP* maupun *UDP* (untuk *port UDP* lebih tinggi).

Perbedaan dengan yang diteliti yaitu: Menggunakan *RYU Controller* sebagai *controller* dalam *Software Defined Network* serta melakukan pengujian menggunakan *device* mikrotik.

3. **Dalam penelitian Muhammad Hikam Hidayat berjudul "Analisis Kinerja dan Karakteristik Arsitektur *Software-Defined Network Berbasis OpenDaylight Controller*" (27 Juli 2018)**(M. H. Hidayat & Rosyid, 2017)

   Pada penelitian tersebut permasalahan yang dibahas adalah pengujian SDN dilakukan untuk mengetahui performasi dari *opendaylight controller* menggunakan berbagai jenis topologi dan juga *Routerboard* mikrotik

   Perbedaan dengan yang diteliti yaitu: implementasi pada metode jaringan *Software Defined Network* menggunakan *Ryu controller*.

4. **Dalam penelitian Resty Annisa berjudul "Pengembangan Manajemen Jaringan Berbasis *Software Defined Network* di Politeknik Sekayu". (Juli-Desember 2017)**(Resty Annisa, 2017)

   Dalam penelitian ini metode yang digunakan adalah *action research* atau penelitian tindakan dimana peneliti mendeskripsikan, menginterpretasi dan menjelaskan suatu situasi sosial pada waktu yang bersamaan dengan melakukan perubahan atau intervensi dengan tujuan perbaikan atau partisipasi. Adapun tahapan penelitian yang dilakukan adalah

   1) tahap diagnose (*diagnosing*)
   2) Membuat rencana (*action planning*)
   3) melakukan tindakan (*action taking*).

      Pada siklus ini dilakukan pengkonfigurasian jaringan berbasis *Software Defined Network* dengan penerapan metode *djikstra*.

      a. melakukan evaluasi (*evaluating*)
      b. setelah *action taking* dianggap sudah cukup kemudian dilakukan evaluasi mengenai *QOS* pada *SDN* kemudian dibandingkan dengan jaringan konvensional.

   Perbedaan dengan yang diteliti yaitu: implementasi pada metode jaringan *Software Defined Network* menggunakan *Ryu controller* dan *openflow protocol* kemudian dibandingkan dengan jaringan konvensional menggunakan metode *simple queue* untuk *management bandwidth*.

5. **Dalam penelitian Teguh Indra Bayu berjudul "Simulasi Konsep *Software Defined Network* (*SDN*) Menggunakan *Rasberry Pi*". (2 Oktober 2018)** (Bayu & Tahan, 2018)

   Pada penelitian tersebut rumusan masalah pada penelitian ini adalah bagaimana merancang dan menerapkan fungsi *ACL* pada konsep *SDN* dengan menggunakan *controller Opendaylight* dan *openflow* sebagai protokol yang digunakan. Berdasarkan latar belakang yang telah dijelaskan, maka penelitian ini bertujuan untuk mengetahui bagaimana menerapkan fungsi *ACL* pada konsep *SDN*. Manfaat dari penelitian ini adalah guna membuat pembaca mengerti bagaimana menerapkan fungsi *ACL* ke dalam *controller Opendaylight* dalam konsep *SDN*.

   Perbedaan dengan yang diteliti yaitu: *controller* yang di pakai yaitu *Ryu controller* dengan mengimplementasikannya bersama *openflow protocol*.

6. **Dalam penelitian Moh Wahyudi Putra berjudul "Analisis Perbandingan Performansi Kontroler *Floodlight*, *Maestro, RYU, POX* dan *ONOS* dalam Arsitektur *Software Defined Network* (SDN)" (10Oktober 2018)**(Putra et al., 2018)

   Pada penelitian ini *Controller* menggunakan *protokol Openflow* untuk melakukan konfigurasi terhadap perangkat jaringan dan memilih jalur trafik data yang optimal. Dalam penelitian dipilih pengujian terhadap *kontroler Floodlight*, *kontroler Maestro*, *kontroler RYU*, *kontroler POX*, dan *kontroler ONOS*. Perlu dilakukan penelitian mengenai performansi kontroler mana yang lebih unggul dalam mendukung mekanisme *Openflow* pada *SDN*.

   Dalam penelitian ini membandingkan kinerja beberapa *controller* seperti *Floodlight*, *Maestro, RYU, POX* dan *ONOS* dalam arsitektur *Software Defined Network*. Metodologi yang digunakan dalam penelitian ini adalah *througput* dan *latency* dengan beberapa scenario pengujian yaitu:

   a. Pengujian *Throughput* Dengan Jumlah *Switch* dan *Host* bervariasi.
   b. Pengujian *Throughput* Dengan Jumlah *Switch* bervariasi.
   c. Pengujian *Latency* Dengan Jumlah *Switch* dan *Host* bervariasi.
   d. Pengujian *Latency* Dengan Jumlah *Switch* bervariasi.

   Perbedaan dengan yang diteliti yaitu: *controller* yang di pakai hanya *Ryu controller* dengan mengimplementasikannya bersama *openflow protocol* tanpa melakukan *benchmark* pada banyak *controller*.

7. **Dalam penelitian Shailendra Mishra berjudul "*Software Defined Networking*: *Research Issues*, *Challenges and Opportunities*" (Agustus 2017)**(Mishra & AlShehri, 2017)

   Dengan menggunakan *platform SDN*, para peneliti mengembangkan banyak aplikasi seperti *load balancing*, virtualisasi jaringan, jaringan hemat energi, kontrol akses dinamis dalam jaringan perusahaan, mobilitas mesin virtual, dll memecahkan masalah beberapa jaringan cabang yang berlokasi, biaya, sumber daya teknis di setiap lokasi, keahlian, terpisah kontrol pesawat untuk konfigurasi, visibilitas terdesentralisasi perangkat jaringan, pisahkan *VLAN* untuk setiap cabang, rekayasa lalu lintas yang kompleks, akses fisik terbatas, *bottleneck bandwidth* di setiap cabang, kami mensurvei literatur, sumber daya web dan buku untuk pengendali *SDN* yang ada seperti *NOX, POX, Ryu, Floodlight*, dan *OpenDaylight* dan lain-lain. Semua pengontrol ini didasarkan pada protokol *OpenFlow*.

   Pada penelitian ini membahas bahwa dalam metode SDN dapat di buat berbagai jenis apikasi jaringan seperti *load balancing*, *cloud computing* dan aplikasi lainnya dimana seluruhnya itu memiliki *controller dan protocol*. Perbedaan dengan yang akan diteliti adalah penekanan terhadap optimasi jaringan dengan hanya menggunakan Ryu *controller* sebagai *framework controller* dan protokol *openflow* pada prosesnya kemudian dibandingkan kinerja jaringan SDN dengan jaringan konvensional.

8. **Dalam penelitian Liehuang Zhu berjudul "*SDN Controllers: Benchmarking & Performance Evaluation*" (12 Februari 2019)**(Zhu et al., 2019)

   Pada penelitian yang dilakukan oleh Liehuang Zhu melakukan *benchmark* terhadap metode *Software Defined Network (SDN)* dengan berbagai *controller* yang ada saat ini yaitu *NOX POX, Floodlight, OpenDaylight (ODL), Open Network Operating System (ONOS)* and *RYU*.

   Perbedaan dengan yang akan diteliti adalah penekanan terhadap optimalisasi jaringan dengan hanya menggunakan *Ryu controller* sebagai *framework*nya dan *protocol openflow* pada prosesnya tanpa melakukan ujicoba menggunakan semua *controller* yang tersedia saat ini.

9. **Dalam penelitian Hend Abdelgader Uissa berjudul "*Software Defined Networking*" (24-26 Maret 2019)**(Eissa et al., 2019)

   Untuk memanfaatkan link redudansi dan berbagi beban di antara *switch*, protokol seperti *PVST* digunakan. Tapi konfigurasi dilakukan secara statis. Menggunakan konsep *Software Defined Network*, dengan mengimplementasikan Algoritma untuk memiliki dinamika memuat berbagi untuk *VLAN* melalui dua buah *Switch* distribusi tergantung pada beban lalu lintas jaringan mereka. Algoritma

digunakan untuk membangun aplikasi jaringan yang dijalankan oleh *Ryu* pengontrol *Switch OpenFlow* memiliki manfaat menggunakan penghitung mereka untuk memonitor lalu lintas dan menghitung jumlah paket mengalir dan entri aliran dipasang di saklar. Itu pesan yang dipertukarkan antara *Ryu* dan *Switch* harus digunakan di sini untuk meminta informasi dari *counter*. Pengontrol akan menggunakan informasi ini tentang lalu lintas ke mengkonfigurasi ulang *Switch* dan menyeimbangkan *traffic* di dalam jaringan tergantung pada beban *VLAN* saat ini

Pada penelitian yang dilakukan oleh *Hend Abdelgader Uissa* ini adalah membuat suatu jaringan computer menggunakan arsitektur *Software Defined Network (SDN)* menggunakan *protocol switch openflow* dan yang di uji pada penelitian ini adalah *load sharing* dari jaringan menggunakan *VLAN* dalam dua buah *switch* distribusi serta menunjukkan redundansi implementasi yang diusulkan dalam topologi perusahaan yang disederhanakan.

Perbedaan dengan yang diteliti adalah tidak terdapat *VLAN* dalam penelitian yang akan dilakukan walaupun tujuannya sama yaitu optimalisasi jaringan dalam sebuah perusahaan/lembaga.

10. **Dalam penelitian Dimitris Syrivelis berjudul *"Pursuing a Software Defined Information-Centric Network*" (10 Oktober 2018)**(Syrivelis et al., 2012)

Dalam penelitian ini membahas mengenai arsitektur *ICN* yang menggunakan dukungan *Software Defined Network* (*SDN*) untuk mengimplementasikan penerusan penting berfungsi dalam konteks arsitektur yang saat ini seluruhnya diimplementasikan dalam perangkat lunak. Bagian penting adalah gagasan tentang *flow*, yang di selaraskan dengan konteks arsitektur *ICN* oleh gagasan *forwarding* konsep. Berdasarkan ini, kami mengusulkan penggantian fungsi penerusan saat ini dalam prototipe *ICN* kami dengan yang secara langsung memanfaatkan kemampuan *SDN* untuk diimplementasikan fungsi penerusan berdasarkan label aliran tidak dikenal yang masuk. Sementara pekerjaan kami adalah langkah awal dalam arah menggabungkan *SDN* dan *ICN* dengan cara yang efisien, kami yakin bahwa sejak awal realisasi ide-ide kami dengan segera diuji dan ditunjukkan dalam pengaturan *testbed* kehidupan nyata. Ketersediaan ini akan mendorong memperdebatkan bagaimana *SDN* dapat mendorong ado psi *ICN* lebih lanjut penyebaran dalam waktu dekat.

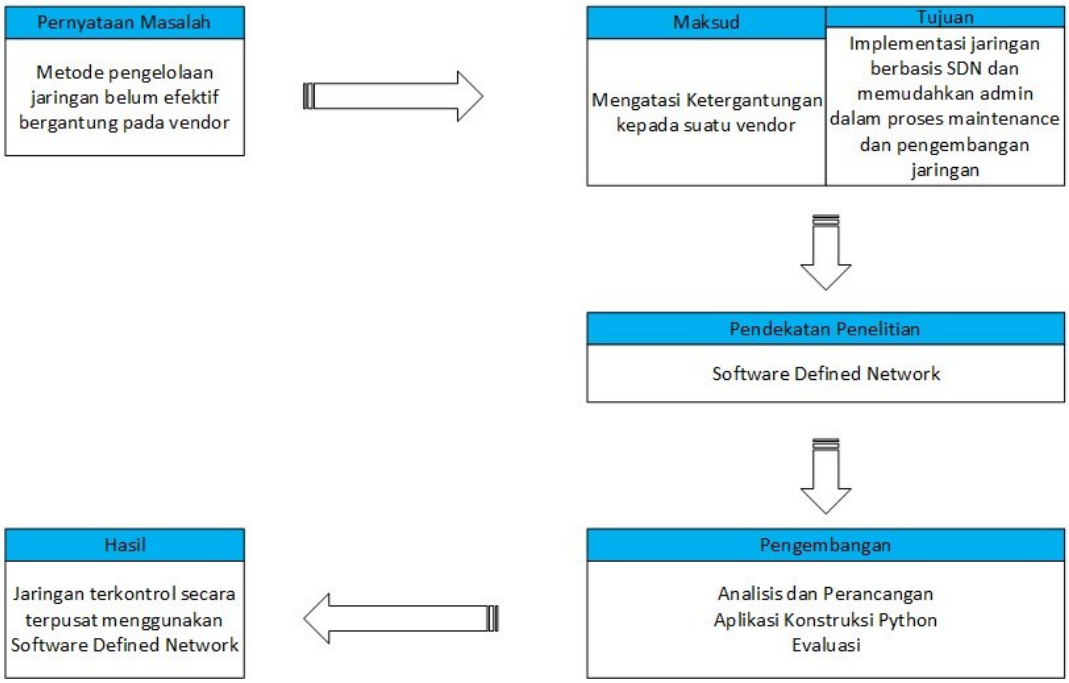
Perbedaan dengan yang diteliti adalah dalam arsitektur yang sudah ada yaitu *SDN* namun hanya berfocus membangun dan mengukur kinerja jaringan tersebut dengan demikian penelitian ini akan berfocus pada perbandingan kinerja jaringan SDN dengan jaringan konvensional.

Tabel 2.6 Penelitian Rujukan

No	Jurnal	Hasil	Perbedaan dengan yang akan dilakukan
1	**Optimasi Rute Untuk *Software Defined Networking-Wide Area Network (SDN-WAN)* Dengan Openflow Protokol**	menggabungkan 2 buah proses *Routing* Pada implementasi *SDN-WAN* berbasis *openFlow*	Melakukan Optimalisasi bandwidth management
2	**Perancangan Simulasi Jaringan Virtual Berbasis *Software-Define Networking***	Merancang dan menguji jaringan *SDN* dengan simulasi pada *Mininet*	Merancang optimalisasi *bandwidth* dan implementasi pada lingkungan FTS
3	**Analisis Kinerja dan Karakteristik Arsitektur *Software-Defined Network Berbasis OpenDaylight Controller***	pengujian dilakukan untuk mengetahui performasi dari *opendaylight controller* menggunakan berbagai jenis topologi dan juga *Routerboard* mikrotik	Penerapan *openflow* pada *Routerboard* mikrotik dengan *Ryu controller*
4	**Pengembangan Manajemen Jaringan Berbasis *Software Defined Network* di Politeknik Sekayu**	pengkonfigurasian jaringan berbasis *Software Defined Network* dengan penerapan metode *djikstra*	Mengoptimalkan *bandwidth management* menggunakan *simple queue*
5	**Simulasi Konsep *Software Defined Network (SDN) Menggunakan Rasberry Pi***	Mengsimulasikan konsep *SDN* dengan menggunakan *perangkat Rasberry Pi*	Penerapan konsep *SDN* menggunakan *Routerboard Mikrotik*
6	**Analisis Perbandingan**	pengujian terhadap *kontroler Floodlight,*	Penerapan *SDN openflow* menggunakan

	Performansi Kontroler *Floodlight, Maestro, RYU, POX* dan *ONOS* dalam Arsitektur *Software Defined Network (SDN)*	*kontroler Maestro, kontroler RYU, kontroler POX, dan kontroler ONOS*	*Ryu Controller* dan *Mikrotik Routerboard*
7	***Software Defined Networking: Research Issues, Challenges and Opportunities***	*Software Defined Network* dapat di buat berbagai jenis apikasi jaringan seperti *Load Balancing*, *Cloud Computing* dan lain	Terfocus pada *bandwidth management* dengan *Ryu Controller* dan *openflow protocol*
8	***SDN Controllers: Benchmarking & Performance Evaluation***	*benchmark* terhadap metode *Software Defined Network* dengan berbagai *controller* yang ada saat ini yaitu *NOX POX, Floodlight, OpenDaylight (ODL), Open Network Operating System (ONOS)* dan *RYU.*	Implementasi *SD*N untuk *bandwidth Management* dengan melakukan perbandingan sebelum dan sesudah di terapkan
9	***Software Defined Networking***	*load sharing* dari jaringan menggunakan *VLAN* dalam dua buah *Switch* distribusi serta menunjukkan redundansi	Menggunakan *mikrotik Routerboard* sebagai *Switch* untuk *bandwidth management*
10	***Pursuing a Software Defined Information-Centric Network***	arsitektur *ICN* yang menggunakan dukungan *Software Defined Network (SDN)*	Arsitektur *SDN* dengan dukungan perangkat mikrotik *Routerboard*

E. Kerangka Pemikiran



Gambar 2.14 Kerangka Pemikiran

Pada penelitian ini terdapat pernyataan permasalahan yang ada yaitu pengelolaan jaringan yang masih belum efektif agar dilakukan optimalisasi jaringan yang ada untuk menetapkan tujuan melalui pendekatan yang digunakan adalah SDN menjadi dasar dari hasil penelitian, pengembangan di bagi ke dalam tiga bagian yaitu analisis dan perancangan yang meliputi inventarisasi peralatan yang sudah ada dan yang akan digunakan pada proses penelitian, konstruksi aplikasi menggunakan *framework* Ryu *controller* dan *openflow switch*, setelah semua proses konstruksi dilakukan kemudian dilakukan simulasi terlebih dahulu menggunakan Mininet, Tahap ketiga yaitu evaluasi tahap ini merupakan tahap pengujian kinerja jaringan untuk memperoleh data sesuai dengan standarisasi TIPHON yaitu *throughput*, *packet loss, delay* dan rata-rata *delay*, dilanjutkan dengan implementasi jaringan yang telah dibuat untuk pengguna internet yaitu dosen, karyawan dan mahasiswa di lingkungan Fakultas Teknik dan sains.

F. Hipotesis

Hipotesis dalam penelitian ini adalah menerapkan metode *Software Defined Network* menggunakan ryu *controller* dan openflow protocol untuk optimasi penggunaan jaringan yang lebih optimal dalam penggunaan, pengembangan dan biaya yang dikeluarkan. Arsitektur jaringan juga dapat memenuhi kebutuhan karena bersifat *dinamis* dan *adaptif.*