BORDER GATEWAY PROTOCOL PADA TEKNOLOGI *SOFTWARE DEFINED NETWORK*

BORDER GATEWAY PROTOCOL ON SOFTWARE DEFINED NETWORK TECHNOLOGY

Veronica Windha, Lukas

Jurusan Teknik Elektro – Fakultas Teknik
Universitas Katolik Indonesia Atma Jaya - Jakarta

Abstrak

Peningkatan kebutuhan internet harus diseimbangkan dengan mempertahankan kinerja *routing*. Protokol *routing Border Gateway Protocol* (BGP) digunakan untuk menghubungkan semua *Autonomous System* (AS) di internet dengan mekanisme eksternal BGP (eBGP) untuk pertukaran informasi *routing* antar-AS. Namun, BGP masih bekerja dalam jaringan tradisional yang kompleks dengan pengendalian dan penerusan paket berada dalam *device* yang sama. Teknologi *Software Defined Network* (SDN) akan memisahkan *control plane* dan *data plane*, dengan pengendalian berpusat pada kontroler dan penerusan paket dijalankan oleh *device* jaringan. Penelitian ini mensimulasikan BGP pada teknologi SDN untuk melihat cara kerja BGP dalam pemilihan jalur jika ada *link* yang putus dan menguji kinerja dari protokol *routing* BGP dalam dua AS yang berbeda pada teknologi SDN. Simulasi dilakukan dengan menggunakan Mininet. Indikator kinerja yang diuji adalah *jitter* dan *throughput*. Dari serangkaian simulasi yang dilakukan, terlihat bahwa protokol *routing* akan mencari jalur yang lain jika terdapat *link* yang putus, *jitter* sebesar 0,44 milisekon dan *throughput* sebesar 99,78%.

Kata kunci: BGP, mininet, *jitter, throughput*

Abstract

There must be a balance between the rising use of internet and the routing performance. Border Gateway Protocol (BGP) connects all Autonomous System (AS) on the internet using external BGP (eBGP) mechanism for routing the information exchange between AS. However, BGP runs in a traditional complex network in which controls and forwards packets within the same device. Software Defined Network (SDN) separates control plane from data plane, where controller manages while network device forwards the packets. This research simulates BGP on SDN technology to evaluate the BGP mechanism in route selection shall there any disconnected link, and measures the performance of BGP routing protocol at two different AS on SDN technology. The simulation was run using Mininet. The performance indicators tested are jitter and throughput. A series of simulation showed that routing protocol would search for another link when disconnected, with a jitter of 0.44 ms and throughput of 99.78%.

Keywords: *BGP, mininet, jitter, throughput*

Tanggal Terima Naskah : 05 Desember 2016
Tanggal Persetujuan Naskah : 11 April 2017

1. PENDAHULUAN

Perkembangan zaman yang semakin maju mengakibatkan masyarakat tidak dapat terlepas dari penggunaan internet. Peningkatan kebutuhan internet harus diseimbangkan dengan mempertahankan kinerjanya, salah satunya dalam hal *routing*. Jaringan internet yang sangat besar membutuhkan protokol *routing* yang dapat mengelola seluruh kegiatan *routing*. Protokol *routing* tersebut adalah *Border Gateway Protocol* (BGP), sebagai penghubung semua *Autonomous System* (AS) di internet. Jumlah AS dalam internet sangat banyak, sehingga untuk mempertahankan kinerja *routing* diperlukan pertukaran informasi *routing* antar AS yang disebut mekanisme *routing external* BGP (eBGP) dan pertukaran informasi *routing* dalam AS yang sama yang disebut mekanisme *routing internal* BGP (iBGP). Namun protokol *routing* BGP tersebut masih bekerja pada jaringan tradisional yang sangat kompleks.

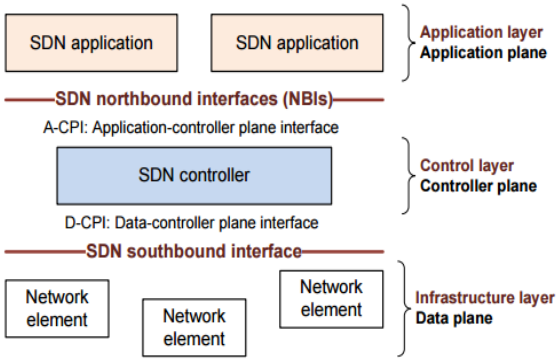
Tugas untuk mengendalikan dan meneruskan paket pada jaringan tradisional digabung menjadi satu dan dilakukan oleh *router*, sehingga jika protokol *routing* eBGP ingin menambah informasi tabel *routing*, maka harus dilakukan konfigurasi pada *router*, hal tersebut sangat rumit dan membutuhkan waktu yang lama jika terdapat banyak sekali *router* yang harus dikonfigurasi, mengingat jumlah AS yang sangat banyak dalam internet. Permasalahan *routing* pada jaringan tradisional dapat diatasi dengan menerapkannya pada teknologi *Software Defined Network* (SDN).

Teknologi SDN memisahkan *control plane* dan *data plane*, dengan penentuan *routing* berpusat pada kontroler dan penerusan paket dijalankan oleh *device* jaringan, sehingga tidak perlu lagi melakukan konfigurasi *device* satu persatu, menghemat waktu, mengurangi kesalahan konfigurasi, dan meningkatkan kinerja jaringan. Oleh karena itu, dilakukan simulasi BGP pada teknologi SDN.

2. KONSEP DASAR

2.1 *Software Defined Network*

Software Defined Network (SDN) merupakan sebuah konsep baru dalam mendesain, membangun, dan mengelola jaringan komputer dengan memisahkan *control plane* dan *data plane* dalam suatu *device* yang berbeda. Pada jaringan SDN, *data plane* berada pada *device* jaringan dan *control plane* berada pada *device* yang disebut kontroler. Arsitektur SDN dapat dilihat pada Gambar 1.



Gambar 1. Arsitektur SDN [1]

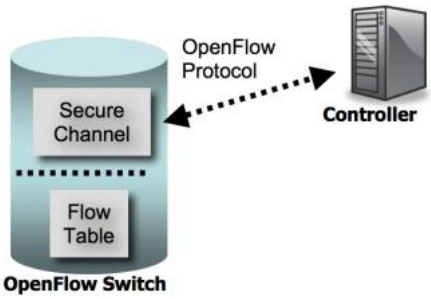
Konsep SDN beroperasi menggunakan protokol *OpenFlow* sehingga *control plane* dapat berkomunikasi dengan *data plane*. *Control plane* memiliki strukur tersentralisasi karena pada *device* jaringan hanya terdiri dari *data plane* yang berfungsi meneruskan paket dan semua keputusan ditentukan oleh kontroler.

2.2 *OpenFlow*

OpenFlow merupakan protokol komunikasi antara *control plane* dan *data plane*. *Device* jaringan yang dikendalikan oleh kontroler *OpenFlow* merupakan *OpenFlow switch.*

2.2.1 *OpenFlow switch*

OpenFlow switch terdiri dari tabel *flow* yang menjalankan *packet lookup* dan *forwarding*, dan sebuah *secure channel* yang terhubung ke kontroler, seperti pada Gambar 2. Semua paket yang diproses oleh *switch*, dibandingkan terhadap tabel *flow*. Jika ditemukan *entry* yang sesuai dengan tabel *flow*, maka paket tersebut akan diteruskan ke *port* yang bersangkutan. Jika tidak ditemukan *entry* yang sesuai dengan tabel *flow*, maka paket diteruskan ke kontroler melalui *secure channel*. Kontroler dapat menambahkan *entry* tersebut pada *flow table switch* atau menghapusnya [2].



Gambar 2. *OpenFlow* [2]

2.2.2 Protokol *OpenFlow*

Kontroler mengelola setiap *OpenFlow switch* melalui *secure channel* dengan menggunakan protokol *OpenFlow*. Protokol *OpenFlow* mendukung tiga tipe pesan, yaitu [2]:

1. *Controller-to-Switch* merupakan pesan yang diinisiasi oleh kontroler, digunakan untuk mengelola atau memeriksa langsung kondisi *switch*.
2. *Asynchronous* merupakan pesan yang diinisiasi oleh *switch*, dikirimkan ke kontroler, digunakan untuk menandakan adanya paket yang datang, perubahan kondisi *switch*, atau *error*.
3. *Symmetric* merupakan pesan yang diinisiasi oleh salah satu dari *switch* atau kontroler.

2.2.3 Kontroler *OpenFlow*

Kontroler pada *control plane* menggunakan protokol *OpenFlow* untuk mengkonfigurasi jaringan dan menghubungkan kontroler ke *device* jaringan, sehingga *server* dapat memberikan informasi ke *device* mana paket harus dikirimkan. Kontroler yang akan digunakan pada penelitian ini adalah POX.

2.3 *RouteFlow*

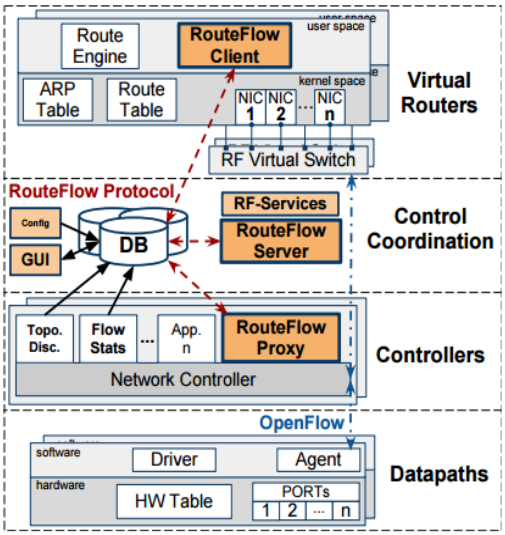
RouteFlow merupakan sebuah *software* yang menyediakan virtualisasi layanan *Internet Protocol* (IP) *routing*, seperti *Open Shortest Path First* (OSPF), *Border Gateway Protocol* (BGP), *Routing Information Protocol* (RIP), melalui jaringan *OpenFlow* yang memisahkan fungsi *control plane* dan *data plane*.

2.3.1 Arsitektur *RouteFlow*

Arsitektur *RouteFlow* terdiri dari tiga bagian, yaitu [3]:

1. *RouteFlow Client* (*RFClient*) merupakan *daemon* pada *Virtual Machine* (VM), menjalankan IP *routing engines* Quagga, mendeteksi perubahan informasi *routing* dan mengirimkan informasi *routing* terbaru kepada *RouteFlow Server*.
2. *RouteFlow Server* (*RFServer*) mengatur VM pada *RFClient*, terhubung dan memberi instruksi ke *RouteFlow Proxy* kapan harus melakukan konfigurasi.
3. *RouteFlow Proxy* (*RFProxy*) merupakan aplikasi kontroler POX *OpenFlow* yang bertanggung jawab terhadap *OpenFlow switch*
   melalui protokol *OpenFlow*. *RFProxy* menginformasikan tentang peristiwa yang terjadi dalam jaringan kepada *RFServer*.

Arsitektur *RouteFlow* dapat dilihat pada Gambar 3.



Gambar 3. Arsitektur *RouteFlow* [4]

2.3.2 Mekanisme kerja *routeFlow*

Pada *RouteFlow*, administrator jaringan diharuskan untuk menginformasikan *RouteFlow* tentang pemetaan jaringan fisik dan jaringan *virtual* yang diinginkan. Konfigurasi tersebut dimuat dan disimpan dalam *database* terpusat.

Ketika terdapat *switch* yang akan bergabung dalam jaringan fisik, maka *switch* mengirim *OpenFlow message* ke *RFProxy*. *RFProxy* menginformasikan setiap *port* fisik *OpenFlow switch* ke *RFServer* dan *RFServer* akan mendaftarkan *port* tersebut. Ketika *RFClient* sudah berjalan, *RFClient* akan menginformasikan *RFServer* tentang setiap *port* dari *virtual switch* dan *RFServer* akan mendaftarkan *port* tersebut.

Quagga routing engine berdasarkan konfigurasi protokol *routing* dalam jaringan *virtual* menghasilkan informasi *routing* yang dikumpulkan oleh *RFClient* kemudian dikirimkan ke *RFServer*. Sebelum trafik tersebut diteruskan ke *data plane*, maka harus terbentuk hubungan antara *data plane* dengan *virtual switch* terlebih dahulu.

Setelah terjadi hubungan antara *data plane* dengan *virtual switch*, *RFServer* meminta *RFClient* untuk men-*trigger* sebuah pesan melalui *virtual switch* yang terhubung dengan *RFProxy*. Sehingga *RFProxy* akan mengetahui koneksi antara *RFClient* dengan *virtual switch* dan menginformasikan *RFServer*. *RFServer* akan menginstruksikan *RFProxy* untuk meneruskan semua trafik yang datang dari VM ke *OpenFlow switch* yang terhubung dengannya dan mengkonfigurasi *OpenFlow switch* menggunakan perintah *OpenFlow* [4].

2.4 *Border Gateway Protocol*

Border Gateway Protocol (BGP) merupakan protokol *routing* yang menghubungkan seluruh *Autonomous System* (AS) pada jaringan internet. AS merupakan sekumpulan *device* jaringan yang berada di bawah administrasi dan strategi *routing* yang sama. BGP memiliki kemampuan melakukan pengumpulan rute, pertukaran rute, dan menentukan rute terbaik menuju ke sebuah lokasi dalam jaringan [5]. BGP menggunakan algoritma *routing distance vector*, dengan setiap *node* hanya mempunyai informasi *hop* berikutnya, proses pertukaran pesan dilakukan secara periodik dengan *node* yang terhubung langsung, pesan tersebut berisi informasi biaya suatu rute, dan penerimaan pesan tersebut men-*trigger* perhitungan ulang dari tabel *routing*. BGP melakukan *path vector routing*, yang merupakan proses penentuan rute berdasarkan jalur terbaik dan terpilih, yang didapat dari *router* BGP lainnya.

BGP termasuk dalam kategori protokol *routing* jenis *Exterior Gateway Protocol* (EGP), dengan *router* dapat melakukan pertukaran rute dari dan ke luar jaringan lokal AS. Berdasarkan tipe pertukaran informasi *routing*, BGP dibagi menjadi dua, yaitu *external* BGP (eBGP) dan *internal* BGP (iBGP). eBGP merupakan sebuah sesi BGP yang terjadi antara dua *router* atau lebih yang memiliki AS yang berbeda, sedangkan iBGP merupakan sesi BGP antara dua *router* atau lebih dalam AS yang sama. Sesi eBGP terjadi pada *router* yang letaknya berada di perbatasan antar AS.

Pada proses pertukaran informasi, protokol iBGP akan memperkenalkan jaringan mana saja yang akan dikenalkan ke AS lain, kemudian protokol eBGP yang akan menyampaikan pesan tersebut sehingga jaringan pada AS lain akan menerima pesan tersebut dan akan menyampaikan ulang pesan tersebut kepada jaringan-jaringan yang dimilikinya. Protokol *routing* BGP dapat dikatakan bekerja pada sebuah *router*, jika sudah terbentuk sesi komunikasi dengan *router* tetangga yang juga menjalankan BGP. Setelah terjalin komunikasi ini, maka kedua buah *router* BGP dapat saling bertukar informasi rute [5].

2.5 *Route Reflector*

Route reflector adalah *router* BGP yang diperbolehkan untuk melanggar aturan dengan melakukan *routing update* yang diperoleh dari iBGP *peer* ke iBGP *peer* yang lain dalam kondisi tertentu, untuk membuat topologi *full-mesh* [6].

2.6 *Open Shortest Path First*

Open Shortest Path First (OSPF) merupakan protokol *routing* yang berada dalam satu *Autonomous System*, termasuk dalam kategori *routing* protokol jenis *Interior Gateway Protocol* (IGP). OSPF menggunakan algoritma *routing link state*, dengan setiap *router* memiliki peta yang lengkap dari topologi, dan jika terdapat *router* yang mati atau terjadi perubahan jaringan, maka setiap *router* dapat menghitung ulang rute baru.

Link state menggunakan metrik *cost* dalam perhitungan untuk memilih jalur dengan melihat nilai *cost* terkecil. *Cost* tersebut bergantung pada *bandwidth* sesuai jenis *interface* atau medium yang dilalui oleh paket, yang berbanding terbalik dengan nilai *cost*, seperti pada persamaan (1) [7]. Pada penelitian ini, jenis *interface* yang digunakan adalah *ethernet*, maka *interface bandwidth* bernilai 10 *megabits per second* (Mbps) dan *cost* akan bernilai 10 [7].

$$cost = \frac{10^8}{Interface\ bandwidth} \quad \text{.....} \quad (1)$$

2.7 *Administrative Distance*

Administrative Distance (AD) adalah kriteria pertama yang digunakan oleh *router* untuk menentukan protokol *routing* mana yang akan digunakan, jika terdapat dua protokol *routing* yang menyediakan informasi *routing* untuk satu tujuan yang sama. Semakin kecil nilai AD, maka protokol *routing* tersebut akan semakin dipercaya untuk digunakan. Nilai *default* AD beberapa protokol *routing* terdapat pada Tabel 1.

Tabel 1. Nilai *default* AD [8]

Route Source	***Default Distance Value***
connected interface	0
eksternal BGP	20
OSPF	110
internal BGP	200

3 PERANCANGAN SIMULASI

Simulasi BGP pada teknologi SDN dilakukan dengan menggunakan emulator jaringan, yaitu Mininet versi 2.0.0. Aplikasi kontroler dilakukan dengan menggunakan *RouteFlow* dengan spesifikasi protokol *OpenFlow* versi 1.0. Mininet dan *RouteFlow* dijalankan pada *Oracle* VM *VirtualBox Manager* versi 4.0.4 r70112 dengan sistem operasi *virtual Ubuntu* 12.04.5 LTS (*Precise Pangolin*) 32-bit.

3.1 Konfigurasi Jaringan Virtual

Pada penelitian ini digunakan delapan buah *switch* sehingga akan dilakukan konfigurasi terhadap delapan buah *RouteFlow Virtual Machine* (RFVM) yang bekerja untuk *OpenFlow virtual switch*. RFVM, dinamakan rfvmAA sampai dengan rfvmHH.

3.2 Konfigurasi Jaringan Fisik

Konfigurasi jaringan fisik menggunakan *file* topologi Mininet dengan membuat topologi jaringan yang terdiri dari satu buah kontroler, delapan buah *OpenFlow switch*, enam buah *host*, enam *link* antara *switch* dengan *host*, dan 11 *link* antar-*switch*, juga termasuk pengalamatan *host*. Konfigurasi topologi pada penelitian ini menggunakan nama *file* ospfebgp1.py yang ditulis dalam bahasa pemrograman *python* dan pengalamatan *host* dalam *file ipconf7*. *File* topologi Mininet dijalankan setelah menjalankan *RouteFlow*, dengan menuliskan perintah berikut pada *Terminal*:

```
$ sudo mn –custom mininet/custom/ospfebgp1.py --topo=rftest7 --
controller=remote,ip=192.168.56.1,port=6633 --pre=ipconf7
```

Contoh berikut ini merupakan penulisan *file* topologi Mininet ospfebgp1.py dalam pembuatan *host*, *switch*, dan *link*:

```
h2 = self.addHost("h2", ip="172.31.6.100/24",
defaultRoute="gw 172.31.6.1")

sE = self.addSwitch("s1")
self.addLink(h2, sF)
self.addLink(sE, sF)
```

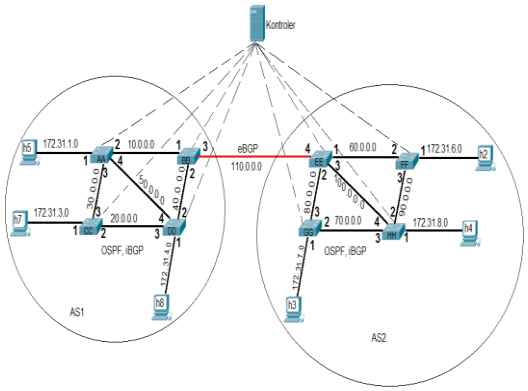
Contoh penulisan *file* pengalamatan ipconf7 terhadap h2 dan h3 adalah sebagai berikut:

h2 ifconfig h2-eth0 172.31.6.100 netmask 255.255.255.0
h3 ifconfig h3-eth0 172.31.7.100 netmask 255.255.255.0
h2 route add default gw 172.31.6.1
h3 route add default gw 172.31.7.1

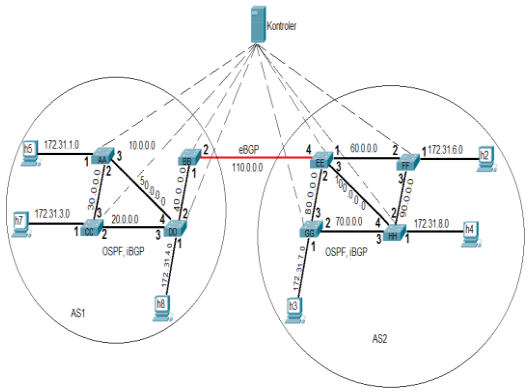
3.3 Skenario Pengujian

Skenario topologi yang digunakan pada penelitian ini terdiri atas delapan buah *OpenFlow switch* untuk merepresentasikan dua bagian nomor AS yang berbeda. Setiap nomor AS memiliki empat buah *OpenFlow switch*. Setiap *OpenFlow switch* terhubung dengan kontroler *RouteFlow* dengan konfigurasi protokol *routing* OSPF dan iBGP, dan satu buah *OpenFlow switch* dari setiap nomor AS dengan protokol *routing* eBGP yang menghubungkan dua nomor AS yang berbeda. Tiga buah *OpenFlow switch* lainnya di setiap AS, akan terhubung langsung ke satu buah *host*. Skenario topologi1 dapat dilihat pada Gambar 4.

Skenario topologi 2 juga dijalankan dengan jumlah *device* yang sama, dengan kondisi jaringan 10.0.0.0, yaitu *link* antara *switchAA* dan *switchBB* terputus. Skenario topologi2 ditunjukkan pada Gambar 5.



Gambar 4. Skenario topologi1



Gambar 5. Skenario topologi2

3.3.1 Simulasi Pengujian Konektivitas

Pada penelitian ini, simulasi dilakukan untuk mengecek konektivitas setiap *node*. Pengecekan konektivitas dilakukan dengan menggunakan perintah *pingall* pada Mininet. Simulasi ini menggunakan topologi1.

3.3.2 Simulasi Pengujian Cara Kerja Protokol *Routing* BGP

Pada penelitian ini, simulasi dilakukan untuk melihat cara kerja protokol *routing* BGP jika ada *link* yang putus. Pemutusan *link* dilakukan pada *link* dari jalur sumber h5 pada AS1 ke tujuan h4 pada AS2, yaitu antara *link switchAA* dengan *link switchBB.* Simulasi ini menggunakan topologi1 dan topologi2.

3.3.3 Simulasi Pengujian Kinerja Protokol *Routing* BGP

Pada penelitian ini, simulasi dilakukan untuk menguji kinerja protokol *routing* BGP dengan menggunakan *iperf*. Pada simulasi dibangkitkan trafik UDP dari *client* ke *server*, dengan h4 sebagai *server* dan h5 sebagai *client.* Simulasi dilakukan dengan menjalankan perintah *iperf* berikut pada Mininet:

>h4 iperf –s –u &
>h5 iperf –c h4 –u –i1

Simulasi ini dilakukan dengan menggunakan topologi1.

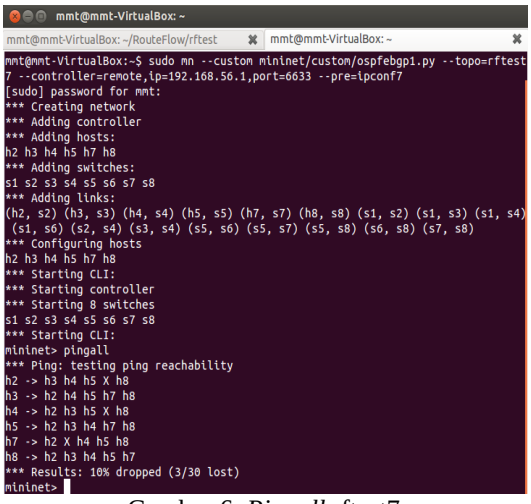
Indikator kinerja yang diuji adalah:

1. *Jitter*
   Jitter adalah perbedaan selang waktu kedatangan antarpaket di terminal tujuan. *Jitter* diukur dalam satuan detik.
2. *Throughput*
   Throughput adalah jumlah data yang berhasil dikirimkan per satuan waktu. *Throughput* diukur dalam satuan bit per detik.

4 HASIL SIMULASI

4.1 Hasil Simulasi Pengujian Konektivitas

Hasil simulasi pengujian konektivitas menggunakan topologi1 dengan perintah *pingall*, yaitu melakukan *ping* antar-*host* pada Mininet seperti pada Gambar 6, menunjukkan hasil pengiriman *ping* berhasil, walaupun terdapat kegagalan sebesar 10%, karena terdapat paket yang hanya berhasil terkirim sebagian, yaitu ketika h2 *ping* ke h6, h4 *ping* ke h7, dan h7 *ping* ke h2. Keberhasilan pengiriman paket ping adalah 90%, sehingga dapat disimpulkan semua *node* saling terhubung.



Gambar 6. *Pingall* rftest7

4.2 Hasil Simulasi Pengujian Cara Kerja Protokol *Routing* BGP

Hasil simulasi pengujian cara kerja protokol *routing* BGP dari sumber h5 ke tujuan h4 adalah sebagai berikut:

a. Sebelum Diputus

Pada topologi1, h5 terhubung langsung dengan *switchAA*, sehingga dilakukan pengecekan informasi *routing* pada *switchAA*. Setiap pengecekan diawali dengan *login* pada setiap *rfvm.*

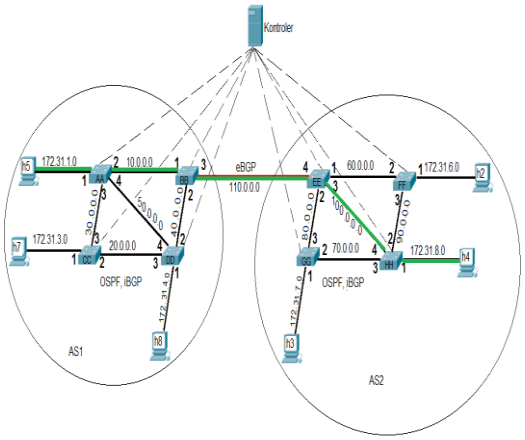
Jalur terbaik menuju 172.31.8.0 dari *switchAA* adalah 10.0.0.2 dengan informasi jaringan 172.31.8.0 didapat dari protokol *routing* iBGP dan pemilihan jalur terbaik dalam AS oleh protokol *routing* OSPF. 10.0.0.2 merupakan *interface eth1 switchBB*, sehingga selanjutnya dilakukan pengecekan pada *switchBB.*

Jalur terbaik menuju 172.31.8.0 dari *switchBB* adalah 110.0.0.5 dengan informasi jaringan dan jalur terbaik didapat dari protokol *routing* eBGP. 110.0.0.5 merupakan *interface eth4 switchEE*, sehingga selanjutnya dilakukan pengecekan pada *switchEE*.

Jalur terbaik menuju 172.31.8.0 dari *switchEE* adalah 100.0.0.8 dengan informasi jaringan dan pemilihan jalur terbaik oleh protokol *routing* OSPF. 100.0.0.8 merupakan *interface* eth4 *switchHH*, sehingga selanjutnya dilakukan pengecekan pada *switchHH*.

Pada *switchHH*, jaringan 172.31.8.0 langsung terhubung dengan *switchHH* melalui *interface* eth1, sehingga merupakan jalur terbaik mencapai tujuan h4.

Hasil simulasi pengujian cara kerja protokol *routing* BGP sebelum *link* diputus dapat dilihat pada Gambar 7, dengan jalur terbaik dari sumber h5 menuju *switchAA*, ke *switchBB*, kemudian ke *switchEE*, *switchHH*, dan sampai ke tujuan h4.



Gambar 7 Pemilihan jalur topologi1

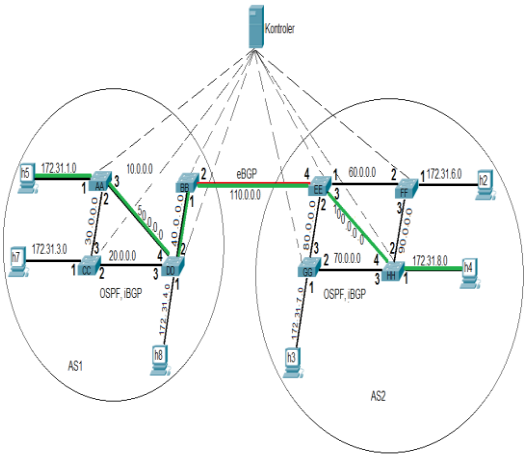
b. Setelah Diputus

Pada topologi2, yaitu setelah pemutusan *link* antara *switchAA* dengan *switchBB*, pengecekan awal juga dilakukan pada *switchAA*. Jalur terbaik menuju 172.31.8.0 dari *switchAA* adalah 50.0.0.4 dengan informasi jaringan 172.31.8.0 didapat dari protokol *routing* iBGP dan pemilihan jalur terbaik dalam AS oleh protokol *routing* OSPF. 50.0.0.4 merupakan *interface eth4 switchDD*, sehingga selanjutnya dilakukan pengecekan pada *switchDD*.

Jalur terbaik menuju 172.31.8.0 dari *switchDD* adalah 40.0.0.2 dengan informasi jaringan 172.31.8.0 didapat dari protokol *routing* iBGP dan pemilihan jalur terbaik dalam AS oleh protokol *routing* OSPF. 40.0.0.2 merupakan *interface eth1 switchBB*, sehingga selanjutnya dilakukan pengecekan pada *switchBB.*

Pengecekan pada *switchBB* untuk mencapai 172.31.8.0 adalah sama dengan pengujian pada topologi1, yaitu rute terbaik menuju *switchEE*, kemudian *switchHH*, dan mencapai tujuan h4.

Hasil simulasi pengujian cara kerja protokol *routing* BGP setelah *link* diputus dapat dilihat pada Gambar 8, dengan jalur terbaik dari sumber h5 menuju *switchAA*, ke *switchDD*, *switchBB*, kemudian ke *switchEE*, *switchHH*, dan sampai ke tujuan h4.



Gambar 8 Pemilihan jalur topologi2

4.3 Hasil Simulasi Pengujian Kinerja Protokol *Routing* BGP

4.3.1 *Jitter*

Dari simulasi yang dijalankan selama 9,9 sekon, diperoleh nilai *jitter* sebesar 0,44 milisekon. Nilai *jitter* dapat dilihat pada Gambar 9.

4.3.2 *Throughput*

Dari simulasi yang dijalankan selama 9,9 sekon, diperoleh nilai *packet loss* sebesar 0,22%, sehingga nilai *throughput* adalah 100% - 0,22% = 99,78%. Nilai *packet loss* dapat dilihat pada Gambar 9.

```
mininet> h5 iperf -c h4 -u -i1
------------------------------------------------------------
Client connecting to 172.31.8.100, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 172.31.1.100 port 36497 connected with 172.31.8.100 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   125 KBytes  1.02 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec
[  3] Sent 889 datagrams
[  3] Server Report:
[  3]  0.0- 9.9 sec  1.25 MBytes  1.06 Mbits/sec   0.440 ms    2/  889 (0.22%)
[  3]  0.0- 9.9 sec  2 datagrams received out-of-order
mininet>
```

Gambar 9 Nilai *jitter* dan *packet loss*

5 KESIMPULAN

Kesimpulan yang diperoleh dari hasil simulasi ini adalah:

1. Protokol *routing* akan mencari jalur yang lain jika terdapat *link* yang putus.
2. *Host* di AS1 dapat mengirimkan trafik UDP ke *host* di AS2 dengan *jitter* sebesar 0.44 milisekon dan *throughput* sebesar 99.78%.

REFERENSI

[1] Momaru, B., Copacio, F., Lazar, G., Dobrota, V. 2003. "Practical Analysis of TCP Implementations: Tahoe, Reno, New Reno". Technical University od Cluj-Napoca.
[2] Murty, R., Monuj. 2004. "Ad Hoc Wireless Networks: Architectures and Protocols". Prentice Hall.
[3] Chen, X., Zhai, H., Wang, J., Fang, Y. 2005. "A Survey on Improving TCP Performance over Wireless Networks". Dept. of Electrical and Computer Engineering. University of Florida.
[4] University of Sorthern California. 1981. "RFC 793-Transmission Control Protocol". http://www.faqs.org/rfcs/rfc793.html.
[5] Dixit, S., Prasad, R. 2003. "Wireless IP and Building the Mobile Internet". Universal Personal Communication.
[6] Akyildiz, I., Wang, X., Wang, W. 2005. "Wireless Mesh Networks: a survey". Computer Networks 47.
[7] Cisco Networking Academy Program. "Module 10: Intermediate TCP/IP".
[8] Allman, M., Paxson, V., Stevens, W. 1999. "RFC2581-TCP Congestion Control". http://www.faqs.org/rfcs/rfc2581.html.