OPEN ACCESS
JOURNAL OF TELECOMMUNICATION, ELECTRONICS, AND CONTROL ENGINEERING (JTECE) ISSN: 2654-8275 (ONLINE)
JTECE. VOL. 4, NO. 2, PP. 66-79, JULY 2022 DOI: 10.20895/JTECE.V4I2.360

Perbandingan Kontroler POX, Ryu dan ONOS pada Arsitektur Software Defined Network (SDN) Menggunakan Topologi Linear

The Comparison of POX, Ryu, and ONOS Controler in The Architecture of Software Defined Network (SDN) by Linear Topology

Eka Wahyudi[1,*], Mega Safira Nuraeni[2], Nanda Iryani[3]

[1,2]D3 Teknik Telekomunikasi, [3]S1 Teknik Telekomunikasi,
Fakultas Teknik Telekomunikasi dan Elektro, Institut Teknologi Telkom Purwokerto
Jl. D.I. Panjaitan No. 128 Purwokerto, Jawa Tengah, Indonesia

[1,*]Penulis korespondensi: ekawahyudi@ittelkom-pwt.ac.id
[2]17101105@ittelkom-pwt.ac.id, [3]nanda@ittelkom-pwt.ac.id

Received on 31-08-2021, accepted on 02-06-2022, published on 04-07-2022

Abstrak

***Software Defined Network* merupakan konsep baru yang memisahkan *data plane* dengan *control plane*. Kontroler menjadi suatu hal yang penting dalam membangun arsitektur *Software Defined Network*, sehingga dibutuhkan informasi terkait performansi kontroler dapat diketahui tingkat kemampuan suatu kontroler, di antaranya kontrolet *POX, Ryu* dan *ONOS*. Pada penelitian ini digunakan topologi *linear* yang menggunakan konfigurasi 10 *switch* dan 10 *host*, 12 *switch* dan 12 *host*, 14 *switch* dan 14 *host* dan 16 *Switch* 16 *host* dengan *background traffic* dari 50 *Mbps* hingga 200 *Mbps*. Dari pengujian diperoleh hasil bahwa kenaikan nilai *throughput, delay* dan *jitter* berbanding lurus dengan meningkatnya jumlah *switch.* Kontroler *POX* memiliki performa terbaik dibandingkan dengan kontroler *Ryu* dan *ONOS* dengan nilai *throughput* yang didapatkan sebesar 5.139,456 *Kbits/sec* hingga 5.142,139 *Kbits/sec*, untuk nilai *delay* yang didapatkan sebesar 0,078 *ms* hingga 0,110 *ms* sedangkan untuk nilai *jitter* diperoleh 0,037 *ms* hingga 0,070 *ms*.**

Kata kunci: Software Defined Network, Topologi Linear, POX, Ryu, ONOS

Abstract

Software-Defined Network is a new concept that separates the data plane from the control plane. Therefore, the controller becomes essential in building a Software Defined Network architecture. Information related to controller performance is needed to know the level of capability of a controller, including POX, Ryu, and ONOS controllers. In this study, a linear topology has been used by the configuration of 10 switches and 10 hosts, 12 switches and 12 hosts, 14 switches and 14 hosts, and 16 switches with 16 hosts with background traffic from 50 Mbps to 200 Mbps. The test found that the increase in throughput, delay, and jitter values is directly proportional to the number of switches. The POX controller has the best performance compared to the Ryu and ONOS controllers, with the throughput value obtained from 5,139,456 Kbits/sec to 5,142.139 Kbits/sec, for the delay value obtained from 0.078 ms to 0.110 ms, while the jitter value it is obtained from 0.037 ms to 0.070 ms.

Keywords: Software Defined Network, Linear Topology, POX, Ryu, ONOS

I. PENDAHULUAN

Perkembangan jaringan yang sangat pesat berimbas pada pengelola jaringan dalam mengelola aliran data. Jaringan konvensional membutuhkan pekerjaan tambahan untuk mengelola aliran data yang semakin bertambah, sehingga memunculkan ide gagasan dalam membangun sebuah arsitektur jaringan baru dengan tujuan meminimalisir pekerjaan pengelola jaringan [1]. *Software Defined Network* (SDN) menjadi terobosan terbaru dalam permasalahan pengelola jaringan terhadap bertambahnya aliran data, *Software Defined Network* memisahkan *data plane* dengan *control plane* pada sebuah perangkat guna mempermudah dalam membangun, mendesain serta mengelola jaringan. *Software Defined Network* memisahkan antara *data plane* dan *control plane* pada sebuah perangkat sehingga pengelolaan sebuah jaringan terpusat pada *control plane* yang di dalamnya terdapat kontroler. Tujuan dari dibentukan arsitektur *SDN* adalah untuk mempermudah dalam memanajemen jaringan, mengurangi biaya pengoperasian jaringan [2]. Unjuk kerja kontroler dengan parameter *Quality of Service* menjadi salah satu cara dalam menentukan performansi sebuah kontroler [3].

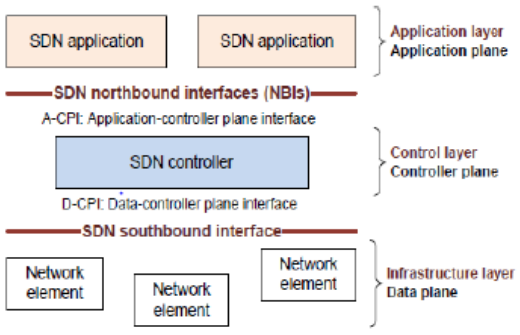
Jurnal penelitian (Mohamed Eltaj,2020) dengan judul "*Performance Evaluation of SDN Controller: FloodLight, POX and NOX"* dengan pengujian terhadap penambahan jumlah *switch* dan untuk hasilnya, kontroler *POX* dan *NOX* mendapatkan nilai *throughput* terbaik sedangkan untuk pengujian *latency*, kontroler *POX* memiliki *delay* lebih baik dibandingkan *NOX* dan *floodlight* [4]. Jurnal penelitian (Moh wahyudi Putra,2018) dengan judul "Analisis Perbandingan Performansi Kontroler *Floodlight, Maestro, Ryu, POX* dan *ONOS* dalam Arsitektur *Software Defined Network"* dengan pengujian parameter *throughput* dan *latency* terhadap jumlah *switch* dan *host* yang bervariasi menghasilkan kontroler *maestro* memiliki nilai *throughput* dan *latency* yang lebih baik dibandingkan kontroler lain yang diujikan [5]. Jurnal penelitian (Mahmood Z. Abdullah,2018) dengan judul "*Performance Evaluation and Comparison of Softwide Defined Networs Controller"* dengan pengujian terhadap jumlah *switch* yang berubah menghasilkan nilai *throughput* semakin menurun seiring beban kontroler semakin meningkat [6]. Pada jurnal penelitian (Sm Shamim) dengan judul "*Performance Analysis of Differernt Openflow based Controller Over Software Defined Network"* pada kontroler Ryu, POX dan Pyretic Openflow menghasilkan nilai kontroler *Pyretic Openflow* lebih baik dibandingkan kontroler *Ryu* dan *POX* [7].

Berdasarkan penelitian tersebut, penulis akan melakukan penelitian performansi pada kontroler *POX*, *Ryu* dan *ONOS* dengan menggunakan *background traffic* dan vasiasi jumlah *switch.* Parameter *Quality of Servicev(QoS)* digunakan untuk mengetahui kualitas suatu kontroler berdasarkan standar dari *TIPHON*. Parameter yang akan diujikan yaitu *Throughput, Delay* dan *Jitter*.

II. KAJIAN PUSTAKA

A. Software Defined Network

Software Defined Network merupakan pemikiran baru dengan karakteristik dinamis, *manageable, consteffective* serta *adaptable*, arsitektur jaringan ini menjadi salah satu arsitektur yang bersifat dinamis yang dapat dikontrol di manapun tanpa harus mendatangi perangkat untuk melakukan *setting* pada perangkat [8].



Gambar 1. Arsitektur *Software Defined Network* [5].

Pada gambar 1 ditunjukkan arsitektur dari *Software Defined Network* yang terdiri dari 3 layer. Layer pertama merupakan infrastruktur atau dapat disebut sebagai *data plane* yang berisi perangkat jaringan. Pada layer kedua terdapat kontrol layer atau *control plane*, pada layer ini kontroler berada dan pada layer ketiga merupakan *application layer* yang berisi aplikasi yang dibutuhkan oleh infrastruktur layer.

B. Openflow

Openflow merupakan sebuah *protocol* yang menghubungkan *control plane* dan *data plane*, sehingga *protocol openflow* memiliki peran utama dalam sebuah konsep arsitektur jaringan *Software Defined Network*. Tugas dari *openflow* adalah memberi akses serta mengatur *forwarding plane* yang terdapat pada *switch* dan *router*. Sehingga apabila hendak membangun sebuah jaringan *Software Defined Network*, maka harus menggunakan perangkat yang telah menerapkan *protocol openflow* [9].

C. Controller

Controller merupakan elemen penting dalam arsitektur *Software Defined Network*. *Controller* mewujudkan bidang kontrol terdistribusi serta dapat membantu dalam pengawasan pada sebuah arsitektur jaringan. Terdapat banyak jenis kontroler yang telah berkembang, beberapa kontroler tersedia dalam *open source* seperti *FloodLight, Beacon, Opendaylight, ONOS, Ryu* [10].

D. POX

POX merupakan salah satu kontroler yang menggunakan bahasa pemrograman *Python*, dalam penggunaannya *POX* dapat diterapkan pada *Windows, Mac OS, Linux* [11].

E. RYU

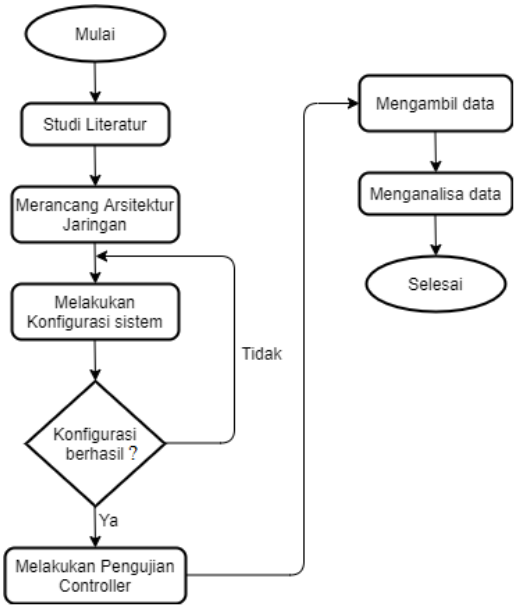
Ryu merupakan jenis dari *controller SDN*, secara umum fungsi kontroler *Ryu* sama dengan fungsi jenis kontroler lainnya. *Ryu* menggunakan bahas pemrograman *Python*, *Ryu* memiliki banyak dokumentasi dari para peneliti. *Ryu* mendukung beberapa *protocol* pada *SDN* yang telah disediakan seperti *openflow, netconf, of-config* [12].

F. ONOS

ONOS termasuk kedalam kontroler yang bersifat *open source* yang berorientasi pada jaringan operator. *ONOS* menggunakan bahasa pemrograman *java* kemudian setiap fitur yang berada pada kontroler *ONOS* telah diaktifkan menggunakan *Apache*. Kontroler *ONOS* juga menjadi salah satu kontroler yang sering digunakan [13].

III. METODE PENELITIAN

Pada penelitian ini, melakukan unjuk kerja pada kontroler *POX, Ryu* dan *ONOS* pada emulator mininet. Dengan menganalisa performansi setiap kontroler yang diuji terhadap parameter *Quality of Service* yang terdiri dari *throughput, delay* dan *jitter*.

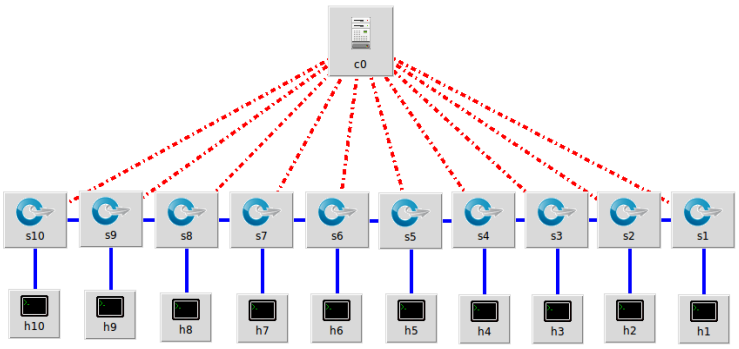


Gambar 2. Metode Penelitian

Pada gambar 2 ditunjukkan metode yang digunakan dalam melakukan penelitian ini, terdiri dari studi literatur, perancangan arsitektur jaringan, konfigurasi pada sistem, pengujian kontroler, pengambian data dan analisa data. Pada studi literatur, peneliti melakukan pengumpulan referensi, materi penelitian sebagai acuan dan tolak ukur dalam melakukan penelitian, pada tahap perancangan arsitektur jaringan peneliti melakukan perancangan topologi dengan emulator *mininet*, pada tahap konfigurasi sistem peneliti menghubungkan *mininet* dengan kontroler, pada tahap pengujian kontroler peneliti melakukan pengujian kontroler sesuai skenario pada tabel 1 pengujian dengan 10 *switch*, 12 *Switch*, 14 *Switch* dan 16 *Switch* serta menggunakan besar data yang konstant sebesar 12,8 *MByte* dan *background traffic* sebesar 50 *Mbps*, 100 *Mbps*, 150 *Mbps* dan 200 *Mbps*, pada tahap pengambilan data dan analisa data peneliti mengumpulkan hasil dari seluruh pengujian yang akan dianalisa performansi pada setiap parameter yang mengacu pada standarisasi TIPHON.

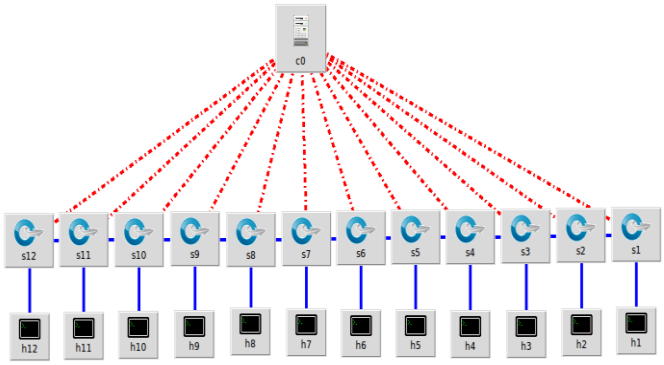
Tabel 1. Skenario Pengujian

Topologi Linear	**Awal**		**Tujuan**	
	Host	**IP Address**	**Host**	**IP Address**
10 Switch dan Host	Host - 10	10.0.0.10	Host - 1	10.0.0.1
12 Switch dan Host	Host - 12	10.0.0.12	Host - 1	10.0.0.1
14 Switch dan Host	Host - 14	10.0.0.14	Host - 1	10.0.0.1
16 Switch dan Host	Host - 16	10.0.0.15	Host - 1	10.0.0.1



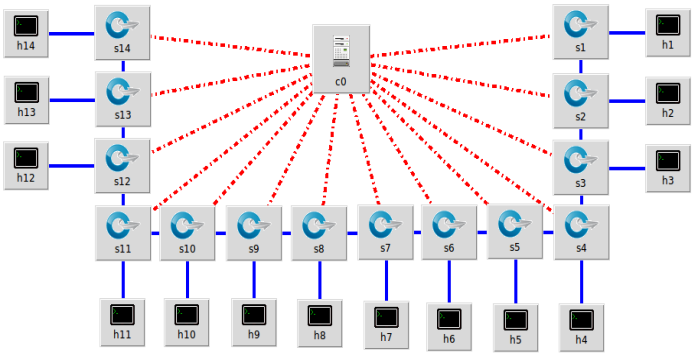
Gambar 3. Topologi *Linear* 10 *switch* dan 10 *Host*

Pada gambar 3 ditunjukkan arsitektur jaringan menggunakan topologi *linear* 10 *Switch* dan 10 *Host*, h1 dan h10 digunakan sebagai komunikasi data yang akan diamati menggunakan *tools D-ITG*, sedangkan h2 dan h9 digunakan sebagai pembangkit *background traffic* dengan menggunakan *tools iperf*.



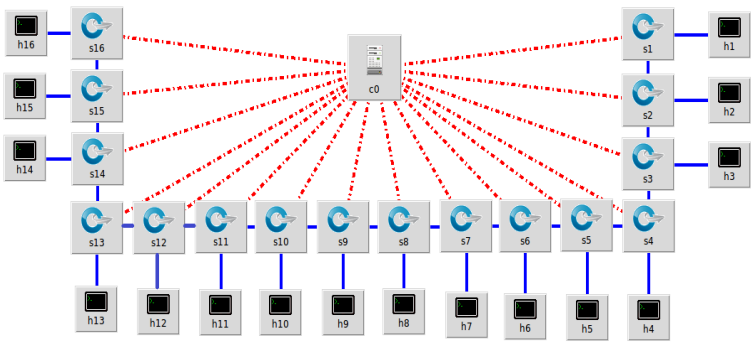
Gambar 4. Topologi *Linear* 12 *Switch* dan 12 *host*

Pada gambar 4 ditunjukkan arsitektur jaringan menggunakan topologi *linear* 12 *Switch* dan 12 *Host*, h1 dan h12 digunakan sebagai komunikasi data yang akan diamati menggunakan *tools D-ITG*, sedangkan h2 dan h11 digunakan sebagai pembangkit *background traffic* dengan menggunakan *tools iperf*.



Gambar 5. Topologi *Linear* dengan 14 *Switch* dan 14 *Host*

Pada gambar 5 ditunjukkan arsitektur jaringan menggunakan topologi *linear* 14 *Switch* dan 14 *Host*, h1 dan h14 digunakan sebagai komunikasi data yang akan diamati menggunakan *tools D-ITG*, sedangkan h2 dan h13 digunakan sebagai pembangkit *background traffic* dengan menggunakan *tools iperf*.



Gambar 6. Topologi *Linear* dengan 16 *Switch* dan 16 *Host*

Pada gambar 6 ditunjukkan arsitektur jaringan menggunakan topologi *linear* 16 *Switch* dan 16 *Host*, h1 dan h16 digunakan sebagai komunikasi data yang akan diamati menggunakan *tools D-ITG*, sedangkan h2 dan h15 digunakan sebagai pembangkit *background traffic* dengan menggunakan *tools iperf*.

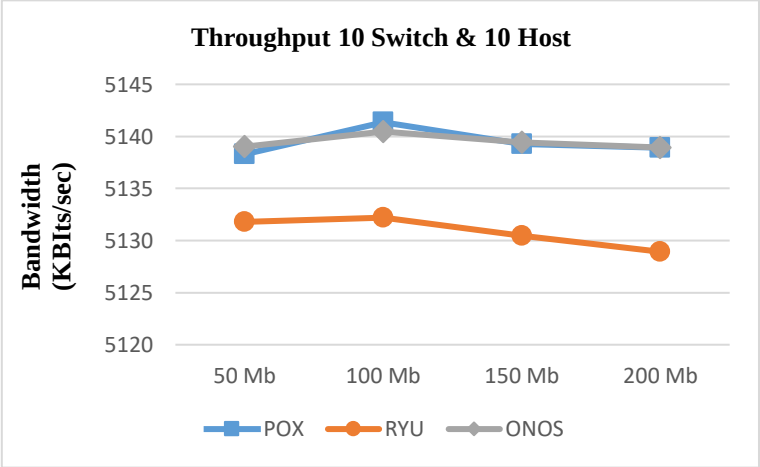
IV. HASIL DAN PEMBAHASAN

Pengujian ini dilakukan untuk pengujian parameter *throughput*, *delay* dan *jitter* pada kontroler *POX*, *Ryu* dan *ONOS* dengan jumlah *switch* dan *host* yang bervariasi dan dengan *background traffic* yang bervariasi.

A. Pengujian nilai *throughput*

Tabel 2. *Throughput* 10 *Switch* dan 10 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (Kbits/sec)	RYU (Kbits/sec)	ONOS (Kbits/sec)
10 Switch & 10 Host	50	5.138,285	5.131,776	5.139,034
	100	5.141,357	5.132,200	5.140,465
	150	5.139,268	5.130,486	5.139,439
	200	5.138,913	5.128,925	5.138,935

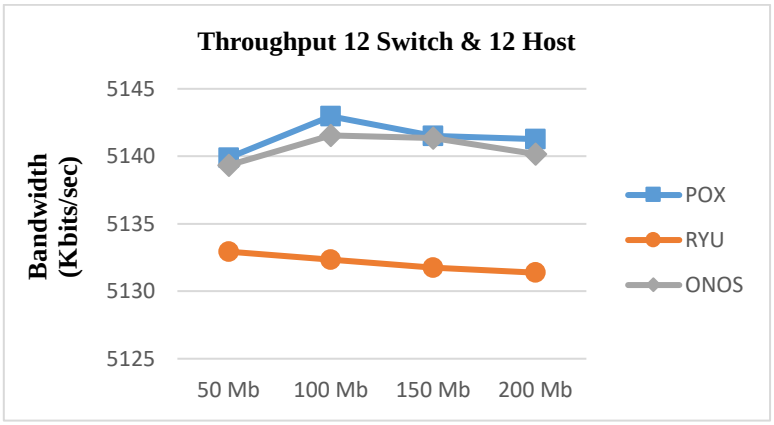


Gambar 7. Grafik *Throughput* 10 *Switch* dan 10 *Host*

Pada tabel 2 dan gambar 7 ditunjukkan hasil pengujian menggunakan 10 *Switch* dan 10 *Host* menunjukan peningkatan pada kontroler *POX*, *Ryu* dan *ONOS* pada pengujian *background traffic* 100 *Mbps* kemudian pada pengujian menggunakan 150 *Mbps* nilainya menurun dan mengalami penurunan kembali pada pengujian 200 *Mbps.* Pengujian menggunakan 10 *Switch* dan 10 *Host* menghasilkan nilai rata rata keseluruhan pada kontroler *POX* sebesar 5.139,456 *Kbits/sec*, kontroler *Ryu* menghasilkan rata rata keseluruhan 5.130,847 *Kbits/sec* dan kontroler *ONOS* menghasilkan nilai rata rata keseluruhan sebesar 5139,468 *Kbits/sec*.

Tabel 3. Throughput 12 Switch dan 12 Host

Nswitch & NHost	Besar Trafik (Mbps)	POX (Kbits/sec)	RYU (Kbits/sec)	ONOS (Kbits/sec)
12 Switch & 12 Host	50	5.139,891	5.132,935	5.139,333
	100	5.142,960	5.132,352	5.141,546
	150	5.141,521	5.131,731	5.141,341
	200	5.141,291	5.131,377	5.140,164

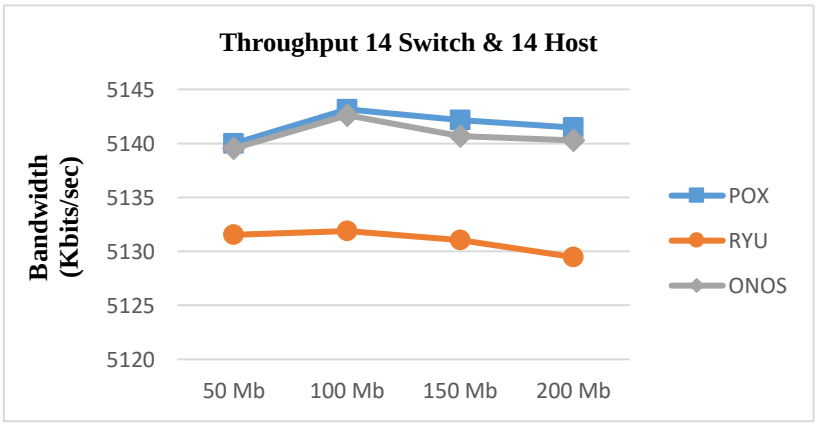


Gambar 8. Grafik *Throughput* 12 *Switch* dan 12 *Host*

Pada tabel 3 dan gambar 8 ditunjukkan hasil pengujian menggunakan 12 *Switch* dan 12 *Host* menunjukan peningkatan pada kontroler *POX* dan *ONOS* pada pengujian *background traffic* 100 *Mbps* kemudian pada pengujian menggunakan 150 *Mbps* nilainya menurun dan mengalami penurunan kembali pada pengujian 200 *Mbps* sedangkan pada kontroler *Ryu throughput* menurun pada pengujian 100 *Mbps* dan 150 *Mbps* kemudian pada percobaan 200 *Mbps* nilainya meningkat. Pengujian menggunakan 12 *Switch* dan 12 *Host* menghasilkan nilai rata rata keseluruhan pada kontroler *POX* sebesar 5.141,416 *Kbits/sec*, kontroler *Ryu* menghasilkan rata rata keseluruhan 5.132,099 *Kbits/sec* dan kontroler *ONOS* menghasilkan nilai rata rata keseluruhan sebesar 5.140,596 *Kbits/sec*.

Tabel 4. *Throughput* 14 *Switch* dan 14 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (Kbits/sec)	RYU (Kbits/sec)	ONOS (Kbits/sec)
14 Switch & 14 Host	50	5.139,987	5.131,529	5.139,541
	100	5.143,154	5.131,881	5.142,606
	150	5.142,179	5.131,050	5.140,690
	200	5.141,441	5.129,470	5.140,278

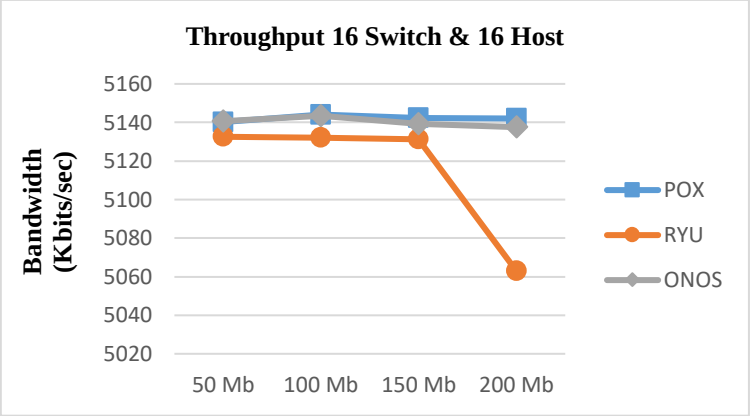


Gambar 9. Grafik *Throughput* dengan 14 *Switch* dan 14 *Host*

Pada tabel 4 dan gambar 9 ditunjukkan hasil pengujian menggunakan 14 *Switch* dan 14 *Host* menunjukan peningkatan pada kontroler *POX*, *Ryu* dan *ONOS* pada pengujian *background traffic* 100 *Mbps* kemudian pada pengujian menggunakan 150 *Mbps* nilainya menurun dan mengalami penurunan kembali pada pengujian 200 *Mbps*. Pengujian menggunakan 14 *Switch* dan 14 *Host* menghasilkan nilai rata rata keseluruhan pada kontroler *POX* sebesar 5.141,490 *Kbits/sec*, kontroler *Ryu* menghasilkan rata rata keseluruhan 5.130,983 *Kbits/sec* dan kontroler *ONOS* menghasilkan nilai rata rata keseluruhan sebesar 5.140,4779 *Kbits/sec*.

Tabel 5. Throughput 16 Switch dan 16 Host

Nswitch & NHost	Besar Trafik (Mbps)	POX (Kbits/sec)	RYU (Kbits/sec)	ONOS (Kbits/sec)
16 Switch & 16 Host	50	5.140,211	5.132,559	5.140,664
	100	5.144,082	5.132,105	5.143,444
	150	5.142,344	5.131,177	5.139,240
	200	5.141,919	5.062,968	5.137,562



Gambar 10. Grafik *Throughput* 16 *Switch* dan 16 *Host*

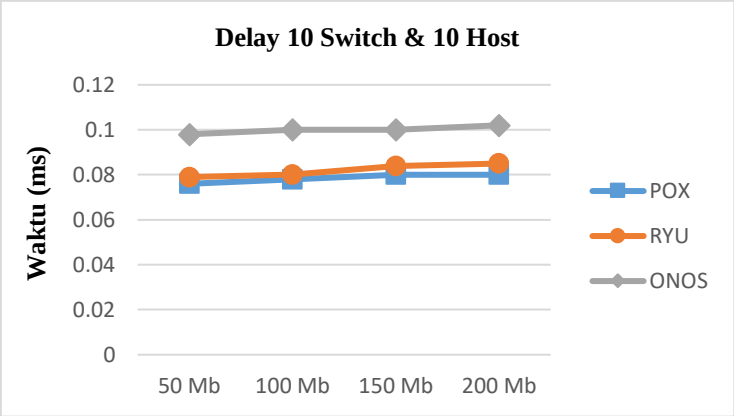
Pada tabel 5 dan gambar 10 ditunjukkan hasil pengujian menggunakan 16 *Switch* dan 16 *Host* menunjukan peningkatan pada kontroler *POX* dan *ONOS* pada pengujian *background traffic* 100 *Mbps* kemudian pada pengujian menggunakan 150 *Mbps* nilainya menurun dan mengalami penurunan kembali pada pengujian 200 *Mbps* sedangkan pada kontroler *Ryu throughput* menurun pada pengujian 100 *Mbps* dan 150 *Mbps* kemudian pada percobaan 200 *Mbps* nilainya kembali menurun. Pengujian menggunakan 16 *Switch* dan 16 *Host* menghasilkan nilai rata rata keseluruhan pada kontroler *POX* sebesar 5.142,139 *Kbits/sec*, kontroler *Ryu* menghasilkan rata rata keseluruhan 5.114,702 *Kbits/sec* dan kontroler *ONOS* menghasilkan nilai rata rata keseluruhan sebesar 5.140,228 *Kbits/sec*.

Nilai *throughput* yang didapatkan meningkat berbanding lurus dengan bertambahnya jumlah *switch* dan *host*. Kenaikan nilai *throughput* disebabkan karena *background traffic* yang diujikan semakin besar sedangkan penurunan nilai *throughput* disebabkan karena pemberian *background traffic* semakin besar dan dalam waktu yang relatif singkat sehingga mengakibatkan penurunan pada performa kontroler tersebut.

B. Pengujian Nilai *Delay*

Tabel 6. *Delay* 10 *Switch* dan 10 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
10 Switch & 10 Host	50	0,076	0,079	0,098
	100	0,078	0,080	0,100
	150	0,080	0,084	0,100
	200	0,080	0,085	0,102

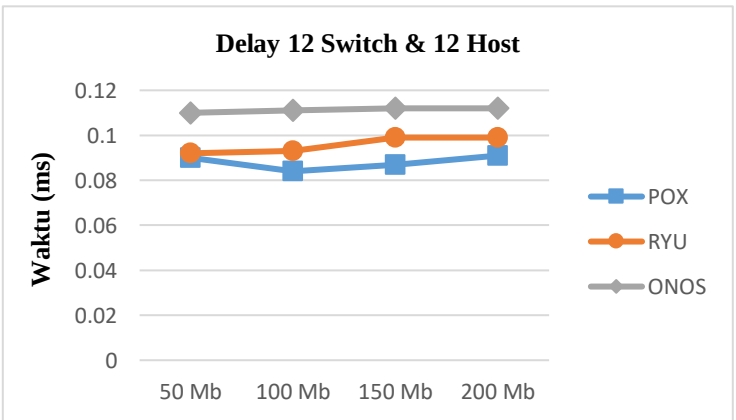


Gambar 11. Grafik *Delay* pada 10 *Switch* dan 10 *Host*

Pada tabel 6 dan gambar 11 ditunjukkan hasil pengujian menggunakan 10 *switch* dan 10 *host*, kontroler *POX* mengalami peningkatan pada pengujian *background traffic* 100 *Mbps* dan nilainya meningkat pada 150 *Mbps* dan stabil pada pengujian 200 *Mbps*, kontroler *Ryu* mengalami peningkatan nilai *delay* setiap *background traffic* yang diujikan semakin besar sedangkan kontroler *ONOS* mengalami peningkatan pada pengujian *background traffic* 100 *Mbps* dan stabil pada pengujian 150 *Mbps* kemudian kembali meningkat pada pengujian 200 *Mbps*. Dari penelitian menggunakan 10 *Switch* dan 10 *host* menghasilkan nilai delay rata rata keseluruhan pada kontroler *POX* sebesar 0,078 *ms*, pada kontroler *Ryu* sebesar 0,082 *ms* dan pada kontroler *ONOS* sebesar 0,100 *ms*.

Tabel 7. *Delay* 12 *Switch* dan 12 *Host*

NSwitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
12 Switch & 12 Host	50	0,090	0,092	0,110
	100	0,084	0,093	0,111
	150	0,087	0,099	0,112
	200	0,091	0,099	0,112

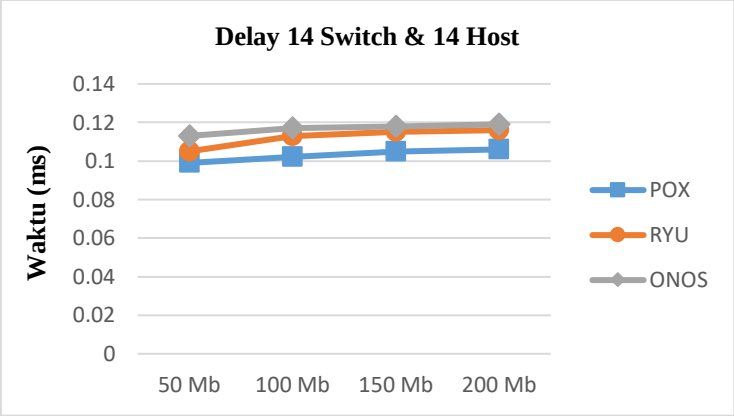


Gambar 12. Grafik *Delay* pada 12 *Switch* dan 12 *Host*

Pada tabel 7 dan gambar 12 ditunjukkan hasil pengujian menggunakan 12 *switch* dan 12 *host*, kontroler *POX* mengalami penurunan pada pengujian *background traffic* 100 *Mbps* dan nilainya meningkat pada 150 *Mbps* dan meningkat kembali pada pengujian 200 *Mbps*, kontroler *Ryu* mengalami peningkatan nilai *delay* pada *background traffic* 100 *Mbps* dan 150 *Mbps* dan stabil pada pengujian 200 *Mbps* sedangkan kontroler *ONOS* mengalami peningkatan pada pengujian *background traffic* 100 *Mbps* dan pengujian 150 *Mbps* kemudian nilainya stabil pada pengujian 200 *Mbps*. Dari penelitian menggunakan 12 *Switch* dan 12 *host* menghasilkan nilai *delay* rata rata keseluruhan pada kontroler *POX* sebesar 0,088 *ms*, pada kontroler *Ryu* sebesar 0,095 *ms* dan pada kontroler *ONOS* sebesar 0,111 *ms*.

Tabel 8. *Delay* 14 *Switch* dan 14 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
14 Switch & 14 Host	50	0,099	0,105	0,113
	100	0,102	0,113	0,117
	150	0,105	0,115	0,118
	200	0,106	0,116	0,119

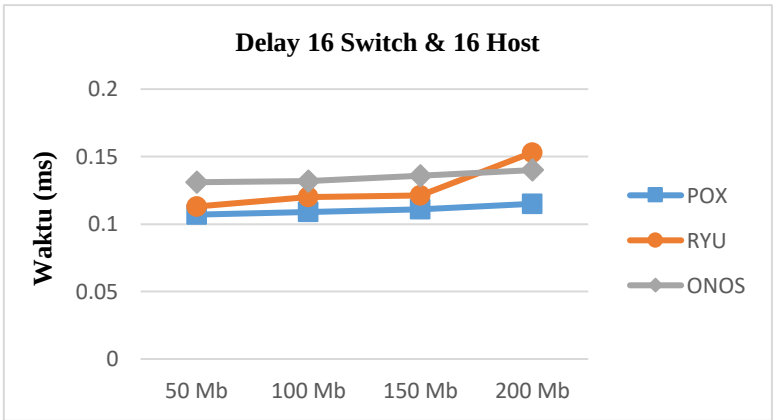


Gambar 13. Grafik *Delay* pada 14 *Switch* dan 14 *Host*

Pada pengujian menggunakan 14 *Switch* dan 14 *Host* pada tabel 9 dan gambar 13, kontroler *POX*, *Ryu* dan *ONOS* mengalami peningkatan seiring bertambahnya nilai *background traffic* yang diujikan. Pada kontroler *POX* mendapatkan nilai *delay* dengan rata rata keseluruhan sebesar 0,103 *ms*, pada kontroler *Ryu* rata rata delay keseluruhan sebesar 0,112 *ms* sedangkan pada kontroler *ONOS* rata rata nilai *delay* yang diperoleh adalah sebesar 0,116 *ms.*

Tabel 9. *Delay* pada 16 *Switch* dan 16 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
16 Switch & 16 Host	50	0,107	0,113	0,131
	100	0,109	0,120	0,132
	150	0,111	0,121	0,136
	200	0,115	0,153	0,140



Gambar 14. Grafik *Delay* pada 16 *Switch* dan 16 *Host*

Pada tabel 9 dan gambar 14 ditunjukkan hasil pengujian menggunakan 16 *Switch* dan 16 *Host*, kontroler *POX*, *Ryu* dan *ONOS* mengalami peningkatan seiring bertambahnya nilai *background traffic* yang diujikan. Pada kontroler *POX* mendapatkan nilai *delay* dengan rata rata keseluruhan sebesar 0,110 *ms*, pada kontroler

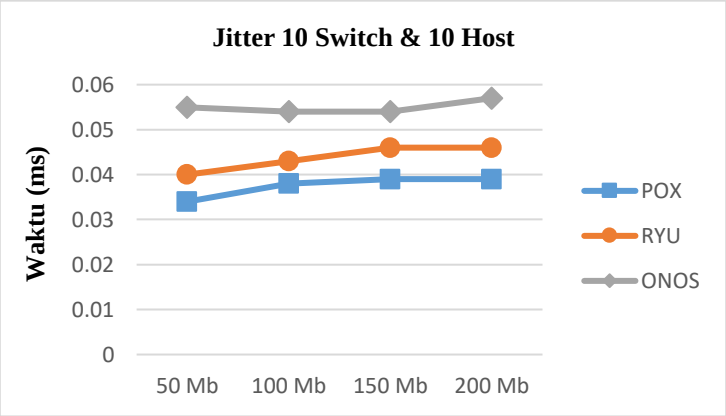
Ryu rata rata *delay* keseluruhan sebesar 0,126 *ms* sedangkan pada kontroler *ONOS* rata rata nilai *delay* yang diperoleh adalah sebesar 0,134 *ms.*

Semakin kecil nilai *delay* maka semakin baik pula performa dari sebuah kontroler, pada penelitian ini semakin benyak jumlah *switch* dan *host* yang digunakan maka semakin bertambah nilai *delay* yang dihasilkan hal ini dikarenakan pada infrastruktur layer, *link* yang terhubung dengan kontroler bertambah banyak sehingga mempengaruhi data yang dikirimkan melewati beberapa *link* dalam kondisi *background traffic* yang dibangkitkan

C. Pengujian Nilai *Jitter*

Tabel 10. *Jitter* Pada 10 *Switch* dan 10 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
10 Switch & 10 Host	50	0,034	0,040	0,055
	100	0,038	0,043	0,054
	150	0,039	0,046	0,054
	200	0,039	0,046	0,057

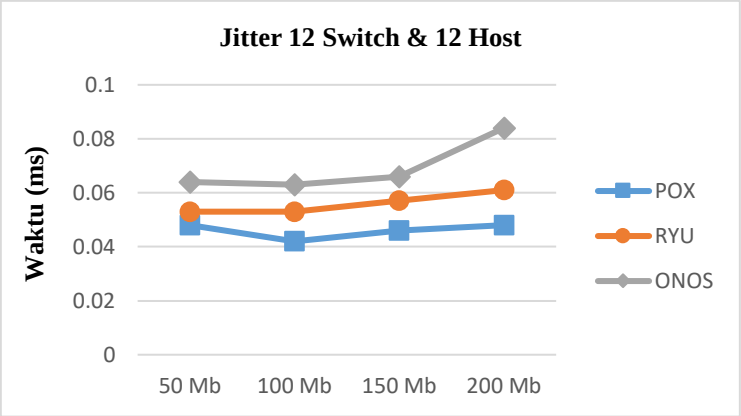


Gambar 15. Grafik *Jitter* Pada 10 *Switch* dan 10 *Host*

Pada tabel 10 dan gambar 15 ditunjukkan hasil pengujian menggunakan 10 *Switch* dan 10 *Host* menunjukan peningkatan pada kontroler *POX* dan *Ryu* setiap *background traffic* yang diujikan semakin besar pada pengujian *background traffic* 150 *Mbps* dan 200 *Mbps* nilainya stabil sedangkan pada kontroler *ONOS jitter* menurun pada pengujian 100 *Mbps* dan 150 *Mbps* kemudian pada percobaan 200 *Mbps* nilainya kembali meningkat. Pengujian menggunakan 10 *Switch* dan 10 *Host* menghasilkan nilai rata rata keseluruhan pada kontroler *POX* sebesar 0,037 *ms*, kontroler *Ryu* menghasilkan rata rata keseluruhan 0,043 *ms* dan kontroler *ONOS* menghasilkan nilai rata rata keseluruhan sebesar 0,055 *ms.*

Tabel 11. *Jitter* pada 12 *Switch* dan 12 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
12 Switch & 12 Host	50	0,048	0,053	0,064
	100	0,042	0,053	0,063
	150	0,046	0,057	0,066
	200	0,048	0,061	0,084

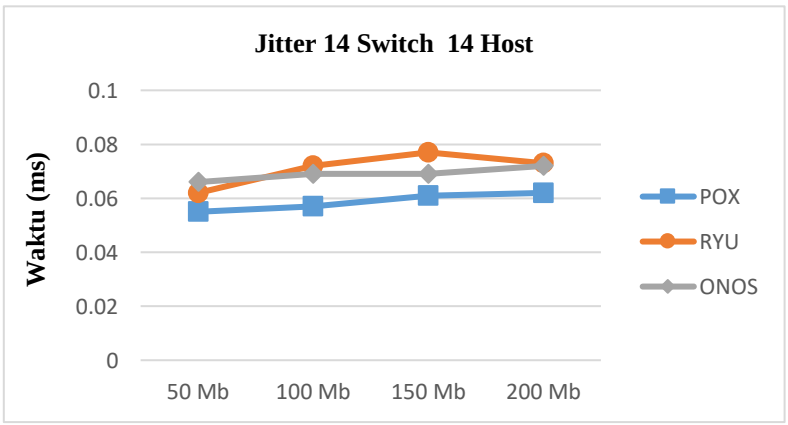


Gambar 16. Grafik *Jitter* Pada 12 *Switch* dan 12 *Host*

Pada tabel 11 dan gambar 16 ditunjukkan hasil pengujian menggunakan 12 *switch* dan 12 *host*, kontroler *POX* mengalami penurunan pada pengujian *background traffic* 100 *Mbps* dan nilainya meningkat pada 150 *Mbps* dan meningkat kembali pada pengujian 200 *Mbps*, kontroler *Ryu* nilai *jitter* stabil pada pengujian 100 *Mbps* dan mengalami peningkatan nilai *jitter* pada *background traffic* 150 *Mbps* dan 200 *Mbps* sedangkan kontroler *ONOS* mengalami penurunan pada pengujian *background traffic* 100 *Mbps* dan pengujian 150 *Mbps* nilainya kembali meningkat kemudian kembali meningkat pada pengujian 200 *Mbps*. Dari penelitian menggunakan 12 *Switch* dan 12 *host* menghasilkan nilai *jitter* rata rata keseluruhan pada kontroler *POX* sebesar 0,046 *ms*, pada kontroler *Ryu* sebesar 0,056 *ms* dan pada kontroler *ONOS* sebesar 0,069 *ms*.

Tabel 12. *Jitter* pada 14 *Switch* dan 14 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
14 Switch & 14 Host	50	0,055	0,062	0,066
	100	0,057	0,072	0,069
	150	0,061	0,077	0,069
	200	0,062	0,073	0,072

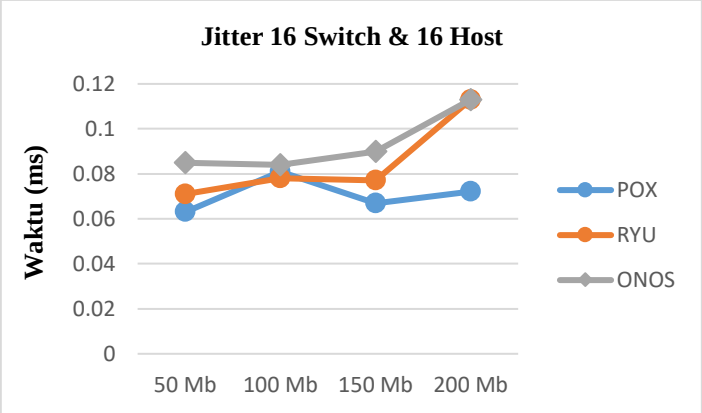


Gambar 17. Grafik *Jitter* pada 14 *Switch* dan 14 *Host*

Pada tabel 12 dan gambar 17 ditunjukkan hasil pengujian menggunakan 14 *switch* dan 14 *host*, kontroler *POX* mengalami peningkatan seiring bertambah besar *background traffic* yang diujikan, kontroler *Ryu* mengalami peningkatan nilai *jitter* pada *background traffic* 100 *Mbps* dan 150 *Mbps* kemudian menurun kembali pada pengujian *background traffic* 200 *Mbps* sedangkan kontroler *ONOS* mengalami peningkatan pada pengujian *background traffic* 100 *Mbps* dan pengujian 150 *Mbps* nilainya stabil kemudian kembali meningkat pada pengujian 200 *Mbps*. Dari penelitian menggunakan 14 *Switch* dan 14 *host* menghasilkan nilai *jitter* rata rata keseluruhan pada kontroler *POX* sebesar 0,058 *ms*, pada kontroler *Ryu* sebesar 0,071 *ms* dan pada kontroler *ONOS* sebesar 0,069 *ms*.

Tabel 13. *Jitter* Pada 16 *Switch* dan 16 *Host*

Nswitch & NHost	Besar Trafik (Mbps)	POX (ms)	RYU (ms)	ONOS (ms)
16 Switch & 16 Host	50	0,063	0,071	0,085
	100	0,081	0,078	0,084
	150	0,067	0,077	0,09
	200	0,072	0,113	0,113



Gambar 18. Grafik *Jitter* Pada 16 *Switch* dan 16 *Host*

Pada tabel 13 dan gambar 18 ditunjukkan hasil pengujian menggunakan 16 *switch* dan 16 *host*, kontroler *POX* mengalami peningkatan pada pengujian *background traffic* 100 *Mbps* dan nilainya menurun pada 150 *Mbps* dan meningkat kembali pada pengujian 200 *Mbps*, kontroler *Ryu* nilai *jitter* meningkat pada pengujian 100 *Mbps* dan mengalami penurunan nilai *jitter* pada *background traffic* 150 *Mbps* dan meningkat kembali pada pengujian 200 *Mbps* sedangkan kontroler *ONOS* mengalami penurunan pada pengujian *background traffic* 100 *Mbps* dan pengujian 150 *Mbps* nilainya kembali meningkat kemudian kembali meningkat pada pengujian 200 *Mbps*. Dari penelitian menggunakan 16 *Switch* dan 16 *host* menghasilkan nilai *jitter* rata rata keseluruhan pada kontroler *POX* sebesar 0,070 *ms*, pada kontroler *Ryu* sebesar 0,084 *ms* dan pada kontroler *ONOS* sebesar 0,093 *ms*.

Semakin kecil nilai *jitter* maka semakin baik pula performa dari sebuah kontroler, pada penelitian ini semakin banyak jumlah *switch* dan *host* yang digunakan maka semakin bertambah nilai *jitter* yang dihasilkan hal ini dikarenakan pada infrastruktur layer, *link* yang terhubung dengan kontroler bertambah banyak sehingga mempengaruhi data yang dikirimkan melewati beberapa *link* dalam kondisi *background traffic* yang dibangkitkan

V. KESIMPULAN

Pada pengujian menggunakan topologi *linear* dengan variasi *switch* dan *host* serta *background traffic*, kenaikan nilai *throughput* disebabkan karena meningkatnya *background traffic* yang diujikan serta jumlah *switch* dan *host* yang bertambah sedangkan penurunan nilai *throughput* disebabkan karena menurunya perfoma sebuah kontroler ketika *background traffic* yang diujikan semakin besar. Untuk nilai *delay* dan *jitter* mengalami peningkatan ketika jumlah *switch* dan *host* yang terhubung pada kontroler bertambah hal ini menyebabkan data yang dikirimkan harus melewati beberapa *link* untuk sampai kepada penerima data. Kontroler *POX* mendapatkan rata rata nilai *throughput* sebesar 5.139,456 *Kbits/sec* hingga 5.142,139 *Kbits/sec*, nilai *throughput* pada kontroler *Ryu* mendapatkan rata rata sebesar 5.130,847 *Kbits/sec* hingga 5.114,702 *Kbits/sec* sedangkan pada kontroler *ONOS* menghasilkan nilai rata rata keseluruhan 5.139,468 *Kbits/sec* hingga 5.140,228 *Kbits/sec*. Pada nilai *delay* yang dihasilkan kontroler *POX* mendapatkan rata rata keseluruhan sebesar 0,078 *ms* hingga 0,110 *ms* sedangkan untuk kontroler *Ryu* mengasilakan rata rata keseluruhan nilai *delay* sebesar 0,082 *ms* hingga 0,126 *ms* sedangkan untuk kontroler *ONOS* menghasilkan nilai *delay* sebesar 0,100 *ms* hingga 0,134 *ms*. Dilihat dari nilai keseluruhan yang diperoleh setiap kontroler, kontroler *POX* memiliki performa yang lebih baik dibandingkan dengan kontroler *Ryu* dan *ONOS* dari segi penerimaan data, dari nilai *delay* dan *jitter*.

KONTRIBUSI PENULIS

MSN Sebagai perancang simulasi dan menjalankan simulasi. NI melakukan review dan analisis hasil simulasi. EW menyusun skenario penelitian dan melakukan pengecekan simulasi agar sesuai skenario.

DAFTAR PUSTAKA

[1] I. M. A. W. I Putu Agus Eka Pratama, "Implementasi dan Analisis Simulasi QOS dan Perfomance Device dengan," *Jurnal Informatika Universitas Pamulang*, vol. IV, pp. 57-64, 2019.

[2] I. M. S. Mokhamad Aguk Nur Anggraini, "Performa Clustering Controllerpada Arsitektur Software Defined Network," *Journal of Informatics and Computer Science*, vol. II, pp. 1-8, 2020.

[3] V. P. Rikie Kartadie, "FLOODLIGHT VS ONOS DALAM UNJUK KERJA," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. IV, pp. 111-121, 2019.

[4] A. H. M. H. Mohamed Eltaj, "Performance Evaluation of SDN Controllers: FloodLight, POX and NOX," *International Journal of Engineering and Applied Sciences (IJEAS)*, vol. VII, no. 8, pp. 28-43, 2020.

[5] E. S. P. W. Y. Moh Wahyudi Putra, "Analisis Perbandingan Performansi Kontroler Floodlight, Maestro, RYU, POX dan ONOS dalam Arsitektur Software Defined Network (SDN)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. II, pp. 3779-3787, 2018.

[6] N. A. A.-a. W. H. Mahmood Z. Abdullah, "Performance Evaluation and Comparison of Software," *International Journal of Scientific Engineering and Science*, vol. II, no. 11, pp. 45-50, 2018.

[7] S. S. H. H. H. Sm Shamim, "Performance Analysis of Different Openflow based Controller Over Software Defined Networking," *Global Journal of Computer Science and Technology: CSoftware & Data Engineering*, vol. XVIII, no. 1, pp. 11-16, 2018.

[8] D. T. W. P. S. S. M. R. L. S. M. Faizal Mulya, "ANALISIS LOAD BALANCING PADA JARINGAN SOFTWARE DEFINED NETWORK (SDN) MENGGUNAKAN ALGORITMA OPTIMASI KOLONI SEMUT," *e-Proceeding of Engineering*, vol. VI, p. 1385, 2019.

[9] W. Y. P. K. Romy Dwi Andika Manullang, "Implementasi K-Shortest Path Routing pada Jaringan Software Defined Network," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. II, pp. 2462-2468, 2018.

[10] M. F. M. Ryhan Uddin, "Performance Analysis of SDN Based Firewalls: POX vs ODL," *International Conference on Advances in Electrical Engineering (ICAEE)*, 2019.

[11] U. D. D. Idris Z. Bholebawa, "Performance Analysis of SDN/OpenFlow Controllers:POX Versus Floodlight," *Wireless Pers Commun*, 2017.

[12] N. Hanan, "Mengenal Salah Satu Controller dalam SDN," 31 Oktober 2018. [Online]. Available: https://medium.com/core-network-laboratory-tech-page/mengenal-salah-satu-controller-dalam-sdn-706ed4624660. [Accessed 31 Agustus 2021].

[13] E. Mulyana, "ONOS," 22 Desember 2017. [Online]. Available: https://www.telematika.org/post/onos/. [Accessed 31 Agustus 2021].