Analisis Kinerja Arsitektur Software-Defined Network Berbasis OpenDaylight Controller

Melissa[1], Susmini Indriani Lestariningati[2]
Jurusan Teknik Komputer, Universitas Komputer Indonesia, Bandung
mels@email.unikom.ac.id[1], susmini.indriani@email.unikom.ac.id[2]

ABSTRAK

Jumlah pengguna Internet beserta layanannya setiap tahunnya semakin meningkat, sehingga kebutuhan akan jaringan yang cepat, handal, dan fleksibel sangat diperlukan. Namun, konfigurasi jaringan atau protokol yang mendasarinya tidak berubah menyebabkan keterbatasan pada saat adanya perubahan atau penambahan infrastruktur jaringan. Dalam menangani masalah tersebut telah dikembangkan sebuah arsitektur baru yaitu, *Software-Defined Network* (SDN). Jaringan SDN ini dapat memisahkan antara data dan control pada jaringan, sehingga pengontrolannya dapat dilakukan secara terpusat. Pada penelitian ini dilakukan analisis kinerja pada arsitektur jaringan SDN dengan menggunakan OpenDaylight sebagai kontrolernya. Analisis kinerja berdasarkan hasil pengukuran parameter *Quality of Sevices* (QoS) yang kemudian dibandingkan dengan hasil parameter QoS pada jaringan konvensional. Selain itu, dari penelitian ini juga dapat diketahui bagaimana fleksibiltas atau skalabilitas pada penerapan SDN.

Kata kunci: Mininet, OpenDaylight, QoS, SDN.

1. PENDAHULUAN

Selama hampir satu dekade jumlah dari pengguna atau perangkat yang terkoneksi dengan Internet, aplikasi serta layanan yang berjalan pada sebuah jaringan meningkat pesat. Namun, konfigurasi jaringan atau protokol yang mendasarinya tidak banyak berubah. Contoh, proses routing biasanya didasarkan pada IP Prefix tujuan dan manajemen jaringan yang masih kaku karena dibatasi oleh perbedaan konfigurasi oleh masing-masing vendor. Sehingga, ketika sebuah aturan telah diterapkan pada sistem jaringan, satu-satunya cara untuk mengubah konfigurasi yaitu dengan mengubah konfigurasi pada keseluruhan perangkat yang terkoneksi. Sehingga sangat menyita waktu dan keterbatasan pada skalabilitas dan mobilitas. Sedangkan saat ini, kebutuhan akan jaringan yang cepat, fleksibel, handal dan tanpa batasan vendor sangat diperlukan.

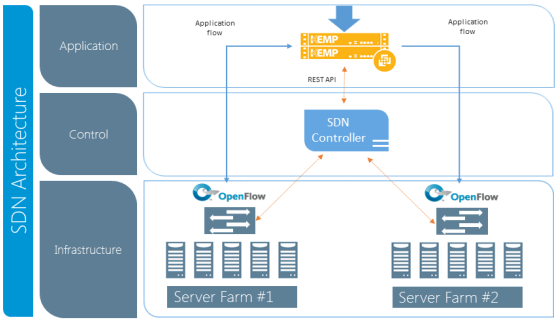
Salah satu pendekatan yang dilakukan mengenai masalah diatas adalah sebuah arsitektur atau protokol jaringan baru yaitu *Software-Defined Network* (SDN). SDN dibuat dengan tujuan menggantikan jaringan konvensional yang ada saat ini, dimana pendekatan ini memungkinkan sebuah jaringan dapat dikelola dan diprogram lebih mudah oleh administrator jaringan. Prinsip utama dari SDN adalah memisahkan bidang kontrol (*Control Plane*) dan bidang data (*Data Plane*), dimana bidang kontrol berisi logika kontrol sedangkan bidang data berisi infrastruktur fisik. Bidang kontrol bertindak sebagai otak dari jaringan yang memiliki kontrol langsung atas bidang data yang bertindak sebagai penerus paket atau switching.

Oleh karena itu, agar dapat mengetahui bagaimana arsitektur SDN bekerja, dibangunlah sebuah sistem dengan arsitektur SDN yang menggunakan sejumlah skenario berbeda. Terdapat beberapa variabel yang akan diuji dalam penelitian ini seperti *delay*, *jitter*, *throughput*, dan *packet loss*. Data yang didapatkan akan diolah dan di analisis sehingga dapat diketahui sejauh mana arsitektur SDN berpengaruh dalam sebuah jaringan nyata dan bagaimana perbandingannya dengan arsitektur jaringan konvensional. Selain itu, dengan menggunakan beberapa skenario dapat diketahui bagaimana fleksibitas dan skalabilitas yang dimiliki oleh arsitektur SDN.

2. TINJAUAN PUSTAKA

2.1. Software-Defined Network

Software-Defined Network (SDN) adalah sebuah arsitektur baru dalam bidang jaringan komputer, yang bersifat dinamis, *manageable*, *cost-effective*, dan *adaptable*, sehingga sangat ideal untuk kebutuhan jaringan saat ini yang bersifat dinamis dan bandwidth yang besar. Arsitektur ini memisahkan antara fungsi *control* dan fungsi *forwarding*, sehingga *network control* tersebut menjadi *directly programmable* (dapat diprogram secara langsung), sedangkan infrastruktur yang mendasarinya dapat diabstraksikan untuk layer aplikasi dan layanan jaringan [1].

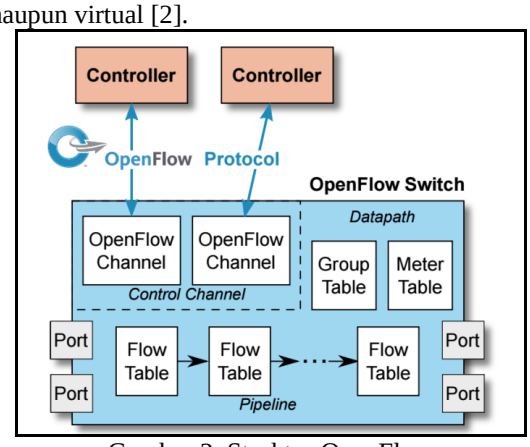


Gambar 1 Arsitektur SDN

Secara umum dalam perangkat jaringan terdapat tiga proses utama, yaitu *Application Plane*, *Control Plane* dan *Data Plane. Application Plane* merupakan bagian paling atas yang menyediakan interface kepada user dalam pembuatan program aplikasi yang kemudian akan mengatur dan mengoptimalkan penggunaan jaringan. *Control Plane* adalah bagian yang berfungsi untuk mengontrol dan memanajemen jaringan, mengkonfigurasi sistem, menentukan informasi *routing table* dan *forwarding table*. *Data Plane* adalah bagian yang bertanggung jawab dalam meneruskan paket, selain itu juga menguraikan header paket, mengatur QoS, dan enkapsulasi paket [2].

2.2. OpenFlow

OpenFlow adalah standar antarmuka komunikasi atau protokol yang menghubungkan antara *data plane* dengan *control plane* pada arsitektur SDN. *OpenFlow* mengijinkan akses langsung ke *data plane* dari sebuah perangkat jaringan seperti switch dan router baik secara fisik maupun virtual [2].



Gambar 2. Struktur OpenFlow

Protokol OpenFlow pada arsitektur jaringan SDN dapat mengatasi kebutuhan bandwidth yang tinggi, aplikasi dan layanan yang bersifat dinamis, dan secara signifikan mengurangi operasi kompleksitas manajemen pada jaringan yang bersifat adaptif dan selalu berubah-ubah. Secara sederhana *OpenFlow* merupakan sebuah antarmuka antara SDN kontroler dengan perangkat switch [5].

2.3. OpenDaylight Controller

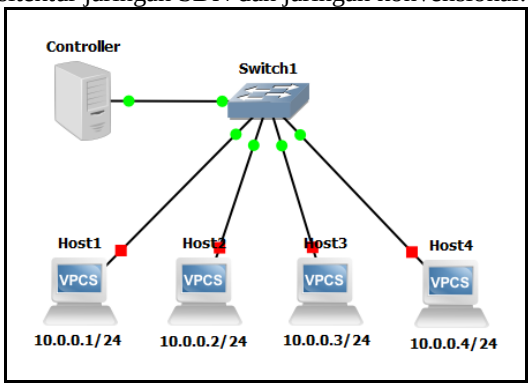
Kontroler SDN merupakan sebuah aplikasi SDN yang dapat mengelola *flow control* untuk mengaktifkan *inteligence networking*. Kontroler SDN bekerja berdasarkan protokol seperti *OpenFlow* yang memungkinkan server memberitahu kemana paket dikirimkan. Perangkat meneruskan paket data yang diterima berdasarkan aturan yang ditetapkan dari kontroler [6].

2.4. Mininet

Mininet adalah emulator jaringan SDN yang dapat mensimulasikan kinerja antara *end-host*, switch, router, kontroler, dan link dalam sebuah kernel Linux [6].

3. PERANCANGAN SISTEM

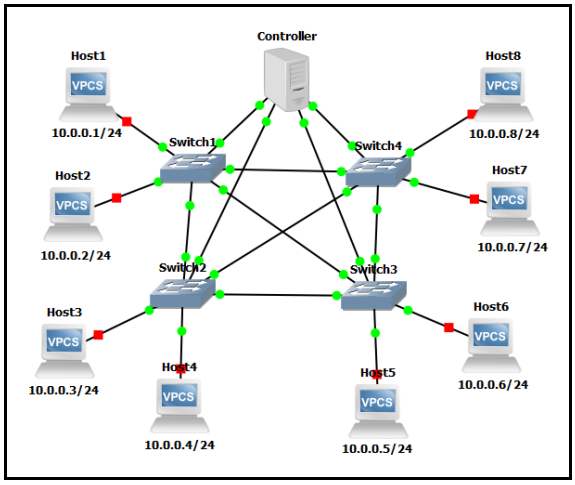
Perancangan sistem dibuat untuk menggambarkan tentang keseluruhan sistem yang akan dibangun, yaitu jaringan Software-Defined Network. Sistem ini dibangun dengan dua topologi yaitu, Topologi Star dan Topologi Mesh. Dan masing-masing topologi akan diterapkan pada arsitektur jaringan SDN dan jaringan konvensional.



Gambar 3. Topologi Star

Pada Topologi Star atau bisa disebut Topologi Single, implementasi akan dilakukan beberapa skenario dengan meningkatkan jumlah host sejumlah range kelas *IP Adress* yang diberikan yaitu 10.10.10.0/24. Disini hanya menggunakan satu *switch OpenFlow* dan satu kontroler.

Selajutnya pada topologi Mesh juga akan dibuat dengan beberapa skenario dengan peningkatan jumlah switch, dimana masing-masing switch menangani dua host. Jumlah host ditentukan dengan range kelas *IP Address* 10.10.10.0/24.



Gambar 4. Topologi Mesh

Dalam mininet topologi mesh disebut *custom topology*, dikarenakan fleksibiltas dari jumlah perangkat, baik jumlah *host*, *switch* maupun

controller. Pada sistem ini hanya menggunakan sebuah kontroler saja.

3.1. Analisa Kebutuhan Perangkat

Untuk membangun sistem jaringan yang berbasis *Software-Defined Network* membutuhkan beberapa komponen perangkat keras dan perangkat lunak yang akan dijelaskan sebagaimana berikut.

Spesifikasi perangkat keras berikut ini yang akan digunakan pada instalasi sistem menggunakan *virtual machine*.

Tabel 1. Kebutuhan Perangkat Keras

Perangkat Keras	Jumlah	Spesifikasi
Personal Computer (PC)	1 unit	▪ CPU AMD A10 (4cores) ~3.8GHz ▪ Memory 6 GB ▪ HDD 60 GB

Sedangkan kebutuhan perangkat lunak yang akan digunakan dalam membangun sistem jaringan *Software-Defined Network* adalah sebagai berikut.

Tabel 2. Kebutuhan Perangkat Keras

Perangkat Lunak	Keterangan
Linux Ubuntu 16.04 LTS	Sistem operasi
Mininet v2.2.2	Emulator jaringan (*Data Plane)*
Open Daylight (Nitrogen)	SDN kontroler (*Control Plane*)
Wireshark	Monitoring jaringan

4. PENGUJIAN SISTEM

Sistem yang telah dirancang akan diuji untuk mengetahui apakah sistem tersebut berjalan sesuai dengan yang diharapkan atau tidak. Gambaran pengujian yang dilakukan antara lain sebagai berikut :

1. Pengujian fungsionalitas perangkat lunak, pengujian dilakukan pada Mininet dan OpenDaylight, untuk mengetahui apakah instalasi Mininet dan OpenDaylight telah benar dapat bekerja dengan baik sebelum diimplementasikan rancangan topologi jaringan.
2. Pengujian konektivitas jaringan, pengujian konektivitas jaringan dilakukan untuk mengetahui apakah topologi jaringan yang telah dibangun dapat berfungsi sesuai dengan yang telah dirancang.
3. Pengujian parameter *Quality of Service* (QoS), pengujian *Quality of Service* dilakukan menggunakan bantuan Wireshark untuk mengetahui apakah hasil yang di dapat sesuai dengan keluaran yang diharapkan.

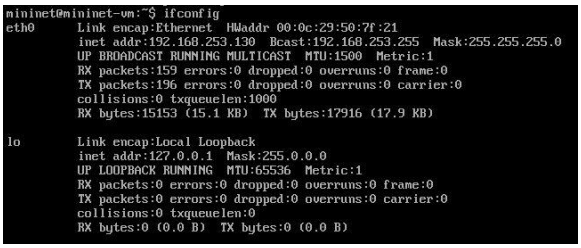
Setelah pengujian, dilakukan analisa untuk membuat kesimpulan dari hasil pengujian yang dilakukan.

4.1. Pengujian Fungsionalitas Perangkat Lunak

1. Melihat network interface dan ip address

Tahap awal ketika setelah selesai instalasi sistem adalah dengan memastikan ip address dan network interface yang digunakan. Karena pada sistem ini data plane dan kontroler terpisah (Control plane) maka masing-masing akan mendapat ip address yang berbeda. Berikut adalah perintah yang digunakan beserta hasil keluaranya :

$ ifconfig

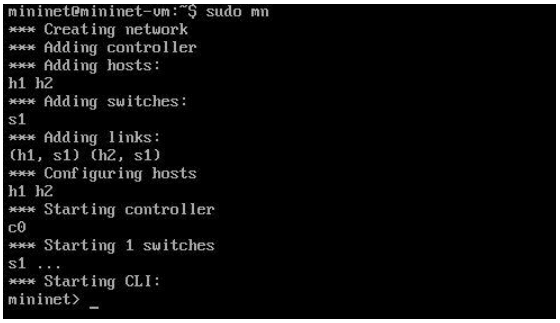


Gambar 5. Tampilan *network interface*

2. Pengujian fungsionalitas

Selanjutnya akan dilakukan pengujian apakah masing-masing sistem (control plane dan data plane) dapat berjalan sesuai dengan fungsinya. Pada Mininet dilakukan dengan menjalankan topologi default atau minimal yang telah disediakan dengan perintah :

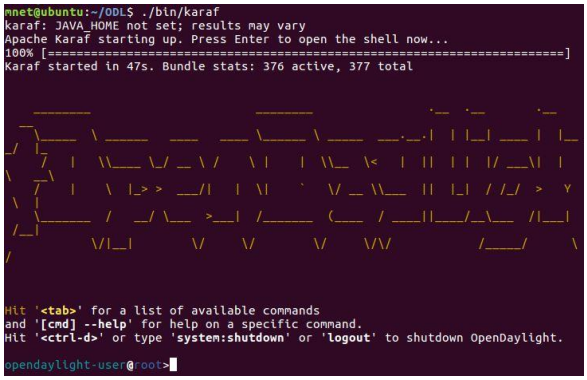
$ sudo mn



Gambar 6. Pengujian pada Mininet

Dan selanjutnya pengujian fungsionalitas pada kontroler yaitu dengan menjalankan kontroler dengan perintah sebagai berikut :

$ ./bin/karaf



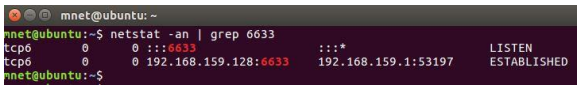
Gambar 7. Pengujian pada kontroler

4.2. Pengujian Konektivitas

Pengujian dilakukan dengan melakukan **pingall** pada topologi minimal yang telah diujikan pada tahap

sebelumnya di mininet. Selanjutnya untuk memeriksa koneksi konektivitas pada kontroler dapat dilakukan dengan perintah dibawah ini. Ketika telah terhubung, port yang berhasil terhubung dengan kontroler akan tampil "*Listen*".

$ netstat -an | grep 6633



Gambar 8. Konektivitas kontroler

4.3. Pengujian Parameter QoS

1. Pengujian Delay

Pengujian delay dilakukan dengan melakukan pengujian paket icmp menggunakan perintah **ping** pada jaringan SDN maupun jaringan konvensional,untuk melihat nilai delay yang dihasilkan.

Tabel 3. Pengujian Delay

Paket ke -	Delay (ms)			
	Topologi Star		Topologi Mesh	
	Kv	SDN	Kv	SDN
1	0.0	0.224	0	0.208
2	1.0	0.224	0	0.260
3	0.0	0.212	0	0.221
4	0.0	0.213	0	0.380
5	0.0	0.213	0	0.244
6	3.997	0.189	0	0.128
7	3.997	0.058	0	0.255
9	0.0	0.058	0	0.061
10	0.0	0.055	0	0.059

Pada topologi star didapat hasil delay kedua jenis arsitektur tidak jauh berbeda, dengan nilai delay tertinggi ada pada arsitektur konvensional yang hampir mencapai 4 ms, dan nilai delay terendah juga diperoleh dari jaringan konvensional dikarenakan berbedanya nilai digit angka penting.

Sedangkan pada topologi mesh nilai delay yang didapat hanya pada nilai SDN, dan pada topologi konvensional masing-masing host tidak dapat terhubung sehingga tidak dapat melakukan paket switching.

2. Pengujian Jitter

Pada pengujian Jitter dilakukan dengan perintah **iperf** pada pengujian paket UDP di mininet untuk menguji besar variasi delay yang didapat. Dengan *background traffic* 100 MB.

Tabel 4. Pengujian Jitter

Paket ke-	Jitter (ms)			
	Topologi Star		Topologi Mesh	
	Kv	SDN	Kv	SDN
1	0.933	0.059	0	0.009
2	1.002	0.081	0	0.085
3	1.024	0.077	0	0.099
4	0.971	0.053	0	0.091
5	1.002	0.050	0	0.086
6	0.915	0.082	0	0.090
7	0.912	0.081	0	0.084
8	0.933	0.084	0	0.090
9	0.924	0.081	0	0.027
10	0.930	0.097	0	0.332

Pada topologi star jitter dengan nilai terendah dimiliki oleh arsitektur SDN dengan nilai 0.050 ms dan tertinggi dimiliki oleh arsitektur konvensional dengan nilai 1.024 ms. Sehingga didapat nilai jitter pada jaringan SDN adalah hasil dua kali lipat nilai jitter pada jaringan konvensional. Sedangkan pada topologi mesh, jaringan SDN memiliki nilai jitter terendah sebesar 0.009 ms dan tertinggi sebesar 0.332 ms.

3. Pengujian Throughput

Pada pengujian Throughput dilakukan dengan perintah **iperf** pada pengujian paket TCP di mininet untuk menguji besar variasi delay yang didapat.

Tabel 5. Pengujian Throughput

Paket ke-	Throughput (Gb/s)			
	Topologi Star		Topologi Mesh	
	Kv	SDN	Kv	SDN
1	1.52	3.90	0	0.064
2	1.46	3.55	0	0.052
3	1.75	3.46	0	0.052
4	2.11	7.10	0	0.087
5	1.88	4.88	0	0.126
6	2.27	7.10	0	0.106
7	2.28	11.5	0	0.098
8	2.25	8.10	0	0.118
9	1.84	6.11	0	0.129
10	1.84	3.04	0	0.123

Pada topologi star hasil throughput terbesar diperoleh pada arsitektur SDN dengan nilai sebesar 8.10 Gb/s dan hasil throughput terkecil didapat oleh arsitektur konvensional dengan nilai sebesar 1.46 Gb/s.

Sedangkan pada topologi mesh di dapat nilai throughput terbesar pada jaringan SDN adalah 0.129 Gb/s atau sama dengan 129 Mbps. Dan terendah adalah 0.052 Gb/s atau sama dengan 52 Mbps.

4. Pengujian Packet Loss

Pada pengujian *Packet Loss* dilakukan dengan perintah **iperf** pada pengujian paket UDP di mininet untuk menguji besar variasi delay yang didapat. Dengan *background traffic* 100 MB.

Tabel 6. Pengujian *Packet Loss*

Paket ke-	Packet Loss (%)			
	Topologi Star		Topologi Mesh	
	Kv	SDN	Kv	SDN
1	8.6	0	0	0
2	8	4.3	0	0
3	9.6	4.5	0	0
4	11	13	0	0
5	9.5	8.5	0	0
6	30	18	0	0
7	8.1	3.9	0	0

8	11	24	0	0
9	9.4	3.8	0	0
10	7.4	38%	0	0

Pada topologi star hasil packet loss terbesar dan terkecil diperoleh pada arsitektur SDN dengan nilai sebesar max = 38% dan min=0%. Sedangkan pada topologi mesh di dapat nilai packet loss terkecil dimiliki oleh arsitektur jaringan SDN, dan konvensional memiliki nilai 0 karena tidak adanya proses switching didalamnya dikarenakan adanya looping oleh switch.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil analisa pengujian terhadap performa jaringan arsitektur SDN dan tradisional, maka dapat diambil beberapa poin kesimpulan sebagai berikut :

1. Arsitektur Software-Defined Network dapat berjalan dengan baik pada Mininet dan OpenDaylight dengan sistem operasi Ubuntu 16.04 LTS. Pada OpenDaylight yang membutuhkan dukungan java environment, hanya dapat di instal Java v.1.8.0.
2. Pada pengujian Delay, didapat pada jaringan SDN masih terbilang besar, dikarenakan adanya persetujuan access kepada kontroler tapi dapat dinyatakan lebih baik dilihat dari nilai rata-rata delay dan lebih stabil dibandingkan dengan arsitektur konvensional.
3. Pada pengujian Jitter, data pengujian pada SDN maupun jaringan konvensional tidak jauh berbeda. Tetapi tetap lebih baik pada arsitektur SDN karena nilai nya masih dibawah pada pengujian arsitektur konvensional.
4. Pada pengujian Throuhput, didapat pada jaringan SDN lebih baik, dengan nilai throughput yang dihasilkan dapat 2 sampai 4 kali lebih besar daripada jaringan konvensional.
5. Pengujian Packet Loss, nilai yang didapat pada arsitektur SDN masih lebih kecil disbanding pada arsitektur konvensional, meskipun pada pengujian nilai tertinggi throughputnya didapat oleh jaringan SDN.
6. Pada Topologi Mesh yang diterapkan pada jaringan konvensional tidak dapat dilakukan karena terhalang oleh factor looping switch. Sehingga membutuhkan topologi tambahan yaitu, Spanning Tree Protocol agar dapat terhubung.

5.2. Saran

Saran-saran yang diajukan agar menjadi masukan dalam kekurangan untuk pengembangan berikutnya adalah sebagai berikut :

1. Menambahkan beberapa macam scenario, termasuk jenis topologi yang digunakan.
2. Melakukan pengujian menggunakan controller yang berbeda.
3. Menganalisis arsitektur SDN yang telah diterapkan pada infrastruktur secara fisik.
4. Melakukan pengujian keamanan jaringan pada arsitektur SDN.

DAFTAR PUSTAKA

[1] Ummah, Izzatul, Desianto A. 2016. *Perancangan Simulasi Jaringan Virtual Berbasis Software-Defined Network.* Jurnal. Department of Computational Science, Telkom University. Bandung.

[2] Hidayat, Muhammad Hikam. 2017. *Analisis Kinerja dan Arsitektur Software-Defined Network Berbasis OpenDaylight Controller.* Jurnal. Universitas Gadjah Mada. Yogyakarta.

[3] Hyojoon, Kim, Nick Feamster. 2013. *Improving Network Management with Software Defined Networking*. Jurnal. Georgia Institute of Technology. Georgia.

[4] Software-Defined Network website. [Online]. Available : https://www.opennetworking.org/sdn-definition/.

[5] Anam, Khoerul, Adrian R. 2017. *Analisis Peforma Jaringan Software Defined Network Berdasarkan Penggunaan Cost Pada Protokol Routing OSPF*. Jurnal. Universitas Gadjah Mada. Yogyakarta.

[6] Mulyana, Eueng. 2015. *Buku Komunitas SDN-RG*. Bandung : Gitbook.

[7] Grguveric, Ivan, Zvonko K, Anthony P. 2015. *Simulation Analysis of Characteristic and Application of Software-Defined Network*. Jurnal. University of Zagreb, Zagreb, Croatia.

[8] Tolongan, Ramba Surya Triputra. 2015. *Perancangan dan Simulasi Arsitektur Software-Defined Networking Berbasis OpenFlow dan OpenDaylight Controller*. Skripsi. Amikom. Yogyakarta.

[9] Anggara, Sawung Murdha. 2015. *Pengujian Performa Controller Software-defined Network (SDN):POX dan Floodlight*. Jurnal Ilmiah. Sekolah Teknik Elektro Informatika. Institut Teknologi Bandung. Bandung.