PENGGUNAAN JALUR DALAM *MULTIPATH ROUTING* SECARA PROPORSIONAL BERDASARKAN KEBUTUHAN *BANDWIDTH* PENGGUNA PADA *SOFTWARE DEFINED NETWORKING*

SKRIPSI

Untuk memenuhi sebagian persyaratan   
memperoleh gelar Sarjana Komputer

Disusun oleh:

Muhamad Miftahur Ridhoilah

NIM: 155150200111174



TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2019

PERSETUJUAN

PENGGUNAAN JALUR DALAM *MULTIPATH ROUTING* SECARA PROPORSIONAL BERDASARKAN KEBUTUHAN *BANDWIDTH* PENGGUNA PADA *SOFTWARE DEFINED NETWORKING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun Oleh :

Muhamad Miftahur Ridhoilah

NIM: 155150200111174

Telah diperiksa dan disetujui oleh:

|  |  |
| --- | --- |
| Dosen Pembimbing I  Achmad Basuki, S.T, M.MG, Ph.D  NIP: 19741118 200312 1 002 | Dosen Pembimbing II  Kasyful Amron, S.T, M.Sc  NIP. 19750803 200312 1 003 |

Mengetahui

Ketua Jurusan **Teknik Informatika**

Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 11 Mei 2020



­

Muhamad Miftahur Ridhoilah

NIM: 155150200111174

PRAKATA

Penulis mengucapkan puji syukur atas semua rizki, nikmat, berkah, bantuan, dan panduan yang diberikan oleh Tuhan Yang Maha Esa dalam menyelesaikan penelitian ini. Banyak kesalahan yang dihadapi Penulis yang tanpa kehadiran-Nya Penulis tidak dapat menyelesaikan penelitian ini.

Proses penyusunan skripsi ini tidak terlepas dari bantuan, bimbingan dan dukungan dari berbagai pihak. Penulis hendak mengucapkan terima kasih kepada mereka yang telah berperan besar dalam penyelesain penelitian ini, baik secara langsung maupun tidak langsung. Mereka adalah

1. Kedua orang tua Penulis yang selalu mendukung dan mendo’akan penulis.
2. Bapak Achmad Basuki, S.T, M.MG, Ph.D dan Bapak Kasyful Amron, S.T, M.Sc selaku dosen pembimbing yang telah sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Teman seperjuangan (Azzu, Rizaldy, Agus, Rino, Vito, Fikar, Arka, Luthfi, Fadil, Sisco, Greg, dan Dani) yang sudah memberikan dukungan, bantuan dan kenangan selama penulis menumpuh perkuliahan.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya Malang (UB) yang telah memberikan kesempatan untuk pelaksanaan penelitian.
5. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika FILKOM UB yang telah memberikan kesempatan untuk pelaksanaan penelitian.

Penulis menyadari bahwa penelitian ini belum sempurna. Oleh karena itu, saran dan kritik Penulis butuhkan untuk penyempurnaan penelitian selanjutnya. Kemudian, Penulis dapat dihubungi dalam alamat surel yang tertera di bawah.

Malang, 11 Mei 2020

Penulis

m.miftahurr@gmail.com

ABSTRAK

**Muhamad Miftahur Ridhoilah, Penggunaan Jalur Dalam *Multipath routing* Secara Proporsional Berdasarkan Kebutuhan *Bandwidth* Pengguna pada Software Defined Networking**

**Pembimbing: Achmad Basuki, S.T, M.MG, Ph.D dan Kasyful Amron, S.T, M.Sc**

*Multipath routing* adalah teknik *routing* yang menggunakan jalur lebih dari satu untuk melakukan pengiriman data dalam jaringan. Penggunaan *multipath routing* memunculkan masalah pembagian trafik jaringan yang tidak proporsional. Pada jaringan dengan *bandwidth* jalur yang berbeda, pembagian trafik jaringan secara rata dapat menyebabkan *bottleneck* di salah satu jalur sehingga terjadi penurunan kinerja dalam pengiriman data. Oleh karena itu, penelitian ini mengusulkan mekanisme penggunaan jalur yang mengatur distribusi trafik jaringan secara proporsional dengan *demand* (kebutuhan *bandwidth* pengguna) dan ketersediaan *bandwidth* jalur. Nilai kebutuhan *bandwidth* pengguna didapatkan dengan memantau trafik jaringan dari pengguna sedangkan nilai ketersersedian *bandwidth* jalur didapatkan dengan memantau nilai pemakaian *bandwidth* jalur. Pemantauan dari nilai yang dibutuhkan dilakukan dengan SFlow-rt. Kemudian nilai hasil pemantauan digunakan sebagai bobot jalur (*bucket weight*) untuk mengatur distribusi trafik jaringan dengan fitur group action OpenvSwitch. Pengujian dari sistem menghasilkan pengiriman data yang menyesuaikan dengan *demand* dan ketersedian *bandwidth*. Hasil distribusi trafik jaringan menunjukkan penggunaan jalur yang melihat *demand* dan mengutamakan jalur dengan ketersedian *bandwidth* yang besar. Waktu respons dari sistem dalam merubah distribusi trafik jaringan berubah sesuai dengan nilai *polling* dari SFlow. Nilai *polling* 1 menghasilkan range waktu respons antara 1 detik sampai 3 detik. Nilai *polling* 3 menghasilkan range waktu respons antara 1 detik sampai 5 detik. Nilai *polling* 10 menghasilkan range waktu respons antara 14 detik sampai 16 detik dengan kemungkinan tidak terjadi perubahan distribusi trafik meskipun *demand* bernilai besar.

Kata kunci: *Multipath routing*, distribusi trafik jaringan, proportional, *bandwidth, bucket weight*

ABSTRACT

**Muhamad Miftahur Ridhoilah, *Multipath Routing with Bandwidth Proportional of Paths Utilization in Software-Defined Networking***

***Supervisors*: Achmad Basuki, S.T, M.MG, Ph.D *and* Kasyful Amron, S.T, M.Sc**

Multipath routing is a routing method that uses more than one path for transferring data in Network. Using multipath routing can cause not proportional traffic distribution. In a network with different bandwidth, evenly distributed traffic can cause a drop in data transfer performance because of the bottleneck in one of the available paths. This paper proposes a system that controls traffic distribution according to the value of demand (traffic from the user) and path unused bandwidth. The system uses SFlow-rt to monitor the value of demand and path unused bandwidth and uses group action feature of OpenvSwitch to distribute the traffic. The values from SFlow will be used to calculate the weight of all available paths that control traffic distribution. The test result shows traffic distribution that considers the value of demand and prioritizes path with more unused bandwidth. The system response time that change traffic distribution is effected by the value of SFlow polling. If the value of SFlow polling is 1, then system response time is in range of 1 to 3 seconds. If the value of SFlow polling is 3, then system response time is in range of 1 to 5 seconds. If the value of SFlow polling is 10, then system response time is in range of 14 to 16 seconds with a chance of no change in traffic distribution even the value of demand is high.

*Keyword: Multipath routing, traffic distribution, proportional, bandwidth, bucket weight*

DAFTAR ISI

[PERSETUJUAN ii](#_Toc38533788)

[PERNYATAAN ORISINALITAS iii](#_Toc38533789)

[PRAKATA iv](#_Toc38533790)

[ABSTRAK v](#_Toc38533791)

[ABSTRACT vi](#_Toc38533792)

[DAFTAR ISI vii](#_Toc38533793)

[DAFTAR TABEL ix](#_Toc38533794)

[DAFTAR GAMBAR x](#_Toc38533795)

[BAB 1 Pendahuluan 1](#_Toc38533796)

[1.1 Latar Belakang 1](#_Toc38533797)

[1.2 Identifikasi Permasalahan 2](#_Toc38533798)

[1.3 Rumusan Masalah 3](#_Toc38533799)

[1.4 Tujuan 3](#_Toc38533800)

[1.5 Manfaat 3](#_Toc38533801)

[1.6 Batasan Masalah 3](#_Toc38533802)

[1.7 Sistematika Pembahasan 4](#_Toc38533803)

[BAB 2 Landasan Kepustakaan 5](#_Toc38533804)

[2.1 Kajian Pustaka 5](#_Toc38533805)

[2.2 Dasar Teori 8](#_Toc38533806)

[2.2.1 *Software* Defined *Networking* (SDN) 8](#_Toc38533807)

[2.2.2 OpenFlow 9](#_Toc38533808)

[2.2.3 *OpenvSwitch* 10](#_Toc38533809)

[2.2.4 *Multipath routing* 11](#_Toc38533810)

[2.2.5 Perangkat Pendukung 11](#_Toc38533811)

[BAB 3 Metodologi Penelitian 14](#_Toc38533812)

[3.1 Analisis Kebutuhan 15](#_Toc38533813)

[3.2 Perancangan 16](#_Toc38533814)

[3.2.1 Perancangan Lingkungan Uji 16](#_Toc38533815)

[3.2.2 Perancangan *Multipath routing* 18](#_Toc38533816)

[3.2.3 Perancangan Pengujian Kinerja 20](#_Toc38533817)

[3.3 Pengujian dan Pembahasan 21](#_Toc38533818)

[3.3.1 Pengujian Distribusi Trafik Jaringan 22](#_Toc38533819)

[3.3.2 Pengujian Waktu Respons Pemilihan Jalur 23](#_Toc38533820)

[3.4 Penarikan Kesimpulan 23](#_Toc38533821)

[BAB 4 Implementasi 24](#_Toc38533822)

[4.1 Implementasi Lingkungan Uji 24](#_Toc38533823)

[4.2 Implementasi *Multipath routing* 25](#_Toc38533824)

[4.2.1 Implementasi Mekanisme Topologi Learning 25](#_Toc38533825)

[4.2.2 Implementasi Mekanisme Pencarian Jalur 26](#_Toc38533826)

[4.2.3 Implementasi Mekanisme Pembagian Trafik Jaringan 27](#_Toc38533827)

[4.3 Implementasi Pengujian Kinerja 31](#_Toc38533828)

[4.3.1 Distribusi Trafik Jaringan 31](#_Toc38533829)

[4.3.2 Waktu Respons Pemilihan Jalur 33](#_Toc38533830)

[BAB 5 Pembahasan Hasil Pengujian 35](#_Toc38533831)

[5.1 Distribusi Trafik Jaringan 35](#_Toc38533832)

[5.2 Waktu Respons Pemilihan Jalur 37](#_Toc38533833)

[BAB 6 Kesimpulan 42](#_Toc38533834)

[DAFTAR REFERENSI 43](#_Toc38533835)

DAFTAR TABEL

[Tabel 2. 2 Tabel Tipe *Group entry* 10](#_Toc38533836)

[Tabel 3. 1 Tabel Parameter Uji 21](#_Toc38533837)

[Tabel 3. 2 Tabel Skenario Pengujian 22](#_Toc38533838)

[Tabel 4. 4 Perintah dengan Topologi Learning 25](#_Toc38533839)

[Tabel 4. 5 Pseudocode Implementasi Mekanisme Pencarian Jalur 26](#_Toc38533840)

[Tabel 4. 6 Pseudocode Implementasi Mekanisme pemantauan Trafik jaringan 28](#_Toc38533841)

[Tabel 4. 7 Pseudocode Implementasi Mekanisme Pemilihan jalur 29](#_Toc38533842)

[Tabel 4. 8 Perintah Iperf3 untuk Pengujian Distribusi Trafik Jaringan 32](#_Toc38533843)

[Tabel 4. 9 Perintah BWM-NG Iperf3 untuk Pengujian Distribusi Trafik Jaringan 32](#_Toc38533844)

[Tabel 4. 10 Hasil Pengujian Distribusi Trafik Jaringan 32](#_Toc38533845)

[Tabel 4. 11 Perintah Iperf3 untuk Pengujian Waktu Respons Pemilihan Jalur 33](#_Toc38533846)

[Tabel 4. 12 Perintah BWM-NG untuk Pengujian Waktu Respons Pemilihan Jalur 33](#_Toc38533847)

[Tabel 4. 13 Hasil Pengujian Waktu Respons 34](#_Toc38533848)

DAFTAR GAMBAR

[Gambar 2. 1 Gambaran Arsitektur dari SDN 8](#_Toc38533773)

[Gambar 2. 2 SFlow-rt 12](#_Toc38533774)

[Gambar 3. 1 Diagram Alur Tahapan Penelitian 14](#_Toc38533775)

[Gambar 3. 2 Rancangan Topologi 17](#_Toc38533776)

[Gambar 4. 1 Topologi 24](#_Toc38533777)

[Gambar 4. 2 Koneksi dari Topologi Jaringan 25](#_Toc38533778)

[Gambar 5. 1 Hasil Distribusi Trafik Jaringan Skenario 2 35](#_Toc38533779)

[Gambar 5. 2 Hasil Distribusi Trafik Jaringan Skenario 3 36](#_Toc38533780)

[Gambar 5. 3 Graph *Throughput* S1 (Nilai *Polling* SFlow 1) 38](#_Toc38533781)

[Gambar 5. 4 Graph *Throughput* S1 (Nilai *Polling* SFlow 3) 39](#_Toc38533782)

[Gambar 5. 5 Graph *Throughput* S1 (Nilai *Polling* SFlow 10) 40](#_Toc38533783)

# Pendahuluan

Pendahuluan membahas tentang latar belakang dari penelitian. Ulasan dari latar belakang berisi masalah yang muncul dalam penggunaan *multipath routing* dan solusi yang diajukan dalam penelitian. Selain latar belakang, pendahuluan berisi pertanyaan yang muncul untuk menyelsaikan masalah yang muncul dari latar belakang, tujuan dari penelitian, manfaat dari penelitian, batasan dari masalah dan sistematika pembahasan dari penelitian. [kurang panjang]

## Latar Belakang

Peningkatan jumlah aplikasi jaringan meningkatkan kebutuhan terhadap kemampuan pengiriman data yang efisien. Salah satu cara yang digunakan untuk meningkatkan kemampuan pengiriman data adalah penggunaan skema pengiriman *multipath routing* dengan *software defined networking* (SDN). *Multipath routing* adalah teknik *routing* yang menggunakan jalur lebih dari satu untuk melakukan pengiriman data(Gorranson, et. al., 2017). Jaringan dapat menghindari terjadinya *congestion* di sebuah jalur saat pengiriman data dengan menggunakan skema *multipath routing*. Jaringan dengan skema *multipath routing* menghasilkan *throughput* pengiriman data yang lebih tinggi dibanding jaringan dengan *singlepath routing* (Chiang, et. al., 2017).

Masalah yang muncul karena penggunaan *multipath routing* adalah pembagian trafik jaringanyang tidak proporsional. Kasus pembagian trafik jaringan yang tidak proporsional dapat terjadi pada jaringan dengan *bandwidth* jalur yang berbeda menggunakan skema *equal* *cost* *multipath routing* (ECMP). Pada ECMP, trafik jaringan dibagi secara rata jika terdapat dua atau lebih jalur terbaik dengan nilai yang sama (RFC 2992, 2000). Pembagian secara rata pada jalur yang memiliki *bandwidth* berbeda dapat menyebabkan *bottleneck* saat pengiriman data. *Bottleneck* terjadi karena jalur dengan *bandwidth* yang rendah mendapatkan beban trafik jaringan yang lebih besar daripada *bandwidth* jalur.

Salah satu solusi dari pembagian trafik jaringan yang tidak proporsional adalah penggunaan bobot jalur (Wirawan, et. al., 2018). Dengan memberikan bobot terhadap jalur, jalur yang digunakan untuk pengiriman data akan sesuai dengan bobot dari jalur yang sudah ditentukan sehingga trafik jaringan terbagi sesuai dengan bobot dari jalur. Namun, pemobobotan jalur pada penelitian Wirawan masih bersifat statis. Kenyataannya, beban trafik jaringan dan kondisi jalur pada jaringan tidak bersifat statis. Beban trafik jaringan berubah sesuai dengan beban trafik jaringan dari pengguna, sedangkan kondisi jalur berubah sesuai penggunaan dari jalur. Oleh karena itu, Wirawan menyarankan untuk menggunakan bobot yang dinamis. Bobot yang dinamis dapat membagi trafik jaringan secara proporsional.

Untuk membagi trafik jaringan dengan lebih proporsional, penelitian ini mengusulkan sistem *multipath routing* dengan penggunaan jalur yang diatur. Penggunaan jalur diatur dengan memberikan bobot pada jalur secara dinamis berdasarkan *demand* dan ketersedian *bandwidth* dari jalur. Hasil yang diharapkan dari penelitian ini adalah sistem *multipath routing* dapat membagi *demand* secara proporsional sesuai dengan ketersediaan *bandwidth* jalur yang berubah-ubah, sehingga tidak terjadi kelebihan beban trafik jaringan di jalur yang memiliki ketersediaan *bandwidth* kecil.

## Identifikasi Permasalahan

Penggunaan *multipath* routing memunculkan masalah pembagian trafik jaringan dalam pengiriman data. Pembagian trafik jaringan yang tidak proporsional dengan ketersedian *bandwidth* sebuah jalur dapat menimbulkan bottleneck dalam pengiriman data. Contohnya adalah beban trafik jaringan yang lebih besar daripada *bandwidth* sebuah jalur. Salah satu solusi yang sudah ada adalah pemberian bobot statis terhadap sebuah jalur, sehingga jalur dengan bobot lebih besar akan lebih sering digunakan untuk mengirimkan data daripada jalur dengan bobot yang kecil. Akan tetapi, beban trafik jaringan dan kondisi jalur pada jaringan tidak bersifat statis. Beban trafik jaringan berubah sesuai dengan beban trafik jaringan dari pengguna, sedangkan kondisi jalur berubah sesuai penggunaan dari jalur.

Untuk membagi trafik jaringan dengan lebih proporsional, penelitian ini mengusulkan sistem *multipath routing* dengan penggunaan jalur yang diatur. Penggunaan jalur diatur dengan memberikan bobot pada jalur secara dinamis berdasarkan *demand* dan ketersedian *bandwidth* dari jalur. Dalam penelitian ini, *demand* adalah kebutuhan *bandwidth* pengguna yang ditunjukkan oleh nilai beban trafik jaringan dari pengguna. Pada kondisi *demand* yang rendah (dapat dilayani dengan 1 jalur), pengiriman data dilakukan dengan jalur yang memiliki ketersedian *bandwidth* terbesar. Pada kondisi *demand* yang tinggi (tidak bisa dilayani dengan 1 jalur), pengiriman data dilakukan dengan semua jalur yang ditemukan dengan rasio pembagian trafik jaringan yang mengikuti ketersediaan *badwidth* jalur.

Sistem *multipath routing* dengan penggunaan jalur yang diatur akan diuji kinerja distribusi trafik jaringan dalam pengiriman data dan waktu respons pemilihan jalur pengatur penggunaan jalur terhadap perubahan *demand*. Distribusi trafik jaringan dilihat dari rata-rata *throughput* dari port keluaran, sedangkan waktu respons dilihat dari waktu saat terjadi peningkatan *throughput* dalam pengiriman data. Pengujian dari sistem dilakukan dengan lingkungan uji SDN yang dibuat dengan emulasikan dengan mininet.

## Rumusan Masalah

Berdasarkan penjabaran latar belakang tersebut, dapat dirumuskan permasalahan sebagai berikut.

1. Bagaimana implementasi sistem *multipath routing* yang membagi trafik jaringan secara proporsional sesuai ketersedian *bandwidth* jalur?
2. Bagaimana sistem *multipath routing* membagi trafik jaringan yang proporsional?
3. Bagaimana kinerja sistem *multipath routing* dengan penggunaan jalur yang diatur?

## Tujuan

Pembagian trafik jaringan secara rata oleh *multipath routing* pada jaringan dengan *bandwidth* jalur yang berbeda-beda dapat menyebabkan *bottleneck* saat proses pengiriman data. *Bottleneck* dapat terjadi karena jalur dengan *bandwidth* yang kecil mendapatkan beban trafik jaringan yang lebih besar daripada *bandwidth* yang dimiliki. Oleh karena itu, penelitian ini bertujuan membagi beban trafik jaringan pada *multipath routing* secara proporsional sesuai dengan kebutuhan *bandwidth* pengguna dan ketersediaan *bandwidth* dari jalur.

## Manfaat

Penelitian ini menghasilkan implementasi *multipath routing* dengan penggunaan jalur sesuai dengan *demand* dan *bandwidth* jalur. Harapannya, penelitian ini dapat menghasilkan alternatif solusi untuk pemilihan jalur di SDN *multipath routing*. Bagi peneliti lain, diharapkan penelitian ini dapat menjadi gambaran tentang *multipath routing* pada SDN dan menunjukkan salah satu cara optimasi yang dapat dilakukan terhadap skema *multipath routing* yang sudah ada.

## Batasan Masalah

Batasan masalah dari penelitian ini adalah :

1. Penelitian dilakukan dengan emulasi jaringan SDN dengan mininet versi 2.2.2.
2. Framework *controller* SDN yang digunakan adalah Ryu versi 4.15.
3. *Tool* yang digunakan untuk memantau jaringan adalah SFlow-rt manager.
4. Pengiriman data di jaringan SDN dilakukan dengan iperf3 berupa pengiriman data TCP.
5. Parameter yang digunakan untuk menguji kinerja adalah penggunaan jalur dalam pengiriman data dan waktu respons pemilihan jalur pengatur penggunaan jalur terhadap perubahan *demand*.

## Sistematika Pembahasan

Sistematika pembahasan berisi rangkuman dari penelitian yang disusun dalam 6 bab pembahasan. Sistematika pembahasan berfungsi sebagai acuan dalam rangka penyusunan penelitian secara sistematis. Sistematika pembahasan dalam penelitian terdiri dari bab 1 pendahuluan, bab 2 landasan kepustakaan, bab 3 metodologi, bab 4 implementasi, bab 5 pengujian dan pembahasan dan bab 6 kesimpulan.

Pada bab pendahuluan dibahas mengenai latar belakang pemilihan topik skripsi. Kesimpulan dari masalah utama yang muncul dirumuskan menjadi beberapa pertanyaan untuk menyelsasi masalah utama. Selain latar belakang, bab pendahuluan membahas tujuan dari penelitian, manfaat dari penelitian, batasan dari masalah penelitian dan sistematika pembahasan.

Pada bab landasan kepustakaan dijelaskan mengenai kajian pustaka dan dasar teori yang berhubungan dengan permasalahan yang dirumuskan sebagai acuan dalam membahas permasalahan yang dihadapi. Kajian pustaka menjelaskan referensi penelitian yang membahas masalah yang berkaitan dengan penelitian ini. Dasar teori menjelaskan teori dasar yang digunakan untuk menyelsaikan masalah.

Pada bab metodologi penelitian dibahas mengenai langkah-langkah sistematis yang dilakukan dalam melakukan penelitian. Langkah-langkah sistematis dalam metodologi digunakan sebagai landasan untuk melakukan penelitian. Langkah-langkah sistematis terwujudkan dalam bentuk rancangan dari penelitian.

Pada bab implementasi dijelaskan langkah-langkah yang diambil dalam penelitian untuk mewujudkan rancangan penelitian. Implementasi yang dilakukan mencakup implementasi dari lingkungan uji, implementasi dari sistem *multipath routing* dan implementasi dari pengukuran kinerja.

Pada bab pembahasan hasil pengujian ditunjukkan hasil pengujian terhadap sistem yang telah diimplementasikan. Kemudian, dilakukan pembahasan terhadap hasil dari pengujian yang didapatkan.

Pada bab kesimpulan dijelaskan tentang kesimpulan penelitian yang telah dilakukan. Kesimpulan yang dijelaskan didapat dari seluruh proses penelitian. Bab ini menjawab rumusan masalah yang telah dirumuskan pada di bab pendahuluan.

# Landasan Kepustakaan

Landasan kepustakaan berisi tentang studi literatur yang telah dilakukan untuk mendukung penelitian ini. Landasan pustaka berisi subbab kajian pustaka dan subbab dasar teori yang digunakan dalam penelitian ini. Kajian pustaka berisi penelitian lain yang membahas tentang *multipath* routing, sedangkan dasar teori berisi dasar-dasar teori sebagai acuan teoritis dalam mengembangkan penelitian.

## Kajian Pustaka

Kajian pustaka membahas beberapa penelitian sebelumnya yang memiliki keterkaitan dengan penelitian yang akan dilakukan. Beberapa penelitian ini berisi teori dan metode yang digunakan untuk mendukung studi kasus dengan objek SDN *multipath routing*. Penelitian pertama membahas tentang skema *multipath routing*. Penelitian kedua membahas tentang implementasi *multipath routing* dengan mekanisme load balancer. Penelitian ketiga membahas tentang implementasi *multipath routing* yang berbeda dengan penelitian pertama dan kedua.

Penelitian pertama berjudul “*A Multipath Transmission Scheme for the Improvement of Throughput over* SDN” (Chiang, et. al., 2017). Penelitian ini membahas skema *multipath routing* yang diimplementasikan dengan SDN. Tujuan dari penelitian ini adalah peningkatan kinerja jaringan dalam melakukan pengiriman data dengan menghindari *congestion* di sebuah jalur.

Penelitian Chiang melakukan pengujian dengan simulasi jaringan SDN pada mininet dan iperf3 untuk melakukan pengiriman data. Topologi yang digunakan dalam pengujian menyediakan tiga *host* dan 3 jalur antara *source* dan *destination*, dengan jalur terbaik adalah jalur dengan jumlah hop paling kecil. *Host* pertama berfungsi sebagai *source* dari pengiriman data, *host* kedua berfungsi sebagai *destination*, dan *host* ketiga berfungsi untuk menciptakan *congestion* dengan mengirimkan data ke *destination* melalui jalur terbaik. Pengujian dimulai dengan *host* ketiga mengirimkan data ke *destination* untuk membuat kondisi *congestion* di jalur terbaik. Kemudian dilakukan pengiriman data dari *source* menuju ke *destination* dengan skema *multipath routing* yang menggunakan semua jalur diantara *source* dan *destination*. Metrik yang digunakan oleh Chiang untuk mengukur kinerja skema *multipath routing* adalah *throughput* dari pengiriman data. Kesimpulan dari penelitian didapatkan dengan membandingkan hasil *throughput* *multipath routing* dengan hasil *throughput* *singlepath routing* untuk.

Hasil pengujian yang didapatkan adalah *throughput* dari jaringan dengan skema *multipath routing* lebih tinggi daripada *throughput* dari jaringan dengan skema *singlepath routing*. Jaringan dengan skema *multipath routing* memiliki *throughput* sebesar 35 Mbits/s, sedangkan jaringan dengan skema *singlepath routing* memiliki *throughput* sebesar 6 atau 7 Mbits/s. Hasil ini menunjukkan skema *multipath routing* berhasil meningkatkan kemampuan jaringan dalam melakukan pengiriman data.

Menurut kesimpulan pada penelitian Chiang, perlu dilakukan penelitian lebih lanjut mengenai pembagian trafik jaringan dari pengiriman data pada *multipath routing*. Hal ini didukung oleh hasil pengujian yang menunjukkan sebagian trafik jaringan masih dilewatkan jalur yang terjadi *congestion*. Oleh karena itu penelitian yang akan dilakukan penulis berfokus terhadap pembagian trafik jaringan dengan memanfaatkan skema *multipath routing* dari penelitian Chiang.

Penelitian kedua berjudul “*Multipath routing* dengan Load-Balancing pada OpenFlow Software-Defined Network” (Wildan, et. al., 2017). Pada penelitian Wildan dibahas masalah pemborosan sumber daya jaringan yang terjadi pada jaringan yang memanfaatkan *routing* *shortest* *path* (*singlepath routing*) dengan topologi yang memiliki lebih dari satu jalur antara sumber dan tujuan. Solusi yang diajukan pada penelitian ini adalah dengan menggunakan *multipath routing* dengan load balancer untuk membagi trafik jaringan secara seimbang sehingga sumber daya jaringan dapat dimanfaatkan secara penuh.

Penelitian Wildan melakukan implementasi sistem *multipath routing* dengan algoritme deep-first search (DFS) sebagai pencarian jalur dan bobot jalur (*bucket* weight) sebagai load balancer. Hasil dari pencarian jalur adalah jalur-jalur yang tidak independen. Bobot dari jalur pada penelitian Wildan bernilai 1 dikurangi nilai sebuah jalur dibagi nilai total semua jalur. Nilai dari sebuah jalur dihitung bedasarkan *bandwidth* referensi OSPF yang bernilai 100 Mb, *bandwidth* jalur dan jumlah dari hop antara sumber dan tujuan. Nilai jalur yang besar menunjukkan jalur tersebut adalah jalur yang tidak efisien dan memiliki bobot jalur yang kecil sehingga frekuensi penggunaan jalur tersebut lebih kecil. Wildan menguji sistem *multipath routing* dengan melakukan simulasi jaringan SDN pada mininet dan *tools* iperf untuk melakukan pengiriman data. Sistem *multipath routing* diuji pada beberapa topologi yang memiliki nilai jalur atau jumlah jalur yang berbeda-beda. Metrik pengujian yang digunakan oleh Wildan untuk menguji sistem *multipath routing* adalah *response time*, *throughput*, delay dan rasio transmisi.

Bedasarkan metrik pengujian yang didefinisikan, sistem *multipath routing* menghasilkan *response time* terbaik (153.70 ms) saat jumlah jalur yang ditemukan berjumlah dan *response time* terburuk (tidak terkirim) jumlah jalur yang ditemukan berjumlah 32. Pada parameter *throughput*, sistem menghasilkan rata-rata *throughput* antara 2,93 hingga 3,60 Gbps. Selanjutnya pada parameter delay, sistem menghasilkan rata-rata delay antara 0,104 hingga 0,119 ms. Hasil pengujian parameter rasio transmisi menunjukkan salah satu jalur memiliki frekuensi penggunaan yang lebih tinggi meskipun jalur yang tersedia memiliki jumlah hop dan *bandwidth* yang identik. Hasil penelitian Wildan menunjukkan perlunya pembatasan jumlah jalur yang ditemukan untuk menghindari paket drop karena proses pencarian jalur yang lama dan penggunaan jalur yang masih bisa diperbaiki.

Penelitian Wildan memiliki pengaruh terhadap penelitian yang akan dilakukan penulis. Pada penelitian yang akan dilakukan penulis menggunakan proses pencarian jalur yang menghasilkan jalur independen. Jalur independen digunakan untuk mengurangi jumlah jalur yang ditemukan sehingga paket tidak didrop karena proses pencarian jalur yang mencari terlalu banyak jalur. Selain jalur inpenden, penelitian yang akan dilakukan penulis menggunakan cara menghitung bobot yang bedasarkan *demand* dan ketersedian *bandwidth* jalur untuk menghasilkan pembagian trafik jaringan yang lebih proporsional.

Penelitian terakhir berjudul “A Joint Approach to *Multipath routing* and *Rate* *Adaptation* for *Congestion* *Control* in OpenFlow Software Defined Network” (Hertiana & Hendrawan, 2015). Masalah pada penelitian Hertiana adalah *congestion* pada jaringan. Pada penelitian Hertiana solusi yang diberikan adalah penggunaan *multipath routing* dengan *rate* *adaptation*.

Penelitian Hertiana menggunakan memberikan nilai batasan (threshold) sebesar 80% terhadap nilai penggunaan jalur untuk mengimplementasikan *multipath routing*. Threshold 80% menunjukkan terjadinya *congestion* di jalur yang digunakan. Saat terjadi *congestion* di jalur yang digunakan, sistem mengarahkan trafik jaringan ke jalur lain. Penelitian Hertiana juga menggunakan rate adaptation yang mengatur volume transmisi data yang dilakukan sesuai dengan kondisi jaringan. Parameter uji yang digunakan pada penelitian Hertiana untuk menguji sistem adalah *throughput, delay* dan *packet loss* dari transmisi data. Pengujian sistem dilakukan di lingkungan uji yang diemulasikan dengan mininet.

Hasil penelitian Hertiana menunjukkan *multipath routing* dengan rate adaptation memiliki hasil *throughput* 37.9 Kbps*, delay* dibawah 0.02 s dan 0 *packet loss.* *Multipath routing* tanpa rate adaptation memiliki hasil *throughput* 37.4 Kbps*, delay* dibawah 0.04 s dan 0 *packet loss. Singlepath routing* memiliki hasil *throughput* 37.4 Kbps*, delay* dibawah 0.04 s dan 0 *packet loss.*

Kelebihan dari penelitian Hertiana adalah penggunaan *threshold* pada nilai penggunaan jalur yang mengatur penggunaan *multipath routing*. Dengan *threshold*, jaringan akan mengirimkan trafik jaringan secara *multipath routing* jika nilai penggunaan jalur mencapai *threshold* yang ditentukan. Penelitian yang akan dilakukan penulis menggunakan penelitian Hertiana sebagai referensi untuk penggunaan threshold sebagai aturan penggunaan *multipath routing* sehingga tidak semua trafik jaringan akan dipecah dan dikirimkan secara *multipath routing*.

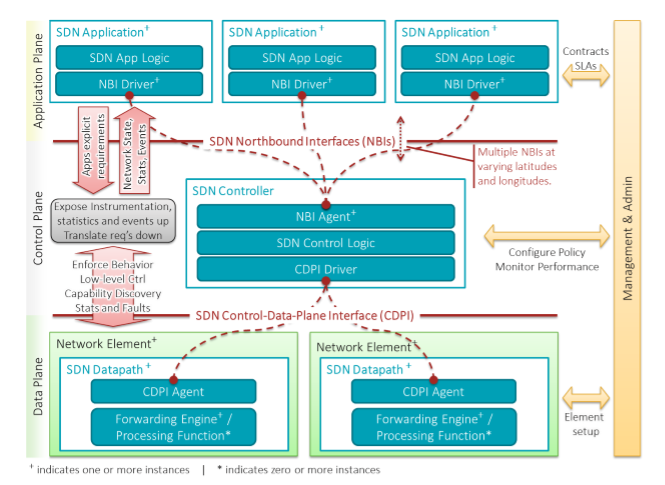
Dari penelitian yang sudah dikaji, penelitian ini akan mengimplementasikan *multipath routing* dengan penggunaan jalur yang proporsional dengan trafik jaringan dari pengguna dan ketersediaan *bandwidth* dari jalur. Trafik jaringan dari pengguna didapatkan dengan memantau jaringan menggunakan *tool* SFlow-rt. Dengan informasi trafik jaringan, *controller* akan menentukan penggunaan jalur.

## Dasar Teori

Sub bab dasar teori menjelaskan tentang teori yang digunakan sebagai landasan dari penelitian yang akan dilakukan. Dasar teori yang digunakan meliputi SDN, OpenFlow, Openvswitch, *multipath routing* dan perangkat pendukung. Sub bab SDN menjelaskan struktur SDN. Sub bab Openflow menjelaskan secara umum cara kerja protokol Openflow. Sub bab OpenvSwitch menjelaskan mekanisme pembagian trafik jaringan yang digunakan dalam penelitian ini. Sub bab *multipath* routing menjelaskan teori *multipath* routing dan algoritme pencarian jalur yang digunakan. Sub bab perangkat pendukung menjelaskan perangkat pendukung yang digunakan dalam penelitian ini.

### *Software* Defined *Networking* (SDN)

SDN adalah paradigma *network*ing yang memisahkan bagian *perangkat keras* untuk penerusan (*data plane*) dan bagian yang mengatur keputusan *(control plane*). SDN menjanjikan management jaringan yang jauh lebih sederhana, inovasi dan evolusi. Ide utama dari SDN adalah membuat *developer* perangkat lunak dapat memanfaatkan *resource* pada jaringan semudah *resource* pada komputer dan *storage* (Nunes, et al., 2014). Pada SDN kecerdasan dari jaringan (aturan penerusan) ada pada *controller* berupa perangkat lunak dan perangkat jaringan hanya berfungsi untuk melakukan penerusan paket. *Protocol* yang dapat digunakan untuk komunikasi antara *controller* dan perangkat penerusan adalah OpenFlow.



Gambar 2. 1 **Gambaran Arsitektur dari SDN**

(Sumber: *Open Networking Foundation*)

Gambar 2.1 menunjukkan gambaran umum dari arsitektur SDN. Pada gambar ditunjukkan terdapat tiga lapisan pada SDN. Lapisan paling atas adalah *application plane* yang berisi SDN *application* berupa logika dari aplikasi SDN dan *NorthBound Interface* (NBI) *drivers* yang mengirimkan logika ke *control plane*. Lapisan tengah merupakan *control plane* yang berisi SDN *controller* yang menerjemahkan logika dari aplikasi SDN dan mengatur jalur pada SDN menggunakan *Control-*Data*-Plane Interface* (CDPI) driver. Lapisan paling bawah adalah *data plane* yang berisi elemen dari jaringan (*Open Networking Foundation*, 2013).

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

1. *Control* dan *data plane* yang terpisah. Fungsionalitas *control plane* dihilangkan dari perangkat jaringan yang berfungsi untuk meneruskan paket.
2. Penerusan paket dilakukan berdasarkan *flow*, bukan berdasarkan *destination* atau tujuan paket. *Flow* secara luas didefinisikan dengan kumpulan nilai paket sebagai suatu kriteria (filter) dan kumpulan aksi (intruksi). Pada konteks SDN/OpenFlow *flow* adalah rentetan paket dari sumber ke tujuan. Paket pada *flow* yang sama diteruskan dengan aturan yang sama oleh *data plane*.
3. Logika kontrol dipindahkan pada suatu entitas terpisah, yang disebut *Controller* SDN atau NOS (Network Operating System). NOS adalah sebuah platform perangkat lunak 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 perangkat lunak yang berjalan di atas NOS dan dapat berinteraksi dengan perangkat-perangkat *data plane* yang dicakupinya.

### OpenFlow

OpenFlow merupakan *protocol* komunikasi standar yang digunakan di jaringan SDN untuk komunikasi antar perangkat pada *control plane* dan pada *data plane* (*Open* Networking Foundation, 2016). Perangkat *control plane* pada OpenFlow dapat berupa *controller* yang menyediakan logika atau mengatur *data plane*., sedangkan perangkat pada *data plane* adalah switch yang hanya melakukan fungsi penerusan paket. Penerusan paket dilakukan berdasarkan suatu set kriteria yang menyeleksi paket dan suatu set tindakan atau *action* yang diterapkan pada paket tersebut. Satuan dari set kriteria beserta tindakannya adalah yang disebut sebagai *flow* (Kreutz, et al., 2015).

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

### *OpenvSwitch*

OpenvSwitch adalah perangkat lunak *multilayer switch* yang dilisensi dibawah *open source* Apache 2.0. OpenvSwitch didesain untuk mendukung manajemen dari antarmuka dan memungkinkan pemrograman fungsi penerusan paket (Linux Foundation, 2016). OpenvSwitch versi terakhir mendukung protocol OpenFlow yang digunakan untuk komunikasi antar *switch* dan *controller*. Dengan menggunakan OpenFlow*, controller* dapat menambahkan, memodifikasidan menghapus isi dari *flow* *table* dan *group table* secara *reactive* maupun *proactive* (*Open Networking Foundation*, 2014).

*Flow* *table* di *switch* berisi kumpulan *flow entry*. Pada setiap *flow entry* terdapat *match fields* dan intruksi. *Match fields* berisi kriteria yang akan dicocokkan dengan *flow* yang masuk. Instruksi berisi aksi yang akan dilakukan atau *group entry* yang ada pada *group table*.

*Group table* di *switch* berisi kumpulan *group entry*. Pada setiap *group entry* terdapat *action* *buckets* dan tipe dari *group entry*. *Action* *buckets* berisi satu atau lebih *bucket* (kumpulan aksi) yang dapat digunakan untuk mengimplementasikan penerusan paket yang kompleks. Tipe dari *group entry* menentukan bagaimana *bucket* didalam *actions* *buckets* digunakan. Pada tabel 2.2 dijelaskan tipe dari *group entry*.

Tabel 2. 2 Tabel Tipe *Group entry*

|  |  |  |
| --- | --- | --- |
| **No.** | **Tipe** | **Keterangan** |
| 1. | *Indirect* | *Bucket* yang dieksekusi hanya satu *bucket* yang sudah didefinisikan. |
| 2. | *All* | Semua *bucket* pada *action* *buckets* dieksekusi. Digunakan untuk *multi*cast atau broadcast penerusan. |
| 3 | *Select* | *Bucket* yang dieksekusi hanya satu. *Bucket* yang dieksekusi dipilih dengan algoritme dari *switch* diluar OpenFlow *protocol*. Algoritme pemilihan dapat dipengaruhi oleh *bucket weight*. |
| 4 | *Fast failover* | *Bucket* yang dieksekusi adalah *bucket* yang pertama kali berstatus hidup. |

Tipe *group entry* yang digunakan dalam *multipath routing* adalah *select*. Pada *group entry* dengan tipe *select* jalur yang akan digunakan dipilih bedasarkan *bucket* *weight*. Jalur yang memiliki *bucket* *weight* lebih besar akan lebih sering dipilih.

### *Multipath routing*

*Multipath routing* adalah teknik *routing* yang menggunakan jalur lebih dari satu untuk melakukan pengiriman data. Menurut (Lee & Choi, 2002), *multipath routing* berpotensi memiliki tingkat tranmisi datayang lebih tinggi dari pada *singlepath routing*. Dengan menggunakan *multipath routing*, trafik jaringan yang tinggi pada jaringan dibagi sesuai jalur-jalur yang ditemukan oleh algoritme sehingga tidak terjadi *bottleneck* disatu jalur.

Dalam implementasi *multipath routing* pada SDN, skema pengiriman paket memiliki tiga proses berurutan. Pertama, *controller* SDN mempelajari topologi dari jaringan. Kedua, pencarian jalur-jalur antara 2 *node* dalam topologi. Ketiga, *controller* SDN mengatur switch untuk membagi trafik jaringan sesuai dengan jalur-jalur yang ditemukan (Chiang, et al., 2017). Salah satu cara *controller* SDN mengatur pembagian trafik jaringan adalah dengan memanfaatkan fitur *group* *table* pada OpenFlow switch.

Salah satu algoritme yang digunakan untuk pencarian jalur dalam *multipath* routing adalah Dijkstra. Algoritma Dijkstra menggunakan st*rate*gi *greedy*, di setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah dipilih dengan simpul lain yang belum dipilih sehingga menghasilkan satu jalur terpendek. Dalam Dijkstra, bobot bersifat tidak negatif.

Algoritme Dijkstra menghasilkan satu jalur terpendek. Pada *multipath routing* dibutuhkan lebih dari 1 jalur. Untuk menghasilkan lebih dari satu jalur sistem melakukan perulangan algoritme Dijkstra. Dalam setiap perulangan, *node* pada *graph* pencarian dikurangi dengan *node* hasil dari jalur yang sudah ditemukan sebelumnya sehingga jalur terpendek yang ditemukan berbeda dengan jalur yang sudah ditemukan.

### Perangkat Pendukung

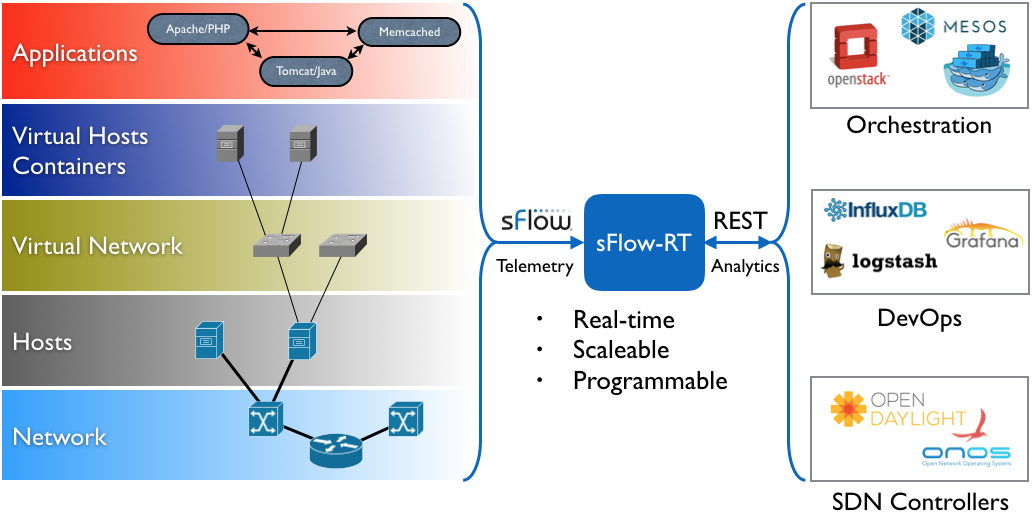
Perangkat pendukung dalam penelitian ini digunakan untuk mengemulasikan lingkungan uji yang mendukung implementasi sistem *multipath routing* dan melakukan pemantauan terhadap kondisi jaringan. Emulasi lingkungan uji dilakukan dengan mininet dan Ryu. Pemantauan kondisi jaringan dilakukan dengan SFlow-rt.

Mininet adalah sebuah emulator jaringan yang membangun jaringan yang terdiri atas *host* *virtual*, switch, *controller* dan *link* (Mininet Team, 2018). 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.

Penelitian ini menggunakan mininet untuk mengemulasikan lingkungan uji berupa jaringan SDN. Topologi jaringan dibuat dan dikonfigurasi menggunakan fitur miniedit. Topologi jaringan terdiri dari virtual switch, virtual *host* dan controller. Selain topologi jaringan, fitur miniedit juga digunakan untuk mengkonfigurasi interval SFlow dari switch.

Ryu merupakan framework berbasis komponen untuk *controller* SDN. Ryu menyediakan komponen perangkat lunak dengan API yang memudahkan developer untuk membangun aplikasi-aplikasi kontrol dan manajemen jaringan. Ryu mendukung OpenFlow versi 1.0 hingga versi 1.5. Untuk mengembangkan aplikasi *multipath routing* dibutuhkan *group* *action*s yang tersedia pada OpenFlow versi 1.1 ke atas. Ryu dibuat dengan bahasa pemrograman python sehingga pengembangan aplikasi untuk Ryu dapat dilakukan dengan bahasa pemrograman python.

SFlow-rt merupakan sebuah *tool* untuk me*monitor* jaringan secara real-time. SFlow-rt menggunakan teknologi analitik asinkron dari InMon. SFlow-rt dapat digunakan untuk mendapatkan datatopologi jaringan (contohnya *incoming* trafik jaringan dan *outgoing* trafik jaringan). Datahasil monitoring dapat digunakan untuk mendukung aplikasi *controller* SDN.



Gambar 2. 2 SFlow-rt

(Sumber: InMon)

Gambar 2.2 menunjukkan bagaimana SFlow-rt bekerja secara umum. SFlow-rt akan menerima aliran telemetri secara kontinu dari agen SFlow yang terpasang pada perangkat jaringan, *host*, dan aplikasi. Interval aliran telemetri dari agen SFlow ditentukan oleh nilai *polling* yang dapat dikonfigurasi di miniedit. Kemudian, telemetri dari agen SFlow dirubah menjadi metrik yang dapat diakses pada sebuah REST API. REST API tersebut memudahkan *controller* SDN untuk mendapatkan datatopologi. Salah satu datatopologi yang didapatkan adalah *incoming* trafik jaringan*, outgoing* trafik jaringan dan *bandwidth*. Data *incoming* trafik jaringan *dan outgoing* trafik jaringan dapat digunakan sebagai parameter *demand* dari *host* dan penggunaan *bandwidth* jalur, sedangkan data *bandwidth* dapat digunakan sebagai *cost* pada pencarian jalur atau parameter yang dipertimbangkan dalam pemilihan jalur.

# Metodologi Penelitian

Penelitian ini merupakan tipe penelitian implementatif karena berupa implementasi sistem yang mengatur penggunaan jalur pada *multipath routing*. Tahapan pertama adalah analisis kebutuhan berupa penjelasan *tools*, parameter dan nilai-nilai yang dibutuhkan untuk melakukan penelitian. Tahapan kedua adalah perancangan sistem untuk menyelsaikan masalah dari penelitian. Kemudian, peneliti mengimplementasikan sistem sesuai hasil rancangan. Setelah dilakukan implementasi, hasil implementasi akan diuji dan dibahas pada tahap pengujian dan pembahasan. Jika seluruh tahapan sudah sesuai, maka ditarik kesimpulan hasil penguijan yang didapatkan. Diagram alur tahapan proses pengerjaan penelitian ini ditunjukkan pada Gambar 3.1.

Gambar 3. 1 Diagram Alur Tahapan Penelitian

## Analisis Kebutuhan

Pada bagian ini dilakukan analisis terhadap kebutuhan untuk merancang dan mengimplementasikan *multipath routing* yang penggunaan jalurnya proporsional dengan *demand* dan ketersediaan *bandwidth*. Penelitian ini membutuhkan perangkat untuk mengemulasikan lingkungan uji yang mendukung implementasi sistem *multipath routing* dan melakukan pemantauan terhadap kondisi jaringan. Untuk memenuhi kebutuhan tersebut dibutuhkan beberapa perangkat lunak, yaitu Mininet, Ryu dan SFlow-rt.

Mininet adalah sebuah emulator jaringan yang membangun jaringan yang terdiri atas *host* *virtual*, switch, *controller* dan *link* (Mininet Team, 2018). 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 digunakan karena mendukung fitur emulasi topologi dengan parameter yang ditentukan user.

Topologi dalam penelitian ini merupakan topologi yang dapat mendukung *multipath routing*. Topologi memiliki lebih dari satu jalur diantara *source* dan *destination*. Selain itu, topologi dibuat bedasarkan pengujian distribusi trafik jaringan. Pengujian pertama adalah pengiriman data dengan *demand* yang rendah (bisa dilayani dengan 1 jalur). Pengujian kedua adalah pengiriman data dengan yang tinggi (tidak bisa dilayani dengan 1 jalur). Pengujian ketiga adalah dua pengiriman data dengan *demand* yang rendah yang dilakukan secara bersamaan.

*Controller* SDN yang digunakan untuk Implementasi sistem *multipath routing* pada penelitian ini adalah Ryu *controller*. Ryu merupakan open-*source* SDN *controller* yang mendukung OpenFlow versi 1.1 ke atas dan memiliki library yang mempermudah implementasi aplikasi yang mengatur trafik jaringan jaringan. OpenFlow versi 1.1 ke atas dibutuhkan untuk fitur *group* table yang digunakan pada *multipath routing* sedangkan library dari Ryu dibutuhkan agar sistem *multipath routing* mengetahui topologi dari jaringan.

Pada penelitian ini, implementasi sistem *multipath routing* membutuhkan mekanisme pemantauan trafik jaringan untuk memantau *demand* dan ketersedian *bandwidth* dari jalur. Mekanisme pemantauan trafik jaringan diimplementasikan dengan *tools* SFlow-rt. Dengan SFlow-rt, mekanisme pemantauan trafik jaringan dapat meminta nilai *demand* dan penggunaan *bandwidth* jalur. Mekanisme pemantauan trafik jaringan meminta nilai *demand* dengan meminta nilai inOctets dari port yang terhubung dengan *host*. Untuk ketersedian *bandwidth* jalur, mekanisme pemantauan trafik jaringan meminta nilai outIfUtilization dari port yang ada pada sebuah jalur. Nilai outIfUtilization menunjukkan persentase penggunaan *link* *bandwidth* yang terhubung dengan port sehingga nilai *bandwidth* yang digunakan dapat diketahui. Nilai ketersediaan *bandwidth* adalah nilai *bandwidth* dari *link* dikurangi nilai *bandwidth* yang digunakan. Hasil dari pemantauan trafik jaringan digunakan pada mekanisme pemilihan jalur dari sistem *multipath routing*.

Pada penelitian ini semua pengujian dilakukan dengan menggunakan *tools* iperf3. *Tools* iperf3 digunakan untuk mengirimkan data antara dua *host*. Pengiriman data dilakukan dengan jumlah koneksi 10 dan *bitrate* yang dibatasi sesuai skenario pengujian yang dibuat. Jumlah koneksi dibuat bernilai 10 agar trafik jaringan dikirimkan dengan tcp *header* yang berbeda sehingga dapat dikirim secara *multipath* routing. Pengukuran dari pengiriman data dilakukan selama 60 detik. Waktu pengukuran dari pengiriman data bernilai 60 detik untuk menghindari file hasil pengukuran yang besar dan memberikan waktu kepada mekanisme untuk merubah pengiriman data sesuai trafik jaringan dari *host*.

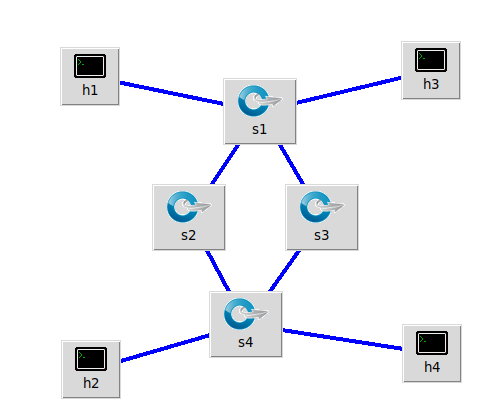
Parameter yang digunakan untuk mengukur kinerja dari sistem *multipath routing* adalah distribusi trafik jaringan dan waktu respons dari mekanisme pengatur penggunaan jalur. Pengujian kedua parameter dilakukan dengan skenario yang berbeda. Pengukuran parameter distribusi trafik jaringan dilihat dari rata-rata *throughput* dari port keluaran dengan BWM-NG, sedangkan waktu respons dilihat dari waktu peningkatan *throughput* saat pengiriman data dilakukan. Penggunaan jalur bernilai baik apabila rasio dari *throughput* mendekati rasio nilai jalur. Waktu respons semakin baik apabila nilai waktu respons semakin kecil.

## Perancangan

Setelah tahap analisis kebutuhan, tahapan selanjutnya adalah merancang sistem untuk memberikan gambaran mengenai alur kerja sistem. Hasil dari perancangan berguna untuk pedoman dari proses implementasi. Perancangan yang dilakukan dalam penelitian ini adalah perancangan lingkungan uji, perancangan *multipath routing* dan perancangan pengujian kinerja.

### Perancangan Lingkungan Uji

Perancangan lingkungan uji merupakan proses untuk merancang topologi jaringan yang akan digunakan dalam pengujian. Topologi yang akan digunakan adalah topologi mesh dengan kapasitas *link* berbeda. Topologi akan mengakomodasi simulasi tiga skenario *demand* dari *host* di jaringan. Skenario pertama adalah pengiriman data dengan *demand* yang rendah (bisa dilayani dengan 1 jalur). Skenario kedua adalah pengiriman data dengan yang tinggi (tidak bisa dilayani dengan 1 jalur). Skenario ketiga adalah dua pengiriman data dengan *demand* yang rendah yang dilakukan secara bersamaan. Gambar 3.2 menunjukkan rancangan dari topologi.



100 Mbits

100 Mbits

100 Mbits

100 Mbits

70 Mbits

30 Mbits

30 Mbits

70 Mbits

Gambar 3. 2 Rancangan Topologi

Topologi memiliki 4 *host* (h1 sampai dengan h4). *Link* pada topologi memiliki *bandwidth* yang tidak seragam. Semua *host* memiliki kapasitas *link* yang sama, yaitu 100 Mbits. Jalur s1-s2-s4 memilki *bandwidth* 70 Mbits dan jalur s1-s3-s4 memiliki *bandwidth* 30 Mbits. Untuk melakukan pengujian sesuai skenario pertama, pengiriman data dilakukan dari h1 menuju h2 dengan batas *bitrate* sebesar 10 Mbits. Pada pengujian sesuai skenario kedua h1 mengirimkan data ke h2 dengan batas *bitrate* sebesar 100 Mbits. Untuk skenario ketiga, h1 mengirimkan data ke h2 dengan batas *bitrate* sebesar 50 Mbits bersamaan dengan h3 mengirimkan data ke h4 dengan batas *bitrate* sebesar 10 Mbits.

Lingkungan uji direalisasikan dengan membuat topologi jaringan SDN di mininet. Implementasi lingkungan uji dilakukan untuk melakukan emulasi jaringan SDN dan pengujian dari penelitian. Topologi jaringan SDN yang diemulasikan dibuat sesuai dengan hasil perancangan topologi.

Pembuatan topologi dilakukan dengan menggunakan fitur miniedit. Konfigurasi yang digunakan adalah switch virtual OpenvSwitch dengan protocol OpenFlow 1.3 dan controller berupa *remote controller* dengan alamat local*host*:6633. Perangkat lunak yang dijalankan sebagai controller SDN adalah Ryu dengan program yang mengimplementasikan *multipath* routing dengan penggunaan jalur yang diatur.

### Perancangan *Multipath routing*

Perancangan *multipath routing* merupakan proses untuk merancang sistem *multipath routing* yang akan digunakan. Bedasarkan penelitian Chiang, skema *multipath routing* terdiri dari 3 proses (Chiang, et. al., 2017). Proses pertama adalah *controller* SDN mempelajari topologi dari jaringan (topologi learning). Proses kedua adalah pencarian jalur-jalur antara 2 *node* dalam topologi. Ketiga, *controller* SDN mengatur switch untuk membagi trafik jaringan sesuai dengan jalur-jalur yang ditemukan. Penelitian ini berfokus pada tahap ketiga dari skema *multipath routing* Chiang.

Perancangan mekanisme topologi learning

Proses topologi learning adalah proses controller SDN mendapatkan informasi topologi dari jaringan. Proses topologi learning dilakukan dengan menggunakan option yang dimiliki Ryu. *Option* yang digunakan adalah observe-*link*. Dengan *option* observe-*link*, sistem *multipath routing* dapat mengetahui topologi dari jaringan saat sistem mulai berjalan.

Perancangan mekanisme pencarian jalur

Proses pencarian jalur pada penelitian ini dibuat bedasarkan algoritme Dijkstra pada penelitian Chiang. Pada penelitian Chiang algoritme Dijkstra diulang untuk menghasilkan lebih dari satu jalur diantara 2 *node*. Algoritme ini digunakan karena menurut hasil penelitian Wirawan memiliki waktu eksekusi yang rendah dan hasil pencarian jalur merupakan jalur independen (Wirawan, et. al., 2018).

Untuk melakukan pencarian jalur, algoritme Dijkstra membutuhkan *cost* dari setiap traversal. Salah satu parameter yang dapat digunakan untuk menentukan *cost* adalah nilai *link* dari jalur. Pada penelitian ini nilai *link* dari jalur dihitung secara OSPF. Rumus dari penilaian jalur secara OSPF ditunjukkan oleh Rumus 3.1.

(3.1)

*Edge* *weight* (*link* *cost*) disimbolkan dengan . menyatakan reference *bandwidth* yang digunakan protokol OSPF, yaitu sebesar 100 Mbps (Cisco, 2005). Sementara menunjukkan *bandwidth* dari *link*. *Cost* dihitung dengan menjumlahkan seluruh edge *weight* pada suatu jalur. Jalur yang terbaik merupakan jalur dengan *cost* terkecil.

Mekanisme pencarian jalur dalam penelitian ini direalisasikan bedasarkan pencarian jalur pada penelitian Chiang (Chiang, et. al., 2017). Pada penelitian Chiang pencarian jalur dilakukan dengan algoritme Dijkstra yang diulang. Dalam setiap perulangan algoritme Dijkstra, *node* pada *graph* pencarian dikurangi *node* jalur yang sudah ditemukan sehingga pencarian jalur akan selalu menghasilkan jalur lain yang independen.

Perancangan mekanisme pembagian trafik jaringan

Proses pembagian trafik jaringan dalam penelitian ini dilakukan dengan 2 mekanisme, yaitu mekanisme pemantauan trafik jaringan dan mekanisme pemilihan jalur. Mekanisme pemantauan trafik jaringan berfungsi untuk mendapatkan nilai *demand* dan *bandwidth* yang sedang digunakan di sebuah jalur sedangkan mekanisme pemilihan jalur berfungsi untuk mengatur jalur yang digunakan untuk pengiriman data. Mekanisme pemilihan jalur bekerja sesuai data dari mekanisme pemantauan trafik jaringan.

Mekanisme pemantauan trafik jaringan diimplementasikan dengan bantuan SFlow-rt manager. Mekanisme pemantaunan trafik jaringan melakukan *request* status inOctets ke SFlow-rt manager untuk setiap port dari switch yang terhubung dengan *host* sumber dan *host* tujuan dan nilai outIfUtilization untuk setiap *output* port dari jalur yang digunakan. *Request* dilakukan 3 detik sekali untuk menghindari *request* yang terlalu banyak dari controller ke SFlow-rt. Nilai dari *demand* adalah nilai maksimal inOctets yang didapatkan dari SFlow-rt. Nilai dari *bandwidth* yang sedang digunakan adalah nilai outIfUtilization dari *output* port dikalikan *bandwidth* dari jalur yang melewati *output* port. Nilai *bandwidth* yang sedang digunakan untuk menghitung ketersedian *bandwidth* dari jalur. Nilai ketersediaan *bandwidth* adalah nilai *bandwidth* dari *link* dikurangi nilai *bandwidth* yang sedang digunakan. Nilai dari *demand* dan nilai ketersedian *bandwidth* digunakan dalam mekanisme pemilihan jalur.

Mekanisme pemilihan jalur dilakukan dengan memanfaatkan *group entry* bertipe *select*. Jika tipe *group entry* adalah *select*, maka pemilihan jalur dapat dipengaruhi dengan *bucket* *weight* (bobot jalur). Rumus dari penghitungan *bucket* *weight* ditunjukkan oleh Rumus 3.2.

(3.2)

Rumus 3.2 menunjukkan rumus yang digunakan untuk *bucket* *weight*. Nilai menyatakan bobot jalur. Bobot jalur didapatkan dari nilai minimal dari yang menunjukkan nilai *bandwidth* dari semua *link* pada sebuah jalur. Nilai menunjukkan nilai *bandwidth* yang sedang dipakai. Nilai menunjukkan nilai ketersediaan *bandwidth* jalur. Pada kondisi awal, Nilai adalah 0 sehingga ketersediaan *bandwidth* bernilai sama dengan bobot jalur (). Ketersedian *bandwidth* dari semua jalur akan dibagi dengan nilai total ketersedian *bandwidth* semua jalur untuk mendapatkan nilai rasio dari jalur. Hasil pembagian dikalikan 10 untuk menghindari nilai dibawah 1 sehingga nilai *bucket* *weight* tidak dibulatkan oleh sistem menjadi nilai 0. Jalur yang memiliki *bucket* *weight* yang tinggi merupakan *bucket* yang lebih sering dipilih.

Pada penelitian ini nilai *bucket* *weight* berubah sesuai hasil pemantauan trafik jaringan dari pengguna (*demand*) dan ketersedian *bandwidth* jalur. Perubahan *bucket* weight akan merubah jalur yang digunakan untuk pengiriman data. Aturan yang digunakan untuk merubah *bucket* *weight* adalah sebagai berikut :

* Jika *demand* lebih dari 80% *bandwidth* jalur utama dan pengiriman data dilakukan secara *singlepath routing*, maka *bucket* *weight* dari semua jalur diubah sesuai rumus 3.2 sehingga trafik jaringan dikirim dengan semua jalur yang memiliki *bucket* weight lebih dari 0. Jalur dengan *bucket* weight lebih tinggi akan lebih sering digunakan.
* Jika *demand* kurang dari 80% *bandwidth* jalur utama dan pengiriman dilakukan secara *multipath routing*, *bucket* *weight* dari jalur utama diubah menjadi nilai maksimal dari *bucket* *weight* (65535) dan *bucket* *weight* jalur alternatif diubah menjadi 0 sehingga jalur alternatif tidak dipilih saat pengiriman data.
* Jika *demand* kurang dari 80% *bandwidth* jalur terbaik dan sistem menggunakan 1 jalur untuk pengiriman data, maka sistem akan memilih jalur sesuai nilai tertinggi dari sisa *bandwidth* jalur.
* Jika nilai *demand* kurang dari sisa *bandwidth* jalur, maka *bucket* *weight* dari jalur yang memiliki sisa *bandwidth* tertinggi diubah menjadi nilai maksimal dari *bucket* *weight* (65535) dan *bucket* *weight* jalur lainnya diubah menjadi 0 sehingga hanya jalur dengan sisa *bandwidth* jalur tertinggi yang digunakan untuk pengiriman data.
* Jika nilai *demand* lebih dari sisa *bandwidth* jalur, maka maka *bucket* *weight* dari semua jalur diubah sesuai rumus 3.2 sehingga trafik jaringan dikirim dengan semua jalur yang memiliki *bucket* weight lebih dari 0. Jalur dengan *bucket* weight lebih tinggi akan lebih sering digunakan.

### Perancangan Pengujian Kinerja

Untuk mengetahui kinerja sistem *multipath routing* dengan penggunaan jalur yang disesuaikan dengan kebutuhan pengguna dan ketersediaan *bandwidth* jalur parameter uji yang digunakan adalah penggunaan jalur dan waktu respons pemilihan jalur pengatur penggunaan jalur. Penggunaan jalur digunakan sebagai parameter uji karena menunjukkan pembagian trafik jaringan pada jaringan sedangkan waktu respons pemilihan jalur digunakan sebagai parameter uji karena menunjukkan kecepatan mekanisme beradaptasi terhadap perubahan *demand*.

Pengujian dilakukan dengan melakukan pengiriman data TCP secara parallel antara *host* sumber dan *host* tujuan. Pengiriman data TCP dilakukan secara parallel karena Openvswitch membutuhkan TCP *header* yang berbeda untuk mengirimkan data secara *multipath routing* (Linux Foundation, 2016). Kemudian, pengiriman data akan dipantau dengan BWM-NG.

Pengiriman data TCP dilakukan dengan menggunakan *tool* iperf3. Pengiriman data dilakukan secara parallel dengan option –P. Kemudian pembatasan bandwith dilakukan terhadap setiap aliran TCP parallel untuk memanipulasi *demand* (trafik jaringan dari *host*) dengan option –b. Simulasi pengiriman data dilakukan selama 60 detik.

Pengujian distribusi trafik jaringan dilakukan dengan memonitor rata-rata *throughput* jaringan saat terjadi pengiriman data TCP. Pengujian dilakukan sebanyak 3 kali dengan volume data dari *host* yang berbeda-beda. Pengujian pertama dilakukan dengan volume data yang dapat dilayani dengan 1 jalur. Pengujian kedua dilakukan dengan volume data yang tidak dapat dilayani dengan 1 jalur. Pengujian ketiga dilakukan dengan 2 pengiriman data yang dilakukan secara bersamaan. Hasil rata-rata *throughput* disetiap jalur menunjukkan kinerja distribusi trafik jaringan.

Pengujian waktu respons pemilihan jalur dilakukan dengan melihat waktu saat terjadi peningkatan *throughput* dalam pengiriman data dengan volume yang besar (tidak bisa dilayani dengan 1 jalur). Volume yang besar akan menunjukkan peningkatan *throughput* karena perubahan mekanisme pengiriman data dari *singlepath* ke *multipath*. Selisih waktu peningkatan *throughput* dan waktu volume pertama kali aliran data mencapai volume yang besar adalah nilai dari waktu respons. Pengujian dilakukan sebanyak 3 kali dengan nilai *polling* SFlow-rt yang berbeda-beda. Nilai *polling* menentukan interval switch mengirimkan paket SFlow ke perangkat SFlow-rt. Nilai *polling* yang digunakan adalah 1 detik, 3 detik, dan 10 detik.

## Pengujian dan Pembahasan

Tahap pengujian dan pembahasan dilakukan setelah implementasi dari rancangan sistem *multipath* routing dilakukan. Pengujian dilakukan untuk mengetahui kinerja sistem *multipath routing* dengan penggunaan jalur yang disesuaikan dengan *demand* dan ketersediaan *bandwidth* jalur. Pengujian dilakukan dengan menggunakan beberapa *tools*, yaitu iperf3 untuk membuat aliran data dan BWM-NG untuk melihat rata-rata *throughput* dan kondisi jaringan saat pengiriman data berlangsung. Parameter uji yang digunakan adalah distribusi trafik jaringandan waktu respons pemilihan jalur.

Tabel 3. 1 Tabel Parameter Uji

|  |  |  |
| --- | --- | --- |
| **No.** | **Parameter** | **Keterangan** |
| 1. | Distribusi Trafik jaringan | Hasil rata-rata *throughput* dari pengiriman data pada *switch* yang bercabang. Dilakukan dengan memantau port switch yang bercabang. |
| 2 | Waktu Respons Pemilihan Jalur | Selisih waktu peningkatan *throughput* dengan waktu nilai volume pengiriman data yang besar. Dilakukan dengan memantau *throughput* saat pengiriman data. |

Tabel 3.1 menunjukkan parameter uji yang digunakan pada penelitian ini. Distribusi trafik jaringan digunakan sebagai parameter uji untuk menunjukkan pengaruh mekanisme pemilihan jalur terhadap distribusi trafik jaringan pada jaringan. Distribusi trafik jaringan dilihat dari rata-rata *throughput* pada port switch yang bercabang menggunakan BWM-NG. Waktu respons pemilihan jalur digunakan sebagai parameter uji karena menunjukkan kecepatan mekanisme beradaptasi terhadap perubahan *demand*. Waktu respons dilihat dari selisih waktu peningkatan *throughput* dan waktu volume pertama kali aliran data mencapai volume yang besar adalah nilai dari waktu respons. Selisih waktu dilihat dengan melakukan pengamatan terhadap *throughput* saat pengiriman data berlangsung. Setiap parameter pengujian diuji dengan skenario pengujian yang berbeda. Skenario pengujian adalah tahapan prosedur untuk melakukan pengujian. Dengan menggunakan skenario pengujian, proses pengujian menjadi lebih terarah.

### Pengujian Distribusi Trafik Jaringan

Tabel 3. 2 Tabel Skenario Pengujian

|  |  |  |
| --- | --- | --- |
|  | ***Demand* 1 (Mbits)** | ***Demand* 2 (Mbits)** |
| Pengujian 1 | 10 | 0 |
| Pengujian 2 | 100 | 0 |
| Pengujian 3 | 10 | 50 |

Tabel 3.2 menunjukkan skenario pengujian distribusi trafik jaringan. Pengujian 1 dilakukan dengan mengirimkan data dengan iperf3 dengan *bitrate* yang dibatasi sebesar 10 Mbits selama 60 detik. Pengujian 1 menunjukkan kinerja sistem *multipath routing* pada kondisi *demand* yang kecil (dapat dilayani dengan 1 jalur). Nilai 10 Mbits digunakan sebagai batas bitrate karena menghasilkan penggunaan *bandwidth* jalur sebesar 10 Mbits, sehingga tidak mungkin terjadi *congestion* di jalur utama yang memiliki kapasitas jalur sebesar 70 Mbits (kapasitas *bandwidth* jalur terbesar).

Pengujian 2 dilakukan dengan mengirimkan data dengan iperf3 dengan *bitrate* yang dibatasi sebesar 100 Mbits selama 60 detik. Pengujian ini digunakan untuk menunjukkan kinerja sistem *multipath routing* pada kondisi *demand* yang besar (tidak dapat dilayani dengan 1 jalur). Nilai 100 Mbits digunakan karena 100 Mbits merupakan nilai total kapasitas *bandwidth* dari semua jalur dalam penelitian ini. Jika nilai *demand* adalah total kapasitas *bandwidth* dari semua jalur, maka hasil distribusi trafik yang baik adalah sama dengan rasio kapasitas *bandwidth* jalur.

Pengujian 3 dilakukan dengan mengirimkan data dengan *bitrate* yang dibatasi sebesar 10 Mbits selama 60 detik saat terjadi pengiriman data dengan *demand* 50 Mbits. Pengujian 3 menunjukkan pengiriman data dengan *demand* yang kecil saat ketersediaan *bandwidth* jalur berkurang. Pengiriman dengan *demand* 50 Mbits dilakukan untuk mengurangi ketersedian *bandwidth* jalur utama sebesar 50 Mbits, sehingga pengiriman dengan *demand* 10 Mbits akan memilih jalur dengan ketersedian *bandwidth* yang terbesar saat mengirim data.

### Pengujian Waktu Respons Pemilihan Jalur

Pengujian waktu respons pemilihan jalur dilakukan dengan pengiriman data dengan iperf3. Pengiriman data dilakukan dengan total *demand* sebesar 100 Mbits sehingga mekanisme pengiriman data akan berubah dari *singlepath* saat tidak ada *demand* ke *multipath* saat *demand* melebihi 80% *bandwidth* jalur utama. Perubahan mekanisme pengiriman data akan meningkatkan *throughput* dari pengiriman data. Pengamatan *throughput* dilakukan dengan BWM-NG. Selisih dari waktu peningkatan *throughput* dengan nilai waktu saat *demand* mencapai lebih dari 80% *bandwidth* jalur utama adalah nilai waktu respons pemilihan jalur. Nilai waktu respons yang terbaik adalah nilai waktu respons yang terendah.

Pengujian waktu respons pemilihan jalur dilakukan sebanyak 5 kali dengan nilai *polling* SFlow-rt yang berbeda-beda. Nilai *polling* menentukan interval switch mengirimkan paket SFlow ke perangkat SFlow-rt. Nilai *polling* yang digunakan adalah 1 detik, 3 detik, dan 10 detik. Nilai 1 detik digunakan karena kurang dari nilai interval *request* nilai *demand* dan penggunaan *bandwidth* dari controller ke SFlow-rt. Nilai 3 detik digunakan karena sama dengan nilai interval *request* nilai *demand* dan penggunaan *bandwidth* dari controller ke SFlow-rt. Nilai 10 detik digunakan karena lebih dari nilai interval *request* nilai *demand* dan penggunaan *bandwidth* dari controller ke SFlow-rt

## Penarikan Kesimpulan

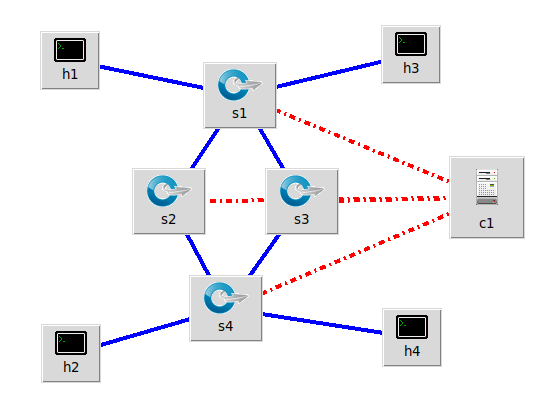
Tahap berikutnya adalah penarikan kesimpulan dari hasil pengujian terhadap sistem yang akan diusulkan. Kesimpulan dari peneliti diharapkan dapat menjawab rumusan permasalahan yang telah dibuat di bab pendahuluan. Kemudian, peneliti akan menyebutkan hal-hal apa saja untuk pengembangan penelitian berikutnya sebagai saran bagi pembaca yang tertarik dengan pekerjaan peneliti.

# Implementasi

Bab implementasi dalam penelitian ini memuat mengenai implementasi sistem *multipath routing* dengan penggunaan jalur yang proporsional bedasarkan kebutuhan *bandwidth* dari *host* mengacu pada perancangan yang sudah dilakukan. Implementasi yang dilakukan pada penelitian ini adalah implementasi lingkungan uji, implementasi mekanisme pencarian jalur, implementasi mekanisme *install* jalur dan mekanisme penggunaan jalur. Implementasi dari mekanisme ditunjukkan dengan pseudocode dari fungsi utama mekanisme.

## Implementasi Lingkungan Uji

Implementasi lingkungan uji dilakukan dengan mengimplementasikan rancangan topologi dengan miniedit pada mininet. Miniedit merupakan *graphical user interface* (GUI). Konfigurasi link dari topologi jaringan dilakukan sesuai dengan hasil rancangan topologi. Kemudian dilakukan konfigurasi terhadap mininet untuk menjalankan protocol OpenFlow versi 1.3 dan *enable* fungsi SFlow di setiap switch pada topologi. Topologi yang digunakan ditunjukkan oleh Gambar 4.1.



IP = 10.0.0.4

IP = 10.0.0.3

IP = 10.0.0.1

IP = 10.0.0.2

100 Mbits

100 Mbits

100 Mbits

100 Mbits

70 Mbits

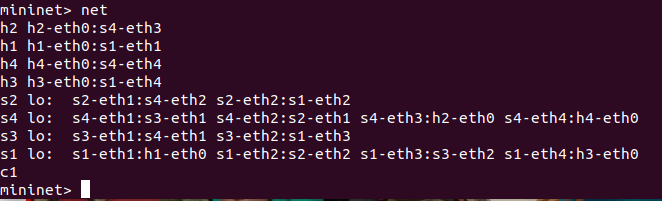
30 Mbits

30 Mbits

70 Mbits

Gambar 4. 1 Topologi

Gambar 4.1 menunjukkan topologi yang digunakan untuk pengujian sistem *multipath routing*. Komponen *host* pada gambar ditunjukkan oleh h1,h2,h3 dan h4. *Host* berjumlah 4 karena pengujian skenario ketiga membutuhkan 2 pengiriman data yang bersamaan sehingga membutuhkan 2 pasang sumber dan tujuan pengiriman data. Komponen switch ditunjukkan oleh s1, s2, s3 dan s4. Komponen *controller* ditunjukkan oleh c0. Komponen *link* ditunjukkan oleh garis biru yang menghubungkan setiap switch dan *host*. Link antara *host* dan switch memiliki *bandwidth* yang sama, yaitu 100 Mbits. Link yang terhubung dengan s2 memiliki *bandwidth* 70 Mbits sedangkan link yang terhubung dengan s3 memiliki *bandwidth* 30 Mbits. Garis merah putus-putus menunjukkan komponen channel khusus antara switch dan *controller*. Koneksi dari topologi jaringan ditunjukkan oleh Gambar 4.2.



Gambar 4. 2 Koneksi dari Topologi Jaringan

Gambar 4.2 menunjukkan koneksi dari topologi jaringan. *Host* h1 dan h3 terhubung dengan s1 melalui port s1-eth1 dan s1-eth4, sedangkan *host* h2 dan h4 terhubung dengan s4 melalui port s4-eth3 dan s4-eth4. Switch s1 dan s4 terhubung melalui s2 atau s3 sehingga di topologi jaringan terdapat 2 jalur yang berbeda. Switch s1 terhubung dengan s2 dan s3 melalui port s1-eth2 dan port s1-eth3. Switch s4 terhubung dengan s2 dan s3 melalui port s4-eth2 dan port s4-eth1.

## Implementasi *Multipath routing*

Implementasi dari sistem *multipath routing* dilakukan sesuai skema *multipath routing* dari Chiang yang terdiri dari 3 proses (Chiang, et. al., 2017). Proses pertama adalah *controller* SDN mempelajari topologi dari jaringan. Proses kedua adalah pencarian jalur antara 2 *node* dalam topologi. Ketiga, *controller* SDN mengatur switch untuk membagi trafik jaringan sesuai dengan jalur yang ditemukan.

### Implementasi Mekanisme Topologi Learning

Tahap pertama dari skema *multipath routing* diimplementasikan dengan menggunakan *option* yang dimiliki Ryu untuk mengimplementasikan tahap pertama dalam skema *multipath routing*. *Option* yang digunakan adalah observe-link. Dengan *option* observe-link, sistem *multipath routing* dapat mengetahui topologi dari jaringan. Tabel 4.4 menunjukkan penggunaan option observe-link dengan Ryu.

Tabel 4. 4 Perintah dengan Topologi Learning

|  |
| --- |
| $ Ryu-manager –-observe-links Dijkstra\_*multipath*.py |

### Implementasi Mekanisme Pencarian Jalur

Implementasi mekanisme pencarian jalur dilakukan dengan Python. Mekanisme pencarian jalur berguna untuk mendapatkan jalur-jalur dari *host* sumber ke *host* tujuan. Hasil dari mekanisme pencarian jalur adalah kumpulan jalur antara *host* sumber dan *host* tujuan terurut dari jalur dengan *bandwidth* terbesar ke jalur dengan *bandwidth* terkecil. Tabel 4.4 menunjukkan pseudocode fungsi utama dari mekanisme pencarian jalur.

Tabel 4. 5 Pseudocode Implementasi Mekanisme Pencarian Jalur

|  |  |
| --- | --- |
| Pseudocode Pencarian Jalur | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | Function Dijkstra(Graph, src, dst):  for each vertex p in Graph: # Initialization  distance[p] = ∞  previous[p] = undefined  distance[src] = 0  while Q is not empty: #Q contains all nodes in Graph  u = node in Q with smallest distance  remove u from Q  for each neighbor p of u:  alt = distance[u] + distance(u, p)  if alt < distance[p]:  distance[p] = alt  previous[p] = u  r = [] # r is the path from dst to src  p = dst  r.append(p) #insert p into the last of r  q = previous[p]  while q is not None:  if q == source:  r.append(q) #insert q into the last of r  break  p = q  r.append(p)  q = previous[p]  r.reverse() #reverse the order of element in r  if source == dst:  path = [source]  else:  path = r  return path    function find\_allpaths  path = Dijkstra(Graph, src, dst)  while path != []  remove path links and nodes from Graph except src and dst  add path to all\_paths  path = Dijkstra(Graph, source, dst)  return all\_paths |

Baris 1-30 pada Tabel 4.5 menunjukkan fungsi algoritme Dijkstra yang digunakan. Hasil dari fungsi algoritme Dijkstra adalah jalur dengan *bandwidth* terbesar. Untuk mendapatkan jalur dengan *bandwidth* terbesar, nilai distance(u,p) pada baris 10 adalah nilai *link* yang dihitung secara OSPF sesuai dengan rancangan penilaian jalur.

Baris 34-35 menunjukkan hasil dari fungsi Dijkstra kemudian digunakan untuk menghapus *node* dan *link* di *graph* pencarian dan disimpan di variabel all\_paths. Baris 33-36 menunjukkan proses fungsi Dijkstra, penyimpanan hasil fungsi Dijkstra dan penghapus *node* dan *link* dari *graph* pencarian diulang sampai fungsi Dijkstra menghasilkan nilai kosong.

### Implementasi Mekanisme Pembagian Trafik Jaringan

Mekanisme pembagian trafik jaringan pada penelitian ini membutuhkan mekanisme yang memantau trafik jaringan dari *host* dan mekanisme yang merubah penggunaan jalur. Oleh karena itu, implementasi penggunaan jalur dibagi menjadi 2 bagian. Bagian pertama adalah implementasi mekanisme pemantauan trafik jaringan. Bagian kedua adalah implementasi mekanisme pemilihan jalur.

#### Implementasi Mekanisme Pemantauan Trafik jaringan

Implementasi mekanisme pemantauan trafik jaringan dilakukan dengan Python dan SFlow-rt. Mekanisme pemantauan trafik jaringan berguna untuk memantau trafik jaringan yang datang dari *host* dan memantau *bandwidth* yang digunakan oleh sebuah jalur. Informasi dari mekanisme pemantauan trafik jaringan digunakan untuk menentukan penggunaan jalur. Tabel 4.6 menunjukkan pseudocode fungsi utama dari mekanisme pemantauan trafik jaringan.

Tabel 4. 6 Pseudocode Implementasi Mekanisme pemantauan Trafik jaringan

|  |  |
| --- | --- |
| Pseudocode Pemantau Traffic | |
| 1  2  3  4  5  6  7  8 | Function traffic\_mon(first\_port, last\_port, port\_in\_path):  inc\_traffic = 0  For each port p in port\_in\_path:  If p is first\_port or last port:  Port\_traffic = p *interface* inoctets  If Port\_traffic > inc\_traffic:  inc\_traffic = Port\_traffic  Return inc\_traffic |

Baris 1 pada Tabel 4.6 menunjukkan informasi yang digunakan untuk mekanisme pemantauan trafik jaringan. Nilai dari first\_port adalah port yang terhubung dengan *host* sumber, nilai dari last\_port adalah port yang terhubung dengan *host* tujuan, dan nilai dari port\_in\_path adalah port yang digunakan oleh jalur. Baris 3-7 menunjukkan proses penentuan nilai trafik jaringan yang datang dari *host* (inc\_traffic). Nilai trafik jaringan yang datang adalah nilai inOctets maksimal dari first\_port dan last\_port. Nilai inOctets didapatkan dengan melakukan *request* ke SFlow-rt manager.

#### Implementasi Mekanisme Pemilihan jalur

Mekanisme pemilihan jalur digunakan untuk mengubah nilai *bucket* *weight* dari *group entry*. Jika *bucket* *weight* diubah, maka penggunaan jalur berubah. Pengubahan *bucket* *weight* ditentukan oleh trafik jaringan hasil pemantau trafik jaringan, *bandwidth* jalur utama dan penggunaan *bandwidth* setiap jalur. Tabel 4.7 menunjukkan pseudocode dari mekanisme pemilih jalur.

Tabel 4. 7 Pseudocode Implementasi Mekanisme Pemilihan jalur

|  |  |
| --- | --- |
| Pseudocode Pemilihan jalur | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | Function path\_updater(first\_port, last\_port, port\_in\_path, paths\_weight, routing, group\_entry):  Pw = paths\_weight  G = group\_entry  out\_port = []  For each bucket b in G:  out\_port.append(b.watch\_port)  While true:  inc\_traffic = traffic\_mon(first\_port, last\_port, port\_in\_path)  for each port p in out\_port:  port\_util = p *interface* out utilization \* p *bandwidth*  p unused\_band = p *bandwidth* - p port util  max\_free\_port = port with least port\_util  max\_free = max\_free\_port unused *bandwidth*    update = none  if inc\_traffic > (max(Pw) \* 0.8):  if routing = *singlepath*:  update = *multipath*  else if routing = *multipath*:  update = *singlepath*  else if inc\_traffic < max\_free:  update = *singlepath*  else if inc\_traffic > max\_free:  update = *multipath*  if update = *multipath*:  routing = *multipath*  for each bucket b in G:  p = b.watch\_port  b.bucket\_weight = (p unused\_band / total\_unused\_band) \* 10  else if update = *singlepath*:  routing = *singlepath*  for each bucket b in G:  if b.watch\_port = max\_free\_port:  b.bucket\_weight = 65335  else:  b.bucket\_weight = 0  sleep(3) |

Baris 1 pada Tabel 4.7 proses persiapan informasi yang relevan untuk mekanisme pemilihan jalur. Nilai dari first\_port adalah port yang terhubung dengan *host* sumber, nilai dari last\_port adalah port yang terhubung dengan *host* tujuan, nilai dari port\_in\_path adalah port yang digunakan oleh jalur, nilai dari paths\_weight adalah list dari nilai jalur (*bandwidth* minimal sebuah jalur), nilai *routing* adalah mekanisme *routing* yang sedang digunakan dan *group entry* adalah *group entry* yang sedang digunakan sistem *multipath routing*. Variabel first\_port, last\_port dan port\_in\_path digunakan untuk fungsi trafik jaringan\_mon yang mengembalikan nilai inc\_traffic pada baris 8. Baris 2-6 menunjukkan informasi yang digunakan untuk merubah jalur. Variabel out\_port didapatkan dari nilai watch\_port dari *bucket* yang ada di *group entry*. Variabel out\_port digunakan untuk mendapatkan nilai penggunaan *bandwidth*, nilai ketersediaan *bandwidth* dari out\_port, port yang memiliki ketersediaan *bandwidth* tertinggi (max\_free\_port) dan nilai maksimal ketersedian *bandwidth* (max\_free) pada baris 10-14.

Baris 16-25 menunjukkan proses pemilihan skema penggunaan jalur (secara *singlepath* atau *multipath*). Skema penggunaan jalur ditentukan bedasarkan nilai *demand*, nilai *bandwidth* jalur terbaik, dan nilai tertinggi *bandwidth* sisa dari semua jalur. Jika *demand* lebih dari 80% *bandwidth* jalur utama dan skema penggunaan jalur adalah *singlepath routing*, maka penggunaan jalur diubah menjadi *multipath routing* (*bucket* *weight* jalur diubah sesuai rumus 3.2). Sebaliknya jika *demand* kurang dari 80% *bandwidth* jalur utama dan skema penggunaan jalurnya adalah *multipath routing*, maka penggunaan jalur diubah menjadi *singlepath routing* (*bucket* *weight* jalur alternatif diberubah menjadi 0). Kemudian jika *demand* kurang dari 80% *bandwidth* jalur terbaik dan sistem penggunaan jalur adalah *singlepath routing*, maka sistem akan memilih jalur sesuai nilai tertinggi dari sisa *bandwidth* jalur (max\_free). Jika nilai *demand* kurang dari max\_free, maka sistem menggunakan jalur secara *singlepath routing* dengan jalur yang memiliki sisa *bandwidth* tertinggi. Jika nilai *demand* lebih dari max\_free, maka sistem menggunakan jalur secara *multipath routing* (*bucket* *weight* jalur diubah sesuai rumus 3.2).

Baris 27 – 38 menunjukkan proses merubah skema penggunaan jalur. Baris 27-31 merupakan proses merubah skema pengiriman data menjadi *multipath routing*. Proses dilakukan dengan memodifikasi *bucket* *weight* dari *group entry*. Nilai *bucket* *weight* yang digunakan sesuai dengan rumus sisa *bandwidth* jalur dibagi total sisa *bandwidth* semua jalur. Baris 32-38 menunjukkan proses perubahan penggunaan jalur menjadi *singlepath routing*. Perubahan penggunaan jalur ke *singlepath* dilakukan dengan merubah *bucket* *weight* semua *bucket* kecuali *bucket* dengan nilai watch\_port sama dengan port dengan ketersediaan *bandwidth* tertinggi. Pada proses ini, jalur yang dipilih adalah jalur yang memiliki sisa *bandwidth* jalur tertinggi.

## Implementasi Pengujian Kinerja

Pengujian kinerja dilakukan dengan mengirimkan data TCP antara 2 *host*. Implementasi dari pengiriman data TCP dilakukan dengan menggunakan *tool* iperf3. *Host* h1 dan h3 berperan sebagai sumber dari pengiriman data sedangkan *host* h2 dan h4 berperan sebagai tujuan dari pengiriman data. Pengiriman data dibuat secara parallel dengan option –P. Kemudian pembatasan bandwith dilakukan terhadap setiap aliran TCP parallel untuk memanipulasi *demand* (trafik jaringan dari *host*) dengan option –b.

### Distribusi Trafik Jaringan

Pengujian distribusi trafik jaringan dilakukan sebanyak 3 kali. Pada pengujian pertama *host* h1 sebagai sumber dan *host* h2 sebagai tujuan. Nilai *bandwidth* setiap aliran data dibatasi sebesar 1 Mbits. Jumlah aliran data parallel adalah 10 sehingga maksimal *demand* (trafik jaringan dari pengguna) sebesar 10 Mbits.

Pada pengujian kedua *host* h1 sebagai sumber dan *host* h2 sebagai tujuan. Nilai *bandwidth* setiap aliran data dibatasi sebesar 10 Mbits. Jumlah aliran data parallel adalah 10 sehingga maksimal *demand* (trafik jaringan dari pengguna) sebesar 100 Mbits.

Pada pengujian ketiga *host* h1 dan h3 sebagai sumber sedangkan *host* h2 dan h4 sebagai tujuan. Pengiriman dari h1 ke h2 dan h3 ke h4 dilakukan bersamaan. Nilai *bandwidth* setiap aliran data dari h1 dibatasi sebesar 1 Mbits sedangkan dari h3 dibatasi sebesar 5 Mbits. Jumlah aliran data parallel adalah 10 sehingga maksimal *demand* (trafik jaringan dari pengguna) dari h1 sebesar 10 Mbits sedangkan dari h2 sebesar 50 Mbits. Tabel 4.8 menunjukkan pengiriman data pada pengujian distribusi trafik jaringan.

Pengukuran kinerja distribusi trafik jaringan dilakukan saat pengiriman data TCP terjadi. Distribusi trafik jaringan dipantau dengan *tool* BWM-NG dengan option yang menampilkan *throughput* saat pengiriman data TCP berlangsung. Tabel 4.9 menunjukkan perintah *tool* BWM-NG pengujian distribusi trafik jaringan.

Tabel 4. 8 Perintah Iperf3 untuk Pengujian Distribusi Trafik Jaringan

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #Pengujian 1  Mininet> h2 iperf3 –s –D  Mininet> h1 iperf3 –c h2 –P 10 –b 1M –t 60 –V  #Pengujian 2  Mininet> h2 iperf3 –s –D  Mininet> h1 iperf3 –c h2 –P 10 –b 10M –t 60 –V  #Pengujian 3  Mininet> h2 iperf3 –s –D  Mininet> h1 iperf3 –c h2 –P 10 –b 1M –t 60 –V  Mininet> h4 iperf3 –s –D  Mininet> h3 iperf3 –c h4 –P 10 –b 5M –t 60 –V |

Tabel 4. 9 Perintah BWM-NG Iperf3 untuk Pengujian Distribusi Trafik Jaringan

|  |  |
| --- | --- |
| 1 | $ Bwm-ng –o csv –I s1-eth2,s1-eth3 –T rate –F [name] |

Tabel 4. 10 Hasil Pengujian Distribusi Trafik Jaringan

|  |  |  |  |
| --- | --- | --- | --- |
| Skenario | Trial | Rata-rata *throughput* (Mbits/s) | |
| s1-eth2 (70 Mbits) | s1-eth3 (30 Mbits) |
| Skenario 1 | 1 | 10.12 | 0.03 |
| 2 | 10.24 | 0.00 |
| 3 | 10.15 | 0.00 |
| 4 | 10.15 | 0.00 |
| 5 | 10.12 | 0.00 |
| Rata-rata | 10.16 | 0.01 |
| Skenario 2 | 1 | 67.66 | 19.72 |
| 2 | 61.40 | 28.12 |
| 3 | 67.63 | 19.43 |
| 4 | 67.66 | 9.54 |
| 5 | 60.91 | 28.34 |
| Rata-rata | 65.05 | 21.03 |
| Skenario 3 | 1 | 40.02 | 18.35 |
| 2 | 58.83 | 0.00 |
| 3 | 44.16 | 14.22 |
| 4 | 39.91 | 15.16 |
| 5 | 58.85 | 0.00 |
| Rata-rata | 48.35 | 9.55 |

Pengamatan *throughput* dilakukan terhadap *interface* switch yang bercabang yaitu s1-eth2 dan s1-eth3. Nilai *outbound* pada *interface* s1-eth2 menunjukkan nilai data yang dikirim melalui jalur 1, sedangkan nilai *outbound* pada *interface* s1-eth3 menunjukkan nilai data yang dikirim melalui jalur 2. Rata-rata kedua nilai trafik jaringan *outbound* dalam setiap pengiriman menunjukkan nilai distribusi dari trafik jaringan.

Hasil pengujian distribusi trafik jaringan ditunjukkan oleh Tabel 4.10. Skenario pertama menghasilkan pengiriman data dengan jalur dengan *bandwidth* lebih besar. Pada skenario kedua, Tabel 4.10 menunjukkan pengiriman data secara *multipath* dengan rasio yang mendekati kapasitas jalur dengan rata-rata *throughput* jalur 1 bernilai 60.91 Mbits sampai 67.66 Mbits dan jalur 2 bernilai 9.54 Mbits sampai 28.34 Mbits. Pada skenario ketiga, pengiriman data dilakukan secara *multipath* atau *singlepath*. Saat pengiriman data secara *singlepath*, rata-rata *throughput* jalur 1 bernilai 58.83 Mbits sampai 58.85 Mbits. Saat pengiriman data secara *multipath*, rata-rata *throughput* jalur 1 bernilai 39.91 Mbits sampai 44.16 Mbits dan jalur 2 bernilai 14.22 Mbits sampai 18.35 Mbits.

### Waktu Respons Pemilihan Jalur

Pengujian kinerja waktu respons pemilihan jalur dilakukan dengan pengiriman data dari *host* h1 ke *host* h2 dengan total *demand* sebesar 100 Mbits. Tabel 4.11 menunjukkan perintah pengiriman data pengujian waktu respons. Pengiriman dilakukan sebanyak 5 kali dengan nilai *polling* dari SFlow yang berbeda-beda. Nilai *polling* yang digunakan adalah 1 detik, 3 detik, dan 10 detik. Pengaturan nilai *polling* dilakukan dengan menggunakan miniedit pada mininet.

Tabel 4. 11 Perintah Iperf3 untuk Pengujian Waktu Respons Pemilihan Jalur

|  |  |
| --- | --- |
| 1  2  3 | #Pengiriman data dengan total *demand* 100 Mbits  Mininet> h2 iperf3 –s –D  Mininet> h1 iperf3 –c h2 –P 10 –b 10M –t 60 –V |

Pengukuran kinerja waktu respons dilakukan dengan melihat data *throughput* dari pengiriman data. Data *throughput* didapatkan dengan *tool* BWM-NG yang berupa file .csv. Data *throughput* yang diambil adalah *throughput* masuk pada *interface* s1-eth1 dan *throughput* keluar pada s1-eth2 dan s1-eth3. Tabel 4.12 menunjukkan perintah *tool* BWM-NG untuk pengujian waktu respons pemilihan jalur.

Tabel 4. 12 Perintah BWM-NG untuk Pengujian Waktu Respons Pemilihan Jalur

|  |  |
| --- | --- |
| 1  2 | #perintah BWM-NG untuk setiap pengiriman data  $ Bwm-ng –o csv –I s1-eth1,s1-eth2,s1-eth3 –T rate –F filename\_*polling*value.csv |

Pengamatan terhadap data *throughput* masuk pada s1-eth1 menghasilkan nilai trafik dari pengguna, sedangkan pengamatan terhadap *throughput* keluar pada s1-eth2 dan s1-eth3 menunjukkan nilai trafik jaringan yang keluar dari s1. Kemudian nilai trafik jaringan dari pengguna dibandingkan dengan nilai trafik jaringan yang keluar dari s1. Selisih waktu saat nilai trafik jaringan dari pengguna lebih dari 80% kapasitas *bandwidth* jalur utama dengan peningkatan nilai trafik jaringan yang keluar dari s1 merupakan nilai waktu respons pemilihan jalur.

Tabel 4. 13 Hasil Pengujian Waktu Respons

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nilai *Polling* | Trial | *Demand* > 80% | Peningkatan *Throughput* | Waktu Respons |
| 1 detik | 1 | 12:10:56 | 12:10:58 | 0:00:02 |
| 2 | 12:12:16 | 12:12:18 | 0:00:02 |
| 3 | 12:13:36 | 12:13:39 | 0:00:03 |
| 4 | 12:14:56 | 12:14:59 | 0:00:03 |
| 5 | 12:16:17 | 12:16:18 | 0:00:01 |
| 3 detik | 1 | 12:44:23 | 12:44:24 | 0:00:01 |
| 2 | 12:45:44 | 12:45:47 | 0:00:03 |
| 3 | 12:47:04 | 12:47:09 | 0:00:05 |
| 4 | 12:48:24 | 12:48:26 | 0:00:02 |
| 5 | 12:49:44 | 12:49:48 | 0:00:04 |
| 10 detik | 1 | 18:46:08 | - | Error |
| 2 | 18:47:28 | 18:47:43 | 0:00:15 |
| 3 | 18:48:48 | 18:49:02 | 0:00:14 |
| 4 | 18:50:08 | 18:50:23 | 0:00:15 |
| 5 | 18:51:28 | 18:51:44 | 0:00:16 |

Hasil pengujian waktu respons ditunjukkan oleh Tabel 4.13. Skenario pertama menghasilkan waktu respons 1 detik sampai 3 detik. Pada skenario kedua, Tabel 4.13 menunjukkan nilai waktu respons antara 1 detik sampai 5 detik. Skenario ketiga menghasilkan pengiriman data secara *singlepath* dan *multipath*. Pengiriman secara *singlepath* menunjukkan tidak ada respons terhadap peningkatan *demand*. Pada skenario ketiga, pengiriman secara *multipath* menunjukkan waktu respons antara 14 detik sampai 16 detik.

# Pembahasan Hasil Pengujian

Pengujian dalam bab ini memuat hasil pengujian yang sudah dilakukan terhadap hasil implementasi yang sudah dilakukan. Pengujian dilakukan bedasarkan rancangan pengujian yang sudah dibuat. Pada bagian ini terdapat dua subbab pengujian yaitu pengujian distribusi trafik jaringan dan pengujian waktu respons pemilihan jalur.

## Distribusi Trafik Jaringan

Pengujian distribusi trafik jaringan dilakukan dengan memantau rata-rata *throughput* jalur 1 (s1-s2-s4) dan jalur 2 (s1-s3-s4) saat pengiriman data. Pemantauan rata-rata *throughput* dilakukan dengan BWM-NG pada port s1-eth2 dan s1-eth3. Port s1-eth2 dipantau karena port s1-eth2 menghubungkan s1 dan s2 sedangkan port s1-eth3 dipantau karena port s1-eth3 menghubungkan s1 dan s3. Tabel 4.10 menunjukkan hasil dari pengamatan distribusi trafik jaringan *multipath* routing dengan mekanisme pemilihan jalur.

Pada pengujian dengan skenario pertama, Tabel 4.10 menunjukkan distribusi trafik jaringan pada pengiriman data dengan batas *demand* 10 Mbits. Pengujian pertama digunakan untuk mengemulasikan pengiriman data dengan *demand* yang rendah (dapat dilakukan dengan satu jalur). Hasil pengujian menunjukkan sistem selalu mengirim data secara *singlepath*. Pengiriman data secara *singlepath* ditunjukkan dari hasil rata-rata *throughput* jalur 1 yaitu 10.12 Mbits/s sampai dengan 10.24 Mbits/s dan jalur 2 yaitu 0.00 Mbits/s sampai dengan 0.03 Mbits/s.

Bandwidth Jalur 1

Gambar 5. 1 Hasil Distribusi Trafik Jaringan Skenario 1

Gambar 5.1 menunjukkan hasil pengujian sistem pada kondisi *demand* yang rendah. Hasil pengujian menunjukkan rata-rata penggunaan jalur 1 sebesar 10.16 Mbits/s dan penggunaan jalur 2 sebesar 0.01 Mbits/s. Rata-rata penggunaan jalur menunjukkan pengiriman data hanya dilakukan dengan jalur 1. Hal ini terjadi karena nilai bobot (*bucket weight*) jalur 2 adalah 0. Sistem *group select* pada OpenvSwitch tidak akan memilih jalur dengan bobot 0 untuk mengirimkan data (*Open Networking Foundation*, 2014). Dalam skenario ini jalur 1 digunakan untuk mengirimkan data karena jalur 1 memiliki ketersediaan bandwidth tertinggi diantara jalur yang ada.

Pada pengujian dengan skenario kedua, Tabel 4.10 menunjukkan distribusi trafik jaringan pada pengiriman data dengan batas *demand* 100 Mbits. Pengujian kedua digunakan untuk mengemulasikan pengiriman data dengan *demand* yang tinggi (tidak dapat dilakukan dengan satu jalur). Pada pengujian skenario 2 nilai maksimal dari demand adalah nilai total dari bandwidth jalur sehingga nilai distribusi trafik yang proporsional adalah nilai yang memiliki rasio *throughput* mendekati rasio *bandwidth* jalur.

Bandwidth Jalur 2

Bandwidth Jalur 1

Gambar 5. 2 Hasil Distribusi Trafik Jaringan Skenario 2

Gambar 5.2 menunjukkan hasil pengujian sistem pada kondisi *demand* yang tinggi. Hasil pengujian menunjukkan pengiriman data secara *multipath* dengan rata-rata penggunaan jalur 1 (65.05 Mbits/s) yang lebih tinggi daripada penggunaan jalur 2 (21.03 Mbits/s). Penggunaan jalur yang berbeda terjadi karena ketersedian *bandwidth* yang berbeda antara kedua jalur. Saat pengiriman akan dilakukan, kondisi jaringan yang kosong (tidak ada pengiriman data) menghasilkan ketersediaan *bandwidth* jalur yang sama dengan *bandwidth* jalur. Mayoritas trafik jaringan dikirim dengan jalur 1 karena jalur 1 memiliki *bandwidth* sebesar 70 Mbits sedangkan jalur 2 memiliki *bandwidth* sebesar 30 Mbits. Distribusi trafik sesuai ketersedian *bandwidth* dilakukan untuk menghindari kelebihan beban di jalur 2.

Bandwidth Jalur 1

Bandwidth Jalur 2

Gambar 5. 3 Hasil Distribusi Trafik Jaringan Skenario 3

Gambar 5.3 menunjukkan distribusi trafik jaringan pada pengiriman data dengan *demand* 10 Mbits dan pengiriman data dengan *demand* 50 Mbits yang dilakukan secara bersamaan. Pengujian ketiga mengemulasikan pengiriman data dengan *demand* yang kecil saat ketersediaan *bandwidth* jalur berkurang. Hasil rata-rata *throughput* menunjukkan pengiriman data secara *singlepath* atau *multipath*. Saat pengiriman data secara *singlepath*, rata-rata *throughput* jalur 1 bernilai 58.83 Mbits sampai 58.85 Mbits. Saat pengiriman data secara *multipath*, rata-rata *throughput* jalur 1 bernilai 39.91 Mbits sampai 44.16 Mbits dan jalur 2 bernilai 14.22 Mbits sampai 18.35 Mbits. .

Hasil pengujian skenario ketiga menunjukkan mekanisme pemilihan jalur dapat mengatur distribusi trafik jaringan sesuai dengan ketersediaan *bandwidth*. Hal ini ditunjukkan saat pengiriman data dilakukan secara *multipath* yang menghasilkan rata-rata *throughput* di jalur 1 selalu lebih besar daripada rata-rata *throughput* di jalur 2. Saat pengiriman dengan *demand* 50 Mbits terjadi, ketersedian dari jalur 1 berkurang, menyisakan 20 Mbits. Kemudian karena pengiriman data dilakukan secara kontinyu selama 60 detik, pada detik berikutnya pengiriman dengan *demand* 50 Mbits dan pengiriman dengan *demand* 10 Mbits terjadi. Pengiriman dengan *demand* 50 Mbits dilakukan dengan 2 jalur karena ketersedian *bandwidth* dari jalur 1 lebih kecil daripada *demand* 50 Mbits. Pengiriman secara *multipath* dari *demand* 50 Mbits menyebabkan ketersedian *bandwidth* di jalur 1 lebih besar daripada di jalur 2 sehingga pengiriman data dengan *demand* 10 Mbits dilakukan dengan jalur 1. Kegagalan dalam mendeteksi ketersediaan *bandwidth* menyebabkan kedua pengiriman dilakukan dengan 1 jalur.

Bedasarkan hasil pengujian distribusi trafik jaringan, mekanisme pemilihan jalur mampu menyesuaikan penggunaan jalur dari *multipath routing* sehingga berdampak pada distribusi trafik jaringan. Pada kondisi *demand* yang rendah mekanisme pemilihan jalur mendistribusikan trafik jaringan secara *singlepath* *routing*. Pada kondisi *demand* yang tinggi mekanisme pemilihan jalur miningkatkan penggunaan jalur 1 (jalur dengan ketersediaan *bandwidth* tertinggi) daripada jalur 2. Hal ini ditunjukkan oleh rasio rata-rata distribusi trafik jaringan 65 : 21 . Rasio distribusi trafik tersebut mendekati nilai kapasitas *bandwidth* yaitu 70 : 30. Pada kondisi *demand* yang rendah dan ketersediaan *bandwidth* berkurang, sistem mengirim data dengan jalur yang memiliki ketersediaan *bandwidth* lebih besar.

## Waktu Respons Pemilihan Jalur

Pengujian waktu respons dilakukan untuk kecepatan mekanisme beradaptasi terhadap perubahan *demand*. Pengujian dilakukan dengan skenario sesuai dengan bab 3.3.2. Pengamatan dari pengujian dilakukan dengan *tools* BWM-NG dengan keluaran berbentuk bentuk file csv. Nilai dari waktu respons diambil dari selisih waktu nilai trafik jaringan inbound melebihi 80% *bandwidth* jalur 1 ( > 56 Mbits) dan waktu nilai total trafik outbound yang meningkat karena penggunaan jalur kedua.

Gambar 5.4 menunjukkan graph peningkatan *throughput* dari salah satu pengujian waktu respons pemilihan jalur dengan nilai *polling* 1. Gambar 5.4 menunjukkan trafik jaringan dari pengguna yang meningkat dari 47 Mbits pada 12:16:16 ke 68 Mbits pada 12:16:17 ke 89 Mbits pada 12:16:18. Peningkatan trafik jaringan dari pengguna hingga lebih dari 56 Mbits menyebabkan penyesuaian penggunaan jalur pada pengiriman data. Hal ini ditunjukkan oleh penggunaan jalur 2 sebesar 27 Mbits pada 12:16:18 yang meningkatan total trafik jaringan yang keluar dari s1 dari 68 Mbits pada 12:16:17 ke 89 Mbits pada 12:16:18.

Pada pengujian dengan nilai *polling* 1, Gambar 5.4 menunjukkan waktu respons pemilihan jalur saat interval pengiriman paket SFlow bernilai 1 detik. Hasil pengujian menunjukkan waktu respons pemilihan jalur bernilai 1 detik. Hasil waktu respons 1 detik didapatkan karena saat mekanisme pembagian trafik jaringan melakukan *request* nilai trafik jaringan dari *host*, sistem selalu mendapat nilai trafik jaringan dari *host* yang terbaru. Sistem selalu mendapatkan nilai trafik jaringan terbaru juga ditunjukkan oleh hasil pengujian pada tabel 4.13 yang menunjukkan waktu respons saat nilai *polling* 1 tidak pernah melebihi waktu *request* nilai *demand* dari sistem. Oleh karena itu, mekanisme pemilihan jalur akan menyesuaikan penggunaan jalur sesuai dengan nilai trafik jaringan dari *host* setiap *request* dilakukan.

Peningkatan Throughput

Demand > 56 Mbits

Gambar 5. 4 Graph *Throughput* S1 (Nilai *Polling* SFlow 1)

Gambar 5.5 menunjukkan graph peningkatan *throughput* dari salah satu pengujian waktu respons pemilihan jalur dengan nilai *polling* 3. Gambar 5.5 menunjukkan trafik jaringan dari pengguna yang meningkat dari 0 Mbits pada 12:49:40 ke 68 Mbits pada 12:49:41 ke 89 Mbits pada 12:49:46. Peningkatan trafik jaringan dari pengguna hingga lebih dari 56 Mbits menyebabkan penyesuaian penggunaan jalur pada pengiriman data. Hal ini ditunjukkan oleh penggunaan jalur 2 sebesar 20 Mbits pada 12:49:46 yang meningkatan total trafik jaringan yang keluar dari s1 dari 68 Mbits pada 12:49:41 ke 88 Mbits pada 12:49:46.

Peningkatan Throughput

Demand > 56 Mbits

Gambar 5. 5 Graph *Throughput* S1 (Nilai *Polling* SFlow 3)

Pada pengujian dengan nilai *polling* 3, Gambar 5.5 menunjukkan waktu respons pemilihan jalur saat interval pengiriman paket SFlow bernilai 3 detik. Hasil pengujian menunjukkan waktu respons pemilihan jalur bernilai 5 detik. Hasil ini didapakatkan karena nilai trafik jaringan dari *host* yang didapatkan saat sistem melakukan *request* belum tentu nilai terbaru. Oleh karena itu, ada kemungkinan mekanisme pemilihan jalur menyesuaikan penggunaan jalur dengan nilai trafik jaringan yang belum berubah. Nilai dari Sflow yang tidak realtime menyebabkan peningkatan waktu yang dibutuhkan mekanisme untuk mengubah penggunaan jalur sesuai trafik jaringan dari *host*. Hal ini ditunjukkan dari hasil pada Tabel 4.13 yang menunjukkan nilai maksimal dari waktu respons sistem meningkat dari 3 detik ke 5 detik.

Peningkatan Throughput

Gambar 5. 6 Graph *Throughput* S1 (Nilai *Polling* SFlow 10)

Gambar 5.6 menunjukkan graph *throughput* dari pengujian waktu respons pemilihan jalur dengan nilai *polling* 10. Gambar 5.6 menunjukkan trafik jaringan dari pengguna yang meningkat dari 0 Mbits pada 18:51:22 ke 68 Mbits pada 18:51:23 ke 91 Mbits pada 18:51:39. Peningkatan trafik jaringan dari pengguna hingga lebih dari 56 Mbits menyebabkan penyesuaian penggunaan jalur pada pengiriman data. Hal ini ditunjukkan oleh penggunaan jalur 2 sebesar 29 Mbits pada 18:51:39 yang meningkatan total trafik jaringan yang keluar dari s1 dari 68 Mbits pada 18:51:23 ke 90 Mbits pada 18:51:39.

Hasil pengujian dengan nilai *polling* 10 menunjukkan peningkatan drastic dari waktu yang dibutuhkan mekanisme untuk mengubah penggunaan jalur sesuai trafik jaringan dari *host*. Peningkatan yang terjadi berbanding lurus dengan nilai *polling* yang digunakan dengan nilai *polling* 1 menghasilkan waktu respons 1 detik sampai 3 detik, nilai *polling* 3 menghasilkan waktu respons 1 detik sampai 5 detik dan nilai *polling* 10 menghasilkan waktu respons 14 detik sampai 16 detik. Selain peningkatan waktu respons, ada kemungkinan sistem tidak melakukan perubahan distribusi trafik jaringan meskipun nilai beban trafik jaringan dari *host* lebih dari 80% *bandwidth* jalur 1. Peningkatan nilai waktu respons yang drastis terjadi karena kemungkinan nilai trafik jaringan dari *host* yang didapatkan oleh mekanisme bukan nilai yang terbaru semakin besar saat nilai *polling* semakin besar.

Bedasarkan hasil pengujian waktu respons pemilihan jalur, waktu respons pemilihan jalur akan berubah jika nilai *polling* berubah. Hasil pengujian menunjukkan nilai waktu respons terbaik jika nilai *polling* bernilai 1 dan akan memburuk jika nilai *polling* meningkat. Peningkatan nilai *polling* memunculkan kemungkinan mekanisme tidak mendapakatkan nilai trafik jaringan dari *host* yang terbaru sehingga mekanisme pemilihan jalur tidak mengubah penggunaan jalur.

# Kesimpulan

Pada bab ini dijelaskan tentang kesimpulan penelitian yang telah dilakukan. Kesimpulan yang dijelaskan didapat dari seluruh proses penelitian. Berikut kesimpulan yang bisa diambil dari penelitian ini:

Implementasi sistem *multipath routing* dilakukan dengan Ryu dan SFlow-rt dengan lingkungan uji yang disimulasikan dengan mininet. Sistem *multipath routing* mendistribusikan trafik jaringan dengan melihat hasil pemantauan *demand* (trafik jaringan dari *host*) dan ketersedian *bandwidth* jalur yang dilakukan dengan *tool* SFlow-rt. Kemudian sistem membagi trafik jaringan sesuai dengan demand dan ketersedian bandwidth jalur.

Pendistribusian trafik jaringan dilakukan sesuai dengan *demand* dan ketersedian *bandwidth*.

Pada pengiriman data dengan batas *demand* 10 Mbits, rata-rata distribusi trafik jaringan adalah 10.16 Mbits/s dibanding 0.3 Mbits/s.

Pada pengiriman data dengan batas *demand* 100 Mbits,rata-rata distribusi trafik jaringan adalah 65.05 Mbits/s dibanding 21.03 Mbits/s.

Pada jaringan yang mengirimkan data 10 Mbits dan 50 Mbits secara bersamaan, pengiriman data dilakukan secara *singlepath* atau *multipath*. Saat pengiriman data secara *singlepath*, rata-rata *throughput* jalur 1 bernilai 58.83 Mbits sampai 58.85 Mbits. Saat pengiriman data secara *multipath*, rata-rata *throughput* jalur 1 bernilai 39.91 Mbits sampai 44.16 Mbits dan jalur 2 bernilai 14.22 Mbits sampai 18.35 Mbits. Rata-rata distribusi trafik jaringan dari pengiriman data 10 Mbits dan 50 Mbits adalah 48.35 Mbits/s dibanding 9.55 Mbits/s.

Sistem multipath routing memiliki kinerja waktu respons yang dipengaruhi oleh nilai *polling* SFlow. Pada jaringan dengan nilai *polling* SFlow 1 detik, waktu respons pemilihan jalur adalah 1 detik hingga 3 detik. Nilai waktu respons pemilihan jalur bertambah seiring nilai *polling* bertambah. Nilai *polling* 3 detik menghasilkan waktu respons bernilai 1 detik hingga 5 detik. Nilai *polling* 10 detik menghasilkan waktu respons berniali 14 detik hingga 16 detik dan ada kemungkinan distribusi trafik jaringan tidak berubah meskipun nilai beban trafik jaringan dari *host* melebihi 80% *bandwidth* jalur utama (jalur dengan *bandwidth* terbesar).

Pada penelitian ini sistem membagi trafik jaringan hampir proporsional dengan ketersediaan bandwidth jalur. Hal ini ditunjukkan dari hasil pengujian distribusi trafik jaringan skenario 1 dan skenario 2. Pada skenario 1 pengiriman data dilakukan secara singlepath karena trafik jaringan dari pengguna bernilai kecil (10 Mbits). Pada skenario 2 pengiriman data dilakukan secara multipath dengan rata-rata rasio penggunaan jalur yang mendekati ketersediaan bandwidth yaitu 65:21 sedangkan ketersediaan bandwidth adalah 70:30.

DAFTAR REFERENSI

Cisco (2005). OSPF Design Guide. [Online] Tersedia di https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-*path*-first-ospf/7039-1.html [diakses pada 10 April 2019]

Chiang, Y., Ke, C., Yu, Y., Chen, Y., & Pan, C. (2017). A *Multipath* Transmission Scheme for the Improvement of *Throughput* over SDN. Proceedings of the 2017 IEEE International Conference on Applied System Innovation, IEEE.

Gorranson, P., Black, C., & Culve, T. (2017). Software Defined Network A Comprehensive Approach 2nd Edition.

Hertiana, S. N. & Hendrawan, A. K. (2015). *A Joint Approach to Multipath routing and Rate Adaptation for Congestion Control in* OpenFlow *Software Defined Network*

Kreutz, D., Verissimo, P. E., Azodolmolky, S., Ramos, F. M., Rothenberg, C. E., & Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE.* IEEE.

Linux Foundation (2016). Open vSwitch Documentation. [Online] Tersedia di : http://docs.openvswitch.org/en/latest

Maulana, W., Yahya, W., & Rizqika, S. (2017). *Multipath routing* dengan Load-Balancing Pada OpenFlow Software-Defined Network. Repositori Jurnal Mahasiswa PTIIK UB, Volume 9, p. 15.

Nunes, B. A. A., Mendonca, M., Nguyen, X., Obraczka, K. & Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys & Tutorials, 16(3), pp. 1617 - 1634.

*Open* Networking Foundation (2013). SDN Architecture. [pdf]

*Open* Networking Foundation (2014). OpenFlow *Switch* Specification Version 1.3.4. [pdf]

RFC 2992 (2000). Analysis of an Equal-*Cost* Multi-Path Algorithm. [Online] Tersedia di : https://*tools*.ietf.org/html/rfc2992 [diakses pada 20 Agustus 2019]

Wirawan, U. Y. T., Yahya, W., & Basuki, A. (2018). Implementasi *Multipath routing* Berbasis Algoritme DFS yang Dimodifikasi, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol 2.

Yahya, W., Basuki, A., Maulana, W., Rizqika, S., & Bhawiyuga, A. (2018). Improving End-to-End Network *Throughput* Using *Multi*ple Best *Path*s *Routing* in Software Defined Networking, IEEE.