ANALISIS EFEK PENGGUNAAN KONTROLER FLOODLIGHT DAN OPENDAYLIGHT PADA PERFORMANSI JARINGAN SDN

ANALYISIS OF EFFECT OF CONTROLLER FLOODLIGHT AND OPENDAYLIGHT ON SOFTWARE DEFINED NETWORK

Krisandi Agusto[1], Ir. Agus Virgono., M.T.[2], Drs. Ir. R Rumani M., Bc.TT., M.Sc.[3]

[1,2,3]Fakultas Teknik Elektro, Universitas Telkom, Bandung

[1]krisandia.ka@gmail.com, [2]avirgono@telkomuniversity.ac.id, [3]rumani@gmail.co.id

Abstrak

Pada jaringan tradisional, penggunaan antara *control plane* dan *forwarding plane* menjadi satu. Desain jaringan seperti ini dianggap kurang fleksibel dalam mengkontrol dan mengatur jaringan tersebut. Ada suatu sistem jaringan yang dianggap lebih efektif dari pada jaringan konvensional yang sudah ada yang disebut dengan *Software Defined Network* (SDN). SDN berfungsi untuk memisahkan secara eksplisit antara *control plane* dengan *data plane* supaya terjadi pemanajemenan jaringan yang lebih bagus. Dengan adanya sistematika ini pengelolaan jaringan akan lebih baik dan lebih fleksibel melalui kontroler yang bersifat *programmable.*

Ada banyak jenis kontroler dengan menggunakan bahasa pemrograman berbeda. Beberapa kontroler ada yang menggunakan bahasa pemrograman java dan sebagian menggunakan bahasa pemrograman python. Pada pengujian ini, kontroler yang digunakan adalah kontroler dengan bahasa pemrograman java. Adapun kontroler yang digunakan yaitu kontroler Floodlight tanpa algoritma Johnson, Foodlight dengan algoritma Johnson dan OpenDaylight. Ketiga model kontroler ini akan diuji performanya menggunakan parameter QoS dengan standarisasi ITU-T G.1010.

Hasil penelitian yang didapat kontroler Floodlight tanpa algoritma Johnson lebih unggul dibandingkan kedua kontroler lainnya dari untuk layanan data dan VoIP. Namun untuk pengiriman layanan video, kontroler OpenDaylight lebih baik dari yang lainnya. Karena hanya kontroler OpenDaylight yang bisa mengirimkan paket video. Untuk pengujian *resource utilization*, konsumsi memori tertinggi dimiliki oleh kontroler OpenDaylight. Berdasarkan standarisasi ITU-T G.1010 kontroler Floodlight tanpa algoritma Johnson, Floodlight menggunakan algoritma Johnson dan OpenDaylight hanya memenuhi standarisasi *delay* untuk layanan data saja.

Kata Kunci: Kontroller, Kinerja, Floodlight, OpenDaylight, QoS, *Resource Utilization*

Abstract

In traditional networks, the use between control plane and forwarding plane becomes one. Network design like this is considered to be less flexible in controlling and managing the network. There is a network system that is considered more effective than existing conventional networks called Software Defined Network (SDN). SDN serves to separate explicitly between the control plane and the data plane so that there is better network management. With this systematics network management will be better and more flexible through programmable controllers.

There are many types of controllers using different programming languages. Some controllers use the Java programming language and some use the Python programming language. In this test, the controller used is a controller with a java programming language. The controllers used are Floodlight controllers without Johnson algorithm, Foodlight with Johnson algorithm and OpenDaylight. These three controller models will be tested for their performance using QoS parameters with ITU-T G.1010 standardization.

The results obtained by the Floodlight controller without the Johnson algorithm are superior to the other two controllers for data and VoIP services. But for sending video services, OpenDaylight controllers are better than others. Because only OpenDaylight controllers can send video packages. For testing resource utilization, the highest memory consumption is owned by the OpenDaylight controller. Based on the ITU-T G.1010 standardization of Floodlight controllers without the Johnson algorithm, Floodlight using Johnson and OpenDaylight algorithms only meets the delay standard for data services only.

Keywords: Controller, Performance, Floodlight, OpenDaylight, QoS, Resource Utilization

1. Pendahuluan

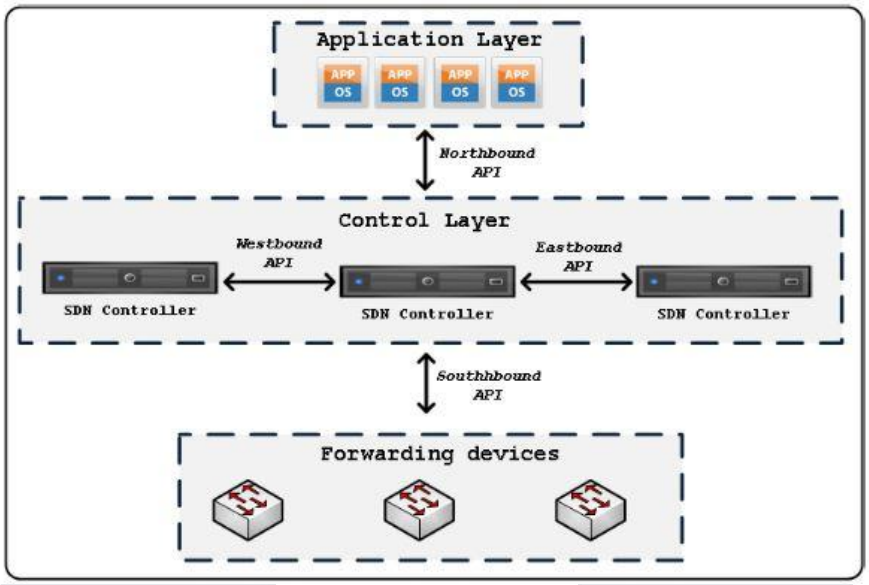
Software Define Network (SDN) merupakan paradigma baru pada teknologi jaringan yang memisahkan control plane dan data plane pada jaringan[1]. Control plane dipisahkan dari data plane dan dipindahkan menuju server terpusat yang disebut dengan kontroler. Peran utama kontroler adalah mengatur rules atau membuat

keputusan, dan peran utama dari control plane adalah secara langsung meneruskan paket berdasarkan rules yang datang dari kontroler[5]. Paradigma seperti ini membuat jaringan lebih efektif dan efisien.

Peran kontroler sangat penting pada jaringan SDN. SDN pada umumnya menggunakan suatu protokol yang disebut dengan protokol OpenFlow. Masih banyak lagi protokol yang ada selain OpenFlow. Tipe kontroler pada jaringan SDN juga yang didukung olej protokol OpenFlow banyak macamnya. Beberapa jenis kontroler tersebut diantaranya adalah POX, NOX, Ryu, Beacon, Maestro, Floodlight, OpenDaylight dan sebagainya. Setiap kontroler diatas memiliki kelamahan dan kelebihannya. Kelemahan dan kelebihannya itu bisa dilihat dari berbagai segi, salah satunya yaitu dari segi performansinya.

Ada beberapa aspek yang mempengaruhi performansi kontroler. Bahasa pemrograman, topologi dan environment simulasi juga mempengaruhi bagaimana performansi kontroler tersebut. Hal inilah yang akan diujikan nantinya dan akan dicoba mengoptimalkan menggunakan salah satu algoritma yang akan dipilih nantinya agar bisa lebih optimal nantinya dalam melakukan traffic.

2. Dasar Teori



Gambar 2.1 Arsitektur Jaringan SDN

Pada gambar 2.1 terdapat arsitektur jaringan SDN, untuk menerapkan jaringan SDN tersebut dibutuhkan emulator mininet yang berfungsi sebagai forwarding plane dan suatu kontroler yang berfungsi sebagai control plane yang dilengkapi oleh protokol openflow.

2.1. Protokol OpenFlow[2]

OpenFlow adalah standar komunikasi antarmuka pertama yang mendefinisikan antara *control plane* dengan *data plane* pada arsitektur jaringan SDN. Untuk menjalankan fungsinya, protokol *OpenFlow* harus bertindak sebagai Kontroler (controller), berfungsi sebagai *server OpenFlow* yang berperan sebagai *control plane*. Penerus (*forwarder*), berfungsi sebagai perangkat yang dijalankan sebagai *data plane*. Dan *OpenFlow flow table*, tabel penerusan yang tidak tergantung pada jenis layanan. Setiap *flow table* mengandung kolom kecocokan dan tindakan terkait. *Forwarder* mencocokkan paket dengan bidang tertentu dalam flow table dan melakukan tindakan spesifik yang terkait dengan bidang pada paket yang cocok.

2.2. Kontoller Floodlight[7]

Kontroller Floodlight merupakan controller dengan kelas enterprise, memiliki lisensi Apache, dan memilik Bahasa pemrograman berbasis java yang dikembangkan oleh Big Switch Network. Floodlight didukung oleh feature OpenFlow yang membuatnya bisa mengatur aliran data pada lingkungan SDN

2.3. Kontroler OpenDaylight[6]

Proyek OpenDaylight terbentuk atas konsorsium Linux Foundation. Kontroler ini didukung oleh standar northbound API, jadi tidak hanya didukung dengan protokol southbound API seperti OpenFlow, I2RS dan NETCONF yang bisa diprogram. Beberapa fitur kontroler ini adalah Java-based (OSGI), modular, pluggable dan juga didukung oleh banyak protokol southbound. OpenDaylight dimanfaatkan dalam jenis jaringan berskala besar.

2.4. Mininet

Mininet merupakan suatu emulator yang memperbolehkan penggunanya secara berulang-ulang untuk membuat prototipe suatu jaringan dalam skala besar menggunakan satu komputer[3]. Fitur pada mininet memperbolehkan pembuatan, komunikasi, kustom dan membagikan prototipe jaringan secara cepat dalam bentuk mekanisme virtualisasi. Topologi awal pada mininet memiliki OpenFlow Kernel Switch yang menghubungkan dua host dengan sebuah kontroler openflow[8].

2.5. Algoritma Johnson[4]

Algoritma Johnson adalah algoritma untuk menyelesaikan *shortest path problem* dengan orientasi mencari jarak terdekat sekaligus bobot paling sedikit. Algoritma Johnson merupakan perpaduan anatara algoritma Bellman Ford dan Algoritma Dijkstra.

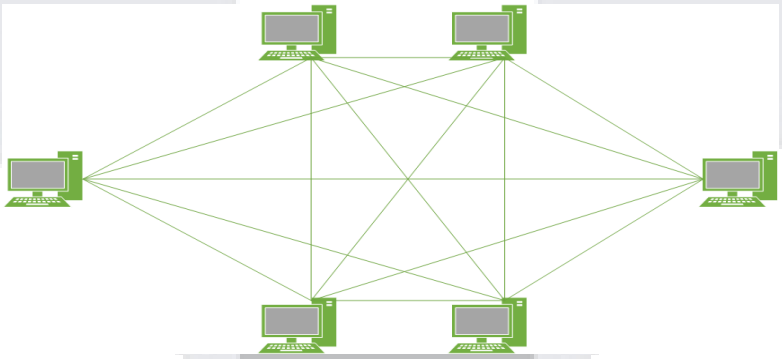
Penyelesaian Algoritma Johnson adalah mengonstruksi graf baru dengan menambahkan titik baru (s) pada graf dan memberi bobot sisi yang keluar dari titik baru tersebut dengan angka 0 seperti Gambar 2.3(a). Langkah selanjutnya adalah mencari jarak terpendek d(s, dari titik s kesemua titik lain, jarak terpendek tersebut digunakan untuk mengubah bobot sisi bernilai positif ŵ(u,v) dengan cara[2] :

$$\hat{\omega}(u,v) = \omega(u.v) + d(s,u) - d(s,v) \text{ [2.1]}$$

Setelah itu titik baru (s) tadi dihapus dan algoritma dijkstra digunakan untuk mencari jalur terpendek dari node sumber ke setiap node lain pada graph baru yang sudah diubah bobotnya menjadi positif.

2.6. Topologi Mesh[5]

Merupakan topologi jaringan komputer yang menghubungkan semua komputer secara penuh, topologi ini adalah topologi yang paling komplek dibanding dengan topologi jaringan lainnya. Topologi jenis ini banyak digunakan oleh penyedia layanan internet (ISP). Konsep dari topologi ini adalah setiap komputer dalam jaringan saling terhubung satu sama lain sehingga jika terjadi kerusakan pada salah satu komputer tidak berpengaruh pada komputer lain atau berpengaruh pada jaringan.



Gambar 2.2 Topologi Mesh

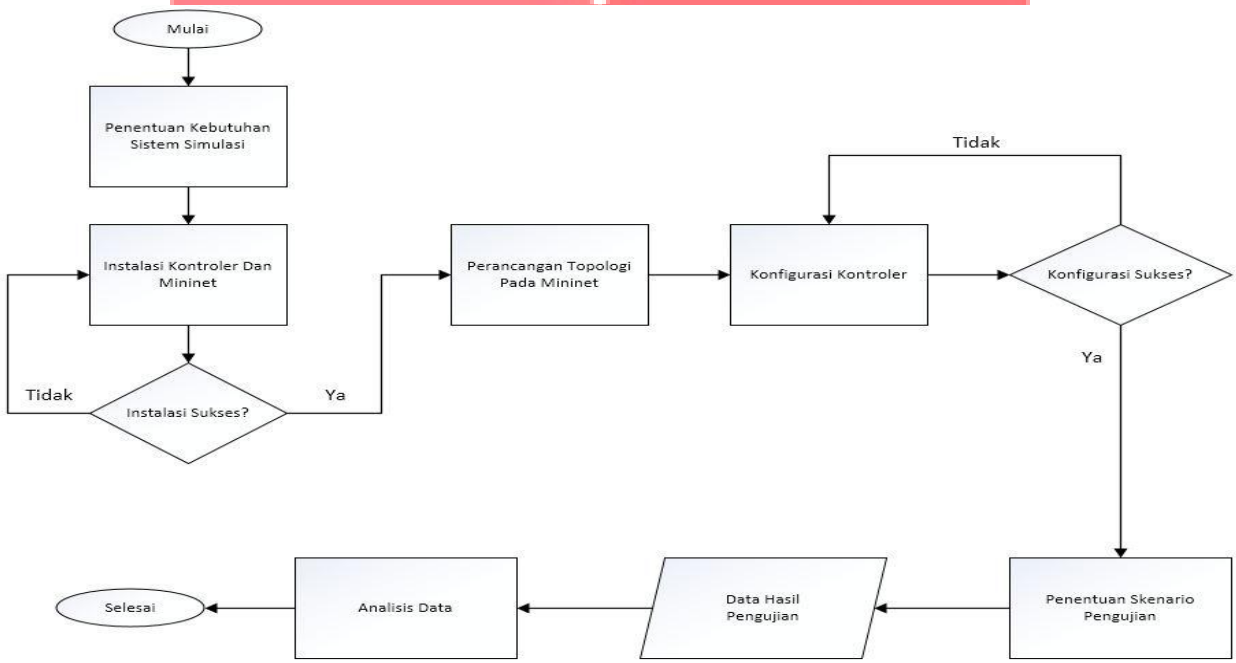
Jenis koneksi pada topologi jaringan mesh terdiri dari 2 jenis, kedua topologi mesh tersebut meliputi:

- Topologi Mesh Fully Connected mempunyai ciri utama dimana setiap komputer dalam jaringan saling terhubung satu sama lain secara penuh. Sebagai contoh jika ada 5 komputer dalam jaringan tersebut maka satu komputer akan terhubung ke 4 komputer lainnya.
- Topologi mesh partial connected topologi jenis ini memiliki ciri yaitu setiap komputer dalam jaringan tersebut tidak semua komputer akan terhubung dengan komputer lainnya sehingga ada beberapa komputer yang saling terhubung satu sama lain dan beberapa komputer tidak saling berhubungan

3. Pembahasan

3.1. Model Sistem Simulasi

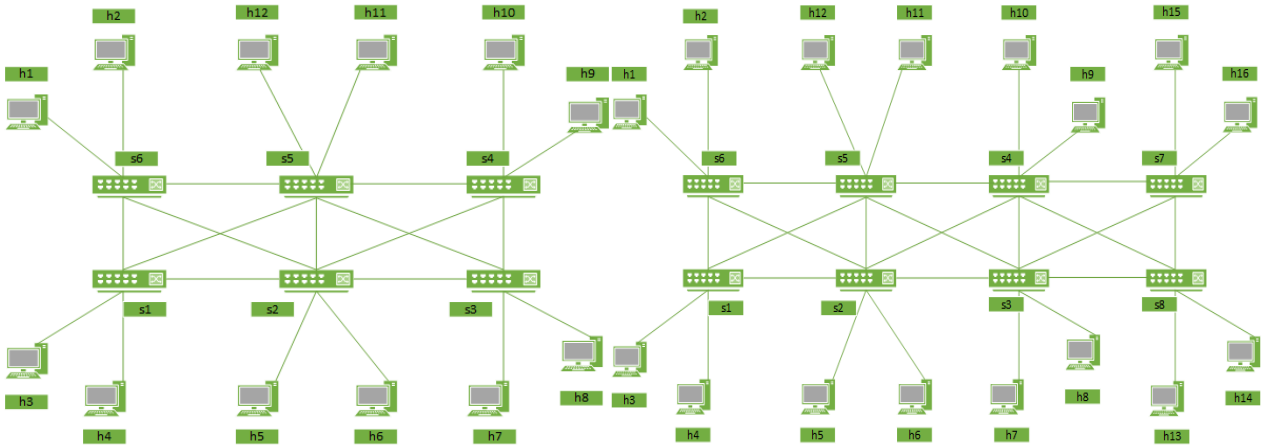
Dimulai dari menentukan kebutuhan untuk melakukan simulasi, lalu melakukan instalasi mininet sebagai emulator dan kontroler yang dibutuhkan yaitu Floodlight dan OpenDaylight. Setelah instalasi sukses, akan dilakukan perancangan topologi sesuai dengan topologi yang sudah dirancang. Lalu untuk pengujian pertama running topologi tanpa melakukan konfigurasi pada kontroler. Setelah itu lakukan penentuan variabel pengujian, pengambilan data, analisis data dan dapatkan hasilnya. Jika sudah selesai, simpan datanya dan lakukan pengujian kepada kontroler yang sudah dikonfigurasi. Lalu lakukan hal yang sama seperti hal sebelumnya hingga didapatkan hasilnya. Setelah kedua-duanya mendapatkan hasil lalu dibandingkanlah hasilnya, mana yang lebih efektif. Flowchart sistem simulasi digambarkan pada gambar 3.1



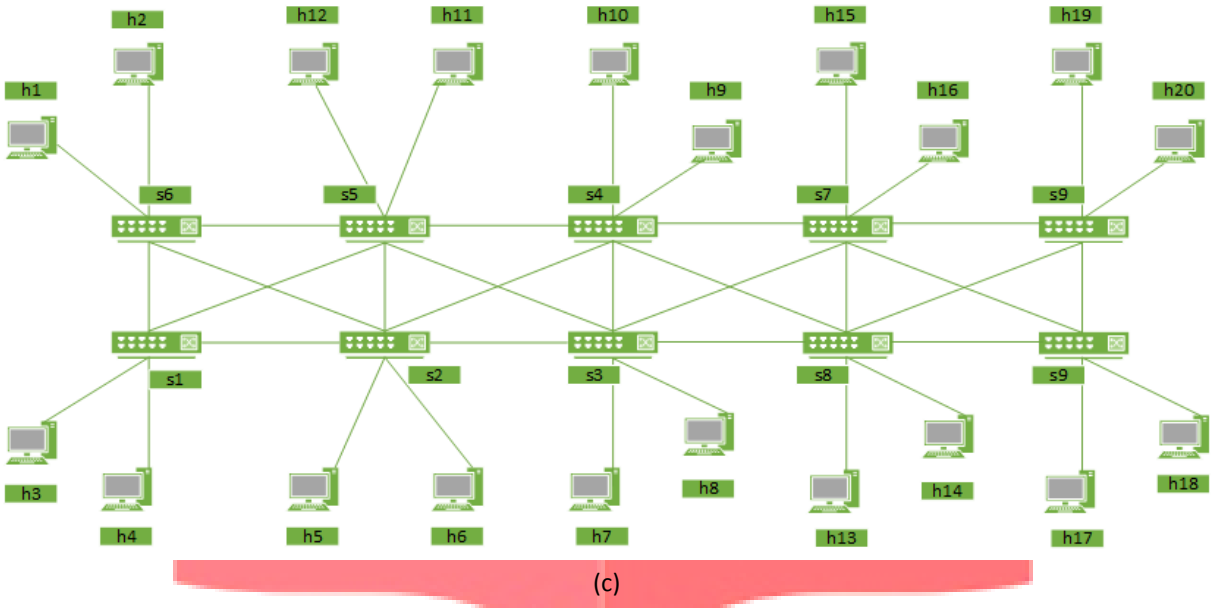
Gambar 3.1 Flowchart Model Sistem Simulasi

3.2. Konfigurasi Forwarding Plane

Pengujian dilakukan menggunakan parameter QoS (*delay*, *jitter*, *throughput* dan *packet loss*) dan resource utilization. Topologi yang digunakan adalah topologi mesh yang sudah dimodifikasi sesuai gambar 3.2 berikut:



(a) (b)



Gambar 3.2 (a) Topologi 6 *switch* 12 *host*, (b) Topologi 8 *switch* 16 *host*, (c) Topologi 10 *switch* 20 *host*

Pengujian *quality of service* menggunakan topologi dengan jumlah switch 6, 8 dan 10. Hal ini dilakukan untuk mengetahui pengaruh penambahan jumlah switch terhadap kinerja routing. Percobaan akan dilakukan sebanyak 30 kali dengan membangkitkan trafik UDP. Trafik UDP berupa data, video streaming, dan Voip dihasilkan dari D-ITG dengan spesifikasi sebagai berikut:

1. Trafik Data dengan spesifikasi inter-departure time (IDT) konstan = 100 pps , ukuran paket yang terdistribusi poisson dengan μ = 38,4 Bytes dengan waktu 20000milisecond
2. Video streaming 24 frame (satu paket per frame) per sekon, ukuran paket yang terdistribusi normal dengan μ = 27791 Bytes dan σ^2 = 6254 Bytes, dan membutuhkan throughput sekitar 5,336 Mbps.
3. VoIP menggunakan G.711 codec tanpa voice activation detection (VAD) sebanyak 100 pps yang berukuran paket 80 Bytes, dan membutuhkan throughput sekitar 70,4 Kbps.

Pada pengujian ini dilakuan variasi pengujian dengan menambahkan background trafik dengan bantuan iperf. Terdapat 5 jenis backround trafik yang digunakan yaitu 0Mbps, 20Mbps, 40Mbps, 60Mbps, dan 80Mbps. Pengujian ini dilakukan tidak menggunakan jaringan GigabitEthernet karena skala jaringan yang digunakan masih kecil. Terlebih untuk pengujian menggunakan GigabitEthernet memerlukan spesifikasi computer *high-end.*

Pengujian resource utilization dilakukan ketika jaringan dalam kondisi steady state. Lalu untuk memeriksa konsumsi memori yang digunakan oleh kontroler tersebut bisa dilakukan dengan mengetikkan command top pada terminal ubuntu. Temukan proses kontroler yang sedang berjalan dan nantinya akan diperlihatkan konsumsi memori dalam bentuk persen. Untuk mengetahui jumlah konsumsi memori dalam MB, maka dibutuhkan perhitungan sebagai berikut:

%memori x total ram

Keterangan:
%memori: konsumsi memori dalam persen
total RAM: jumlah RAM yang dialokasikan untuk *operating system*

3.3. Hasil Pengujian

3.3.1. Pengujian *QoS*

Pada layanan data secara umum kontroler floodlight tanpa algoritma johnson cenderung mengalami sedikit penurunan *delay* ketika jumlah *background traffic* ditambahkan . Walau terjadi sedikit fluktuasi dari bandwith 0Mbps ke 20Mbps, namun ini memperlihatkan karakteriskiknya. Hal yang sama tidak berlaku untuk kontroler floodlight dengan algoritma johnson. Pada hasil pengujian *delay* menggunakan layanan data diatas, peningkatan jumlah *switch* mempengaruhi nilai *delay* data. Untuk controller OpenDaylight kenaikan jumlah *switch*

dan trafik berbanding lurus dengan delay datanya. Setiap terjadi penambahan *switch*, nilai *delay* meningkat dan ketika terjadi kenaikan nilai trafik *delay*-nya cendrung naik. Perubahan yang agak signifikan terlihat ketika kontroler floodlight berada pada trafik 60Mbps ke 80Mbps. Itu terjadi kaerna bandwith link sudah mendekati ambang. Jika kita menggunakan referensi standarisasi QoS ITU.T G.1010, tiga kontroler diatas memenuhi standarnya, karena nilai *delay*-nya tidak lebih dari *60 second.* Nilai *delay* terkecil dimiliki oleh kontroler floodlight tanpa algoritma johnson.

Untuk layanan VoIP, kontroler floodlight tanpa algoritma johnson mengalami kenaikan *delay* ketika jumlah *host* nertambah. Semakin bertambah jumlah *host*, *delay*-nya akan naik. Untuk perubahan nilai background traffic tidak terlalu mempengaruhi *delay*-nya. Begitu juga yang terjadi pada kontroler floodlight ketika ditambahkan algoritma johnson dan OpenDaylight tanpa algoritma johnson. Terjadi kenaikan *delay* ketika jumlah *host* bertambah. Kontroler OpenDaylight mengalami cendrung mengalami peningkatan *delay* ketika *background traffic*-nya meningkat. Berdasarkan standarisasi QoS ITU.T G.1010. kontroler OpenDaylight tidak memenuhi standar ini. Karena *delay*-nya ada yang menyentuh nilai diatas 400ms.

Pengujian *delay* pada layanan video, hanya bisa dilakukan pada kontroler OpenDaylight. Karena floodlight tidak bisa mengirim packet dalam jumlah besar. Ini bug yang masih belum diperbaiki oleh developer kontroler floodlight sekarang. Berdasarkan hasil pengujian, penambahan jumlah *host* berdampak pada kenaikan nilai *delay*. Jika jumlah *host* bertambah, maka nilai *delay* akan naik. Sorotan khusus pada kontroler OpenDaylight ketika memiliki *background traffic* 60Mbps dan 80Mbps. Topologi dengan model 10 *switch* tidak bisa mengirimkan data video karena mengalami *hang* pada PC yang digunakan. Ini terjadi karena *background traffic* yang sudah mendekati bandwith ambang pada link dan jumlah *switch* yang banyak dengan ukuran data yang besar. Jadi terjadilah keantrian dalam mengirim data, selama antrian juga akan memakan resource lebih banyak dan menyebabkan PC menjadi *hang* dan menyebabkan pengiriman data tidak bisa diteruskan.

Berdasarkan hasil pengujian diatas, nilai *jitter* pada layanan data meningkat seiring bertambahnya jumlah *host*. Perubahan *jitter* tiap perubahan *background traffic*-nya tidak terlalu signifikan. Namun nilai *jitter* terkecil dipegang oleh kontroler floodlight tanpa algoritma johnson. Berdasarkan standarisasi QoS ITU.T G.1010. ketiga kontroler ini tidak memenuhi standar. Karena *jitter*-nya melebihi angka 1ms.

Pada pengujian *jitter* untuk layanan VoIP, secara umum ketiga model kontroler mengalamami kenaikan nilai jitter saat terjadi penambahan jumlah host. Jika dilihat dari standarisasi QoS ITU.T G.1010, tidak ada satupun kontroler yang memenuhi standarisasi tersebut. Karena nilai jitternya lebih dari 1ms. Kontroler floodlight tanpa algoritma johnson lebih baik dalam nilai jitternya karena kontroler ini memiliki nilai jitter terkecil diantara dua kontroler lainnya.

Untuk pengujian jitter pada layanan video hanya bisa dilakukan pada kontroler opendaylight, karena kontroler floodlight tidak bisa meneruskan paket dengan ukuran besar. Jika diteruskan, nilai keterangan jitternya akan memiliki keluaran *nan*. Ini bug yang belum diperbaiki oleh *developer* floodlight. Untuk kontroler opendaylight perubahan jumlah *host* mempengaruhi kenaikan nilai *jitter*. Semakin bertambah jumlah host maka jitternya akan bertambah. Sama seperti pada layanan VoIP, terjadi *hang* pada komputer percobaan saat dilakukan pengujian pada model topologi 10 switch dengan *bandwith* 60Mbps dan 80Mbps. Ini terjadi karena *background traffic* yang sudah mendekati bandwith ambang pada link dan jumlah *switch* yang banyak dengan ukuran data yang besar. Jadi terjadilah keantrian dalam mengirim data, selama antrian juga akan memakan resource dan menyebabkan PC menjadi *hang* dan menyebabkan pengiriman data tidak bisa diteruskan.

Untuk perubahan nilai *throughput* pada layanan data, VoIP dan video di ketiga kontroler tersebut masih konstan. Secara umum, perubahan jumlah *host* dan kenaikan *background traffic* tidak mempengaruhi nilai thorughputnya. Hal ini dikarenakan besar *background traffic* masih dibawah besar maksimal yang disediakan *link*. Berdasarkan standarisasi QoS ITU.T G.1010 tidak ada kontroler yang memenuhi standar tersebut karena nilai throughput-nya lebih dari 64kbps. Semua data hasil pengujian QoS tertera pada lampiran.

3.3.2 Pengujian *Resource Utilization*

Tujuan dilakukannya pengujian *resource utilization* adalah untuk mengetuhui seberapa besar konsumsi memori yang digunakan oleh kontroler ketika belum memiliki Algoritma Johson dan ketika sudah memiliki Algrotima Johnson didalamnya. Berdasarkan hasil pengujian, secara umum terjadi peningkatan konsumsi memori pada masing-masing kontroler pada setiap peningkatan jumlah switch. Kenaikan jumlah switch berbanding lurus

dengan konsumsi memori oleh kontroler/ Kontroler OpenDaylight lebih terbilang paling banyak memakan memori sebesar 782.336MB pada jumlah switch 10. Semua data hasil pengujian resource utilization tertera pada lampiran.

4.1. Kesimpulan

Berikut kesimpulan dari hasil penelitian tugas akhir ini:

1. Pada pengujian performansi kontroler Floodlight tanpa algoritma Johnson, Floodlight dengan algoritma Johnson dan OpenDaylight tanpa algoritma Johnson mempelihatkan perubahan jumlah *switch* dari 6 *switch*, 8 *switch* dan 10 *switch* berbanding lurus dengan perubahan nilai QoS terutama pada nilai *delay* dan *jitter*. Kontroler Floodlight tanpa algoritma Johnson lebih unggul dibandingkan kedua kontroler lainnya dari untuk layanan data dan VoIP. Ini terlihat dari jumlah *delay* pada layanan data dan VoIP. Adapun nilai *delay* maksimal pada model kontroler ini adalah 88.38ms. Namun untuk pengiriman layanan video, kontroler OpenDaylight lebih baik dari yang lainnya. Karena hanya kontroler OpenDaylight yang bisa mengirimkan paket video. Adapun nilai *delay* maksimal untuk pengiriman layanan video adalah 1195ms.
2. Untuk peningkatan *background traffic* yang berkisar dari 0-80Mbps, secara umum kontroler OpenDaylight memiliki perubahan nilai delay dan jitter yang berbanding lurus sesuai kenaikkan background traffic-nya untuk semua layanan. Perubahan nilai *delay* dan *jitter* untuk semua layanan sering terjadi pada *background traffic* yang bernilai di 60-80Mbps. Pada kontroler floodlight tidak terjadi perubahan seperti ini.
3. Untuk hasil pengujian *resource utilization*, konsumsi memori tertinggi dimiliki oleh kontroler OpenDaylight. OpenDaylight mengkonsumsi memori sebanyak 782MB (19.1%) dari total memori *virtual machine* yaitu4094MB. Lalu diikuti oleh Floodlight dengan algoritma Johnson sebanyak 262MB (6.4%) dan terakhir Floodlight tanpa algoritma johnson sebanyak 245MB (6%).
4. Berdasarkan standarisasi ITU-T G.1010 kontroler Floodlight tanpa algoritma Johnson, Floodlight menggunakan algoritma Johnson dan OpenDaylight hanya memenuhi standarisasi *delay* untuk layanan data saja. Untuk nilai *jitter* dan *throughput*-nya tidak memenuhi standar disemua layanan. Adapun nilai *delay* maksimal untuk layanan data pada tiga kontroler tersebut yaitu 88ms, 82ms dan 636ms dimana itu belum melebihi 60s.

5.2. Saran

Saran yang diberikan untuk penelitian selanjutnya:

1. Menggunakan algoritma lain dan metode lain agar menemukan hasil QoS yang lebih baik pada kontroler floodlight.
2. Melakukan penelitian lebih lanjut pada kontroler OpenDaylight agar nilai QoS nya lebih baik dan menemukan cara bagaimana cara menambahkan algoritma lain pada kontroler OpenDaylight

DAFTAR PUSTAKA

[1] Siddharth Valluvan,T.Manoranjitham, V.Nagarajan , "A STUDY ON SDN CONTROLLERS," *Siddharth Valluvan*et al. /International Journal of Pharmacy & Technology* , p. 9, 2016.

[2] Shie-Yuan Wang, Hung-Wei Chiu, Chih-Liang Chou, "Comparisons of SDN OpenFlow Controllers over EstiNet: Ryu vs. NOX," *The International Symposium on Advances in Software Defined Networks,* p. 6, 2015.

[3] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient SDN Controllers," *IEEE Conference on Network Softwarization (NetSoft),* p. 6, 2015.

[4] Mr. Sachin Ashok Vanjari, Dr. R. B. Ingle, P.G. Student, Dean Second Shift, "Network Traffic Control Using Distributed Control Plane of Software Defined Networks," *I nternational Journal of Computer Science Trends and Technology (IJCST) – Volume 3 Issue 5,* p. 5, 2015.

[5] FAID Souad, Mohamed MOUGHIT, Noureddine IDBOUFKER , "OpenFlow Controllers Performance Evaluation," *International Journal of Emerging Research in Management &Technology ISSN: 2278-9359 (Volume-5, Issue-5),* p. 7, 2016.

[6] KaramjeetKaur, VipinGupta, "Performance Analysis Of Python Based OpenFlow Controllers," *International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences,* p. 4, 2016.

[7] SHIVA ROWSHANRAD, VAJIHE ABDI, MANIJEH KESHTGARI, "PERFORMANCE EVALUATION OF SDN CONTROLLERS: FLOODLIGHT AND OPENDAYLIGHT," *IIUM Engineering Journal, Vol. 17,*, p. 11, 2016.

[8] Pooja, Manu Sood, "SDN and Mininet: Some Basic Concepts," *Int. J. Advanced Networking and Applications* , p. 4, 2015.

[9] Selcuk Yazar, Eldem Uchar, "POX CONTROLLER PERFORMANCE FOR OPENFLOW NETWORKS," *Management and Education Vol. 1,* p. 10, 2013.

[10] Sugam Agarwal, Murali Kodialam, T. V. Lakshman, "Traffic Engineering in Software Defined Networks," *Proceedings IEEE INFOCOM,* p. 9, 2013.

[11] M. Sisov, Building a Sofware-Defined Networking System with OpenDaylight Controller, Helsinki: Helsinki Metropolia University of Applied Sciences, 2016.

[12] Seitz, N., NTIA/ITS,, "ITU-T QoS Standards for IP-Based Networks," *IEEE Communication Magazine,* p. 6, 2003.

[13] Bruce Hartpence, Rossi Rosario, "Software Defined Networking for Systems and Network Administration Programs," *The USENIX Journal of Education in System Administration,* p. 15, 2016.

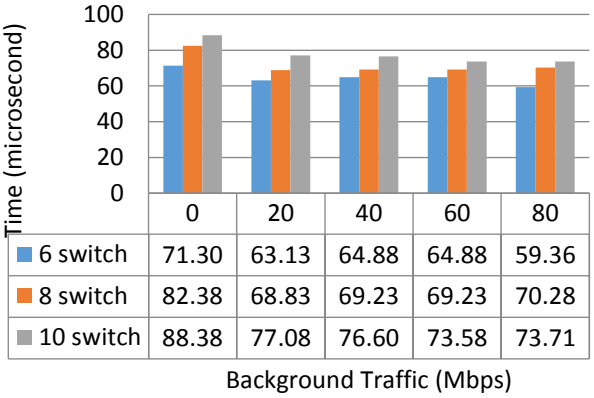
[14] Wawan Apriadi, R. Rumani, Sofia Naning Hertiana, "SIMULATION AND PERFORMANCE ANALYSIS OF SOFTWARE DEFINE NETWORK (SDN) USING BELLMAN FORD ALGORITHM AS ROUTING," p. 18, 2016.

LAMPIRAN

HASIL PENGUJIAN QoS

1. *Delay*

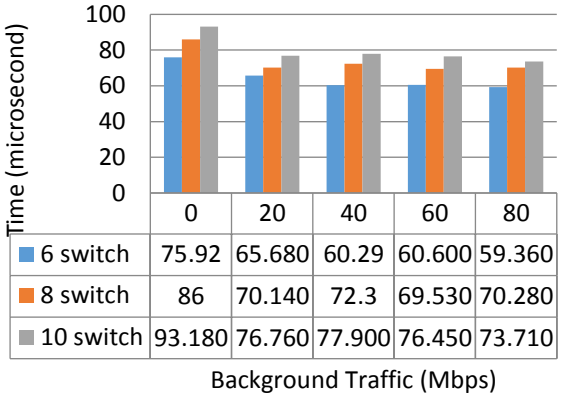
Delay Data Floodlight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	71.30	63.13	64.88	64.88	59.36
8 switch	82.38	68.83	69.23	69.23	70.28
10 switch	88.38	77.08	76.60	73.58	73.71

Background Traffic (Mbps)

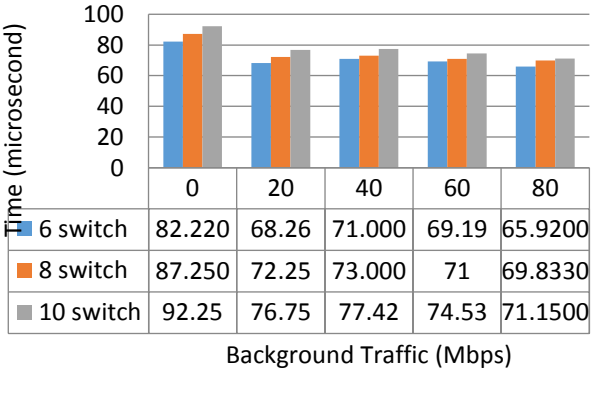
Delay VoIP Floodlight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	75.92	65.680	60.29	60.600	59.360
8 switch	86	70.140	72.3	69.530	70.280
10 switch	93.180	76.760	77.900	76.450	73.710

Background Traffic (Mbps)

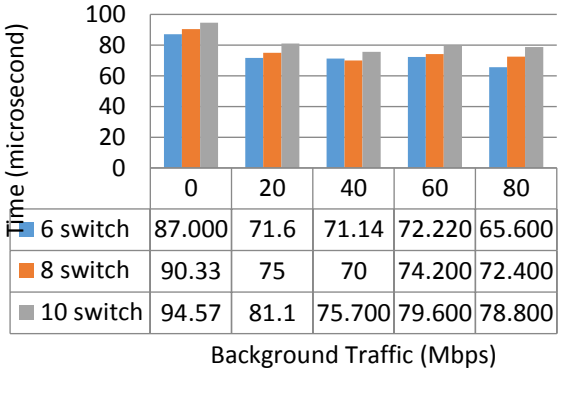
Delay Data Floodlight Dengan Algoritma Johnson



	0	20	40	60	80
6 switch	82.220	68.26	71.000	69.19	65.9200
8 switch	87.250	72.25	73.000	71	69.8330
10 switch	92.25	76.75	77.42	74.53	71.1500

Background Traffic (Mbps)

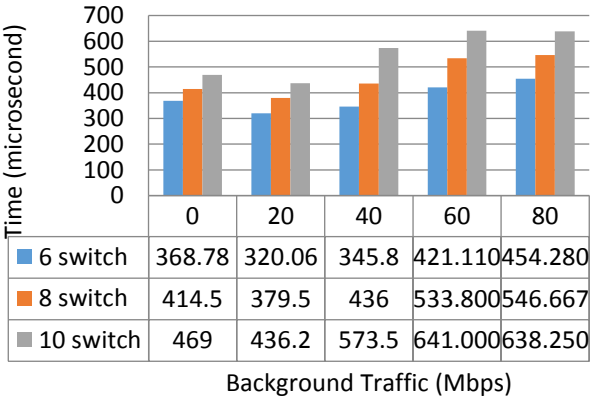
Delay VoIP Floodlight Dengan Algoritma Johnson



	0	20	40	60	80
6 switch	87.000	71.6	71.14	72.220	65.600
8 switch	90.33	75	70	74.200	72.400
10 switch	94.57	81.1	75.700	79.600	78.800

Background Traffic (Mbps)

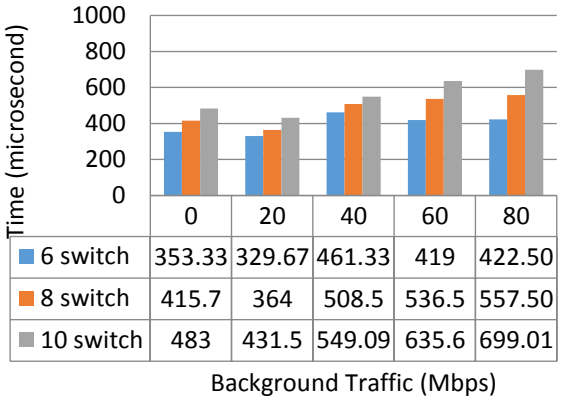
Delay Data OpenDaylight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	368.78	320.06	345.8	421.110	454.280
8 switch	414.5	379.5	436	533.800	546.667
10 switch	469	436.2	573.5	641.000	638.250

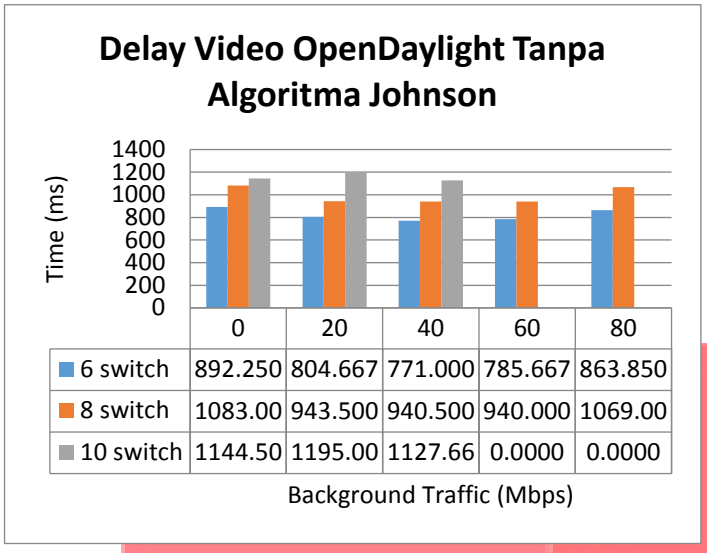
Background Traffic (Mbps)

Delay VoIP OpenDaylight Tanpa Algoritma Johnson

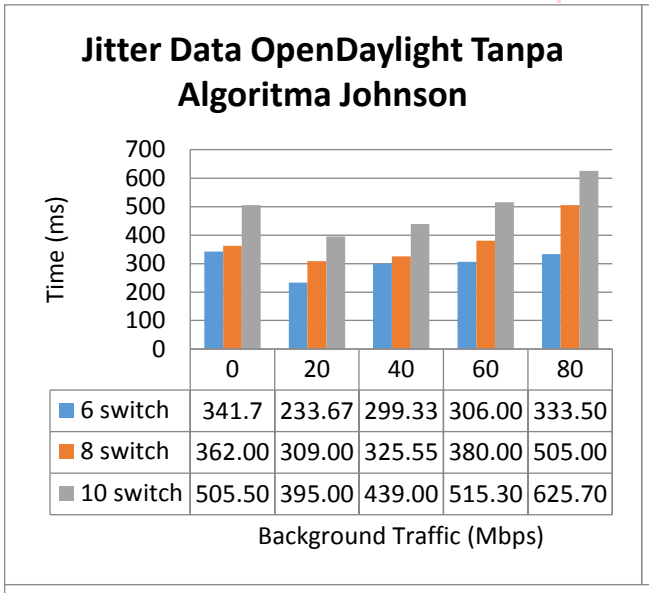


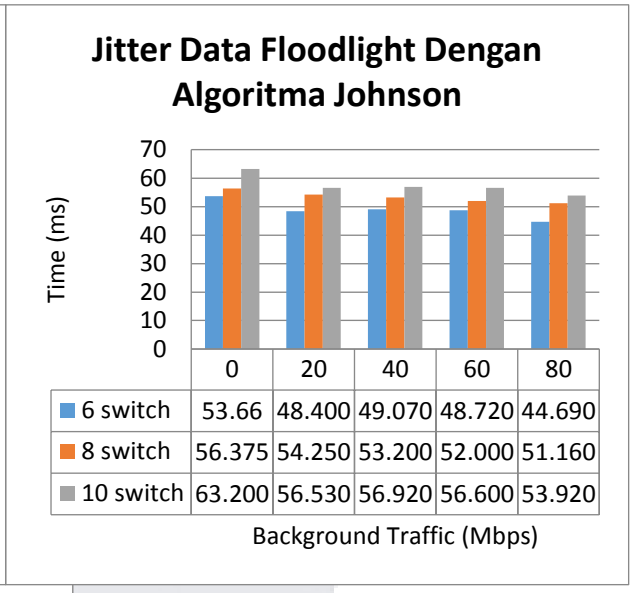
	0	20	40	60	80
6 switch	353.33	329.67	461.33	419	422.50
8 switch	415.7	364	508.5	536.5	557.50
10 switch	483	431.5	549.09	635.6	699.01

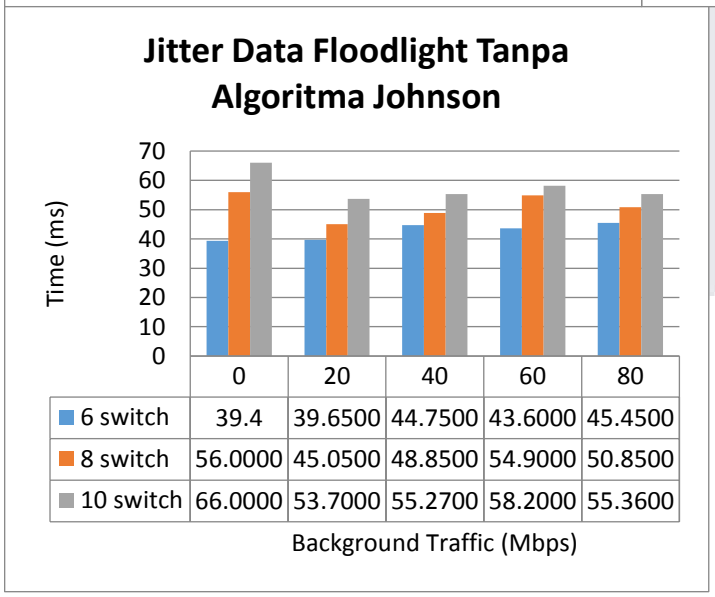
Background Traffic (Mbps)

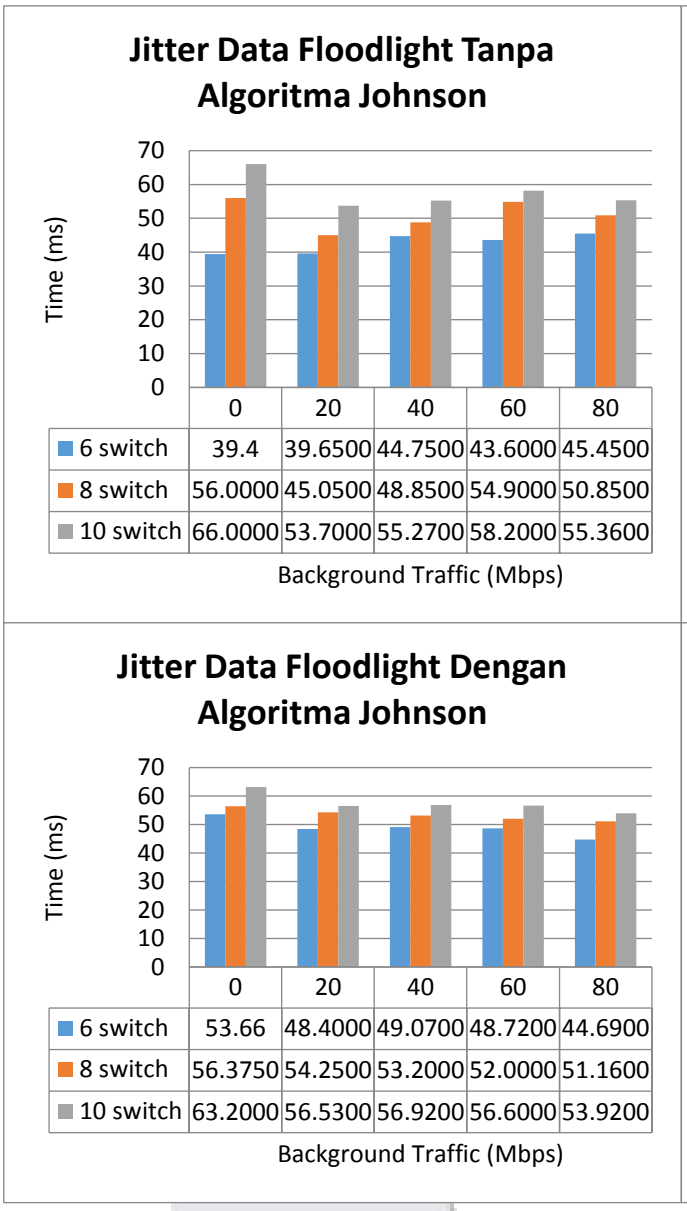

Delay Video OpenDaylight Tanpa Algoritma Johnson
Time (ms)
1400
1200
1000
800
600
400
200
0
0 20 40 60 80
6 switch 892.250 804.667 771.000 785.667 863.850
8 switch 1083.00 943.500 940.500 940.000 1069.00
10 switch 1144.50 1195.00 1127.66 0.0000 0.0000
Background Traffic (Mbps)

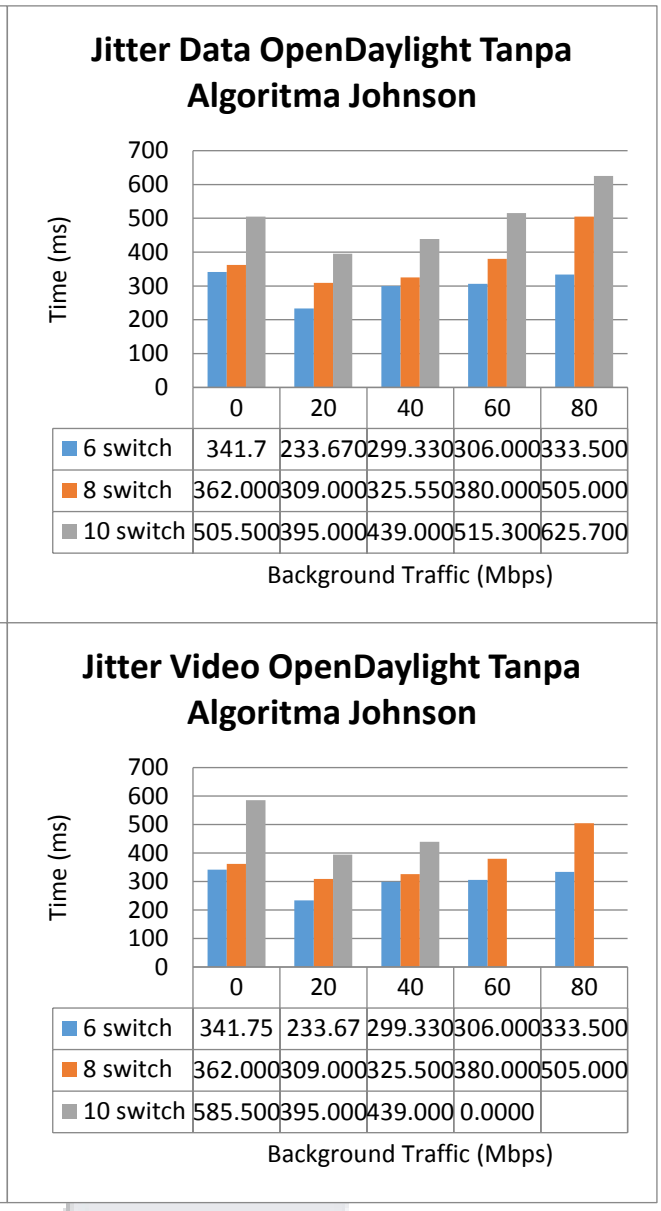
2. Jitter


Jitter Data OpenDaylight Tanpa Algoritma Johnson
Time (ms)
700
600
500
400
300
200
100
0
0 20 40 60 80
6 switch 341.7 233.67 299.33 306.00 333.50
8 switch 362.00 309.00 325.55 380.00 505.00
10 switch 505.50 395.00 439.00 515.30 625.70
Background Traffic (Mbps)

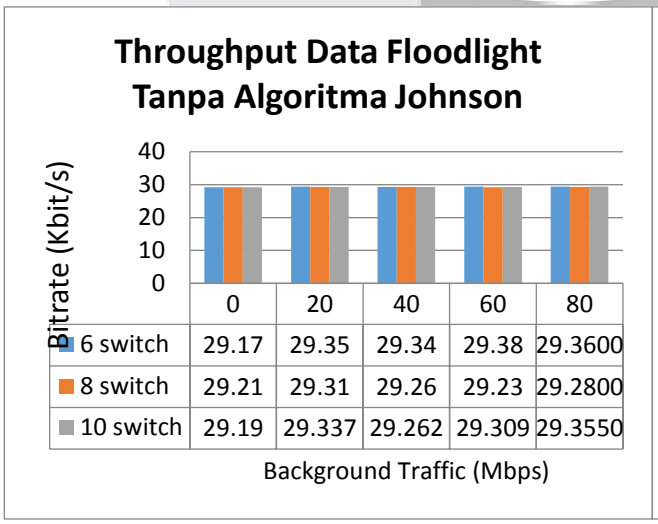

Jitter Data Floodlight Dengan Algoritma Johnson
Time (ms)
70
60
50
40
30
20
10
0
0 20 40 60 80
6 switch 53.66 48.400 49.070 48.720 44.690
8 switch 56.375 54.250 53.200 52.000 51.160
10 switch 63.200 56.530 56.920 56.600 53.920
Background Traffic (Mbps)

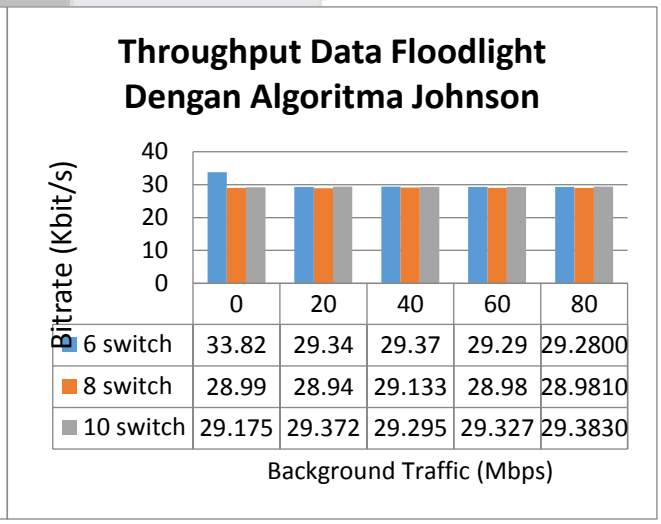

Jitter Data Floodlight Tanpa Algoritma Johnson
Time (ms)
70
60
50
40
30
20
10
0
0 20 40 60 80
6 switch 39.4 39.6500 44.7500 43.6000 45.4500
8 switch 56.0000 45.0500 48.8500 54.9000 50.8500
10 switch 66.0000 53.7000 55.2700 58.2000 55.3600
Background Traffic (Mbps)



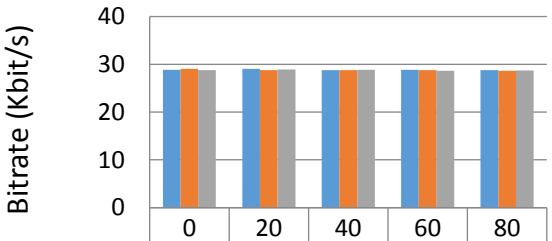


3. Throughput





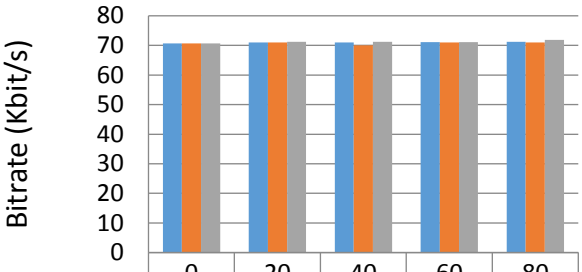
Throughput Data OpenDaylight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	28.84	29.03	28.79	28.88	28.800
8 switch	29.05	28.754	28.79	28.8	28.647
10 switch	28.76	28.95	28.85	28.65	28.699

Background Traffic (Mbps)

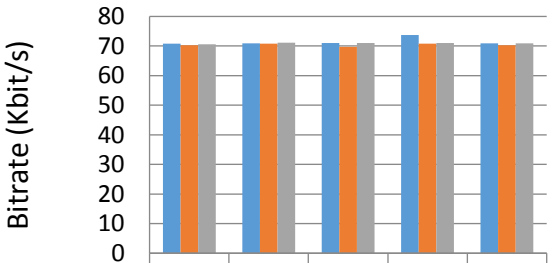
Throughput VoIP Floodlight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	70.63	70.97	70.98	71.11	71.1600
8 switch	70.699	71.007	70.012	71.019	70.9800
10 switch	70.71	71.19	71.244	71.085	71.8820

Background Traffic (Mbps)

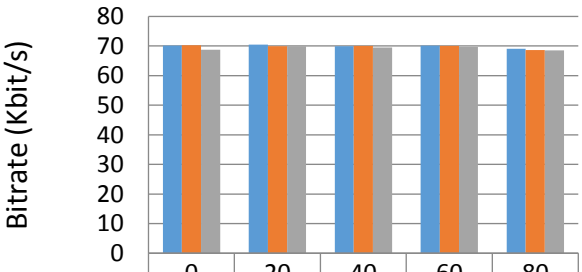
Throughput VoIP Floodlight Dengan Algoritma Johnson



	0	20	40	60	80
6 switch	70.78	70.96	71.02	73.73	70.920
8 switch	70.27	70.84	69.74	70.83	70.240
10 switch	70.61	71.169	71.027	71.07	70.960

Background Traffic (Mbps)

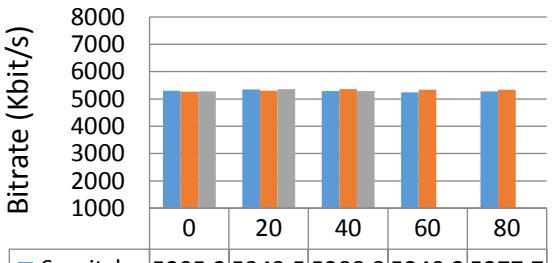
Throughput VoIP OpenDaylight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	70.16	70.45	69.99	70.172	69.0500
8 switch	70.3	69.917	70.049	70.097	68.6400
10 switch	68.8	69.95	69.54	69.727	68.4900

Background Traffic (Mbps)

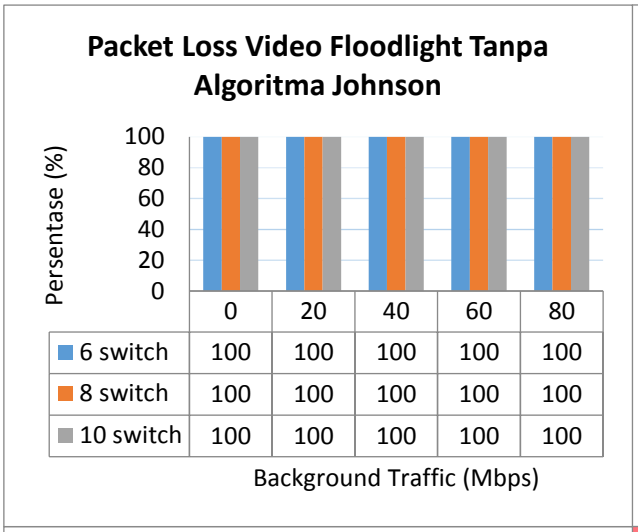
Throughput Video OpenDaylight tanpa Algoritma Johnson

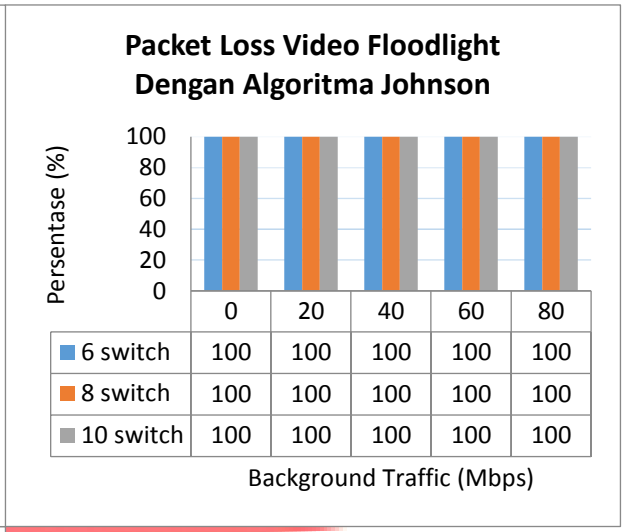


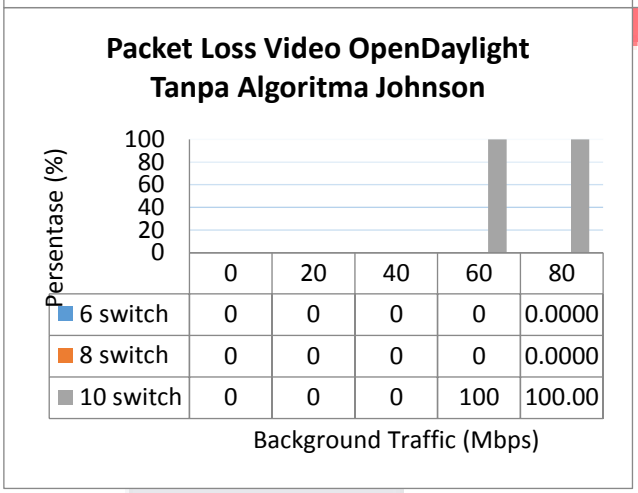
	0	20	40	60	80
6 switch	5295.2	5343.5	5288.8	5240.2	5277.7
8 switch	5268.6	5295.2	5354.9	5340.5	5330.1
10 switch	5275.9	5362	5290.8		

Background Traffic (Mbps)

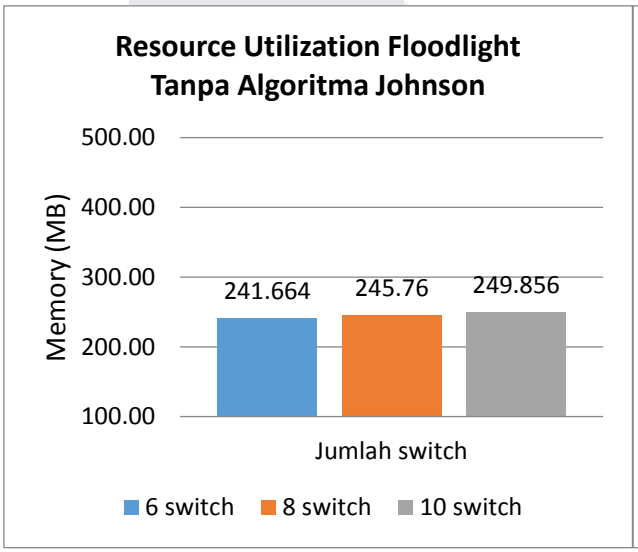
4. Pakcet Loss

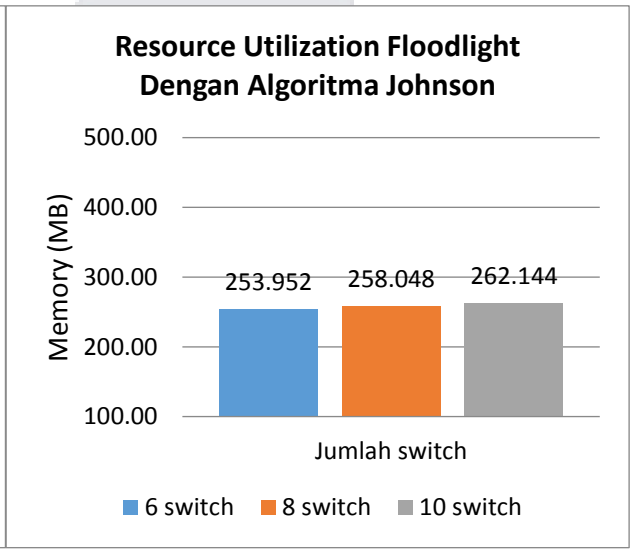


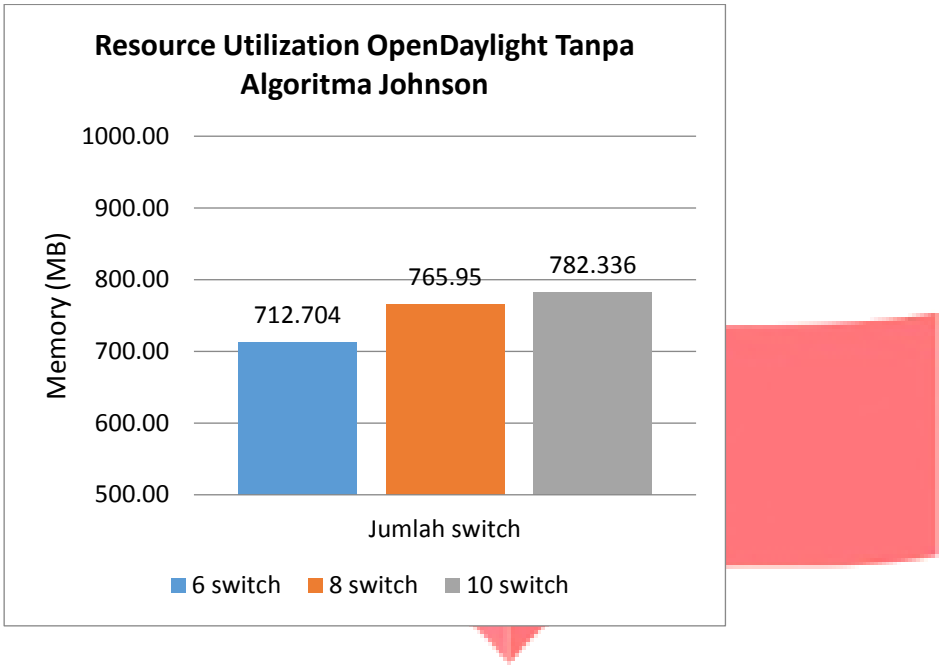




HASIL PENGUJIAN RESOURCE UTILIZATION






Resource Utilization OpenDaylight Tanpa Algoritma Johnson
1000.00
900.00
800.00
700.00
600.00
500.00
Memory (MB)
712.704
765.95
782.336
Jumlah switch
6 switch
8 switch
10 switch