BAB 2
LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada bab ini dilakukan kajian terhadap penelitian-penelitian sebelumnya. Berikut tabel perbandingan penelitian terdahulu dan yang sekarang:

Tabel 2.1 Kajian Pustaka

No	Nama Penulis, Tahun dan Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1	Yen, Jin Y. (1970). "An algorithm for finding shortest routes from all source nodes to a given destination in general networks"	Pencarian Jalur terpendek ke semua *nodes*	Menggunakan *Djikstra*	Menggunakan *Yen algorithm* digunakan untuk mencari jalur terpendek antara dua *node*.
2	*Syahidillah, W., M. (2017). Multipath routing dengan load-balancing pada openflow software-defined network.*	Melakukan pencarian *Multipath* pada SDN Openflow	Melakukan implementasi dengan menggunakan algoritme DFS	Melakukan implementasi dengan menggunakan modifikasi *Yen algorithm* dan DFS

2.2 Dasar Teori

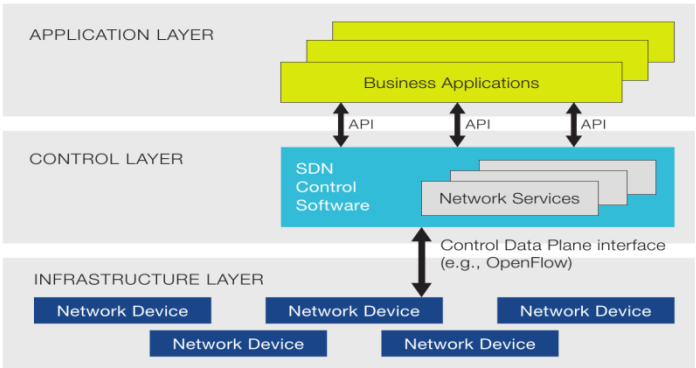
Pada subbab ini, dijabarkan berbagai teori-teori yang dapat mendukung penelitian ini, yaitu berbagai teori untuk mengembangkan suatu protokol *multipath routing* dengan kapabilitas *Yen algorithm* pada *OpenFlow Software-Defined Network*.

2.2.1 *Software-Defined Networking*

Konsep dari *Software Defined Network* (SDN) diperkenalkan pertama kali oleh Martin Cascado pada tahun 2007 di Stanford University dengan tulisan pada jurnalnya yang berjudul "*Ethane: Taking Control of The Enterprise*".*Ethane* dijelaskan sebagai sebuah arsitektur jaringan baru untuk perusahaan dengan cakupan yang besar.

Software Defined Network (SDN) adalah istilah yang merujuk pada konsep/paradigma baru dalam mendisain, mengelola dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan dan inovasi di bidang ini yg semakin lama semakin kompleks. Konsep dasar SDN adalah dengan melakukan pemisahan eksplisit antara *control* dan *forwarding plane*, serta kemudian melakukan abstraksi sistem dan meng-isolasi kompleksitas yg ada pada komponen atau sub-sistem dengan mendefinisikan antar-muka *(interface*) yg standard.

Hal tersebut dilakukan dengan memisahkan *control plane*, yaitu kecerdasan atau logika dari suatu perangkat jaringan yang mengatur bagaimana suatu paket dikirimkan pada suatu jaringan, dan *data plane*, yaitu perangkat yang menjalankan kegiatan penerusan atau pengiriman paket berdasarkan informasi yang diberikan *control plane*. Dengan SDN, suatu jaringan yang cenderung bersifat terdistribusi tadi dapat dipandang secara tersentralisasi sehingga memungkinkan pengaturan dan otomasi dari layanan jaringan yang lebih efisien (Astuto, Mendon¸ca, Nguyen, Obraczka, & Turletti, 2014).



Gambar 2.1 Arsitektur *Software-Defined Networking*

Sumber: sdxcentral.com (2015)

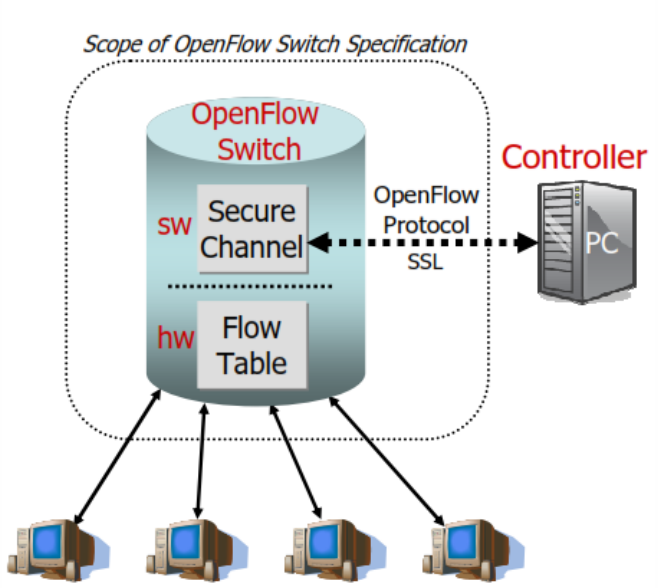
Menurut (Kreutz, et al., 2015), SDN merupakan arsitektur jaringan dengan empat pilar.

1. *Control* dan *data plane* dipisahkan. Fungsionalitas *control plane* dihilangkan dari perangkat jaringan dan hanya sebagai perangkat sederhana yang meneruskan paket *(switch)*.
2. Keputusan penerusan paket dilakukan berdasarkan *flow*, bukan berdasarkan *destination* atau tujuan paket. *Flow* secara luas didefinisikan sebagai suatu set kriteria atau filter terhadap paket, tergantung kapabilitas dari implementasinya. Sebagai contoh, *flow* dapat berupa sebuah koneksi TCP, atau semua paket yang berasal dari alamat MAC tertentu, atau semua paket dari *port switch* yang sama (McKeown, et al., 2008). Suatu *flow* pada suatu tabel *flow* biasa disertai dengan suatu set instruksi atau tindakan yang berlaku terhadap *flow* tersebut.
3. Logika kontrol dipindahkan pada suatu entitas terpisah, yang disebut *Controller* SDN atau NOS (*Network Operating System*). NOS adalah sebuah platform *software* yang menyediakan *library,* sumber daya, dan abstraksi untuk memfasilitasi pemrograman logika kontrol untuk mengatur perilaku perangkat-perangkat penerus paket (*switch*) berdasarkan suatu gambaran jaringan yang abstrak dan tersentralisasi.
4. Jaringannya dapat diprogram melalui aplikasi *software* yang berjalan di atas NOS dan dapat berinteraksi dengan perangkat-perangkat *data plane*

2.2.2 OpenFlow

OpenFlow adalah standar protokol/antarmuka komunikasi yang didefinisikan di antara lapisan kontrol (*control plane*) dan *forwarding* (*data plane*) dari sebuah arsitektur SDN (*Open Networking Foundation*, 2016). *OpenFlow* menyediakan protokol terbuka untuk memrogram tabel *flow* dari berbagai *router* dan *switch* yang berbeda. *OpenFlow* menyediakan cara yang standar dan terbuka untuk sebuah *Controller* SDN untuk berkomunikasi dengan sebuah *switch* yang terpasang *OpenFlow*. (McKeown, et al., 2008).

Secara umum, *OpenFlow* memiliki cara kerja sebagai berikut: ketika sebuah s*witch OpenFlow* menerima paket yang belum pernah ditemui sebelumnya, yang tidak memiliki entri *flow* yang cocok, maka *switch* tersebut akan mengirimkan paket ini ke C*ontroller*. *Controller* kemudian akan mengambil keputusan untuk menangani paket ini. *Controller* dapat melakukan *drop* terhadap paket tersebut, atau dapat menambahkan entri *flow* yang akan mengarahkan *switch* tentang bagaimana meneruskan paket yang serupa nantinya. (*Open Networking Foundation*, 2016).



Gambar 2.2 Arsitektur *OpenFlow*

Sumber: (McKeown, et al., 2008)

Pada *switch OpenFlow*, sesuai dengan konsep SDN, dilakukan penghapusan fungsi *control plane*. *Data plane* dari sebuah Switch OpenFlow terdiri atas tabel *flow* dan set instruksi yang berlaku untuk setiap entri *flow*. Set instruksi yang dapat didukung oleh sebuah Switch OpenFlow minimum terdiri atas (namun juga dapat ditambahkan):

1. Meneruskan (*forward*) paket dari suatu *flow* menuju suatu port atau beberapa port tertentu.
2. Mengenkapsulasi dan meneruskan paket menuju *Controller.* Tindakan ini biasa digunakan untuk paket pertama dari *flow* baru, sehingga *Controller* dapat memutuskan jika *flow* tersebut harus ditambahkan dalam tabel *flow* atau tidak. Selain itu, tindakan ini dapat digunakan agar *Controller* dapat memproses suatu paket untuk keperluan tertentu.
3. Melepaskan (*drop*) paket dari *flow*. Tindakan ini dapat digunakan untuk menjaga keamanan jaringan sehingga paket-paket berbahaya tidak bersikulasi dalam jaringan.

Sebuah *switch OpenFlow* setidaknya terdiri atas tiga bagian: (1) tabel *flow*, dengan tindakan yang terkait dengan setiap entri *flow*, untuk memberitahu *switch* bagaimana *flow* diproses, (2) saluran/kanal aman (*Secure Channel*) yang menghubungkan *switch* dengan *Controller* SDN, yang memungkinkan berbagai perintah dan paket dapat dikirimkan antara *Controller* dan *Switch* menggunakan (3) Protokol *OpenFlow*, yang menyediakan cara yang terbuka dan standar untuk *Controller* agar dapat berkomunikasi dengan *switch*. (McKeown, et al., 2008).

2.2.3 *Controller*

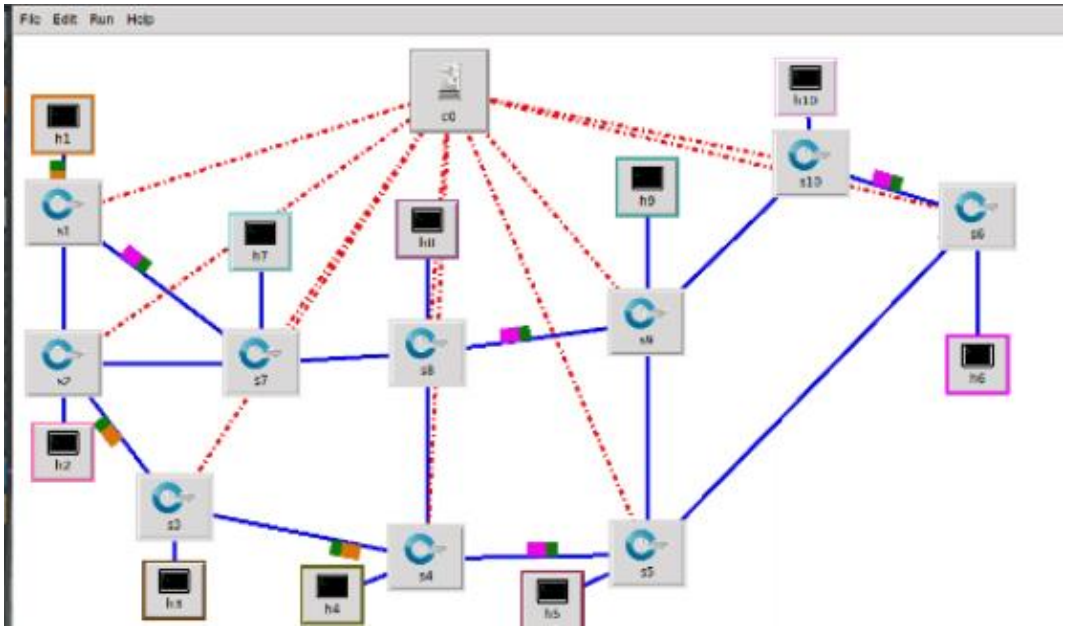
Controller SDN adalah sebuah aplikasi dalam SDN yang mengelola kontrol *flow* untuk memungkinkan jaringan yang cerdas. *Controller* SDN didasarkan pada sebuah protokol, seperti *OpenFlow*, yang memungkinkan *server* untuk memberitahu *switch* ke mana suatu paket harus dikirimkan. (Rouse, SDN *controller* (*software-defined networking controller*), 2012).

Controller SDN berfungsi sebagai suatu sistem operasi (OS) untuk jaringan. Semua komunikasi antara aplikasi dan perangkat jaringan harus melalui *controller*. Protokol *OpenFlow* menghubungkan *software Controller* dengan perangkat jaringan sehingga perangkat lunak *server* dapat memberitahu *switch* mana paket harus dikirimkan. Karena *control plane* jaringan diimplementasikan dalam bentuk perangkat lunak daripada *firmware* dari perangkat keras, *traffic* jaringan dapat dikelola secara lebih dinamis dan pada tingkat yang jauh lebih rinci. (Rouse, *OpenFlow controller*, 2013).

2.2.4 *Multipath Routing*

Multipath routing merupakan pendekatan *routing* alternatif yang dapat mendistribusikan *traffic* jaringan dengan berbagai rute atau jalur yang terdapat pada jaringan (Jo, Pan, Liu, & Butler, 2014). Secara umum, *multipath routing* dapat dikatakan lebih efisien daripada *single path routing*. Internet akan lebih efisien dan handal jika router dapat membagi *traffic* dengan baik melalui *multipath*, dan memiliki satu atau beberapa rute alternatif untuk meningkatkan keamanan, performa, dan kehandalan dari suatu jaringan (He & Rexford, 2008).

Traffic jaringan dibagi oleh *multipath routing* dan mengirimkan *traffic* melalui jalur-jalur yang berbeda pada jaringan. Hal tersebut berbeda dengan skema *single path routing* yang hanya meneruskan paket melalui satu jalur yang terbaik, dimana *congestion* atau kemacetan pada jaringan akan terjadi. Pada *multipath routing,* kemacetan tersebut dapat dihindari karena *traffic* jaringan dapat diteruskan melalui beberapa kemungkinan jalur terbaik. (Ramdhani, Hertiana, & Dirgantara, 2016).



Gambar 2.3 Ilustrasi *Multipath routing Yen algorithm*

2.2.5 *Mininet*

Mininet adalah sebuah *emulator* jaringan yang membangun jaringan yang terdiri atas *host* virtual, *switch*, *controller* dan *link* (Mininet Team, 2016).*Mininet* merupakan sebuah sistem untuk melakukan *rapid prototyping* jaringan yang sangat besar pada sumber daya yang terbatas, seperti sebuah laptop (Lantz, Heller, & McKeown, 2010). *Mininet* menggunakan virtualisasi berbasis proses untuk menjalankan banyak *host* dan *switch* pada satu *kernel OS*. *Mininet* memanfaatkan *virtual software switch OpenvSwitch* untuk membangun topologi *OpenFlow* SDN. *Mininet* dapat dijalankan dalam mode CLI (*Command Line Interface*) maupun dalam GUI (*Graphical User Interface*) yang disediakan bernama *Miniedit*.

2.2.6 *Ryu*

Ryu merupakan *framework* berbasis komponen *software defined network* yang menyediakan komponen *software* dengan API yang membuatnya mudah untuk mengembangkan dan mengelola aplikasi kontrol dan manajemen jaringan. Pengembangan aplikasi untuk Ryu dapat dilakukan dengan menggunakan bahasa *Python* atau dengan mengirimkan pesan JSON melalui API yang tersedia. Ryu mendukung berbagai protokol untuk memanajemen jaringan antara lain *OpenFlow, NetConf, Of-config* dll. Kebutuhan atas Ryu sebagai *OpenFlow controller* adalah karena *Ryu* mendukung *OpenFlow* versi 1.0 hingga 1.5, dimana pada *OpenFlow* versi 1.1 tersedia *group actions* yang dapat digunakan untuk *multipath routing*. *Ryu* di sini digunakan sebagai *Controller* SDN dan sebagai

framework untuk mengembangkan sebuah sistem *multipath routing* di *OpenFlow* SDN.

2.2.7 *Yen Algoritm*

Yen algorithm merupakan pengembangan dari algoritme Dijkstra. Y*en algorithm* menghitung atau mencari satu sumber jalur K-terpendek. algoritme jalur terpendek untuk menemukan jalan terbaik, kemudian melanjutkan untuk menemukan K - 1 penyimpangan dari jalur terbaik (Yen, Jin Y.1970). Algoritme dapat dipecah menjadi dua bagian, menentukan jalan k-terpendek pertama dan kemudian menentukan semua jalur k-terpendek lainnya. K-Shortest Path routing merupakan pengembangan dari algoritma Shortest Path Routing yang digunakan pada jaringan. Dasar dari algoritma routing yang digunakan dalam K-*Shortest* path routing adalah algoritme routing pada *shortest path routing*. K-*Shortest Path Routing* digunakan untuk mencari sebanyak K-jalur yang memiliki *cost* paling kecil dalam jaringan. K-*Shortest path routing* termasuk ke dalam mekanisme *multipath routing* dimana, K-*Shortest path routing* mampu menggunakan lebih dari 1 jalur untuk mengirimkan data, hal ini akan meminimalisir terjadinya kemacetan pada jaringan.

2.2.7 DFS (Depth-First-Search)

Salah satu algoritme penelusuran struktur graf / pohon berdasarkan kedalaman. Simpul ditelusuri dari root kemudian ke salah satu simpul anaknya (misalnya prioritas penelusuran berdasarkan anak pertama [simpul sebelah kiri]), maka penelusuran dilakukan terus melalui simpul anak pertama dari simpul anak pertama level sebelumnya hingga mencapai level terdalam. Setelah sampai di level terdalam, penelusuran akan kembali ke 1 level sebelumnya untuk menelusuri simpul anak kedua pada pohon biner [simpul sebelah kanan] lalu kembali ke langkah sebelumnya dengan menelusuri simpul anak pertama lagi sampai level terdalam dan seterusnya.