# denar

*by* Denar 4

**Submission date:** 20-Jul-2023 02:29PM (UTC+0700)
**Submission ID:** 2133957118
**File name:** ANALISIS KOMPARASI PENERAPAN VIRTUAL LOCAL AREA NETWORK PADA SDN.pdf (352.61K)
**Word count:** 3838
**Character count:** 22243

Seminar Nasional Teknologi dan Rekayasa (SENTRA) 2020
ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

# ANALISIS KOMPARASI PENERAPAN *VIRTUAL LOCAL AREA NETWORK* PADA *SOFTWARE DEFINED NETWORK*

**Irhaz Muhajir[1], Denar Regata Akbi [2], Fauzi Dwi Setiawan Sumadi[3]**
[1,2,3]Universitas Muhammadiyah Malang, Malang

Kontak Person:
Irhaz Muhajir
Jalan Raya Tlogomas No. 246 Tlogomas, Kota Malang, Jawa Timur 65144
E-mail: irhazmuhajir97@gmail.com

***Abstrak***

*Industri telekomunikasi dengan skalabilitas yang tinggi tentunya sangat membutuhkan prototipe teknologi jaringan untuk dapat bekerja secara terpusat di berbagai kondisi ekosistem jaringan. Melalui teknologi Virtual Network yaitu VLAN yang dapat mendedikasikan perangkat jaringan tersendiri yang secara fisik terletak pada satu struktur jaringan yang logis. Tujuan dari peneilitan ini adalah untuk menganalisis implementasi VLAN pada lingkungan Software Defined Networking (SDN). Beberapa tipe controller dievaluasi menggunakan metrik jaringan seperti throughput, jitter, latency dan packet delivery ratio yang memanfaatkan Openflow sebagai protokol komunikasi yang menghubungkan bidang kontrol dan bidang data. Peneliti memfokuskan untuk mengevaluasi kemampuan controller SDN Opendaylight, floodlight, ONOS, POX dan Ryu pada Openflow stream forwarding yang dapat memasang dan menghapus header VLAN. Hasil pengujian skenario yang diusulkan menunjukkan bahwa semua jenis controller dapat berjalan dengan baik dengan memberikan instruksi ke lapisan forwarding yang dilewati paket data saat dikirimkan. Melalui pengujian konektivitas dan parameter metrik jaringan, karakteristik controller yang diharapkan dan direkomendasikan adalah controller yang berjalan pada bahasa pemrograman Java. Performa dan modularitas yang tinggi sebagai keunggulannya, fitur yang memadai, meskipun lebih sulit untuk dipahami dibandingkan dengan yang berjalan pada berbasis python. Controller jaringan SDN seperti ONOS mendapatkan hasil yang signifikan dalam proses pengiriman data aktual dengan nilai rata-rata nilai throughput 13.570 kbps. Sedangkan Opendaylight menunjukkan persentase delivery ratio yang stabil untuk semua skenario pengiriman data*

***Kata kunci:*** *SDN, VLAN, VLAN Tagging, Controller*

## 1. Pendahuluan

Gagasan dibalik munculnya paradigma jaringan Software Defined Network (SDN) tentu bukanlah menjadi suatu hal yang baru, pasalnya pada penelitian sebelumnya mendapatkan hasil bahwa pengembangan banyak dilakukan seiring meningkatnnya kompleksitas dan pengelolaan jaringan [1]. Terlepas dari itu, controller SDN memiliki banyak keterbatasan pada kinerja dan skalabilitasnya[2]. Dalam skala jaringan yang besar dan meningkatnya beban dalam jaringan juga menjadikan solusinya semakin sulit[3]. Banyak solusi yang ditawarkan dalam penggunaan metode seperti contoh algoritma SSOA yang ditawarkan oleh Abdelhamied A. Ateya dkk dalam penelitiannya. Programabilitas yang diberikan oleh controller sangat mempengaruhi pengembangan yang efisien untuk mengelola sumber daya yang sangat kompleks pada jaringan SDN[4]. Pada dasarnya penyeimbangan antara switch dan controller sangat penting untuk meningkatkan penggunaan sumber daya dan stabilitas serta mengurangi overhead pada controller[5]. Berdasarkan hal itu penyebaran controller SDN dituliskan berdasarkan skema latensi yang difokuskan pada transmission delay (TD) dan propagration delay (PD) untuk mempertimbangkan masalah penempatan pada controller SDN[6].

Meskipun konsep jaringan SDN dan protokol OpenFlow dapat menggantikan semua jenis manajemen jaringan yang terbilang sangat kompleks seperti saat ini, namun penggunaan VLAN (Virtual LAN) seperti yang disebutkan dalam penelitian sebelumnya masih di perlukan[7]. Tujuan utama teknologi VLAN adalah untuk membatasi ¬broadcast domain dengan membagi host berdasarkan input port ke switch atau alamat MAC[8]. Dengan VLAN memungkinkan dapat memanajemen suatu host agar terhubung dan berkomunikasi seolah-olah berada pada satu jaringan yang sama walapun secara fisik berbeda[8]. Dengan mengelompokkan sejumlah jaringan menjadi bagian-bagian kecil dalam skala yang besar seperti institusi pendidikan atau perusahaan pada infrastruktur fisik yang sama memungkinkan eksploitasi yang efektif dengan biaya yang rendah[9]. VLAN dinamis biasanya berbasis MAC Address dan switch akan mendeteksi dan mengafiliasi VLAN host yang terhubung ke salah satu portnya[10]. Sedangkan VLAN statis administrator jaringan harus melakukan konfigurasi secara manual

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

untuk membuat VLAN pada switch kemudian menentukan port ke VLAN menggunakan Command Line Interface (CLI)[7].

Pada penelitian yang sudah pernah dilakukan sebelumnya oleh Rohmat Tulloh, Ridha Muldina Negara, Arif Nur Hidayat menghasilkan bahwa VLAN dapat berjalan sesuai dengan apa yang diinginkan dan berjalan dengan baik. Penulis menyebutkan dalam pengujiannya bahwa setup time untuk jumlah host yang semakin bertambah setup time akan semakin besar dengan menggunakan controller POX[8]. Nilai RTT nya untuk tiga jenis topologi didapatkan hasil yang stabil mulai dari 0,2 – 6 second[8]. Dapat dilihat dari host yang memiliki VLAN ID yang sama dapat berkomunikasi sedangkan dengan VLAN ID yang berbeda tidak dapat berkomunikasi, dikarenakan POX controller memiliki fungsi dapat membuat dan melepas header VLAN yang akan masuk ke switch dan juga mencocokan header VLAN sehingga adopsi teknologi VLAN dapat bekerja pada jarigan SDN[8].

Pada penelitian terdahulu yang dilakukan oleh Sawung Murdha Anggara juga membahas mengenai performa controller pada jaringan SDN. Dari hasil pengujian yang telah dilakukan menunjukan bahwa nilai latency pada kontroler floodlight lebih kecil dibandingkan kontroler POX dengan jumlah switch yang variatif. Meskipun untuk jumlah switch diatas 100 buah, floodlight akan mengalami penurun yang drastis mencapai hampir 51% namun POX memberikan jaminan dengan waktu respon yang konstan dengan berapun jumlah switchnya dibandingkan floodlight[11]. Sedangkan untuk nilai throughputnya floodlight mampu memberikan aliran data yang lebih besar dibandingkan POX mencapai jumlah switch 1000 buah. Tapi POX memberikan jaminan pengelolaan aliran data yang konstan dibanding floodlight meskipun dengan dengan kemampuan pengelolaan aliran data yang lebih rendah dibandingkan floodlight[11].

Pada penelitian lain juga seperti yang dilakukan oleh Rahmatullah Khondoker, Adel Zaalouk, Ronald Marx, Kpatcha Bayarou yang membahas mengenai pemilihan controller SDN sesuai kebutuhan yang digunakan berdasarkan persayaratan tertentu[12]. Metode yang digunkan untuk pemilihan controller berdasarkan kriteria tertentu adalah Multi Criteria Decision Making (MCDM). Analytic Hierarchy Process (AHP) adalah salah satu metode MCDM yang memberikan nilai pada fitur disetiap controller yang akan dianalisa. Pendekatan ini melihat dari segi modularitas dan produktivitas setiap controller pada performansi yang diberikan. Akhirnya dari hasil analisa yang dilakukan pada kelima kontroler tersebut Ryu dipilih sebagai yang terbaik berdasarkan ketentuan dari AHP.

Penelitian ini diusulkan untuk menganalisa performansi controller jenis lain pada jaringan VLAN SDN. Diperlukan sebuah kriteria tertentu sebagai acuan pengguna untuk memilih pengontrol yang terbaik pada teknologi SDN yang berbasis VLAN. Meskipun controller dikembangkan berdasarkan kebutuhan perkembangan zaman terutama pada pengembangan pengontrol jaringan SDN, penelitian seperti ini masih belum pernah dilakukan sebelumnya. Jenis controller yang akan dianalisa adalah Floodlight, Opendaylight, ONOS, Pox dan Ryu dimana Floodlight, Opendaylight dan ONOS diprogram menggunakan bahasa pemrograman java sedangkan Pox dan Ryu menggunakan python. Parameter yang ditentukan pada penelitian ini berupa *latency, throughput, jitter, packet delivery ratio*.

## 2. Metode Penelitian

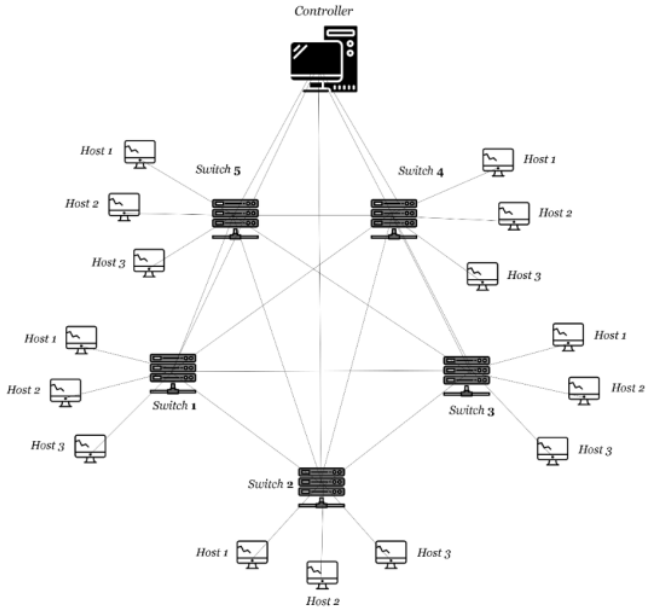
Rancangan topologi yang akan digunakan pada penelitian ini adalah topologi mesh dengan 5 buah switch (OVS) dan 15 host dimana masing-masing switch terkoneksi dengan 3 buah host. Masing-masing host dikonfigurasi dengan VLAN ID, VLAN ID yang digunakan adalah 200, 300 dan 400. Topologi dapat dilihat pada gambar 1:

Konfigurasi pada tabel 2 adalah custom script yang dibuat sendiri menggunakan bahasa pemrograman python, dimana kita dapat mengkonfigurasi sendiri sesuai dengan keinginan kita. Setelah script selesai dibuat, kita akan jalankan menggunakan mininet sehingga akan membentuk topologi jaringan seperti pada gambar 1.

Pengujian konektivitas sangat berpengaruh besar terhadap pengujian verifikasi VLAN. Artinya bahwa secara hirarki pengujian konektivitas bertujuan untuk mengetahui apakah controller dapat mengenali topologi jaringan yang dijalankan pada mininet. Topologi yang sebelumnya sudah terkonfigurasi memberikan port tag VLAN di masing-masing host pada open virtual switch (OVS). Pengujian ini dilakukan dengan mengirimkan paket ICMP dari host 1 yang terhubung secara langsung dengan switch 1 menuju host 13 yang terhubung secara langsung dengan switch 5. Host 1 dan host 13 memiliki tag VLAN ID yang sama yaitu 100, secara konsisten host yang memiliki tag vlan id 200 dan

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

300 juga dilakukan pengujian yang sama dengan vlan id 100. Tentunya setiap controller memiliki konfigurasi yang berbeda untuk mengaktifkan fungsi memasang dan melepas header VLAN. Namun, kontroler floodlight dan opendaylight sama-sama dapat mengaktifkan fungsi ini secara otomatis ketika sebuah port diberikan tag VLAN ID sehingga penulis tidak perlu lebih lanjut mengkonfigurasi controller yang diuji.




**Gambar 1** Topologi Dalam Penelitian

**Tabel 1** IP Address dan VLAN ID

| Switch | Host | IP Adrress | VLAN ID |
|---|---|---|---|
| | 1 | 10.0.0.1/16 | 200 |
| *Switch* 1 | 2 | 10.0.0.2/16 | 300 |
| | 3 | 10.0.0.3/16 | 400 |
| | 1 | 10.1.0.1/16 | 200 |
| *Switch* 2 | 2 | 10.1.0.2/16 | 300 |
| | 3 | 10.1.0.3/16 | 400 |
| | 1 | 10.2.0.1/16 | 200 |
| *Switch* 3 | 2 | 10.2.0.2/16 | 300 |
| | 3 | 10.2.0.3/16 | 400 |
| | 1 | 10.3.0.1/16 | 200 |
| *Switch* 4 | 2 | 10.3.0.2/16 | 300 |
| | 3 | 10.3.0.3/16 | 400 |
| | 1 | 10.4.0.1/16 | 200 |
| *Switch* 5 | 2 | 10.4.0.2/16 | 300 |
| | 3 | 10.4.0.3/16 | 400 |

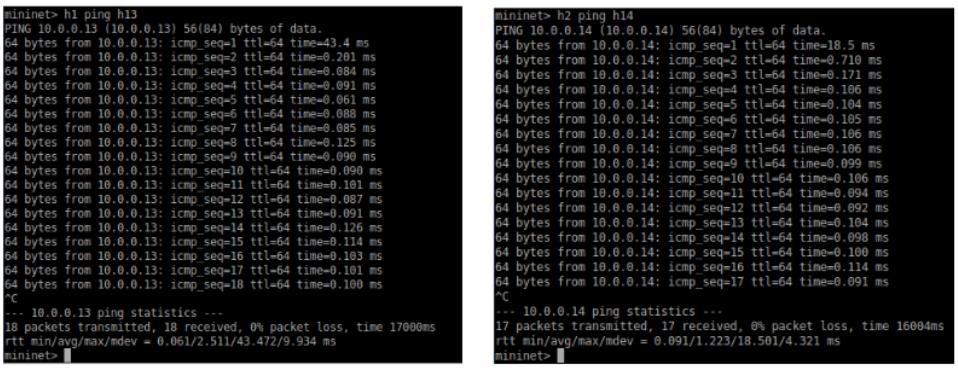
Selain itu, pengujian pengambilan data *latency, throughput, jitter, packet delivery ratio* dilakukan dengan tujuan untuk mengetahui seberapa besar tingkat efektivitas kecepatan aktual transfer data dari node awal menuju node akhir pada jaringan *Software Defined Network* (SDN) berbasis VLAN. Dengan skenario h1 akan menjadi server sedangkan h13 akan menjadi *client* yang mengirimkan beban *traffic* UDP secara variatif mulai dari 64Mb, 128Mb sampai dengan 1024Mb menggunakan aplikasi

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

iperf. Iperf digunakan untuk mendefinisikan jenis paket yang akan dikirimkan dan variasi beban *traffic* pada skenario pengujian ini. H1 dan h13 adalah *host* yang memiliki tag VLAN ID yang sama yaitu 100, pada skenario yang telah dijelaskan diatas dilakukan secara berkala sebanyak 2-3 kali untuk mendapatkan hasil yang akurat. Untuk mendapatkan nilai *latency, throughput, jitter, packet delivery ratio* digunakan aplikasi wireshark yang secara bersamaan melakukan monitoring jaringan pada saat h13 mengirimkan beban *traffic* variatif menuju h1.

## 3. Hasil dan Pembahasan

Setelah dilakukan pengujian mendapatkan hasil bahwa setiap host yang memiliki VLAN ID yang sama berhasil mengirimkan dan menerima paket ICMP, sedangkan antar host yang memiliki tag VLAN ID yang berbeda tidak berhasil dalam mengirimkan dan menerima paket ICMP. Berikut hasil yang didapatkan




a) VLAN ID 100 b) VLAN ID 200

**Gambar 2** Hasil Pengujian Verifikasi VLAN ID *a) dan b)

**Tabel 2** Hasil Pengujian *Throughput*

| Traffic Size | Floodlight (kbps) | Opendaylight (kbps) | POX (kbps) | ONOS (kbps) | Ryu (kbps) |
|---|---|---|---|---|---|
| 64 Mb | 4.550 | 8.761 | 879 | 13.570 | 5.114 |
| 128 Mb | 5.884 | 9.126 | 3.475 | 7.097 | 8.357 |
| 256 Mb | 13.487 | 14.058 | 4.551 | 17.671 | 7.679 |
| 512 Mb | 8.741 | 15.034 | 16.737 | 18.537 | 10.089 |
| 1024 Mb | 7.585 | 13.755 | 11.472 | 7.262 | 13.093 |

Berdasarkan tabel perbandingan *throughput* diatas menunjukkan hasil pada beban traffic sebesar 64Mb, ONOS adalah controller dengan nilai paling besar dibandingkan dengan yang lainnya yaitu 13,570 kbps. Nilai paling kecil dihasilkan dari kontroler POX yaitu 879 kbps, sedangkan controller ODL, floodlight dan Ryu cenderung nilainya berada dibawah ONOS dan diatas POX. Ketika beban traffic diberi kelipatan dengan nilai 128Mb ONOS menurun drastis berada dibawah kontroler ODL dan Ryu, menjadi urutan ketiga dalam penanganan flow/sec yang diberikan oleh controller.

Nilai fluktuatif throughput yang dapat dilihat pada grafik perbandingan diatas dengan terus menerus ditambah beban ¬traffic tidak terlepas dari modularitas dan link infrastructure layer tiap masing-masing kontroler. Dimana ONOS mendominasi dengan rata-rata menghasilkan nilai throughput paling besar, artinya bahwa ONOS mampu merespon aliran flow/sec yang lebih besar dibandingkan

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

dengan kontroler lain. Secara fitur ONOS juga termasuk high performance dalam hal modularity, berdasarkan application domain skalabilitas ONOS lebih besar yaitu data center, WAN dan transport.

Penambahan jumlah beban traffic cukup berpengaruh pada penanganan flow disetiap controller, berbeda dengan ODL yang memberikan nilai throughput cukup stabil disetiap perubahan traffic karena modularitas dan skalabilitasnya cukup tinggi. Di lain sisi floodlight memiliki kelemahan menempati ruang memori yang cukup besar untuk memberikan aliran respon terhadap request yang cukup besar[18]. Hal ini mempengaruhi nilai throughput yang tidak cukup tinggi yang dihasilkan oleh floodlight dari skenario pada penelitian ini.

**Tabel 3** Hasil Pengujian *Jitter*

| Traffic Size | Floodlight (ms) | Opendaylight (ms) | POX (ms) | ONOS (ms) | Ryu (ms) |
|---|---|---|---|---|---|
| 64 Mb | 125.051 | 25.233 | 27.003 | 24.927 | 124.477 |
| 128 Mb | 99.331 | 0.132 | 0.111 | 0.326 | 27.638 |
| 256 Mb | 0.007 | 0.015 | 2.479 | 0.004 | 0.780 |
| 512 Mb | 0.024 | 0.001 | 0.002 | 0.003 | 4.756 |
| 1024 Mb | 0.001 | 0.010 | 0.001 | 222.114 | 1.028 |

Hasil pengujian yang dibandingan melalui tabel diatas menunjukan nilai jitter ODL adalah yang paling stabil dibandingkan dengan controller jenis lainnya dengan waktu penundaan disetiap node adalah 0,001 ms. Rata-rata nilai yang didapatkan kurang dari 0,133 ms pada skenario pengujian 128Mb dan kelipatan selanjutnya dengan nilai paling tinggi 25,233 ms pada beban traffic 64Mb. Meskipun kontroler ONOS dan floodlight sama-sama dibangun menggunakan java, dapat dilihat bahwa ONOS memiliki kinerja lebih baik dengan selisih cukup jauh pada skenario traffic 64Mb. Berdasarkan fitur modularitas dan domain aplikasi ONOS lebih unggul dengan high-performance serta skalabilitas yang luas. Sesuai dengan hasil penelitian yang dilakukan oleh Arwa Hamad Eljack, Ahmed Hassan M. Hassan dan Hala Hassan Elami menunjukkan hasil waktu penundaan terbesar ONOS adalah 0,153 ms dan floodlight mencapai 25 ms [23]. Pada skenario terakhir pengujian ini ONOS menunjukkan latency yang paling tinggi, hal ini bisa saja terjadi tergantung kondisi suatu jaringan pada saat pengujian berlangsung. Sedangkan POX dan Ryu cenderung menunjukkan nilai jitter yang fluktuatif naik atau turun dari keseluruhan skenario pengujian, namun POX memiliki kinerja yang lebih baik dibanding Ryu karena masih masih memenuhi standar ITU-T G dengan kategori degradasi bagus yaitu dibawah 75 ms.

**Tabel 4** Hasil Pengujian *Latency*

| Traffic Size | Floodlight (ms) | Opendaylight (ms) | POX (ms) | ONOS (ms) | Ryu (ms) |
|---|---|---|---|---|---|
| 64Mb | 0.00013 | 0.00013 | 0.00028 | 0.00011 | 0.00013 |
| 128Mb | 0.00012 | 0.00010 | 0.00026 | 9.4 | 0.00013 |
| 256Mb | 0.00002 | 3.0 | 0.00025 | 0.00002 | 0.00013 |
| 512Mb | 2.5 | 2.2 | 0.00002 | 2.3 | 0.00012 |
| 1024Mb | 2.3 | 0.00002 | 2.3 | 3.6 | 0.00010 |

Berdasarkan tabel perbandingan diatas menunjukkan nilai latency pada kontroler ONOS memiliki respon yang lebih cepat dibandingkan dengan kontroler floodlight, opendayligth, POX dan Ryu pada traffic 64Mb. Hal ini dipengaruhi berdasarkan hasil pengujian throughput pada ONOS yang mampu menangani aliran flow/sec yang cukup besar pada skenario pengujian dengan traffic yang didefinsikan sebesar 64Mb. Karena pada dasarnya bandwith dan latency adalah dua hal yang saling berkaitan dalam hal kecepatan suatu jaringan, biasanya rendah dan tinggi nilai latency dipengaruhi bandwith dan sumber penundaan atau delay[22]. Namun pada skenario selanjutnya dengan beban traffic

ISSN (Cetak) 2527-6042
eISSN (Online) 2527-6050

128 ONOS justru menunjukkan nilai latency paling besar dibandigkan dengan kontroler lain. Berbeda dengan Ryu yang menunjukkan nilai cukup stabil di semua skenario pengujian ini dengan traffic 64-1024 Mb. Hal ini memungkinkan Ryu digunakan untuk layanan komunikasi seperti konfrensi video dan streaming online[24]. Seperti pada penelitian dengan judul "Performance Analysis of POX and Ryu with Different SDN Topologies" menunjukkan persentase penurunan latency pada Ryu sampai 99,90% untuk topologi linear, tree dan DCN. Floodlight dan POX menghasilkan nilai cukup tinggi mencapai 2,5 miliseconds untuk POX dan 2,3 ms untuk floodlight. Sedangkan ODL menunjukkan hasil yang hampir mirip dengan POX dengan latency paling besar mencapai 3,0 miliseconds.

**Tabel 6** Hasil Pengujia *Packet Delivery Ratio*

| Traffic Size | Floodlight (%) | Opendaylight (%) | POX (%) | ONOS (%) | Ryu (%) |
|---|---|---|---|---|---|
| 64 Mb | 99,95 | 99,93 | 99,50 | 99,94 | 99.92 |
| 128 Mb | 99,94 | 99,96 | 99,97 | 99,92 | 99,87 |
| 256 Mb | 94,08 | 99,91 | 99,98 | 99,99 | 99,88 |
| 512 Mb | 99,91 | 99,99 | 99,96 | 99,99 | 99,88 |
| 1024 Mb | 99,92 | 99,94 | 99,99 | 99,89 | 99,98 |

Hasil perbandingan tabel diatas menunjukkan bahwa kontroler ODL memiliki tingkat keberhasilan pengiriman data diatas 99,90% dari keseluruhan beban traffic variatif. Hal ini dikarenakan hasil pengujian jitter yang stabil pada kontroler Opendayligth, karena semakin besar tingkat jitter akan semakin kecil tingkat persentase PDR. Seperti yang terjadi pada controller ONOS saat pengujian dengan beban traffic 1024Mb yang memberikan nilai jitter tinggi sehingga mempengaruhi persentase PDR yaitu 99,89%, lebih kecil dibandingkan dengan keseluruhan variatif beban traffic. Selain itu POX juga menunjukkan nilai PDR yang cukup stabil pada skenario pengujian 128Mb dan kelipatan berikutnya. Meski terbilang cukup mudah untuk memprogram POX kelemahan mendasar darinya adalah masalah performa dan ruang lingkup pengebangan kontroler tersebut [25]. Floodlight memiliki kinerja yang lebih baik jika dibanding Ryu dengan nilai rata-rata PDR diatas 99,90% kecuali pada saat beban traffic didefinisikan 256Mb. Controller yang dirancang menggunakan bahasa pemrograman java relatif memberikan performansi yang lebih baik dibandingkan dengan yang dibangun menggunakan python, dalam hal ini adalah packet delivery ratio maupun beberapa parameter lain yang digunakan dalam penelitian ini.

## 4. Kesimpulan

Setelah melakukan eksperimen dan analisis terhadap hasil skenario pengujian, dapat disimpulkan bahwa kontroler yang dirancang menggunakan bahasa pemograman java menjamin dalam beberapa metric jaringan yang dibutuhkan saat ini. Hal ini didukung dengan high performance dari modularitas controller berbasis java seperti ONOS dan Opendaylight. ONOS menjamin kestabilan dalam hal penanganan aliran flow/sec yang cukup besar dibandingkan dengan controller yang di program menggunakan python. Sedangkan Opendaylight memiliki waktu penundaan yang cukup singkat pada saat pengujian dilakukan. Pada saat sebuah paket ditransmisikan dalam jaringan penundaan waktu sering terjadi dan mempengaruhi persentase keberhasilan pengiriman. ODL menunjukkan persentase pengiriman yang stabil dengan nilai diatas 99,90% untuk semua skenario yang diuji. Meski dalam hal waktu respon kontroler terhadap paket yang masuk Ryu adalah yang paling stabil, mempertimbangkan metric jenis lain sangat diperlukan pada penelitian di masa depan dengan kriteria yang lebih mendekati dengan kebutuhan layanan jaringan di masa mendatang.

## Referensi

[1] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several Software Defined Networking controllers," 2015 12th Int. Conf. Telecommun. Mod. Satell. Cable Broadcast. Serv. TELSIKS 2015, pp. 223–226, 2015, doi: 10.1109/TELSKS.2015.7357774.
[2] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," J. Netw. Comput. Appl., vol. 103, pp. 101–118, 2018, doi: 10.1016/j.jnca.2017.11.015.
[3] A. A. Ateya et al., "Chaotic salp swarm algorithm for SDN multi-controller networks," Eng. Sci. Technol. an Int. J., vol. 22, no. 4, pp. 1001–1012, 2019, doi: 10.1016/j.jestch.2018.12.015.
[4] K. Poularakis, Q. Qin, E. M. Nahum, M. Rio, and L. Tassiulas, "Flexible SDN control in tactical ad hoc networks," Ad Hoc Networks, vol. 85, pp. 71–80, 2019, doi: 10.1016/j.adhoc.2018.10.012.
[5] F. AL-Tam and N. Correia, "Fractional switch migration in multi-controller software-defined networking," Comput. Networks, vol. 157, pp. 1–10, 2019, doi: 10.1016/j.comnet.2019.04.011.
[6] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum control latency of SDN controller placement," Proc. - 15th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 10th IEEE Int. Conf. Big Data Sci. Eng. 14th IEEE Int. Symp. Parallel Distrib. Proce, pp. 2175–2180, 2016, doi: 10.1109/TrustCom.2016.0334.
[7] V. G. Nguyen and Y. H. Kim, "SDN-based enterprise and campus networks: A case of VLAN management," J. Inf. Process. Syst., vol. 12, no. 3, pp. 511–524, 2016, doi: 10.3745/JIPS.03.0039.
[8] R. Tulloh, R. M. Negara, and A. N. Hidayat, "Simulasi Virtual Local Area Network (VLAN) Berbasis Software Defined Network (SDN) Menggunakan POX Controller," J. INFOTEL - Inform. Telekomun. Elektron., vol. 7, no. 2, p. 129, 2015, doi: 10.20895/infotel.v7i2.40.
[9] M. B. Lehocine and M. Batouche, "Flexibility of managing VLAN filtering and segmentation in SDN networks," 2017 Int. Symp. Networks, Comput. Commun. ISNCC 2017, 2017, doi: 10.1109/ISNCC.2017.8071999.
[10] M. Koerner and O. Kao, "MAC Based Dynamic VLAN Tagging with OpenFlow for WLAN Access Networks," Procedia Comput. Sci., vol. 94, pp. 497–501, 2016, doi: 10.1016/j.procs.2016.08.077.
[11] S. M. Anggara, "Pengujian Performa Kontroler Software-defined Network (SDN): POX dan Floodlight 2012/2 013," p. 44, 2015.
[12] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," 2014 World Congr. Comput. Appl. Inf. Syst. WCCAIS 2014, 2014, doi: 10.1109/WCCAIS.2014.6916572.
[13] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," Proc. 18th Mediterr. Electrotech. Conf. Intell. Effic. Technol. Serv. Citizen, MELECON 2016, no. 978, pp. 18–20, 2016, doi: 10.1109/MELCON.2016.7495430.
[14] I. Ummah, "Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking," Indones. J. Comput., vol. 1, no. 1, pp. 95–106, 2016, doi: 10.21108/indojc.2016.1.1.20.
[15] J. Chen, J. Chen, J. Ling, and W. Zhang, "Failure recovery using vlan-tag in SDN: High speed with low memory requirement," 2016 IEEE 35th Int. Perform. Comput. Commun. Conf. IPCCC 2016, 2017, doi: 10.1109/PCCC.2016.7820627.
[16] A. Mendiola et al., "Multi-domain bandwidth on demand service provisioning using SDN," IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conf. Work. Software-Defined Infrastruct. Networks, Clouds, IoT Serv., pp. 353–354, 2016, doi: 10.1109/NETSOFT.2016.7502407.
[17] S. Asadollahi and B. Goswami, "Experimenting with scalability of floodlight controller in software defined networks," Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEECCOT 2017, vol. 2018-Janua, pp. 288–292, 2018, doi: 10.1109/ICEECCOT.2017.8284684.
[18] C. Fancy and M. Pushpalatha, "Performance evaluation of SDN controllers POX and floodlight in mininet emulation environment," Proc. Int. Conf. Intell. Sustain. Syst. ICISS 2017, no. Iciss, pp. 695–699, 2018, doi: 10.1109/ISS1.2017.8389262.
[19] A. Bondkovskii, J. Keeney, S. Van Der Meer, and S. Weber, "Qualitative comparison of open-source SDN controllers," Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp., no. Noms, pp. 889–894, 2016, doi: 10.1109/NOMS.2016.7502921.

[20] Y. Zhao, L. Iannone, and M. Riguidel, "On the performance of SDN controllers: A reality check," 2015 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Network, NFV-SDN 2015, pp. 79–85, 2016, doi: 10.1109/NFV-SDN.2015.7387410.

[21] A. Vishnu Priya and N. Radhika, "Performance comparison of SDN OpenFlow controllers," Int. J. Comput. Aided Eng. Technol., vol. 11, no. 4–5, pp. 467–479, 2019, doi: 10.1504/IJCAET.2019.100444.

[22] M. I. Lali, R. U. Mustafa, F. Ahsan, M. S. Nawaz, and W. Aslam, "Performance Evaluation of Software Defined Networking vs. Traditional Networks," A quartelry Int. Sci. J., vol. 54, no. 1, pp. 16–22, 2017, [Online]. Available: http://www.thenucleuspak.org.pk/index.php/Nucleus/article/view/95/24.

[23] A. H. Eljack, A. H. M. Hassan, and H. H. Elamin, "Performance analysis of ONOS and floodlight SDN controllers based on TCP and UDP traffic," Proc. Int. Conf. Comput. Control. Electr. Electron. Eng. 2019, ICCCEEE 2019, 2019, doi: 10.1109/ICCCEEE46830.2019.9071189.

[24] J. Ali, S. Lee, and B. H. Roh, "Performance analysis of POX and Ryu with different SDN topologies," ACM Int. Conf. Proceeding Ser., pp. 244–249, 2018, doi: 10.1145/3209914.3209931.

[25] M. F. M. Ryhan Uddin, "Performance Analysis of SDN Based Firewalls: POX vs. ODL," IEEE, 2019, doi: 10.1109/ICAEE48663.2019.8975667.

# denar

ORIGINALITY REPORT

| 6% | 5% | 1% | 3% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Submitted to Universitas Islam Indonesia<br>Student Paper | 3% |
| 2 | media.neliti.com<br>Internet Source | 2% |
| 3 | eprints.umm.ac.id<br>Internet Source | 1% |

| | | | |
|---|---|---|---|
| Exclude quotes | Off | Exclude matches | < 1% |
| Exclude bibliography | On | | |