BAB 2
DASAR TEORI

2.1 Kajian Pustaka

Penelitian Abhimata Zuhra Pramudita dan I Made Suartana, pada tahun 2020 yang berjudul "Perbandingan Performa *Controller Opendaylight* dan *POX* pada Arsitektur *Software Defined Network (SDN)"* membahas tentang perbandingan analisis *Quality of Service* (QoS) terhadap *controller Opendaylight* dan *POX*, parameter yang digunakan sebagai uji performa adalah *throughput, packet loss, delay* dan *jitter*. Penelitian ini menggunakan *tools* D-ITG sebagai *software* dalam melakukan uji performa. Berdasarkan penelitian yang telah dilakukan, diperoleh hasil bahwa *controller Opendaylight* memiliki performa yang lebih baik jika dibandingkan dengan *controller POX*. Hal ini diketahui dari data yang diperoleh di mana pada *controller* nilai rata-rata *throughput* sebesar 325,682 Mb/s, rata-rata nilai *delay* sebesar 0,313395s dan rata-rata nilai *packet loss* yang diperoleh 4,59%. Sedangkan untuk *controller POX*, nilai *throughput* yang diperoleh sebesar 318,749 Mb/s, rata-rata nilai *delay* sebesar 0,622309s dan rata-rata nilai *packet loss* sebesar 10,11%, di mana hasil ini diperoleh setelah melalui pengujian sebanyak 10 kali[2].

Penelitian yang dilakukan oleh Nanda Iryani, Afifah Dwi Ramadhani, dan Mayang Karmila Sari pada tahun 2021 yang berjudul tentang "Analisis Performansi *Routing OSPF* menggunakan *Ryu Controller* dan *POX Controller* pada *Software Defined Network* (SDN)" membahas tentang bagaimana performansi dari *routing* OSPF menggunakan *Ryu controller* dan *POX controller* pada topologi jaringan *fat free* berdasarkan parameter QoS yang berstandar TIPHON. Tujuan dari penelitian ini adalah untuk melakukan Analisa terhadap performansi *Ryu controller* dan *POX controller* pada jaringan topologi *fat free* yang berdasarkan pada parameter QoS, yaitu *delay, jitter, packet loss* dengan standar TIPHON. Melalui penelitian ini diperoleh hasil bahwa transmisi data menggunakan *traffic protocol TCP* dengan *Ryu controller* dan *protocol* UDP yang lebih baik, yaitu dengan nilai *delay* sebesar 49,44% dan *delay* yang lebih stabil sebesar 0,01%. *Jitter* yang dihasilkan sebesar 27,59% lebih baik daripada *POX*

controller yang sebesar 99,97% sedangkan *traffic protocol UDP* menggunakan *Ryu controller* lebih baik sebesar 0,035%. *Packet loss* dari kedua *controller* didapatkan hasil protokol TCP yang sangat bagus, yaitu sebesar 0% sedangkan protokol UDP menggunakan *POX controller* sebesar 83,13% menggunakan *Ryu controller* 16,87%[5].

Penelitian yang dilakukan oleh Nurul Zaenal Abidin yang berjudul "Analisis Performansi *Controller POX* dan *Ryu* pada Jaringan *Software Defined Network* dengan Protokol *Spanning Tree*" pada tahun 2021 membahas tentang bagaimana performansi dari tiap-tiap kontroler yang akan diuji dengan menggunakan protokol *spanning tree* sebagai salah satu parameternya. Selama pengujian berlangsung dialiri *traffic* UDP dengan variasi *background traffic* mulai dari 50 sampai 200 MB. Hasil yang didapatkan yaitu *controller ryu* memiliki nilai QoS yang lebih baik dengan rata-rata nilai *throughput* sebesar 3.182,197 Kbps, rata-rata nilai *packet loss* sebesar 0%, rata-rata nilai *delay* sebesar 0,050 ms dan rata-rata nilai *jitter* 0,014 ms, daripada POX yang memiliki rata-rata nilai *throughput* sebesar 2.009,392 Kbps, rata-rata nilai *packet loss* sebesar 0,524%, rata-rata nilai *delay* sebesar 82,305 ms dan rata-rata *jitter* 6,232 ms[1].

2.2 Dasar Teori

2.2.1 Jaringan Komputer

Jaringan komputer adalah jaringan telekomunikasi yang memungkinkan antar komputer untuk saling berkomunikasi dengan bertukar data, jaringan komputer dibangun dengan kombinasi *hardware* dan *software.* Saat 2 atau lebih komputer saling berkomunikasi atau bertukar data sebenarnya ada bagian-bagian dari jaringan komputer yang menjadi pihak yang menerima atau meminta layanan disebut dengan *client* dan yang memberikan atau mengirimkan disebut dengan *server. Design* seperti ini disebut dengan sistem *client-server*[6].

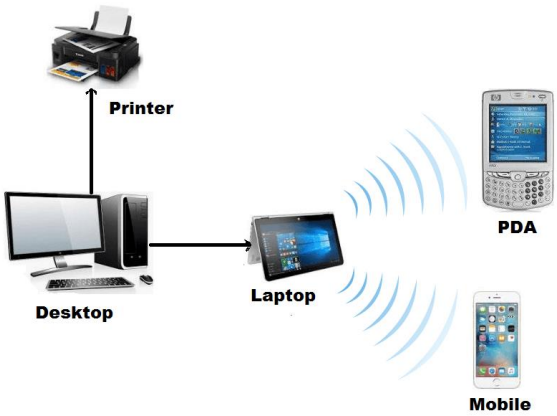
A. Klasifikasi Jaringan Komputer

Ada beberapa jenis jaringan komputer yang diklasifikasikan menurut cakupan areanya, yaitu:

1. *Personal Area Network*

   Personal Area Network (PAN) merupakan jaringan komunikasi dalam satu perangkat lain dengan perangkat lainnya didalam jarak yang sangat dekat

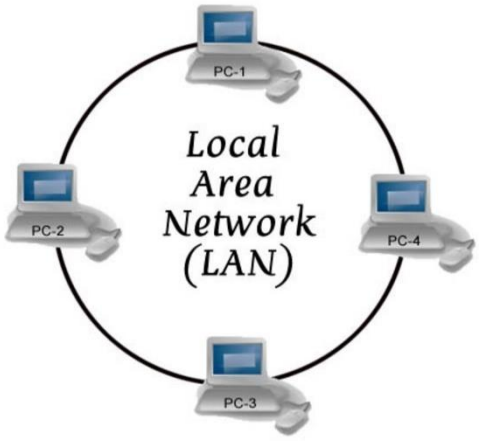
atau hanya beberapa meter. Selain itu, PAN juga bisa dipakai untuk komunikasi diantara perangkat pribadi sendiri atau komunikasi interpersonal. Kontrol PAN dilakukan memakai authoritas pribadi, sedangkan teknologi yang dipakai adalah *Bluetooth* dan juga *Wireless Application Protocol (WAP)*[7]. Gambar 2.1 merupakan contoh cakupan jaringan PAN.



Gambar 2.1 Contoh cakupan PAN

2. *Local Area Network*

Local Area Network (LAN) adalah konsep yang menghubungkan perangkat jaringan dalam jarak yang relatif pendek. Biasanya digunakan untuk gedung sekolah, kantor, rumah, dan lain-lain. Konsep jaringan LAN ini cenderung menggunakan konektivitas tertentu, terutama *Ethernet* dan Token *Ring*.

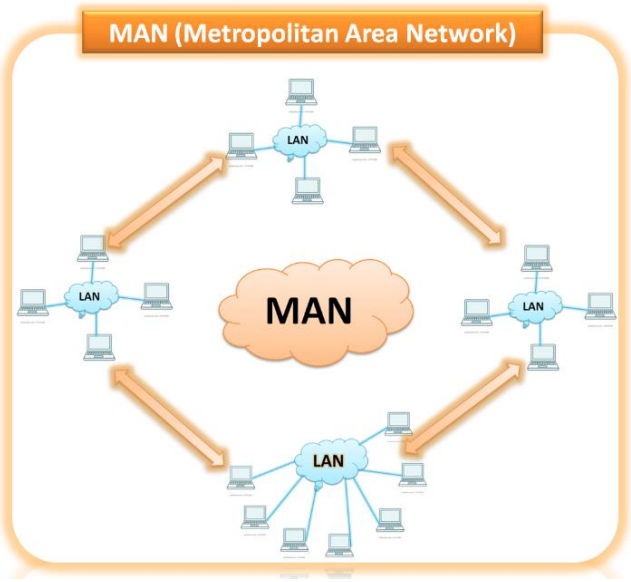


Gambar 2.2 Contoh cakupan LAN

Ada juga LAN yang menggunakan teknologi jaringan *Wireless* atau nirkabel dengan WI-FI dan dikenal dengan nama *Wireless Local Area Network (WLAN)*[8]. Untuk contoh cakupan LAN dapat dilihat pada gambar 2.2.

3. *Metropolitan Area Network*

Metropolitan Area Network (MAN) adalah konsep yang menghubungkan perangkat jaringan dari satu kota ke kota lainnya. Jika penggunaan LAN sudah tidak memungkinkan untuk membangun jaringan maka jaringan MAN akan digunakan, karena cakupannya lebih besar dibandingkan LAN maka MAN menggunakan perangkat khusus dan memerlukan operator telekomunikasi yang bertugas sebagai penghubung antar jaringan komputer[9]. Gambar 2.3 merupakan contoh cakupan jaringan MAN.



Gambar 2.3 Contoh cakupan MAN

4. *Wide Area Network*

WAN atau *Wide Area Network* adalah konsep yang menghubungkan perangkat jaringan komputer yang mencakup wilayah super luas dan menggunakan peralatan yang super canggih apabila dibandingkan dengan MAN dan LAN.

Konsep jaringan ini sendiri biasanya digunakan untuk menghubungkan suatu jaringan dari negara satu dengan negara lainnya alias antar negara bahkan bisa juga antar benua[10]. Untuk mengetahui contoh cakupan WAN, perhatikan gambar 2.4.

Gambar 2.4 Contoh Cakupan WAN

B. Topologi Jaringan Komputer

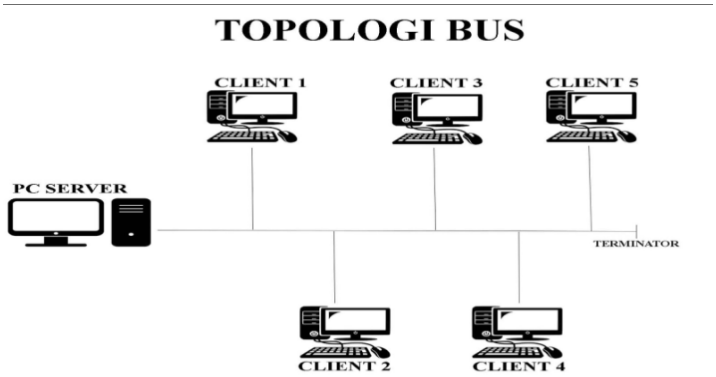
Dalam pembangunan jaringan komputer ini sendiri tidak lepas dari namanya topologi, di mana topologi ini sendiri dapat diartikan sebagai bentuk atau struktur virtual jaringan yang mengacu pada tata letak perangkat yang nantinya akan terhubung walaupun bentuk ini tidak selalu sesuai dengan tata letak fisik sebenarnya dari perangkat jaringan.

Topologi jaringan dapat dikategorikan ke dalam tipe dasar berikut, yakni:

1. Topologi *Bus*

   Topologi *bus* merupakan salah satu jenis topologi jaringan yang hanya menggunakan satu kabel saja sebagai media komunikasi atau media transmisi dan kabel tersebut yang akan menjadi pusat bagi seluruh *server* yang saling terhubung. Masing-masing komputer dihubungkan ke kabel utama dengan menggunakan konektor BNC yang kemudian diakhiri dengan terminator. Topologi Bus merupakan topologi yang memiliki kecepatan pengiriman data yang tinggi dan jumlah terminal dapat ditambah dan dikurangi tanpa mengganggu kinerja komputer yang sedang berjalan.[11]. Berikut beberapa karakteristik topologi Bus:

1. Sangat sederhana dalam instalasi
2. Node- node dihubungkan secara serial sepanjang kabel, dan pada kedua ujung kabel ditutup dengan terminator. Terdapat kabel utama sebagai pusat lalu lintas data.
3. Sangat ekonomis dan dalam segi pembiayaan.
4. Paket-paket data saling bersimpangan pada suatu kabel
5. Tidak diperlukan hub atau switch, yang banyak diperlukan adalah Tconnector dan konektor BNC pada setiap Ethernet card.
6. Problem yang sering terjadi adalah jika salah satu node rusak, maka jaringan keseluruhan dapat down, sehingga seluruh node tidak bisa berkomunikasi dalam jaringan tersebut. Gambar 2.5 merupakan contoh topologi bus:

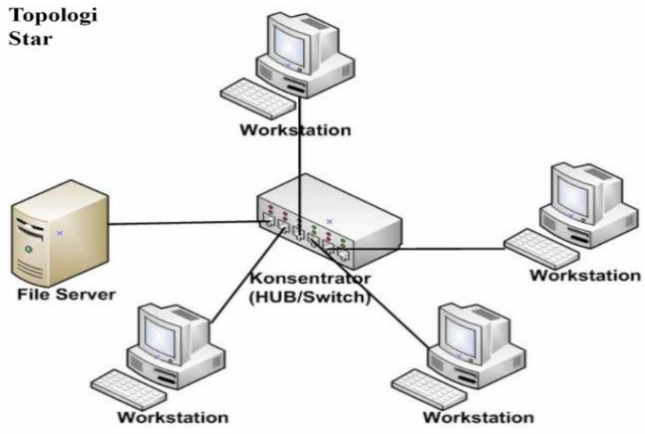


Gambar 2.5 Topologi *Bus*

2. Topologi *Star*

Topologi *star* adalah topologi yang memiliki 1 penghubung sebagai pusat (HUB atau *Switch*) dari setiap komputer yang terhubung. Hub berada di *central* dan berfungsi untuk menghubungkan satu komputer ke setiap komputer yang terhubung dan menghubungkan komputer ke *file server*.

Cara kerjanya yaitu apabila ingin bertukar data satu sama lain maka data itu akan mengalir ke HUB atau *switch* terlebih dahulu baru kemudian akan menuju ke komputer yang meminta atau yang akan menerimanya[12]. Gambar 2.6 merupakan contoh gambar topologi star:

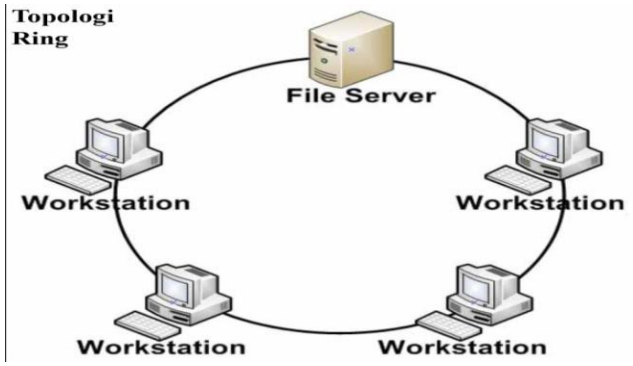


Gambar 2.6 Topologi *Star*

3. Topologi *Ring*

   Topologi *Ring* adalah topologi jaringan yang bentuk rangkaiannya menyerupai cincin dan berupa titik yang mana masing-masing titik bagian kanan dan kiri terhubung ke dua titik lainnya sampai seluruh komputer saling terhubung.

   Titik yang ada pada topologi ini berfungsi untuk memperkuat sinyal disetiap rangkaiannya atau bisa juga disebut *repeater*, dengan metode seperti ini maka sinyal dan aliran data akan tetap stabil[13]. Gambar 2.7 adalah contoh gambar topologi *ring:*



Gambar 2.7 Topologi *Ring*

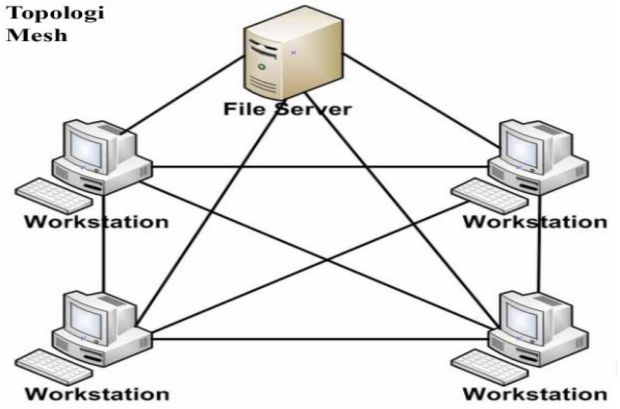
4. Topologi *Mesh*

   Topologi *mesh* adalah topologi yang jaringannya dapat terhubung satu sama lain secara acak dan tidak teratur. Karena komputer langsung

terhubung dengan komputer yang dituju maka arus data dapat langsung dilakukan dengan cepat tanpa harus melalui komputer lain.

Masing-masing komputer setidaknya memiliki 2 jenis sambungan, pertama ialah kabel yang terhubung dengan komputer lainnya dan kedua ialah kabel lainnya yang terhubung ke *file server*[14].

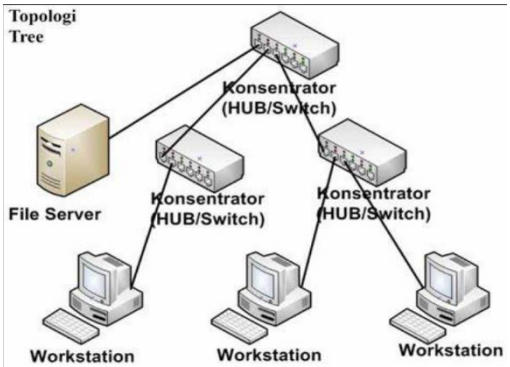
Gambar 2.8 adalah contoh gambar topologi *mesh:*



Gambar 2.8 Topologi *Mesh*

5. Topologi *Tree*

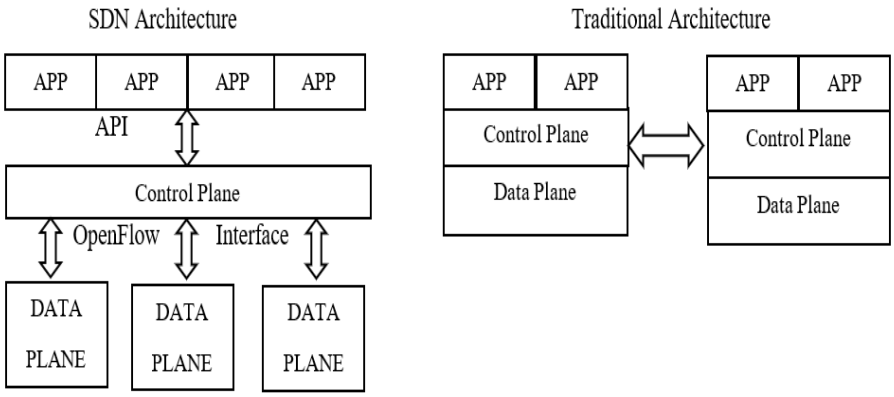
   Topologi *tree* adalah adalah kombinasi dari topologi *star* dan topologi *bus* namun yang membedakannya adalah topologi ini terdapat banyak *hub* atau *switch* dalam jaringan dan sistem hierarkinya, di mana masing-masing *hub* atau *switch* terhubung dengan *file server*[15]. Gambar 2.9 adalah contoh gambar topologi *tree*.



Gambar 2.9 Topologi *Tree*

C. Perkembangan Jaringan Konvensional

Konsep daripada jaringan SDN adalah melakukan pemisahan antara *control plane* dan *data plane*, di mana *data plane* berada pada perangkat networking dan *control plane* berada pada entitas terpisah yang diketahui sebagai *"controller"* yang kemudian akan menentukan bagaimana perilaku jaringan.



Gambar 2.10 Perbedaan Jaringan Konvensional dan Jaringan SDN

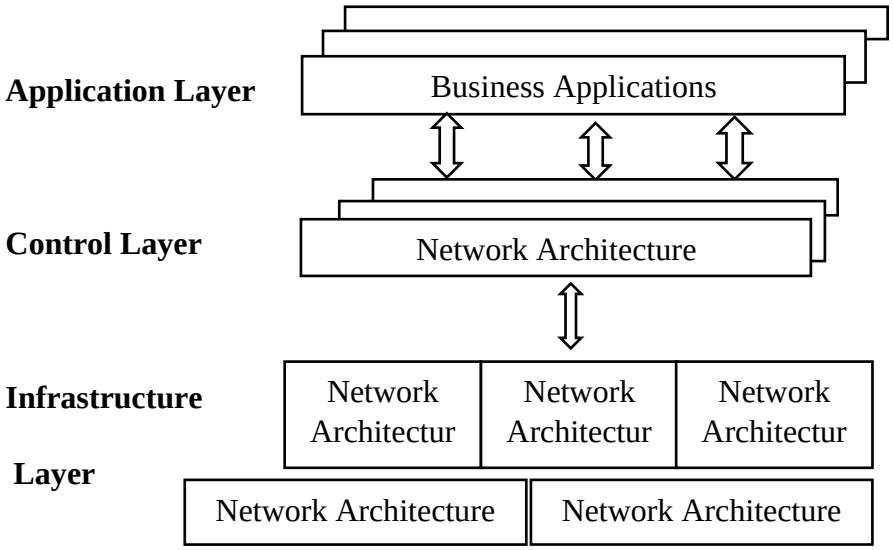
Sedangkan pada gambar 2.10 dapat dilihat bagaimana bentuk arsitektur daripada jaringan konvensional. Konsep pada jaringan konvensional yaitu di mana *control plane* dan *data plane* berada dalam satu perangkat *networking* yang sama. Sehingga, ketika ingin membangun sebuah jaringan yang besar, maka dibutuhkan konfigurasi yang lebih kompleks dan cukup sulit karena perangkat harus dikonfigurasi secara satu persatu, hingga muncullah teknologi SDN yang dapat melakukan konfigurasi terpusat dengan lebih cepat dan efisien[16].

D. Software Defined Network (SDN)

Software Defined Network (SDN) merupakan sebuah konsep pendekatan baru dengan tujuan untuk mendesain, mengelola, dan membangun jaringan komputer melalui pemisahan terhadap *control plane* dan *data plane*. *Control Plane* merupakan sebuah komponen pada jaringan yang berfungsi untuk mengontrol jaringan, yaitu termasuk didalamnya melakukan konfigurasi sistem, manajemen jaringan, menentukan informasi *routing table* dan *forwarding table*.

Data plane merupakan komponen yang bertanggungjawab meneruskan paket, menguraikan *header* paket, mengatur QoS dan enkapsulasi paket. Konsep utama pada SDN adalah melakukan sentralisasi jaringan dengan semua pengaturan yang berada pada *control plane*. Dalam SDN, kecerdasan jaringan secara logis terpusat pada *controller* dan perangkat jaringan berperan menjadi *packet forwarding* sederhana yang kemudian dapat dikonfigurasi melalui *open interface* pada *data plane*. *Open interface* inilah yang disebut dengan *openflow*[17].

Openflow merupakan sebuah protokol atau standar komunikasi antarmuka yang berada antara *control* dan *forwarding layer*. *Openflow* diimplementasikan dengan dua sisi yaitu dari sisi perangkat jaringan, contohnya seperti *router* dan *switch* serta pada sisi *controller* SDN sehingga memungkinkan untuk mendapatkan akses langsung dari *forwarding layer*. Arsitektur SDN terbagi menjadi 3 jaringan *layer,* jaringan yang pertama adalah *application layer,* kemudian *control layer* dan jaringan yang terakhir adalah *infrastructure layer* seperti yang dapat dilihat pada gambar 2.11.



Gambar 2.11 *Logical view* dari arsitektur SDN

1. *Application Layer*

   Application layer merupakan bagian *layer* yang bertugas sebagai antarmuka untuk melakukan pengontrolan jaringan dan melakukan fungsi konfigurasi, fungsi evaluasi dan fungsi kontrol. Layer ini

diperoleh dari integrasi dengan *control layer* sehingga dapat dipahami oleh pengelola jaringan.

2. *Control Layer*

   Seperti namanya, *control layer* berfungsi sebagai otak dalam aktivitas jaringan dan berfungsi sebagai *control plane*.

3. *Infrastructure Layer*

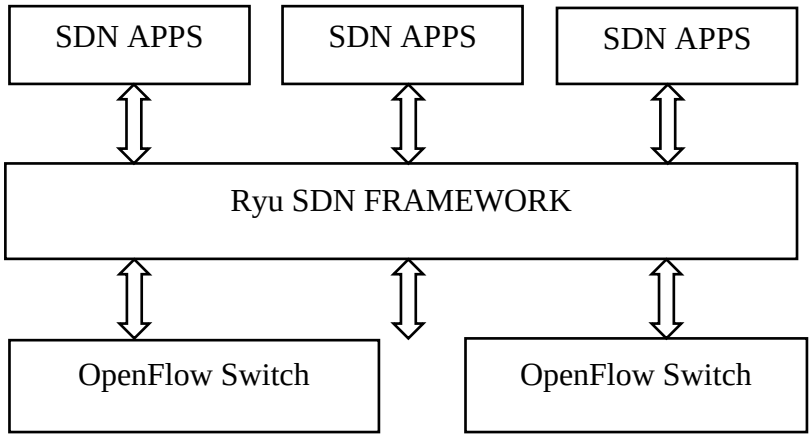
   Infrastruktur layer berada pada lapisan terbawah dalam arsitektur SDN. Lapisan ini berisi perangkat keras *(hardware*) yang memiliki peran sebagai *forwarding plane* berdasarkan pada *data plane*, perangkat keras dapat berupa *router* dan *switch*[18].

   Berikut beberapa jenis dari *controller* SDN:

   a. *Ryu*

      Ryu merupakan kerangka kerja pada jaringan SDN yang berbasis komponen. *Ryu* menyediakan komponen perangkat lunak dengan *API* yang telah terdefenisikan dengan dengan baik yang memudahkan pengguna untuk membuat aplikasi manajemen dan kontrol jaringan baru. *Ryu* sepenuhnya diimplementasikan dengan *python*[2]. Ryu banyak digunakan karena mempunyai beberapa keunggulan dibanding dengan kontroler lain,diantaranya:

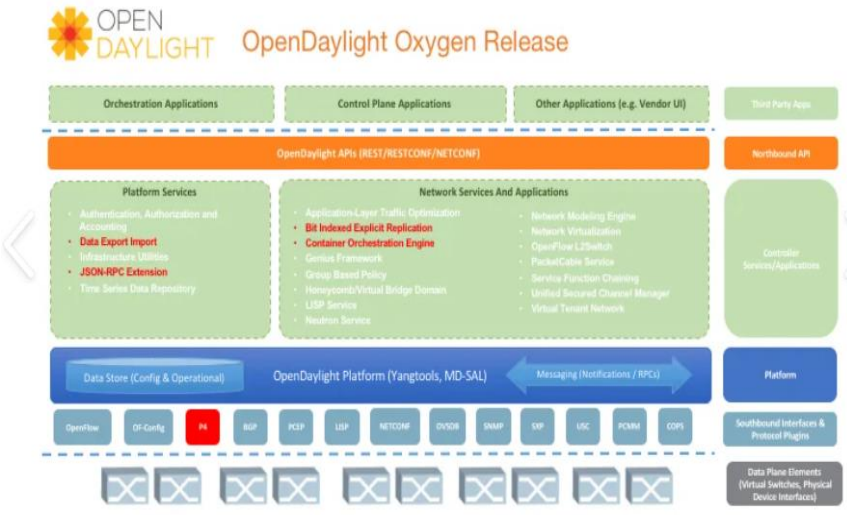
      - Ryu menyediakan banyak komponen yang berguna untuk aplikasi SDN
      - Ryu dapat digunakan untuk menggabungkan komponen untuk membangun aplikasi
      - Ryu diketahui memiliki tingkat keamanan yang cukup baik dan menjadi salah satu kontroler yang paling baik
      - Komponen lama pada Ryu dapat dimodifikasi sesuai dengan kebutuhan dan dapat diterapkan pada komponen yang baru. Gambar 2.12 merupakan bentuk dari kontroler *Ryu.*



Gambar 2.12 Ryu Framework

b. *Opendaylight*

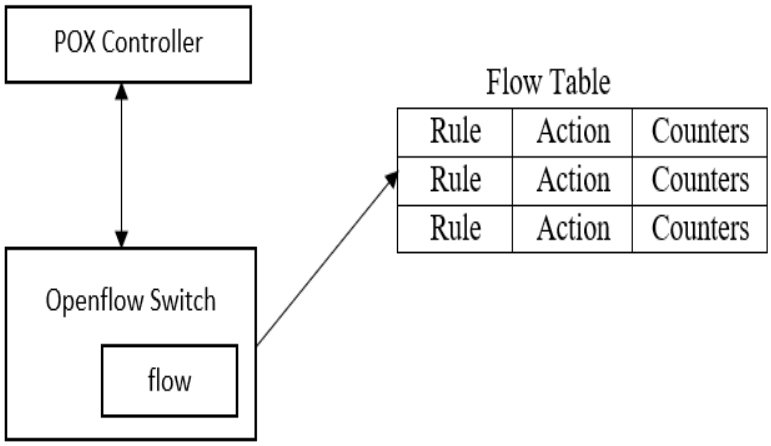
Opendaylight controller merupakan sebuah *platform* yang digunakan untuk melakukan pengembangan serta pemodelan terhadap *network control software*. *Opendaylight* menggunakan *java* sebagai bahasa pemrogramannya. *Opendaylight controller* bekerja pada *layer control plane* yang berperan sebagai sebuah *network controller*. *Opendaylight* juga memiliki beberapa *tools* yang digunakan sebagai implementasi dari konsep SDN yang sesuai dengan kebutuhan pengguna. Gambar 2.13 merupakan logo dari *opendaylight controller*[19].



Gambar 2.13 *Opendaylight Controller*

c. *POX*

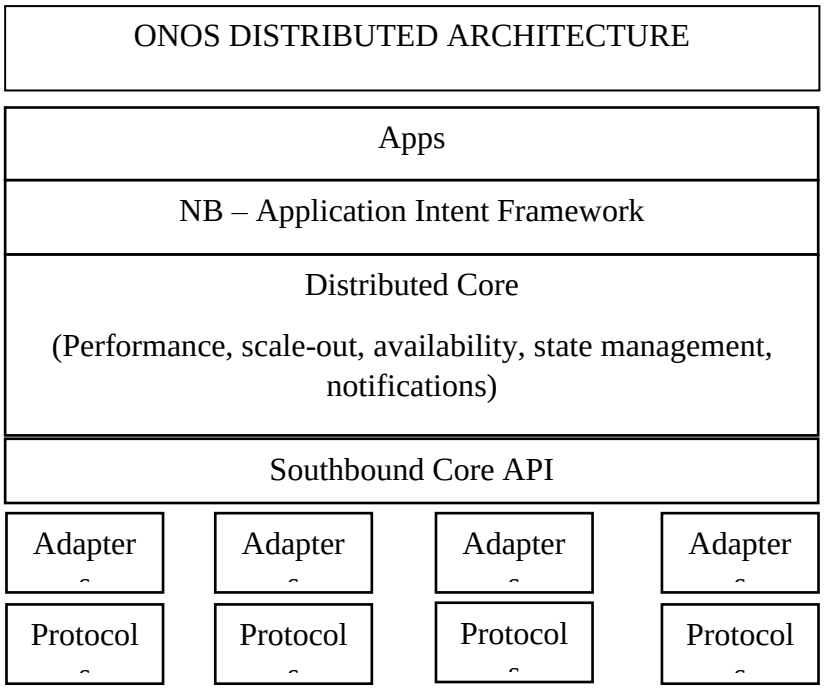
POX adalah *platform* yang digunakan untuk pengembangan dan pemodelan pada *network control software*. POX menggunakan *python* dalam bahasa pemrogramannya. POX bekerja pada *layer control plan* sebagai sebuah *network controller*. Dalam arsitektur, *routeflow* POX berada pada RFProxy yang bertanggungjawab untuk interaksi dengan *Openflow switch (datapath)* melalui protokol *openflow*. POX memiliki beberapa komponen yang dapat digunakan ulang untuk membuat SDN *controller* sesuai dengan kebutuhan pengguna. Pada gambar 2.14 merupakan penjelasan dari POX *controller*[3].



Gambar 2.14 POX *Controller*

d. ONOS

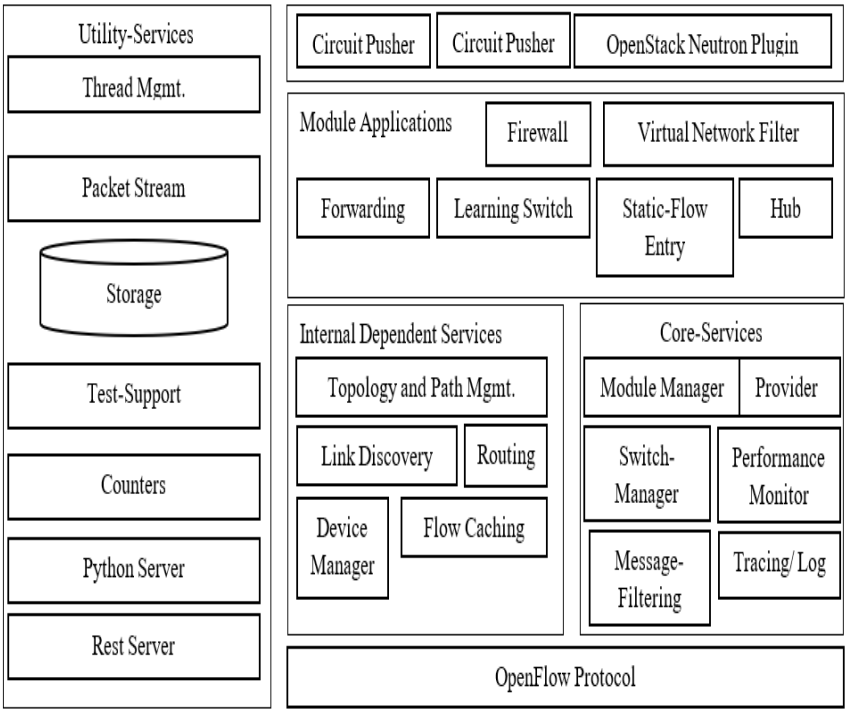
ONOS *(Open Network Defined Networking)* merupakan sebuah sistem operasi jaringan SDN *open source* terkemuka yang bertujuan untuk membangun solusi SDN generasi selanjutnya yang ditargetkan secara khusus di *Service Provider* dan *Mission Critical Network.* ONOS dibangun untuk memberikan ketersediaan tinggi (*high availability), scale-out,* dan performansi kinerja jaringan yang dibutuhkan. Kontroler ini berbasis java[20] . Gambar 2.15 merupakan arsitektur dari kontroler ONOS.



Gambar 2.15 ONOS *Controller Architecture*

e. Floodlight

Floodlight merupakan kontroler yang ada pada SDN yang memiliki bahasa pemrograman berbasis java.



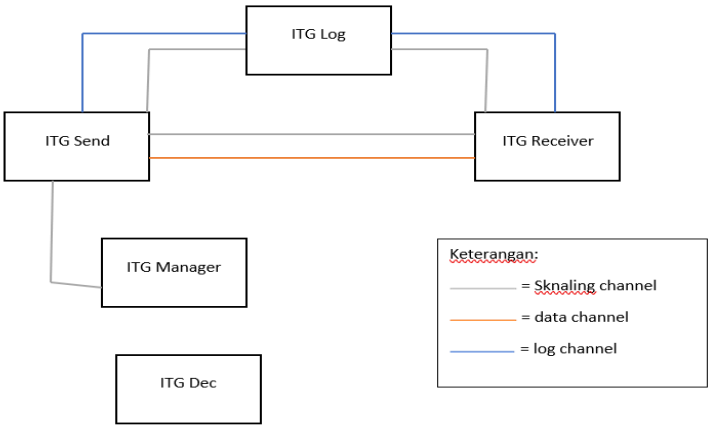
Gambar 2.16 Tampilan *Floodlight Controller*

Floodlight dikembangkan oleh *Big Switch Network Floodlight* dan didukung oleh *feature Openflow* yang membuatnya bisa mengatur aliran data pada lingkungan SDN.

Floodlight merupakan salah satu kontroler *enterprise* terbuka berlisensi *apache* yang dikembangkan oleh komunitas pengembang di *Big Switch Network.* Pada penerapannya, *floodlight* menggunakan bahasa pemrograman Java[3]. Gambar 2.16 merupakan gambaran dari *floodlight controller*.

2.2.2 D-ITG

Distributed Internet Traffic Generator (D-ITG) adalah sebuah *platform* yang mampu menghasilkan lalu lintas data ditingkat paket secara akurat. D-ITG juga merupakan alat pengukuran jaringan yang dapat mengukur metrik kinerja secara paling umum (misalnya menghitung nilai QoS) pada tingkat paket[1].

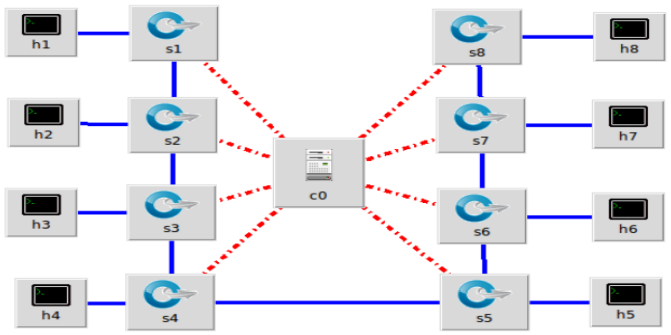


Gambar 2.17 D-ITG *Software Architecture*

2.2.3 Emulator (mininet)

Mininet merupakan sebuah emulator untuk membangun jaringan yang terdiri atas *controller, switch* dan *host*. *Mininet* merupakan sebuah sistem untuk melakukan *rapid prototyping* jaringan yang sangat besar pada sumber daya yang terbatas. *Mininet* adalah solusi yang dianggap paling unggul dalam hal kemudahan penggunaan, performansi, akurasi dan

skalabilitas[21]. Mininet pada penelitian ini digunakan untuk simulasi jaringan SDN dengan bentuk topologi yang menyesuaikan.



Keterangan:
h = host (perangkat)
co = *controller*
s = *switch*

Gambar 2.18 Tampilan Mininet

2.2.4 *Quality of Services (QoS)*

Quality of Service (QoS) adalah didefinisikan sebagai suatu metode untuk mengetahui kualitas dari kinerja jaringan yang dirasakan secara *riil*. Tujuan QoS adalah untuk memenuhi kebutuhan-kebutuhan layanan yang berbeda, yang menggunakan infrastruktur yang sama.

Parameter *Quality of Service* pada penelitian ini terdiri dari:

1. *Throughput*

   Throughput yaitu kecepatann (*rate*) transfer data efektif yang diukur dalam satuan *bit per second* (bps). *Throughput* adalah jumlah total kedatangan paket yang sukses diamati pada tujuan selama interval waktu tertentu dibagi dengan durasi interval waktu tersebut[22]. Untuk mengetahui hasil dari nilai *throughput*, dapat dilihat pada penjelasan 2.1.

$$\text{Throughput} = \frac{packet\ received\ (kb)}{time\ transmitted\ (s)} \quad \ldots\ldots(2.1)$$

Klasifikasi standarisasi nilai throughput berdasarkan TIPHON (1999) diuraikan pada tabel 2.1.

Tabel 2.1 Klasifikasi Nilai *Throughput*

Kategori Throughput	Throughput (%)
Sangat Bagus	>1200 Kbps
Bagus	700 s/d 1200 Kbps
Sedang	338 s/d 700 Kbps
Jelek	0 s/d 338 Kbps

2. *Delay*

Delay (latency) merupakan jeda waktu yang ditempuh paket ketika dikirimkan dan diterima. *Delay* dapat dipengaruhi oleh jarak, media fisik, congesti atau juga waktu proses yang lama [23].

Klasifikasi standarisasi delay berdasarkan TIPHON (1999) diuraikan pada tabel 2.2.

Tabel 2.2 Klasifikasi Nilai *Delay*

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

3. *Jitter*

Jitter diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket diakhir perjalanan jitter. *Jitter* juga dapat didefenisikan sebagai variasi *delay* yang berhubungan erat dengan *latency*. Jitter dapat diamati dalam karakteristik seperti frekuensi pulsa berturut-turut, sinyal amplitudo, atau fase dari sinyal periodik[24].

Klasifikasi standarisasi delay berdasarkan TIPHON (1999) diuraikan pada tabel 2.3.

Tabel 2.3 Klasifikasi Nilai *Jitter*

Kategori Jitter	Besar Delay
Sangat Bagus	0 ms s/d 75 ms
Bagus	0 s/d 75 ms
Sedang	75 s/d 125 ms
Jelek	125 s/d 225 ms

4. *Packet Loss*

Packet loss merupakan jumlah paket yang hilang saat pengiriman paket data ke tujuan, hal ini dapat disebabkan *collision* dan *congestion* pada jaringan[24].

$$PL = \left(\frac{data\ yang\ dikirim - data\ yang\ diterima}{paket\ data\ yang\ dikirim}\right) \times 100\% \ldots. \quad (2.2)$$

Terdapat beberapa hal yang dapat memicu timbulnya *packet loss:*

1. Jaringan kelebihan beban
2. Perangkat yang kadaluwarsa dan tidak stabil
3. Kesalahan software/hardware, dst.

Klasifikasi standarisasi *delay* berdasarkan TIPHON (1999) diuraikan pada tabel 2.4.

Tabel 2.4 Klasifikasi Nilai *Packet Loss*

Kategori Degradasi	Besar Delay
Sangat Bagus	0% - 3%
Bagus	3% - 15%
Sedang	15% - 25%
Jelek	25 %