Penerapan Algoritma *Floyd-Warshall* untuk Pemilihan Rute *Routing* OSPF pada Jaringan SDN

Aida Kartika Alit Rahmasari[1], I Made Suartana[2]

[1,2]Jurusan Teknik Informatika/Teknik Informatika, Universitas Negeri Surabaya
[1]aidarahmasari16051204016@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

***Abstrak*— *Routing* merupakan proses pencarian lintasan yang akan digunakan untuk berkomunikasi dalam jaringan. Pada jaringan konvensional, proses *routing* dan *forwarding* dilakukan pada satu perangkat, mengingat keadaan jaringan yang semakin berkembang menjadi sangat heterogen perlu adanya perkembangan pada arsitektur jaringan agar dapat memenuhi tantangan dalam mengatur dan mengoptimalkan sumber daya yang ada. SDN merupakan arsitektur jaringan yang memisahkan fungsi kontrol dengan *forwarding*. Kelebihan SDN dibanding dengan jaringan konvensional yaitu, *programmable controller* yang mengatur jaringan secara terpusat. Pada penelitian ini, skenario jaringan SDN yang dibangun pada *Mininet* dengan menerapkan algoritma *floyd-warshall* sebagai algoritma *routing* pada kontroler *Ryu* berhasil mencari jalur dengan bobot terpendek pada tiga desain topologi. Penghitungan *Link Metric* dilakukan menggunakan bandwidth yang diatur pada setiap *link* di semua topologi. *Time convergence* yang dihasilkan pada penerapan algoritma ini sebesar 0.0884 *ms* pada topologi 1, 0.*1234* ms pada topologi 2 dan 0.1889 *ms* pada topologi 3. Sedangkan uji end-to-end QoS dengan bantuan *software* D-ITG, menghasilkan nilai mencapai 2331.90 Kbps untuk topologi 1, 1426.76 Kbps untuk topologi 2, 1369.36 Kbps untuk topologi 3 untuk throughput. *Delay* pada topologi 1 adalah 0.0595, pada topologi 2 adalah 2240.4431 *ms* dan pada topologi 3 adalah 2292.092 *ms*. *Jitter* mencapai 0.0344 *ms* pada topologi 1, 1.2223 *ms* pada topologi 2, dan 1.1784 *ms* pada topologi 3. *Packet loss* mencapai 0% pada topologi 1, 39.9171% pada topologi 2, dan 19.8571% pada topologi 3.**

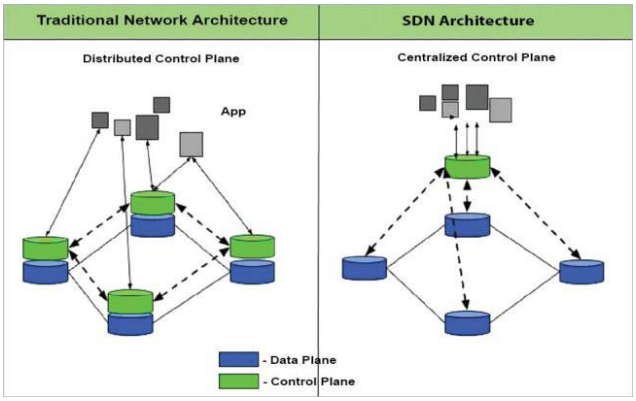
***Kata Kunci*— Routing, SDN, Floyd-Warshall, Mininet, Link Metric**

I. PENDAHULUAN

Perkembangan infrastruktur jaringan yang semakin pesat berdampak pada keadaan lalu lintas data pada jaringan. Banyaknya jenis perangkat baru yang terlibat dan besarnya data yang didistribusikan membuat jaringan menjadi semakin heterogen dan kompleks. Keadaan ini menimbulkan tantangan baru dalam pengelolaan sumberdaya jaringan agar tetap optimal menangani keadaan jaringan yang heterogen dan kompleks [1] [2].

SDN merupakan arsitektur jaringan yang memisahkan antara *network intellegence* dengan *data plane*. Pada SDN dikenal istilah *controller* yang melakukan semua fungsi kontrol terhadap jaringan. *Controller* pada SDN juga berperan sebagai *Network Operating System* (NOS). SDN menerapakan konsep *logically centralized control* yang memiliki kendali penuh terhadap jaringan sehingga dapat mengumpulkan status jaringan dan data konfigurasi secara *real-time* [3] sehingga semua perangkat di bawah kontroler hanya dapat menjalankan fungsi *forwarding.*

Konsep ini sangat berbeda dengan jaringan konvensional. Sebagai contoh, penentuan jalur *routing* pada jaringan konvensional dilakukan oleh perangkat yang sama dengan perangkat yang menjalankan fungsi *forwarding*. Pada jaringan SDN, penentuan rute *routing* akan dilakukan pada *controller* dengan membuat *flow table* yang akan didistribusikan pada semua *switch* yang terhubung dengan *controller*. *Switch* ini yang akan bertugas mengenali paket masuk dan aksi apa yang dilakukan terhadap paket yang masuk. Ilustrasi perbedaan kerja jaringan tradisional dengan SDN ada pada Gbr. 1 berikut ini



Gbr. 1 Perbedaan jaringan konvensional dengan SDN [4]

Pada *routing* di jaringan, penggunaan algoritma pencarian jalur terpendek sudah sangat umum digunakan. Ihsan dkk melakukan penelitian untuk menguji kinerja algoritma *Floyd-Warshall* dalam menentukan rute komunikasi pada jaringan SDN. Konfigurasi perutean ini diterapkan pada kontroler terpusat Ryu. Penelitian ini menganalisis parameter QoS (*delay* dan *packet loss*), waktu konvergensi dan overhead traffic. Pengujian *overhead traffic* dilakukan untuk mencari tahu pengaruh *update* dari kontroler dan *flow table* terhadap besar *overhead traffic*, sedangkan pengujian waktu konvergensi dilakukan untuk mengukur waktu yang diperlukan jaringan untuk mencapai keadaan *steady*. Hasilnya, algoritma *Floyd-Warshall* masih berada dalam standar ITU-T

G.1010. Didapatkan waktu konvergensi dengan rata-rata nilai 17.71446 detik [5].

Ibrahim Attamimi, Widhi Yahya, dan Mochammad Hannats Hanafi melalukan pengujian dengan membandingkan kinerja algoritma *Floyd-Warshall* dengan algortima *Dijkstra* sebagai algoritma *routing* berdasarkan penilaian *convergence time, throughput, link failure,* dan *resource usage.* Kedua algoritma ini diujikan pada 3 jenis topologi dengan jumlah *node* yang berbeda. Hasilnya, performa algoritma *Floyd-Warshall* yang diterapkan pada kontroler jenis *Ryu* unggul pada segi *recovery time* ketika terdapat *link failure*. Sedangkan algortima *Dijkstra* unggul dalam hal *convergence time,* dan *resource usage.* Sedangkan untuk *throughput*, kedua algoritma ini tidak memberikan perbedaan yang signifikan [6].

Ridha Muldina Negara dan Rohmat Tulloh pada tahun 2017 membahas tentang analisis penerapan *Software Defined Network* (SDN). Dengan kontroler *routeflow* dan algoritma OSPF, beberapa skenario jaringan diterapkan untuk mendapatkan hasil yang dapat dipertanggungjawabkan. Pada penelitian ini, parameter yang diukur adalah *time convergence* dan *Quality of Service* (QoS) menggunakan standar nilai performa sesuai ITU.T G1010. Hasilnya, kontroler POX yang digunakan pada penelitian ini menghasilkan *time convergence* yang terus meningkat seiring bertambahnya jumlah *switch*. Untuk parameter QoS yang diujikan pada tiap topologi dan penambahan skenario jaringan masih dalam rentang nilai yang ditetapkan pada ITU-T G.1010. Besarnya topologi jaringan dan beban trafik jaringan menyebabkan kinerja OSPF kurang baik [7].

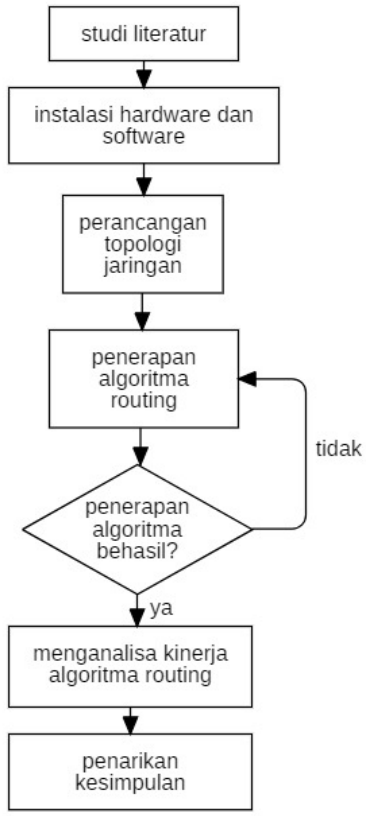
Faizal Ramadhan, Rakhmadhany Primananda dan Widhi Yahya menggunakan Algoritma *Dijkstra* sebagai algoritma pencarian jalur *routing*. Penggunaan algoritma *Dijkstra* sebagai dasar untuk kontroler ONOS, dengan membandingkan metode *geoDistance* dengan *linkMetric* dalam penentu *link weight* telah menghasilkan *latency* sebesar 0,092 *ms* untuk *geoDistance* dan 0,097 *ms* untuk *linkMetric.* Penelitian ini juga menghasilkan waktu konvergensi yang dibutuhkan kontroler dalam penerapan algortima Dijkstra, yaitu sebesar 1,405 *s* [8].

Penelitian selanjutnya oleh Asabere, E. D., Panford, J. K., & Hayfron-Acquah, J. B pada tahun 2017. Penelitian ini membandingkan performa kontroler SDN, yaitu *libfluid, ONOS, OpenDaylight, POX* dan *Ryu*. Parameter yang diukur adalah end-to-end delay yang diukur melalui parameter RTT dan *end-to-end throughput*. Jurnal ini menggunakan topologi linear dengan jumlah *switch* yang berbeda untuk menguji ketahanan kontroler ketika terjadi penambahan *workload*. Jumlah switch yang digunakan adalah 8, 16, 32, 64, …, dst. Hasilnya, semua jenis kontroler mengalami penurunan nilai throughput seiring bertambahnya jumlah *switch* pada jaringan. Kontroler *Ryu* berhenti bekerja ketika jumlah *switch* mencapai 512 *switch*. Begitu juga pada parameter *ETE-Delay*, nilai *ETE-Delay* masing-masing kontroler semakin bertambah seiring bertambahnya jumlah *switch* di jaringan. Kontroler ONOS menghasilkan nilai RTT yang paling kecil dibanding kontroler lain [9].

Penelitian ini berfokus pada penggunaan *link bandwidth* sebagai *linkMetric*. *Bandwidth* yang dikonfigurasi pada tiap-tiap *link* dikonversi menjadi *OSPF cost* menggunakan standar *OmniSecu* [10]. *LinkMetric* kemudian digunakan untuk penghitungan jarak pada algoritma *Floyd-warshall* dalam pencarian jalur dengan bobot terpendek di jaringan SDN. Penelitian ini akan menggunakan kontroler *Ryu* dan *Mininet.*

II. METODOLOGI PENELITIAN

Alur penelitian dalam menerapkan algoritma *Floyd-Warshall* untuk penentuan rute *routing* pada jaringan SDN dimulai dengan studi literatur. Studi ini dilakukan untuk mengumpulkan bahan-bahan dan teori yang mendukung dilakukannya penelitin ini. Tahapan penelitian selanjutnya dilakukan dengan membangun sistem yang diteliti. Penelitian ini membutuhkan *software* emulasi jaringan bernama *Mininet* yang dipasang pada PC dengan sistem operasi *Ubuntu*. Selain *software* emulasi jaringan, hal lain yang perlu dipasang pada PC yaitu *controller* jaringan SDN *Ryu,* dan D-ITG. Sistem yang telah dibangun kemudian dianalisis kinerja dan diambil suatu kesimpulan berdasarkan hasil yang telah didapatkan. Alur pada penelitian ini dijelaskan pada Gbr. 2



Gbr. 2 Alur penelitian

Parameter penilaian didasarkan pada nilai waktu konvergensi dan *End-to-End QoS*, yang didapat melalui 3 skenario pengujian meliputi Uji Skalabilitas, Uji Waktu Konvergensi dan Uji *End-to-End QoS* menggunakan standar TIPHON.

A. *Perancangan Sistem*

Proses perancangan sistem pada penelitian ini adalah sebagai berikut:

1) *Perancangan algoritma Floyd-Warshall*: Perancangan Algoritma Floyd-Warshall yang akan menjadi inti pada controller SDN dibangun menggunakan Bahasa Python, disesuaikan dengan kebutuhan *controller Ryu*. Perancangan algoritma *Floyd-Warshall* pada penelitian ini akan ditambahkan fungsi *Path Reconstruction* untuk mendapatkan jalur yang dilalui. Algoritma *Floyd-Warshall* akan menghitung jarak dari titik i ke titik j melalui titik k dengan persamaan (1) [11]:

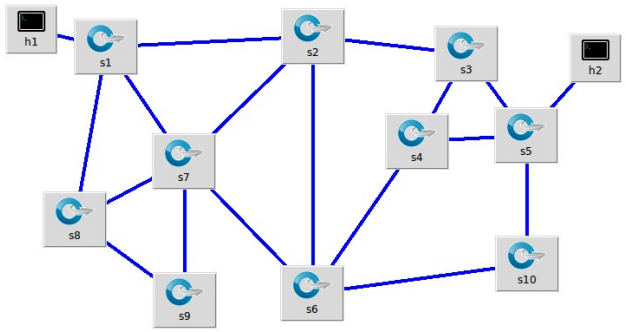
$$d_{ij}^{(k)} = \begin{cases} W_{ij} & if\ k = 0 \\ min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & if\ k \geq 1 \end{cases} \quad (1)$$

Dengan:

$d_{ij}^{(k)}$ = jarak antara titik i dan j melalui titik k

W_{ij} = bobot antara titik i dan j

Pseudo code algoritma *Floyd-Warshall* dengan *Path reconstruction* pada Gbr. 3



Gbr. 3 Topologi 1

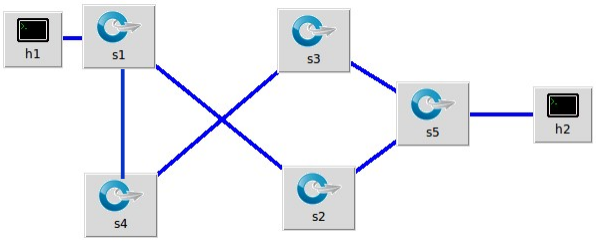
```
   #compute the shortest path using Floyd-Warshall
   and Path Reconstruction.

1  Initialize: n // n adalah jumlah node pada
   topologi
2  Initialize: dist |n|x|n| = infinity // array
   untuk menyimpan jarak minimum
3  Initialize: next|n|x|n| = null //array untuk
   menyimpan index node

4  FloydWarshall()
5    for each vertex v do
6      dist[v][v] ← 0
7      next[v][v] ← v

8    for each edge(u,v) do
9      dist[u][v] ← w(u,v)
10     next[u][v] ← v

11   for k from 1 to |n| do
12     for i from 1 to |n| do
13       for j from 1 to |n| do
14         if dist[i][j]>dist[i][k]+dist[k][j] then
15           dist[i][j] ← dist[i][k] + dist [k][j]
16           next[i][j] ← next [i][k]

17 Path(u,v)
18   if next[u][v] = null then
19     return []
20   path = [u]
21   while u ≠ v
22     u ← next[u][v]
23     path.append(u)
24   return path()
```

Gbr. 4 Pseudo Code Floyd-Warshall

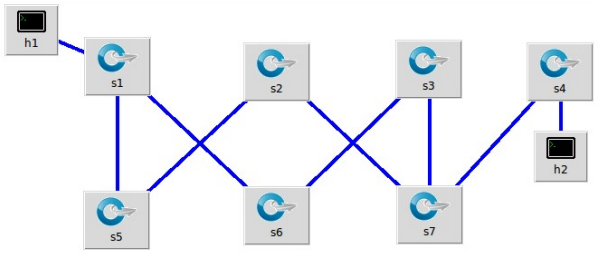
Algoritma *Floyd-Warshall* bekerja dengan melakukan perulangan hingga jarak antar semua pasangan titik pada suatu graf dapat ditemukan. Inti dari Algoritma ini ada pada baris 11 hingga 16 dari Gbr.3.

Untuk mengetahui titik mana saja yang dilalui agar dapat mencapai tujuan dengan jarak yang paling kecil, algoritma ini membutuhkan suatu fungsi tambahan. Fungsi *Path* pada baris 17 hingga 24 dari Gbr.3 merupakan fungsi untuk mencatat titik yang dilalui agar dapat mencapai tujuan dengan bobot yang paling kecil.

2) Perancangan Topologi: Perancangan topologi dilakukan pada Mininet. Penelitian ini menggunakan tiga bentuk topologi dengan jalur dan jumlah *switch* yang berbeda. Desain topologi *1,2 dan 3 dapat dilihat masing-masing pada Gbr.4, Gbr. 5 dan Gbr.6*



Gbr. 5 Topologi 2



Gbr. 6 Topologi 3

3) *Perancangan linkMetric:* Perancangan *linkMetric* didasarkan pada bandwidth yang diatur tiap jalur. Pada tabel I merupakan nilai *bandwidth* dan *cost* tiap-tiap *link*.

TABEL I
BANDWIDTH DAN COST TIAP LINK

Topologi	Src	↔	Dst	Bandwidth (Mbps)	OSPF Cost
1	1		7	10	10
			2	1000	1
			8	1,544	64
	2		7	1,544	64
			3	1000	1
			6	1,544	64
	3		4	10	10
			5	10	10
	4		5	1,544	64
			6	1000	1
	5		10	1000	1
	7		8	1,544	64
			9	10	10
	6		7	10	10
			10	1,544	64
	8		9	1,544	64
2	1		4	10	10
			2	1,544	64
	2		5	10	10
	3		5	1,544	64
			4	10	10
3	1		5	10	10
			6	1,544	64
	2		5	10	10
			7	1,544	64
	3		6	10	10
			7	1000	1
	7		4	1,544	164

LinkMetric didapatkan dari OSPF *reference.* Penentuan *OSPF Cost* berdasarkan *interface bandwidth.* Apabila *interface bandwidt*h bernilai 10 Mbps, maka *cost* pada *link* tersebut bernilai 10 [10]. Pada tabel II dapat dilihat nilai *cost* dari *bandwidth link.*

TABEL II
OSPF COST

Bandwidth	Cost
100 Gbps	1
10 Gbps	1
1 Gbps	1
10 Mbps	10
1,544 Mbps	64
768 Kbps	133
384 Kbps	266
128 Kbps	781

III. HASIL DAN PEMBAHASAN

A. *Uji Skalabilitas*

Pengujian ini dilakukan dengan menerapkan algoritma *floyd-warshall* pada tiga bentuk topologi dengan jumlah *switch* dan *link* yang berbeda. Pengujian ini dilakukan pertama kali sebelum lanjut pada tahap uji berikutnya. Jumlah *switch* dan *link* pada tiap topologi dapat dilihat pada tabel III.

TABEL III
JUMLAH SWITCH DAN LINK TIAP TOPOLOGI

Topologi	*Switch*	*link*
1	10	16
2	5	5
3	7	7

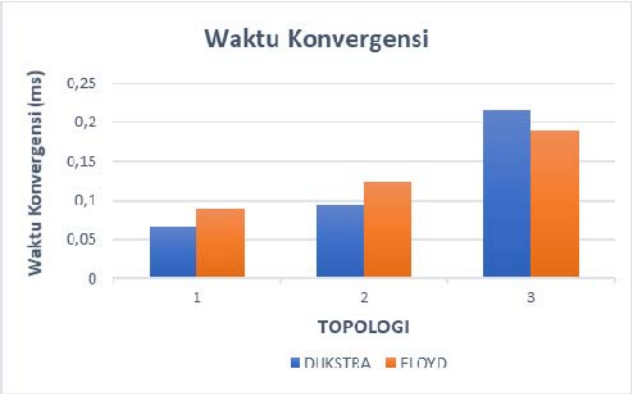
Pada pengujian ini didapatkan jarak dan jalur yang dilalui untuk mendistribusikan paket pada tiap topologi. Pada topologi 1, *host* terletak pada *switch* 5 dan 1, pada topologi 2, host terletak pada *switch* 1 dan 5, dan pada topologi 3, *host* terletak pada *switch* 1 dan 4. Jalur dan jarak pada tiap topologi dapat dilihat pada tabel IV.

TABEL IV
JARAK DAN JALUR TIAP TOPOLOGI

Topologi	Src	Dst	Floyd-Warshall		Dijkstra	
			Jarak	Jalur	Jarak	Jalur
1	1	5	12	1,2,3,5	12	1,2,3,5
	5	1	12	5,3,2,1	12	5,3,2,1
2	1	5	74	1,2,5	74	1,2,5
	5	1	74	5,2,1	74	5,2,1
3	1	4	139	1,6,3,7,4	139	1,6,3,7,4
	4	1	139	4,7,6,3,1	139	4,7,3,6,1

B. Uji Waktu Konvergensi

Pengujian waktu konvergensi ini bertujuan untuk mengetahui waktu yang dibutuhkan kontroler untuk membentuk tabel *routing*. Uji waktu konvergensi dilakukan dengan mengirimkan paket *ping* dan mengamati RTT. Pada jaringan yang menggunakan algoritma *Dijkstra*, rata-rata waktu konvergensi dalam 7 kali percobaan adalah 0.0654 *ms* pada topologi 1, 0.0944 *ms* pada topologi 2, dan 0.2154 *ms* pada topologi 3. Sedangkan pada jaringan dengan algoritma *Floyd-Warshall*, rata-rata waktu konvergensi pada topologi 1 adalah 0.0884 *ms*, 0.1234 *ms* pada topologi 2 dan 0.1889 *ms* pada topologi 3. Perbandingan rata-rata waktu konvergensi pada kedua algoritma dapat dilihat pada Gbr. 7



Gbr. 7 Rata-rata waktu konvergensi

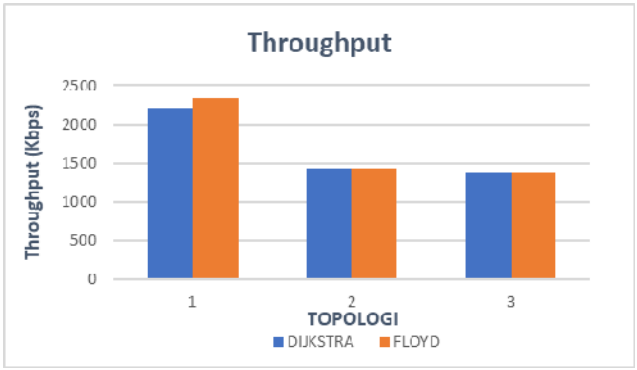
C. Uji End-to-End QoS

Pengujian QoS dilakukan untuk mengetahui kualitas jaringan berdasarkan parameter *delay*, *jitter*, *throughput* dan *packet-loss*. Pengujian dilakukan dengan mengirimkan paket UDP menggunakan *software D-ITG* dalam waktu 10 detik. Setelah waktu pengiriman selesai, *D-ITG* akan menunjukkan nilai-nilai untuk tiap parameter dan nilai tersebut akan dicocokkan dengan tabel TIPHON. Standar TIPHON [12] dapat dilihat pada tabel V.

TABEL V
STANDAR TIPHON

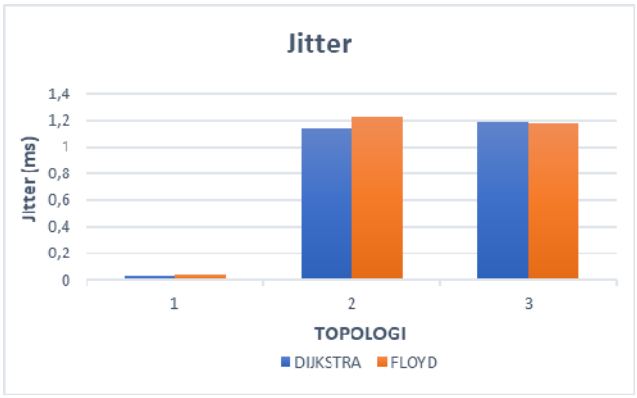
Kategori	*jitter* (ms)	*packet loss* (%)	*delay* (ms)
Sangat bagus	0	0	< 150
Bagus	75	3	< 250
Sedang	125	15	< 350
Jelek	225	25	< 450

Hasil uji *end-to-end QoS* untuk tiap algoritma dalam 7 kali percobaan, didapatkan rata-rata nilai untuk *throughput* adalah sebagai berikut: 2212.95 Kbps pada topologi 1, 1424.89 Kbps pada topologi 2, dan 1368.73 Kbps pada topologi 3 menggunakan algoritma *Dijkstra*. Sedangkan pada algoritma *Floyd-Warshall* mencapai 2331.90 Kbps untuk topologi 1, 1426.76 Kbps untuk topologi 2, 1369.36 Kbps untuk topologi 3. Perbandingan nilai rata-rata untuk parameter *throughput* dapat dilihat pada Gbr. 8.



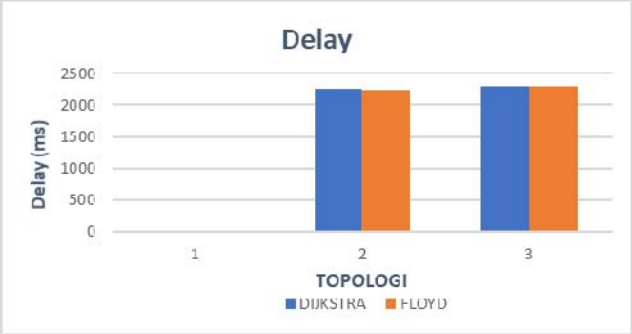
Gbr. 8 Rata-rata nilai *Throughput*

Nilai rata-rata *jitter* untuk *Floyd-Warshall* adalah 0.0344 *ms* pada topologi 1, 1.2223 *ms* pada topologi 2, dan 1.1784 *ms* pada topologi 3. Sedangkan jika menggunakan algoritma *Dijkstra*, didapatkan rata-rata nilai *jitter* 0.0304 *ms* pada topologi 1, 1.144 *ms* pada topologi 2, dan 1.18557 *ms* pada topologi 3, perbandingan nilai rata-rata *jitter* pada kedua algoritma dapat dilihat pada Gbr. 9



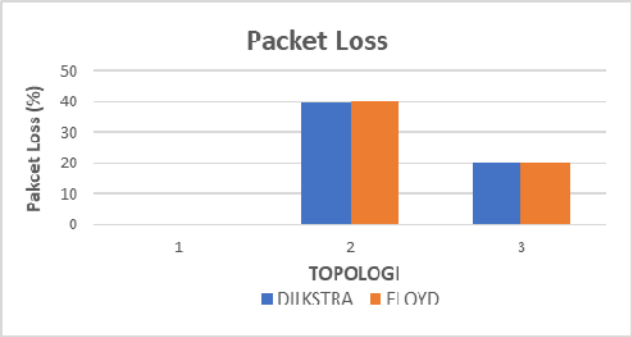
Gbr. 9 Rata-rata nilai *jitter*

Nilai rata-rata *delay* untuk *Dijkstra* mencapai 0.0505 *ms* pada topologi 1, 2258.432 *ms* pada topologi 2, dan 2295.3045 *ms* pada topologi 3. Sedangkan pada *Floyd-Warshall, delay* pada topologi 1 adalah 0.05957 *ms*, pada topologi 2 adalah 2240.4431 *ms* dan pada topologi 3 adalah 2292.092 *ms*. Perbandingan nilai rata-rata *delay* pada kedua algoritma dapat dilihat pada Gbr. 10



Gbr. 10 Rata-rata nilai *delay*

Besaran rata-rata *packet loss* pada jaringan yang menggunakan algoritma *Dijkstra* adalah 0% pada topologi 1, 39.6814% pada topologi 2, dan 20.3428% pada topologi 3. Sedangkan pada algoritma *Floyd-Warshall,* packet loss mencapai 0% pada topologi 1, 39.9171% pada topologi 2, dan 19.8571% pada topologi 3. Perbandingan *packet-loss* pada kedua algortima dapat dilihat pada Gbr. 11



Gbr. 11 Rata-rata nilai *packet loss*

Berdasarkan standar TIPHON, hasil uji *jitter* pada kedua algoritma masuk kategori sangat bagus dan tidak melebihi standar TIPHON pada semua topologi. Hasil uji *delay* pada kedua algoritma yang dihubungkan dengan topologi 2 dan 3 tergolong buruk, sedangkan pada topologi 1 masuk kategori sangat bagus. Sedangkan untuk uji *packet loss,* kedua algoritma sama-sama buruk pada topologi 2, dan meningkat pada topologi 3 dan 1 hingga masuk kategori sangat baik karena mencapai 0%.

IV. KESIMPULAN

Berdasarkan beberapa skenario pengujian terhadap algoritma *Dijkstra* dan *Floyd-Warshall* yang digunakan sebagai algoritma *routing* pada jaringan SDN, didapatkan hasil sebagai berikut:

1. Penerapan algoritma *Floyd-warshall* sebagai algoritma dalam *routing* OSPF untuk menentukan jalur terpendek dapat dilakukan dengan bantuan *software* emulasi jaringan *mininet*. Algoritma *Floyd-warshall* yang diterapkan pada kontroler *Ryu* berhasil menemukan jalur dengan bobot terkecil dalam komunikasi paket data di jaringan SDN. Tidak ada perbedaan jalur terpendek yang dihasilkan algoritma *Floyd-Warshall* dengan *Dijkstra.*
2. Terdapat perbedaan hasil pegujian terhadap penerapan algoritma *Floyd-warshall* dan algoritma *Dijkstra* sebagai berikut: Algortima *Floyd-Warshall* menghasilkan nilai rata-rata *jitter* 0.0344 *ms* pada topologi 1, 1.2223 *ms* pada topologi 2, dan 1.1784 *ms* pada topologi 3 dan *Throughput* yang mencapai 2331.9 Kbps untuk topologi 1, 1426.76 Kbps untuk topologi 2, 1369.36 Kbps untuk topologi 3. Rata-rata *delay* pada topologi 1 adalah 0.0595, pada topologi 2 adalah 2240.4431 *ms* dan pada topologi 3 adalah 2292.092 *ms. Packet loss* mencapai 0% pada topologi 1, 39.9171% pada topologi 2, dan 19.8571% pada topologi 3. Rata-rata waktu konvergensi pada topologi 1 adalah 0.0884 *ms*, 0.1234 *ms* pada topologi 2 dan 0.1889 *ms* pada topologi 3. Sedangkan hasil untuk algoritma *Dijkstra,* rata-rata *throughput* 2212.95 Kbps pada topologi 1, 1424,89 Kbps pada topologi 2, dan 1368.73 Kbps pada topologi 3. Rata-rata *jitter* mencapai 0.0304 *ms* pada topologi 1, 1.144 *ms* pada topologi 2, dan 1.18557 *ms* pada topologi 3. Sedangkan untuk parameter *delay,* algoritma *Dijkstra* mencapai 0.0505 *ms* pada topologi 1, 2258.432 *ms* pada topologi 2, dan 2295.3045 *ms* pada topologi 3. Rata-rata *packet loss* pada jaringan adalah 0% pada topologi 1, 39.6814% pada topologi 2, dan 20.3428% pada topologi 3. Untuk rata-rata waktu konvergensi adalah 0.0654 *ms* pada topologi 1, 0.0944 *ms* pada topologi 2, dan 0.2154 *ms* pada topologi 3.
3. Penggunaan *bandwidth link* sebagai *link metric* membuat pengaturan *bandwidth* pada tiap *link* berpengaruh pada waktu konvergensi dan *End-to-End QoS* yang dihasilkan. Pada topologi 1 yang memiliki total jarak terpendek, membutuhkan waktu konvergensi paling sedikit dibanding dengan topologi 2 dan 3. Begitu pula dengan hasil uji *end-to-end QoS*, topologi 1 memiliki hasil *end-to-end QoS* paling bagus dibanding dengan topologi 2 dan 3 pada kedua algoritma.

V. SARAN

1. Penelitian selanjutnya dapat dilakukan dengan membangun jaringan SDN pada keadaan jaringan sebenarnya tanpa menggunakan *software* emulasi jaringan.
2. Parameter yang diujikan juga mempertimbangkan keadaan perangkat keras yang digunakan, seperti mengukur *CPU dan memory usage.*
3. Menjalankan suatu *service* tertentu untuk pengujian kinerja algoritma *routing.*

REFERENSI

[1] K. Anam & R. Adrian, "Analisis Performa Jaringan Software Defined Network Berdasarkan Penggunaan Cost Pada Protokol Ruting Open Shortest Path First," CITEE, pp. 1-8, 2017.

[2] J. Xie, F. R. Yuy, T. Huang, R. Xie, J. Liu, C. Wangz dan Y. Liu, "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," IEEE Communications Surveys & Tutorials, pp. 1-39, 2018.

[3] S. Tomovic, M. Radonjic dan I. Radusinovic, "Bandwidth-Delay Constrained Routing Algorithms for Backbone SDN Networks," TELSIKS 2015, pp. 227-230, 2015.

[4] J. Bhatia, R. Govani dan Y. Modi. (2017) OpenSource. [Online], http://opensourceforu.com/2017/10/primer-software-defined-networking-sdn-openflow-standard/, tanggal akses: 13 Mei 2020.

[5] I. A. Saputra, Rumani. R. M. dan S. N. Hertiana, "Uji Performansi Algoritma Floyd-Warshall pada Jaringan Software Defined Network (SDN)," Jurnal Elektronika dan Telekomunikasi, Vol. 16(2), hal. 52-58, 2016.

[6] I. Attamimi, W. Yahya & M. H. Hanafi, "Analisis Perbandingan Algoritma Floyd-Warshall dan Dijkstra untuk Menentukan Jalur Terpendek Pada Jaringan Openflow," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, hal. 1842-1849, 2017.

[7] R. M. Negara and R. Tulloh, "Analisis Simulasi Penerapan Algoritma OSPF Menggunakan RouteFlow pada Jaringan Software Defined Network (SDN)," Jurnal Infotel Vol.9 No.1 Februari 2017, hal. 75-83, 2017.

[8] F. Ramadhan, R. Primananda and W. Yahya, "Implementasi Routing Berbasis Algoritme Dijkstra Pada Software Defined Networking Menggunakan Kontroler Open Network Operating System," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, pp. 2531-2541, 2018.

[9] E. D. Asabere, J. K. Panford and J. B. Hayfron-Acquah, "Comparative Analysis Of Convergence Times Between Ospf, Eigrp, Is-Is And Bgp Routing Protocols In A Network," International Journal of Computer Science and Information Security (IJCSIS) Vol. 15(12), pp. 225-227, 2017.

[10] (2014) OmniSecu. [Online], http://www.omnisecu.com/cisco-certified-network-associate-ccna/what-is-ospf-metric-value-cost-and-ospf-default-cost-reference-bandwidth.php, tanggal akses: 14 April 2020.

[11] A. Ojo, N.-W. Ma and I. Woungang, "Modified Floyd Warshall Algorithm for Equal Cost Multipath in Software Defined Data Center," in Workshop on Advances in Software Defined and Context Aware Cognitive Network (IEEE SCAN-2015), 2015.

[12] Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS) TR 101 329 V2.1.1, 1999.