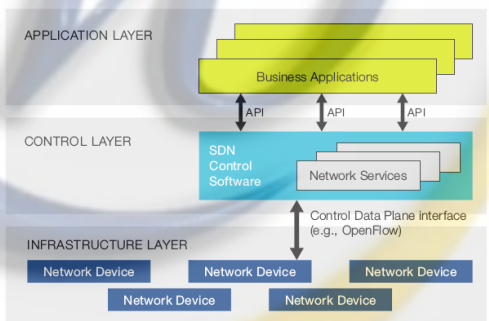
BAB II

LANDASAN TEORI

2.1 Software Defined Networking (SDN)

Software Defined Networking (SDN) merupakan sebuah pendekatan arsitektur jaringan komputer yang memisahkan *control plane* dari sebuah perangkat jaringan komputer (*switch* atau *router*) dengan *data plane* perangkat jaringan komputer tersebut. Pemisahan *data-plane* dan *control-plane* ini memungkinkan untuk memprogram perangkat tersebut sesuai dengan yang diinginkan secara terpusat (SDN*Controller*), sehingga hal ini memungkinkan untuk mengontrol, memonitor, dan mengatur sebuah jaringan komputer dari sebuah titik (*node*) terpusat tersebut [2]. Arsitektur *Logical-SDN* dengan control terpusat ditunjukan pada Gambar. 2.1.



Gambar 2.1 *Software-Defined Networking Architecture* [1]

Pada teknologi jaringan konvensional, sistem pembuat keputusan kemana arus data dikirimkan dibuat menyatu dengan perangkat kerasnya. Namun di dalam teknologi SDN memiliki dua karakteristik. Pertama, SDN memisahkan antara *control plane* dari *data plane*. Kedua, SDN menggabungkan *control plane* setiap perangkat menjadi sebuah kontroler yang berbasiskan *programmeable software*. Sehinga sebuah kontroler tersebut dapat mengontrol banyak perangkat dalam sebuah *data plane*. Di dalam SDN sebuah jaringan tersentralisasi dalam sebuah kontroler yang berbasiskan

software yang dapat memelihara jaringan secara keseluruhan, Sehingga dapat mempermudah dalam mendesain dan mengoperasikan jaringan karena hanya melalui sebuah *logical point* [3].

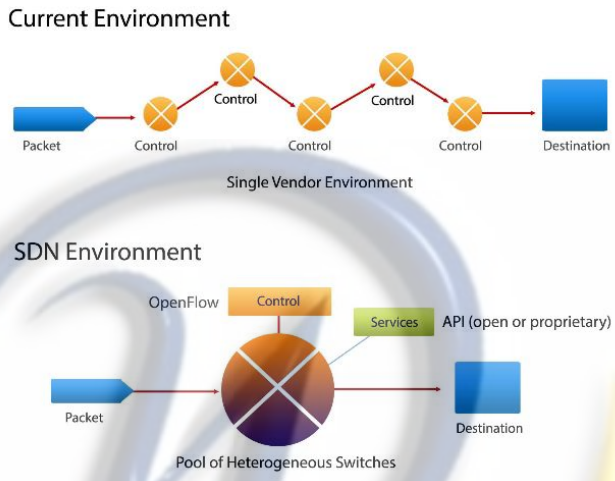
SDN adalah mengubah cara untuk merancang dan mengelola jaringan. SDN memiliki kontrol terpisah dan logika data, sehingga dapat mudah untuk mengelola arsitektur jaringan dalam skenario berdasarkan *virtual*. Keputusan kontrol dinamis untuk menangani lalu lintas memungkinkan *switch* tanpa menimbulkan biaya *overhead* pemeliharaan. SDN merupakan arsitektur futuristik yang kuat, mudah untuk mengelola, lebih murah dan fleksibel [9].

SDN adalah sebuah konsep pendekatan jaringan komputer dimana sistem pengkontrol dari arus data (*control-plane*) secara fisik dipisahkan dari perangkat kerasnya (*data-plane*). Umumnya, sistem pembuat keputusan kemana arus data akan dikirimkan dibuat menyatu dengan perangkat kerasnya. *Software-defined networking* memungkinkan penggunanya untuk menulis aplikasi untuk mengelola layanan jaringan termasuk *load-balancing, routing, acces control, multicast,* dan tugas-tugas rekayasa lalu lintas lainnya [10].

Software Defined Networking (SDN) adalah arsitektur jaringan yang muncul di mana kontrol jaringan dipisahkan dari forwarding dan langsung diprogram. kontrol migrasi ini, sebelumnya terikat erat di perangkat jaringan individu, ke dalam perangkat komputasi yang diakses dan memungkinkan infrastruktur dasar dicabut untuk aplikasi dan layanan jaringan, yang dapat mengobati jaringan sebagai entitas *logical* atau virtual. [1].

Jaringan intelijen (*logical*) terpusat di *softwarebased* SDN *controller*, yang mempertahankan pandangan global jaringan. Akibatnya, jaringan muncul untuk aplikasi dan kebijakan mesin sebagai, *logical switch* tunggal. Dengan SDN, perusahaan dan operator mendapatkan kontrol vendor-

independen di seluruh jaringan dari titik *logical* tunggal, yang sangat menyederhanakan desain jaringan dan operasi. SDN juga sangat menyederhanakan perangkat jaringan sendiri, karena mereka tidak lagi perlu memahami dan memproses ribuan standar protokol tetapi hanya menerima instruksi dari pengendali SDN [1].



Gambar 2.2 *Comparison between Conventional Network and* SDN *Network* [4]

2.2 Keunggulan SDN

Keunggulan SDN antara lain [11]:

1. Directly programmable: Dapat diprogram secara langsung karena control plane terpisah dari forwarding plane (data plane).
2. Agile: Pemisahan antara control plane dengan forwarding plane menyebabkan administrator jaringan dapat secara dinamis menyesuaikan flow traffic network sesuai kebutuhan organisasi/institusi.
3. Centrally managed: Network intelligence berada terpusat di SDN controller, dan mengelola satu gambaran umum (global view) yang utuh mengenai jaringan, sehingga tampak bagi pengguna hanya terdiri atas satu logical switch saja.

4. Programmatically configured: SDN memungkinkan network administrator untuk mengelola sumberdaya jaringan dengan sangat cepat, melalui program SDN yang terotomatisasi dan dapat ditulis sendiri karena program tersebut tidak bergantung pada proprietary software atau device.
5. Open standards-based and vendor-neutral: Format instruksi controller plan didefinisikan menggunakan open standard dan tidak tergantung pada vendor-specific device atau protokol tertentu, sehingga dapat diimplementasikan pada jaringan apapun tanpa terikat oleh vendor, dan pada akhirnya akan mempermudah inovasi di bidang jaringan.

Dalam [12] dijelaskan bahwa salah satu kelebihan SDN bila dibandingkan teknologi yang sudah ada adalah sifatnya yang realistic namun light-weight. Simulator jaringan yang telah ada sebelumnya (misalnya ns2 dan Opnet) tidak menyediakan lingkungan yang realistic, artinya kode yang dijalankan pada simulator tersebut jauh berbeda dari kode yang dijalankan di real network. Sementara itu, simulasi jaringan menggunakan VM (virtual machine) dapat memberikan lingkungan yang realistic, namun VM bersifat heavy-weight sehingga cukup berat

2.3 QoS (Quality of Service)

QoS adalah teknik untuk mengelola *bandwidth, delay, jitter,* dan *paket loss* untuk aliran dalam jaringan. Tujuan dari mekanisme QoS adalah mempengaruhi setidaknya satu diantara empat parameter dasar QoS yang telah ditentukan [13].

QoS didesain untuk membantu *end user (client)* menjadi lebih produktif dengan memastikan bahwa user mendapatkan performansi yang handal dari aplikasi-aplikasi berbasis jaringan. QoS mengacu pada kemampuan jaringan untuk menyediakan layanan yang lebih baik pada trafik jaringan tertentu melalui teknologi yang berbeda-beda. QoS merupakan suatu

tantangan yang besar dalam jaringan berbasis IP dan internet secara keseluruhan. Tujuan dari QoS adalah untuk memenuhi kebutuhan-kebutuhan layanan yang berbeda, yang menggunakan infrastruktur yang sama. QoS menawarkan kemampuan untuk mendefinisikan atribut-atribut layanan yang disediakan, baik secara kualitatif maupun kuantitatif [13]. Fungsi-fungsi QoS dijelaskan sebagai berikut [13]:

1. Pengkelasan paket untuk menyediakan pelayanan yang berbeda-beda untuk kelas paket yang berbeda-beda.
2. Penanganan kongesti untuk memenuhi dan menangani kebutuhan layanan yang berbeda- beda.
3. Pengendalian lalu lintas paket untuk membatasi dan mengendalikan pengiriman paket- paket data.
4. Pensinyalan untuk mengendalikan fungsi-fungsi perangkat yang mendukung komunikasi di dalam jaringan IP.

Tabel 2II.1 Indeks Parameter QoS

Nilai	Persentase (%)	Indeks
3,8 - 4	95 - 100	Sangat Bagus
3 - 3,79	75 - 94,75	Bagus
2 - 2,99	50 - 74,75	Sedang
1 - 1,99	25 - 49,75	Buruk

(Sumber: TIPHON)

Ada beberapa parameter QoS yang dapat digunakan untuk mengukur kinerja jaringan antara lain :

2.3.1 Throughput

Throughput adalah jumlah data jaringan yang mengirim atau menerima data, atau jumlah data yang diproses dalam satu ruang waktu

yang ditentukan. Ini memiliki sebagai unit dasar dari tindakan bit per detik (bit /s atau bps) [14].

Throughput adalah laju data aktual per satuan waktu. *Throughput* bisa disebut sebagai *Bandwidth* dalam kondisi yang sebenarnya. *Bandwidth* lebih bersifat tetap, sementara *Throughput* sifatnya dinamis tergantung trafik yang sedang terjadi. *Throughput* mempunyai satuan Bps (Bits per *second*) [13]. Kategori *Throughput* diperlihatkan di Tabel 2.2.

Tabel 2II.2 Kategori *Throughput*

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

(Sumber: TIPHON)

Untuk mengukur nilai Throughput digunakan Persamaan (1) [15]:

$$Throughput = \frac{\text{ukuran data yang diterima}}{\text{waktu pengiriman data}} \quad \text{.....(1)}$$

2.3.2 Delay

Delay adalah waktu yang dibutuhkan paket untuk mencapai tujuan, karena adanya antrian, atau mengambil rute yang lain untuk menghindari kemacetan yang di perlihatkan pada Tabel 2.3 [15].

Delay adalah waktu antara ketika perangkat menerima sebuah *frame* dan ketika *frame* yang diteruskan keluar dari port tujuan, *serialization delay* adalah waktu yang dibutuhkan untuk benar-benar mengirimkan paket atau *frame* , dan *end to-end* [14].

Tabel 2II.3 Kategori dari *Delay (Latency)*

Kategori Delay	Besar delay (ms)	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 ms s/d 300	3
Sedang	300 ms s/d 450	2
Jelek	> 450 ms	1

(Sumber: TIPHON)

Delay dapat dicari dengan membagi antara panjang paket (L, *packet length* (bit/s)) dibagi dengan *link bandwith* (R, *link bandwith* (bit/s)). Pada Tabel 4 diperlihatkan kategori dari latensi dan besar *delay* [15].

Untuk mengukur nilai *Delay* digunakan Persamaan (2, 3) [15]:

$$\text{Delay} = \frac{\text{packet length}}{\text{link bandwidth}} \quad \text{.....(2)}$$

Atau

$$\text{Rata-rata Delay} = \frac{Total\ Delay}{Total\ Paket\ Yang\ Di\ Terima} \quad \text{.....(3)}$$

2.3.3 Packet loss

Packet Loss merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan. Pada Tabel 2.4 ditunjukkan nilai indeks dan kategori *Packet Loss* [15].

Tabel 2II.4 Kategori *Packet Loss*

Kategori Packet Loss	Packet Loss (%)	Indeks
Sangat Bagus	0% - 2%	4
Bagus	3% - 14%	3
Sedang	15% - 24%	2
Jelek	>25 %	1

(Sumber: TIPHON)

Untuk mengukur nilai *Packet Loss* digunakan Persamaan (4) [15]:

$$\text{Packet loss} = \frac{Y}{A} \text{ X } 100\% \quad \text{.....(4)}$$

Keterangan:

Y = Paket data dikirim – paket data diterima

A = Paket data dikirim

2.3.4 Jitter

Jitter adalah perbedaan antara interpacket kedatangan dan keberangkatan-yaitu, variasi dalam penundaan dari satu paket yang lain [14]. *Jitter* merupakan variasi *delay* yang terjadi akibat adanya selisih waktu atau interval antar kedatangan paket di sisi penerima. Keberadaan *jitter* buffer akan menambah nilai *end-to end delay*.

Jitter diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket di akhir perjalanan *jitter* yang diperlihatkan pada Tabel 2.5 [15].

Tabel 2II.5 Kategori dari *Jitter*

Kategori Jitter	Jitter (ms)	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

(Sumber: TIPHON)

Untuk mengukur nilai Jitter digunakan Persamaan (5) dan Persamaan (6) [15] :

$$Packet\ loss = \frac{Total\ variasi\ delay}{Total\ paket\ yang\ diterima} \quad .....(5)$$

Total variasi delay = Delay – (rata – rata delay).....(6)

2.4 Open vSwtch (OVS)

Open vSwitch (OVS) adalah *multilayer virtual swicth* dilisensikan di bawah open source lisensi Apache 2.0. Hal ini dirancang untuk memungkinkan otomatisasi jaringan besar melalui ekstensi program, sementara masih mendukung antarmuka manajemen standar dan protokol. Selain itu, ia dirancang untuk mendukung distribusi di beberapa server fisik mirip dengan VMware's vNetwork atau vswitch or Cisco's Nexus 1000V. [5].

OVS dirancang untuk menjadi fleksibel, portabel dan berada dalam *hypervisor* atau manajemen domain, untuk menyediakan konektivitas antara mesin virtual dan *interface* fisik.

Hal ini dapat beroperasi sebagai dasar L2 *switch* dalam konfigurasi *standalone*, mendukung VLAN, SPAN, RSPAN, ACL, kebijakan QoS, *port bonding*, *trunking*, GRE dan IPsec, *tunneling*, dan per-VM *traffic policing*. Arus visibilitas dengan NetFlow dan sFlow juga disediakan. Untuk mendukung integrasi ke lingkungan virtual, OVS mengeluarkan *interface* untuk memanipulasi *forwarding state* dan mengelola konfigurasi pada saat *run-time*, yang memungkinkan spesifikasi tentang bagaimana paket ditangani berdasarkan L2 mereka, L3, dan *header* L4 [8].

2.5 White Box Switch

White Box Switch adalah komponen dari SDN terletak di *Southbound* (infrastruktur) layer dan mengatur aliran data sesuai dengan perintah dari SDN *controller* (lapisan kontrol). Tidak seperti saklar konvensional, *White Box Switch* hanya sebuah saklar kosong ("blank" switch) yang tidak memiliki otak seperti saklar konvensional. Semua aktivitas kotak putih telah diatur oleh SDN kontroler.

White Box Switch mengandalkan sistem operasi (OS). sistem operasi umum untuk memakai *White Box Switch* adalah dengan OS Linux karena Linux OS yang terbuka dan gratis yang tersedia dan membantu menyesuaikan perangkat untuk kebutuhan [5]. Untuk menjadi sebuah *switch*, *White Box Switch* membutuhkan *software* virtual *swicth* yang dipasang di OS. Salah satu yang populer adalah Open vSwitch (OVS).

2.6 Mininet

Mininet adalah emulator jaringan *software-defined networking* yang dapat mensimulasikan kinerja antara *end-host, switch, router, controller*, dan *link* dalam sebuah *kernel* Linux. *Mininet* merupakan sebuah sistem virtualisasi yang dapat menggambarkan jaringan yang besar dengan hanya menggunakan sebuah *laptop*. *Mininet* bersifat *open source*, sehingga proyek yang telah dilakukan berupa *source code, scripts*, dan dokumentasi yang dapat dikembangkan oleh siapapun [16].