InComTech: Jurnal Telekomunikasi dan Komputer
vol.11, no.1, April 2021, 73-84
http://publikasi.mercubuana.ac.id/index.php/Incomtech
P-ISSN: 2085-4811 E-ISSN: 2579-6089

Analisis Performansi *Routing* OSPF menggunakan RYU *Controller* dan POX *Controller* pada *Software Defined Networking*

Nanda Iryani, Afifah Dwi Ramadhani, Mayang Karmila Sari

Teknik Telekomunikasi, Institut Teknologi Telkom Purwokerto,
Jl. Panjaitan, Jawa Tengah 53417, Indonesia
*Email Penulis Koresponden: nanda@ittelkom-pwt.ac.id

***Abstrak*:**
Software Defined Networking (SDN) adalah teknologi baru yang dikembangkan untuk mengatasi masalah kompleksitas konfigurasi jaringan dengan pengelolaan lebih terpusat karena memisahkan antara *control plane* dan data *plane*. Penelitian ini menganalisis perbandingan performansi SDN menggunakan ryu *controller* dan pox *controller. Protocol* yang digunakan adalah *protocol* TCP dan UDP sedangkan *routing*nya menggunakan OSPF. Penerapan algoritma dijkstra dari perutean OSPF ke topologi *fat tree* pada SDN akan diukur unjuk kerjanya berdasarkan parameter *Quality of Service* yaitu *delay*, *jitter*, dan *packet loss* pada skenario tanpa *background traffic*. Transmisi data menggunakan *traffic protocol* TCP dengan RYU *controller* dan protokol UDP lebih baik karena memiliki *delay* sebesar 49.44% dan *delay* yang lebih stabil sebesar 0.01 %. *Jitter* yang dihasilkan adalah 27.59% lebih baik daripada POX *controller* sebesar 72.41% dan untuk protokol UDP menggunakan POX *controller* 99.97% sedangkan *traffic protocol* UDP menggunakan RYU *controller* lebih baik sebesar 0.03%. *Packet loss* dari kedua *controller* didapatkan hasil protokol TCP sangat bagus sebesar 0% sedangkan protokol UDP menggunakan POX *controller* 83.13%, menggunakan RYU *controller* 16.87%.

***Katakunci*:**
Pox *controller;*
Routing OSPF*;*
Ryu *controller;*
Software Defined Networking;
Topologi *fat tree;*

Riwayat Artikel:
Diserahkan 21 November 2020
Direvisi 8 Januari 2021
Diterima 14 Januari 2021
Dipublikasi 1 April 2021

DOI:
10.22441/incomtech.v11i1.10187

1. PENDAHULUAN

Jaringan komputer terdapat banyak komputer yang saling terhubung maka semakin kompleks konfigurasi yang dilakukan [1]. Mengatasi kompleksitas konfigurasi pada jaringan konvensional, dibuatlah teknologi *Software Defined Networking* (SDN) yang membuat konfigurasi menjadi terpusat dan lebih mudah untuk dikelola [2]. SDN adalah sebuah konsep pendekatan jaringan dengan membuat pengontrol dan arus data dipisahkan dengan perangkat kerasnya. Awal

mula terciptanya teknologi SDN dimulai setelah Sun Microsystems merilis java pada tahun 1995. Baru pada tahun 2008 SDN dikembangkan di UC Berkeley *and* Stanford University. Protokol *openflow* untuk komunikasi antara *controller* dengan perangkat jaringan [3, 4, 5, 6]. Protokol *openflow* sudah diimplementasi pada beberapa SDN *controller* seperti *controller* ONOS, Onix, POX, RYU, dan lainnya [7].

Penelitian ini akan menggunakan *controller* RYU dan POX kemudian diimplementasikan routing OSPF. OSPF adalah routing protokol berstandar terbuka yang berjenis IGRP. Protokol routing OSPF mendistribusikan informasi pada tabel routing untuk pencarian rute terpendek, OSPF dapat mencari jalur alternatif lainnya. menentukan rute terpendek menggunakan algoritma dijkstra pemilihan rute berdasarkan kapasitas trafik data yang dapat melewati rute tersebut [8]. Ada juga peneliti yang sudah melakukan penelitian tentang RouteFlow menggunakan *protocol* OSPF dengan hasil kategori baik menurut standarisasi ITU-TG.114 sedangkan pada penelitian kami akan membandingkan performansi dari dua *controller* yang digunakan yaitu *controller* RYU dan *controller* POX [2]. Teknologi SDN yang terus berkembang seiring dengan kebutuhan akan jaringan terutama dalam melakukan konfigurasi jaringan yang semakin kompleks maka performansi menjadi hal krusial yang harus diperhatikan. Tolak ukur pengukuran performansi pada jaringan dikenal dengan *Quality of Service* (QoS).

Penelitian dilakukan untuk mengetahui bagaimana performansi dari routing OSPF menggunakan RYU *controller* dan POX *controller* pada topologi jaringan *fat tree* berdasarkan parameter QoS yang berstandar TIPHON. Dari permasalahan di atas dan beberapa penjabaran tentang SDN, penulis memutuskan melakukan penelitian yang berjudul "Analisis performansi routing OSPF menggunakan ryu *controller* dan pox *controller* pada software defined networking". Tujuan dari penelitian ini adalah untuk menganalisa performansi dari RYU *controller* dan POX *controller* pada topologi jaringan fat tree berdasarkan parameter Quality of Service yaitu delay, jitter dan packet loss yang berstandar TIPHON.

2. METODE

2.1. Studi Literatur

Metode yang dilakukan dari beberapa jurnal yang digunakan sebagai acuan. Penelitian oleh Simarmata dkk menggunakan aplikasi GNS3 untuk mensimulasikan jaringan konvensional, dan mensimulasikan jaringan SDN menggunakan aplikasi Mininet dengan 8 *router* dan 8 *host* [3]. Penelitian kedua oleh Muldina dan Tulloh menggunakan topologi dengan jumlah switch berbeda dan berdasarkan parameter convergence time. Penambahan jumlah switch mengakibatkan penambahan convergence time [4]. Penelitian ketiga oleh Afan dan Virgono yang melakukan perbandingan antara RYU dan POX controller pada performansi jaringan SDN. Pada penelitian ini didapatkan hasil QoS [8].

Penelitian keempat oleh Reinhart dkk dilakukan pada 5 perangkat forwading plane yang terhubung dengan laptop yang terinstal POX controller di dalam VMware yang berfungsi sebagai Control Plane [9]. Penelitian terakhir oleh Irmawati dkk yang dilakukan dengan 4 switch yang terhubung dengan control plane sebagai pengendali sebuah jaringan [10].

2.2. Perangkat Lunak (*Software*)

Pada penelitian perangkat lunak yang digunakan sebagai *tool* dan aplikasi ditunjukkan pada Tabel 1.

Tabel 1. Tool dan Aplikasi

No	Nama *Software*	Versi	Fungsi
1	*Pox controller*	0.5.0	SDN *Controller*
2	*Ryu controller*	3.3.4	SDN *Controller*
3	*D-ITG*	2.8.1	*Traffic Generator*
4	*Openflow*	1.5	*Protocol SDN*
5	*Mininet*	2.2.2	*Emulator* jaringan

a. *Software defined network* adalah sebuah konsep pendekatan jaringan dengan membuat pengontrol dan arus data dipisahkan dengan perangkat kerasnya. Karakteristik jaringan SDN adalah menggabungkan *semua control plane* dari setiap perangkat menjadi satu *controller* yang *programmeable*. Agar tersentralisasi sehingga mempermudah untuk mendesain dan mengoperasikan jaringan [6].
b. Pox *controller* adalah perangkat lunak terbuka atau *open source*. Pox merupakan sebuah *platform* untuk pengembangan dan *prototyping* aplikasi jaringan yang cepat. *Pox diutamakan untuk suatu penelitian* [11].
c. *Ryu controller* adalah perangkat lunak terbuka atau open source. *Ryu* memiliki APIs yang memudahkan untuk manajemen perangkat router dan switch jaringan SDN [12].
d. D-ITG adalah platform yang mampu menghasilkan lalu lintas data di tingkat paket secara akurat mereplikasi proses stokastik yang tepat untuk kedua IDT *(inter departure time) dan* ps (*packet size*) variable juga mendukung generasi traffic ipv4 dan ipv6 dan mampu menghasilkan lalu lintas di lapisan jaringan transportasi dan aplikasi [13][14].
e. *Openflow* adalah jenis dari *application protocol interfaces* (APIs) di jaringan *software defined networking* (SDN) yang digunakan untuk pengontrolan atau mengatur *traffic flows* pada *switch* [15].

2.3. Parameter Penelitian

Quality of service didefinisikan sebagai kemampuan untuk menjamin kebutuhan jaringan tertentu dan kehandalan dalam memenuhi *service level agreement* (SLA) antara provider aplikasi dan *end user*. *Switch* dan *router* bisa menandai *traffic* sehingga memungkinkan administrator jaringan untuk memprioritaskan lalu lintas tertentu, dan memberikan kemampuan untuk mendefinisikan atribut layanan yang disediakan baik secara kualitatif maupun kuantitatif [16].

Delay adalah total waktu untuk mengirimkan satu paket data dari pengirim ke penerima. *Delay* yang sering dialami oleh *traffic* yang lewat adalah *delay* transmisi. Persamaan perhitungan *Delay*:

$$Delay(ms) = \frac{Packet\ Length}{Link\ bandwith} \quad (1)$$

Packet loss adalah parameter yang menjelaskan suatu kondisi jumlah total paket *yang* telah hilang selama pengiriman data karena *collision* dan *congestion* [17].

Tabel 2. *Delay*/Latensi [16]

Kategori	Besar Delay	Indeks
Sangat memuaskan	< 150 ms	4
Memuaskan	150 s/d 300 ms	3
Kurang memuaskan	300 ms s/d 450 ms	2
Tidak memuaskan	>450 ms	1

Tabel 3. *Packet loss* [16]

Kategori	Besar Packet loss	Indeks
Sangat bagus	0	4
Bagus	3	3
Sedang	15	2
Jelek	25	1

Persamaan perhitungan *Packet loss:*

$$Packet\ Los(\%) = \frac{(\text{Paket data dikirim} - \text{paket diterima})\ \text{x}\ 100\%}{\text{Paket data yang dikirim}} \quad (2)$$

Jitter adalah variasi dari suatu *delay end-to-end*. Level yang tinggi pada suatu *jitter pada* aplikasi berbasis UDP adalah situasi yang tidak dapat diterima aplikasi *real-time*, seperti audio dan video [17].

Tabel 4. *Jitter* [17]

Kategori	Besar Jitter	Indeks
Sangat bagus	0 ms	4
Bagus	0 ms s/d 75 ms	3
Sedang	75 ms s/d 125 ms	2
Jelek	126 ms s/d 225 ms	1

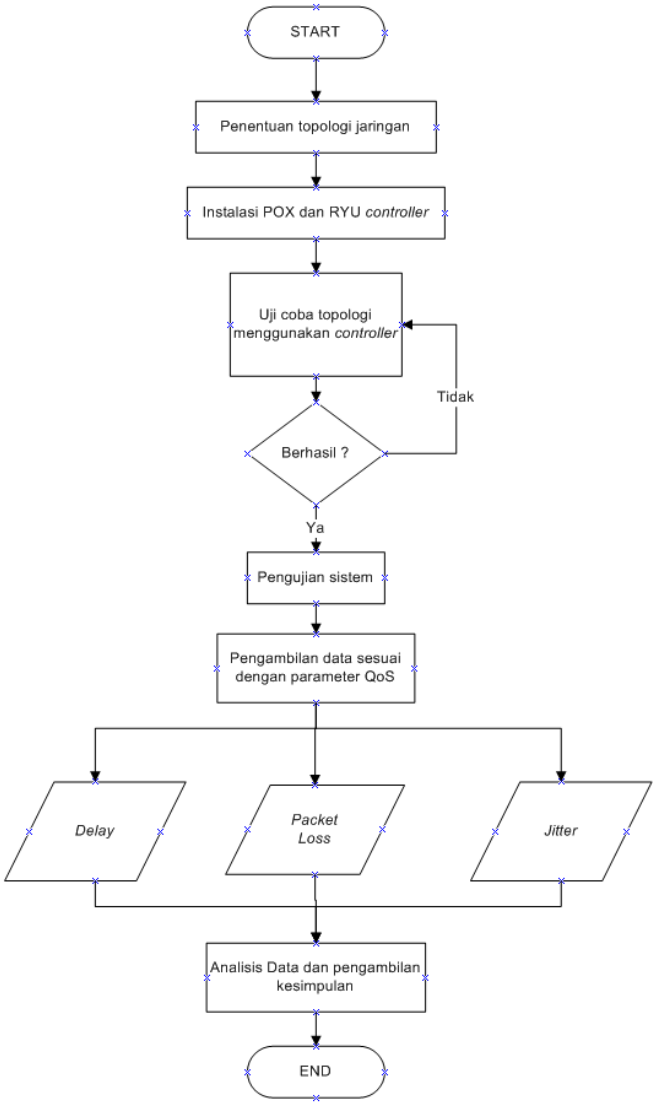
Persamaan perhitungan *Jitter:*

$$Jitter(ms) = \frac{\text{Total variasi } delay}{\text{Total paket yang diterima}} \quad (3)$$

2.3. Alur Penelitian

Penelitian ini memiliki alur yang harus dilakukan agar mendapatkan hasil yang diinginkan. Alur penelitian ini terdapat pada Gambar 1.

Gambar 1 menunjukkan alur penelitian pada perancangan sistem agar hasil dari penelitian dapat dicapai dengan baik. Dimulai dari penentuan topologi jaringan performansi *routing* OSPF menggunakan RYU dan POX *controller* pada jaringan SDN [18, 19, 20]. Selanjutnya dilakukan instalasi SDN *controller* RYU dan POX pada perangkat pengujian. Setelah itu dilanjutkan dengan uji coba topologi jaringan menggunakan RYU dan POX *controller*. Jika gagal maka uji coba topologi menggunakan RYU dan POX *controller* akan dilakukan ulang, Jika uji coba berhasil dilanjutkan dengan pengujian terhadap sistem *routing* OSPF pada RYU dan POX *controller*. selanjutnya mengambil data dengan parameter-parameter yang telah ditentukan yaitu berupa nilai dari parameter *delay*, *jitter*, dan *packet loss.*



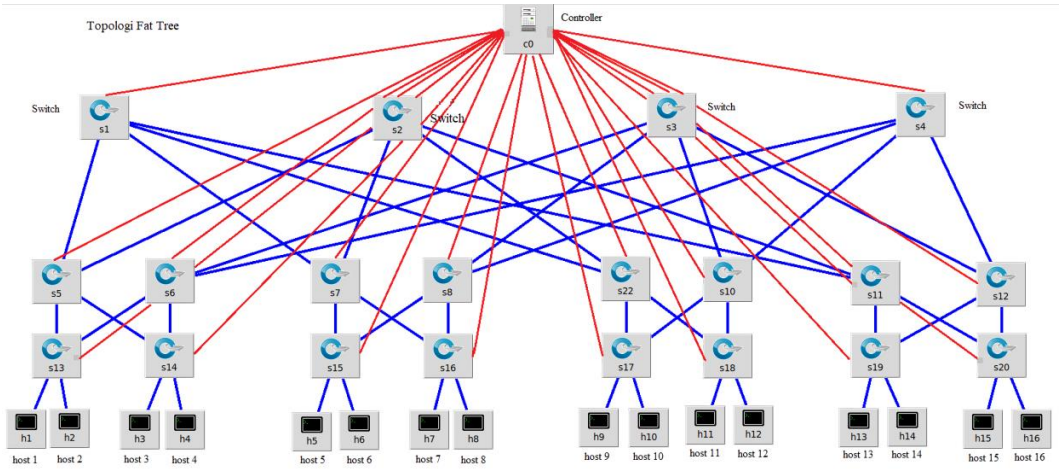
Gambar 1. Diagram Alur Penelitian

Tahapan terakhir adalah analisis terhadap data-data yang telah diperoleh untuk mengetahui performansi *routing* OSPF menggunakan RYU dan POX *controller* yang dijalankan pada *emulator* Mininet dengan menggunakan *Distributed Internet Traffic Generator* (D-ITG). Kesimpulan dilakukan dengan memperhatikan tujuan penelitian agar diperoleh hasil yang sesuai. Saran diberikan untuk mendorong pada penelitian lainnya yang akan mengambil tema serupa.

2.5. Skenario Penelitian

Topologi jaringan *fat tree* ditunjukkan pada Gambar 2. Topologi jaringan ini *dijalankan* dengan menggunakan *software emulator mininet.*

Pengujian bertujuan mendapatkan hasil performansi *routing* OSPF dari kedua *controller* tanpa adanya *background traffic*. Terdapat 4 skenario pengujian dengan pengujian pengiriman *packet transmission control protocol* (TCP) dan *User Data Protocol* (UDP) dari satu host ke *host* lainnya masing-masing skenario akan diuji coba sebanyak 30 kali, sehingga diambil nilai rata rata dari setiap parameter.



Gambar 2. Topologi Jaringan *Fat Tree*

Tabel 5. Tabel Skenario Penelitian

No	Host		Paket TCP yang dikirim		Waktu pengiriman	Banyaknya pengujian
	Tx	Rx	Ukuran	Jumlah		
1	h1	h16	1 Mbits	10.000.000 pps	15 s	30
2	h1	h16	10 Mbits	10.000.000 pps	15 s	30
3	h1	h2	1 Mbits	10.000.000 pps	15 s	30
4	h1	h2	10 Mbits	10.000.000 pps	15 s	30

3. HASIL DAN PEMBAHASAN

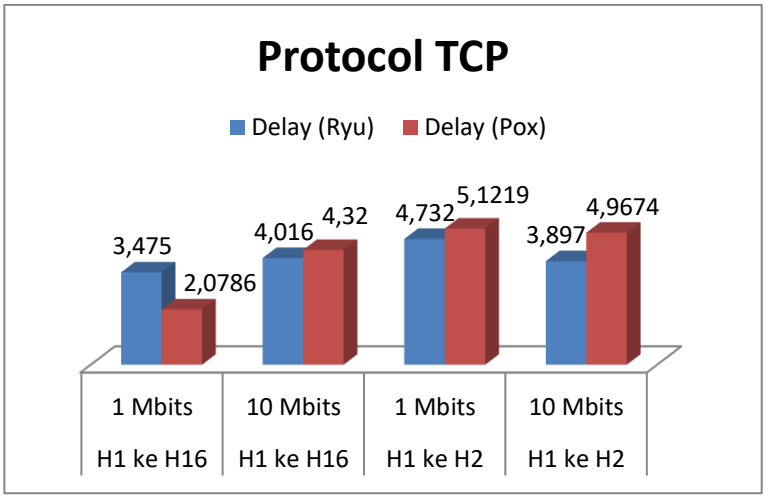
Penelitian ini melakukan analisis performansi *routing* OSPF menggunakan controller ryu dan *controller* pox pada *software defined networking.* Pengukuran unjuk kerja dilakukan dengan menganalisa *traffic* protokol UDP dan TCP yang berjalan selama 15 detik pada topologi jaringan uji tanpa *background traffic*. Pengambilan data uji dilakukan menggunakan empat skenario *seperti* pada Tabel 5. *Traffic* data yang dikirimkan dibangkitkan dengan menggunakan D-ITG. Parameter kerja yang digunakan adalah *delay*, *jitter* dan *packet loss.* Hasil pengukuran kemudian akan dianalisis menggunakan standarisasi yang di keluarkan ETSI (TIPHON).

3.1. Pengukuran *Delay*

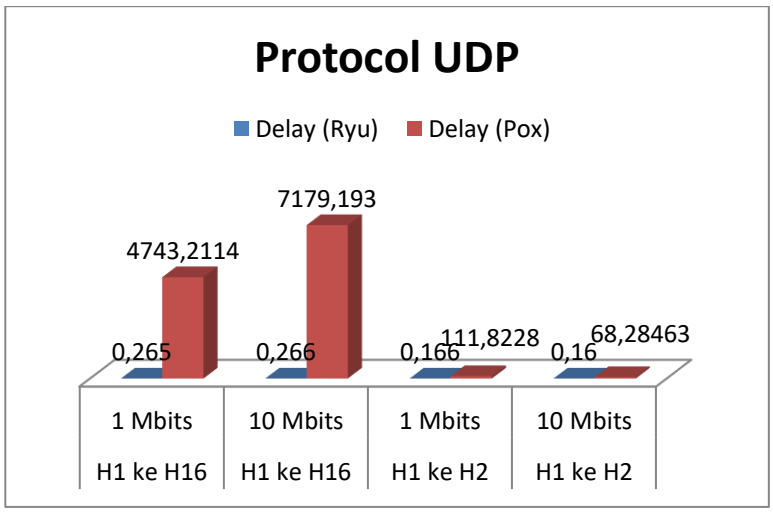
Delay merupakan waktu yang dibutuhkan pengiriman paket sampai pada *penerima*. Untuk standar nilai parameter *delay* yang dapat dilihat pada Tabel 2.

Hasil pengukuran parameter *delay* untuk data uji coba 4 skenario di 2 *controller* pada *traffic protocol* TCP ditampilkan pada Gambar 3. dimana untuk *traffic protokol* TCP skenario pertama menghasilkan nilai yang paling rendah daripada skenario lain nya. disini pengiriman menggunakan *traffic protocol* TCP menggunakan POX *controller* mendapatkan nilai rata-rata sebesar 2.0786 ms dan pengiriman menggunakan *traffic protocol* TCP menggunakan RYU *controller* mendapatkan nilai rata-rata sebesar 3.475 ms.

Setelah mengetahui hasil tersebut, berdasarkan standar nilai *delay*, baik untuk *pengiriman* menggunakan POX atau RYU *controller* mendapat kategori sangat memuaskan. Nilai rata-rata *delay* pada POX *controller* didapatkan 4.12 ms dan pada RYU *controller* didapatkan 4.03 ms.



Gambar 3. *Delay* pada *Protocol* TCP



Gambar 4. *Delay* pada *Protocol* UDP

Hal ini menjelaskan bahwa nilai yang didapatkan kurang dari 150 ms tetapi untuk pengiriman yang paling bagus digunakan adalah pada *traffic protocol* TCP menggunakan RYU *controller*.

Selanjutnya dilakukan pengukuran parameter *delay* untuk data uji coba 4 skenario di 2 *controller* pada *traffic protocol* UDP ditampilkan pada Gambar 4. dimana untuk *traffic protokol* UDP skenario keempat menghasilkan nilai yang paling rendah daripada skenario lain nya. disini pengiriman menggunakan *traffic protocol* UDP menggunakan POX *controller* mendapatkan nilai rata-rata sebesar 68.28463 ms dan pengiriman menggunakan *traffic protocol* UDP menggunakan RYU *controller* mendapatkan nilai rata-rata sebesar 0.16 ms.

Setelah mengetahui hasil tersebut, berdasarkan standar nilai *delay*, baik untuk pengiriman menggunakan POX atau RYU *controller* mendapat kategori sangat memuaskan karena nilai rata-rata *delay* pada POX *controller* didapatkan 3025.6 ms dan pada RYU *controller* didapatkan 0.21 ms dan itu menjelaskan nilai yang didapatkan kurang dari 150 ms tetapi untuk pengiriman yang paling bagus digunakan adalah pada traffic protocol UDP menggunakan RYU controller.

Berdasarkan hasil dari parameter delay dari skenario 1 hingga skenario 4 dengan menggunakan perbandingan keseluruhan nilai delay yang didapat maka diperoleh untuk POX controller mendapatkan nilai delay pada protokol TCP sebesar 50.56% sedangkan pada RYU *controller* mendapatkan nilai *delay* pada protokol TCP sebesar 49.44%. Pada *protocol* UDP menggunakan POX *controller* mendapatkan nilai *delay* 99.99%. Sedangkan pada *traffic protocol* UDP menggunakan RYU *controller* didapatkan nilai 0.01%. Namun pada POX *controller* terkadang terdapat *nilai delay yang* tiba-tiba melonjak tinggi sehingga dapat dikatakan tidak *reliable* untuk transmisi data menggunakan *traffic protocol* UDP menggunakan POX *controller*. Oleh karena itu, dapat disimpulkan meskipun nilai *delay* yang dihasilkan pada kedua *controller* dapat dikatakan bagus, namun untuk transmisi data menggunakan *traffic protocol* TCP menggunakan RYU *controller* mendapatkan 49.44% ataupun UDP menggunakan RYU *controller* mendapatkan 0.01% itu adalah nilai yang lebih baik karena memiliki nilai *delay* yang lebih kecil dan lebih stabil daripada menggunakan POX *controller*.

3.2. Pengukuran *Jitter*

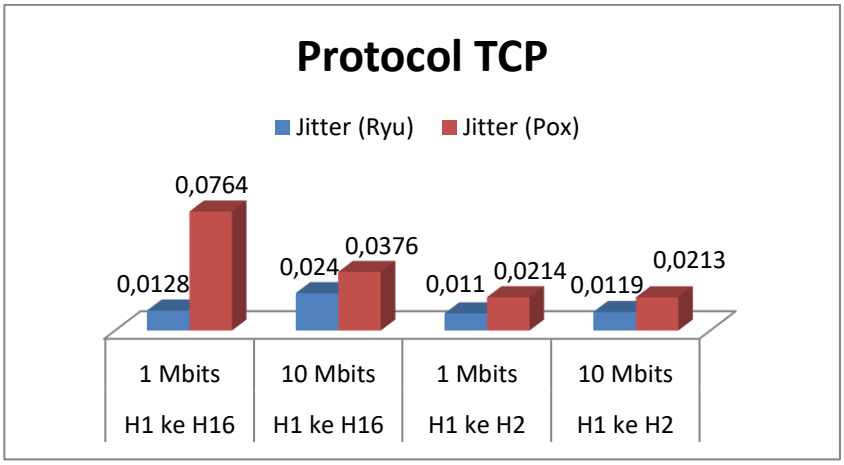
Pengukuran *jitter* dilakukan dengan membangkitkan trafik data dari pengirim ke penerima menggunakan software D-ITG. Untuk standar nilai parameter jitter yang dapat dilihat pada Tabel 4.

Hasil pengukuran parameter jitter untuk data uji coba 4 skenario di 2 controller pada traffic protocol TCP ditampilkan pada Gambar 5. dimana untuk traffic protocol TCP skenario ketiga menghasilkan nilai yang paling rendah daripada skenario lain nya. disini pengiriman menggunakan traffic protocol TCP menggunakan POX *controller* mendapatkan nilai rata-rata sebesar 0.0214 ms dan pengiriman dengan *traffic protocol* TCP menggunakan RYU *controller* mendapatkan nilai rata-rata sebesar 0.011 ms.

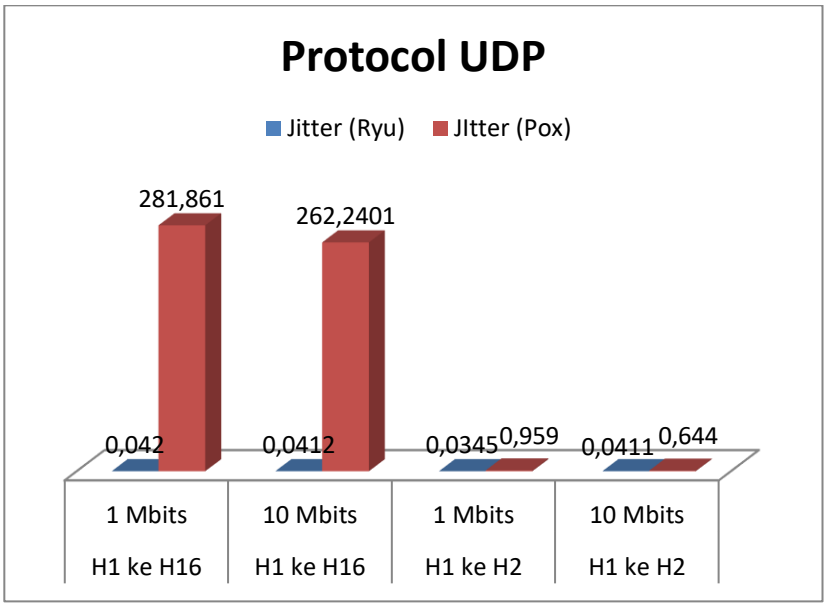
Setelah mengetahui hasil tersebut, berdasarkan standar nilai *jitter*, baik untuk pengiriman menggunakan POX atau RYU *controller* mendapat kategori bagus karena nilai rata-rata jitter pada POX *controller* didapatkan 0.03 ms dan pada RYU *controller* didapatkan 0.019 ms dan itu menjelaskan nilai yang didapatkan kurang dari 75 ms tetapi untuk pengiriman yang paling bagus digunakan adalah pada traffic protocol TCP menggunakan RYU controller.

Selanjutnya dilakukan pengukuran parameter jitter untuk data uji coba 4 skenario di 2 controller pada traffic protocol UDP ditampilkan pada Gambar 6. dimana untuk traffic protokol UDP skenario keempat menghasilkan nilai yang paling rendah daripada skenario lain nya. disini pengiriman menggunakan *traffic protocol* UDP menggunakan POX *controller* mendapatkan nilai rata-rata sebesar 0.644 ms dan pengiriman menggunakan *traffic protocol* UDP menggunakan RYU *controller* mendapatkan nilai rata-rata sebesar 0.0411 ms.

Setelah mengetahui hasil tersebut, berdasarkan standar nilai *jitter*, untuk pengiriman pada *traffic protocol* UDP menggunakan RYU *controller* mendapat pada POX *controller* didapatkan 136.42 ms itu menjelaskan nilai rata-rata *jitter* yang didapatkan lebih dari 75 ms dan pada RYU *controller* didapatkan 0.039 ms dan itu menjelaskan nilai rata-rata jitter yang didapatkan kurang dari 75 ms, dan untuk pengiriman yang paling bagus digunakan adalah pada traffic protocol UDP menggunakan RYU controller.



Gambar 5. *Jitter* pada *Protocol* TCP



Gambar 6. *Jitter* pada *Protocol* UDP

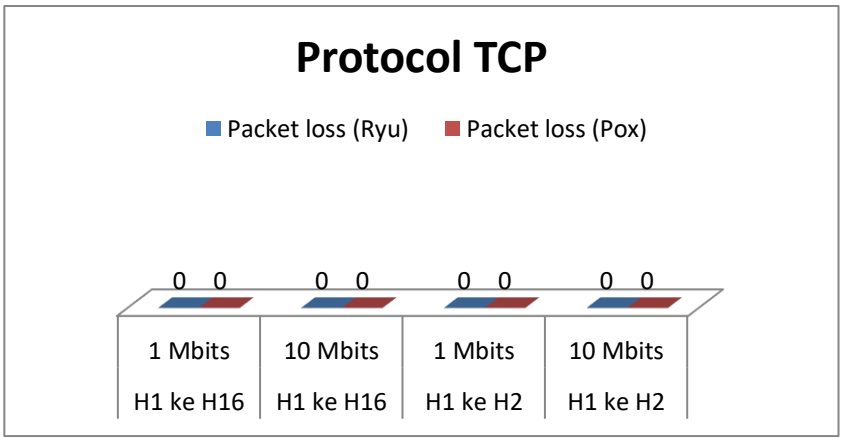
Berdasarkan hasil dari parameter jitter yang didapat dari skenario 1 hingga skenario 4 dengan menggunakan POX controller mendapatkan nilai jitter pada protocol TCP sebesar 72.41% sedangkan RYU controller mendapatkan nilai jitter pada protocol TCP sebesar 27.59%. Pada traffic protocol UDP menggunakan POX controller mendapatkan sebesar 99.97%. Sedangkan pada traffic *protocol* UDP menggunakan RYU *controller* sebesar 0.03%. Namun pada POX *controller* pada *traffic protocol* UDP terkadang terdapat nilai *jitter* yang dihasilkan tiba-tiba melonjak tinggi sehingga dapat dikatakan tidak *reliable* untuk transmisi data menggunakan *traffic protokol* UDP menggunakan POX *controller*. Oleh karena itu, dapat disimpulkan bahwa transmisi data menggunakan *traffic protocol* TCP menggunakan RYU *controller* sebesar 0.0597 ms dan UDP menggunakan RYU *controller* sebesar 0.1588 ms dan ini menunjukkan menggunakan RYU *controller* memiliki nilai *jitter* yang lebih kecil daripada menggunakan POX *controller*.

3.3. Pengukuran Packet Loss

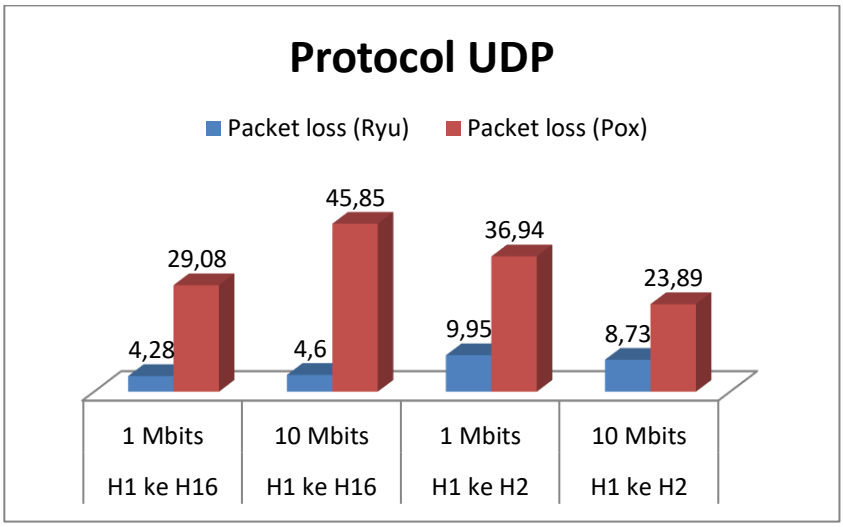
Packet loss merupakan banyaknya paket yang hilang selama proses transimisi ke tujuan dilakukan. Paket hilang terjadi ketika satu atau lebih paket data yang melewati suatu jaringan gagal mencapai tujuannya. Untuk standar nilai parameter delay yang digunakan dapat dilihat pada Tabel 3.

Hasil pengukuran parameter packet loss untuk data uji coba 4 skenario di 2 controller pada traffic protocol TCP ditampilkan pada Gambar 7. disini pengiriman menggunakan *traffic protocol* TCP menggunakan POX *controller* ataupun RYU *controller* mendapatkan kategori nilai sama yaitu 0%. Setelah mengetahui hasil tersebut, berdasarkan standar nilai packet loss, untuk traffic protokol TCP menggunakan POX controller ataupun RYU controller mendapatkan kategori sangat bagus karena mendapatkan nilai 0 sebesar 100%.

Selanjutnya dilakukan pengukuran parameter packet loss untuk data uji coba 4 skenario, di 2 controller pada traffic protocol UDP ditampilkan pada Gambar 8. disini pengiriman menggunakan traffic protocol UDP menggunakan POX controller mendapatkan nilai rata-rata terendah sebesar 4.28 ms di skenario pertama dan pengiriman menggunakan *traffic protocol* UDP menggunakan RYU *controller* mendapatkan nilai rata-rata terendah sebesar 23.89 ms.



Gambar 7. *Packet loss* pada *Protocol* TCP



Gambar 8. *Packet loss* pada *Protocol* UDP

Berdasarkan standar nilai *packet loss*, untuk pengiriman menggunakan RYU *controller* mendapat kategori sedang, karena nilai rata-rata *packet loss* yang didapatkan 6.89 %. Sedangkan pengiriman menggunakan POX *controller* mendapatkan kategori jelek, karena nilai rata rata *packet loss* yang didapatkan sebesar 33.94%. Pengiriman yang paling bagus digunakan adalah pada *traffic protocol* UDP menggunakan RYU controller.

Berdasarkan pengujian hasil packet *loss* menggunakan POX controller mendapatkan nilai pada *protocol* TCP sebesar 0% dan RYU *controller* mendapatkan nilai pada *protocol* TCP sebesar 0% dan *traffic protocol* UDP menggunakan POX *controller* mendapatkan nilai 83.13%. Sedangkan pengiriman menggunakan RYU *controller* mendapatkan nilai sebesar 16.87%. Oleh karena itu, dapat disimpulkan bahwa nilai *packet loss* yang digunakan untuk transmisi data menggunakan *traffic protocol* TCP dengan RYU *controller* ataupun POX *controller* baik karena memiliki nilai *packet loss* 0%. sedangkan untuk transmisi data menggunakan *traffic protocol* UDP menggunakan RYU *controller* lebih baik karena memiliki nilai *packet loss* yang lebih kecil yaitu sebesar 16.87% daripada menggunakan POX *controller*.

4. KESIMPULAN

Routing OSPF dapat diimplementasikan di jaringan SDN dengan menggunakan algoritma djikstra pada RYU *controller* dan POX *controller* untuk melakukan pemilih rute pengiriman data. Transmisi data menggunakan *traffic protocol* TCP menggunakan RYU *controller* lebih baik karena memiliki nilai *delay* sebesar 49.44% dan untuk protokol UDP lebih baik juga karena memiliki nilai *delay* yang lebih stabil sebesar 0.01 %. Nilai *jitter* yang dihasilkan menggunakan *traffic protocol* TCP menggunakan RYU *controller* mendapatkan nilai 27.59% lebih baik daripada POX *controller* yang mendapatkan nilai sebesar 72.41% dan untuk protokol UDP menggunakan POX *controller* mendapatkan nilai 99.97% sedangkan *traffic protocol* UDP menggunakan RYU *controller* lebih baik karena mendapatkan nilai sebesar 0.03%. Nilai *packet loss* dari kedua *controller* didapatkan hasil protokol TCP sangat bagus karena mendapatkan nilai sebesar 0% sedangkan protokol UDP menggunakan POX *controller* mendapatkan nilai sebesar 83.13%, menggunakan RYU *controller* mendapatkan nilai sebesar 16.87%. Dari parameter *delay, jitter, dan packet loss* dihasilkan perbandingan dan dapat disimpulkan POX *controller* kurang bagus untuk *traffic protocol* UDP karena memiliki jitter yang tinggi dan *delay* yang sangat fluktuatif yang berarti koneksi tidak *reliable* dan tidak disarankan karena memiliki nilai jelek yaitu sebesar 99.65% untuk mentransmisikan data. Sedangkan RYU *controller* lebih bagus untuk *traffic protocol* TCP maupun UDP karena memiliki nilai sebesar 0.35%, sangat bagus untuk *traffic protocol* UDP karena *delay* kecil dan stabil.

REFERENSI

[1] B. S. Elenbogen, "Computer Network Management: Theory and Practice", *Proceding of the 30th SIGCSE Technical Symposium on Computer Science Education,* New Orleans, USA, March 1999, vol 31, no. 1, pp.119-121

[2] R. F. Simarmata, R. Tulloh, Y. S. Haryani, "Simulasi jaringan Software Defined Network menggunakan protokol routing OSPF dan Ryu controller," *E-Proceeding of Applied Science*, vol. 4, no. 3, pp. 2887-2896, December 2018

[3] R. M. Negara dan R. Tulloh, "Analisis simulasi penerapan algoritma OSPF menggunakan routeflow pada jaringan software defined network", *Jurnal Infotel*, vol. 9, no. 1, pp. 75-83, Februari 2017, doi: 10.20895/infotel.v9i1.172

[4] K. Nugroho and D. P. Setyanugroho, "Analisis Kinerja RouteFlow pada Jaringan SDN menggunakan Topologi Full-Mesh," *ELKOMIKA*, vol. 7, no. 3, pp. 585-599, 2019, doi: 10.26760/elkomika.v7i3.585

[5] S. Valluvan, T. Monaranjitham, and V. Nagarajan, "A Study on SDN Controllers", *International Journal of Pharmacy and Technology*, vol. 8, no. 4, pp. 5234-5242, December 2016

[6] S. Wang, H. Chiu, and C. Chou, "Comparisons of SDN OpenFlow Controllers over EstiNet: Ryu vs NOX", *Internatioal Symposium of Advance Software Define Networks*, Barcelona, Spain, April 2015, pp. 1-6

[7] R. Afan, A. Virgono, and M. Rumani, "Analis Efek Penggunaan Kontroller RYU dan POX pada Performansi Jaringan SDN", *E-Proceeding of Engineering*, vol. 5, no. 3, 2018

[8] Y. G. Reinhart, "Implementasi Jaringan Software Defined Network Dengan Routing OSPF dan POX sebagai Controller", *Thesis*, Telkom University, Bandung, Indonesia, 2018

[9] A. Irmawati, I. D. Irwati, dan Y. S. Hariyani, "Implementasi Protokol Routing OSPF pada Software Defined Network berbasis RouteFlow", *E-Proceeding of Applied Science*, vol. 3, no. 2, pp. 1067-,1074 August 2017

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd Ed., Cambridge, MIT Press, 2001

[11] NN., "Software Defined Networking," *Cisco*, 30 Agustus 2013

[12] N. Hanan, "Mengenal Salah Satu Controller dalam SDN", *Core Network Technology*, 2018

[13] Y. S. Hariyani, I. Irawati, S. Dwi, and M. Nuruzzamanirridha, "Routing Implementation Based-On Software Defined Network Using Ryu Controller and Openvswitch", Jurnal Teknologi, vol. 78, no. 5, pp. 295-298, 2016, doi: 10.11113/jt.v78.8315

[14] S. Avallone, S. Guadagno, D. Emma, A. Pescape and G. Ventre, "D-ITG distributed Internet traffic generator," *Proceeding of 1st International Conference on the Quantitative Evaluation of Systems, 2004 (QEST 2004)*, Netherland, 2004, pp. 316-317, doi: 10.1109/QEST. 2004.1348045

[15] M. Bushell, K. Hoque, and D. Dean, "Researching the Network Trap," In: *The Network Trap. Work, Organization, and Employment*, Springer, Singapore, 2020, doi.org/10.1007/978-981-15-0878-3_3

[16] S. Azodolmolky, *Software Defined Network with OpenFlow*, Packt Publishing Ltd., USA, 2013

[17] T. R. Hapsari, "Berkenalan dengan OpenFlow," *Core Network Laboratory*, 2018

[18] S. Solnushkin, "Fat Tree Design", *ClusterDesign.org*, 2013

[19] S. U. Masruroh, A. Fiade, M. F. Iman and Amelia, "Performance evaluation of routing protocol RIPv2, OSPF, EIGRP with BGP," *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, 2017, pp. 1-7, doi: 10.1109/INNOCIT. 2017.8319134

[20] R. Rasudin, "Quality of Service (QoS) pada Jaringan Internet dengan Metode Hirerarchy Token Bucket", *Techsi*, vol. 6, no. 1, 2014, doi: 10.29103/techsi.v6i1.172