BAB 1 PENDAHULUAN

1.1 Latar belakang

Software-Defined Network (SDN) sebuah konsep pendekatan jaringan komputer dimana sistem pengkontrol dari arus data dipisahkan dari perangkat lainnya. menurut (Antonio Capone, Carmelo Cascone, Alessandro Q.T. Nguyen, & Brunilde Sanso, 2015), pendekatan jaringan secara traditional diganti dengan jaringan terpusat yang dapat mengatur *traffic* dengan melalui aturan pemrograman *forwading* yang tingkat rendah ke dalam *switch* berdasarkan abstraksi yang sederhana dari fungsi *switch* seperti didefinisikan di *OpenFlow* dengan *flow table*. Meskipun SDN dan *OpenFlow* menyediakan fleksibilitas yang besar dan sebuah platform yang sangat kuat untuk memprogram beberapa tipe aplikasi jaringan yang inovatif tanpa keterbatasan yang dialami protokol-protokol terdistribusi, untuk mengimplementasi fungsi-fungsi tradisional penting pada SDN dan *OpenFlow*, seperti ketahanan kegagalan, tidak mudah maupun efesien, karena reaksi terhadap event atau kejadian pada jaringan akan selalu melibatkan controller pusat (pemberitahuan dan instalasi aturan baru *forwading*) dengan *delay* dan beban sinyal yang tidak dapat dihindari (Antonio Capone, Carmelo Cascone, Alessandro Q.T. Nguyen, & Brunilde Sanso, 2015). Dalam SDN sering terjadi sebuah gangguan *link* yang mengganggu pada traffic.

Link failure atau kegagalan *link* pada suatu jaringan merupakan salah satu permasalahan yang perlu diperhatikan oleh operator jaringan. Efek dari *link failure* yaitu *packet loss* yang akan menyebabkan pengurangan *throughput* pada jaringan sebenarnya (Ying-Dar Lin, Hung-Yi Teng, Chia-Rong Hsu, Chun-Chieh Liao, & Yuan-Cheng Lai, 2016). Untuk mengatasi hal tersebut diperlukan suatu mekanisme manejemen kegagalan sehingga layanan jaringan dapat disediakan dengan lancar dan dengan gangguan yang minimal. Tantangannya adalah dibutuhkan reaksi yang cepat terhadap kegagalan dari komponen jaringan dengan mengkonfigurasi ulang alokasi sumber daya agar dapat memanfaatkan infrastruktur yang bertahan untuk menyediakan layanan (Antonio Capone, Carmelo Cascone, Alessandro Q.T. Nguyen, & Brunilde Sanso, 2015).

Pada versi *openflow* terbaru yaitu *openflow* 1.3 memperkenalkan fitur terbaru pada *OpenFlow* yaitu mekanisme *fast-failover*, untuk memungkinkan reaksi yang lokal dan cepat terhadap kegagalan tanpa perlu meminta keputusan dari *controller* terpusat. Fitur ini diperoleh dengan memasang beberapa tindakan terhadap *entry flow* yang sama dan menerapkannya berdasarkan status dari *link* (aktif atau gagal) (Antonio Capone, Carmelo Cascone, Alessandro Q.T. Nguyen, & Brunilde Sanso, 2015). Pada mekanisme *Fast-failover,* controller secara periodik memperhitungkan *multiple path* secara berkala untuk *source-destination* dan memasangl *flow* dan *group entry* pada masing-masing *switch*. Ketika kesalahan *link* terdeteksi, *switch* dapat mengalihkan jalur ke jalur yang lain (Ying-Dar Lin, Hung-Yi Teng, Chia-Rong Hsu, Chun-Chieh Liao, & Yuan-Cheng Lai, 2016). Koneksi

dari host dapat menata ulang *link* yang rusak ketika jalur sudah dialihkan ke jalur yang lain untuk sementara. Dengan memanfaatkan *multipath routing, controller* menghitung jalur yang mungkin sebagai pohon akar yang terpisah antara masing-masing *source-destination* (Ghannami & Shao, 2016).

Routing adalah proses menentukan rute atau jalur yang diambil oleh paket dimana mereka mengalir dari pengirim ke penerima. Algoritme yang menghitung jalur ini disebut algoritme *routing*. Sebuah algoritme routing akan menentukan, Misalnya, jalan di sepanjang dimana paket mengalir dari H1 ke H2 (Kurose & Ross, 2013). Algoritme routing yang digunakan adalah algoritme *Dijkstra* dan algoritme DFS. Algoritme *Link*-State atau yang biasa disebut dengan algoritme *Dijkstra*. Dalam algoritme *link*-state, topologi jaringan dan bobot semua *link* diketahui oleh setiap node, topologi dan bobot *link* tersebut kemudian akan diproses sebagai masukan untuk algoritme LS. Dalam prakteknya, hal ini dicapai dengan setiap node melakukan *broadcast* yang berisi paket *Link*-State dimana paket tersebut berisi bobot *link* yang terhubung dan identitas masing-masing *node*. Sedangkan Algoritme DFS yaitu metode pencarian jalur yang dilakukan dengan menggunakan struktur data *stack* yang dapat menyimpan rute kemudian melakukan ekspansi lagi meskipun lagi sudah ditemukan satu jalur ke tujuan (Maulana S., Yahya, & Rizqika A., 2017).

Pada penelitian sebelumnya tentang analisis *Fail Path* Arsitektur *Software Defined Network* menggunakan *Dijsktra Algorithm* yang pernah dilakukan oleh Eka Putri Aprilianingsih dari Universitas Brawijaya pada 2017 lalu. Pada penelitiannya melakukan Analisa *fail Path* yang menggunakan algoritme *Dijkstra*. Pada penelitiannya, sistem tidak menggunakan sebuah mekanisme Fast-Failover untuk mengatasi link yang putus tetapi menggunakan algoritme *Dijkstra* untuk mencari jalur pendek ketika ada *fail path*. (Aprilianingsih, 2017)

Oleh karena itu, penulis akan melakukan implementasi *Fast-failover Link* pada *multipath routing* jaringan *OpenFlow software-defined network* yang berdasarkan pada gangguan di *link* topologi dan penelitian ini diimplementasikan pada *Mininet* dan memakai *Ryu Controller* yang menggunakan Bahasa *Phyton*. Dengan itu diharapkan dapat mengatasi *link* seperti *link* yang putus pada topologi.

1.2 Rumusan masalah

Berdasarkan pada latar belakang diatas, dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana mekanisme fast-failover dapat mengalihkan ke alternatif *link* ketika *link* diputus?
2. Bagaimana kinerja Performa mekanisme fast-failover pada jaringan Software-Defined Network?

1.3 Tujuan

Penelitian ini bertujuan untuk menerapkan manajemen gangguan link pada yang dapat menemukan beberapa jalur pada jaringan dan ketika ada

gangguan link pada primary link sistem akan mengarahkan ke jalur alternatif yang ada.

1.4 Manfaat

Adapun manfaat penelitian yang diharapkan adalah sebagai berikut:

a. Manfaat bagi penulis

Dapat meningkatkan dan memantapkan pengetahuan teoritis maupun aplikatif terhadap SDN *OpenFlow*.

b. Manfaat bagi jaringan

1. Dapat menghasilkan jaringan dengan kehandalan yang tinggi.
2. Dapat mengurangi resiko gangguan pada jaringan.

1.5 Batasan masalah

Agar pembahasan masalah dapat terarah dengan baik dan tidak menyimpang dari pokok permasalahan, maka penulis membtasi permasalahan yang akan dibahas, yakni:

1. Implementasi SDN yang digunakan adalah *OpenFlow*.
2. Menggunakan *Ryu Controller* sebagai *controller SDN* sehingga Bahasa pemrograman yang digunakan adalah *python*.
3. Dilakukan dalam lingkungan emulasi jaringan *Mininet*.
4. Gangguan hanya pada *Link* yang putus.

1.6 Sistematika pembahasan

Uraian singkat mengenai struktur penulisan pada masing-masing bab adalah sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini mencakup latar belakang, rumusan masalah, tujuan yang ingin dicapai, manfaat yang dapat diperoleh, batasan masalah dan sistematika penulisan.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menguraikan kajian pustaka dan dasar teori dari sumber-sumber yang relevan untuk diguanakan panduan dalam penelitian ini.

BAB 3 METODOLOGI

Bab ini berisi pembahasan meotodologi yang digunakan pada penelitian ini. Di sini berisi pemaparan langkah kerja terdiri atas, studi pustaka, analisis kebutuhan, desain sistem, implementasi, pengujian, teknik analisis dan tek pengambilan kesimpulan.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas tentang rancangan apa saja yang akan dibuat dalam penelitian dan tentang penerapan *Multipath Routing* dengan *Fast-failover Link* pada *OpenFlow* Software-Defined Network dengan menggunakan Ryu sebagai controller dan Mininet sebagai simulasi.

BAB 5 PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian dan analisa terhadap percobaan yang sudah dilakukan pada sistem.

BAB 6 PENUTUP

Bab ini berisikan kesimpulan dan saran yang didapat dari pengujian sistem.