ELKOMIKA | ISSN (p): 2338-8323 | ISSN (e): 2459-9638 | Vol. 7 | No. 3 | Halaman 585 – 599
DOI : http://dx.doi.org/10.26760/elkomika.v7i3.585 September 2019

Analisis Kinerja *RouteFlow* pada Jaringan SDN (*Software Defined Network*) menggunakan Topologi *Full-Mesh*

KUKUH NUGROHO, DHIMAS PRABOWO SETYANUGROHO

S1 Teknik Telekomunikasi Fakultas Teknik Telekomunikasi dan Elektro
Institut Teknologi Telkom Purwokerto
Email: kukuh@ittelkom-pwt.ac.id

Received 22 Juli 2019 **|** *Revised* 5 Agustus 2019 **|** *Accepted* 3 September 2019

ABSTRAK

Perangkat controller dalam jaringan SDN berfungsi untuk memberikan informasi tabel flow ke perangkat switch sebagai acuan dalam menentukan informasi rute. Kecepatan dalam memproses permintaan tabel flow yang dilakukan oleh perangkat switch tergantung dari pemilihan jenis controller yang digunakan. Terdapat beberapa jenis controller yang bisa digunakan untuk mengatur aliran trafik data dalam jaringan SDN. Pada penelitian ini akan digunakan jenis controller RouteFlow dengan menggunakan algoritma routing Djikstra. Jaringan uji menggunakan topologi Full-Mesh yang menyediakan koneksi penuh ke semua switch, dimana terdapat dua protokol layer transport yang digunakan untuk mengirimkan data dalam jaringan SDN yaitu TCP dan UDP. Hasil pengukuran kualitas jaringan menunjukkan bahwa penggunaan protokol UDP menghasilkan nilai delay, jitter, dan throughput yang lebih baik dibandingkan dengan TCP. Waktu konvergensi jaringan yang dihasilkan sebesar 12,18 ms ketika ukuran data yang dipertukarkan sebesar 64 Byte.

***Kata kunci**: Controller, Full-Mesh, RouteFlow, Software Defined Network*

ABSTRACT

The controller device in the SDN network is applied to provide flow table information for all switches as a reference in determining route information. The speed in processing the flow table requests that are made by the switches depends on the choice of the type of controller used. There are several types of controllers that can be used to manage the data traffic flow in the SDN network. This research utilizes the RouteFlow controller and the Djikstra routing algorithm. The test network applies the Full-Mesh topology that provides fully connections to all switches, wherein two transport protocols are used to transmit data in SDN networks, i.e., TCP and UDP. The results of network performance measurements show that the use of the UDP obtains better values in delay, jitter, and throughput than TCP. The network convergence time is 12.18 ms when the size of data exchanged is set to 64 Bytes.

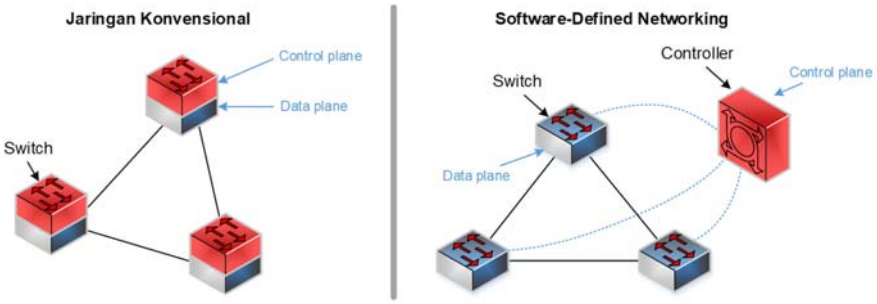
***Keywords**: Controller, Full-Mesh, RouteFlow, Software Defined Network*

1. PENDAHULUAN

Pertumbuhan pengguna Internet di Indonesia mengakibatkan meningkatnya penggunaan trafik data. Berdasarkan data APJII tahun 2016 **(APJII, 2016)**, pengguna Internet telah mencapai 132,7 juta orang dari total 256,2 juta populasi penduduk di Indonesia. Penetrasi pengguna Internet pada tahun tersebut adalah sebanyak 52% dari total jumlah populasi penduduk. Jumlah pengguna Internet meningkat sebesar 7,9% pada tahun 2017. Jumlah pengguna Internet yang meningkat ini berdampak pula pada tingkat kompleksitas jaringan Internet itu sendiri. Jaringan harus mampu menampung jumlah pengguna yang besar dan memberikan layanan data yang optimal. Kebutuhan dalam meningkatkan kapasitas *bandwidth* saluran tidak bisa dihindari, agar kualitas jaringan tetap optimal dengan pertambahan jumlah pengguna Internet yang besar.

Jaringan komputer menggunakan konsep keterkaitan konektifitas antar perangkat. Data harus bisa dikirimkan antar perangkat dalam jaringan. Definisi jaringan juga tidak terbatas hanya dalam satu lingkup wilayah saja. Misalnya untuk jaringan WAN yang secara wilayah geografis berada antar kota atau bahkan antar negara yang berbeda, dan konfigurasi jaringan harus dilakukan pada masing-masing perangkat tidak terpusat pada satu perangkat. Hal ini dapat menimbulkan masalah dengan cakupan jaringan yang luas disertai dengan jumah perangkat yang banyak. Apabila terdapat masalah pada satu atau beberapa perangkat, maka komunikasi antar perangkat dalam jaringan tidak bisa dilakukan. Diperlukan waktu tunggu sampai jaringan dalam kondisi layak digunakan. Dalam artian, antar perangkat ujung bisa saling berkomunikasi. Solusi yang bisa digunakan adalah dengan menggunakan metode konfigurasi perangkat secara terpusat **(Lara, Kolasani, & Ramamurthy, 2014)**. Konsep ini terdapat pada penerapan jaringan SDN (*Software Defined Network*). Dengan menerapkan konsep konfigurasi jaringan secara terpusat, waktu penanganan gangguan jaringan menjadi lebih cepat dan proses konfigurasi perangkat menjadi lebih fleksibel **(Hu, Hao, & Bao, 2014)**.

Konsep yang diberikan pada penerapan jaringan SDN adalah memisahkan antara bagian fungsi *control* dan *forwarding* yang terdapat pada bagian *data plane* **(Kreutz et al., 2015)**. Pada konsep jaringan konvensional, fungsi *control* dan *forwarding* berada dalam satu perangkat yang sama. Perbedaan mendasar antara konsep jaringan konvensional dengan jaringan SDN terdapat pada keterangan Gambar 1.



Gambar 1. Perbedaan jaringan konvensional dengan SDN

Jaringan SDN memberikan solusi konfigurasi jaringan secara terpusat dengan cara memisahkan fungsi *control* dalam sebuah perangkat jaringan. Salah satu contoh dari fungsi *control* dalam sebuah perangkat adalah fungsi *routing* yang terdapat dalam perangkat router. Dalam konsep jaringan SDN, perangkat *switch* tidak bisa melakukan fungsi *routing* melainkan hanya bisa melakukan fungsi *forwarding* atau meneruskan paket ke perangkat tujuan sampai paket tersebut diterima oleh perangkat tujuan. Fungsi *control* sepenuhnya dilakukan secara

terpusat oleh sebuah perangkat yang dinamakan sebagai *controller*. Terdapat berbagai macam jenis *controller* yang bisa digunakan seperti NOX, yang merupakan jenis *controller* yang pertama kali digunakan, POX, *Beacon*, dan *Floodlight* **(Putra, Pramukantoro, & Yahya, 2018)**. Pada penelitian ini lebih difokuskan pada penggunaan jenis *controller RouteFlow* yang digunakan untuk mengatur arah aliran trafik data pada jaringan SDN. Perangkat *controller* akan bertugas dalam menemukan topologi jaringan SDN yang sedang digunakan dan menentukan rute terbaik ke semua alamat *network* tujuan.

Teknologi jaringan SDN memberikan solusi dalam mempermudah proses konfigurasi perangkat. Sistem konfigurasi yang terpusat pada perangkat *controller* menjadikan waktu konfigurasi perangkat lebih cepat. Perangkat *switch* dalam jaringan SDN akan menerima informasi rute berupa tabel *flow* dari *controller*. Tabel tersebut akan digunakan oleh perangkat *switch* dalam melakukan fungsi *routing*. Selain fungsi routing, terdapat fungsi *security* **(Yao & Yan, 2016)** atau *load balancing* **(Liao, Kuai, & Lu, 2016)** yang juga bisa diterapkan pada jaringan SDN. Tingkat kualitas jaringan juga dipengaruhi oleh pemilihan jenis *controller* yang tepat. Diperlukan uji laboratorium agar diketahui perbandingan tingkat kualitas jaringan terhadap penggunaan jenis *controller* tertentu. Beberapa uji perbandingan jenis *controller* telah dilakukan diantaranya dengan membandingkan *controller Floodlight*, *Ryu*, ONOS, *Maestro*, dan POX **(Putra et al., 2018)**. Parameter yang diujikan dalam menguji kelima jenis *controller* tersebut adalah *throughput* dan *latency*. *Controller Maestro* menghasilkan kualitas jaringan yang lebih baik, dimana dihasilkan nilai *latency* sebesar 500 sampai 4000 *flow*/s dan *latency* sebesar 1000 sampai 5000 ms. Uji kualitas jaringan SDN dengan menggunakan jenis *controller* tertentu juga bisa disertakan dengan mengaktifkan aplikasi *routing*. Misalnya dengan menguji penggunaan *controller* Ryu dengan menggunakan algoritma *routing Floyd-Warshall* **(Saputra, M., & Hertiana, 2017)**. Pengujian dilakukan dengan mengirimkan aliran paket data, *voice* dan *video*. Parameter uji adalah dengan melihat nilai *overhead* dari trafik ketika semua aplikasi dijalankan pada jaringan SDN dan menguji parameter waktu konvergensi jaringan. Dari hasil uji menunjukkan bahwa pengaruh penambahan *overhead* trafik dipengaruhi oleh adanya aktifitas pertukaran paket yang berisi informasi rute (*flow table*) yang dilakukan oleh perangkat *controller*. Waktu konvergensi jaringan memiliki nilai rata-rata 17,7 detik. Implementasi jaringan SDN lebih banyak menggunakan teknologi *hypervisor* **(Drutskoy, Keller, & Rexford, 2013)**, penggunaan komputer virtual dalam komputer fisik. Terdapat solusi lain yang bisa digunakan yaitu dengan menggunakan teknologi *container* **(Djomi, Munadi, & Negara, 2018)**. Perbandingan antara kedua teknologi tersebut menghasilkan kualitas jaringan SDN yang lebih baik ketika digunakan teknologi *container* dibandingkan dengan teknologi *hypervisor* dilihat dari parameter *delay*, *packet loss* dan CPU *utilization*. Dengan mengimplementasikan layanan FTP dihasilkan niai *delay* sebesar 0,12 ms dan 6,21 ms untuk layanan *video streaming*. Kinerja CPU masih dibawah 1% ketika digunakan teknologi *Docker container*.

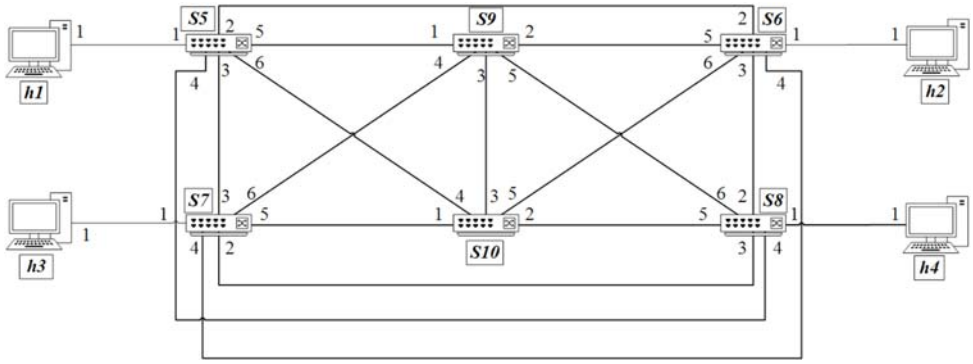
2. METODOLOGI

Pada penelitian ini dilakukan uji kualitas jaringan SDN dengan menggunakan jenis *controller RouteFlow* dan topologi jaringan SDN yang digunakan adalah *full-mesh*. Penggunaan *controller RouteFlow* digunakan untuk melakukan fungsi khusus yaitu untuk menyediakan layanan *routing* dalam jaringan SDN **(Nunes, Mendonca, Nguyen, Obraczka, & Turletti, 2014)**. Pada masing-masing *switch* nantinya akan menggunakan mekanisme *routing* dinamis dengan mengaktifkan protokol *routing* OSPF. Dalam konsep SDN, cara mengaktifkan protokol *routing* OSPF dimulai pada sisi perangkat *controller* yang terdapat pada bagian *control plane* kemudian akan diteruskan ke perangkat *switch* (*data plane*). Oleh karena itu, *RouteFlow* perlu diaktikan pada sisi perangkat *controller* tersebut. Dengan menggunakan *controller RouteFlow*, jalur data

yang digunakan untuk merutekan paket akan dibuat secara otomatis dan tersimpan dalam tabel *flow* dalam perangkat *switch*. Pemilihan topologi jaringan *full-mesh* yang digunakan untuk proses uji kualitas jaringan SDN lebih ditekankan pada adanya konsep *fault-tolerance* yang memberikan jaminan komunikasi antara dua perangkat ketika jalur utama mengalami masalah. Penggunaan topologi *full-mesh* juga akan menyediakan jaringan dengan tingkat *availability* yang tinggi. Tahapan penelitian dimulai dari proses perancangan topologi jaringan yang digunakan untuk proses uji sistem, dilanjutkan dengan penentuan skenario pengujian jaringan.

2.1. *Perancangan Topologi Jaringan*

Tahap perancangan dimulai dengan menentukan topologi jaringan yang digunakan untuk menguji kinerja *controller RouteFlow* pada jaringan SDN. Topologi jaringan yang digunakan untuk proses mengujian sistem adalah *full-mesh* yang menghubungkan semua perangkat *switch* seperti yang terlihat pada Gambar 2. Antar perangkat *switch* mempunyai alternatif jalur cadangan ke semua perangkat *switch* dalam jaringan.



Gambar 2. Topologi jaringan uji

Jumlah perangkat *switch* yang digunakan untuk proses pengujian jaringan SDN dengan menggunakan *controller RouteFlow* adalah sebanyak enam buah *switch*, empat komputer (*host*) yaitu h1, h2, h3, h4 dan satu buah perangkat *controller*. Konsep pembangunan jaringan dengan menggunakan topologi *full-mesh* ini akan menyediakan tingkat ketersediaan jaringan yang tinggi **(Kurniawan, NNurfajar, Dwi, & Yunan, 2017)**. Apabila ada salah satu jalur yang putus atau bermasalah, maka komunikasi antar *end-user* masih bisa dilakukan.

2.2. *Skenario Pengujian Jaringan*

Tahapan penelitian berikutnya adalah merancang skenario pengujian jaringan. Pada pengujian skenario pertama, komputer h1 dan h4 melakukan proses pertukaran data dengan tanpa adanya trafik data dari komputer lain yang menggukaan jaringan. Proses pertukaran data di mulai dari komputer h1 ke h4 dan tidak sebaliknya. Pada pengujian skenario kedua, proses pertukaran data antara komputer h1 dan h4 dibebani dengan adanya trafik data dari komputer lain yaitu proses pertukaran data antara komputer h2 dan h3. Trafik data yang dibangkitkan oleh kedua komputer tersebut dinamakan juga dengan istilah *background traffic*.

Dalam mengukur kualitas jaringan SDN, dimana parameter yang diukur adalah *delay*, *throughput*, dan *jitter*, terdapat perangkat lunak tambahan yang digunakan yaitu D-ITG (*Distributed Internet Traffic Generator*). Dua komputer yang saling mempertukarkan data akan diaktifkan perangkat lunak ini, baik di sisi pengirim maupun penerima. Sedangkan untuk mengukur kualitas jaringan ketika digunakan protokol *routing* OSPF, parameter pengujian yang digunakan adalah waktu konvergensi jaringan.

```
h1 :/d-itg/bin # ./ITGSend -a <ip address h4> -T <protokol> -C <jumlah paket per detik>
-c <besar paket (byte)> -t <waktu pengiriman (ms)> -x <nama log file>.log

h4 :/d-itg/bin # ./ITGRecv
```

Gambar 3. Konfigurasi D-ITG di Sisi Pengirim dan Penerima

Pada skenario pengujian jaringan, pertama kali komputer h1 dan h4 akan saling mempertukarkan data dengan ukuran yang bervariasi, dimulai 4,8 Mbyte, 6,4 Mbyte, 8 Mbyte, 9,6 Mbyte, 11,2 Mbyte dan 12,8 Mbyte. Protokol di *layer transport* yang digunakan untuk mengirimkan data juga dibuat berbeda yaitu TCP (*Transmission Control Protocol*) dan UDP (*User Datagram Protocol*). Pada saat mengirimkan data, perangkat lunak D-ITG dijalankan pada kedua komputer tersebut. Pada contoh kasus ini, komputer h1 sebagai komputer pengirim dan h4 sebagai komputer penerima. Perintah dalam mengkonfigurasi D-ITG baik pada sisi komputer pengirim dan penerima terlihat seperti pada keterangan Gambar 3. Kualitas jaringan akan diamati selama waktu pengiriman data. Hasil dari pengukuran akan tersimpan dalam *log file* di dalam komputer h1. Selanjutnya dari hasil pengukuran akan dilakukan analisa kualitas jaringan SDN pada saat diimplementasikan *controller RouteFlow*. Pengujian akan dilakukan untuk melihat pengaruh perubahan ukuran data terhadap kualitas jaringan SDN dengan menggunakan protokol di *layer transport* yang berbeda. Penggunaan protokol TCP pada saat pengujian digunakan untuk merepresentasikan komunikasi data yang sifatnya *real-time* dan UDP digunakan untuk merepresentasikan komunikasi data yang sifatnya *unreal-time*.

Jaringan SDN yang akan diujikan mengimplementasikan algoritma *routing* Djikstra. Salah satu protokol routing yang menggunakan algoritma tersebut adalah OSPF (*Open Shortest Path First*). Protokol *routing* OSPF menggunakan acuan *bandwidth* dalam menentukan jalur terbaik ke semua alamat *network* tujuan. Pada skenario perancangan topologi jaringan SDN awal seperti yang terlihat pada keterangan Gambar 2, masing-masing *interface switch* menggunakan acuan *bandwidth* saluran sebesar 10 Mbps. Dalam mengimplementasikan protokol *routing* OSPF dengan menggunakan *controller RouteFlow*, terdapat satu perangkat lunak yang akan digunakan yaitu Quagga. Perangkat lunak ini difungsikan untuk mengaktifkan protokol *routing* dalam jaringan SDN, seperti RIP, OSPF, atau IS-IS. Pada proses pengujian jaringan SDN ini digunakan protokol *routing* OSPF.

Berdasarkan keterangan Gambar topologi perencanaan jaringan SDN pada Gambar 2, trafik data yang dibangkitkan adalah berasal dari komputer h1 dan h4. Terdapat dua komputer yang difungsikan sebagai pembeban trafik dalam jaringan SDN yaitu komputer h2 dan h3. Istilah dari trafik data ini adalah trafik *background*. Adanya trafik yang dipertukarkan antara komputer h2 dan h3 mengakibatkan trafik jaringan menjadi lebih terbebani ketika komputer h1 dan h4 melakukan proses pertukaran data. Komputer h2 dan h3 juga akan melakukan proses pertukaran data dengan ukuran data yang bervariasi, dimulai dari 33 Mbyte, 66 Mbyte, 99 Mbyte, 132 Mbyte, 165 Mbyte dan 198 Mbyte. Perangkat lunak yang digunakan untuk membangkitkan trafik data tambahan pada penelitian ini adalah *iperf*. Konsep dalam menjalankan aplikasi *iperf* menggunakan sistem *client* dan *server*. Pada saat melakukan pengujian, komputer h2 akan difungsikan sebagai *iperf server* dan h3 sebagai *iperf client*. Konfigurasi *iperf* disisi komputer h2 dan h3 terlihat pada keterangan Gambar 4.

```
h2 :/ # iperf -s -u -i <interval waktu pengiriman>

h3 :/ # iperf -u -c <ip address server> -i <interval waktu pengiriman>
-t <waktu pengiriman> -b <besar bandwidth>
```

Gambar 4. Konfigurasi *Iperf* Disisi Komputer *Client* dan *Server*

Pembangkitan trafik tambahan (*background traffic*) juga menggunakan skenario penggunaan protokol *layer transport* yang berbeda yaitu TCP dan UDP. Ukuran data yang digunakan sebagai pembeban trafik dibuat bervariasi, sedangkan komunikasi antara komputer h1 dan h4 sebagai komputer utama menggunakan ukuran data yang tetap yaitu sebesar 12,8 Mbyte.

Jaringan SDN yang sudah dibuat dengan menggunakan bantuan perangkat lunak Mininet akan diuji kualitas jaringan tersebut untuk mengetahui tingkat kehandalan jaringan dalam menangani aliran trafik data yang mengalir pada jaringan tersebut. Salah satu parameter yang digunakan untuk menguji tingkat kehandalan jaringan SDN yang sudah dibangun adalah dengan menggunakan parameter nilai *delay*. Besaran nilai waktu yang dibutuhkan paket untuk bisa sampai ke perangkat penerima akan dinilai. Persamaan yang digunakan untuk menghitung besaran nilai *delay* adalah sebagai berikut **(Manual et al., 2013)** :

$$\sigma = \sqrt{\frac{1}{N}(d_i - d)^2} \quad (1)$$

Nilai *delay* yang dihasilkan merupakan nilai rata-rata *delay* dari *delay* waktu pengiriman masing-masing paket. Konstanta N pada Persamaan (1) di atas menandakan jumlah total paket yang dikirimkan dari perangkat pengirim ke penerima. Selain penghitung parameter *delay*, kualitas jaringan juga akan dihitung dengan menggunakan parameter nilai *jitter*.

$$Jitter = \frac{\sum_{i=1}^{n} |D_i|}{N} \quad (2)$$

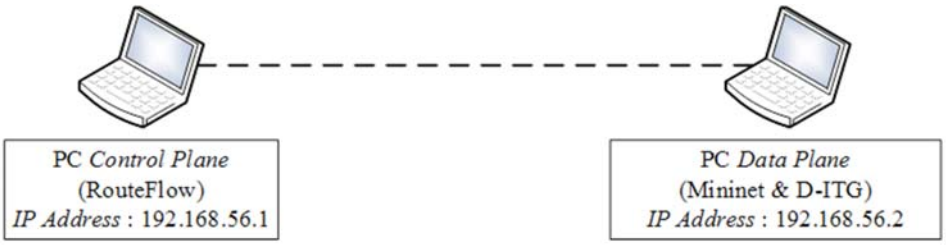
Jitter merupakan variasi *delay* antar pengiriman paket. Konstanta D pada Persamaan (2) merupakan selisih waktu antara pengiriman paket ke-i dengan paket ke (i-1) atau paket sebelumnya. Nilai rata-rata dari selisih total antar pengiriman paket tersebut merupakan hasil dalam menentukan nilai *jitter*. Selain *delay* dan *jitter*, nilai *throughput* juga digunakan sebagai parameter dalam mengukur kualitas jaringan SDN.

$$Throughput = \frac{\sum paket\ yang\ diterima\ (bit)}{waktu\ pengamatan} \quad (3)$$

Ketiga parameter yang digunakan untuk mengukur kualitas jaringan SDN dilakukan dengan menggunakan perangkat lunak D-ITG. Perangkat lunak tersebut akan diaktifkan baik pada sisi komputer pengirim maupun penerima.

2.3. *Perancangan Sistem Jaringan*

Konsep perancangan sistem jaringan hanya menggunakan dua buah komputer yang saling terhubung dengan menggunakan kabel UTP. Terdapat salah satu komputer yang akan difungsikan sebagai perangkat *controller* dan komputer yang lain digunakan sebagai komputer dalam membuat jaringan SDN. Skenario dari perancangan sistem jaringan terlihat pada keterangan Gambar 5 berikut.



Gambar 5. Topologi Sistem

Sesuai dengan keterangan Gambar 5, terlihat satu komputer yang akan diaktifkan jenis *controller RouteFlow* dengan menggunakan alamat IP 192.168.56.1. Komputer tersebut disimulasikan akan mengatur jaringan SDN yang akan dibuat pada komputer lain yang akan difungsikan sebagai bagian *data plane*. Bagian ini dalam sistem jaringan SDN akan menyediakan fungsi jaringan. Perangkat-perangkat *switch* akan dihubungkan dan dibuat pada bagian ini. Pada skenario topologi sistem jaringan komputer yang akan difungsikan untuk membuat jaringan SDN menggunakan alamat IP 192.168.56.2. Kedua komputer tersebut terhubung secara langsung dengan menggunakan kabel UTP. Dalam membuat jaringan SDN pada komputer yang difungsikan sebagai bagian data plane akan digunakan perangkat lunak Mininet dan sebagai pembangkit trafik data dalam jaringan SDN adalah D-ITG.

Perancangan topologi jaringan SDN yang akan diujikan, diimplementasikan dalam perangkat lunak Mininet. Penggunaan Mininet merupakan solusi sebelum mengimplementasikan sebuah *controller* dalam jaringan SDN nyata. Mininet merupakan solusi untuk mensimulasikan jaringan SDN. Sistem simulasi yang digunakan untuk menguji jaringan SDN menggunakan bantuan dua perangkat komputer yang terhubung langsung dengan menggunakan kabel UTP seperti yang terlihat pada keterangan Gambar 5. Salah satu komputer difungsikan hanya untuk menjalankan perangkat lunak Mininet. Pada komputer tersebut, perancangan topologi jaringan uji seperti yang terlihat pada keterangan Gambar 2 diimplementasikan. Topologi jaringan yang sudah dibuat dihubungkan dengan perangkat *controller* yang terpisah secara fisik, tidak menyatu pada komputer yang dijalankan perangkat lunak Mininet. Jenis *controller* yang digunakan untuk proses uji sistem adalah *RouteFlow*. Sistem operasi yang digunakan baik untuk menjalankan Mininet maupun *controller RouteFlow* adalah Ubuntu 12.04.

Pada keterangan Gambar 2, komputer H1 dan H4 adalah komputer utama yang melakukan proses pertukaran data. Dengan menggunakan dua komputer tersebut, kualitas jaringan SDN sesuai dengan perancangan topologi jaringan awal akan diujikan dengan menggunakan parameter QoS (*Quality of Service*); *throughput*, *delay*, *jitter*, dan waktu konvergensi jaringan. Peran dari komputer H2 dan H3 adalah sebagai pemberi atau pembangkit trafik tambahan (*background trafik*) dalam jaringan. Selama terjadinya proses komunikasi, tahapan pengujian jaringan SDN dilakukan untuk mengetahui kualitas jaringan tersebut. Pengujian yang dilakukan pada penelitian ini menggunakan dua skenario yaitu dengan dan tanpa adanya trafik *background* atau trafik dari komputer lain yang menggunakan jaringan selain komputer utama yang sedang melakukan proses komunikasi.

3. HASIL PENGUJIAN

Percobaan pengujian jaringan SDN seperti yang terlihat pada keterangan Gambar 2 bertujuan untuk menganalisa pengaruh perubahan ukuran paket terhadap kualitas jaringan SDN baik itu dengan dan tanpa adanya trafik tambahan (*background traffic*) dalam jaringan. Parameter untuk mengukur kualitas jaringan SDN yang digunakan diantaranya adalah *delay*, *jitter* dan *throughput*. Selain itu terdapat parameter tambahan untuk mengukur kualitas kinerja jaringan penggunaan protokol *routing* OSPF yaitu dengan melakukan pengukuran waktu konvergensi jaringan.

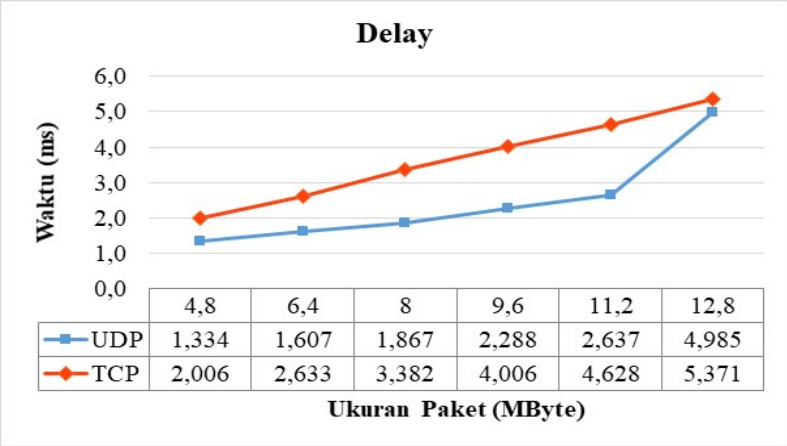
3.1 Pengukuran Parameter *Delay*

Waktu paket dikirimkan dari perangkat pengirim sampai ke penerima akan dicatat untuk melihat kualitas jaringan SDN dilihat dari sisi parameter nilai *delay*. Persamaan untuk menghitung nilai *delay* terdapat pada Persamaan (1). Hasil dari pengukuran nilai *delay* kemudian akan dilihat apakah nilai tersebut masih dalam kategori baik, sedang, atau buruk menurut standarisasi yang diberikan ITU-T G.114 **(Systems, 2003)**.

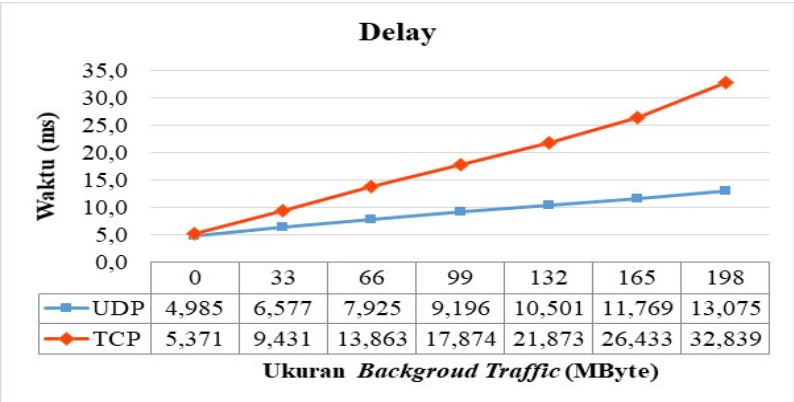
Tabel 1. Standarisasi Nilai *Delay* dari ITU-T G.114

Delay	***Category***	***Delay***
(Latency)	*Good*	0-150 ms
Standard	*Medium*	150-400 ms
	Poor	>400 ms

Perhitungan nilai *delay* difokuskan pada paket yang dipertukarkan antara komputer h1 dan h4. Besaran ukuran paket yang dipertukarkan juga bervariasi, dimulai dari ukuran paket 4,8 Mbyte sampai 12,8 Mbyte. Kemudian proses komunikasi antara komputer h1 dan h4 akan disimulasikan dengan memberikan beban tambahan pada jaringan SDN. Komputer h2 dan h3 adalah komputer yang akan memberikan beban dengan cara saling mempertukarkan paket dengan ukuran yang bervariasi, dimulai dari ukuran paket 33 Mbyte sampai 198 Mbyte.



Gambar 6. Hasil Pengukuran Nilai *Delay* Tanpa *Background Traffic*



Gambar 7. Hasil Pengukuran Nilai *Delay* Dengan *Background Traffic*

Hasil pengukuran nilai *delay* ketika komputer h1 mengirim paket ke h4 dengan variasi nilai dari besaran ukuran paket yang dikirimkan dari komputer h1 ke h4 terlihat pada keterangan Gambar 6. Hasil pengukuran kualitas jaringan SDN dengan menggunakan parameter *delay*, *jitter*, *throughput* dan waktu konvergensi jaringan merupakan nilai rata-rata dari tiga puluh kali percobaan pengukuran. Dari hasil pengukuran terlihat bahwa pengiriman paket dengan menggunakan protokol di *layer transport* TCP mempunyai nilai yang lebih besar dibandingkan

dengan penggunaan protokol UDP. Dengan kata lain, paket yang dikirimkan oleh komputer h1 akan lebih cepat sampai ke komputer h4 jika digunakan protokol UDP dibandingkan dengan penggunaan protokol TCP. Proses pengiriman paket yang dikirimkan oleh komputer h1 menggunakan asumsi bahwa informasi rute yang menuju ke alamat network yang digunakan oleh komputer h4 sudah tersimpan dalam tabel *flow* pada perangkat *switch* S5. Sebelum bisa melakukan proses *forwarding*, *switch* S5 akan menggunakan jalur yang aman dengan menggunakan bantuan protokol TSL (*Transport Layer Security*) menuju ke perangkat *controller* untuk menanyakan informasi rute menuju ke komputer h4. Misalnya pada saat mengirimkan paket dengan ukuran 4,8 Mbyte, penggunaan protokol UDP menghasilkan nilai *delay* sebesar 1,334 ms, sedangkan jika menggunakan protokol TCP nilai *delay* paket yang dihasilkan adalah sebesar 2,006 ms. Dengan menambah ukuran paket yang dikirimkan dari komputer h1 ke h4, terlihat tren nilai *delay* juga semakin besar. Ukuran dari *header* TCP relatif lebih besar daripada UDP, sehingga paket yang dikirimkan dengan menggunakan protokol TCP akan menghasilkan nilai *delay* yang relatif lebih besar jika dibandingkan dengan UDP jika protokol tersebut digunakan untuk mengirimkan data ke perangkat penerima. Besar ukuran *header* TCP adalah sebesar 20 Byte, berbeda dengan besar ukuran *header* UDP yaitu sebesar 8 Byte.

Pada saat melakukan pengukuran dengan menggunakan tambahan trafik dalam jaringan (*background traffic*), besaran ukuran data yang dipertukarkan antara komputer utama yaitu h1 dan h4 dibuat tetap yaitu sebesar 12,8 Byte. Hasil dari pengukuran nilai *delay* dengan menambah trafik data dari komputer lain selain komputer h1 dan h4 terlihat pada keterangan Gambar 7. Selain komputer h1 dan h4 aktif dalam melakukan proses pertukaran data, komputer h2 dan h3 juga ikut serta menggunakan jaringan. Paket data juga dipertukarkan oleh kedua komputer tersebut. Ukuran *header* dari paket yang menggunakan protokol UDP lebih kecil dibandingkan dengan TCP. Hal ini menyebabkan paket yang dikirimkan dengan menggunakan protokol UDP menghasilkan nilai *delay* yang lebih kecil jika dibandingkan ketika menggunakan protokol di *layer transport* TCP. Hal ini bisa dilihat dari hasil pengukuran nilai *delay* pada keterangan Gambar 7. Dengan menambah ukuran trafik tambahan dalam jaringan, nilai *delay* yang dihasilkan juga semakin besar. Misalnya ketika menggunakan trafik tambahan ke dalam jaringan SDN sebesar 198 Mbyte, nilai *delay* yang dihasilkan dengan menggunakan protokol TCP yatu sebesar 32,84 ms. Namun, hasil pengukuran nilai *delay* tersebut masih dalam kategori baik menurut acuan standar ITU-T G.114 tentang nilai *delay*.

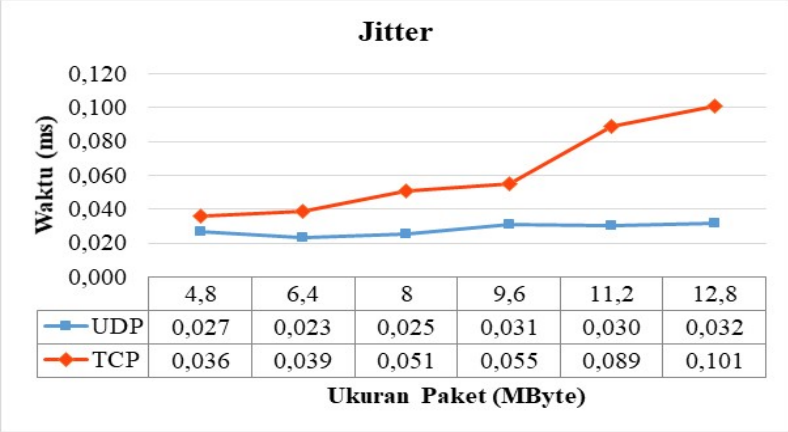
3.2 Pengukuran Parameter *Jitter*

Hasil pengukuran nilai *delay* akan berhubungan erat dengan nilai *jitter*. Jika *delay* adalah waktu yang dibutuhkan paket untuk bisa sampai ke perangkat penerima, maka *jitter* adalah interval waktu antar kedatangan paket. Pengukuran nilai *jitter* dilakukan dengan menggunakan pendekatan nilai rata-rata, dimana Persamaan (2) adalah rumus yang digunakan untuk menghitung nilai *jitter* dengan menggunakan perangkat lunak D-ITG. Seperti pada pengukuran nilai *delay*, hasil dari pengukuran nilai *jitter* akan dibandingkan dengan standarisasi nilai *jitter* terkait kualitas jaringan yang dikeluarkan oleh ITU-T G.144.

Tabel 2. Standarisasi Nilai *Jitter* dari ITU-T G.114

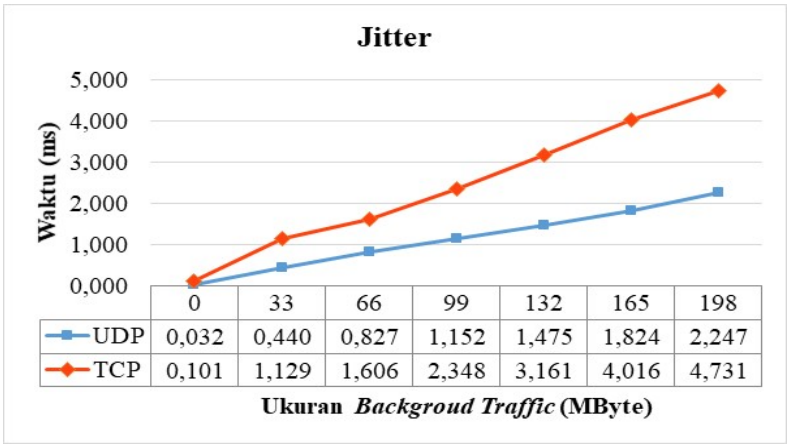
Jitter	***Category***	***Jitter***
Standard	*Good*	0-20 ms
	Medium	20-50 ms
	Poor	>50 ms

Menurut standarisasi nilai *jitter* yang dikeluarkan oleh ITU-T G.114, hasil pengukuran nilai *jitter* dikatakan bagus direntang angka 0 sampai 20 ms. Nilai tersebut yang akan dijadikan acuan hasil pengukuran kualitas jaringan SDN dengan menerapkan *controller RouteFlow*. Pengukuran nilai *jitter* dilakukan dengan menggunakan dan tanpa menggunakan skenario pembebanan trafik dalam jaringan.



	4,8	6,4	8	9,6	11,2	12,8
UDP	0,027	0,023	0,025	0,031	0,030	0,032
TCP	0,036	0,039	0,051	0,055	0,089	0,101

Gambar 8. Hasil Pengukuran Nilai *Jitter* Tanpa *Background Traffic*



	0	33	66	99	132	165	198
UDP	0,032	0,440	0,827	1,152	1,475	1,824	2,247
TCP	0,101	1,129	1,606	2,348	3,161	4,016	4,731

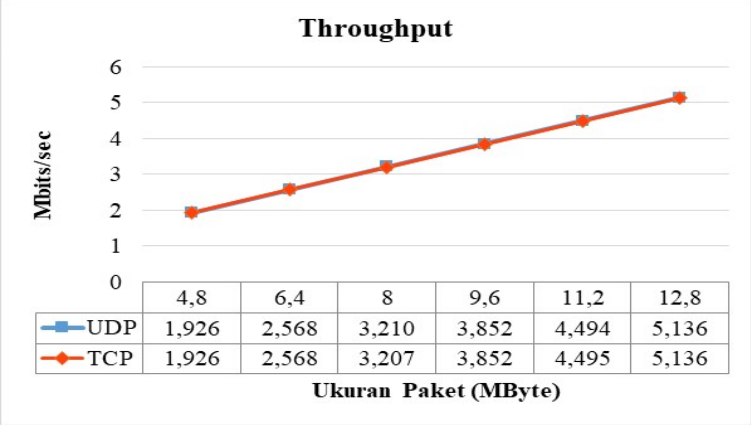
Gambar 9. Hasil Pengukuran Nilai *Jitter* Dengan *Background Traffic*

Hasil pengukuran nilai *jitter* seperti yang terlihat pada keterangan Gambar 8 dan 9 memperlihatkan pola hasil yang sama seperti pada pengukuran nilai *delay*. Penggunaan protokol di *layer transport* TCP menghasilkan nilai *jitter* yang relatif lebih besar dibandingkan ketika digunakan protokol UDP. Pola tren nilai *jitter* dengan memperbesar ukuran data yang dipertukarkan antara komputer h1 dan h4 cenderung naik. Pada penggunan ukuran data maksimal yaitu sebesar 12,8 Mbyte, penggunaan protokol TCP menghasilkan nilai *jitter* sebesar 0,101 ms, sedangkan pada penggunaan protokol UDP menghasilkan nilai *jitter* sebesar 0,032 ms. Hasil nilai tersebut masih dalam kategori baik menurut standarisasi nilai *jitter* yang dikeluarkan oleh ITU-T G.114. Begitupula ketika menambahkan beban trafik data (*background traffic*) ke dalam jaringan SDN seperti yang diperlihatkan pada keterangan Gambar 9. Dengan membuat besaran ukuran data yang dipertukarkan antara komputer h1 dan h4 tetap yaitu sebesar 12,8 Mbyte, kemudian menambahkan beban trafik data yang dilakukan oleh komputer h2 dan h3 sampai penggunaan ukuran data sebesar 198 Mbyte, terdapat lonjakan kenaikan nilai *jitter* yang cukup besar. Pada penggunaan protokol TCP menghasilkan nilai *jitter* sebesar

4,731 ms, sedangkan pada penggunaan protokol UDP menghasilkan nilai *jitter* sebesar 2,247 ms. Perbedaan ukuran *header* yang digunakan oleh protokol UDP dan TCP memperlihatkan adanya perbedaan hasil pengukuran nilai *jitter*. Misalnya pada saat melakukan pengukuran dengan tanpa adanya beban trafik tambahan dalam jaringan. Penggunaan protokol UDP menghasilkan nilai *jitter* 215,62% lebih kecil dibandingkan dengan penggunaan protokol TCP.

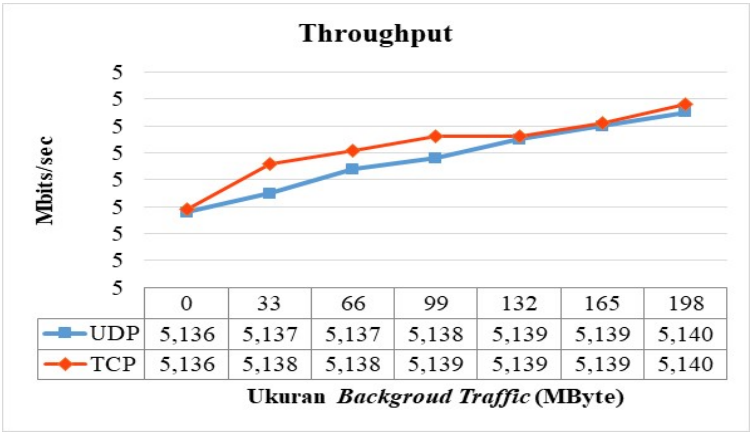
3.3 Pengukuran Parameter *Throughput*

Selain menggunakan parameter *delay* dan *jitter*, kualitas jaringan SDN juga diukur dengan menggunakan parameter nilai *throughput*. Pengukuran parameter *throughput* dilakukan untuk mengetahui besaran *bandwidth* sebenarnya yang digunakan oleh komputer untuk mengirimkan data. Nilai *bandwidth* saluran yang digunakan pada jaringan adalah sebesar 10 Mbps. Namun, nilai *throughput* yang diterima oleh masing-masing komputer biasanya tidak sebesar nilai *bandwidth* saluran. Besaran nilai *throughput* akan tergantung dengan jumlah komputer yang menggunakan jaringan atau besaran trafik data yang mengalir dalam jaringan **(Nugroho & Fallah, 2018)**. Pengukuran nilai *throughput* juga dibandingkan dengan dan tanpa adanya penggunaan tambahan trafik data.



	4,8	6,4	8	9,6	11,2	12,8
UDP	1,926	2,568	3,210	3,852	4,494	5,136
TCP	1,926	2,568	3,207	3,852	4,495	5,136

Gambar 10. Hasil Pengukuran Nilai *Throughput* Tanpa *Background Traffic*



	0	33	66	99	132	165	198
UDP	5,136	5,137	5,137	5,138	5,139	5,139	5,140
TCP	5,136	5,138	5,138	5,139	5,139	5,139	5,140

Gambar 11. Hasil Pengukuran Nilai *Throughput* Dengan *Background Traffic*

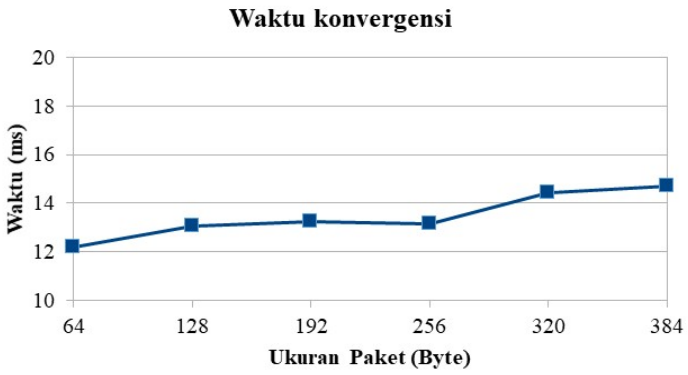
Pengukuran nilai *throughput* juga dilakukan dengan menggunakan bantuan perangkat lunak D-ITG. Titik pengukuran dilakukan pada komputer h1 sebagai komputer pengirim dan h4 sebagai komputer penerima. Perangkat lunak D-ITG diaktifkan pada kedua komputer tersebut.

Namun perintah yang digunakan agak sedikit berbeda untuk komputer pengirim dan penerima seperti yang terlihat pada keterangan Gambar 3. Persamaan (3) digunakan sebagai rumus untuk menghitung nilai *throughput* yang dihasilkan oleh perangkat lunak D-ITG. Hasil pengukuran nilai *throughput* pada saat komputer h1 dan h4 saling mempertukarkan data dengan ukuran yang bervariasi terlihat pada keterangan Gambar 10. Dari hasil pengukuran terlihat bahwa baik itu pada saat menggunakan protokol TCP maupun UDP sebagai protokol di *layer transport* yang digunakan untuk mengirimkan paket dihasilkan nilai pengukuran yang relatif sama. Seperti pada saat komputer h1 mengirimkan data dengan ukuran 4,8 Mbyte. Hasil yang didapatkan baik itu dengan menggunakan protokol TCP maupun UDP sama yaitu sebesar 1,926 Mbps. Besaran nilai *throughput* akan bertambah besar dengan menaikkan ukuran paket yang dipertukarkan antar dua komputer. Hal ini membuktikan bahwa ukuran *header* yang dimiliki oleh kedua protokol tersebut tidak mempengarahui besaran nilai *throughput* yang dihasilkan.

Pengujian skenario berikutnya adalah dengan menambahkan trafik tambahan ke dalam jaringan SDN. Komputer h2 dan h3 dipaksakan untuk mempertukarkan data dengan ukuran yang bervariasi. Perangkat lunak *iperf* diaktifkan baik pada komputer h2 sebagai pengirim dan h3 sebagai penerima. Peran dari perangkat lunak tersebut adalah untuk membangkitkan trafik data. Hasil dari pengukuran nilai *throughput* dengan menambahkan trafik data ke dalam jaringan SDN terlihat pada keterangan Gambar 11. Dengan menambahkan trafik tambahan ke dalam jaringan terlihat bahwa penggunaan protokol TCP menghasilkan nilai *throughput* yang relatif lebih besar dibandingkan dengan penggunaan protokol UDP. Pada pengujian ini terlihat bahwa ukuran *header* TCP yang lebih besar dibandingkan dengan UDP mempengaruhi nilai *throughput*. Dengan menggunakan ukuran *header* yang lebih besar, secara tidak langsung akan mempengarahui beban trafik dalam jaringan. Dari penjelasan Gambar 11 terlihat bahwa penggunaan protokol TCP memberikan pembebanan trafik yang relatif lebih besar dibandingkan UDP. Hal ini memberikan dampak pada pengujian nilai *throughput*. Misalnya pada melakukan pembebanan jaringan sebesar 33 Mbyte, 66 Mbyte, dan 99 Mbyte, penggunaan protokol TCP memberikan nilai *throughput* 0,019% lebih besar dibandingkan ketika menggunakan protokol UDP.

3.4 Pengukuran Waktu Konvergensi

Pengukuran waktu konvergensi ditujukan untuk mengetahui kualitas protokol *routing* OSPF dalam menangani perubahan topologi jaringan ketika jalur utama mengalami masalah. Pengukuran waktu konvergensi dilakukan di dalam jaringan, sedangkan pengukuran terhadap parameter *delay*, *jitter* dan *throughput* dilakukan pada sisi perangkat akhir (*end-device*). Skenario yang dilakukan pada saat pengujian waktu konvergensi adalah dengan memutuskan jalur utama yang digunakan oleh komputer h1 untuk mengirimkan paket ke komputer h4. Jalur utama yang digunakan sesuai skenario topologi jaringan uji yang terlihat pada keterangan Gambar 2 adalah h1->s5->s4->h4. Ketika ada paket yang dikirimkan oleh komputer h1, paket tersebut akan masuk ke *switch* s5 lewat port 1 dan dikeluarkan ke port 4. Sebelum diteruskan ke komputer tujuan, paket tersebut akan masuk ke *switch* s4 lewat port 4 dan akan dikeluarkan lewat port 1 menuju ke komputer h4. Hasil dari pengukuran waktu konvergensi terlihat pada keterangan Gambar 12.



Gambar 12. Hasil Pengukuran Waktu Konvergensi

Hasil pengukuran waktu konvergensi dilakukan ketika komputer h1 mengirimkan paket ke h4, kemudian port 4 di *switch* s5 yang dijadikan sebagai jalur utama diputus. Ketika terjadi perubahan topologi jaringan, protokol *routing* OSPF yang teraktifkan pada masing-masing *switch* akan bekerja dalam menghimpun informasi rute ke semua *switch* tetangga. *Switch* s5 akan menanyakan ke semua *switch* tetangga tentang jalur terbaik untuk menuju ke komputer h4. Setelah jalur utama diputus, jalur yang digunakan oleh *switch* s5 untuk menuju ke komputer h4 berubah menjadi h1->s5->s6->s8->h4. Nomor port yang digunakan oleh *switch* s5 untuk menuju ke *switch* s6 adalah port 2. Dari *switch* s6, paket akan diteruskan ke *switch* s8 melalui port 3. Dari *switch* s8, paket akan diteruskan lewat port 1 menuju ke komputer h4. Waktu peralihan dari awal port 4 di *switch* s5 diputus sampai bisa digunakan port 2 untuk meneruskan paket menuju ke komputer h4 dijadikan acuan dalam menghitung waktu konvergensi jaringan. Gambar 12 memperlihatkan hasil pengukuran waktu konvergensi dengan variasi besaran ukuran paket yang dikirimkan dari komputer h1 menuju ke h4. Dari hasil pengukuran terlihat bahwa dengan memperbesar ukuran data yang dikirimkan ke komputer h4, waktu konvergensi jaringan yang dihasilkan juga relatif membesar (naik). Misalnya ketika ukuran data yang dikirimkan sebesar 64 Byte, waktu konvergensi jaringan yang dihasilkan sebesar 12,18 ms. Akan tetapi ketika ukuran data diperbesar menjadi 384 Byte, besaran waktu konvergensi jaringan yang dihasilkan naik menjadi 14,71 ms.

4. KESIMPULAN

Hasil pengujian kualitas jaringan SDN dengan menggunakan *controller RouteFlow* dan protokol *routing* OSPF masih dalam kategori baik menurut standarisasi yang dikeluarkan oleh ITU-T G.114 baik itu jika dilihat dari sisi parameter nilai *delay* maupun *jitter*. Penggunaan protokol di *layer transport* UDP menghasilkan nilai *delay* dan *jitter* yang lebih baik dibandingkan dengan TCP. Peningkatan ukuran data yang dipertukarkan dalam jaringan juga akan berbanding lurus dengan hasil pengukuran nilai *delay* dan *jitter*. Dengan menaikkan ukuran data, nilai *delay* dan *jitter* yang dihasilkan juga semakin besar. Begitupula dengan tren hasil pengukuran nilai *throughput*. Penambahan beban trafik data dalam jaringan relatif tidak mengubah hasil pengukuran nilai *throughput* yaitu di angka nilai rata-rata 5,138 ms baik pada penggunaan protokol di *layer transport* UDP maupun TCP. Selain kedua parameter tersebut, hasil pengukuran terhadap kualitas protokol *routing* OSPF juga dilakukan dengan menggunakan parameter waktu konvergensi jaringan. Dari hasil pengukuran terlihat bahwa waktu yang dibutuhkan agar jaringan dapat dikatakan dalam keadaan konvergen adalah sebesar 12,18 ms ketika ukuran paket yang dipertukarkan sebesar 64 Byte. Besaran waktu konvergensi tersebut naik sebesar 20,77% ketika ukuran paket yang dipertukarkan dibuat menjadi 384 Byte.

DAFTAR RUJUKAN

APJII. (2016). Penetrasi dan Perilaku Pengguna Internet Indonesia. Retrieved May 14, 2019, from https://apjii.or.id/downfile/file/surveipenetrasiinternet2016.pdf

Djomi, M. I., Munadi, R., & Negara, R. M. (2018). Analisis Kualitas Layanan FTP danVideo Streaming berbasis Network Function Virtualization menggunakan Docker Containers. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *6*(2), 180. https://doi.org/10.26760/elkomika.v6i2.180

Drutskoy, D., Keller, E., & Rexford, J. (2013). Scalable network virtualization in software-defined networks. *IEEE Internet Computing*, *17*(2), 20–27. https://doi.org/10.1109/MIC.2012.144

Hu, F., Hao, Q., & Bao, K. (2014). A Survey on Software-Defined Network and OpenFlow : From Concept to Implementation. *IEEE Communications Surveys & Tutorials*, *16*(4), 2181–2206. https://doi.org/10.1109/COMST.2014.2326417

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Kurniawan, M. T., Nurfajar, A., Dwi, O., & Yunan, U. (2017). Desain Topologi Jaringan Kabel Nirkabel PDII-LIPI dengan Cisco Three-*Layer*ed Hierarchical menggunakan NDLC. *Jurnal Elkomika*, *4*(1). https://doi.org/10.26760/elkomika.v4i1.47

Lara, A., Kolasani, A., & Ramamurthy, B. (2014). Network innovation using open flow: A survey. *IEEE Communications Surveys and Tutorials*, *16*(1), 493–512. https://doi.org/10.1109/SURV.2013.081313.00105

Liao, W. H., Kuai, S. C., & Lu, C. H. (2016). Dynamic load-balancing mechanism for software-defined networking. *Proceedings - 2016 International Conference on Networking and Network Applications, NaNA 2016*, 336–341. https://doi.org/10.1109/NaNA.2016.66

Manual, D., Botta, A., Donato, W. De, Dainotti, A., Avallone, S., & Pescap, A. (2013). *D-ITG 2.8.1 Manual*. 1–35.

Nugroho, K., & Fallah, M. S. (2018). Implementasi Load Balancing menggunakan Teknologi EtherChannel pada Jaringan LAN. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *6*(3), 420. https://doi.org/10.26760/elkomika.v6i3.420

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, *16*(3), 1617–1634.

https://doi.org/10.1109/SURV.2014.012214.00180

Putra, M. W., Pramukantoro, E. S., & Yahya, W. (2018). Analisis Perbandingan Kualitas Kontroller Floodlight , Maestro , RYU , POX Dan ONOS Dalam Arsitektur Software Defined Network (SDN). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(10), 3779–3787.

Saputra, I. A., M., R. R., & Hertiana, S. N. (2017). Uji Kualitas Algoritma Floyd-Warshall Pada Jaringan Software Defined Network (SDN). *Jurnal Elektronika Dan Telekomunikasi*, *16*(2), 52. https://doi.org/10.14203/jet.v16.52-58

Systems, D. (2003). G.114 (05/2003). *Networks*.

Yao, Z., & Yan, Z. (2016). Security in software-defined-networking: A survey. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10066 LNCS*(4), 319–322. https://doi.org/10.1007/978-3-319-49148-6_27