BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Pada Penelitian ini, penulis mengkaji penelitian-penelitian sebelumnya dan dijadikan sebagai pedoman dalam pelaksaan penelitian. Berikut tabel perbandingan penelitian terdahulu dan sekarang:

Tabel 2. 1 Kajian Pustaka

No	Nama Penulis, Tahun dan Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1	Dependral Dhakal.; Bishal Pradhan.; Sunil Dhima. *Campus Network Using Software Defined Network.* International Journal Of Computer Applications. 2016 (0975-8887)	Menggunakan jaringan Software Defined Network	Menggunakan software defined network untuk pengujian terhadap bandwidth dan Jitter	Menggunakan software defined network untuk melakukan load balancing
2	Wildan Maulana Syahidillah. 2017. Multipath Routing Dengan Load-Balancing Pada OpenFlow Software-Defined Netwok	Menggunakan Ryu Controller dalam mengimplementasikan Load Balancing pada OpenFlow Software-Deefined Network	Multipath Routing pada software Defined Network dengan algoritme Least Traffic	Menggunakan algoritme shortest delay pada web server
3	Erine Karantha Denny Laksana. 2016. Analisis Openflow Load Balancing Web Server dengan Algoritme Least Connection pada Software Defined Network	Load Balancing pada Web Server dan diterapkan pada Arsitektur Software Defined Network dan Openflow	Menggunakan algoritme least connection pada web server	Menggunakan algoritme shortest delay

2.2 Dasar Teori

Pada sub bab ini, dijabarkan berbagai teori yang mendukung penelitian *Analisis dan Implementasi Load Balancing pada Web Server dengan Algoritme Shortest Delay pada Software Defined Network*.

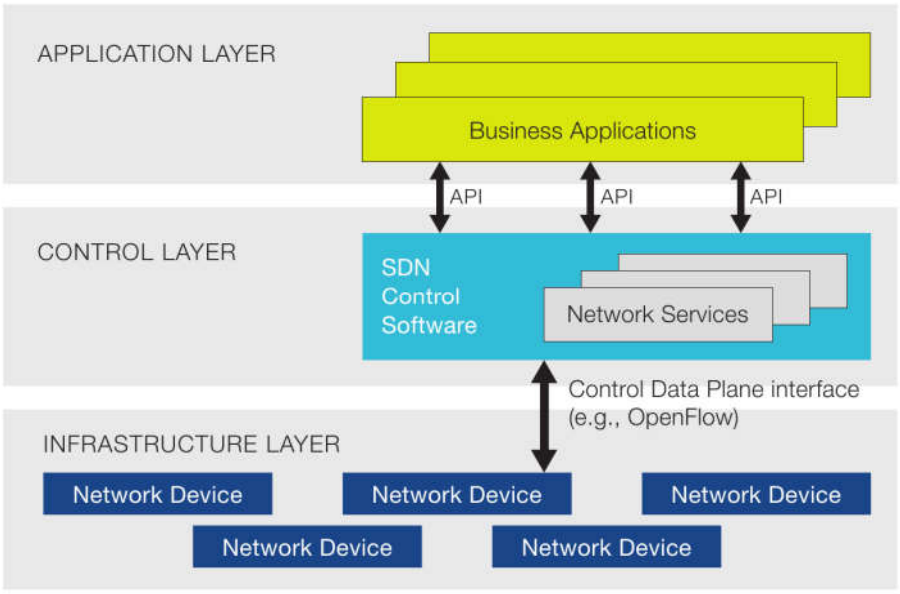
2.2.1 Software Defined Network

Software Defined Network merupakan konsep/paradigma baru dalam menganalisis dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan di bidang jaringan yang semakin lama semakin komplek. SDN menyediakan konsep arsitektur yang membuat jaringan mudah dikelola, dinamis dan hemat biaya, yaitu mengontrol jaringan untuk dapat di program secara langsung dan infrastrukturnya dapat diabstraksi dengan berbagai layanan jaringan.

Konsep itu dilakukan dengan pemisahan eksplisit antara *control plane* dan *data plane. Control plane*, yaitu perangkat jaringan yang mengatur suatu paket di kirim pada suatu jaringan dan *data plane,* yaitu perangkat yang digunakan untuk meneruskan pengiriman paket berdasarkan informasi yang diberikan *control plane*. Dengan SDN, jaringan yang ada dikelola secara tersentralisasi sehingga pengaturan dan otomasi yang diberikan seragam pada layanan jaringan memungkinkan lebih efisien sesuai keinginan yang di kehendaki (Rohmat Tulloh, Ridha Muldina Negara, & Arif Nur Hidayat, 2015).

Beberapa aspek penting dari SDN (Aris Cahyadi Risdianto, Muhammad Arif, & Eueung Mulyana, n.d.) :

1. Adanya pemisahan secara fisik/eksplisit antara forwading/data-plane dan control-plane.
2. Antarmuka standard (vendor-agnosic) untuk pemrograman perangkat jaringan.
3. Control-plane yang terpusat (secara logika) atau adanya sistem operasi jaringan yang mampu membnetuk peta logika (logical map) dari seluruh jaringan dan kemudian memrepresentasikannya melalui (sejenis) API (Application Programming Interface).



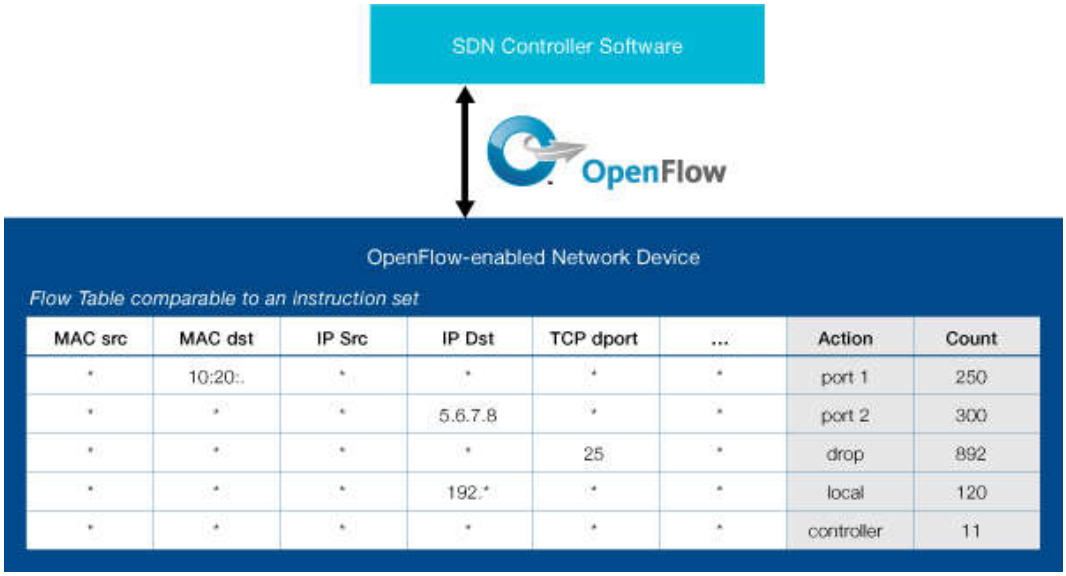
Gambar 2. 1 Arsitektur dari Software Defined Network

Sumber : (E. Bayshore Road & Palo Alto, 2012)

Pada gambar 2.1 dapat dilihat bahwa control layer memegang peran mengendalikan sistem jaringan. Setiap perangkat dapat beroperasi sesuai perintah Controllernya. Cara komunikasi perangkat dan Controllernya menggunakan suatu protocol, yaitu *openflow.* *Openflow* bisa dikatakan sebuah control unit processor pada computer (E. Bayshore Road & Palo Alto, 2012).

2.2.2 Openflow

Openflow merupakan protocol komunikasi yang di definisikan antara control dan forwarding pada arsitektur *software defined network* (E. Bayshore Road & Palo Alto, 2012). *Openflow* memungkinkan akses langsung pada perangkat jaringan, seperti *router dan switch*. Controller pada *Openflow* dapat memanipulasi data seperti menambah, menghapus dan memperogram table *flow* dari berbagai router dan switch yang berbeda. Paket *openflow* berisi paket tentang source MAC, destination MAC, source IP, destination IP dan TCP port yang digunakan untuk mengarahkan dan meneruskan paket. Berikut gambar 2.2 tabel dari openflow pada software defined network.



Gambar 2. 2 Openflow Software Defined Network

Sumber: (E. Bayshore Road & Palo Alto, 2012)

Pada gambar 2.2 di atas dapat di jelaskan bahwa mekanisme protocol *openflow* adalah saat switch *openflow* menerima paket dan tidak terdapat kecocokan dengan entri *flow* maka switch *openflow* tersebut mengirimkan paket ke *Controller*. *Controller* dapat melakukan apakah paket akan diteruskan atau dibuang. Ketika paket sesuai kecocokan maka paket di forwarding ke *Controller openflow*. Kemudian Controller merespon paket yang datang tersebut berdasarkan entri flow. Sehingga komunikasi antar control dan forwarding/data plane melalui Controller bisa memberikan respon terhadap paket yang datang (Senthil Ganesha N & Ranjani S, 2015).

2.2.3 Web Server

Web Server merupakan sebuah hardware atau software yang menyediakan layanan pada client atau pengguna web dengan menggunakan protocol HTTP atau HTTPS, berkas yang terdapat dalam situs web yang digunakan oleh client menggunakan aplikasi seperti *web browser*. *Web server* dapat diartikan sebuah pusat yang berfungsi untuk pengiriman dan penerimaan data, mengatur *request dan receive* antara server dan client serta penggunaannya untuk menempatkan situs web (Nugroho, 2014).

Berdasarkan penjelasan di atas dapat di simpulkan fungsi dari *web server* sebagai penyedia layanan yang di dalamnya terdapat informasi atau data yang dibutuhkan oleh website dan berguna untuk memindahkan data yang di minta oleh user. *Web server* akan memberikan data yang di *request* oleh user dan user memberikan *response* berupa tampilan halaman website yang berbentuk halaman html.

2.2.4 Controller Ryu

Ryu merupakan *framework* jaringan berbasis komponen untuk software defined network. *Ryu* menyediakan komponen software dengan API yang berguna bagi *developer* untuk membangun aplikasi manajemen dan control jaringan (Ryu SDN Framework Community, 2014). Aplikasi *Ryu* mendukung bahasa pemrograman python dan mengirimkan pesan JSON melalui API yang tersedia. Ryu mendukung beberapa interface yaitu paling support terhadap linux, menggunakan virtualisasi dengan mininet dan open vswitch serta mendukung protokol manajemen jaringan seperti *OpenFlow, NetConf, Of-config*. Ryu mendukung *OpenFlow Controller* versi 1.0, 1.2, 1.3, 1.4, 1.5 dan Nicira Extensions. *Controller Ryu* digunakan untuk mengembangkan sebuah sistem operasi untuk SDN yang memiliki kualitas cukup tinggi untuk digunakan dalam jaringan besar (Al-Somaidai, 2014).

2.2.5 Load balancing

Load balancing merupakan metode/proses yang digunakan untuk pendistribusian beban ke beberapa server atau perangkat jaringan. Ketika banyak permintaan dari client maka server akan terbebani karena harus melakukan pelayanan apa yang diminta client.

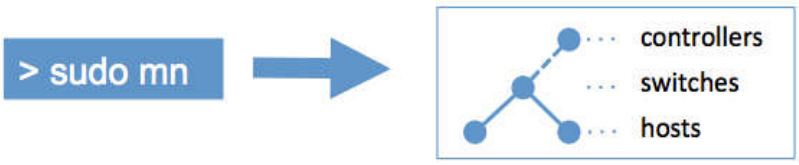
Tujuan dari load balancing untuk mengoptimalkan sumber daya, memaksimalkan throughput (response time), meminimkan waktu respon dan menghindari pembebanan berlebihan di satu sumber daya (Setyawan, 2014).

Load balancing adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal , mengoptimalkan throughput, memperkecil waktu tanggap dan menghindari overload pada salah satu jalur koneksi (Gustin Anggraeni, Sukiswo, & Ajub Ajulian Zahra, 2014).

2.2.5.1 Algoritme Shortest Delay

Algoritme ini membagi pekerjaan ke server dengan waktu tunda paling pendek. *Delay* yang akan dialami oleh pekerjaan = (Ci + 1) /Ui, jika sebuah koneksi dikirim ke server ke-i. Ci adalah jumlah koneksi pada server ke-I dan Ui adalah tingkat layanan (service *rate*) dari server ke-i. Server load balancing akan mengecek lamanya *delay* masing-masing server tiap satuan waktu dan membuat urutan prioritas *server node*. Algoritme ini termasuk algoritme dinamis karena bobot masing-masing server node bisa berubah dengan cepat. Server node dengan delay terendah akan mendapat prioritas pertama dalam mendapatkan pekerjaan. Sedangkan Server node dengan delay terbesar akan mendapat prioritas terakhir dalam mendapatkan pekerjaan (Alsyaibani, 2013).

2.2.6 Mininet



Gambar 2. 3 Mininet Single Command

Sumber: (Mininet team, 2017)

Mininet adalah suatu emulator jaringan, yang membuat jaringan virtual secara realistis, menjalankan *real* kernel, *switch* dan *application code*, pada satu mesin tunggal (VM, *cloud* atau *native*), dalam hitungan detik, dengan satu perintah seperti terlihat pada Gambar 2.5 (Mininet Team, 2012). Mininet adalah software untuk melakukan prototyping jaringan yang sangat besar dengan menggunakan hardware yaitu laptop (Rohmat Tulloh, Ridha Muldina Negara, & Arif Nur Hidayat, 2015). Mininet menggunakan virtualisasi untuk menjalankan multihosting dan switch pada satu kernel operating sistem. Mininet penting untuk membangun *Openflow-SDN.* Mininet bisa menjalankan code bersama-sama pada sebuah mesin dalam mode *Command Line Interface (CLI)* disediakan bernama miniedit dan programnya pun dapat diubah-ubah sesuai keinginan pemakai.

2.2.6.1 Miniedit

Miniedit merupakan *User Interface* pada *mininet* yang dapat digunakan untuk membuat topologi jaringan *OpenFlow* dan melakukan fungsi seperti mininet itu sendiri. Sebenarnya tanpa menggunakan *Miniedit,* dapat juga membuat topologi jaringan *OpenFlow* yaitu dengan cara menggunakan perintah-perintah pada terminal. Disini *Miniedit* berfungsi untuk memudahkan pengguna untuk membuat topologi pada *mininet* dengan tanpa melakukan perintah-perintah yang ada pada terminal. Dengan *Miniedit*, untuk membuat topologi pada *miniedit* menggunakan sistem drag and drop. Berikut tampilan mininet pada gambar 2.4.

Gambar 2. 4 GUI-based Mininet Interface *(Miniedit)*

Berikut merupakan penjelasan komponen-komponen yang ada pada *Miniedit* yang dapat digunakan untuk membuat topologi jaringan seperti tabel 2.2.

Tabel 2. 2 Komponen Miniedit

No	Ikon	Nama	Fungsi
1		Kursor	Memilih dan Memindahkan komponen seperti *switch* dan *Controller*.
2		*Host*	Bertindak sebagai *host device* yang menggunakan layanan jaringan pada topologi.
3		*OpenFlow switch*	Bertindak sebagai *data plane* pada SDN.
4		Legacy *Switch*	Bertindak sebagai *switch* tradisional yang meneruskan paket menggunakan pemetaan alamat MAC berdasarkan letak *port* bersambungnya.
5		Legacy *Router*	Bertindak sebagai *router* tradisional yang mengambil keputusan penerusan paket berdasarkan protokol *routing*.
6		*Link*	Berfungsi untuk menghubungakan antara satu komponen dengan komponen yang lainnya.
7		*Controller*	Bertindak sebagai *Control Plane* pada SDN.

2.2.7 Httperf

Httperf merupakan program untuk mengukur kinerja dari *web server* dan *httperf* menyediakan fitur fleksibel dalam pembuatan beban kerja pada *web* atau *http server* (Mursanto, 2009). *Httperf* dapat membangkitkan sejumlah paket load dan mendukung HTTP/1.0 dan HTTP/1.1. *Httperf* merupakan tool yang digunakan untuk melakukan pengujian pada *web server.* Beberapa parameter pada *httperf* yang digunakan antara lain throughput, *connection rate* dan *reply time* pada suatu web server.

1. *Throughput* adalah Kecepatan transfer antara server dan client yang yang mempunyai satuan KB *(kilobytes)* per detik dan MB *(megabits)* per detik.
2. *Connection request/second* adalah banyaknya koneksi yang terhubung pada user dalam satu waktu *(rate).* Semakin tinggi nilai *connection rate* pada suatu sistem maka kualitas sistem semakin bagus, sedangkan semakin kecil nilai *connection rate* maka semakin buruk kualitas sistem tersebut.
3. *Reply time* adalah memberikan informasi tentang berapa lama waktu yang dibutuhkan server merespon dan berapa lama waktu yang dibutuhkan untuk menerimanya *(response)*. Semakin kecil nilai *reply time* dari sebuh

sistem maka semakin baik, sedangkan semakin besar nilai *reply time* dari sebuah sistem maka semakin buruk.

Berdasarkan penjelasan di atas, pengujian dilakukan pada client untuk melakukan *request* pada *web server*. Request berguna untuk memberikan beban permintaan *client* pada *server* dalam satu waktu sehingga dapat mengetahui kemampuan server dalam merespon permintaan dari *client*. *Rate* pada pengujian ini berguna untuk mengetahui tingkat kecepatan koneksi dalam satu waktu.