Terbit *online* pada laman web jurnal : http://innovatics.unsil.ac.id

Innovation in Research of Informatics (INNOVATICS)

| ISSN (Online) 2656-8993 |

Analisis Performa Segment Routing dan Reactive Routing pada Software-defined Networking

Lutfi Kiki Fuadi [1], Nur Widiyasono[2]

[1,2]*Jurusan Informatika, Universitas Siliwangi, Jl. Siliwangi No.24 Tasikmalaya, Indonesia*
[1]*157006077@student.unsil.ac.id,* [2]*nur.widiyasono@unsil.ac.id*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 31-12-2021
Revisi Akhir: 23-02-2022
Diterbitkan *Online*: 24-02-2022

KATA KUNCI

Mininet,
QoS,
SDN,
Reactive Routing,
Segment Routing,

KORESPONDENSI

Telepon: -

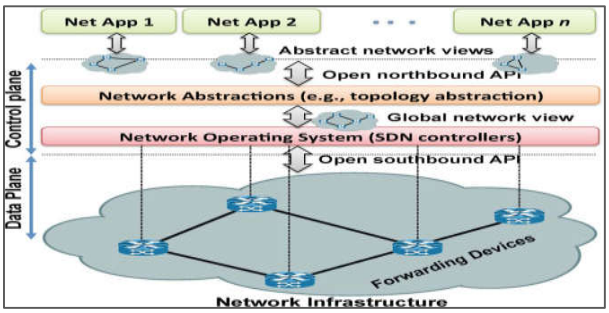
E-mail: 157006077@student.unsil.ac.id

ABSTRACT

Software-defined Networking (SDN) is a new paradigm in computer network management that separates the control plane and data plane. The controller is an important component in an SDN-based computer network. The purpose of this research is to conduct a comparative study of the performance of segment routing and reactive routing in SDN. In this study, ONOS was chosen as the controller used in the experiment because it supports the implementation of segment routing and reactive routing. The experiment of implementing the two routing protocols on an SDN network with a full-mesh and tree (depth 5) network topology is the scenario that is tried in this study. QoS results based on TIPHON parameters (throughput, packet loss, delay and jitter) were recorded from each experiment. The throughput of each experiment was 6.97 MB/s, 5.80 MB/s, 1.65 ms and 2.16 ms, indicating index 4 (very good). Packet loss in the full-mesh topology experiment was 5.16% and 5.53% with index 3 (good), while in tree (depth 5) the results were 2.46% and 2.16% showed index 4 (very good). The delay for each experiment was 2.99 ms, 3.17 ms, 4.65 ms and 5.43 ms, indicating an index of 4 (very good). The jitter for each experiment was 7.10 ms, 7.85 ms, 18.30 ms and 18.13 ms, indicating index 3 (good).

1. PENDAHULUAN

Software-Defined networking (SDN) merupakan arsitektur jaringan yang memisahkan komponen: *data plane, control plane* dan *application plane*[1]. *Northbound interface* memungkinkan *application plane* berinteraksi dengan *control plane* dan *southbound interface* memungkinkan *control plane* dapat berinteraksi dengan *data plane* [2]. SDN merupakan paradigma baru untuk mengelola jaringan agar lebih sederhana dan mengurangi kompleksitas dalam teknologi jaringan [3]. SDN merupakan generasi lanjut dari konsep jaringan canggih yang menyajikan pendekatan baru terhadap arsitektur jaringan komputer dan memungkinkan diterapkan beberapa konsep seperti *advance security*, *monitoring*, dan *high availability* [4]. Arsitektur jaringan komputer ini memisahkan *control plane* dari sebuah perangkat jaringan komputer *(switch atau router)* dengan *data plane* perangkat jaringan komputer tersebut [5]. Pemisahan *data plane* dan *control plane* memungkinkan SDN untuk membuat kontrol terpusat pada seluruh jaringan [6] [7]. Arsitektur ini bersifat *directly programmable*, sehingga menjadikan SDN memiliki karakteristik dinamis, *manageable*, *cost-effective* dan *adaptable* sehingga ideal untuk kebutuhan aplikasi saat ini [8] [9]. *Control Plane* merupakan komponen pada jaringan yang berfungsi untuk mengontrol jaringan, konfigurasi sistem, manajemen jaringan dan menentukan informasi bagian yang bertanggung jawab melakukan *forwarding packet* [10]. Interaksi antara *control plane* dan *data plane* tersebut terjadi pada *southbound interace* dengan implementasi protokol OpenFlow pada kedua sisi interface [11].



Gambar 1. *SDN Architecture and Its Fundamental Abstractions [12]*

Kondisi jaringan yang fluktuatif mengakibatkan suatu jaringan menerima beberapa lalu lintas yang sangat besar sehingga dapat menimbulkan kemacetan lalu lintas data di jaringan jika tidak dikelola dengan tepat. Kemacetan lalu lintas data di jaringan terjadi ketika sebuah jaringan yang harus melayani dan memproses lalu lintas melebihi kapasitas [13]. Pengelolaan dan pemrograman jaringan pada *Software-defined Networking* dilakukan dalam komponen *controller*, sehingga *controller* merupakan komponen sangat penting [14].

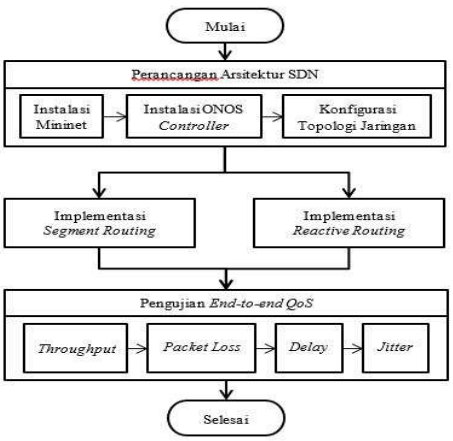
2. ULASAN PENELITIAN TERKAIT

Beberapa SDN controllers yang umum digunakan diantaranya: NOX, Floodlight, Ryu dan ONOS. Aspek yang perlu diperhatikan dalam pemilihan controller diantaranya *productivity* [15]. ONOS (Open Network Operating System) merupakan salah satu *distributed SDN control platform* yang mendukung peningkatan kinerja, skalabilitas dan ketersediaan jaringan [16].

Tujuan dari penelitian ini mengukur performa penerapan *segment routing* dan *reactive routing* pada SDN menggunakan ONOS controller dan diimplementasikan pada topologi *full-mesh* dan *tree (depth 5)* secara bergantian. Hasil pengujian *Quality of Service (QoS)* masing-masing protokol routing dibandingkan dan diukur berdasarkan indeks penilaian standar *Telecommunications* and *Internet Protocol Harmonization Over Networks (TIPHON)*.

3. METODOLOGI

Tahapan yang dilakukan pada penelitian ini terdiri dari perancangan arsitektur SDN, implementasi protokol routing dan penghitungan QoS yang digambarkan pada gambar 2.



Gmbar 2. Tahapan Penelitian

3.1 Perancangan Arsitektur Sistem

a. Instalasi Mininet

Mininet merupakan salah satu emulator SDN yang berjalan pada sistem operasi Linux[17]. Mininet dianggap sebagai solusi paling unggul dalam hal kemudahan penggunaan, performansi, akurasi, skalabilitas dan mampu menyediakan lingkungan yang realistis dan nyaman *(convenience)* dengan harga yang murah *(low cost)* [8]. Prinsip kerja mininet adalah membuat virtual hosts sebagai sebuah proses yang berbasis pada metode virtualisasi dan mekanisme *namespace* jaringan [18]. Perintah untuk instalasi emulator Mininet pada Linux Ubuntu ditampilkan pada gambar 3.

```
$ sudo mkdir -p /opt && cd /opt
$ git clone git://github.com/mininet/mininet
$ cd mininet
$ git tag
$ git checkout -b 2.2.2
$ util/install.sh -sfv
```

Gambar 3. Inslatasi Mininet

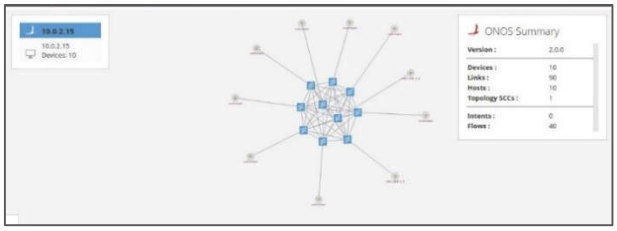
b. Instalasi ONOS Controller

ONOS merupakan sistem operasi yang berbasis pemrograman java [19]. Versi ONOS Controller yang digunakan pada percobaan yaitu 2.0.0 dan dipasang dalam direktori root (/) pada Linux Ubuntu dengan perintah seperti ditampilkan pada gambar 4.

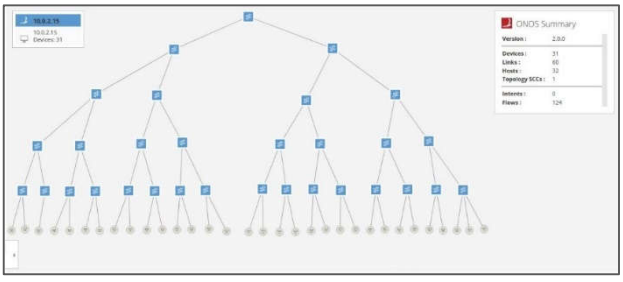
```
$ sudo wget
https://repo1.maven.org/maven2/org/ONOSproject/
  ONOS-releases/2.0.0/ONOS-2.0.0.tar.gz
$ sudo tar xzzf ONOS-2.0.0.tar.gz
$ sudo chown -R sdn:sdn ONOS
```

Gambar 4. Inslatasi ONOS Controller

Desain topologi yang disimulasikan yaitu *topologi full-mesh* dengan 10 switch dan *topologi tree (depth 5)* dengan 31 switch.



Gambar 5. Topologi Full Mesh

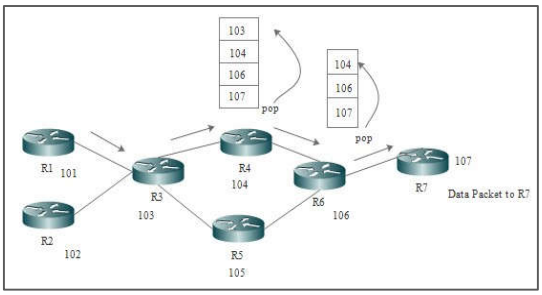


Gambar 6. Topologi Tree (depth 5)

3.2 Implementasi Protocol Routing

a. Segment Routing

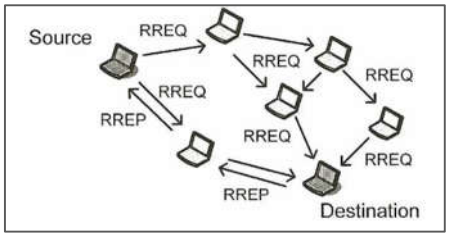
Segment routing menyediakan penerusan paket melalui jalur jaringan dengan menentukan daftar segment [20]. *Segment Routing* pada tampilan *web user interface ONOS Controller* termasuk dalam *traffic engineering* yang dijalankan melalui menu *application*. *Segment routing* menyediakan penerusan paket melalui jalur jaringan dengan menentukan daftar segment [20].



Gambar 7. Segment Routing [20]

b. Reactive Routing

Reactive routing atau *on-demand routing* merupakan suatu protokol *routing* pasif yang dapat digunakan untuk menemukan jalur ketika ada permintaan paket yang perlu dikirim oleh *source address* [21] [22]. *Reactive Routing* pada *web user interface ONOS Controller* termasuk dalam *traffic engineering* yang dijalankan melalui menu application.



Gambar 8. Reactive Routing [23]

3.3 Quality of Service

QoS merupakan teknik pengukuran karakteristik dan kualitas jaringan [24]. Empat parameter dasar QoS diantaranya *bandwidth, packet loss, delay* dan *jitter*.

3.3.1 Throughput

Throughput adalah laju data aktual per satuan waktu atau biasa dikenal sebagai bandwidth dalam kondisi yang sebenarnya. *Throughput* dapat diukur dan dilakukan penilaian terhadap hasil pengukuran tersebut.

TABEL 1. KRITERIA THROUGHPUT STANDAR TIPHON [25]

Kategori *Throughput*	*Throughput (bps)*	Indeks
Sangat Bagus	100	4
Bagus	75	3
Sedang	50	2
Jelek	< 25	1

3.3.2 *Packet Loss*

Packet *Loss* menunjukkan jumlah total paket yang hilang karena *collision* dan *congestion* pada jaringan. Kategori penilaian *Packet Loss* ditunjukkan pada *Tabel 2*.

TABEL 2. KRITERIA PACKET LOSS STANDAR TIPHON [25]

Kategori *Packet Loss*	*Packet Loss (%)*	Indeks
Sangat Bagus	0% – 2%	4
Bagus	3% – 14%	3
Sedang	15% – 24%	2
Jelek	>25 %	1

3.3.3 *Delay*

Delay disebut juga *latency* merupakan waktu yang dibutuhkan paket untuk mencapai tujuan, karena adanya antrian, atau mengambil rute yang lain untuk menghindari kemacetan. Kategori penilaian *Delay* ditunjukkan pada *Tabel 3*.

TABEL 3. KRITERIA DELAY STANDAR TIPHON [25]

Kategori *Delay*	*Delay (%)*	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 – 300 ms	3
Sedang	300 – 450 ms	2
Jelek	> 450 ms	1

3.3.4 *Jitter*

Jitter merupakan *variasi delay* yang terjadi akibat adanya selisih waktu atau interval antar kedatangan paket di sisi penerima. Kategori penilaian *Jitter* ditunjukkan pada *Tabel 4*.

TABEL 4. KRITERIA JITTER STANDAR TIPHON [25]

Kategori *Jitter*	*Jiiter (%)*	Indeks
Sangat Bagus	0 ms	4
Bagus	0 – 75 ms	3
Sedang	76 – 125 ms	2
Jelek	126 – 225 ms	1

4. HASIL DAN PEMBAHASAN

4.1 QoS Segment Routing

4.1.1 *Topologi Full-mesh*

TABEL. 5 QOS SEGMENT ROUTING PADA TOPOLOGI FULL-MESH

Parameter	Hasil Pengukuran	Indeks	Kategori
Throughput	6,97 MB/s	4	Sangat Bagus
Packet Loss	5,16 %	3	Bagus
Delay	2,99 ms	4	Sangat Bagus
Jitter	7,10 ms	3	Bagus

4.2 QoS Reactive Routing

Penerapan *segment routing* pada topologi *full-mesh* menghasilkan throughput dan *delay* dengan kategori sangat bagus, sedangkan *packet loss* dan *jitter* termasuk kategori bagus.

4.2.1 *Topologi Tree (depth 5)*

TABEL 6. QOS REACTIVE ROUTING PADA TOPOLOGI TREE (DEPTH 5)

Parameter	Hasil Pengukuran	Indeks	Kategori
Throughput	1,65 MB/s	4	Sangat Bagus
Packet Loss	2,45 %	4	Sangat Bagus
Delay	4,65 ms	4	Sangat Bagus
Jitter	18,30 ms	3	Bagus

Protokol *reactive routing* pada topologi *tree (depth 5)* menghasilkan *throughput, packet loss* dan *delay* dengan kategori sangat bagus, sedangkan *jitter* kategori bagus.

4.3 *QoS Reactive Routing*

4.3.1 *Topologi Full-mesh*

TABEL 7. QOS REACTIVE ROUTING PADA TOPOLOGI FULL-MESH

Parameter	Hasil Pengukuran	Indeks	Kategori
Throughput	5,80 MB/s	4	Sangat Bagus
Packet Loss	5,53 %	3	Bagus
Delay	3,17 ms	4	Sangat Bagus
Jitter	7,85 ms	3	Bagus

Penerapan *reactive routing* pada topologi *full-mesh* menghasilkan *throughput* dan *delay* dengan kategori sangat bagus, sedangkan *packet loss* dan *jitter* termasuk kategori bagus. Hasil ini ekuivalen dengan QoS *segment routing* pada topologi yang sama.

4.3.2 *Topologi Tree (depth 5)*

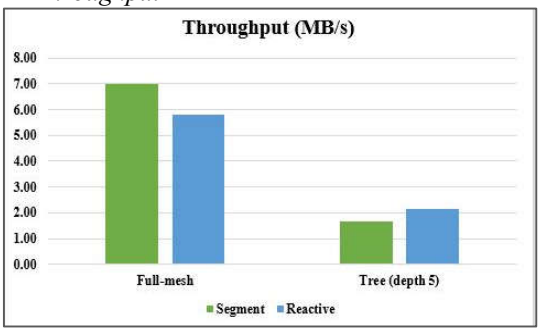
TABEL 8. QOS REACTIVE ROUTING PADA TOPOLOGI TREE (DEPTH 5)

Parameter	Hasil Pengukuran	Indeks	Kategori
Throughput	2,16 MB/s	4	Sangat Bagus
Packet Loss	2,16%	4	Sangat Bagus
Delay	5,43 ms	4	Sangat Bagus
Jitter	18,13 ms	3	Bagus

Protokol *reactive routing* pada topologi *tree (depth 5)* menghasilkan *throughput, packet loss* dan *delay* yang dengan kategori sangat bagus, sedangkan *jitter* menunjukkan kategori bagus. Hasil ini juga ekuivalen dengan QoS pada topologi *tree (depth 5)* yang menerapkan *segment routing*.

4.4 *Perbandingan QoS Segment Routing dan Reactive Routing*

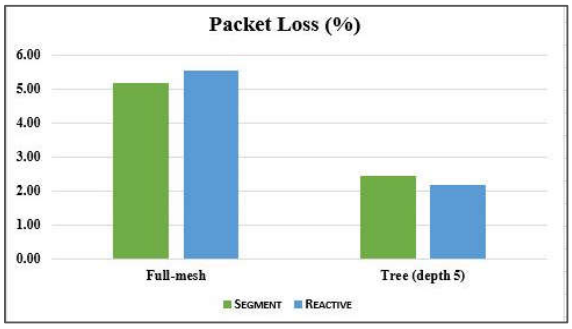
4.4.1 *Throughput*



Gambar 9.Grafik Perbandingan Throughput

Gambar 9 menunjukkan nilai *throughput segment routing* pada topologi *full-mesh* lebih tinggi. Tetapi pada topologi *tree (depth 5)* nilai *throughput reactive routing* lebih tinggi.

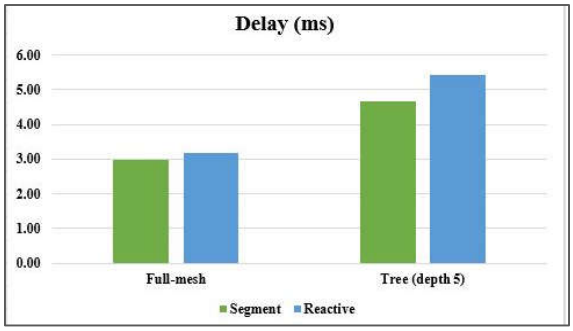
4.4.2 *Packet Loss*



Gambar 10. Grafik Perbandingan Packet Loss

Gambar 10. menunjukkan nilai *packet loss segment routing* pada topologi *full-mesh* lebih rendah, tetapi pada topologi *tree (depth 5)* nilai *packet loss reactive routing* yang lebih rendah.

4.4.3 *Delay*



Gambar 11. Grafik Perbandingan Delay

Gambar 11. menunjukkan nilai *delay segment routing* lebih rendah dibandingkan *reactive routing* pada kedua topologi.

4.4.4 *Jitter*

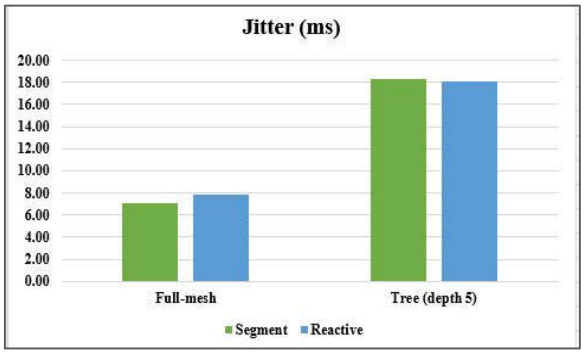


Figure 2 Grafik Perbandingan Jitter

Figure 14 menunjukkan nilai *jitter segment routing* pada topologi *full-mesh* lebih rendah, tetapi pada topologi *tree (depth 5)* nilai *jitter reactive routing* yang lebih rendah.

5. KESIMPULAN

Berdasarkan hasil percobaan pada penelitian, *reactive routing* memiliki nilai *throughput* dan *packet loss* lebih kecil, tetapi nilai *delay* dan *jitter* lebih besar dibandingan dengan *segment routing*. Implementasi topologi lain seperti hybrid, jumlah node yang lebih banyak dan variasi kelas IP yang berbeda dari setiap network, penggunaan protokol routing yang berbeda, penggunaan emulator jaringan lain seperti GNS3 ataupun EstiNet merupakan beberapa tantantangan yang dapat dilakukan pada penelitian berikutnya.

DAFTAR PUSTAKA

[1] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," Comput. Networks, vol. 75, no. PartA, pp. 453–471, 2014.

[2] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," Comput. Networks, vol. 72, pp. 74–98, 2014.

[3] M. Z. Masoud, Y. Jaradat, and I. Jannoud, "On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm," 2015 IEEE Jordan Conf. Appl. Electr. Eng. Comput. Technol. AEECT 2015, pp. 0–4, 2015.

[4] S. Kaur, K. Kaur, and V. Gupta, "Implementing Static Router based on Software Defined Networking," 2016 Int. Conf. Comput. Tech. Inf. Commun. Technol. ICCTICT 2016 - Proc., pp. 358–360, 2016.

[5] D. Satasiya and D. Raviya Rupal, "Analysis of Software Defined Network firewall (SDF)," Proc. 2016 IEEE Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2016, pp. 228–231, 2016.

[6] Z. Wang, D. Tao, and Z. Lin, "Dynamic Virtualization Security Service Construction Strategy for Software Defined Networks," Proc. - 12th Int. Conf. Mob. Ad-Hoc Sens. Networks, MSN 2016, no. November 2010, pp. 139–144, 2017.

[7] S. R. Afif, P. Sukarno, and M. A. Nugroho, "Analisis Perbandingan Algoritma Naive Bayes dan Decision Tree untuk Deteksi Serangan Denial of Service (DoS) pada Arsitektur Software Defined Network (SDN) Pendahuluan Studi Terkait," e-Proceeding Eng., vol. 5, no. 3, pp. 7515–7521, 2018.

[8] I. Ummah, "Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking," Indones. J. Comput., vol. 1, no. 1, pp. 95–106, 2016.

[9] D. Vicino, C. H. Lung, G. Wainer, and O. Dalle, "Evaluating the impact of Software-Defined Networks' Reactive Routing on bittorrent performance," Procedia Comput. Sci., vol. 34, pp. 668–673, 2014.

[10] M. H. Hidayat and N. R. Rosyid, "Analisis Kinerja dan Karakteristik Arsitektur Software-Defined Network Berbasis OpenDaylight Controller," Citee, no. 2085–6350, pp. 194–200, 2017.

[11] R. Afan, A. Virgono, and R. Rumani M., "ANALISIS EFEK PENGGUNAAN KONTROLER RYU DAN POX PADA PERFORMANSI JARINGAN SDN ANALYISIS OF EFFECT OF CONTROLLER RYU AND POX ON SDN NETWORK PERFORMANCE," vol. 2, no. 14, pp. 68–82, 2018.

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

[13] S. N. Hertiana, Hendrawan, and A. Kurniawan, "Performance analysis of flow-based routing in software-defined networking," Proc. - Asia-Pacific Conf. Commun. APCC 2016, pp. 579–585, 2016.

[14] S. L. Hanifa and R. Kartadie, "UJI PERFORMA KONTROLER SOFTWARE-DEFINE NETWORK FLOODLIGHT vs ONOS," JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform., vol. 3, no. 2, pp. 138–144, 2018.

[15] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," 2014 World Congr. Comput. Appl. Inf. Syst. WCCAIS 2014, 2014.

[16] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," J. Netw. Comput. Appl., vol. 80, pp. 200–218, 2017.

[17] M. Saad Waheed, M. Al Mufarrej, M. Sobhieh, A. Al Barrak, A. Baig, and A. Al Mazyad, "Implementation of virtual firewall function in SDN (software defined networks)," 2017 9th IEEE-GCC Conf. Exhib. GCCCE 2017, pp. 1–9, 2018.

[18] S. Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," Proc. - Int. Symp. Comput. Commun., 2014.

[19] S. Reynita Sari, R. Munadi, and D. Dwi Sanjoyo, "ANALISIS PERFORMANSI SEGMENT ROUTING PADA SOFTWARE DEFINED NETWORK MENGGUNAKAN KONTROLER ONOS," vol. 44, no. 12, pp. 2–8, 2019.

[20] O. M. Mon and M. T. Mon, "Quality of Service Sensitive Routing for Software Defined Network Using Segment Routing," Isc. 2018 - 18th Int. Symp. Commun. Inf. Technol., no. Iscit, pp. 255–260, 2018.

[21] C. Liu, A. Raghuramu, C. N. Chuah, and B. Krishnamurthy, "Piggybacking network functions on SDN reactive routing: A feasibility study," SOSR 2017 - Proc. 2017 Symp. SDN Res., pp. 34–40, 2017.

[22] J. Jiang and G. Han, "Routing Protocols for Unmanned Aerial Vehicles," IEEE Commun. Mag., vol. 56, no. 1, pp. 58–63, 2018.

[23] G. E. Rizos, D. C. Vasiliadis, and E. Stergioy, "Improving performance on mobile ad hoc networks by using proxy service," WSEAS Trans. Commun., vol. 7, no. 12, pp. 1137–1146, 2008.

[24] R. Wulandari, "ANALISIS QoS (QUALITY OF SERVICE) PADA JARINGAN INTERNET (STUDI KASUS : UPT LOKA UJI TEKNIK PENAMBANGAN JAMPANG KULON – LIPI)," J. Tek. Inform. dan Sist. Inf., vol. 2, no. 2, pp. 162–172, 2016.

[25] ETSI, "Tr 101 329 V2.1.1 (1999-06)," Telecommun. Internet Protoc. Harmon. Over Networks, vol. 1, pp. 1–37, 1999.

BIODATA PENULIS

Lutfi Kiki Fuadi
Mahasiswa Jurusan Informatika Fakultas Teknik Universitas Siliwangi

Nur Widiyasono
Dosen Jurusan Informatika Fakultas Teknik Universitas Siliwangi