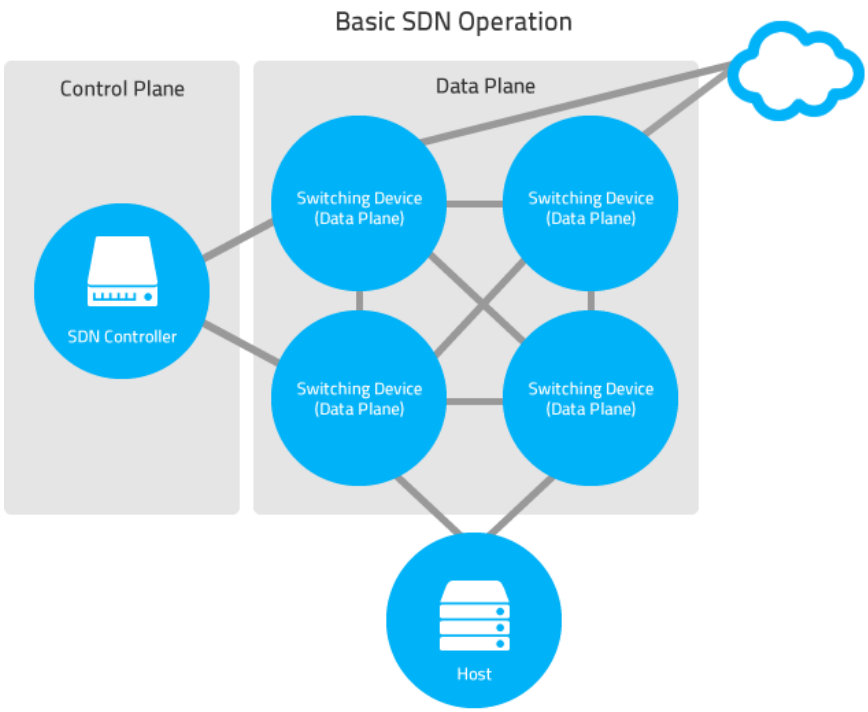
BAB II
LANDASAN TEORI

2.1 Software Define Network

Software Define Network merupakan sebuah konsep baru di jaringan khususnya dalam bidang infrastruktur. Paradigma baru dalam mendesain, mengelola, dan mengimplementasikan jaringan untuk kebutuhan yang lebih kompleks untuk memisahkan dari *control plane* dan *data plane*. Sehingga memudahkan untuk akses dan dikelola, karena integrasi jaringanya secara terpusat (Fahri et al., 2018). Menggunakan *Software Define Network* (SDN) berbasis *openflow* ini, maka *software* dapat melihat secara langsung pada *topologi* jaringan ini menggunakan *remote control* sehingga dapat di akses dari jarak jauh. *Openflow* dapat diakses secara langsung pada *data plane* tanpa melakukan konfigurasi di perangkat jaringan baik itu fisik ataupun *virtual*. *Software define network* adalah *arsitektur* baru yang *dinamis*, dapat dikelola, aman, hemat biaya, mudah beradaptasi, ideal untuk menggunakan *bandwith* tinggi, dan pengoptimalan sumber daya jaringan dengan cepat melalui program otomatisasi yang dinamis serta tidak bergantung pada pemilik perangkat lunak.



Gambar 2.1 Topologi Software Define Network (Kidung faza, 2017)

2.2 Mininet

Mininet merupakan sebuah emulator untuk mensimulasikan pada jaringan untuk sebuah penelitian, riset, pengerjaan, dan pengembangan khususnya di bidang *Software Define Network* dan *Openflow* (Aprilianingsih et al., 2017). Mininet ini dapat membuat prototype atau rancangan jaringan berskala besar dengan cepat pada sumber daya yang terbatas (seperti pada *virtual machine*). Mininet ini diciptakan untuk tujuan riset di bidang *Software Define Network* (SDN) dan *Openflow*.

2.3 Control Plane

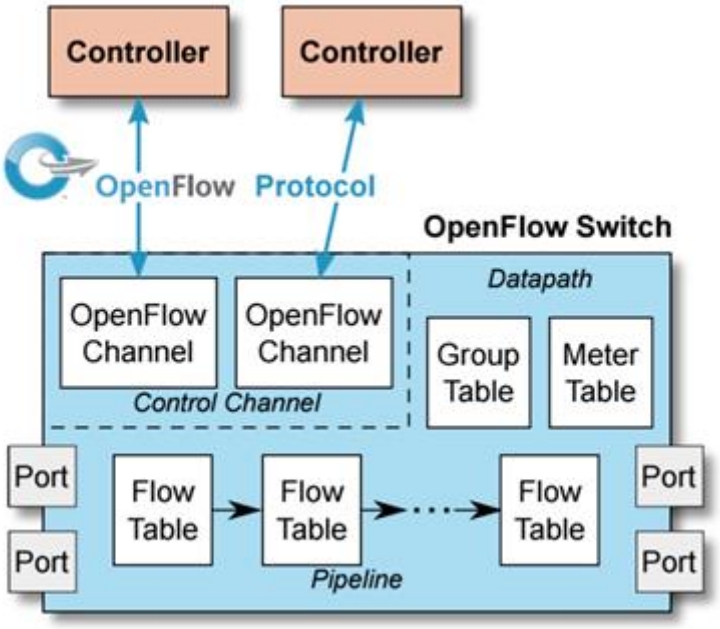
Control plane merupakan bagian yang mengatur atau mengontrol dari sebuah paket untuk diteruskan. *Control plane* ini adalah otak untuk menjalankan fungsi dari suatu konfigurasi, manajemen lalu lintas, keamanan, jalur *routing*, analisis, dan lainya (cloudflare, 2021).

2.4 Data Plane

Komponen yang utama selain *control plane* yaitu *data plane*. *Data plane* merupakan komponen yang berfungsi untuk meneruskan paket, menguraikan *header* paket, *QOS, access list* dan mengenkapsulasi paket (Hidayat & Perdana, 2020).

2.5 Openflow

Openflow merupakan implementasi dari *Software Define Network* yang standarisasinya mendefinisikan dari antarmuka sebuah sistem komunikasi antara *control plane* dan *data plane* pada arsitektur jaringan. Desain ini awalnya untuk membuat eksperimen dalam gedung yang bisa dijadikan penelitian, namun dilihat untuk kedepanya dapat dijadikan potensi pada fungsi *protokol layer* 2 dan *layer* 3 yang selama ini terintegrasi pada perangkat *switch* ataupun *router* (Halomoan et al., 2017). *Openflow* ini merupakan standart pada *Software Define Network* (SDN) yang terdiri dari dua entitas yaitu *switch openflow* yang dapat mengimplementasikan *forwarding* data, dan *controller* untuk mengimplementasikan *control plane*. *OpenFlow* menggunakan konsep aliran untuk mengidentifikasi aliran jaringan berdasarkan aturan pencocokan yang telah ditentukan sebelumnya , dapat diprogram secara statis atau dinamis melalui perangkat lunak kontrol Sofware Define Network (SDN).



Gambar 2.2 *Openflow* (Transmissia Ratu Hapsari, 2018)

2.6 Operasi untuk Pengujian

Pengujian ini dilakukan untuk mengetahui dari perbedaan kedua *controller*, sehingga dilakukan pengukuran komponen dari *resource utilization, delay, jitter, throughput*.

- ***Resource Utilization***

  Resource utilization merupakan pemanfaatan presentasi untuk mengukur dari sumber daya oleh sistem untuk mengetahui efektifitas memori dan *CPU* selama *controller* digunakan (Safida Reynita Sari, Dr.Ir Rendy Munadi, M.T., Danu Dwi Sanjoyo, S.T., 2019).

- ***Delay***

  Delay adalah waktu dari jeda paket yang disebabkan oleh transmisi atau jarak yang dikirimkan dari suatu *node* ke *node* lainya. Delay dalam suatu *node* disababkan karena ada pengaruh dari antrian yang panjang untuk menghindari dari kemacetan rute lain. Tabel 2.1 indeks karakteristik delay.

Tabel 2. 1 Indeks Karakteristik *Delay*

Kategori *Delay*	**Besar *Delay***	**Index**
Sangat Baik	≤ 150 ms	4
Baik	150 - 300 ms	3
Cukup	300 - 450 ms	2
Buruk	≥ 450 ms	1

Sumber : *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON); *General aspects of Quality of Service* (QoS) (ETSI, 2007).

Persamaan perhitungan *delay* dilihat pada rumus 1.

$$\textit{Delay}\ \text{rata-rata} = \frac{\text{Total } \textit{delay}}{\text{Total paket yang diterima}} \quad (1)$$

- ***Packet Loss***

Packet Loss merupakan suatu parameter banyaknya paket yang hilang karena terjadi *collision* atau *congestion* selama transmisi pada tujuan. Tabel 2.2 menunjukan index karakteristik *packet loss.*

Tabel 2. 2 Indeks karakteristik *packet loss*

Kategori *Packet Loss*	***Packet loss***	**Index**
Sangat Baik	0 - 3%	4
Baik	3 - 15%	3
Cukup	15 - 25%	2
Buruk	> 25%	1

Sumber : *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON); *General aspects of Quality of Service* (QoS) (ETSI, 2007).

Persamaan perhitungan *packet loss* dilihat pada rumus 2.

$$\textit{Packet loss} = \frac{\text{Paket data dikirim – packet data diterima}}{\text{Paket data dikirim}} \quad (2)$$

- ***Throughput***

Throuhput merupakan *bandwith* aktual atau *bandwith* sebenarnya yang terukur pada suatu waktu tertentu dalam jaringan menggunakan rute internet yang spesifik ketika men*download* sebuah file. Tabel 2.3 menunujukan indeks karakteristik *throughput.*

Tabel 2. 3 Indeks karakteristik throughput

Kategori *Throughput*	***Throughput***	**Index**
Sangat Baik	≥ 2,1 Mbps	4
Baik	1200 Kbps – 2,1 Mbps	3
Cukup	700 – 1200 Kbps	2
Kurang Baik	338 – 700 Kbps	1
Buruk	0 – 338 Kbps	0

Persamaan perhitungan *throughput* dilihat pada rumus 3.

$$\textit{Throughput} = \frac{\text{Paket data yang diterima}}{\textit{Bandwith} \text{ yang tersedia}} \quad (3)$$

2.7 Aplikasi untuk pengujian

a. Iperf

Pengujian untuk mengukur perfoma dari contoller seperti delay, jitter, dan throughput adalah iperf (Muzawi, 2016). Iperf merupakan salah satu tool yang dapat mengukur performa dari link baik dari sisi TCP maupun UDP.

Iperf adalah *software* yang bersifat open source yang dapat dijalankan dalam sistem operasi linux, unix dan windows.

b. Phoronix Test Suite

Phoronix test suite adalah pengukuran komperhensif yang menyediakan rangkaian extensible untuk pengujian analisis performa virtual mechine yang dikenal sebagai benchmarking dan testing tool (Sudha, 2019). Aplikasi ini bersifat open source dan dapat digunakan dalam sistem operasi seperti linux, windows, dan apple.

2.8 Ubuntu Desktop

Ubuntu Desktop merupakan sebuah sistem operasi khusus untuk perangkat desktop. Sistem operasi ini berisi aplikasi yang cocok untuk penggunaan sehari-hari. Ada beberapa set perangkat lunak, perangkat lunak multimedia, dan browser web yang mendukung produktivitas kantor. Adapun beberapa kelebihan menggunakan sistem operasi ubuntu diantaranya :

a. Gratis dan bersifat open source
b. Mudah digunakan
c. Aman digunakan
d. Support yang kuat
e. Mudah dikustomisasi

Sejak diluncurkan, sistem operasi ini telah menjadi favorit banyak orang karena mudah dipasang dan digunakan. Lingkungan desktop default Ubuntu disebut Unity, yang merupakan lingkungan desktop dengan alat pencarian canggih yang dapat menemukan semua aplikasi. Lingkungan desktop ini juga terintegrasi dengan aplikasi lain, seperti pemutar audio, pemutar video, dan media sosial (dewaweb, 2020).

2.9 Wireshark

Wireshark adalah perangkat lunak analisis lalu lintas Jaringan komputer, memiliki fungsi yang sangat berguna profesional jaringan, administrator jaringan, peneliti, pengembang perangkat lunak jaringan. Wireshark telah menjadi

penganalisis protokol jaringan ini sudah terkenal dan telah menjadi standar di berbagai industri, dan apakah proyek tindak lanjut dimulai pada tahun 1998. Wireshark digunakan oleh para profesional jaringan untuk analisis, pemecahan masalah, pengembangan perangkat lunak dan protokol juga digunakan untuk tujuan pendidikan (Hanipah & Dhika, 2020).

2.10 Virtual Machine

Virtual machine merupakan sebuah sistem virtual yang pembuatan awalnya berbentuk fisik menjadi berbentuk perangkat lunak atau virtual. Seperti sistem operasi, server dan perangkat penyimpanan, dan peralatan jaringan. Perangkat lunak yang digunakan untuk virtualisasi disebut hypervisor. Perangkat lunak ini digunakan untuk membuat dan mengelola mesin virtual, sehingga dapat digunakan pada perangkat keras secara bersamaan dengan operating sistem yang asli (Kasus et al., 2019).

2.11 State Of The Art

Kajian ini, peneliti membuat *State Of The Art* yang bersumber dari jurnal, website, buku yang memiliki keterkaitan dengan *Software Define Network*. Tujuan dibuatkannya ini untuk mengetahui keterbaruan, masalah, dan dapat dijadikan referensi untuk peneliti.

Berikut *State Of the Art* yang peneliti buat sebagai berikut :

Tabel 2. 4 State of the art

No	Nama Pengarang	Judul	Masalah	*Controller*	Hasil
1.	Irhaz Muhajir, Denar Regata Akbi, Fauzi Dwi Setiawan Sumadi. (2020)	Analisis Komparasi Penerapan *Virtual Local Area Network* Pada *Software Defined Network*	Skalabilitas tinggi sangat dibutuhkan prototipe teknologi jaringan untuk dapat bekerja secara terpusat	*OpenDayLight, Floodlight, ONOS, POX* dan *RYU*	Hasil skenario pengujian, didukung dengan *high performance* dari *modularitas controller* berbasis *java* seperti *ONOS* dan

					OpenDayLight. ONOS menjamin kestabilan dalam hal penanganan aliran *flow/sec* yang cukup besar dibandingkan dengan *controller* yang di program menggunakan *python*. Sedangkan *OpenDayLight* memiliki waktu penundaan yang cukup singkat pada saat pengujian dilakukan. *OpenDayLight (ODL)* menunjukkan bahwa persentase pengiriman yang stabil dengan nilai diatas 99,90% untuk semua skenario yang diuji. Meski dalam hal waktu respon kontroler terhadap paket yang masuk *RYU* adalah yang paling stabil.
2.	Abhimata Zuhra Pramudita, I Made Suartana. (2020)	Perbandingan Performa *Controller OpenDayLight* dan RYU pada Arsitektur	Pemilihan controller yang bagus bukan hanya dalam pengujian delay, throughput, dan packet loss saja.	*Opendaylight* dan *RYU*	Hasil dari peniliti ini menyatakan bahwa *controller* Ryu memiliki performa lebih baik dari pad *controller OpenDayLight* dalam parameter *throughtput*,

		Software Defined Network			*delay*, dan *packet loss*. Pengujian Resource Utilization, controller OpenDaylight memiliki hasil performa yang lebih baik dari controller Ryu dilihat dari event per second yang dihasilkan oleh performa CPU dan Memory
3.	A. Wisnu Priya dan N. Radhika. (2019)	*Performance comparison of* SDN *OpenFlow controller*s	Teknologi jaringan tradisional dapat menambah beban seorangan administrator jaringan.	*NOX, POX, RYU,* dan *FloodLight*	Hasil penilitian ini menunjukan bahwa *delay* dan *throuhgput controller floodlight* lebih baik dibandingkan dengan *controller* lainya di semua kondisi.
4	Sizka L. Hanifa dan Rikie Kartadie. (2018)	Uji Performa Kontroler *Software-Define Network Floodlight* Vs *ONOS*	Perlu adanya analisis untuk melihat seberapa baiknya performa dari sebuah kontroler tersebut mengingat pentingnya fungsi dari kontroler itu sendiri.	*Floodlight* dan ONOS	Skenario hasil dari pengujian performa kedua *controller* ini menunjukan bahwa memberikan nilai *latency* yang baik pada jumlah *switch* dibawah 60 *switch*. Namun dalam performa dari segi *throughput* dan *latency Floodlight* memberikan respon yang lebih baik dibandingkan dengan kontroler ONOS.

5.	Xing Ye, Guozhen heng , Xingguo Luo. (2017)	*Maximizing* SDN *control resource utilization via switch migration*	Ketidakseimbangan beban bidang kontrol terdistribusi yang secara statis dipetakan ke bidang data	*Distributed Hopping Algorithm* (DHA), *Beacon*	Hasil dari pengujian ini membuktikan bahwa menerapkan algoritma DHA dapat memperpanjang *beacon controller* dan *protokol openflow* pada eksperimen numerik kinerja algoritma.
6.	Mohamed I. Hamed , Basem M.ElHalawany, Mostafa M. Fouda, dan Adly S. Tag Eldien. (2017)	*A Novel Approach for Resource Utilization and Management in* SDN	Penerapan server banyak yang mengalami kemacetan jaringan dan kelebihan beban.	*RYU, Load balancing Round Robin* statis	Hasilnya menunjukkan bahwa skema yang diusulkan mencapai keandalan dan pemanfaatan sumber daya yang lebih tinggi dari pada algoritma *load balancing Round-robin* dan berbasis acak. Selain itu, waktu respons yang lebih rendah serta tingkat transaksi dan *throughput* yang lebih tinggi. Skema yang diusulkan menunjukkan skalabilitas lebih dan karakteristik biaya rendah.
7.	Egi Taufik Hidayah	Analisis Perbandingan	Banyaknya penelitian yang	ONOS dan RYU	Hasil pengujian performa Controller RYU, POX, Opendaylight dan ONOS

		Performa Resource Utilization Controller Ryu, Pox, Opendaylight Dan Onos Pada Software Define Network	membandingkan performa *controller* namun jarang membandingkan dari segi performa *resource utillization.*		telah berhasil dilakukan dan menghasilkan performa yang berbeda, dalam hal ini controller RYU lebih baik dalam parameter throughput dengan nilai rata-rata paling tinggi yaitu 1164,8 Mbps dibanding dengan Controller lainnya. Pengujian dalam paramater packet loss menghasilkan performa yang berbeda dari setiap Controller yang di uji, dalam hal ini Controller POX lebih baik dibandingkan dengan Controller RYU, ONOS, dan Opendaylight. Karena POX menghasilkan packet loss paling rendah dengan nilai rata-rata 2,47%. Pengujian delay yang paling kecil dari semua controller yang sudah di uji yaitu RYU dengan nilai rata-rata 0,128667s. 2. Pengujian pada resource utillization, pox memiliki nilai 82.86% paling rendah dibandingkan dengan

					Onos (100%), Ryu (100%), Opendaylight (100%). Jadi Pox memiliki kinerja yang lebih baik dalam penggunaan CPU dibanding dengan Onos, Ryu, dan Opendaylight. Dalam penggunaan memory Controller Ryu mendapatkan nilai rata-rata 24,4 %, paling kecil dibandingkan dengan nilai rata-rata penggunaan memory dari Onos (66,3 %), Opendaylight (51,1 %), Pox (24,8 %). Sehingga Controller Ryu dapat dipertimbangkan untuk digunakan pada arsitektur sistem dengan memory yang terbatas