BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian Hend Abdelgader Eissa, Kenz A. Bozed, dan Hadil Younis, pada tahun 2019 yang meneliti teknologi *Software Defined Networking* secara detail untuk menganalisis arsitektur *Software Defined Networking* dan standar *OpenFlow* secara detail, serta melakuan implementasi ke jaringan perusahaan minyak. Hasil dari penelitian tersebut bahwa *Software Defined Networking* memudahkan administrasi jaringan, mempercepat inovasi, mengurangi biaya, dan bisa menerapkan pemograman pada jaringan serta mengoptimasi tingkat *load sharing* pada perangkat *switch* [1].

Penelitian Cristian Andrei BARON, pada tahun 2018 yang meneliti *NoSQL Riak* dan *Redis database*. Hasil penelitian tersebut menyebutkan bahwa penggunaan *Redis database* untuk penyimpanan data yang besar seperti gambar ataupun video kurang cocok digunakan, akan tetapi untuk penggunaan *database* sederhana *Redis database* dapat memindahkan data secara cepat dan lebih baik dari *Riak database* sehingga cocok untuk pembuatan *database* untuk *list* sederhana [2].

Penelitian Cheng Li, Zhengrui Qin, Ed Novak, dan Qun Li, pada tahun 2018 yang meneliti tentang pengamanan infrastruktur jaringan *Software Defined Networking* pada jaringan *IoT-Fog* dari penyerangan *MitM*. Hasil dari penelitian ini yaitu dengan penggunaan metode *Bloom Filter* untuk mendeteksi penyerangan jaringan sederhana lebih efisien karena untuk sistem pendeteksinya menggunakan distribusi random yang dimana dalam menentukan *list* alamat IP yang terkena penyerangan ditentukan secara acak dan cepat sehingga walau pada hasil monitor alamat IP yang ditampilkan tidak rapi akan tetapi daftar alamat IP yang berhasil dikenai *Bloom Filter* sudah lengkap *dan Bloom Filter* ringan untuk penerapannya pada sebuah jaringan *Software Defined Networking* [3].

Penelitian Nurul Zaenal Abidin, pada tahun 2021 yang meneliti tentang performansi antara penggunaan *Ryu* dan *Pox* kontroler pada jaringan SDN dengan protokol *spanning tree*. Hasil dari penelitian ini yaitu penggunaan *Ryu* sebagai

kontroler jaringan SDN dengan protokol *spanning tree* lebih baik dari pada penggunaan *Pox* sebagai kontroler, dengan hasil *throughput Ryu* lebih besar dari pada *Pox*, nilai *packet loss* antara *Ryu* dan *Pox* yaitu 0% dan 0,54%, dan nilai *delay* yang dihasilkan oleh *Ryu* lebih kecil dari pada *Pox*. Sehingga dapat disimpulkan penggunaan *Ryu* sebagai kontroler lebih baik performansinya dari pada penggunaan *Pox* sebagai kontroler jaringan SDN [4].

Penelitian O. Khazar , pada tahun 2020 yang meneliti tentang SQL dan Non-SQL *database*. Hasil dari penelitian tersebut yaitu pada penggunaan *database* yang bersifat Non-SQL cocok digunakan untuk pembuatan daftar tulis dan daftar *string* sederhana sehingga cocok digunakan untuk penggunaan yang memerlukan pengolahan database yang sederhana dan cepat, sedangkan untuk penggunaan SQL *database* cocok digunakan untuk penyimpanan tabel, gambar, dan *list* gabungan dari gambar dan teks sehingga cocok untuk penggunaan *database* kompleks dan membutuhkan banyak jenis penyimpanan data [5].

2.2 DASAR TEORI

2.2.1 *Software Defined Networking*

Software Defined Networking (SDN) adalah pemisah fungsi perangkat fisik di dalam sebuah jaringan antara *control plane* dan *forwarding plane*, dimana *control plane* menjadi pusat pengatur untuk kinerja beberapa *devices* [1]. SDN memisahkan sisi *control plane* dari *data plane*, kemudian SDN mengintegrasi *conrol plane* sehingga satu *control plane* dapat mengatur lebih banyak elemen *data plane*.

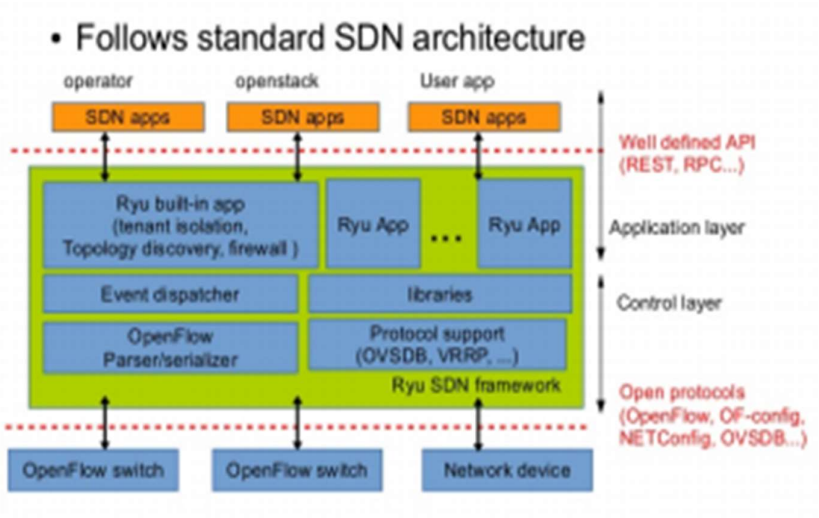
Pemisahan dari *control plane* dan *data plane* didefinisikan sebagai *Application Programming Interface* (API) diantara *Network device* dan *controller* jaringan SDN [7]. Contoh API pada jaringan SDN adalah protokol *OpenFlow,* sebuah *switch* dengan antarmuka yang dapat menerapkan pemograman dapat mengijinkan *controller* untuk berkomunikasi dan memasang konfigurasi atau *rule* pada *switch* tersebut. *OpenFlow switch* dapat berfungsi sebagai *router, switch, firewall,* dan *network address translator* [6].

Konsep utama pada SDN adalah sentralisasi jaringan dengan semua pengaturan berada pada control plane [6]. Dalam SDN terdapat protokol yang paling menonjol yaitu Open Flow [4]. Open Flow adalah sebuah protokol atau standar komunikasi antarmuka yang berada antara control dan forwarding layer. Selain itu, konsep dari SDN sendiri dapat mempermudah dan mempercepat inovasi pada jaringan sehingga diharapkan muncul ide-ide baru yang lebih baik dan dapat di implementasikan [7]. Pada arsitektur SDN terdapat dua komponen utama, yaitu Control Plane dan Data Plane. Control Plane adalah komponen pada jaringan yang berfungsi untuk mengontrol jaringan, yaitu konfigurasi sistem, manajemen jaringan, menentukan informasi routing table dan for warding table. Data Plane merupakan komponen yang bertanggungjawab meneruskan paket, menguraikan header paket, mengatur QoS, dan enkapsulasi paket [1].

Arsitektur SDN diterapkan pada perangkat MikroTik untuk diamati performansi dan karakteristik yang dimiliki oleh jaringan tersebut kemudian dibandingkan dengan arsitektur tradisional. Secara umum dalam perangkat jaringan terdapat tiga proses utama, yaitu Control Plane dan Data Plane. Control Plane adalah komponen pada jaringan yang berfungsi untuk mengontrol jaringan, yaitu konfigurasi sistem, manajemen jaringan, menentukan informasi routing table dan forwarding table. Data Plane adalah bagian yang bertanggung jawab memforward/ meneruskan paket, selain itu juga menguraikan header paket, mengatur QoS, dan enkapsulasi paket [4].

2.2.1 *Ryu*

Ryu merupakan salah satu kontroler jaringan SDN yang banyak digunakan pada penelitian, bersifat *open source* dan berada dalam naungan lisensi *Apache 2.0*, menggunakan bahasa *Python* yang sepenuhnya digunakan sebagai inti pemograman sistem *Ryu* [2]. Dikembangkan dan didukung oleh NTT *cloud data center* serta *GitHub* sebagai penyedia *Main source code* yang dikembangkan oleh komunitas *Ryu* sendiri sehingga memudahkan penggunanya untuk belajar *Ryu* [2].



Gambar 2.1 Arsitektur SDN pada *Ryu controller* [2].

Sama seperti kontroler SDN lainnya, *Ryu* mendukung protokol *OpenFlow*, Ryu dapat membuat sebuah paket *OpenFlow* dan mengatur *event* dari paket yang keluar atau masuk. *Ryu* mempunyai penggunaan fungsi *library* yang mendukung semua operasi pemrosesan sebuah paket, *Ryu* mendukung protokol lainnya seperti *XFlow, OF-Config, NETCONF, OpenvSwitch, OVSDB,* dan lainnya. Selain itu *Ryu* mendukung penggunaan GRE dan VLAN di dalam paket *library Ryu* [2].

RYU adalah kerangka kerja SDN berbasis komponen, dikembangkan oleh Nippon Telegraph and Telephone (NTT). RYU adalah pengontrol SDN sepenuhnya diimplementasikan dengan Python. Nama RYU berasal dari kata dalam bahasa Jepang yang berarti “mengalir,” yang mana adalah naga Jepang,salah satu dewa air. Ia mengatur kontrol "aliran" untuk mengaktifkan kecerdasan jaringan. Kontroler RYU menyediakan komponen perangkat lunak dengan aplikasi yang terdefinisi dengan baik antarmuka program (API) yang memudahkan pengembang membuat manajemen jaringan baru dan aplikasi kontrol [4].

RYU menyediakan komponen perangkat lunak dengan API terdefinisi dengan baik yang memudahkan pengembang untuk membuat manajemenjaringan baru dan aplikasi kontrol. RYU mendukung berbagai protokol untuk mengelola perangkat jaringan, seperti OpenFlow, Netconf, OF-config, dll. Tentang OpenFlow, RYU mendukung sepenuhnya 1.0, 1.2, 1.3, 1.4, 1.5, dan Nicira Extensions. Semua

kode tersedia secara bebas di bawah lisensi Apache 2.0. RYU berbasis bahasa python dan bersifat Open source [4]

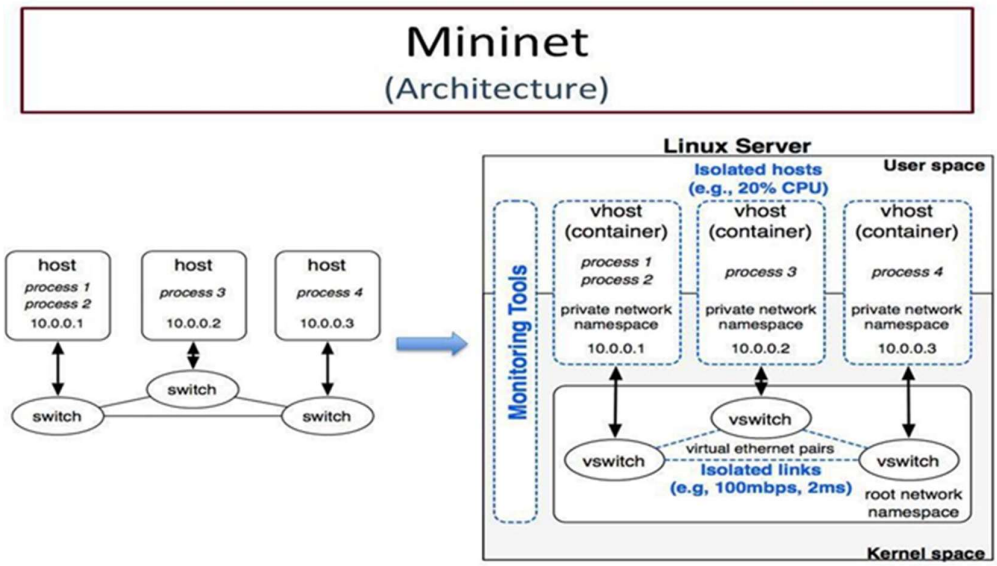
2.2.2 *Mininet*

Mininet merupakan sebuah *emulator* jaringan SDN dimana dapat menghasilkan simulasi jaringan yang hampir sama seperti jaringan nyata. *Mininet* dapat diaplikasikan untuk menjalankan baik itu jaringan berskala kecil maupun besar [3]. Salah satu kelebihan mininet yaitu hasil dari simulasi yang dijalankan dapat dengan mudah diterapkan pada jaringan nyata. *Mininet* mendukung semua kebutuhan jaringan SDN yang dapat diterapkan seperti *basic forwarding*, *firewall, monitoring,* dan lainnya [3]. *Mininet* mempunyai kelebihan utama yaitu dengan penerapan pendekatan OS-Level virtualization memungkinkan melakukan pemrosesan jaringan dengan *range* yang luas dan dapat menangani ratusan *node* jaringan di dalam *Mininet* dan dapat menjalankan serta melakukan *debug* jaringan secara *real-time* [7].

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) [8]. 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. [4] 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 [6].

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 [4].



Gambar 2.2 Arsitektur dari Mininet. [4]

Salah satu alasan mininet sering digunakan untuk penelitian karena mininet dapat membantu dalam membuat topologi sesuai dengan kebutuhan atau keinginan perancang, banyak yang telah membuktikannya dengan topologi yang cukup kompleks, lebih besar dan topologi internet seperti yang digunakan untuk penelitian pada BGP. Fitur lain yang sangat bagus dari mininet yaitu memungkinkan untuk kostumisasi packet forwarding sepenuhnya [9]. Berikut adalah kelebihan mininet:

1. Menyediakan testbed jaringan yang sederhana dan murah untuk mengembangkan aplikasi OpenFlow.

2. Memungkinkan beberapa pengembang untuk bekerja sama secara independen pada topologi yang sama.

3. Mendukung system-level regression tests yang berulang dan mudah dikemas.

4. Memungkinkan pengujian topologi yang kompleks tanpa perlu memasang jaringan fisik.

5. Menggunakan CLI pada sebuah topologi dan Openflow untuk debugging atau melakukan tes pada jaringan luas.

6. Mendukung pembuatan topologi custom secara acak, sehingga dapat memudahkan pengguna dalam membuat topologi berdasarkan kebutuhan pengguna.

7. Menyediakan API Python yang mudah dan extensible untuk penciptaan jaringan dan eksperimen [9].

2.2.3 *Firewall*

Firewall merupakan sistem keamanan jaringan yang memungkinkan untuk mengkontrol dan memonitor trafik data yang masuk dan keluar dari suatu sistem berdasarkan *security rules* yang sudah ditentukan [5]. *Firewall* berfungsi untuk mencegah akses yang tidak di inginkan atau tidak di izinkan untuk keluar maupun masuk dari jaringan atau sistem tertentu. *Firewal*l didefinisikan sebagai sekumpulan sistem keamanan yang berada diantara jaringan dalam (*trusted network*) dan jaringan luar ataupun antara perangkat didalam jaringan yang sama [5]. Firewall digunakan untuk mengamankan jaringan dan merupakan sistem yang baik pada perangkat lunak maupun perangkat keras dengan membatasi, menyaring atau bahkan dapat menolak satu atau semua jaringan local maupun jarungan yang ada diluar. Pada umumnya firewall diterapkan dalam suaru gateway antara jaringan local dan jaringan internet, serta secara umum mengontrol dan membatasi akses ke jaringan.

2.2.4 *Redis database*

Redis (Remote Dictionary Server) merupakan salah satu *NoSQL database* yang mendukung struktur data yang lebih maju akan tetapi hanya sebatas *document-oriented* dan pengurutan data sesuai dengan waktu data yang dimasukan kedalam *database* [2]. *Redis database* merupakan salah satu *NoSQL database* yang memiliki *trading durability* (pengiriman data *update* antara server *database* dan server pemanggil *value database*) tercepat. *Redis database* mempunyai *toolkit* yang lengkap seperti urutan data ataupun *stack* untuk di blok, dan *toolkit* fungsi lainnya [2].

Redis database menggunakan suatu operasi untuk membuat dan meng-*update* data menggunakan fungsi *SET & MSET* dengan ketentuan *syntax* sebagai berikut : *SET <key> <value>* ataupun *MSET <key1> <value1> <key2> <value2>*. *Redis database* tidak hanya menyimpan data *string* melainkan juga data *numeric* dan mendukung data jenis *interger, float,* dan beberapa operasi bilangan simple seperti *increment & decrement* [2].

2.2.5 *Quality of Service (QoS)*

Quality of Service (QoS) adalah suatu cara untuk menentukan kualitas jaringan. QoS merupakan bentuk usaha guna mendefinisikan karakteristik dan sifat dari suatu jaringan. Parameter Qos yang dinilai berupa parameter seperti *delay*, *jitter* atau packetloss. Jaringan dapat melakukan penggunaan *bandwidth* yang bervariasi untuk memenuhi permintaan [1].

1. *Throughput*

Throughput merupakan kemampuan dari suatu jaringan dalam mentransmisikan data perdetik Satuannya yaitu *bit* per *second* (bps). Rumus perhitungan *throughput* seperti pada persamaan 2.1 [2]. Untuk menentukan klasifikasi nilai *throughput* yang didapat dapat berpatokan pada standarisasi TIPHON seperti yang terlihat pada tabel 2.1.

$$\text{Throughput} = \frac{Jumlah\ data\ yang\ dikirim}{Waktu\ pengiriman\ data} \quad \textbf{(2.1)}$$

Tabel 2.1 Kategori *Throughput*

Kategori	Besar *Throughput (%)*	Indeks
Sangat Bagus	<100	4
Bagus	<75	3
Sedang	<50	2
Jelek	<25	1

2. *Delay*

Delay merupakan waktu rata-rata yang dibutuhkan oleh paket untuk terkirim dari node sumber ke node tujuan. *Delay* mempengaruhi kualitas layanan (QoS) karena menunjukkan bahwa paket membutuhkan waktu untuk mencapai tujuannya dalam waktu yang lebih lama seperti yang ditunkukkan pada persamaan 2.2 [3]. Untuk menentukan klasifikasi nilai *delay* yang didapat dapat berpatokan pada standarisasi TIPHON seperti yang terlihat pada table 2.2.

$$\text{Delay rata} - \text{rata} = \frac{Total\ Delay}{Total\ paket\ yang\ diterima} \quad \textbf{(2.2)}$$

Tabel 2. 2 Kategori *Delay*

Kategori	Besar *Delay*	Indeks
Sangat Bagus	<150 ms	4
Bagus	150 ms s/d 300 ms	3
Sedang	300 ms s/d 450 ms	2

Jelek	>450ms	1

3. Jitter

Jitter merupakan variasi *delay* pada proses transmisi yang terjadi karena adanya perbedaan waktu atau interval kedatangan antar paket yang diterima oleh node tujuan. Cara mengatasi permasalahan *jitter* yaitu paket data yang datang dihimpun terlebih dahulu pada *jitter* buffer selama waktu tertentu hingga paket dapat diterima pada sisi receiver dengan urutan yang benar. Perhitungan *jitter* dapat dilihat pada persamaan 2.3 [1]. Untuk menentukan klasifikasi nilai *jitter* yang didapat dapat berpatokan pada standarisasi TIPHON seperti yang terlihat pada tabel 2.3.

$$Jitter = \frac{total\ variasi\ delay}{total\ paket\ yang\ diterima} \quad \textbf{(2.3)}$$

Tabel 2.3 Kategori *Jitter*

Kategori	*Jitter*	Indeks
Sangat Bagus	0 ms	4
Bagus	0 ms s/d 75 ms	3
Sedang	75 ms s/d 125 ms	2
Jelek	125 ms s/d 225 ms	1

4. *Packet loss*

Packetloss merupakan persentase data yang gagal dikirim dari total data yang ditransmisikan. *Packet loss* dapat terjadi karena adanya kemacetan pada saluran transmisi akibat padatnya *traffic*. Perhitungan *packet loss* dapat dilihat pada

persamaan 2.4 [1]. Untuk menentukan klasifikasi nilai *packet loss* yang didapat dapat berpatokan pada standarisasi TIPHON seperti yang terlihat pada table 2.4.

$$Packet\ loss = \frac{paket\ data\ yang\ dikirim - paket\ data\ yang\ diterima}{paket\ data\ yang\ dikirim} \text{ x } 100\% \quad \textbf{(2.4)}$$

Tabel 2.4 Kategori *packet loss*

Kategori	*Packet loss*	Indeks
Sangat Bagus	0 %	4
Bagus	3 %	3
Sedang	15 %	2
Jelek	25 %	1