# MINIMALISASI PEMILIHAN RUTE OVERLAP PADA EQUAL COST MULTIPATH ROUTING (ECMP) DENGAN PENDEKATAN SOFTWARE DEFINED NETWORKING

## *Minimalization of Overlapped Path Selection on Equal Cost Multipath Routing (ECMP) with Software Defined Networking Paradigm*

**Ramadhika Dewanto[1], Rendy Munadi[2], Ridha Muldina Negara[3]**

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik, Universitas Telkom

[1]ramadhikade@students.telkomuniversity.ac.id, [2]rendymunadi@telkomuniveristy.ac.id, [3]ridhanegara@telkomuniversity.ac.id

**Abstrak**

Equal Cost Multipath Routing (ECMP) adalah proses routing yang memanfaatkan seluruh pilihan jalur yang tersedia antara satu node dengan node lainnya dalam jaringan dengan memetakan setiap jalur dengan seluruh kemungkinan trafik (pasangan source-destination) secara statis. Konfigurasi tersebut memungkinkan adanya pengiriman dua atau lebih trafik melalui rute-rute dengan komponen link yang beririsan, meskipun ada rure alternatif yang sedang tidak digunakan. Software Defined networking (SDN) memiliki kemampuan untuk membuat ECMP lebih dinamis dengan cara mengukur bandwidth yang tersisa dari setiap link dalam jaringan secara real-time oleh controller. Hasil pengukuran bandwidth yang dilakukan kemudian dijadikan dasar oleh controller untuk melakukan penentuan jalur yang dilewati sebuah trafik. Dalam tugas akhir ini, penulis membangun sebuah sistem penerapan ECMP berbasis SDN yang dapat mencegah terjadinya kongesti dengan melakukan pengukuran bandwidth setiap pilihan rute sebelum pengiriman setiap trafik, sehingga trafik-trafik yang ada sebisa mungkin tidak dilewatkan melalui rute-rute dengan komponen link yang sama (overlap). Skema yang diajukan terbukti menghasilkan nilai throughput 14.21% lebih tinggi dan delay 99% lebih rendah dibanding ECMP standar ketika terjadi kongesti serta memiliki nilai standar deviasi load 75.2 % lebih rendah dibanding load balancer round robin.

Kata kunci : ECMP, Overlapping, SDN, load balancing

***Abstract***

*Equal Cost Multipath Routing (ECMP) is a routing application where all available paths between two nodes is utilized by statically mapping each path to possible traffics between source and destination hosts in a network. This configuration can lead to congestion if there are two or more traffics being transmitted into paths with overlapping links, despite the availability of less busy paths.Software Defined Networking (SDN) has the ability to increase the dynamicity of ECMP by allowing controller to monitor available bandwidths of all links in the network in real-time. The measured bandwidth is then implemented as the basis of the calculation to determine which path a traffic will take. In this final project, a SDN-based ECMP application that can prevent network congestion is made by measuring available bandwidth of each available paths beforehand, thus making different traffics transmitted on non-overlapped paths as much as possible. The proposed increased the throughput by 14.21% and decreased the delay by 99% in comparison to standard ECMP when congestion occured and has 75.2% lower load standard deviation in comparison to round robin load balancer.*

*Keyword: ECMP, Overlapping, SDN, load balancing*

## 1. Pendahuluan

Equal Cost Multipath Routing (ECMP) adalah proses routing yang memanfaatkan seluruh pilihan jalur yang tersedia antara satu node dengan node lainnya dalam jaringan. Pada jaringan konvensional, jalur-jalur yang tersedia masing-masing akan dipasangkan dengan kemungkinan trafik yang ada (dengan hash function), dan tidak akan mengalami perubahan kecuali terjadi perubahan arsitektur (statis) [1] . Terbatasnya pilihan jalur akan menyebabkan beberapa trafik dipasangkan dengan jalur-jalur yang memiliki sebagian link yang beririsan (overlapping). Ketika setidaknya dua pasangan source-destination dengan jalur terpilih yang beririsan mengirimkan data secara bersamaan, maka kongesti akan terjadi, meskipun ada pilihan jalur lain yang dapat digunakan [2][3].

Pada Software Defined Networking (SDN), setiap switch terhubung dengan controller dan isi dari *routing table* pada *switch-swtich* tersebut akan ditentukan oleh controller [4]. Dengan kemampuan mengukur bandwidth dari setiap link yang ada dalam jaringan, controller dapat memilih jalur yang paling tidak sibuk untuk masing-masing trafik yang akan dikirimkan pada saat itu, sedemikian rupa sehingga kongesti dapat diminimalisasi.

Penelitian [2] dan [3] telah berhasil membuat sistem ECMP yang lebih dinamis dibandingkan penerapan ECMP pada jaringan konvensional. Penelitian [2] menggunakan controller floodlight yang berbasis Java, penelitian [3] melakukan pemrograman OpenFlow pada FPGA, sementara pada tugas akhir penulis menggunakan controller Ryu berbasis Bahasa pemrograman python.

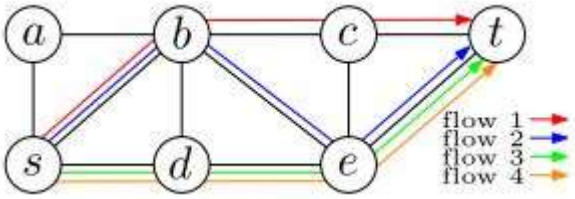
Pada penelitian [2], setiap link diberikan sebuah threshold. Ketika trafik pada suatu link melebihi threshold, maka beberapa trafik akan dialihkan ke jalur lain. Pada penelitian [3], seluruh trafik pada jaringan akan diukur. Ketika sebuah trafik memiliki ukuran yang lebih dari 10% dari kapasitas link, trafik tersebut dialihkan ke jalur lain. Sementara pada tugas akhir ini penulis akan mengusulkan skema alternatif yaitu dengan pemilihan jalur yang paling tidak sibuk terlebih dahulu oleh controller sebelum proses pengiriman data dilakukan.

Analisis yang dilakukan adalah pengukuran parameter-parameter Quality of Service (QoS) seperti delay, throughput, dan packet loss antara spanning tree protocol (STP), ECMP, dan ECMP berbasis SDN dengan skema yang diajukan pada saat pengiriman trafik-trafik yang dapat memicu terjadinya kongesti. Kemampuan *load balancing* dari sistem yang dibuat juga akan diukur dengan cara menghitung nilai standar deviasi *load*, throughput, dan packet loss saat pengiriman beberapa layanan VoIP.

## 2. Dasar Teori

### 2.1 Equal Cost Multipath Routing

*Equal Cost Multipath Routing* (ECMP) ialah salah satu teknik *routing* yang digunakan untuk memastikan seluruh jalur yang tersedia dengan *cost* sepadan antara satu *node* dengan *node* lainnya digunakan. [5]




Gambar 1. Ilustrasi ECMP[5]

Pada penerapan ECMP, setiap *router* akan menggunakan menggunakan *hash function* kepada setiap *header trafik* seperti IP pengirim dan IP tujuan untuk memetakan setiap *trafik* ke beberapa jalur yang tersedia.[1]. ECMP ditambahkan sebagai proses *load balancing* dari algoritma *routing Open Shortest Path First* (OSPF) pada jaringan konvensional. Implementasi ini didasarkan pada penggunaan algoritma Dijkstra dalam OSPF yang dapat dimodifikasi untuk menyimpan seluruh pilihan jalur antara satu *node* ke *node* lainnya. [10]. Proses *load balancing* yang terjadi bersifat statis terhadap kondisi jaringan karena isi tabel *routing* tidak akan berubah kecuali terjadi perubahan topologi.

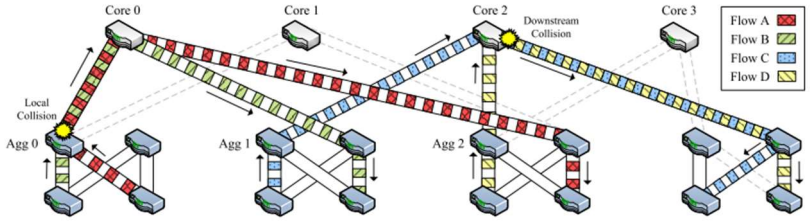



Figure 2: Examples of ECMP collisions resulting in reduced bisection bandwidth. Unused links omitted for clarity.

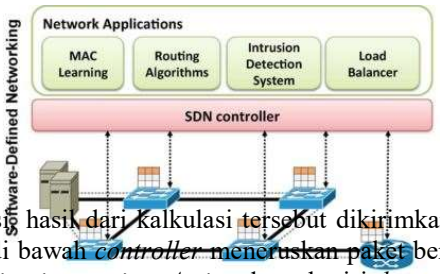
Gambar 2. Kongesti pada Penerapan ECMP [3]

Dikarenakan jumlah jalur yang terbatas dalam sebuah topologi, sangat memungkinkan dua atau lebih topologi dipasangkan dengan jalur-jalur yang memiliki *link* beririsan. Ketika *trafik-trafik* dikirimkan pada saat yang bersamaan, maka kongesti akan terjadi. Jalur-jalur lain yang seharusnya bisa digunakan untuk menghindari kongesti, tidak dipilih karena sudah dipasangkan dengan kemungkinan *trafik* lain oleh *hash function*.

### *2.2 Software Defined Networking [4]*

SDN merupakan sebuah paradigma baru dalam dunia jaringan yang memisahkan *control plane* dan *data plane,* baik secara *logical* ataupun secara *physical. Control plane* merupakan bagian yang berfungsi sebagai

"otak" dari sebuah jaringan, melakukan proses kalkulasi untuk fungsi seperti *routing, load balancer, intrusion detection system, topology learning,* dan lain sebagainya.

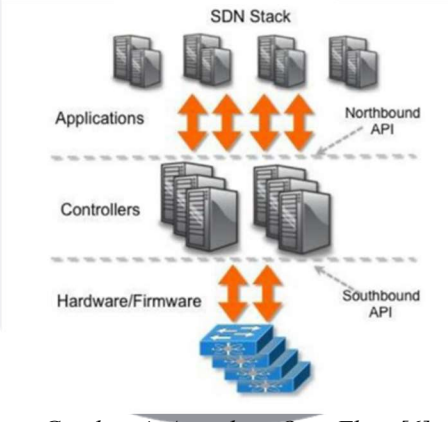



Gambar 3. Arsitektur SDN [5]

Setelah melakukan kalkulasi hasil dari kalkulasi tersebut dikirimkan ke *switch* untuk mengisi sebuah *trafik table. Switch* yang berada di bawah *controller* meneruskan paket berdasarkan isi dari *trafik table* yang umumnya terdiri dari *source, destination, action. Action* dapat berisi *drop, flood,* atau *out* ke nomor *port* yang telah ditentukan oleh *controller.* Jika ada paket dengan tujuan yang tidak ada dalam *trafik table,* maka *switch* akan mengirimkan *packet_in* ke *controller. S*etelah *controller* melakukan kalkulasi, *controller* akan melakukan pembaharuan *trafik table.* Dikarenakan fungsinya yang hanya meneruskan data tanpa melakukan kalkulasi, *swith-switch* yang terhubung dengan *controller* disebut dengan *data plane.*

### *2.3 Ryu Controller* dan OpenFlow [8][9]

*Ryu controller merupakan salah satu jenis controller SDN yang menggunakan bahasa pemrograman python sebagai dasar pengembangannya. Ryu dapat berkomunikasi dengan switch-switch yang terhubung dengan menggunakan protokol NETCONF, OF-Config, OpenFlow 1.0,1.1,1.3, 1.4, dan 1.5.*
*Jika menggunakan protokol OpenFlow, controller Ryu memiliki beberapa fungsi utama seperti:*
*- Packet-In Handler : Digunakan untuk menerima packet-in dari setiap switch, untuk kemudian di ekstraksi datanya (contoh: ip address asal dan ip address tujuan) dan dilakukan kalkulasi yang dibutuhkan.*
*- Datapath.send_msg: Digunakan untuk mengirimkan instruksi terhadap switch melalui protokol OpenFlow. 2 instruksi yang sering digunakan pada tugas akhir ini adalah OFPTrafikMod untuk memperbaharui trafik table pada switch, dan OFPTrafikStatsRequest untuk meminta kondisi jaringan dari setiap switch.*
*- EventSwitchEnter dan EventLinkAdd: Digunakan untuk mengetahui identitas switch dan pemetaan link antar switch pada saat switch pertama kali terhubung dengan controller.*
*- TrafikStatsReply Handler: Digunakan untuk menampung laporan kondisi jaringan setelah mengirim TrafikStatsRequest.*




*Gambar 4. Arsitektur OpenFlow [6]*

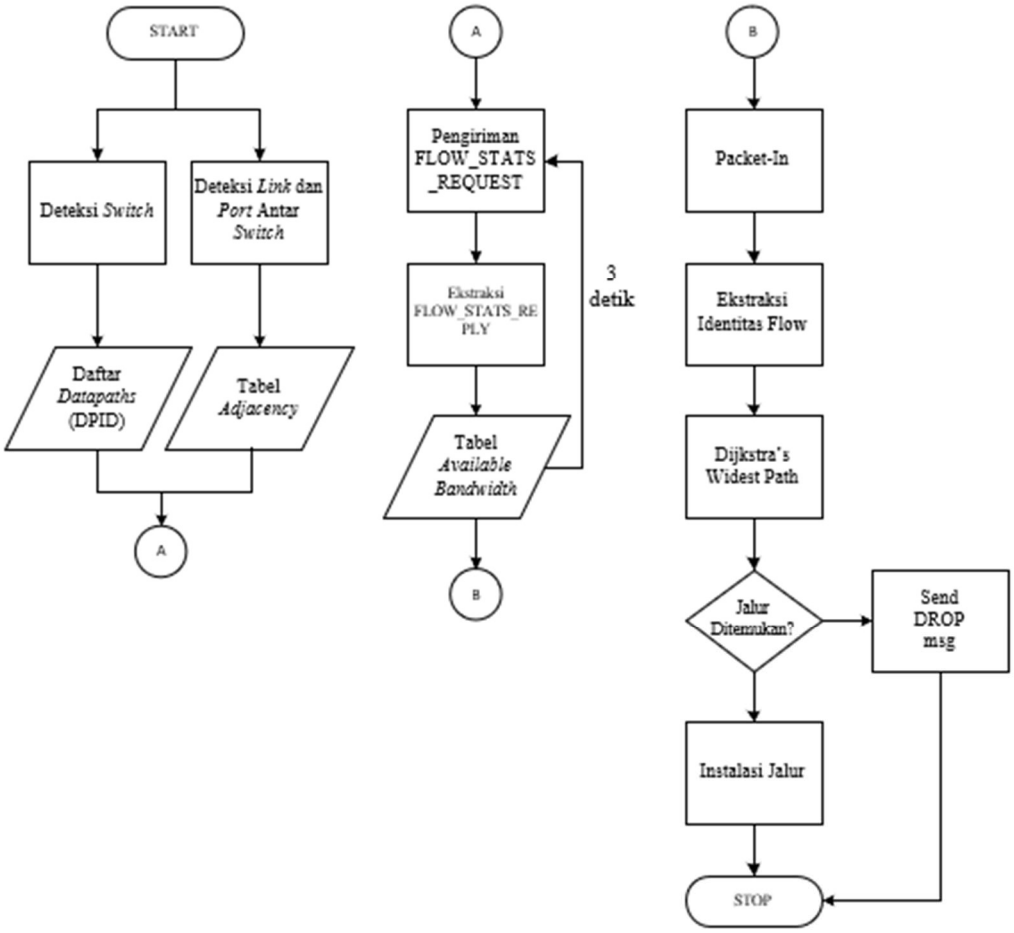
*Trafik table yang dimiliki oleh setiap switch juga didasarkan pada protokol OpenFlow. Pada protokol OpenFlow 1.3 yang digunakan pada penelitian ini, format untuk setiap trafik dalam trafik table terdiri dari 6 komponen utama, yaitu:*
*- Header fields – digunakan untuk mencocokan paket sesuai dengan hasil ekstraksi header paket seperti : port_in, alamat Ethernet pengirim, alamat Ethernet penerima, alamat IP, protokol IP, TCP/UDP.*
*- Counters – statistik dari setiap trafik setiap jumlah paket yang telah cocok (match) dengan kriteria trafik tersebut, durasi trafik, dan lain sebagainya.*
*- Actions – aksi yang diterapkan ketika proses pencocokan berhasil. Jenis aksi yang tersedia antara lain adalah out untuk meneruskan paket ke sebuah port tertentu, flood untuk broadcast paket, drop untuk membuang paket, serta controller untuk mengirimkan packet-in menuju controller.*
*- Priority – untuk memberikan nilai prioritas untuk setiap trafik yang ada dalam trafik table*

*- Timeout – Idle timeout digunakan untuk menentukan berapa lama aturan trafik akan dihapus setelah tidak terjadi pencocokan, dan hard timeout digunakan untuk menentukan waktu yang diizinkan sebelum sebuah trafik dihapus dari trafik table, terlepas dari adanya pencocokan atau tidak.*

## 3. Pembahasan

### 3.1 Desain Model Sistem

Dalam tugas akhir ini akan dibuat pemodelan sistem ECMP berbasis SDN yang dapat melakukan pencarian jalur dengan *bandwidth* tersisa maksimum dari seluruh pilihan jalur dari satu *node* menuju *node* lainnya untuk menghindari terjadinya kongesti. Secara umum tugas akhir ini memiliki skema sebagai berikut :



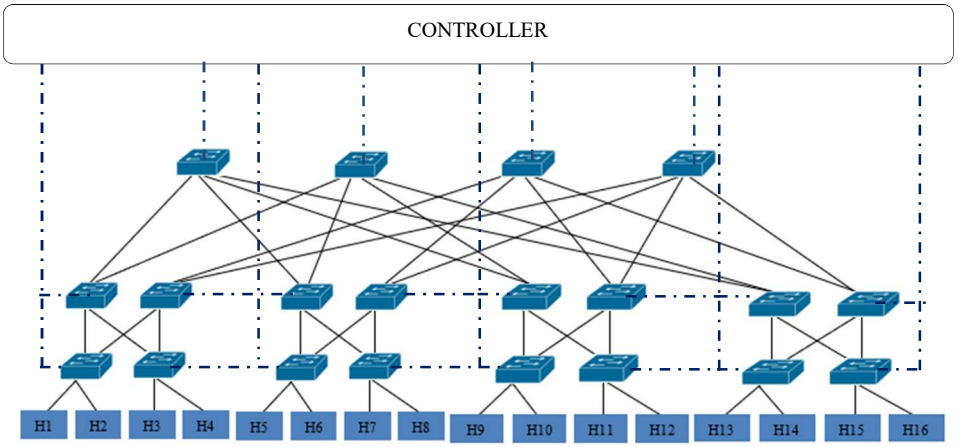

Gambar 5. Diagram alir sistem ADT

Dapat dilihat pada gambar 5 bahwa sistem yang dibuat terdiri dari 3 komponen program dalam *controller* yang terdiri dari (sesuai urutan eksekusi):

a. Komponen topologi *learning* yang bertugas untuk memetakan topologi dari *switch-switch* dalam jaringan yang terhubung dengan *controller* pada saat *mininet* (perangkat lunak untuk emulasi jaringan) pertama kali dihidupkan.
b. Komponen *network monitoring* yang bertugas untuk mendapatkan informasi trafik dari setiap *link* dalam jaringan
c. Komponen program utama yang bertugas menerima dan memproses permintaan pencarian jalur dari *switch* berdasarkan informasi dari komponen topologi *learning* dan *network monitoring*.

### 3.2. Topologi Jaringan

Topologi jaringan yang digunakan adalah topologi umum pada data *centre* yaitu topologi *fattree* dengan ilustrasi sebagai berikut:




Gambar *6*. Hasil Pengukuran QoS Pengujian Pemilihan Jalur

### 3.3. Skenario Evaluasi

Pengujian sistem yang dibuat dilakukan dengan 2 skenario evaluasi yaitu pengujian pemilihan jalur dan pengukuran nilai standar deviasi *load*. Penjelasan kedua skenario tersebut adalah sebagai berikut:

***a. Pengujian Pemilihan Jalur***

Pengujian pertama dilakukan dengan cara mengirimkan 3 buah trafik data berukuran 4.5 Mbit/s pada *link-link* 8 Mbit/s pada saat penerapan STP, ECMP dan ECMP SDN untuk mengetahui seberapa besar nilai QoS membaik ketika kongesti akibat pemilihan jalur berhasil/tidak berhasil dihindari. Parameter yang diukur adalah *throughput, delay,* dan *packet loss.*

***b. Pengukuran Nilai* Standar Deviasi *Load***

Pengujian kedua dilakukan dengan cara mengirimkan 4 buah layanan VoIP antara 2 buah client dan diukur nilai throughput, packet loss, dan standar deviasi loadnya untuk mengetahui seberapa baik beban didistribusikan dalam jaringan dibandingkan dengan algoritma l*oad balancing round robin*. Pengukuran nilai standar deviasi *load* diukur dengan rumus berikut:

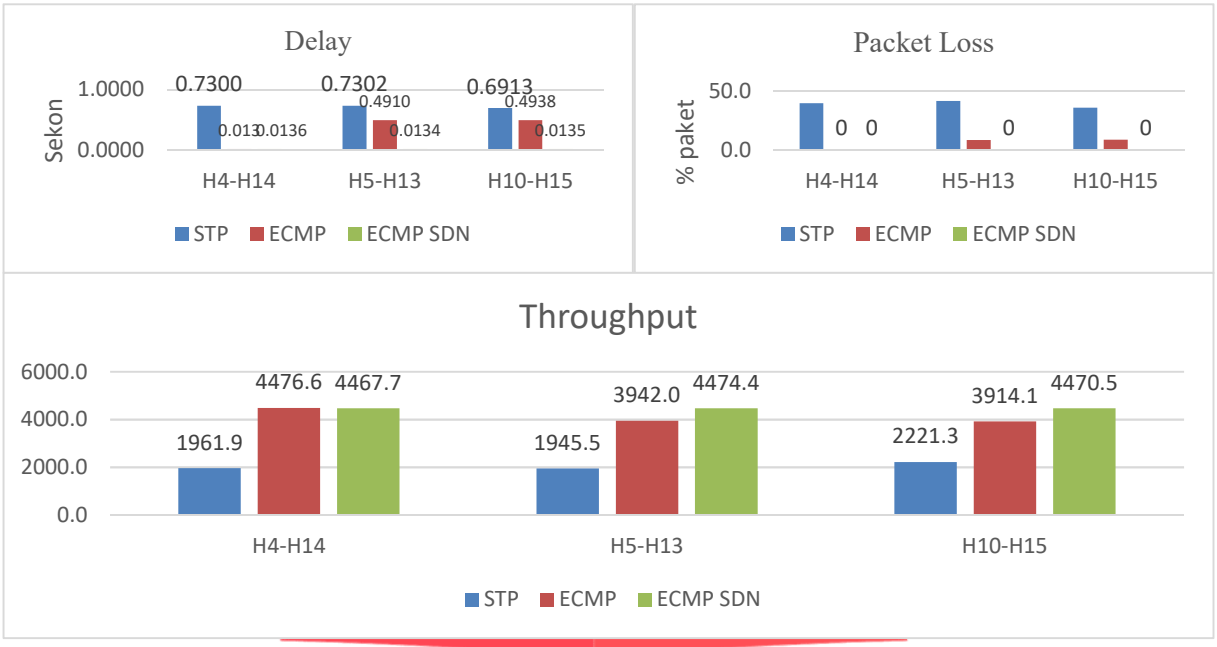
$$LSD = \sqrt{\frac{\sum_{i=1}^{n}(load_i - \overline{load})^2}{n}} \quad (1)$$

Dimana *load* menyatakan nilai *throughput* dari setiap koneksi VoIP dan n menyatakan banyaknya koneksi VoIP yang ditransmisikan.

### 3.4. Hasil Sistem

Hasil yang didapat dari skenario pengujian pemilihan jalur dan pengukuran standar deviasi *load* adalah sebagai berikut:

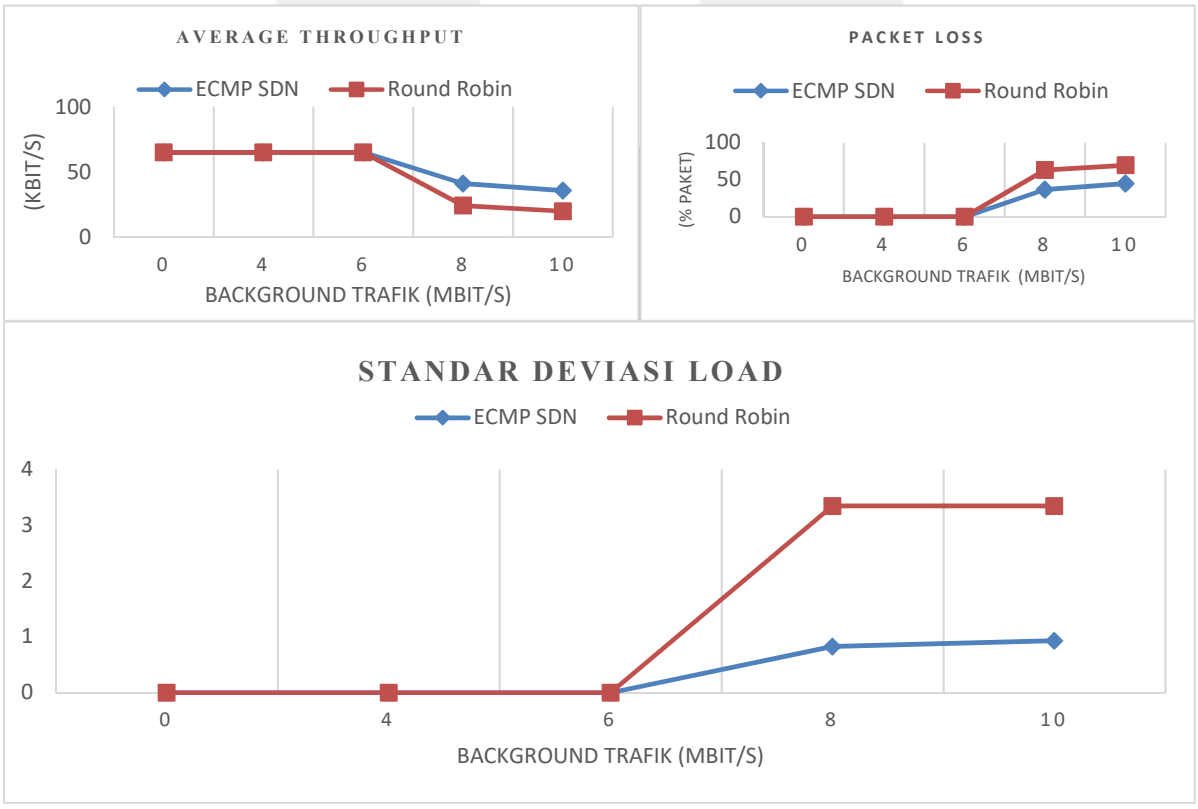
***a. Pengujian Pemilihan Jalur***




Gambar 7. *Hasil Pengukuran QoS Pengujian Pemilihan Jalur*

Pada Gambar 7 merupakan hasil pengukuran *throughput, delay,* dan *packet loss* saat mengirimkan 3 buah trafik data 4.5 Mbps melalui *link* 8 Mbps. Dengan keberhasilannya mencegah terjadinya kongesti, nilai *throughput* dari ECMP SDN untuk seluruh trafik stabil mendekati 4.5 Mbit/s (dengan packet loss 0 dan delay ~0.013 sekon), lebih baik dari STP dengan throughput terendah bernilai 1.9 Mbit/s (dengan packet loss mencapai 41.5% dan delay 0.73 sekon) dan ECMP dengan throughput terendah 3.9 Mbit/s (dengan packet loss mencapai 8.7% dan delay 0.49 sekon). Selisih nilai delay ECMP SDN dengan nilai ECMP biasa ketika tidak terjadi kongesti hanya mencapai 0.006 sekon, sehingga dapat ditarik kesimpulan penambah proses sebelum pemilihan jalur tidak mempengaruhi delay dari trafik yang dikirimkan secara signifikan.

### *c. Pengukuran Nilai* Standar Deviasi *Load*




Gambar 8. Hasil Pengukuran Standar Deviasi Load

Pada Gambar 8 merupakan hasil pengukuran standar deviasi *load* dari 4 layanan VoIP saat penerapan ECMP SDN dan *round robin*. Sebuah sistem dinyatakan dapat mendistribusikan beban dengan baik jika nilai standar deviasi loadnya baik jika dan hanya jika nilai throughput dan packet lossnya juga baik [7]. Program ECMP SDN yang dibuat terbukti lebih baik dari round robin, dibuktikan dengan nilai standar deviasi load yang lebih kecil 75.2% (dengan throughput 71% lebih besar dan packet loss 26.36% lebih rendah) saat background trafik 8 Mbit dan 72.13% (dengan throughput 80.5% lebih besar dan packet loss 24.35% lebih rendah) saat saat background trafik 10 Mbit/s.

## 4. Kesimpulan

Setelah sistem dibuat dan dilakukan analisis pada sistem berdasarkan skenario ditentukan, maka dapat diambil beberapa kesimpulan sebagai berikut:

- Skema yang diajukan berhasil menghindari kongesti dengan melakukan pengukuran *bandwidth* dari setiap *link* dalam jaringan sebelum trafik dikirimkan
- Pengujian pertama dilakukan dengan mengirimkan 3 trafik data 4.5 Mbit/s pada *link-link* 8 Mbit/s dengan hasil sebagai berikut:
  - Ketika terjadi kongesti, ECMP SDN memiliki nilai *throughput 14.21%* lebih tinggi dan *delay 99%* lebih rendah dibanding ECMP standar
  - Ketika tidak terjadi kongesti, ECMP SDN sedikit kurang efektif dibandingkan ECMP standar (penurunan *throughput* sebesar 0.19% dan peningkatan sebesar 3.46% pada *delay*) akibat adanya proses kalkulasi tambahan sebelum pengiriman
- Pengujian kedua dilakukan dengan cara mengirimkan 4 layanan VoIP antara 2 *client* bersamaan dengan *background traffic* untuk mengukur standar deviasi *load*. ECMP SDN terbukti memiliki nilai LSD 75.2% lebih kecil dibanding *load balancer round robin*.

**Daftar Pustaka:**


[1] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," *Doc. RFC 2992, IETF*, pp. 1–8, 2000.

[2] Hailong Zhang, Xiao Guo, Jinyao Yan, Bo Liu, and Qianjun Shuai, "SDN-based ECMP algorithm for data center networks," *2014 IEEE Comput. Commun. IT Appl. Conf.*, pp. 13–18, 2014.

[3] M. Al-Fares, S. Radhakrishnan, and B. Raghavan, "Hedera: Dynamic Trafik Scheduling for Data Center Networks.," *Nsdi*, p. 19, 2010.

[4] F. Ieee *et al.*, "Software-Defined Networking : A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] M. Chiesa, G. Kindler, and M. Schapira, "Trafik Engineering with {ECMP}: An Algorithmic Perspective," *Proc. IEEE INFOCOM*, vol. 25, no. 2, pp. 1590–1598, 2014.

[6] A. Martins, "Critical Ethernet based on OpenFlow," 2014.

[7] Y. L. Lan, K. Wang, and Y. H. Hsu, "Dynamic load-balanced path optimization in SDN-based data center networks," *2016 10th Int. Symp. Commun. Syst. Networks Digit. Signal Process. CSNDSP 2016*, pp. 0–5, 2016.

[8] Getting Started - Ryu 4.2.4 documentation. [Online]. https://ryu.readthedocs.io/en/latest/getting_started.html#what-s-ryu

[9] Open Networking Foundation. (2009) OpenFlow Switch Specification 1.0.0. [Online]. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/

[10] Cisco. ECMP Load Balancing. [Online]. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mp_l3_vpns/configuration/xe-3s/asr903/mp-l3-vpns-xe-3s-asr903-book/mp-l3-vpns-xe-3s-asr903-book_chapter_0100.pdf