UJI PERFORMA KONTROLER FLOODLIGHT DAN OPENDAYLIGHT SEBAGAI KOMPONEN UTAMA ARSITEKTUR SOFTWARE-DEFINED NETWORK

Rikie Kartadie[1)], Barka Satya[2)]

[1)]Teknik Informatika STMIK AMIKOM Yogyakarta
[2)]Manajemen Informasi STMIK AMIKOM Yogyakarta
Jl Ring road Utara, Condongcatur, Sleman, Yogyakarta 55281
email : r.kartadie@amikom.ac.id [1)] barka.satya@amikom.ac.id[2)]

Abstrak

Komponen utama dari Software- Defined Network adalah kontroler yang secara langsung melakukan kontrol terhadap datapath dari perangkat. Kontroler bertanggungjawab untuk memelihara semua aturan jaringan dan mendistribusikan petunjuk yang sesuai untuk perangkat jaringan. Sepuluh hal yang penting dalam pemilihan kontroler menurut Asthon,M. (2013) salah satunya adalah performa dari kontroler.

Karena begitu vitalnya fungsi dari kontroler dalam arsitektur jaringan, performa dari kontroler ini perlu diuji, pengujian dilakukan untuk mengetahui tingkat throughput dan latency dari kontroler. Sehingga diperoleh informasi yang tepat kemampuan dari kontroler yang akan digunakan.

Hasil percobaan yang dilakukan, kontroler Floodlight dan OpenDayLight memberikan nilai yang nilai latency yang baik pada jumlah switch dibawah 60 switch. Sedangkan untuk nilai throughput, Floodlight memberikan nilai sebesar 1500,38 flow/detik pada jumlah host 450, dan OpenDayLight memberikan nilai yang lebih rendah yaitu 354,05 flow/detik pada jumlah host 450. Nilai tertinggi latency yang diberikan oleh Floodlight adalah 2780.04 respon/detik pada jumlah 20 switch sedangkan OpenDayLight memberikan nilai latency tertinggi adalah 2160.06 respon/detik pada jumlah 20 switch.

***Kata kunci**: Kontroler, OpenDayLight, Floodlight, Software-Defined Network*

1. Pendahuluan

Komponen utama dari *Software-Defined Network* adalah kontroler yang secara langsung melakukan kontrol terhadap datapath dari perangkat. Kontroler bertanggung jawab untuk menentukan bagaimana menangani paket dan mengelola tabel *flow* dengan menambahkan dan menghapus isi tabel *flow* melalui *secure channel*. Kontroler pada dasarnya memusatkan kecerdasan jaringan, sedangkan jaringan mempertahankan *dataplane forwarding* yang didistribusikan melalui Switch OpenFlow. Untuk alasan ini kontroler menyediakan antarmuka untuk mengelola, mengendalikan dan Administrasi tabel *flow* yang Switch ini.[1]

Sepuluh hal yang penting dalam pemilihan kontroler menurut Asthon,M. (2013) salah satunya adalah performa dari kontroler. Salah satu fungsi utama dari kontroler SDN adalah untuk membentuk *flow*. Dengan demikian, dua dari metrik kinerja kunci yang terkait dengan kontoler SDN adalah waktu *setup flow* dan jumlah *flow* per detik. Metrik kinerja ini sangat mempengaruhi ketika pengontrol SDN tambahan harus dikerahkan. Misalnya, jika switch dalam arsitektur SDN memulai memberikan *flow* yang lebih besar daripada yang dapat didukung oleh kontroler SDN yang ada, sehingga lebih dari satu kontroler harus diimplementasikan.[2]

Rumusan dan tujuan

Karena begitu vitalnya fungsi dari kontroler dalam arsitektur jaringan. Performa dari kontroler dirasa perlu untuk dilakukan analisis untuk melihat seberapa baik performa dari kontroler. Pengujian dilakukan un- tuk mengetahui tingkat *throughput* dan *latency* dari kontroler, sehingga diperoleh informasi yang tepat kemampuan dari kontroler yang akan digunakan.

Tinjauan pustaka

Muntaner, GR. De Tejdana (2012) melakukan pengujian kontroler NOX, Maestro, Beacon, dan Trema. Dari keempat kontoler tersebut merupakan kontroler yang tidak menggunakan GUI dalam melakukan input table *flow* [1], sedangkan penelitian ini menguji kontroler yang mengunakan GUI dalam melakukan input table *flow*.

Turull,D. Dkk (2014) memfokuskan pada aplikasi virtualisasi jaringan, dan mengukur delay yang terjadi pada pesan ICMP, waktu transfer dari koneksi TCP, dan kerugian paket dalam lalu lintas UDP, berbagai penundaan antara switch dan kontroler [3], sedang penelitian ini memfokuskan pada *throuhput* dan *latency* dengan mensimulasi jumlah *flow* per detiknya dan mensimulasi jumlah switch yang terlibat.

Landasan teori

Kontroler memiliki dua perilaku: reaktif dan proaktif. Dalam **pendekatan reaktif**, paket pertama dari aliran yang diterima oleh switch memicu kontroler untuk menyisipkan *flow* di setiap jaringan switch OpenFlow. Pendekatan ini relatif paling efisien penggunaan memori tabel *flow* yang ada, tapi setiap *flow* baru menyebabkan waktu setup tambahan kecil. Dalam **pendekatan Proaktif**, kontroler melakukan pra-populasi tabel *flow* di setiap switch. Pendekatan ini tidak memiliki tambahan waktu setup aliran karena aturan *forwarding* didefinisikan terlebih dahulu. Jika switch kehilang- an koneksi dengan kontroler, tidak mengganggu lalu lintas. Namun, operasi jaringan memerlukan manajemen rumit, misalnya, membu-tuhkan agregat (wildcard) aturan untuk menutup semuarute.[4]

Protokol OpenFlow menggunakan protokol TCP port 6633. Protokol OpenFlow mendukung tiga jenis pesan 1) Pesan ***Controller-to-switch*** : Pesan ini dikirim hanya oleh kontroler untuk switch; melakukan fungsi konfigurasi switch, memodifikasi kemampuan switch, dan juga mengelola tabel *Flow*. 2) Pesan ***Symmetric***: Pesan ini dikirim di kedua arah untuk melaporkan masalah koneksi switch-controller. 3) Pesan ***Asyn-chronous***: Pesan ini dikirim oleh switch ke kontroler untuk mengumumkan perubahan dalam jaringan dan beralih kondisi. Semua paket yang diterima oleh switch dibandingkan terhadap tabel *Flow*. Jika paket apapun cocok masuk tabel *flow*, tindakan untuk entri yang dilakukan pada paket, misalnya, meneruskan paket ke port tertentu. Jika tidak ada pertandingan (tidak ada dalam tabel *flow*), paket akan diteruskan ke kontroler yang bertanggung jawab untuk menentukan bagaimana menangani paket tanpa entri *flow* yang *valid*.[5]

Kontroler *Floodlight*, kontroler *Software-Defined Network* terbuka yang merupakan kontroler OpenFlow kelas *enterprise*, *Apache-lisensi*, dan berbasis Java. Pengembangan dari kontroler ini didukung oleh komunitas pengembang termasuk sejumlah insinyur dari Big Switch Networks. Fitur dari *Floodlight* adalah :

• *Module loading system* yang membuatnya lebih sederhana dan mudah untuk di kembangkan.
• Mudah diatur dengan dipedensi, mendukung baik switch virtual maupun switch fisik.
• Dapat menangani jaringan campuran OpenFlow dan non-OpenFlow.
• Dirancang agar memiliki performa tinggi - adalah inti dari produk komersial dari *Big Switch Networks*.
• Mendukung *OpenStack* (link) orkestrasi *platform cloud.*

Kontroler *OpenDayLight*, Gopal, I., (2013) menya-takan bahwa *OpenDayLight* adalah sebuah proyek *Open Source software* dalam naungan Linux Foundation. Tujuannya melanjutkan penerapan dan inovasi *Software-Defined Networking* (SDN) melalui penciptaan *framework* yang umum dikalangan indus- tri [6].

OpenDayLight merupakan sebuah implementasi dari konsep *Software-Defined Network* (SDN) dan meman-faatkan beberapa *tools* berikut:
• *Maven*: *OpenDayLight* menggunakan Maven untuk lebih mudah membangun otomatisasi. Maven menggunakan pom.xml (Project Object Model) untuk script depedensi antara bundel dan juga untuk menjelaskan bundel apa untuk memuat dan dijalankan. • *OSGi*: Kerangka ini adalah *back-end* dari *OpenDayLight* karena memungkinkan secara dinamis pemuatan bundel dan file paket JAR, dan mengikat bundel bersama-sama untuk *xchanging* informasi.
• antarmuka JAVA: *Java interface* yang digunakan untuk aktifitas *listening*, spesifikasi, dan membentuk pola. Ini adalah aktifitas utama di mana bundel tertentu melaksanakan fungsi *call-back* untuk aktifitas dan juga untuk menunjukkan kondisi tertentu.
• *REST API* : Ini adalah *northbound API* seperti host tracker, *flow* programmer, static routing, dan sebagainya.[7]

Throughput secara umum merupakan tingkat capaian atau tingkat di mana sesuatu dapat diproses. Dalam hal ini troughput kontroler adalah besaran jumlah flow pada tiap detik yang dapat ditanganinya.

Latency penundaan waktu antara saat sesuatu dimulai dan saat ini atau menjadi terdeteksi [8] atau dapat pula diartikan sebagai interval waktu antara stimulasi dan respon atau, dari sudut pandang yang lebih umum, sebagai waktu tunda antara penyebab dan efek dari beberapa perubahan fisik dalam sistem yang diamati. Dalam hal ini *latency* dari kontroler adalah jumlah respon yang dapat diberikan oleh kontroler dalam tiap detiknya.

cbench **(***Collective Benchmark***)** didefiniskan sebagai sebuah *Benchmarking tool* yang dapat digunakan untuk mengevaluasi paramater *latency* dan *through- put* dari kontroler. Tool *cbench* ini adalah bagian dari OFlops. [9]

Alur penelitian

Langkah-langkah penelitian yang dikerjakan pada pe-nelitian ini dapat dilihat pada alur penelitian seperti gambar 1 dan dapat dijelaskan sebagi berikut:

1. Instalasi kontroler yang akan diujikan.
2. Uji kesiapan kontroler *Floodlight* dan *OpenDay-Light*, menguji keberhasilan langkah pertama dengan emulator mininet yang diinstall dan dijalankan pada *VirtualBox,* dengan skenario topologi single, 5 host. Bila kontroler tidak dapat mengenal switch pada mininet, maka proses instalasi akan di ulang.

Gambar 1. Alur penelitian

3. Melakukan pengujian *latency Floodlight dan OpenDayLight* dengan jumlah switch mulai hingga 450 switch dan jumlah host setiap switch adalah 5 host.

4. Melakukan pengujian *throughput Floodlight dan OpenDayLight* dengan jumlah switch optimal dari pengujian *latency* dan melakuakan variasi jumlah host hingga 450 host.

5. Dokumentasi hasil.

6. Menarik kesimpulan dan saran.

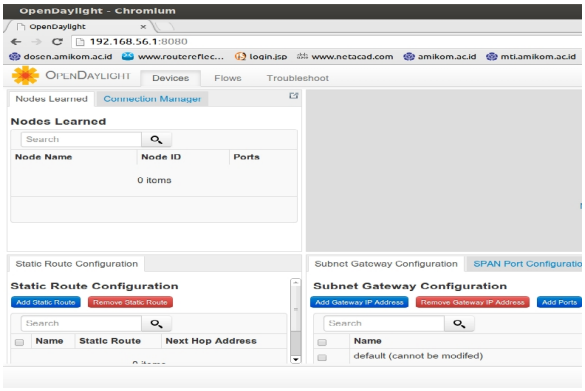
Alur penelitian dapat dilihat pada gambar 1 diatas.

2. Pembahasan

Instalasi kontroler *OpenDayLight*

Dilakukan installasi *OpenDayLight* "Hydrogen" edisi *base* dengan langkah seperti dibawah ini;

1. Instalasi kontroler dengan perintah-perintah pada terminal seperti terlihat dibawah ini,
   apt-get update
   apt-get install maven git openjdk-7-jre openjdk-7-jdk
   git clone http://git.opendaylight.org/gerrit/p/controller.git
   cd controller/opendaylight/distribution/opendaylight/
   mvn clean install
   cd target/distribution.opendaylight-0.1.0-SNAPSHOT-osgipackage/opendaylight
   JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-i386
2. Untuk menjalankan kontroler *OpenDayLight*, dapat dilakukan dengan perintah;
   ~/controller/opendaylight/target$ sudo ./run.sh
2. Setelah semua dijalankan, maka kontroler *OpenDayLight* GUI dapat diakses pada
3. *http://localhost:8080/index.html,* dengan admin sebagai user dan password.



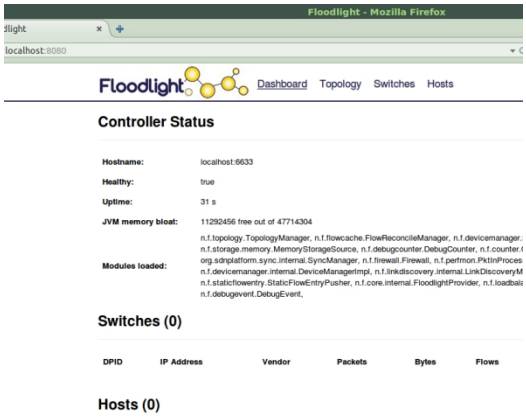
Gambar 2. GUI OpenDayLight

Sedangkan tampilan GUI dari *OpenDayLight* dapat dilihat pada gambar 2.

Instalasi kontroler *Floodlight*

Instalasi *Floodlight* lebih sederhana dibandingkan dengan kontroler *OpenDayLight*, langkah-langkah instalasi adalah sebagai berikut;

1. Insatalasi dipendensi yang dibutuhkan oleh Floodlight dengan perintah *~$sudo apt-get install build-essential default-jdk ant python-dev eclipse*
2. Kemudian melakukan kloning file *Floodlight* dari web github dengan perintan *~$git clone git://github.com/floodlight/floodlight.git*
3. Setelah semua terinstal, *Floodlight* dapat dijalankan pada folder *Floodlight* dengan perintah;
   ~$cd floodlight/target
   ~/floodlight/target$sudo java -jar floodlight.jar



Gambar 3. GUI *Floodlight*

4. Setelah semua dijalankan, maka kontroler *Floodlight* GUI dapat diakses pada *http://localhost:8080/ui/index.html*

Sedangkan tampilan GUI dari *Floodlight* dapat dilihat pada gambar 3.

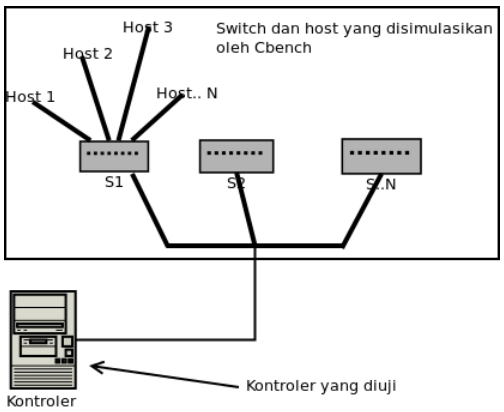
Kedua kontroler yang diinstal telah dapat dikenali emulator mininet.

Skenario Uji

Pengujian dilakukan dengan skernario meletakkan kontroler dan *cbench* diletakkan pada *virtualbox* dalam 1 host/PC dan dijalankan pada sumber daya yang sa- ma.

Skenario seperti terlihat pada gambar 4.

Spesifikasi PC/Laptop yang digunakan dalam uji ini adalah sebagai berikut;
- *CPU : Intel® Pentium(R) CPU 987 @ 1.50GHz × 2*
- *RAM : SODIMM DDR3 2Gb 1333MHz*
- *Operation System : Linux UBUNTU 12.04 LTS kernel 3.13.0-36-generic*.



Gambar 4. Skenario uji

Pada pengujian digunakan seluruh *threats CPU* yang ada, tidak dilakukan perubahan/variasi pada *threats.*

Uji *latency*
Pengujian *latency* dilakukan dengan perintah sebagai berikut:
~/oflops/cbench$ cbench -c [ip-kontroler] -p 6633 -l 5 -m 5000 -D 5 -M 5 -s [jumlah switch]

Uji *throughput*
Pengujian *throughput* dilakukan dengan perintah sebagai berikut:
~/oflops/cbench$ cbench -c [ip-kontroler] -p 6633 -l 5 -m 5000 -D 5 -M [jumalh MAC] -s [jumlah switch] -t

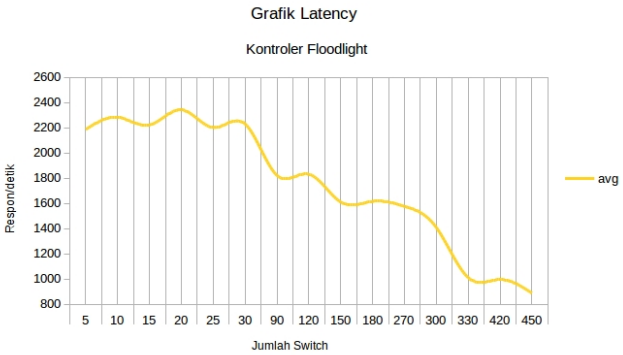
Pada uji *trhoughtput* dan *latency* diberikan beban dengan panjang pengujian 5000 ms dan diberikan jeda mulai pengujian setelah *features_reply* diterima selama 5 ms.

Hasil uji *latency* pada kontroler *Floodlight*, sample data dapat dilihat pada tabel 1 dimana pengujian dilakukan pada jumlah switch 5 hingga 450 switch.

Tabel 1. Sample data *latency* pada *Floodlight*

n Switch	20	90	120	330	450
Min	2065.22	1615.83	1661.75	246.97	195.25
Max	2780.04	2028.07	2003.42	1505.84	1239.06
Avg	2342.25	1821.06	1831.35	1010.41	887.08

Hasil nilai rata-rata (avg) pengujian *latency* pada *Floodlight*, dapat juga dilihat pada gambar 5 berupa grafik.



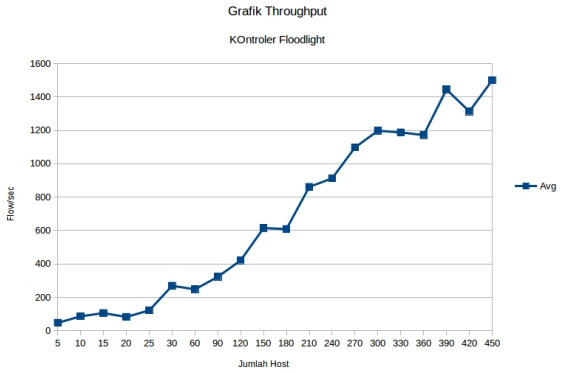
Gambar 5. Grafik *Latency Floodlight*

Dari data yang terlihat tabel 1 dan gambar 5 diatas, bila dilihat dari data rata-rata (avg) respon/detik vs switch , dapat dilihat bahwa semakin banyak switch yang terlibat, semakin rendah pula respon yang dapat diberikan oleh kontroler dalam mengangani *flow*. Keterlambatan kontroler *Floodlight* dimulai pada switch berjumlah diatas 60, sehingga jumlah switch optimum untuk kontroler *Floodlight* adalah antara 1 switch hingga 60 switch. Pada pengujian *throughput* digunakan jumlah switch sebanyak 20 switch.

Hasil uji *throughput* pada kontroler *Floodlight*
Sample data dapat dilihat pada tabel 2 dimana pengujian dilakukan pada range 5 – 450 host dengan jumlah switch 20 switch.

Tabel 2. Sample data *throughput*pada *Floodlight*

n Host	20	90	120	330	450
Avg (Flow/sec)	82.6216	323.1068	421.5658	1187.7874	1500.3798



Gambar 6. Grafik *throughputFloodlight*

Bila dilihat dari data rata-rata (avg) *flow*/detik vs jumlah host yang ditampilakan pada tabel 2 dan gambar 6 diatas, kontroler *Floodlight* dapat memberikan kemampuan yang baik karena dengan meningkatnya jumlah host yang ada, maka semakin tinggi pula jumlah flow/detik yang dapat di layani oleh kontroler.

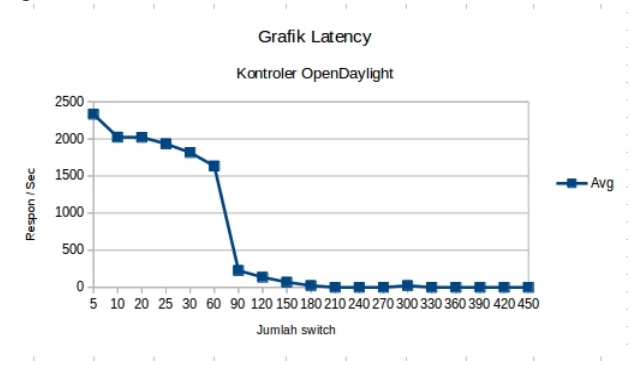
Hasil uji *latency* pada kontroler *OpenDayLight*, sample data dapat dilihat pada tabel 3 dimana pengujian dilakukan pada jumlah switch 5 hingga 450

switch. Skenario dan langkah pengujian sama dengan skenario dan langkah pengujian pada kontroler *Floodlight*.

Tabel 3. Sample data *latency* pada *OpenDayLight*

N-Switch	*20*	*60*	*90*	*120*	*270*	*300*	*450*
Min	1790.2	891.6	66.2	48.6	0	23.2	0
Max	2160.6	1968.2	385.4	264.4	0	23.2	0
Avg	2021.74	1634.4	226.55	136.5	0	23.2	0

Hasil nilai rata-rata (avg) pengujian *latency* pada *OpenDayLight*, dapat juga dilihat pada gambar 7 berupa grafik , dibawah ini.



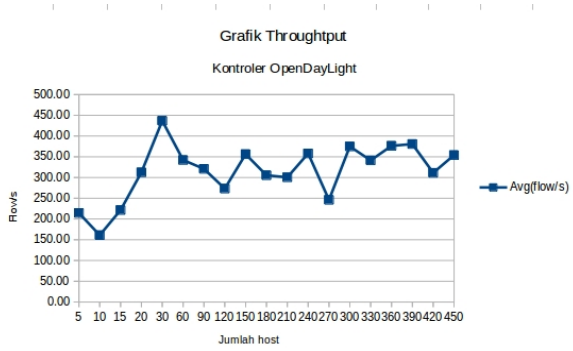
Gambar 7. Grafik *Latency OpenDyaLight*

Dari data yang terlihat pada tabel 3 dan gambar 7 diatas, bila dilihat dari data rata-rata (avg) respon/detik vs switch , dapat dilihat bahwa semakin banyak switch yang terlibat, semakin rendah pula respon yang dapat diberikan oleh kontroler dalam mengangani *flow*. Keterlambatan kontroler *OpenDayLight* dimulai pada switch berjumlah diatas 90, namun dapat dilihat bahwa kontroler *OpenDayLight* hanya mampu menangani hingga 90 switch. Pada switch diatas 90 kontroler tidak dapat lagi merespon *request* dari switch.

Hasil uji *throughput* pada kontroler *OpenDay- Light*
Sample data dapat dilihat pada tabel 4 dimana pengujian dilakukan pada range 5 – 450 host dengan jumlah switch 20 switch.

Tabel 4. Sample data *throughput* pada *OpenDayLight*

n(Host)	20	90	120	330	450
Avg(flow/s)	312.58	320.8582	273.3472	341.331	354.0466



Gambar 8. Grafik *throughput OpenDyaLight*

Bila dilihat dari data rata-rata (avg) *flow*/detik vs jumlah host yang ditampilakan pada tabel 4 dan gambar 8 diatas, kontroler *Floodlight* dapat memberikan kemampuan yang baik, pada penambahan jumlah host kemampuan *flow*/detik dari kontroler *OpenDayLight* menunjukkan angka yang tidak me- ningkat pada tiap penambhan jumlah *host*. Namun pada kontroler ini, menunjukkan angka yang relatif kurang stabil.

3. Kesimpulan

Dari hasil pengujian yang telah dilakukan diperoleh hasil bahwa kontroler *Floodlight* dan *OpenDayLight* memberikan nilai *latency* yang baik pada jumlah switch dibawah 60 switch. Nilai tertinggi *latency* yang diberikan oleh *Floodlight* adalah 2780.04 respon/detik pada jumlah 20 switch sedangkan *OpenDayLight* memberikan nilai *latency* tertinggi adalah 2160.06 respon/detik pada jumlah 20 switch. Untuk nilai *throughtput*, *Floodlight* memberikan nilai sebesar 1500,38 *flow*/detik pada jumlah host 450, dan *OpenDayLight* memberikan nilai yang lebih rendah yaitu 354,05 *flow*/detik pada jumlah host 450. Dapat disimpulkan bahwa kontroler *Floodlight* lebih baik performa-nya dibandingkan *OpenDayLight* untuk jumlah switch yang besar, dan *Floodlight* mampu memberikan *throughput* yang lebih besar diban- dingkan *OpenDayLight.*

Untuk penelitian selanjutnya dapat dilakukan untuk skenario pengujian yang lain terhadap kontroler, jumlah host dan switch yang berbeda.

Daftar Pustaka

[1] Muntaner, GR. De Tejdana., Februari 2013, "Evaluation of OpenFlow Controllers, 2012", http://www.valleytalk.org/wp-content/uploads/2013/02/Evaluation_Of_OF_Controllers.pdf
[2] Asthon,M., 10 oktober 2014, "Ten Things to Look for in an SDN Controller", 2013, https://www.necam.com/docs/?id=23865bd4-f10a-49f7-b6be-a17c61ad6fff
[3] Turull D.,; Hidell M.; Sj¨din P., 2014, "Performance evaluation of OpenFlow controllers for network virtualization", KTH Royal Institute of Technology, School of ICT Kista, Sweden, High Performance Switching and Routing (HPSR), IEEE 15th International Conference, 10.1109/HPSR.2014.6900881
[4] Marcial P. Fernandez, 2013,"Evaluating OpenFlow Controller Paradigms", ICN 2013 : The Twelfth International Conference on Networks, ISBN: 978-1-61208-245-5

[5] Heller, B., november 2012, "Openflow switch specification, version 1.0.0", 2009, www. openflowswitch.org/documents/openflow-spec-v1.0.0.pdf
[6] Gopal I., 12 September 2013, "Introduction to OpenDayLight, Linux Foundation colaboration Project", 16 April 2013, http://www.opennetsummit.org/pdf/2013/presentations/inder_gopal.pdf
[7] Anonim, 10 Oktober 2014, "OpenDaylight User Guide", Linux Foundation colaboration Project, 29 September 2014, https://opendaylight.org/sites/opendaylight/files/bk-developers-guide-20141002.pdf
[8] Anonim, 8 Januari 2013, "What is Network Latency and Why Does It Matter?", 03b Networks, http://www.o3bnetworks.com/media/40980/white paper_latency matters.pdf
[9] McKeown,N., et al., 1 Mei 2013, "OpenFlow: Enabling Innovation in Campus Networks" , Stanford University, 2008

Biodata Penulis

Rikie Kartadie, S.T., **M.Kom.,** Mendapatkan gelar Sarjana Teknik (S.T.) pada tahun 2001 dari Universitas Pembangunan Nasional "Veteran" Yogyakarta, Mendapatkan gelar Master Komputer (M.Kom.) pada Mei 2014 Konsentrasi Sistem Informasi dari STMIK AMIKOM Yogyakarta, dan menjadi pengajar pada program studi S1 Teknik Informatika STMIK Amikom Yogyakarta.

Barka Satya, M.Kom., Mendapatkan gelar Sarjana Komputer (S.Kom) pada tahun 2005 dari STMIK AMIKOM Yogyakarta, Mendapatkan gelar Master Komputer (M.Kom.) pada September 2012 dari STMIK AMIKOM Yogyakarta, dan menjadi pengajar program studi Manajemen Informatika STMIK Amikom Yogyakarta..