# SIMULASI DAN ANALISIS PERFORMANSI PROTOKOL RUTING EBGP PADA SDN (*SOFTWARE DEFINED NETWORK*)

# SIMULATION AND PERFOMANCE ANALYSIS OF EBGP ROUTING PROTOCOL ON SDN (SOFTWARE DEFINED NETWORK)

**Fahry Adnantya[1], Sofia Naning Hertiana, ST., MT.[2], Leanna Vidya Yovita ST., MT.[3]**

[1,2,3] Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]fahryadnantya@telkomuniversity.ac.id, [2]sofiananing@telkomuniveristy.co.id,
[3]leannavidya@telkomuniversity.ac.id

**Abstrak**

**Jumlah pengguna Internet yang meningkat menyebabkan peneliti untuk mempertahankan kinerja Internet, salah satunya ruting. Border Gateway Protocol (BGP) adalah protokol ruting yang menghubungan semua domain jaringan di Internet. Sedangkan *external* BGP (eBGP) merupakan mekanisme yang terjadi ketika pertukaran informasi antar *autonomous system* (AS). Namun protokol tersebut masih berjalan pada perangkat jaringan tradisional, yang mana fungsi *control plane* dan *forwarding plane*-nya berada dalam satu perangkat. *Software defined network* (SDN) merupakan konsep jaringan yang memisahkan fungsi *control-plane* (kontroler) dengan *dataplane*. Hal tersebut mengakibatkan perangkat jaringan (misalnya *router, switch*) hanya meneruskan/tidak meneruskan perintah dari kontroler. Penelitian ini mensimulasikan protokol ruting eBGP pada *platform* SDN, sekaligus menganalisis kinerjanya dengan parameter QoS, waktu konvergensi, *routing overhead* dan *resource utilization*. Hasil penelitian menunjukkan bahwa simulasi protokol eBGP dapat dilakukan pada jaringan SDN, dengan hasil pengukuran Qos (*delay, jitter, packet loss, throughput*) memenuhi standar ITU-T jika dialiri *background traffic* sampai 75 Mbps. Hasil waktu konvergensi memiliki rentang nilai dari 158 sampai dengan 167 detik. Hasil nilai ruting overhead yakni 27 KB (4 *switch),* 39 KB (6 *switch*) dan 66 KB (9 *switch*). Sedangkan persentase penggunaan memori kontroler (*resource utilization*) masing-masing yaitu 8.2% (4 switch), 8.7% (6 *switch*) dan 9.9% (9 *switch*).**

**Kata kunci : *software defined network*, protokol ruting eBGP**

**Abstract**

**The increasing number of Internet users challenges researches to maintain the network performance, including network routing. Border Gateway Protocol (BGP) is routing protocol that connects all the existing autonomous system (AS) on Internet. External BGP (eBGP) is the mechanism that is used to connect the between networks with different ASes. However, eBGP is running on a traditional network devices, the control plane (controller) and data-plane is located in single device. Software defined network (SDN) is the emerging network paradigm that tries to separate control-plane from dataplane. It leads the network devices (e.g routers, switches) will forward the following commands from the controller. This research is to simulate eBGP routing protocols on SDN platform, as well as analyzing its performance in terms of Quality of Service, convergence-delay, routing-overhead and resource utilization. The result shows that eBGP protocol can be simulated on SDN with QoS values (delay, jitter, packet loss, throughput) meets the ITU-T standards, up to 75 Mbps background traffic. The convergence-delay has range of values from 158 to 167 seconds. The result of routing overhead is respectively 27 KB (4 switches), 39 KB (6 switches) and 66 KB (9 switch) and resource utilization of controller is respectively switch, 8.2% (4 switches), 8.7% (6 switches) and 9.9% (9 switch)**

**Keyword: software defined network, eBGP routing protocol**

## 1. Pendahuluan

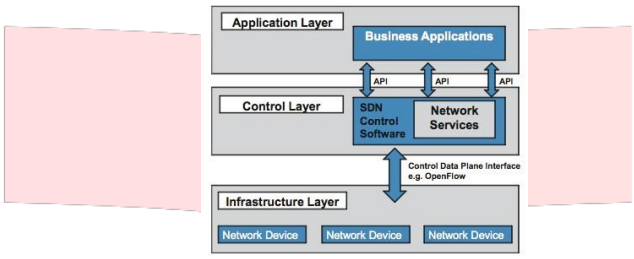
Perkembangan Internet telah mengubah cara hidup masyarakat dalam berbagai aspek. Penelitian telah menemukan bahwa setiap 10% peningkatan penetrasi jaringan pita lebar (*broadband*) berdampak positif pada pertumbuhan PDB sebesar 1.21 hingga 1.38%.[1] Peningkatan kebutuhan Internet ternyata menyebabkan

banyaknya tantangan untuk mempertahankan perfomansinya, salah satunya adalah melalui manajemen ruting. *Border Gateway Protocol* (BGP) merupakan protokol ruting yang menghubungkan jaringa-jaringan domain di Internet. Sedangkan *external* BGP (eBGP) merupakan mekanisme yang terjadi ketika dua domain jaringan saling bertukar informasi. Namun eBGP berjalan pada jaringan tradisional, yang mana fungsi kontrol dan *forwarding* berada dalam satu perangkat. *Software Defined Networking* (SDN) menjadi topik penelitian menarik karena konsepnya yang mencoba memisahkan fungsi kontrol jaringan dan *forwarding*, salah satunya terhadap manajemen ruting. Sehingga penelitian ini mensimulasikan protokol eBGP pada *platform* SDN sekaligus menganalisanya terhadap parameter *Quality of Service*, *convergence-delay, routing overhead* dan *resource utilization*.

## 2. Dasar Teori dan Simulasi Sistem

### 2.1 Software Defined Network dan OpenFlow

*Software Defined Newtork* merupakan paradigm jaringan yang memisahkan *control* dan *data plane*. Metode ini memungkinkan administrator jaringan untuk mengontrol kerja perangkat melalui sebuah kontroler, tanpa harus mengonfigurasi perangkatnya satu-satu..[4]




Gambar 2.1 Arsitektur SDN [4]

Standar SDN yang paling banyak digunakan adalah OpenFlow.[3] OpenFlow terbagi dalam *switch* dan kontroler OpenFlow. *Switch* tersebut bertugas meneruskan paket, sedangkan kontroler bertanggung-jawab untuk menentukan *decision rules* dan meneruskan perintahnya ke *switch*. Perintah tersebut dalam OpenFlow dikenal dengan *flow entry* dan masuk ke dalam *flow table*. Sebuah *flow entry* berisi *matching fields, countres* dan instruksi. Ketika sebuah paket cocok terhadap sebuah *matching fields*-nya, *counters* akan memperbarui dan instruksi dieksekusi terhadap paket tersebut. *Matching fields* tersebut dapat berupa alamat IP, port TCP dll. Sedangkan instruksi misalnya berupa penggantian alamat IP sebuah paket. [5]

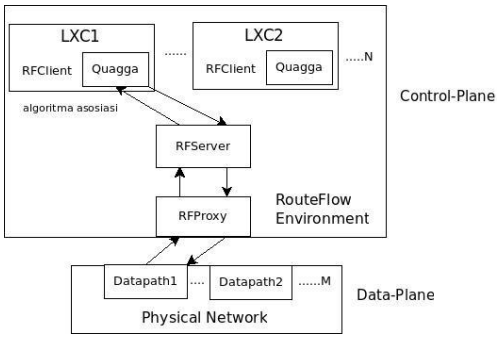
### 2.2 Border Gateway Protocol

Border Gateway Protocol (BGP) adalah protokol ruting inter-domain yang memanfaatkan algortima path vector. BGP mempunyai empat jenis message, yaitu (1) *Open Message* digunakan sebagai message pertama jika sudah terbentuk koneksi TCP di antara kedua sisi (pengirim dan penerima), (2) *KeepAlive Message* dikirim berulang kali untuk memastikan koneksi masih berjalan, (3) *Notification Message* dikirim untuk merespon jika terjadi error, (4) *Update Message* berisi informasi rute yang dikirimkan antar router BGP.[2]

Jaringan domain dalam BGP dikenal dengan *autonomus system* (AS). Setelah koneksi BGP terbentuk dengan sebuah AS, maka kedua AS dapat saling mengirimkan *update message*. BGP hanya memilih satu jalur terbaik kepada setiap tujuannya. Sebuah *update message* digunakan untuk memberi tahu BGP manakala terdapat *switch* baru yang tergabung atau terdapat *switch* yang *down. Update message* berisikan *prefix*, aksi (penggabungan atau pemutusan) dan beberapa atribut seperti AS_PATH atau NEXT_HOP. Bagian paling penting dari AS_PATH adalah AS *sequence*, karena AS *sequence* berisi daftar AS mana saja yang harus dilewati jika ingin mengirimkan paket ke sebuah tujuan.[3]

### 2.3 RouteFlow

RouteFlow terbentuk atas penggabungan proyek OpenFlow dengan *routing engine* Quagga[6]. Sistem ini terdiri dari kontroller OpenFlow (RFProxy), RFClient dan independent Server (RFServer). Tujuan utama dibuatnya RouteFlow adalah menerapkan virtualisasi IP routing secara terpusat, dengan memisahkan fungsi *control-plane* dan *data-plane*. Penelitian ini memanfaatkan RouteFlow sebagai sistem yang berjalan pada *control-plane.* Arsitektur RouteFlow terdiri dari, (1) **RFServer** adalah *standalone application* yang memegang kendali pusat kontrol jaringan. RFServer mengatur virtual machine (VM) yang berjalan pada RFClient dan mengatur *logic process* (seperti *event processing*, VM mapping, *resource management*), (2) **RFProxy** adalah kontroler POX yang bertugas meneruskan kebijakan protokol (misalnya *update route*, konfigurasi datapath) dari RFServer ke *data-plane*, (3) **RFClient** adalah daemon pada VM yang bertugas mendeteksi perubahan informasi ruting dan memberitahukannya ke RFServer. Tugas tersebut dikomunikasikan dengan *routing engine* (Quagga).

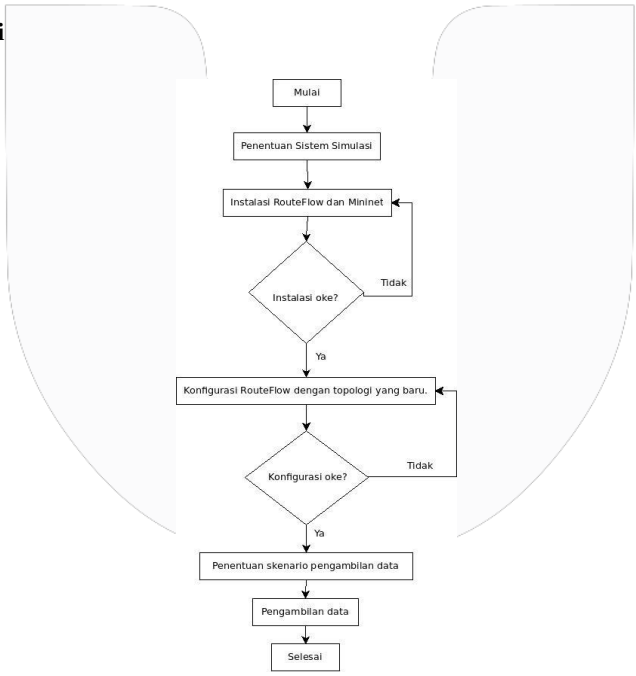



Gambar 2.2 Arsitektur Sistem RouteFlow

Mekanisme RouteFlow dimulai dengan *data-plane* yang mengirimkan *route update* dengan protokol ruting standar (OSPF, BGP) beserta port-nya dalam bentuk *OpenFlow message* ke kontroler (RFProxy). *Message* tersebut diteruskan ke RFServer dalam bentuk *mapping message*. RFServer mengirimkan *mapping message* kepada *routing engine* (Quagga) yang bersangkutan dengan mengecek parameter *dp_id* dan *dp_port*. Kemudian *routing engine* tersebut mengeluarkan *route update* sesuai dengan protokol rutingnya (OSPF, BGP) dan mengirimkannya ke RFClient. Sebelum *route update* diperintahkan ke *data-plane*, maka RFServer melakukan asosiasi antara port pada *virtual switch* dengan port *data-plane*. Setelah terjadi asosiasi, RFServer meminta RFClient untuk menyebarkan *message* kepada *virtual switch* yang terhubung dengan RFProxy. Sehingga RFProxy mengetahui koneksi setiap RFClient dengan *virtual switch*-nya. Kemudian RFServer memerintah RFProxy meneruskan *route update* tersebut ke *data-plane*. RFProxy menerjemahkan konfigurasi protokol ke dalam paket OpenFlow dan mengirimkannya ke *data-plane*.

## 3. Perancangan Simulasi dan Analisa Hasil Pengukuran

### 3.1 Perancangan Simulasi




Gambar 3.1 Alur perancangan simulasi

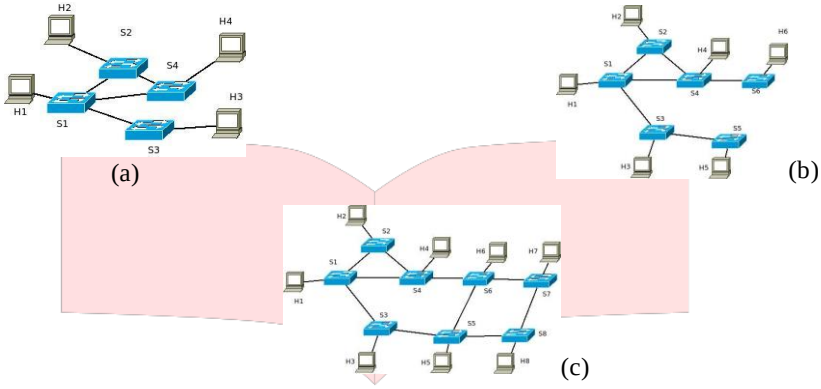
Pengerjaan tugas akhir dimulai dengan menentukan sistem *control-plane* dan *data-plane* yang digunakan. RouteFlow bekerja sebagai *control-plane* dan emulator Mininet sebagai *data-plane*. Instalasi RouteFlow dan Mininet dilakukan pada dua komputer yang terpisah. Pengecekan instalasi keduanya dilakukan dengan menjalankan file **Rftest1** dan **Rftest2**. Kedua file tersebut menjalankan topologi jaringan satu *switch* dan dua *host* (**Rftest1**) dan empat *switch* dan empat *host* (**Rftest2**) yang menjalankan protokol OSPF. Pengecekan konektivitas jaringan dilakukan dengan *command pingall* pada Mininet.

Tahap selanjutnya dilakukan modifikasi konfigurasi pada RouteFlow dan Mininet. Tugas akhir ini mengkonfigurasi topologi jaringan dengan 4-*switch-host* (4 *switch* dan 4 *host*), *6-switch-host*, *8-switch-host*, *9-switch-host* dengan protokol eBGP. Konfigurasi dilakukan pada *control-plane* (config, network, quagga, CSV,

rftest) dan *data*-plane. (topologi.py). Jika konfigurasi sudah berjalan, maka ditentukan skenario pengambilan data.

### 3.2 Skenario Pengujian Sistem

Terdapat tiga parameter yang diukur pada tugas akhir ini, antara lain *Quality of Service* (*delay, jitter, throughput, packet loss*), waktu konvergensi dan *routing overhead*. Parameter-parameter tersebut digunakan sebagai dasar analisis sistem yang digunakan. Topopologi yang digunakan dalam setiap pengukuran adalah topologi *abielene* dengan jumlah *switch* dan *host* yang bervariasi. Seperti pada Gambar 3.2, topologi *abielene* tersebut dihubungkan dengan arsitektur RouteFlow. Sehingga setiap switch *dataplane* terhubung dengan kontroler RouteFlow.




Gambar 3.2 (a) Topologi 4-*host-switch*, (b) Topologi 6-*host-switch*, (c) Topologi 8-*host-switch*

Pengukuran QoS dilakukan untuk mengetahui performansi jaringan apabila dilewatkan trafik data, *video streaming*, dan VoIP. Untuk pengujian QoS ini dilakukan beberapa kali pengujian dengan perubahan terhadap jumlah *switch* dan pemberian *background traffic* (0 – 125 Mbps).

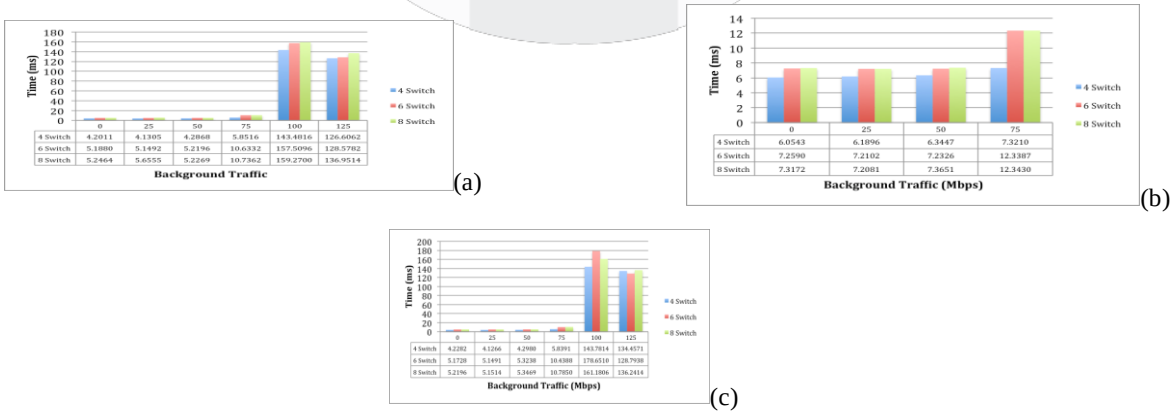
Pengukuran waktu konvergensi menggunakan paket ICMP *ping* sebagai paket yang dikirimkan dari *node* sumber ke *node* tujuan. Ketika *ping* sedang dikirimkan ke *node* tujuan, salah satu *link* yang dilewati paket tersebut diputus. Sehingga pengukuran waktu konvergensi dilakukan selama jaringan membutuhkan waktu untuk memperbarui tabel rutingnya agar kembali saling berkomunikasi.

*Routing Overhead* merupakan berapa banyak paket kontrol yang dibutuhkan jaringan agar semua *node* dapat saling berkomunikasi. Satuan yang digunakan adalah bytes. Pengukuran ini dilakukakan untuk mengetahui paket ruting yang dibutuhkan selama proses perubahan topologi. Perekaman paket dilakukan pada *interface* kontroler *eth0* yang terhubung ke semua *switch.*

Pengukuran *resource utilization* dilakukan untuk mengetahui berapa persentasi memori perangkat kontroler jaringan untuk mengendalikan *switch-switch* tersebut. Pengukuran dilakukan pada setiap topologi untuk melihat perbedaan konsumsi memori.

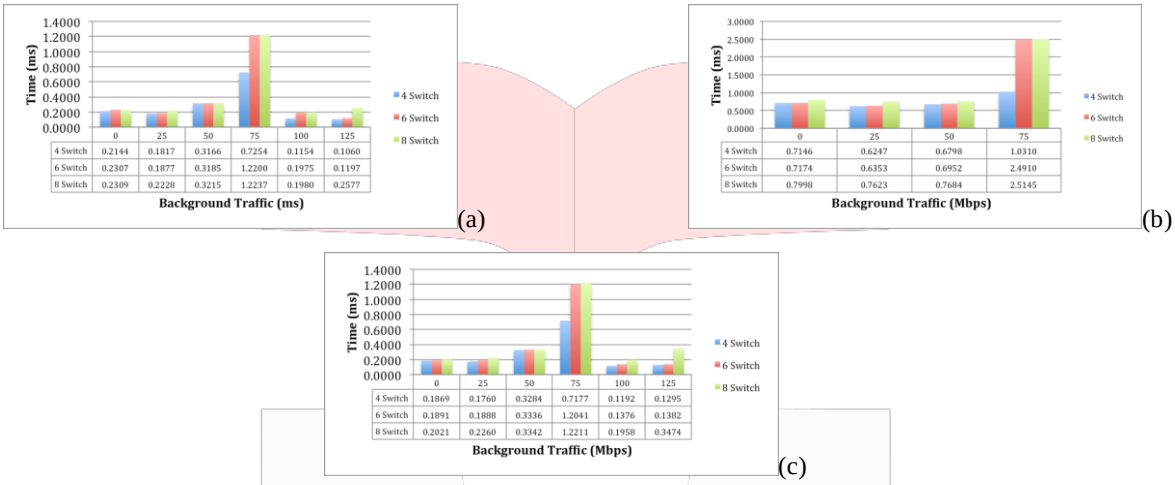
### 3.3 Analisis Hasil Simulasi

#### 3.3.1 Pengukuran *Quality of Service*




Gambar 3.3 (a) Delay data, (b) Delay Video Streaming, (c) Delay VoIP

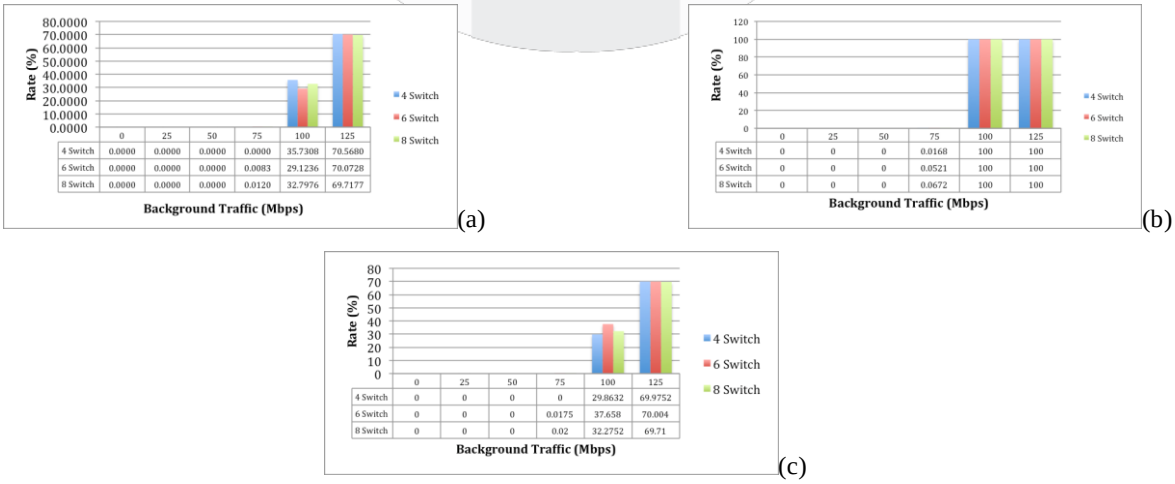
Gambar 4.1 menampilkan *delay* untuk setiap jenis trafik dan sampel yang sukses. Besar *delay* ketiga trafik memiliki nilai yang konstan walaupun dialiri *background traffic* sampai dengan 50 Mbps, dan mengalami sedikit peningkatan ketika dialiri *background traffic* 75 Mbps. Perbedaan *delay* mengalami peningkatan berbanding lurus dengan besar *background traffic*. Hal ini disebabkan utilitas jaringan yang tinggi menyebabkan antrian tiap *node* bertambah, maka waktu kedatangan paket akan lebih lama. Namun trafik data dan VoIP mengalami peningkatan *delay* yang signifikan ketika jaringan dialiri *background traffic* 100 Mbps. Hal ini menandakan kondisi *link* mencapai *bottleneck*. Kapasitas *link* yang hanya 100 Mbps, harus melewatkan tiga jenis trafik beserta *background traffic* sebesar 100 Mbps. Sedangkan trafik video mengalami peningkatan *delay* ketika jaringan dialiri *background traffic* 75 Mbps. Namun *delay* berubah menjadi nol ketika dialiri *background traffic* dari 100 hingga 125 Mbps. Hal tersebut terjadi karena tidak adanya mekanisme prioritas terhadap jenis trafik yang berbeda. Ketika jaringan mengalami bottleneck, maka jaringan hanya dapat meneruskan paket-paket tertentu. Dalam hal ini paket video memiliki *delay* paling besar dibandingkan dengan kedua trafik lainnya. Sehingga *switch* hanya meneruskan paket yang memiliki *delay* lebih rendah.




Gambar 3.4 (a) Jitter data, (b) Jitter Video Streaming, (c) Jitter VoIP

Gambar 4.2 menampilkan kondisi *jitter* untuk setiap jenis trafik dan sampel yang sukses. Ketiga trafik mengalami peningkatan nilai *jitter* ketika dialiri *background traffic* 75 Mbps. Hal ini terjadi karena pengaruh beban trafik dan besarnya kongesti di dalam jaringan yang mengakibatkan *delay* paket bervariasi. Namun *jitter* trafik data dan VoIP ketika *background traffic* 100 Mbps menurun. Hal ini terjadi karena jaringan mengalami kondisi *bottleneck*, sehingga waktu kedatangan semua paket relatif sama. Sedangkan *jitter* trafik video menjadi nol ketika dialiri *background traffic*. Hal ini disebabkan tidak adanya mekanisme prioritas terhadap tiga jenis trafik tersebut.

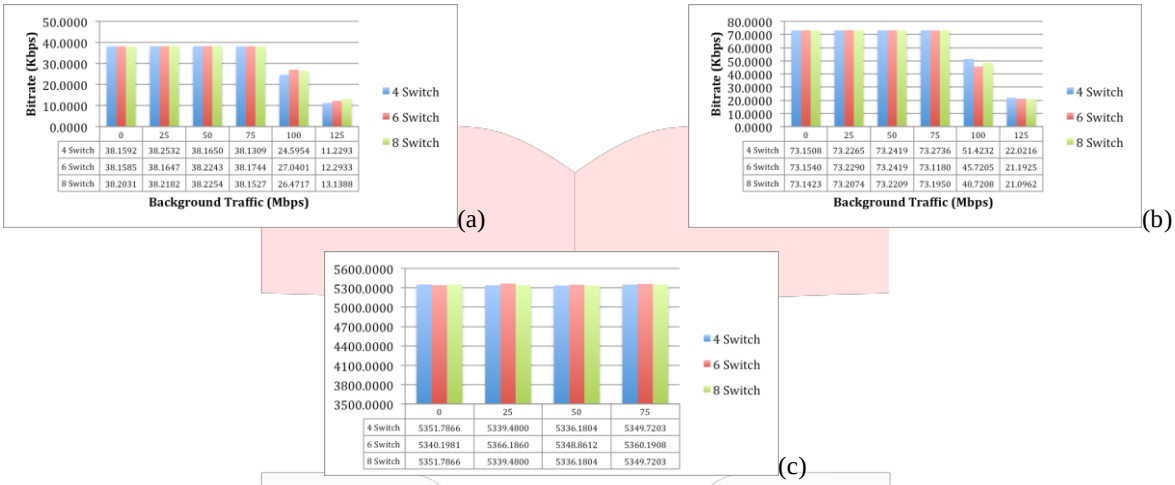
Hasil pengukuran nilai *jitter* oleh tiga jenis trafik masih memenuhi standar ITU-T (di bawah 1 ms) jika dialiri *background traffic* sampai 50 Mbps. Pada aliran *background traffic* sebesar 75 Mbps nilainya melebih 1 ms, namun nilai *jitter* kembali stabil pada aliran *background traffic* sampai dengan 125 Mbps.



Gambar 3.5 (a) Packet loss data, (b) Packet loss Video Streaming, (c) Packet loss VoIP

Gambar 3.5 menampilkan persentase *packet loss* setiap jenis trafik dan sampel yang sukses. Trafik data dan VoIP mulai terdapat *packet loss* ketika *background traffic* sebesar 100 Mbps, masing-masing dengan nilai 35.7 % dan 29.8 %. Nilai tersebut meningkat sampai dengan *background traffic* 125 Mbps. Namun trafik video mengalami peningkatan drastis hingga mencapai 100 % pada *background traffic* 100 Mbps dan seterusnya. Hal ini terjadi kongesti pada jaringan yang dipengaruhi oleh keadaan *link* dan banyaknya paket yang harus dilewatkan. Oleh karena itu, pada saat jaringan dalam kondisi *bottleneck*, maka nilai *packet loss* meningkat.

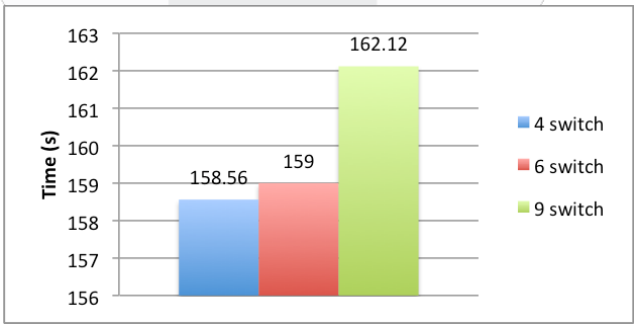
Hasil pengukuran *packet loss* oleh tiga layanan masih memenuhi standar (di bawah 3%) jika dialiri *background traffic* sampai dengan 75 Mbps.




Gambar 3.6 (a) Throughput data, (b) Throughput Video Streaming, (c) Throughput VoIP

Gambar 4.4 menampilkan *throughput* setiap jenis trafik dan sampel yang sukses. Sampai dengan *background traffic* 75 Mbps, nilai *throughput* ketiga jenis trafik kosnstan, masing-masing bernilai 5,4 Mbps (Video), 73 Kbps (VoIP) dan 38 (data). Nilai-nilai tersebut sesuai dengan perhitungan *bandwidth* minimal untuk setiap layanan. Namun ketiga trafik mengalami penurunan ketika dialiri *background traffic* 100 Mbps. Nilai *throughput* mengalami penurunan seiring dengan bertambahnya *background traffic* dan turun secara signifikan ketika *background traffic* melebihi kapasitas *link* yang tersedia. Jika jumlah paket yang dikirimkan meningkat, maka jumlah paket yang sampai di *node* tujuan akan berkurang. Hal ini sesuai dengan pengertian *throughput*, yakni besarnya jumlah paket yang diterima dalam satuan waktu.

### 3.3.2 Pengukuran Waktu Konvergensi

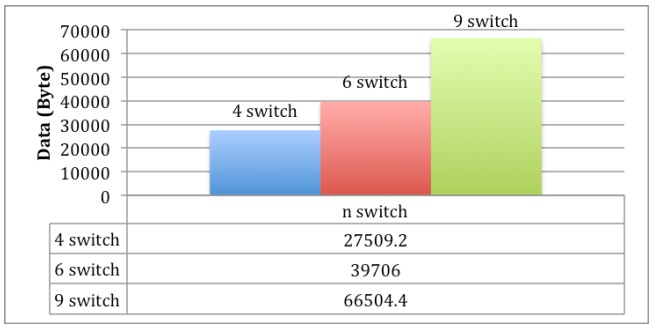



Gambar 3.7 Waktu konvergensi

Gambar 4.5 menunjukkan grafik peningkatan waktu konvergensi masing-masing topologi terhadap perubahan *background traffic*. Terlihat bahwa semakin banyak *switch* menyebabkan semakin besar waktu konvergensinya. Dalam mendeteksi adanya *link-failure*, BGP secara berkala mengirimkan KeepAlive *message* antar *switch*. Jika sebuah *switch* tidak menerima *message* tersebut tiga kali berturut-turut, maka dipastikan terjadi *switch* tersebut *down*. Simulasi ini tidak mengubah parameter waktu interval KeepAlive *message* dan waktu *hold-down*. Maka default waktu interval KeepAlive *message* adalah 60 detik dan waktu *hold-down* adalah tiga kali waktu interval KeepAlive *message*, yakni 180 detik. Jika mengacu pada hasil pengukuran, ketiga topologi jaringan tersebut sudah kembali konvergen sebelum mencapai waktu *hold-down*. Namun nilainya bervariasi. Hal

ini juga dipengaruhi oleh jumlah *switch* yang melakukan *update* tabel ruting selama proses konvergensi. Semakin banyaknya *switch* menyebabkan semakin lama waktu untuk memperbarui tabel ruting.

### 3.3.3 Pengukuran *Routing Overhead*

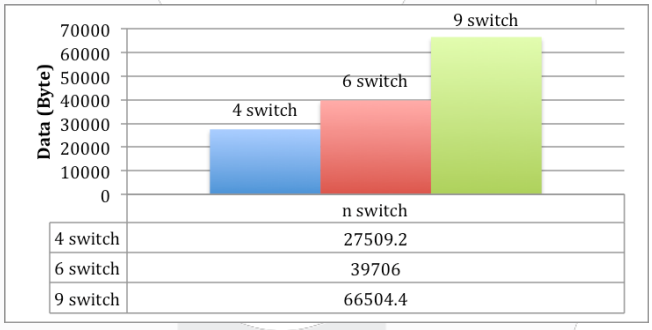



| n switch | |
|---|---|
| 4 switch | 27509.2 |
| 6 switch | 39706 |
| 9 switch | 66504.4 |

Gambar 3.8 *Routing Convergence*

Gambar 3.8 menampilkan grafik hasil pengukuran *overhead* terhadap masing-masing topologi. Hasil pengukuran menunjukkan semakin banyak *switch* yang terdapat dalam topologi menyebabkan peningkatan ukuran *overhead*. Hal ini disebabkan protokol BGP secara berkala mengirimkan KeepAlive *message* antar *switch* yang digunakan untuk memastikan koneksi BGP masih berjalan. Hal ini juga disebabkan faktor kontroler yang digunakan. Konfigurasi 1:1 ruting *virtual-switch* dengan *datapath* pada RouteFlow juga menyebabkan banyaknya jumlah *overhead*. Jika terdapat *virtual switch* yang semakin banyak, maka RFClient mengirimkan RouteInfo *message* yang lebih banyak. Hal tersebut mengakibatkan peningkatan jumlah OpenFlow *message* yang dikirimkan ke *datapath*. Sehingga semakin banyak jumlah *switch datapath*, menyebabkan peningkatan jumlah *overhead*.

### 3.3.4 Pengukuran *Resource Utilization*




| n switch | |
|---|---|
| 4 switch | 27509.2 |
| 6 switch | 39706 |
| 9 switch | 66504.4 |

Gambar 3.9 *Resource Utilization*

Gambar 4.7 memperlihatkan hubungan jumlah *switch* dengan utilitas memori perangkat yang digunakan. Semakin banyak *switch* yang terhubung ke jaringan maka beban jaringan yang digunakan semakin besar. Kenaikan beban jaringan per *switch* dihitung sekitar 15 MB. Oleh karena itu, hal tersebut dapat diasumsikan bahwa perangkat dapat menangani sistem dengan jumlah *switch* 250 *switch*, dengan menggunakan memori 3.8 GB.

## 4. Kesimpulan

Simulasi protokol ruting eBGP pada *software defined networking* dapat dilakukan, dengan hasil evaluasi sebagai berikut :

1. Protokol ruting eBGP dapat disimulasikan pada jaringan SDN.
2. Hasil pengukuran tiga parameter QoS (*delay, jitter, packet loss, throughput*) memenuhi kriteria standar ITU-T G.1010 sampai dengan penambahan *background traffic* 75 Mbps.
3. Dengan memperlakukan variasi topologi dan *background traffic*, nilai waktu konvergensi mempunyai rentang waktu antara 158 – 168 detik. Semakin banyak jumlah *switch* menyebabkan waktu konvergensi semakin lama.
4. Hasil rata-rata *routing overhead* yaitu 27,509 KB (4 *switch*), 39,706 KB (6 *switch*) dan 66,504 KB (9 *switch*). Hal ini menunjukkan bahwa semakin banyak jumlah *switch* menyebabkan peningkatan nilai *overhead*-nya.

5. Hasil pengukuran *memory utilization* yaitu 8,2 % (4 *switch*), 8,7 % (6 *switch*) dan 9,9 % (9 *switch*). Hal ini menunjukkan bahwa semakin banyak jumlah *switch* menyebabkan peningkatan konsumsi memori pada kontroler.

**Daftar Pustaka**

[1] "*Trends in Telecommunication Reform 2010–2011: Enabling Tomorrow's Digital World*, International Telecommunications Union, 2011.
[2] B. A. Forouzan, "Data communication and Networking", McGraw-Hill, New York, 2007.
[3] G. Adrian, "Evaluating the Effect of SDN Centralization on Internet Routing Convergence", ETH Zurich, 2004.
[4] H. Fei, dkk, "A Survey on Software-Defined Network (SDN) and OpenFlow: From Concept to Implementation" IEEE Communication Surveys & Tutorials, 2013.
[5] *Open Networking Foundation*, http://www.opennetworking.org/.
[6] J. Stringer, C. Owen, "RouteMod: A Flexible Approach to Route Propagation", 2013.