Uji Performansi Algoritma Floyd-Warshall pada Jaringan *Software Defined Network* (SDN)

Performance Analysis of Floyd-Warshall Algorithm on Software Defined Network (SDN)

Ihsan Aris Saputra*, R. Rumani M., dan Sofia Naning Hertiana

Fakultas Teknik Elektro, Telkom University.
Jalan Telekomunikasi No. 1, Sukapura, Bojongsoang, Bandung, Jawa Barat 40257, Indonesia

Abstrak

Penentuan rute pada sebuah jaringan *software defined network* (SDN) merupakan salah satu contoh topik yang menarik untuk diteliti. Algoritma penentuan rute terpendek pada jaringan SDN sangatlah menentukan apakah jaringan SDN yang dibangun dengan algoritma tersebut sudah optimal. Salah satu algoritma penentuan rute terpendek yaitu algoritma Floyd-Warshall, yang akan diuji coba dan dianalisis apakah sudah termasuk algoritma yang optimal pada jaringan SDN dengan membandingkan dengan standarisasi yang ada. Pengujian akan dilakukan dengan mengirimkan paket data, VoIP dan video dengan melihat *overhead traffic* dan QoS (*delay* dan *packet loss*). Algoritma Floyd-Warshall akan digunakan pada pengontrol Ryu dan menggunakan Mininet sebagai emulator jaringan dengan topologi berbasis Abiline. Hasil simulasi dan pengujian algoritma Floyd-Warshall sebagai penentuan jalur terbaik dalam jaringan SDN, mendapatkan hasil yang memenuhi standarisasi. Nilai dari QoS yang didapat untuk *delay* masih berada pada nilai yang menjadi standar ITU-T G.1010. *Packet loss* yang dihasilkan semua jenis layanan sudah memenuhi standar ITU-T G.1010 yaitu 0% hingga saat pada jaringan diberikan *background traffic* melebihi kapasitas *link* yaitu pemberian sebesar 75 Mbps. Dalam pengujian waktu konvergensi didapatkan waktu dengan rata-rata nilai 17.71446 detik. Kemudian untuk *overhead traffic* menunjukan bahwa perubahan *overhead* dipengaruhi oleh *controller update* dan juga *flow update*, dimana ketika sering terjadinya *controller update* dan *flow update* maka semakin besar juga *overhead* yang didapat.

Kata kunci : *software defined network (SDN),* Floyd-Warshall, *overhead, QoS*

Abstract

Determination of routes on a network software defined network (SDN) is one example of an interesting issue. On the application of the algorithm as the shortest route of weaving on a network is to determine whether the network SDN built with the algorithm is optimal. One algorithm determining the shortest route that is Floyd-Warshall algorithm, which will be in trials and in the analysis of whether it has been included on the optimal algorithm with comparing SDN network with existing standardization. Testing will be done by sending packets of data, VoIP and video by looking at the overhead traffic and QoS (delay and packet loss). Floyd-Warshall algorithm will be used on the controller Ryu and use Mininet as a network emulator using Abiline-based topology. The results of the simulation and testing of algorithms Floyd-Warshall as determining the best path in the SDN network, getting results that meet standardization. The value of the acquired QoS for delay still be at a value that becomes the standard ITU-T G.1010. Packet loss is generated all kinds of services to meet the standard ITU-T G.1010 is 0% until the given background network traffic exceeds the capacity of the link is the provision of 75 Mbps. In testing the convergence time obtained by the time the average value of 17.71446 second. Then for overhead traffic overhead addressing changes influenced by the flow controller updates and updates, which when frequent updates and flow controller updates the greater the overhead obtained.

Keywords : *software defined network (SDN), Floyd-Warshall, overhead, QoS*

I. PENDAHULUAN

Perkembangan teknologi jaringan belakangan ini berkembang pesat, dimana perkembangannya membuat kita lebih dimudahkan baik dalam membangun, memonitoring atau memelihara suatu jaringan komputer. Dengan pesatnya perkembangan teknologi jaringan memunculkan sebuah paradigma baru dalam teknologi jaringan yaitu *software defined network* (SDN). SDN adalah istilah yang merujuk pada konsep/paradigma baru dalam mendesain, mengelola dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan dan inovasi dibidang ini yg semakin lama semakin kompleks. Jika pada jaringan konvensional jalur data (*data plane*) dan jalur kontrol (*control plane*) dijadikan satu pada satu perangkat sedangkan pada jaringan SDN memisahkan paket jalur data dan jalur kontrol [1]. Dengan pemisahan antara jalur kontrol dengan jalur data pada jaringan SDN memudahkan dalam membangun, memonitoring atau memelihara suatu jaringan komputer dengan ketentuan yang dibuat. Berdasarkan kelebihan dari paradigma SDN tersebut, dalam penelitian ini akan diimplementasikan algoritma Floyd-Warshall sebagai

* Corresponding Author.
Email: ihsanariis@students.telkomuniversity.ac.id
Received: November 2, 2016; Revised: December 5, 2016
Accepted: December 6, 2016
Published: December 20, 2016

doi: 10.14203/jet.v16.52-58

penentuan jalur yang akan dipilih dimana algoritma Floyd-Warshall adalah salah satu algoritma *shortest path* dengan pencarian jalur *all pair shortest path*. Berdasarkan kelebihan tersebut algoritma Floyd-Warshall yang mempunyai jenis *all pairs* artinya penentuan lintasan terpendek dapat ditentukan dari semua pasangan simpul, kecepatan dalam penentuan lintasan terpendek sangat cepat apabila diterapkan dalam suatu sistem, performansinya stabil, dan keputusan yang nantinya diambil saling terkait [2]. Dengan melakukan analisis melihat dari *overhead* dan *QoS* (*delay* dan *packet loss*) untuk menentukan apakah algoritma Floyd-Warshall bisa menjadi algoritma penentuan jalur terbaik berdasarkan standarisasi ITU-T yaitu G.1010.

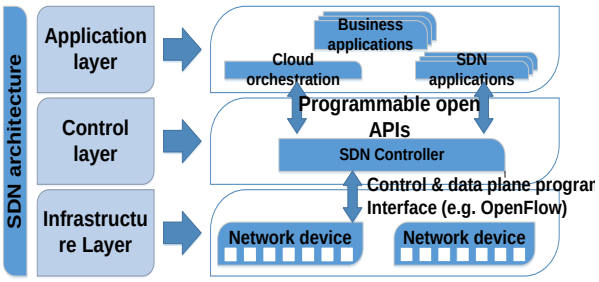
II. TEORI

A. *Software Defined Network*

Metode jaringan *Software-Defined Network* adalah sebuah metode dalam bebuat suatu jaringan komputer dengan memisahkan jalur data dan jalur kontrol, dimana pada metode ini *switch* akan berfungsi sebagai jalur data sedangkan untuk jalur kontrol terdapat pada kontroler [3]. Tujuan dari metode jaringan SDN ini diantaranya adalah untuk memudahkan dalam membangun suatu jaringan komputer, memudahkan dalam memelihara dan juga memberikan keamanan pada suatu jaringan komputer. Prinsip kerja dari *switch* pada metode jaringan SDN antara lain adalah menghubungkan antara *switch* dengan *switch*, *switch* dengan kontroler, dan *switch* dengan *host*. Dalam metode jaringan SDN terdapat salah satu protokol yang digunakan untuk menghubungkan antara kontroler dengan *switch* yaitu protokol OpenFlow, pada protokol OpenFlow terdapat OpenFlow *switch* yang berfungsi sebagai *switch* dan OpenFlow kontroler yang berfungsi sebagai kontroler [4]. OpenFlow [5] adalah standar pertama mengenai antarmuka komunikasi antara *control layer* dan *forwarding layer* dalam arsitektur SDN. OpenFlow memungkinkan akses secara langsung atau manipulasi dari forwarding layer berupa perangkat jaringan seperti *switch* dan *router* atau yang bersifat virtual (*hypercisor - based*). OpenFlow diimplementasikan dari kedua sisi antarmuka baik dari sisi infrastruktur jaringan (seperti halnya *switch* dan *router*) dan kontrol perangkat lunak SDN.

Arsitektur SDN [5] dapat dilihat pada Gambar 1 terdiri dari tiga buah layer yang dapat diakses melalui *open* API (*Application Progammable Interface*) , yaitu:

- Layar aplikasi yang terdiri dari bisnis aplikasi *end-user* untuk layanan komunikasi SDN. Batas antara layar aplikasi dan layar kontrol adalah sebuah *northbound* API
- Layar kontrol yang menyediakan fungsi kontrol terhadap perilaku *forwarding* di dalam jaringan melalui sebuah antarmuka terbuka
- Layar infrastruktur terdiri dari elemen dan perangkat jaringan yang dapat menyediakan *packet switch*ing dan *forwarding*.



Gambar 1. Arsitektur SDN

B. Mininet

Mininet [6] merupakan sebuah sistem virtualisasi yang dapat menggambarkan jaringan yang besar dengan hanya menggunakan sebuah laptop. Mininet bersifat *open source*, sehingga proyek yang telah dilakukan berupa *source code*, *scripts*, dan dokumentasi yang dapat dikembangkan oleh siapapun. Karakteristik yang dimiliki Mininet yaitu:

- ***Flexible***, topologi dan fungsionalitas yang baru akan sistem yang dibuat harus didefinisikan dalam bentuk *software* menggunakan bahasa yang familiar dan *operating system*.
- ***Deployable***, jika ingin membangun sebuah *prototype* dengan fungsionalitas yang benar untuk jaringan yang bersifat *hardware-based*, *testbed* yang dibuat tidak memerlukan perubahan terhadap kode atau konfigurasi yang dirubah sebelumnya.
- ***Interactive***, mengatur dan menjalankan jaringan harus berdasarkan waktu yang rill, sebagaimana berhubungan dengan jaringan yang sebenarnya.
- ***Scalable***, lingkup *prototype* ini dapat menggambarkan jaringan dengan jumlah *switch* dari ratusan hingga ribuan hanya dalam satu laptop.
- ***Realistic***, perilaku dari *prototype* dapat menunjukan perilaku jaringan yang sebenarnya dengan tingkat kepercayaan yang tinggi.
- ***Share-able***, *prototype* yang telah dibentuk dapat saling dibagikan antar kolabotor untuk dijalankan atau dimodifikasi dalam eksperimen selanjutnya.

C. Floyd-Warshall

Algoritma Floyd-Warshall adalah algoritma penghitungan jalur terpendek dimana algoritma tersebut dapat mencari semua jarak dari setiap simpul (*all pairs shortest path*) yang artinya dapat digunakan untuk menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik [7]. Algoritma Floyd-Warshall memiliki input *graf* berarah dan berbobot (V,E), yang berupa titik (*node*/vertex V) dan sisi (edge E). Pada sisi E diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi *graf* ini untuk memiliki siklus dengan bobot negatif atau perulangan pada jalur yang diambil.

```
          Floyd-Warshall Pseudo Algorithm
/* Assume a function edgeCost(i,j)
which returns the cost of the edge from i to j
(infinity if there is none). Also assume that
n is the number of vertices and edgeCost(i,i)
= 0 */

int path[][];

/* A 2-dimensional matrix. At each step in the
algorithm, path[i][j] is the shortest path
from i to j using intermediate vertices
(1..k-1). Each path[i][j] is initialized to
edgeCost(i,j) or infinity if there is no edge
between i and j.*/

procedure FloydWarshall()
for k := 1 to n
 for i := 1 to n
  for j := 1 to n
    path[i][j] =
     min(path[i][j],path[i][k]+path[k][j]);
    path[i][j]= path[i][k]+path[k][j]
```

Gambar 2. *Pseudocode* Floyd-Warshall

III. PERANCANGAN DAN ANALISIS

A. Perancangan Sistem

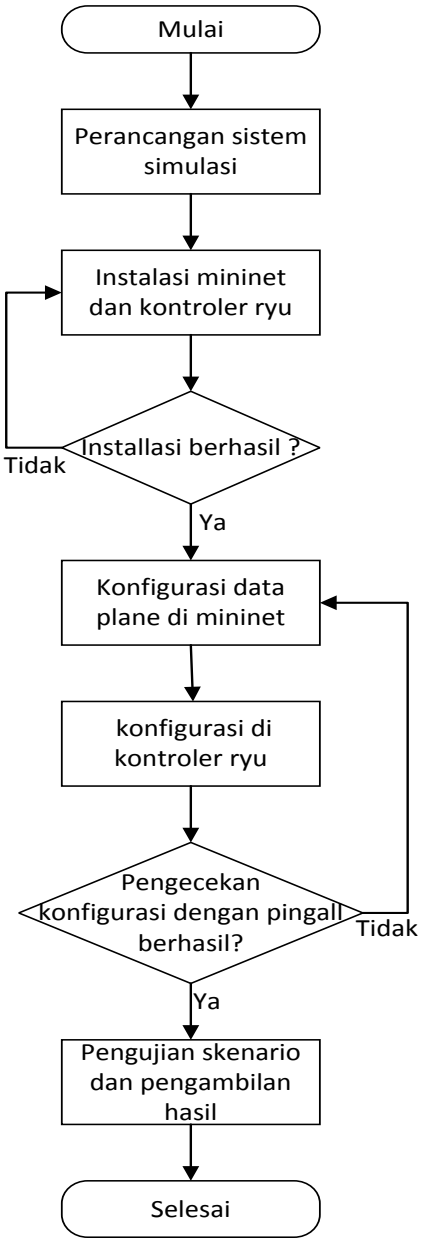
Pada simulasi yang dibuat, simulasi dijalankan berdasarkan dengan *flowchart* pada Gambar 3.

TABEL 1
BANDWIDTH DAN *DELAY* PADA *LINK*

Link	*Bandwidth*	*Delay*
S1 – S2	70 Mbps	3 ms
S1 – S3	100 Mbps	4 ms
S2 – S3	90 Mbps	3 ms
S2 – S4	100 Mbps	3 ms
S3 – S5	100 Mbps	1 ms
S5 – S7	90 Mbps	2 ms
S4 – S7	80 Mbps	3 ms
S4 – S6	70 Mbps	1 ms
S6 – S8	80 Mbps	4 ms
S7 – S8	100 Mbps	1 ms
S7 – S9	100 Mbps	4 ms
S8 – S9	90 Mbps	1 ms
S8 – S10	100 Mbps	2 ms
S9 – S11	80 Mbps	3 ms
S10 – S11	100 Mbps	1 ms
S10 – S12	100 Mbps	4 ms
S10 – S13	100 Mbps	1 ms
S11 – S12	80 Mbps	2 ms
S12 – S15	100 Mbps	2 ms
S13 – S14	70 Mbps	4 ms
S14 – S15	100 Mbps	1 ms
S14 – S16	80 Mbps	1 ms
S15 – S16	100 Mbps	1 ms

1) Topologi

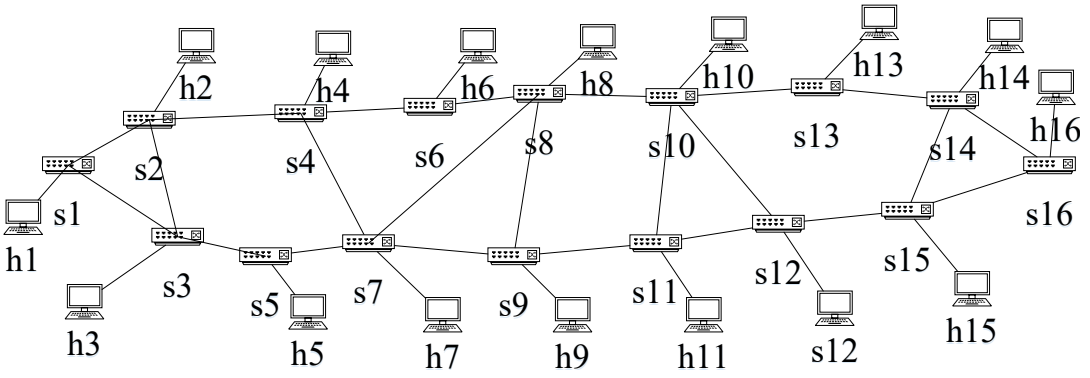
Topologi yang digunakan mengacu pada topologi abiline dengan ditambahkan sedikit perubahan seperti yang terlihat pada Gambar 4. Dengan ketentuan *bandwith* dan *delay* setiap *link* yang telah ditentukan secara acak terlihat pada Tabel 1.



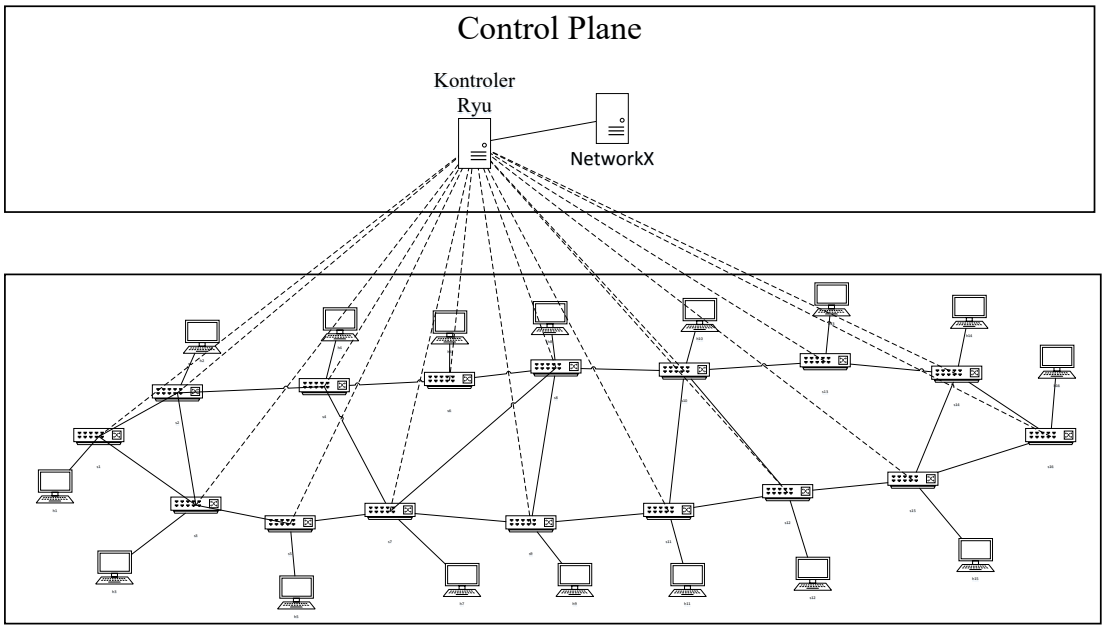
Gambar 3. *Flowchart* Sistem

2) Model Sistem Simulasi

Sistem ini mensimulasikan jaringan berbasis SDN menggunakan Ryu SDN *framework* sebagai kontroler dengan menggunakan NetworkX sebagai perangkat lunak yang akan memanipulasi topologi jaringan pada jalur data menjadi sebuah *graph* untuk kemudian mengimplementasikan algoritma Floyd-Warshall pada kontroler dengan mengakses *library* yang sudah tersedia pada NetworkX untuk penentuan jalur yang ada pada topologi yang sudah dibuat sedangkan untuk bagian jalur data menggunakan emulator Mininet, seperti terlihat pada Gambar 5.



Gambar 4. Topologi



Gambar 5. Model Sistem Simulasi

3) Simulasi setting

Untuk melakukan simulasi pada penelitian ini digunakan beberapa perangkat tambahan, yaitu:

- Iperf, digunakan untuk membangkitkan trafik.
- *Distributed Internet Traffic Generator* (D-ITG), digunakan untuk membangkitkan trafik.
- Wireshark, digunakan untuk melihat *overhead traffic.*

Untuk penentuan bobot pada tiap *node* menggunakan perhitungan dari *metric routing*. *Metric*s merupakan variable jaringan yang digunakan untuk menentukan jalur yang akan dilewati oleh sebuah paket dalam suatu jaringan. Nilai yang digunakan untuk menentukan nilai *metric* bermacam-macam berdasarkan *routing protocol* yang dipakai, namun nilai *metric* ini bisa diatur secara statis oleh administrator jaringan berupa suatu nilai bilangan bulat. Pada penelitian ini untuk bobot pada setiap *link* menggunakan *metric routing* yang digunakan berdasarkan *bandwidth* (OSPF), *delay* dan *hop*. Dimana pada penggunaan *metric bandwidth* menggunakan rumus dari OSPF yaitu [8]:

Metric = 100Mbps/(interface *bandwidth*)

Sedangkan untuk *delay* yaitu nilai *delay* pada tiap *link* dan untuk *hop* kapasitas tiap *link*-nya adalah 1. Pengaturan parameter simulasi ditunjukkan pada Tabel 2.

TABEL 2
PENGATURAN PADA SIMULASI

Parameter	Keterangan
Bandwith pada *link*	70 Mbps - 100 Mbps
Delay pada *link*	0 ms - 4 ms
Jumlah *host*	16
Jumlah *switch*	16
Jumlah *link*	39
Controller	Ryu
Perangkat tambahan	Mininet, Iperf, D-ITG Wireshark

B. Simulasi dan Analisis

Pengujian dilakukan terhadap tiga parameter yaitu, waktu konvergensi dimana waktu yang dibutuhkan ketika jalur kontrol dan jalur data terhubung sampai dengan jaringan dalam keadaan *steady state* dan siap digunakan dan yang dibutuhkan kontroler untuk menemukan jalur baru ketika terjadi pemutusan, pengaruh jumlah *overhead traffic* pada jaringan ketika adanya perubahan waktu *controller update* dan *flow*

update, dan QoS (*delay* dan *packet loss*) dengan cara melakukan pengiriman beberapa paket yaitu paket data, VoIP dan video mengunakan *traffic generator* D-ITG dengan ketetuan pengiriman sebagai berikut :

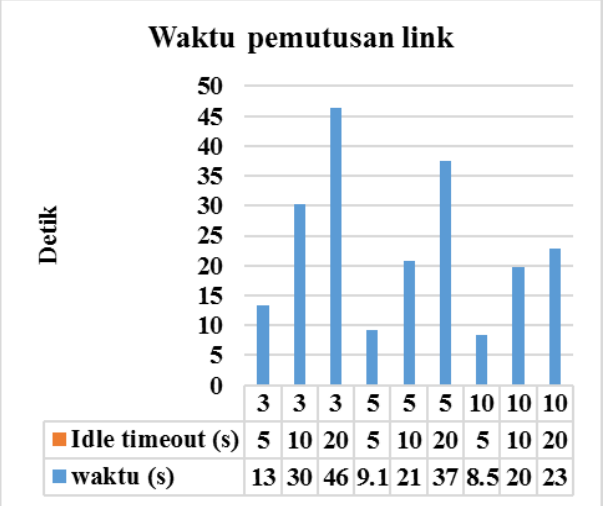
- Data yang dibangkitkan adalah paket berbasis WWW dengan ukuran paket menggunakan distribusi Pareto dengan nilai k = 18,5 dan σ = 1,1 dengan rata-rata *inter-arrival time* 0,125 menggunakan jenis *transport layer* UDP membutuhkan *bandwidth* sebesar 32 Kbps [9].
- VoIP dengan G.711 *codec* tanpa *voice activation detection* (VAD) sebanyak 100 pps dengan ukuran paket 80 *bytes*, sehingga membutuhkan *bandwidth* 70,4 kbps [10].
- *Video streaming* H.264/MPEG *codec* sebanyak 24 *frame* (satu paket per *frame*) per sekon, dengan ukuran paket terdistribusi normal μ = 27791 *bytes* dan σ^2= 6254 *bytes*, sehingga membutuhkan *bandwidth* sekitar 5,4 Mbps [11].

Setelah paket data dikirim, dibangkitkan juga *background traffic* dengan menggunakan Iperf dimana untuk besaran *background traffic* yang di bangkitkan yaitu : 0 mb, 15 mb, 30 mb, 45 mb, 60 mb, 75 mb ,dan 90 mb.

1) Waktu Konvergensi

Waktu konvergensi adalah waktu yang dibutuhkan sebuah jaringan untuk mencapai keadaan *steady state* pada semua *link*. Pengukuran waktu konvergansi dimulai dari keadaan jalur data terhubung dengan jalur kontrol sampai semua *link* terhubung dengan pengecekan menggunakan perintah *pingall*. Berdasarkan Gambar 6, hasil dari 30 k ali pengujian didapatkan waktu rata-rata yang dibutuhkan jaringan untuk mencapai keadaan *steady state* yaitu 17.71446 detik.

Untuk pemutusan *link*, pengujian ini dilakukan dengan mengirimkan paket *ping* dari satu *host* ke *host* lain, lalu ditengah pengiriman paket *ping* dilakukan pemutusan *link* di rute yang dipilih. Lalu pada kontroler diatur waktu dari *idle timeout* dan *hard timeout*. Pada Gambar 7 menunjukan waktu yang dibutuhkan untuk menemukan jalur yang baru ketika ada pemutusan *link*.

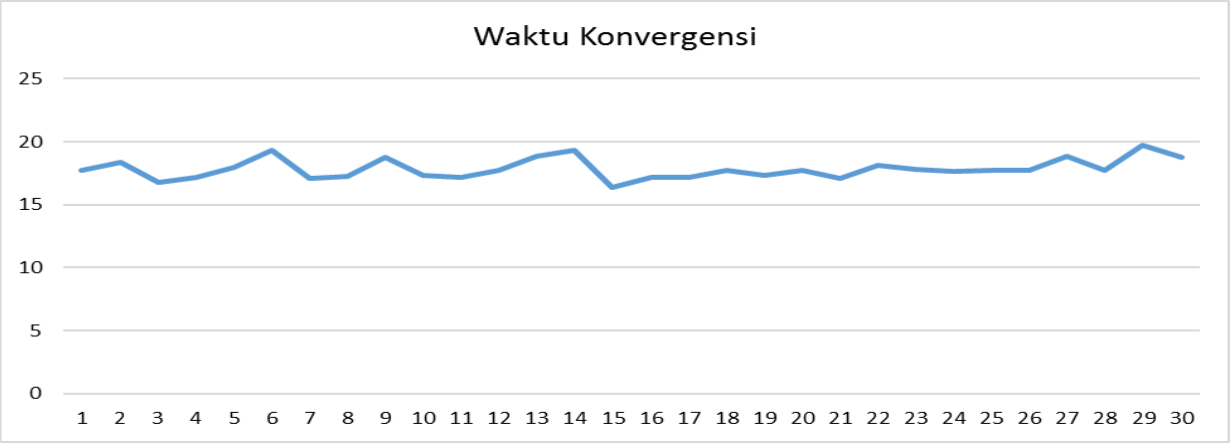


Gambar 7. Waktu Terbentuknya Rute Baru

Dari hasil yang didapat semakin kecil *idle timeout* semakin cepat waktu yang dibutuhkan untuk menemukan jalur baru. Dikarenakan semakin kecil *idle timeout* semakin cepat juga *flow table* yang ada di-*switch* untuk melakukan pembaruan ketika *flow* tersebut tidak digunakan.

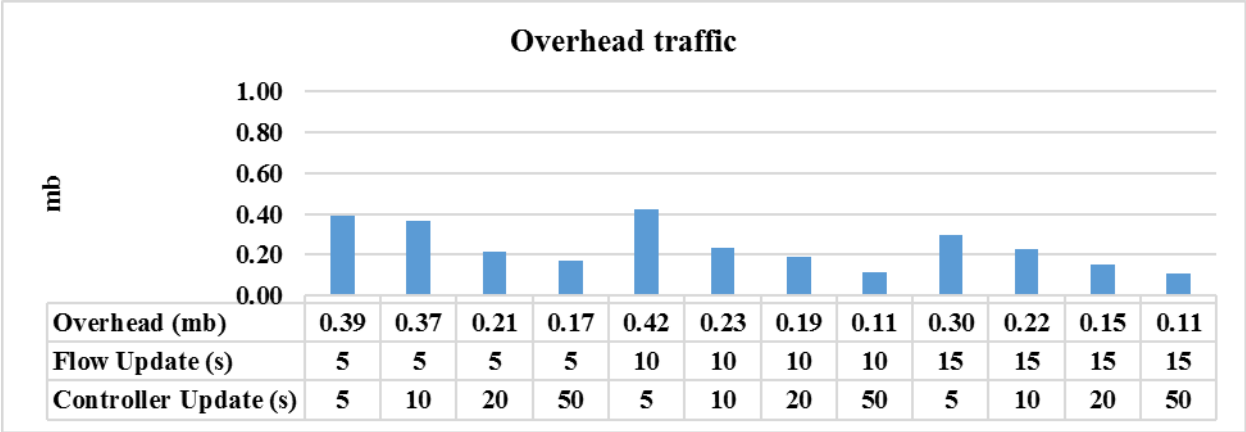
2) Overhead Traffic

Dalam pengujian *overhead traffic* pengukuran *overhead* dilakukan menggunakan Wireshark untuk melihat apakah pengaruh *controller update* dan *flow update* terhadap besarnya *overhead traffic*. Dapat dilihat pada Gambar 8, hasil dari pengujian *overhead traffic* terhadap *controller update* dan *flow update*. Dimana ketika waktu *controller update* di atur waktu *update* dengan variasi waktu 5, 10, 20, dan 50 de tik terlihat perubahan pada *overhead* yang dihasilkan. Didapat hasil semakin cepat waktu kontroler untuk melakukan *update* semakin banyak juga *overhead* yang ada. Begitu juga kebalikannya semakin lama waktu *controller update* semakin sedikit *overhead* yang didapat. Hal ini dikarenakan ketika kontroler melakukan *update* maka kontroler akan mengirimkan paket stat_request dan akan nerima stat_reply dari *switch* sehingga semakin sering kontroler melakukan *update* semakin sering juga kontroler mengirim paket stat_request dan menerima stat_reply yang membuat semakin besar juga *overhead* dari *trafic*.



Gambar 6. Waktu Konvergensi

e-ISSN: 2527-9955
p-ISSN: 1411-8289

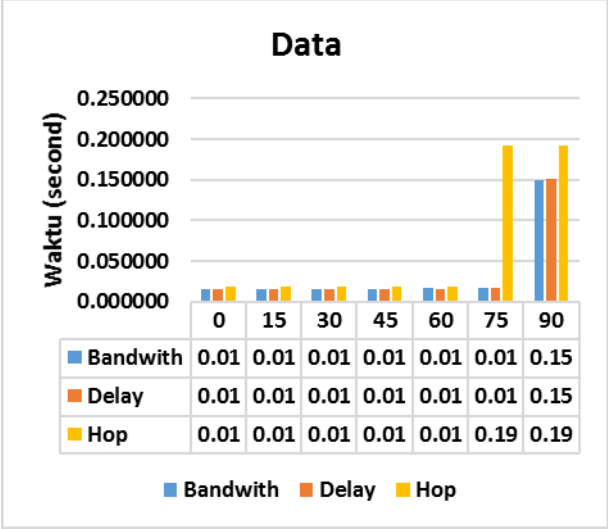


Overhead (mb)	0.39	0.37	0.21	0.17	0.42	0.23	0.19	0.11	0.30	0.22	0.15	0.11
Flow Update (s)	5	5	5	5	10	10	10	10	15	15	15	15
Controller Update (s)	5	10	20	50	5	10	20	50	5	10	20	50

Gambar 8. *Overhead Traffic*

3) QoS (Delay dan Packet Loss)

Dalam pengujian *QoS* berdasarkan *metric* yang berbeda menggunakan beberapa parameter antara lain perhitungan *metric* OSPF yang berdasarkan *bandwidth*, *metric* berdasarkan *delay*, dan *metric* berdasarkan *hop*. *Delay* adalah waktu yang dibutuhkan oleh sebuah paket untuk melakukan perjalanan dari satu *node* awal ke *node* tujuan. *Delay* yang diukur adalah *one-way delay*. Dengan jenis layanan yang akan dijalankan adalah data, video, dan VoIP. Dengan skenario *bandwidth* dan *delay* yang telah di atur.

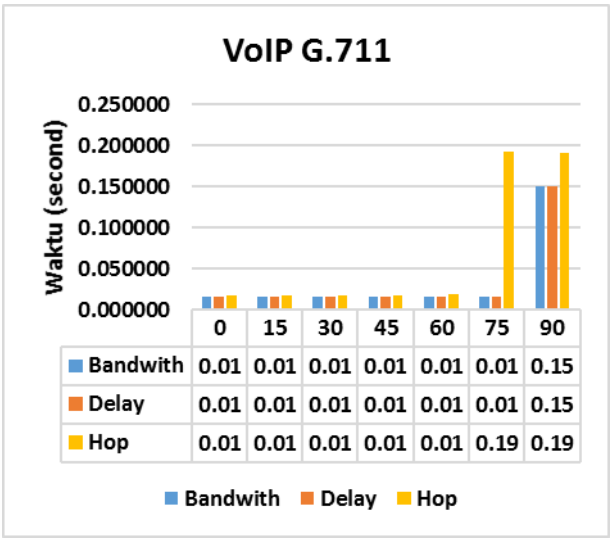


	0	15	30	45	60	75	90
Bandwith	0.01	0.01	0.01	0.01	0.01	0.01	0.15
Delay	0.01	0.01	0.01	0.01	0.01	0.01	0.15
Hop	0.01	0.01	0.01	0.01	0.01	0.19	0.19

Bandwith Delay Hop

Gambar 9. Hasil *Delay* Data

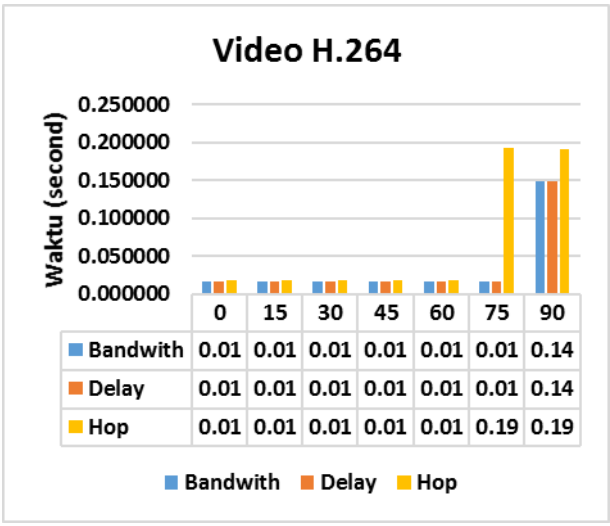
Berdasarkan hasil yang dapat dilihat pada Gambar 9, Gambar 10 dan Gambar 11 pengujian *delay* dengan layanan data, video, dan VoIP. Hasil yang didapat ketika pemberian *background traffic* di bawah dari kapasitas *link* yang ada, pada penggunaan *metric delay* menghasilkan *delay* yang lebih kecil dibandingkan dengan *metric* OSPF dan *metric hop*. Hal ini dikarenakan pada *metric delay* perhitungan untuk menentukan jalur menggunakan *delay* pada *link* yang paling terkecil sesuai dengan algoritma Floyd-Warshall. Namun ketika pemberian *background traffic* yang diberikan melebihi dari kapasitas *link* penggunaan *metric* OSPF lebih memiliki nilai *delay* yang lebih kecil dibandingkan dengan *metric delay* dan *metric hop*. Pada *metric hop* mengalami kenaikan *delay* yang signifikan pada pemberian *backround traffic* 75 Mbps karena pada perhitungan jalur menggunakan *metric hop* melewati *link* yang memiliki kapasitas maksimal sebesar 70 Mbps dan kenaikan pada pemberian *background traffic* 90 Mbps pada setiap *metric* dikarenakan setiap *metric* melewati *link* yang memiliki kapasitas maksimal 90 Mbps.



	0	15	30	45	60	75	90
Bandwith	0.01	0.01	0.01	0.01	0.01	0.01	0.15
Delay	0.01	0.01	0.01	0.01	0.01	0.01	0.15
Hop	0.01	0.01	0.01	0.01	0.01	0.19	0.19

Bandwith Delay Hop

Gambar 10. Hasil *Delay* VoIP

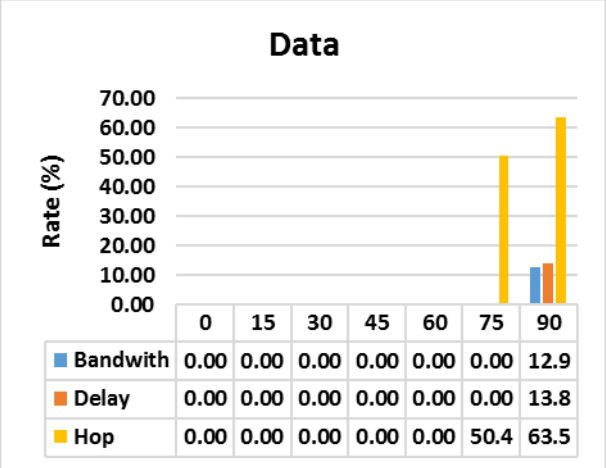


	0	15	30	45	60	75	90
Bandwith	0.01	0.01	0.01	0.01	0.01	0.01	0.14
Delay	0.01	0.01	0.01	0.01	0.01	0.01	0.14
Hop	0.01	0.01	0.01	0.01	0.01	0.19	0.19

Bandwith Delay Hop

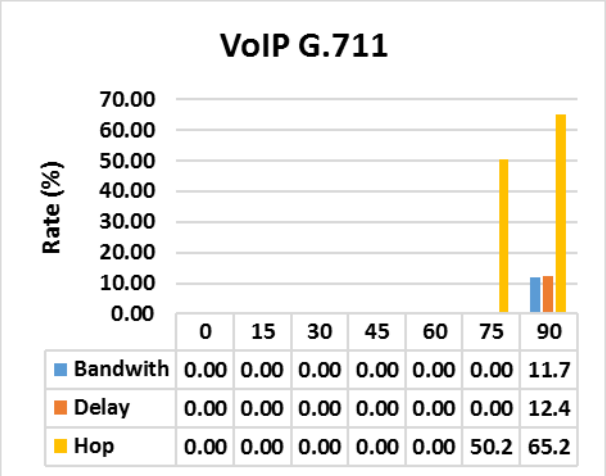
Gambar 11. Hasil *Delay* Video

Packet loss adala banyaknya paket gagal yang dikirimkan dibandingkan dengan banyaknya paket yang dikirim.



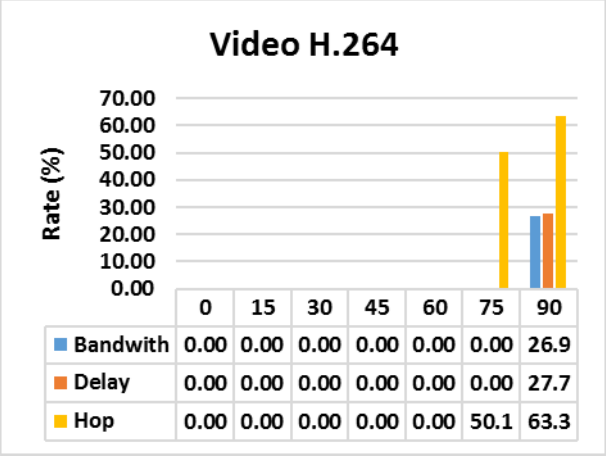
	0	15	30	45	60	75	90
Bandwith	0.00	0.00	0.00	0.00	0.00	0.00	12.9
Delay	0.00	0.00	0.00	0.00	0.00	0.00	13.8
Hop	0.00	0.00	0.00	0.00	0.00	50.4	63.5

Gambar 12. Hasil *Packet Loss* Data



	0	15	30	45	60	75	90
Bandwith	0.00	0.00	0.00	0.00	0.00	0.00	11.7
Delay	0.00	0.00	0.00	0.00	0.00	0.00	12.4
Hop	0.00	0.00	0.00	0.00	0.00	50.2	65.2

Gambar 13. Hasil *Paket Loss* VoIP



	0	15	30	45	60	75	90
Bandwith	0.00	0.00	0.00	0.00	0.00	0.00	26.9
Delay	0.00	0.00	0.00	0.00	0.00	0.00	27.7
Hop	0.00	0.00	0.00	0.00	0.00	50.1	63.3

Gambar 14. Hasil *Packet Loss* Video

Pada Gambar 12, Gambar 13, dan Gambar 14 menunjukan nilai *packet loss* dari hasil pengujian pada layanan data, *VoIP* dan video. Hasil pengujian menunjukan munculnya *packet loss* terjadi ketika ada penambahan *background traffic* sebesar 75 M bps dengan menggunakan *metric* berdasarkan *hop* karena pada penentuan jalur menggunakan *metric hop* melewati *link* yang memiliki kapasitas 70 M bps. Kemudian meningkat ketika ditambah *background traffic* sebesar 90 Mbps pada semua *metric* yang diujikan. Semakin besar *background traffic* yang ditambahkan semakin besar juga persentasi *packet loss* yang terjadi. Hal ini dikarenakan *packet loss* dipengaruhi oleh kondisi kapasitas dari *link* dan juga kepadatan pada *link*.

KESIMPULAN

Uji performasi algoritma Floyd-Warshall sebagai penentuan jalur terbaik dapat disimulasikan pada jaringan SDN dengan menggunakan kontroler Ryu dan Mininet sebagai bagian mensimulasikan jalur data, dengan hasil evaluasi performansi sebagai berikut :

1. Nilai waktu *link failure* yang dihasilkan ketika terjadinya pemutusan *link* menunjukan hasil yang lebih cepat dibandingkan dengan waktu konvergensinya, dikarenakan jumlah *link* yang terdapat pada topologi berkurang sehigga algoritma lebih cepat dalam menentukan jalur.
2. *Overhead traffic* yang didapat berbanding lurus dengan penambahan jumlah *switch* pada jalur data. Dimana semakin banyak jumlah *switch* semakin banyak juga jumlah *overhead traffic* yang ada.
3. Hasil QoS pada pengujian beda *metric routing* dengan parameter *delay* dan *packet loss* masih dalam rentang nilai yang ditetapkan pada ITU-T G.1010 [12] ketika diberikan *background traffic* masih di bawah kapasitas *link*.

DAFTAR PUSTAKA

[1] Mulyana, E. *Buku Komunitas SDN-RG.* Bandung: GitBook, 2014.

[2] Setyawati Handaka. M, "Perbandingan Algoritma Dijkstra (Greedy), Bellman-Ford (BFS-DFS), dan Floyd-Warshall (Dynamic Programming) dalam Pengaplikasian Lintasan Terpendek pada Link-State Routing Protocol," *Journal,* Bandung: IT, 2010.

[3] V. Listiani, *Analisis Peformansi SDN (Software Defined Network) Menggunakan Protokol Routing OSPF (Open Shortes Path First)*, Bandung, 2015.

[4] Göransson, P., & Black, C, *Software Defined Networks A Comprehensive Approach.* United State of America: Morgan Kaufmann, 2014.

[5] Open Networking Foundation., "OpenFlow Switch Specification: Version 1.0.0 (Wire Protocol 0x01).," 2011.

[6] B. Lantz, B. Heller and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks,"*in Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pp. 19, 2010.

[7] Cisco. (2005, August). OSPF Design Guide. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/7039-1.html

[8] A. F. Sani, N. K. T. Tastrawati and I. M. E. Dwipayana, "Algoritma Floyd-Warshall Untuk Menentukan Jalur Terpendek Dalam Evakuasi Tsunami di Kelurahan Sanur," *E-Jurnal Matematika,* vol. 2, no. 1, pp. 1-5, 2003.

[9] UMTS Forum, UMTS/IMT -2000 Spectrum.

[10] Cisco. (2016, April). Voice Over IP - Per Call Bandwidth Consumptio. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html#anc8

[11] A, S, Donato, E., Pescape, A., & Ventre, G, A Practical Demonstration of Network Traffic Generation, in *Proceedings of The 8th IASTED International Conference.* Hawaii, 2004, p 138-143.

[12] Series G: transmission systems and media, digital systems and networks. Quality of service and performance. End-user multimedia QoS categories. International Telecommunication Union, 2001.