JURNAL INFOTEL
Informatika - Telekomunikasi - Elektronika

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Analisis Simulasi Penerapan Algoritma OSPF Menggunakan *RouteFlow* pada Jaringan *Software Defined Network* (SDN)

Ridha Muldina Negara[1], Rohmat Tulloh[2]
[1]Fakultas Teknik Elektro, Universitas Telkom
[2]Fakultas Ilmu Terapan, Universitas Telkom
[1,2]Jalan Telekomunikasi Terusan Buah Batu Bandung 40257
Email korespondensi : ridhanegara@telkomuniversity.ac.id

Dikirim 24 Januari 2017, Direvisi 08 Februari 2017, Diterima 12 Februari 2017

Abstrak - Pada jaringan konvensional, konfigurasi *protocol routing* sangat tidak fleksibel, tidak efisien dan konfigurasi dilakukan pada tiap perangkat. Hal ini tentu saja tidak dapat memenuhi tuntutan operasional saat ini yang rata-rata memiliki jaringan yang besar dan perangkat jaringan yang memiliki spesifikasi berbeda. *Software Defined Network* (SDN) muncul sebagai harapan untuk permasalahan kompleksitas jaringan konvensional. Paradigma baru SDN melakukan pemisahan antara *control plane* dan *forwarding plane*. *RouteFlow* merupakan salah satu komponen berbasis *software* yang dapat mengaplikasikan *protocol routing* konvensional pada jaringan SDN. *Open Shortest Path First* (OSPF) merupakan sebuah protokol *routing* konvensional yang memiliki kemampuan untuk mendeteksi perubahan topologi jaringan dengan cepat dalam sebuah jaringan yang besar. Protokol *routing* OSPF diterapkan pada teknologi SDN menggunakan *RouteFlow* dengan tujuan untuk mempermudah dalam mengontrol jaringan dengan sistem terpusat. *Time convergence* dan parameter *Quality of Service* (*Throughput*, *Delay*, *Jitter* dan *Packet Loss*) diukur dengan skenario pemutusan *link*, penambahan jumlah *switch* dan *background traffic*. Hasil pengukuran *time convergence* menunjukan bahwa penambahan jumlah *switch* mempengaruhi pertambahan waktu konvergensi, sedangkan untuk parameter *Quality of Service* (QoS) pada peningkatan topologi *switch* didapatkan hasil yang masih sesuai dengan standar ITU-T G.1010 namun apabila ditambahkan *background traffic* yang memenuhi 50% *bandwidth* jaringan maka QoS memburuk.

Kata kunci - Software Defined Network, OSPF, RouteFlow, Mininet, QoS

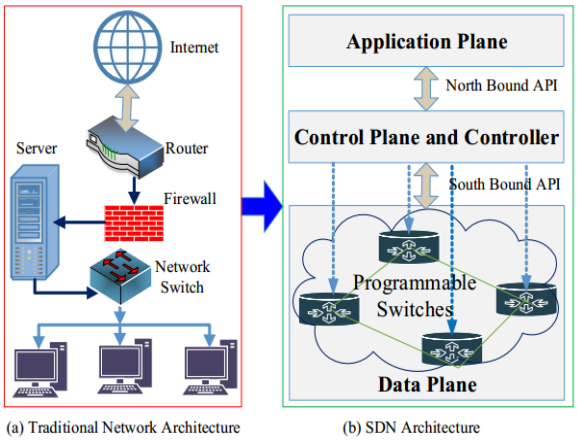
Abstract - In a conventional network, routing protocol configuration is very inflexible, inefficient and configuration is performed on each device. This of course cannot meet today operational demands which on average have a large network and network devices of different specifications. Software Defined Network emerged as the hope for the conventional network complexity issues. The new paradigm SDN separation between the control plane and forwarding plane. RouteFlow is one component-based software that can apply a conventional routing protocol on the network Software Defined Network. Open Shortest Path First (OSPF) is a conventional routing protocol that has the ability to detect changes in the network topology quickly in a large network. OSPF routing protocol is applied on Software Defined Network technology uses RouteFlow with the aim to simplify the control network with a centralized system. Time Convergence and the parameters of Quality of Service (Throughput, Delay, Jitter and Packet Loss) are measured with link termination scenarios, increasing the number of switches and background traffic. Time convergence measurement results showed that the increase in the number of switches affect time overconvergence, while the parameters of Quality of Service (QoS) on the increment of switch topology complexity, the results obtained is in accordance with ITU-T G.1010 standard, but when added to background traffic for 50 % bandwidth QoS network then deteriorate.

Keywords - Software Defined Network, OSPF, RouteFlow, Mininet, QoS

I. PENDAHULUAN

Infrastruktur Teknologi Informasi dan Komunikasi (TIK) yang terus berkembang dengan peningkatan sejumlah perangkat dan aplikasi *Internet-of-Things* (IOT) dan *cyber physical systems*. *Software-Defined Networking* (SDN) dianggap sebagai teknologi yang mampu mengelola seluruh jaringan secara efisien dan mengubah arsitektur jaringan yang kompleks menjadi sederhana dan mudah dikelola. Studi terbaru menunjukkan bahwa jaringan tradisional tidak mampu memuaskan tuntutan pertumbuhan karena semua komponen dalam jaringan yang terintegrasi secara vertikal membentuk struktur yang kompleks yang sulit untuk dikelola. Jaringan tradisional mampu mendukung kebijakan vendor secara spesifik dan tidak menawarkan fleksibilitas untuk lingkungan jaringan yang dinamis [1].

Gambar 1 menggambarkan perbandingan transformasi tingkat tinggi arsitektur jaringan tradisional dan arsitektur SDN. SDN dianggap sebagai paradigma jaringan generasi berikutnya dengan hardware independen di mana jaringan perangkat dari vendor manapun bisa dikontrol melalui SDN.



Gambar 1. Perbandingan Arsitektur Jaringan Sederhana dan Arsitektur Sederhana *Software Defined Network* (SDN)

SDN memisahkan *application plane*, *data plane* dan *control plane*, yang memiliki dua komponen utama yaitu *controller* dan *switch*. *Controller* SDN bertanggung jawab atas pengelolaan seluruh jaringan sedangkan *switch* jaringan bertanggung jawab untuk operasi berdasarkan petunjuk yang disebarkan melalui *controller* SDN. Tidak seperti jaringan tradisional, dimana seluruh sistem perlu dikonfigurasi ulang untuk meng-*upgrade* sistem, pada SDN hanya perangkat lunak (*software*) saja yang perlu diperbarui, sehingga lebih nyaman untuk *upgrade* dan mengurangi biaya keseluruhan [2].

Salah satu *virtual environment* yang dapat bekerja sebagai *controller* khusus yang menyediakan fungsi protokol *routing* adalah *RouteFlow*. Pada jaringan konvensional, protokol *routing* menjadi bagian penting dalam pengaturan jaringan, dimana proses konfigurasinya harus dilakukan pada tiap *device* jaringan, hal tersebut membuat jaringan konvensional tidak efisien dan tidak fleksibel. *Open Shortest Path First* (OSPF) merupakan sebuah *protoko routing* yang memiliki kemampuan untuk mendeteksi perubahan topologi jaringan dengan cepat dalam sebuah jaringan yang besar.

Untuk itu, melihat dari permasalahan di atas, dalam penelitian ini dilakukan analisis dan simulasi protokol *routing* OSPF pada teknologi SDN dengan *RouteFlow* sehingga mempermudah dalam pengontrolan jaringan dengan sistem terpusat, untuk melihat apakah peroutingan OSPF dapat berjalan dengan baik di jaringan SDN.

Penelitian terkait QoS pada SDN pertama kali dilakukan oleh Heller, dkk [3], dimana pertimbangan jarak *controller* ke *switch* bertujuan untuk mengurangi *delay end-to-end* dan untuk menentukan jumlah *controller* yang optimal. Pada penelitian tersebut belum mempertimbangkan beban kerja *controller*, sehingga solusi yang dihasilkan tidak adaptif terhadap trafik yang dinamis. Selain itu Zhang, Hailong, dkk [4] juga melakukan penelitian terkait algoritma OSPF menggunakan *Floodlight controller*, dengan mengukur parameter *forwarding delay* dan *time convergence* namun tidak melihat pengaruhnya terhadap QoS dan penambahan beban jaringan seperti peningkatan *background traffic*.

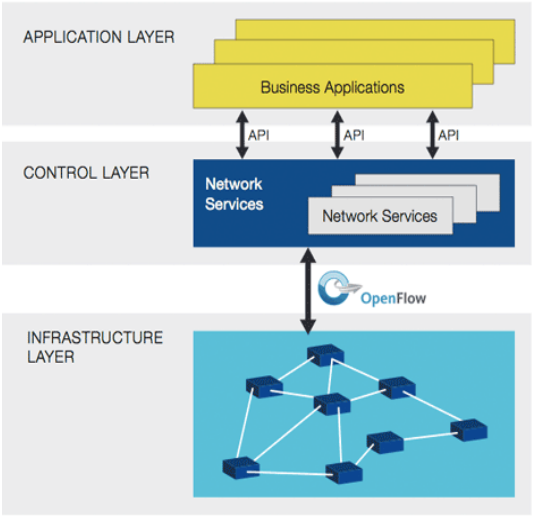
Kemudian Zeng, dkk [5], telah melakukan penerapan OSPF pada jaringan SDN untuk mengetahui nilai *failover time*. Penelitian tersebut juga menggunakan *RouteFlow* tetapi hanya mencoba pada skenario delapan buah *switch* kerena fokus terhadap pengaruh multiple *link failure* dan tidak mencoba melihat pengaruh dari pertambahan jumlah *switch*, dan lain-lain. Berdasarkan penelitian-penelitian di atas, penelitian ini berfokus pada analisis performansi pengaruh pertambahan besar topologi dan *background traffic* terhadap jaringan SDN menggunakan *routing* OSPF berdasarkan parameter *network convergence time* dan *quality of service* (QoS).

II. METODE PENELITIAN

A. *Software Defined Network (SDN)*

Menurut *Open Networking Foundation* (ONF), SDN didefinisikan sebagai arsitektur jaringan yang memisahkan kontrol dan fungsi *forwarding* untuk mengaktifkan kontrol jaringan menjadi terprogram langsung dan infrastruktur *underlying* untuk aplikasi dan layanan jaringan [6]. Gambar 2 menunjukkan pandangan logis dari arsitektur SDN.

Arsitektur SDN membagi jaringan menjadi 3 layer yaitu *application layer*, *control layer* dan *insfrastructure*/data *layer*. *Application layer* merupakan *interface* terhadap seorang admin atau peneliti dalam mengelola atau mengembangkan jaringan SDN. *Control plane* berisi suatu *controller* bersifat terpusat dan *based on software*. *Subordinate hardware* dikontrol sepenuhnya oleh *controlling plane* atau *controller* dalam melakukan *forwarding decission*. Semua *subordinate* terhubung ke *controller* [7].



Gambar 2. *Logical View* dari SDN Arsitektur

Seperti ditunjukkan pada Gambar 2, sebagian besar layanan jaringan (yaitu, protokol dan kontrol) terpusat di perangkat lunak kontrol SDN. SDN *controller* mengelola semua perangkat jaringan di lapisan infrastruktur dasar, yang ~~mana~~ merupakan *single logical switch virtual*. Operator jaringan dapat mengontrol jaringan melalui *interface* standar (misalnya, *OpenFlow*) secara independen dari vendor perangkat jaringan.

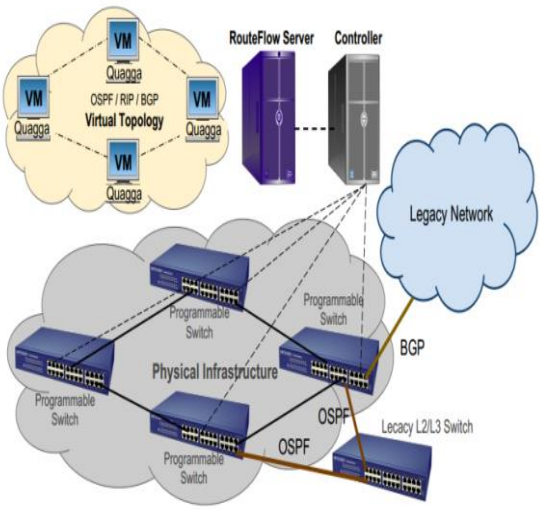
Perangkat - perangkat jaringan hanya dilengkapi dengan fungsi data *forwarding* yang dikendalikan oleh pengontrol SDN. Hal ini membuat desain dan operasi jaringan menjadi sederhana dan lebih efisien. API antara lapisan aplikasi dan lapisan kontrol dapat memberikan virtualisasi lingkungan jaringan dan sarana untuk menerapkan berbagai kebijakan mengenai *routing*, kontrol akses, rekayasa *traffic*, manajemen daya, dan lain-lain.

B. Open Shortest Path First (OSPF)

Routing OSPF telah banyak digunakan sebagai *Interior Gateway Protocol* berbasis *link-state* (IGP) pada jaringan IP. Jaringan. OSPF mengumpulkan informasi *link state* dari *router* yang ada dan membangun sebuah grafik topologi dari jaringan. Untuk rute paket, OSPF menghitung *shortest path tree* untuk setiap rute menggunakan metode yang didasarkan pada Algoritma Dijkstra. Untuk menentukan jalur terpendek, OSPF membutuhkan pemberian bobot setiap *link* di jaringan. *Link* bobot didistribusikan sebagai *link state* [8]. Domain untuk OSPF terletak dalam satu *autonomous system* (AS)[9].

C. RouteFlow

RouteFlow adalah salah satu *framework* untuk SDN, yang sedang dikembangkan sebagai proyek *open source* [10]. Pendekatan ini berjalan pada IP *routing* protokol (misalnya, BGP dan OSPF) di server *RouteFlow* terpusat dan menghasilkan *forwarding information base* (FIB) sesuai dengan protocol *routing* yang dikonfigurasi. *Server* mengumpulkan IP dan ARP tabel yang harus diterjemahkan ke dalam aturan *OpenFlow* yang akhirnya dipasang di *switch* yang diprogram terkait dalam jaringan [11]. Gambar 3 menunjukkan arsitektur pendekatan *RouteFlow* [12]. Sebuah server *RouteFlow* dihasilkan dan beroperasi pada mesin virtual (VM) di *control plane* SDN. Sebuah VM server *RouteFlow* dipetakan ke *switch OpenFlow* tertentu pada bidang *data forwarding*. VMS membangun topologi virtual dan menjalankan protokol *routing* terbuka, Quagga [13][14]. *RouteFlow* server terus memantau status tabel *routing* di VMS. Jika perubahan ditemukan dari tabel *routing*, server *RouteFlow* membuat entri aliran yang sesuai dan memberikan ke *OpenFlow Switch* terkait dengan segera.

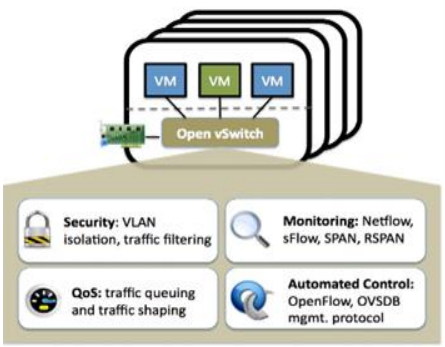


Gambar 3. Arsitektur *RouteFlow*

Arsitektur *RouteFlow* menggambarkan secara spesifik dan efisien untuk menampung IP *routing* protokol pada bidang kontrol SDN.

D. Open vSwitch

Open vSwitch adalah software multilayer yang didesain untuk digunakan sebagai *virtual switch* dalam lingkungan *virtual server*. *Open vSwitch* diinisiasi oleh perusahaan Nicira, sebuah perusahaan yang terfokus pada bidang virtualisasi jaringan. *Open vSwitch* mendukung standar manajemen *interface* (seperti sFlow, Netflow, RSOAN, CLI). *Open vSwitch* berfungsi sebagai *virtual switch* dalam lingkungan VM yang dirancang untuk mendukung distribusi di beberapa *server* fisik *(physical servers*).

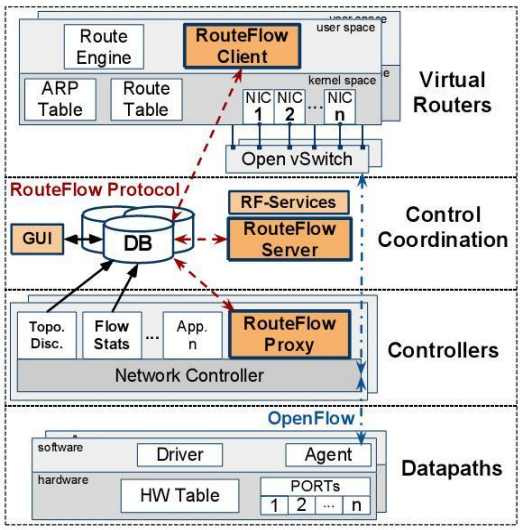


Gambar 4. Open v*Switch*

Open vSwitch mendukung beberapa teknologi virtualisasi yang berbasis Linux seperti Xen/Xenserver, KVM, dan *VirtualBox*. Dalam *virtual switch*, diklasifikasikan menjadi dua jalur, *fast path* dan *slow path*. Pada *fast path* terjadi pemrosesan paket-paket data seperti: *packet forwarding, trafficking,* dan enkapsulasi paket VLAN. Sedangkan jalur *slow path* difokuskan pada pengelolaan dengan *interface external* [15].

E. Algoritma Sistem

Terbentuk atas penggabungan proyek *OpenFlow* dengan *routing engine* Quagga. Bertujuan untuk menyediakan layanan *routing* virtual IP pada perangkat keras *OpenFlow-enabled* mengikuti paradigma SDN.



Gambar 5. Algoritma Sistem

Terdiri atas.

a) *RF Client* : *daemon* pada VM yang bertugas mendeteksi perubahan informasi *routing* dan menginformasikannya ke RF *Server*.
b) *RF Server* : Aplikasi *standalone* yang memegang kendali pusat kontrol jaringan. RF *Server* mengatur VM yang berjalan pada RF Client dan *logic process.*
c) *RF Proxy* : *Controller* yang bertugas meneruskan kebijakan protokol (misal: *update rute*, konfigurasi *data path*) dari RF *Server* ke *data plane*.

F. Perangkat Simulasi

Dalam menjalankan simulasi digunakan perangkat keras berupa sebuah laptop dengan spesifikasi sebagai berikut.

Tabel 1. Spesifikasi Perangkat

Spesifikasi	Laptop
Processor	Intel® Core™ i7-4720HQ Processor 2.60 Ghz
RAM	16 GB
Operating System	Windows 10, 64 bit
Fungsi	*Control plane* dan *data plane*

Selain perangkat keras, perangkat lunak yang dibutuhkan dalam simulasi ini yaitu.

a) Dua *Virtual Machine* yang digunakan pada laptop sebagai *control plane* dan *data plane.*
b) *Distributed Internet Traffic Generator* (D-ITG), digunakan untuk membangkitkan trafik dan
c) *Wireshark*, digunakan untuk meng-*capture* trafik saat pengambilan data *traffic overhead*.

Tabel 2. Spesifikasi *Virtual Machine*

Spesifikasi	VM
RAM	2 GB
Operating System	Ubuntu 12.04 LTS 64 bit
Fungsi	*Control plane : RouteFlow* dan *data plane :* Mininet

G. Perancangan Skenario

Untuk melakukan pengujian kinerja penjaluran diperlakukan beberapa skenario analisis data sehingga memungkinkan hasil yang valid untuk pertanggungjawaban. Berikut ini adalah beberapa skenario dalam penelitian ini.

a) Skenario 1 menggambarkan kondisi jaringan normal dengan bandwidth 100 Mbps.
b) Skenario 2 menggambarkan kondisi jaringan telah terjadi kepadatan bandwidth background traffic dengan range 0 Mbps – 125 Mbps.
c) Skenario 3 menggambarkan kondisi dimana jaringan normal namun terdapat link terputus pada jaringan, dengan bandwidth default range 0 Mbps – 125 Mbps. Diasumsikan link yang putus adalah antara *switch*1 dan *switch*4 (*Time Convergence*)

H. Parameter Pengujian

Parameter yang diukur adalah *time convergence* dan *Quality of Service (QoS)* menggunakan standar nilai performansi sesuai ITU.T G1010.

Tabel 3. Referensi Standarisasi QoS ITU.T G.1010

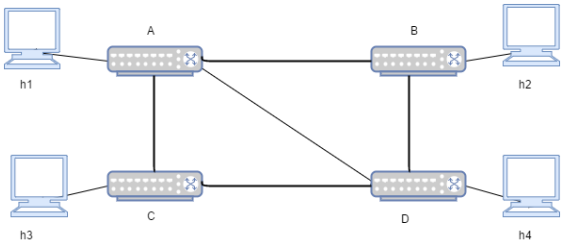
Parameter Performansi	Data	VoIP	Video
One Way Delay (Latency)	Preffered < 15 s; Acceptable < 60s Nb : Amount of Data 10kB-10MB	Preffered < 150 ms; Acceptable < 400s	< 10 s
Jitter	NA	< 1 ms	NA
Throughput	NA	4 - 64 kbps	16 - 384 kbps
Packet Loss	0%	< 3%	< 1%

I. Topologi Sistem

Digunakan empat topologi dengan jumlah *switch* berbeda, yaitu 4, 6, 8 dan 10 *switch*.

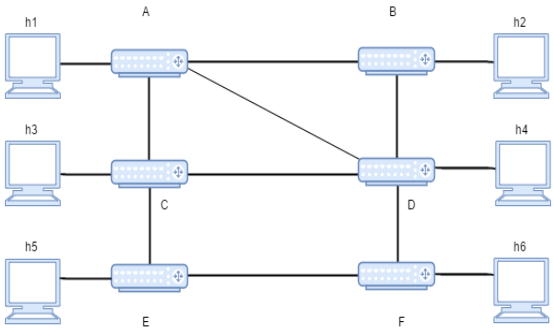
Topologi 1 terdiri dari 4 buah *switch OpenFlow*, 1 buah *controller* POX, 4 buah *host*, 5 buah link dan

eight weight diatur *static* sesuai skenario. Pada Gambar 6 ditunjukkan topologi 1.



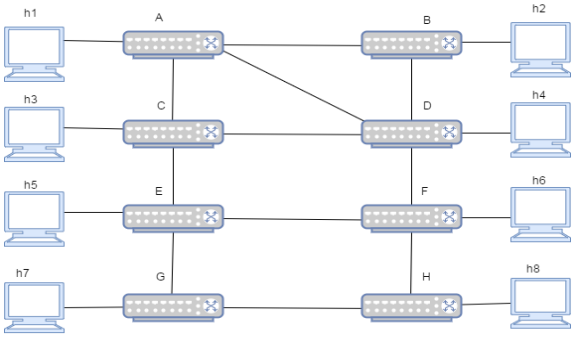
Gambar 6. Topologi 1

Topologi 2 terdiri dari 6 buah *switch OpenFlow*, 1 buah *controller* POX, 6 buah *host*, 8 buah *link* dan *eight weight* diatur *static* sesuai skenario. Pada Gambar 7 ditunjukkan topologi 2.



Gambar 7. Topologi 2

Topologi 3 terdiri dari 8 buah *switch OpenFlow*, 1 buah *controller* POX, 8 buah *host*, 11 buah link dan *eight weight* diatur static sesuai skenario. Pada Gambar 8 ditunjukkan topologi 3.



Gambar 8. Topologi 3

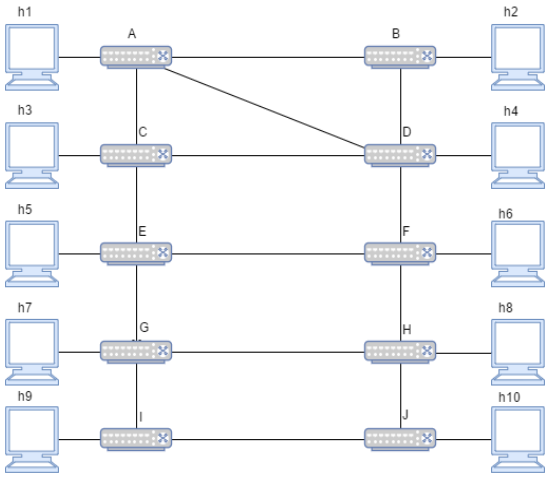
Topologi 4 terdiri dari 10 buah *switch OpenFlow*, 1 buah *controller* POX, 10 buah *host*, 14 buah *link* dan *eight weight* diatur static sesuai skenario. Pada Gambar 9 ditunjukkan topologi 4.

J. Konfigurasi RouteFlow

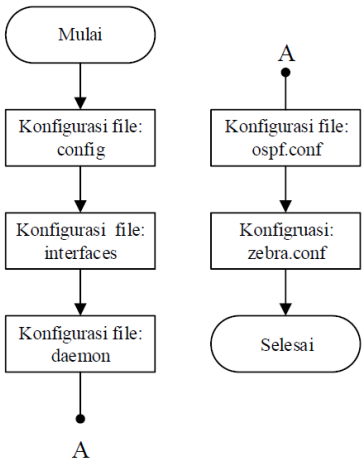
Langkah awal yaitu instalasi *RouteFlow* pada VM dengan spesifikasi yang telah dijelaskan sebelumnya. Ketentuan utamanya yaitu sistem operasi Ubuntu 12.04 LTS dan memiliki RAM minimal 2 GB. *RouteFlow* yang digunakan dalam penelitian ini adalah *RouteFlow* dengan *controller* POX dan *OpenFlow*.

RouteFlow terdiri dari beberapa komponen utama di dalamnya seperti linux container, MongoDB, dan POX. Bagian yang perlu dikonfigurasi yaitu RFClient yang terletak di dalam linux container (LXC).

LXC ini berperan sebagai RFVM (*RouteFlow Virtual Machine*). Selain itu ada konfigurasi *file* rftest yang merupakan program utama untuk menjalankan *RouteFlow*. Pada Gambar 10 ditunjukkan konfigurasi RFVM.



Gambar 9. Topologi 4



Gambar 10. Langkah Konfigurasi RFVM

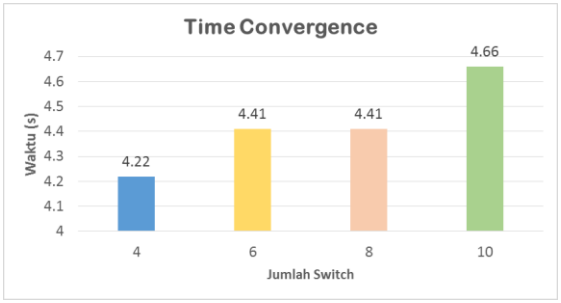
III. HASIL DAN PEMBAHASAN

A. Hasil Pengujian Parameter Time Convergence

Time convergence adalah waktu yang dibutuhkan untuk mengembalikan jaringan pada kedudukan *steady state* setelah adanya perubahan terhadap keadaan jaringan tersebut. Pada penelitian ini yaitu perubahan kondisi jaringan akibat adanya pemutusan *link*.

Pada Gambar 11 ditunjukkan nilai rata-rata *time convergence* pada topologi dengan jumlah *switch* 4, 6, 8 dan 10. Untuk nilai *convergence time* yang didapatkan pada penelitian ini, secara keseluruhan

meningkat seiring bertambahnya jumlah *switch*, namun perbedaan waktu tidak terlalu signifikan karena setiap *routing engine* memiliki konfigurasi yang sama yaitu menggunakan konfigurasi *default*. Untuk semua *RFClient* diatur *hello-interval* sebesar 1 *second* dan *dead-interval* sebesar 4 *second*.



Gambar 11. Grafik Hasil Pengukuran *Time Convergence*

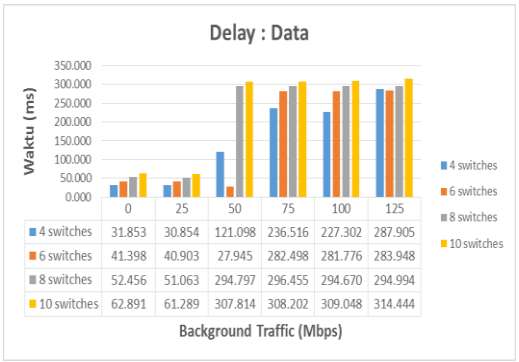
Setiap *routing engine* OSPF akan melakukan penyebaran LSA *update*. Sehingga *routing engine* dapat melakukan perhitungan terhadap informasi dari *link state* yang ada pada jaringan menggunakan algoritma SPF. Ketika terjadi pemutusan *link* maka *link state* yang sebelumnya terbentuk akan berubah. Algoritma SPF melakukan perhitungan kembali untuk mencari *path* terbaik untuk mengirimkan paket ke penerima. Hasil perhitungan tersebut dimasukan ke dalam *forwarding information base* yang kemudian dikirimkan ke bagian *data plane*.

B. *Hasil Pengujian Quality of Service (QoS)*

Parameter yang dianalisis untuk melihat performansi jaringan ini adalah *delay*, *jitter*, *packet loss*, dan *throughput* dari ketiga jenis layanan yaitu VoIP, data, dan video streaming.

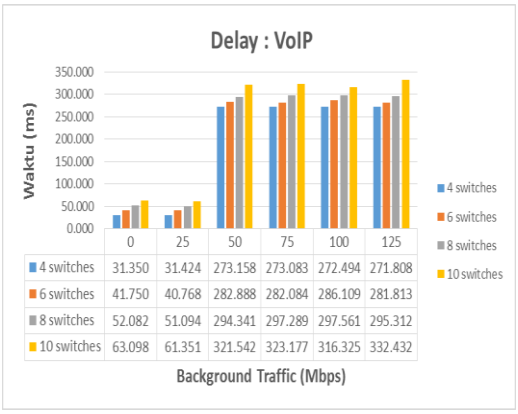
a) *Delay*

Delay adalah waktu yang diperlukan paket untuk melakukan perjalan dari *node* sumber ke *node* tujuan. *Delay* yang diukur dalam penelitian ini adalah *one-way delay*.



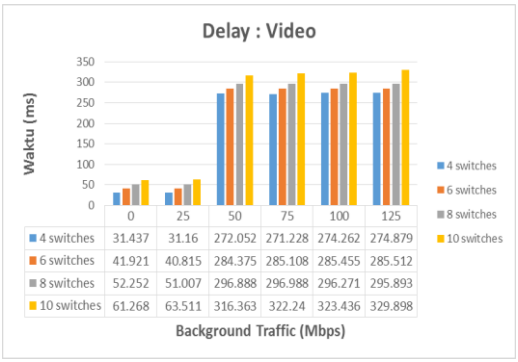
Gambar 12. Grafik Hasil Pengukuran *Delay* Layanan Data

Delay rata – rata untuk layanan Data menunjukan hasil yang meningkat, seiring bertambahnya jumlah switch dan background traffic Hal ini dikarenakan OSPF yang digunakan tidak mendukung *multipath*, sehingga jalur yang dipilih untuk menuju node tujuan berupa *single path*. *Single path* ini yang menyebabkan ketika kondisi jaringan penuh akan terjadi antrian yang lebih lama sehingga *delay* pun akan meningkat. Selain itu dengan jumlah node yang bertambah, maka waktu yang dibutuhkan untuk mencapai penerima pun bertambah lama.



Gambar 13. Grafik Hasil Pengukuran *Delay* Layanan VoIP

Di Gambar 13 terlihat *delay* rata – rata untuk layanan VoIP menunjukan hasil yang meningkat, seiring bertambahnya jumlah *switch* dan *background traffic*. Hasil pengujian *delay* VoIP di atas menunjukan nilai yang tidak jauh berbeda dengan *delay* pada layanan Data. Hal ini disebabkan tidak adanya skala prioritas untuk setiap jenis paket dan jalur yang dilalui semua paket sama.

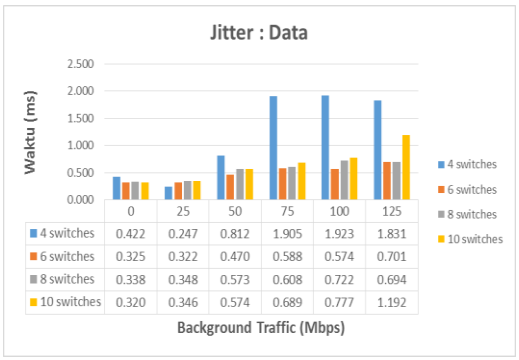


Gambar 14. Grafik Hasil Pengukuran *Delay* Layanan Video

Pada Gambar 14. Terlihat besar *delay* untuk layanan Video memiliki tren yang sama dengan VoIP dan data. Peningkatan pesat ditunjukkan pada penambahan *background traffic* 25 hingga 50 Mbps. Penambahan *background trafik* dari 25 hingga 125 Mbps menunjukan nilai yang konstan untuk setiap jumlah *switch* yang berbeda. Berdasarkan hasil yang didapat dari pengujian dan mengacu pada standarisasi yang ditentukan ITU-T yaitu G.1010, trafik data dan VoIP, yang dialirkan pada jaringan SDN dengan protokol OSPF telah memenuhi standar, tetapi khusus untuk layanan Video hanya memenuhi standar saat *background traffic* dibawah 50Mbps saja, hal ini menunjukkan bahwa besar *background traffic* mempengaruhi delay layanan video.

b) *Jitter*

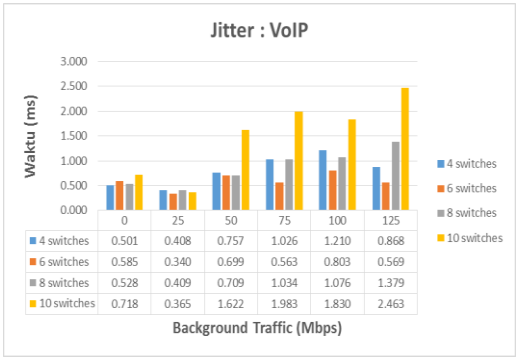
Jitter merupakan variasi *delay* yang diakibatkan oleh panjang antrian dalam suatu pengolahan data dan *reassemble* paket data di akhir pengiriman yang diakibatkan kegagalan sebelumnya.



	0	25	50	75	100	125
4 switches	0.422	0.247	0.812	1.905	1.923	1.831
6 switches	0.325	0.322	0.470	0.588	0.574	0.701
8 switches	0.338	0.348	0.573	0.608	0.722	0.694
10 switches	0.320	0.346	0.574	0.689	0.777	1.192

Gambar 15. Grafik Hasil Pengukuran *Jitter* Layanan Data

Kepadatan *link* juga dapat menyebabkan tumbukan atau *congestion* di dalam jaringan, sehingga *delay* akan semakin bervariasi untuk setiap paket yang berhasil diterima. Pada Gambar 15 menunjukkan hasil *jitter* untuk layanan Data. *Jitter* pada setiap paket yang diterima mengalami fluktuasi waktu untuk setiap penambahan *background traffic* dan *switch*. Namun dapat kita lihat pada topologi dengan 4 *switch* ketika penambahan *background traffic* dari 50 hingga 125 Mbps, nilai *jitter* lebih tinggi dibandingkan dengan topologi lainnya.

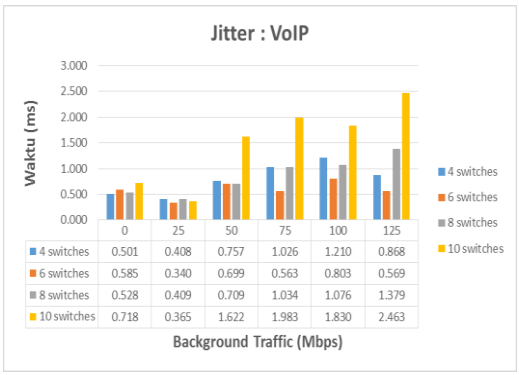


	0	25	50	75	100	125
4 switches	0.501	0.408	0.757	1.026	1.210	0.868
6 switches	0.585	0.340	0.699	0.563	0.803	0.569
8 switches	0.528	0.409	0.709	1.034	1.076	1.379
10 switches	0.718	0.365	1.622	1.983	1.830	2.463

Gambar 16. Grafik Hasil Pengukuran *Jitter* Layanan VoIP

Pada Gambar 16 menunjukkan hasil *jitter* untuk layanan VoIP. *Jitter* pada setiap paket yang diterima mengalami fluktuasi waktu untuk setiap penambahan *background traffic* dan *switch*. Namun dapat kita lihat pada topologi dengan 10 *switch* ketika penambahan *background traffic* dari 50 hingga 125 Mbps, nilai *jitter* lebih tinggi dibandingkan dengan topologi lainnya dan juga sudah melebihi standar ITU.T yaitu < 1 ms.

Pada Gambar 17 Nilai *jitter* untuk layanan video memiliki tren yang berbeda dengan layanan Data tetapi sama dengan layanan VoIP (nilai *jitter* yang tinggi pada jumlah *switch* 10). Karena ukuran paket video yang lebih besar dari keduanya, *delay* untuk setiap paketnya semakin bervariasi.



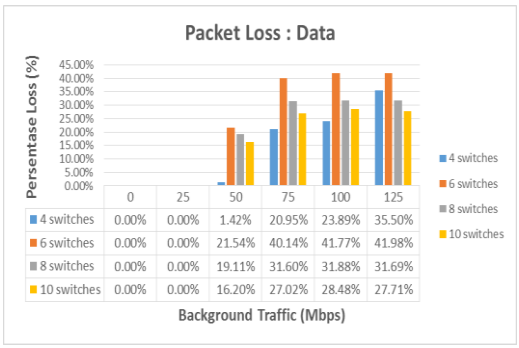
	0	25	50	75	100	125
4 switches	0.501	0.408	0.757	1.026	1.210	0.868
6 switches	0.585	0.340	0.699	0.563	0.803	0.569
8 switches	0.528	0.409	0.709	1.034	1.076	1.379
10 switches	0.718	0.365	1.622	1.983	1.830	2.463

Gambar 17. Grafik Hasil Pengukuran *Jitter* Layanan Video

Nilai *jitter* untuk layanan Data yang berhasil dikirimkan telah memenuhi standar ITU-T G.1010. Namun untuk layanan VoIP dan *Video* pada *background traffic* 50 hingga 125 Mbps nilai *jitter* masih di atas 1 ms.

c) *Packet Loss*

Packet loss adalah banyaknya paket yang gagal dikirimkan dibandingkan dengan banyaknya paket yang dikirimkan.

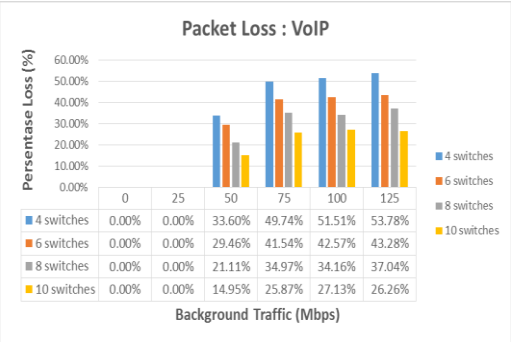


	0	25	50	75	100	125
4 switches	0.00%	0.00%	1.42%	20.95%	23.89%	35.50%
6 switches	0.00%	0.00%	21.54%	40.14%	41.77%	41.98%
8 switches	0.00%	0.00%	19.11%	31.60%	31.88%	31.69%
10 switches	0.00%	0.00%	16.20%	27.02%	28.48%	27.71%

Gambar 18. Grafik Hasil Pengukuran *Packet loss* Layanan Data

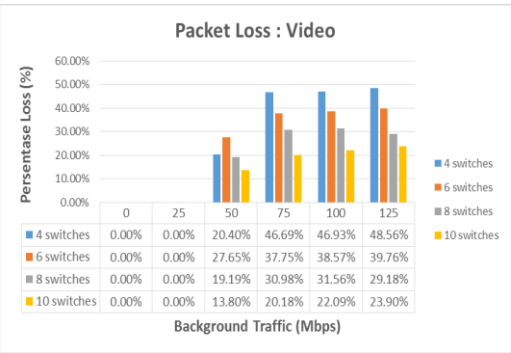
Gambar 18 menunjukkan nilai *packet loss* untuk layanan Data. Pada penambahan background traffic 0 hingga 25 Mbps tidak ada paket yang hilang. Namun pada penambahan background traffic 50 hingga 125 Mbps terdapat *packet loss* dengan rentang nilai > 20%. Hal ini terjadi karena *packet loss* dipengaruhi oleh kondisi *link* serta banyaknya paket yang dialirkan pada jaringan.

Gambar 19 dan Gambar 20 menunjukkan nilai *packet loss* untuk layanan VoIP dan *Video*. Pada penambahan *background traffic* 0 hingga 25 Mbps tidak ada paket yang hilang. Namun pada penambahan *background traffic* 50 hingga 125 Mbps terdapat *packet loss* dengan rentang nilai > 20%. *Packet loss* pada layanan tersebut menunjukkan bahwa semakin banyak jumlah *switch*, maka paket yang hilang semakin sedikit.



	0	25	50	75	100	125
4 switches	0.00%	0.00%	33.60%	49.74%	51.51%	53.78%
6 switches	0.00%	0.00%	29.46%	41.54%	42.57%	43.28%
8 switches	0.00%	0.00%	21.11%	34.97%	34.16%	37.04%
10 switches	0.00%	0.00%	14.95%	25.87%	27.13%	26.26%

Gambar 19. Grafik Hasil Pengukuran *Packet loss* Layanan VoIP

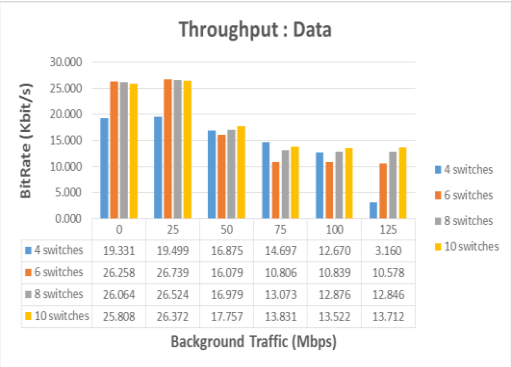


	0	25	50	75	100	125
4 switches	0.00%	0.00%	20.40%	46.69%	46.93%	48.56%
6 switches	0.00%	0.00%	27.65%	37.75%	38.57%	39.76%
8 switches	0.00%	0.00%	19.19%	30.98%	31.56%	29.18%
10 switches	0.00%	0.00%	13.80%	20.18%	22.09%	23.90%

Gambar 20. Grafik Hasil Pengukuran Packet loss Layanan Video

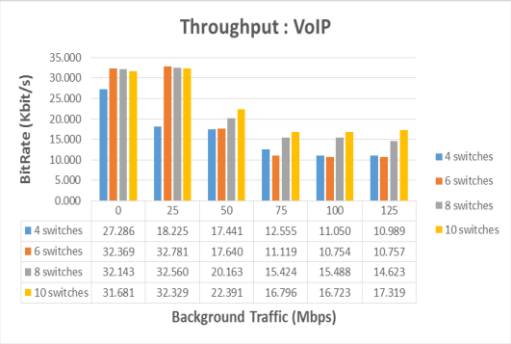
d) *Throughput*

Merupakan perbandingan banyaknya paket yang diterima per waktu pengamatan.



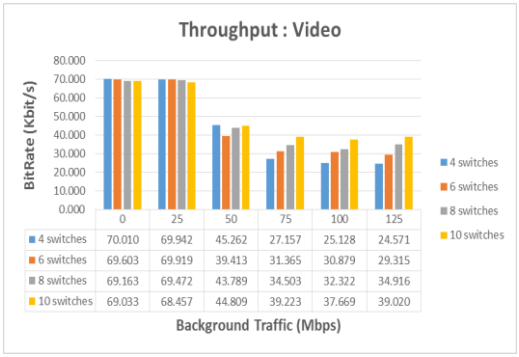
	0	25	50	75	100	125
4 switches	19.331	19.499	16.875	14.697	12.670	3.160
6 switches	26.258	26.739	16.079	10.806	10.839	10.578
8 switches	26.064	26.524	16.979	13.073	12.876	12.846
10 switches	25.808	26.372	17.757	13.831	13.522	13.712

Gambar 21. Grafik Hasil Pengukuran *Throughput* Layanan Data



	0	25	50	75	100	125
4 switches	27.286	18.225	17.441	12.555	11.050	10.989
6 switches	32.369	32.781	17.640	11.119	10.754	10.757
8 switches	32.143	32.560	20.163	15.424	15.488	14.623
10 switches	31.681	32.329	22.391	16.796	16.723	17.319

Gambar 22. Grafik Hasil Pengukuran *Throughput* Layanan VoIP



	0	25	50	75	100	125
4 switches	70.010	69.942	45.262	27.157	25.128	24.571
6 switches	69.603	69.919	39.413	31.365	30.879	29.315
8 switches	69.163	69.472	43.789	34.503	32.322	34.916
10 switches	69.033	68.457	44.809	39.223	37.669	39.020

Gambar 23. Grafik Hasil Pengukuran *Throughput* Layanan Video

Gambar 21, 22 dan 23 menunjukan hasil pengukuran *throughput* untuk setiap layanan pada setiap topologi dan semua kondisi penambahan *background traffic*.

Dapat dilihat bahwa secara keseluruhan, semakin tinggi penambahan *background traffic*, maka jumlah *throughput* semakin kecil, dan semakin banyak jumlah *switch* maka nilai *throughput* semakin besar. Untuk ketiga jenis layanan, nilai *throughput* yang dihasilkan sudah sesuai dengan perhitungan *bandwidth* minimal yang dibutuhkan. Untuk nilai *throughput*, tidak didefinisikan berapa besar yang harus dicapai setiap layanan.

IV. PENUTUP

A. *Kesimpulan*

Berdasarkan parameter *time convergence* pada 4 jenis topologi didapatkan waktu sebesar 4.22 ms (untuk 4 *switch*), 4.41 ms (untuk 6 *switch*), 4.41 ms (untuk 8 *switch*) dan 4.66 ms (untuk 10 *switch*). Nilai *convergence time* yang dihasilkan menunjukan nilai yang berbeda terhadap penambahan jumlah *switch*. Penambahan jumlah *switch* mengakibatkan penambahan *convergence time*.

Hasil pengujian QoS pada topologi 4, 6, 8 dan 10 *switch* dengan parameter *delay*, *jitter*, *packet loss*, dan *throughput* masih dalam rentang nilai yang ditetapkan pada ITU-T G.1010. Namun pada penambahan *background traffic* 50 hingga 125 Mbps untuk layanan Data, VoIP dan Video, nilai parameter *jitter* dan *packet loss* memburuk sehingga melebihi standar ITU-T G.1010. Besarnya topologi jaringan dan beban trafik jaringan menyebabkan kinerja OSPF kurang baik.

B. *Saran*

Selanjutnya dapat mencoba menggunakan bentuk topologi dan jumlah *switch* dan *host* yang berbeda untuk mengetahui skalabilitas sistem simulasi yang dibuat.

DAFTAR PUSTAKA

[1] Rawat Danda B, Reddy. Swetha R., “Software Defined Networking Architecture, Security and Energy

Efficiency: A Survey," IEEE Communications Surveys & Tutorials. 2016.

[2] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," IEEE Proceedings, vol. 103, no. 1, pp. 14–76, 2015.

[3] B. Heller, R. Sherwood, N. McKeown, "The *Controller* Placement Problem," First Workshop on Hot Topics in Software-Defined Networks, pp. 7-12, 2012.

[4] Zhang, H., & Yan, J. (2015). Performance of SDN Routing in Comparison with Legacy Routing Protocol. 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 0–3. https://doi.org/10.1109/CyberC.2015.30

[5] Zeng, P., Nguyen, K., Shen, Y., & Yamada, S. (2014). On The Resilence of Software Defined Routing Platform. APNOMS.

[6] ONF White Paper, "Software-Defined Networking: The New Norm for Networks", ONF, (2012).

[7] Sudiyatmoko, Abu Riza, Sofia Naning Hertiana, and Ridha Muldina Negara. "Analisis Performansi Perutingan Link State Menggunakan Algoritma Djikstra Pada Platform Software Defined Network (SDN)." JURNAL INFOTEL 8.1 (2016): 40-46.

[8] Nakahodo, Y., Naito, T., & Oki, E. (2014). IMPLEMENTATION OF SMART-OSPF IN HYBRID SOFTWARE-DEFINED NETWORK. IEEE IC-NIDC 2014, 0–4. Retrieved from http://ieeexplore.ieee.org/document/7000328/

[9] ONF White Paper, "SDN Architecture Overview", ONF, (2013).

[10] Moy, J. (1998). OSPF Version 2. IETF.

[11] RouteFlow, Diunduh pada laman http://www.OpenFlowhub.org/display/RouteFlow/RouteFlow+Home\

[12] M. Nascimento, C. Rothenberg, M. Salvador, C. Corrêa, S. Lucena, and M. Magalhães, "Virtual routers as a service: the RouteFlow approach leveraging software-defined networks", Proceedings of the 6th International Conference on Future Internet Technologies, (2011) June 13-15; Seoul, Korea..

[13] Quagga Routing Suite, http://www.nongnu.org/quagga/.

[14] M. Nascimento, C. Rothenberg, M. Salvador, and M. Magalhães, QuagFlow: Partnering Quagga with OpenFlow. Proceedings of the ACM SIGCOMM 2010 conference, (2010) August 30-September 3; New Delhi, India.

[15] Production Quality, Multilayer Open Virtual Switch, openvswitch.org, 2016.