ANALISIS EFEK PENGGUNAAN KONTROLER RYU DAN POX PADA PERFORMANSI JARINGAN SDN

ANALYISIS OF EFFECT OF CONTROLLER RYU AND POX ON SDN NETWORK PERFORMANCE

Romi Afan[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]romiafan@gmail.com, [2]avirgono@telkomuniversity.ac.id, [3]rumani@telkomuniversity.ac.id

Abstrak

Perkembangan jaringan komputer terus meningkat untuk memenuhi kebutuhan dan menciptakan jaringan yang handal. Software Defined Network (SDN) merupakan salah satunya. SDN merupakan pemisahan sistem pengontrol arus data dari perangkat kerasnya. Konsep tersebut memberikan fleksibilitas pada jaringan sehingga dapat mengolah jaringan menggunakan kontroler tanpa harus menyentuh perangkat keras. Pada SDN kendali jaringan di pusatkan dalam control plane yang dapat secara pintar mengatur jaringan berdasarkan kondisi keseluruhan jaringan. Untuk melakukan itu SDN menggunakan interface yang disebut kontroler. Kontroler ini umumnya menggunakan kontroler OpenFlow (NOX, POX, Beacon, Floodlight, MuL, Maestro, Ryu). Dalam kontroler OpenFlow sendiri memiliki basis Bahasa pemograman yang berbeda seperti python dan java. Masing-masing kontroler memiliki kelebihan, kekurangan dan penggunaan yang berbeda. Pada penelitian ini membandingkan nilai QoS (Quality of Service) jaringan yang dibangun menggunakan kontroler Ryu dan POX. Kedua kontroler ini diuji pada topologi full-mesh yang jumlah switch 6, 8 dan 10 dan masing-masing switch mempunyai 2 host yang terhubung. Nilai QoS yang didapat untuk kedua kontroler masih didalam standar ITU-T G.1010 1010 yaitu delay kecil dari 15 s untuk data, kecil dari 150 ms untuk VoIP dan kecil dari 10 s untuk video.

Kata kunci: ***Software Defined Network, Openflow, Mininet, Ryu, POX***

Abstract

The development of computer networks continues to increase to meet the needs and create a reliable network. Software Defined Network (SDN) is one of them. SDN is the separation of data flow controller system from hardware. The concept provides flexibility on the network so that it can process the network using a controller without having to touch the hardware. In the SDN network control is centered in a control plane that can intelligently manage the network based on the overall condition of the network. To do that the SDN uses an interface called a controller. These controllers generally use OpenFlow controllers (NOX, POX, Beacon, Floodlight, MuL, Maestro, Ryu). In OpenFlow controllers themselves have different programming Language bases like python and java. Each controller has advantages, disadvantages and different uses. In this study comparing network QoS values built using Ryu and POX controllers. Both of these controllers are tested in full-mesh topologies with the number of switches 6, 8 and 10 and each switch has 2 connected hosts. The QoS value obtained for both controllers is still in the ITU-T G.1010 standard 1010, delay less than 15 s in data, less than 150 ms in VoIP and less than 10 s in video.

Keywords: Software-Defined Network, Controller, Performance, Ryu, POX

1. Pendahuluan

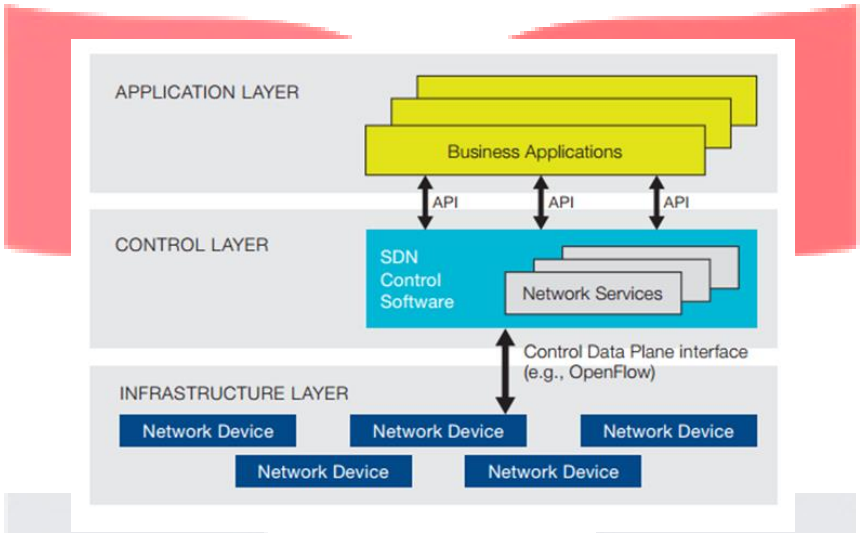
Software Defined Network (SDN) adalah sebuah konsep pendekatan jaringan komputer dimana system pengontrol dari arus data dipisahkan dari perangkat kerasnya. Pemisahan tersebut memberikan SDN flexibility, programmability, new capabilities dalam perangkat jaringan yang dapat menuntaskan masalah pada jaringan konvensional[9].

Pada SDN, perangkat keras jaringan tidak lagi mengatur jalur data dan tugas tersebut diberikan pada control plane. Control plane mengatur jalur data dengan mengambil keputusan menurut keadaan jaringan secara keseluruhan. Oleh Karena itu kontroler pada SDN merupakan elemen penting dalam performansi jaringan. Kontroler pada SDN umumnya menggunakan protocol OpenFlow. SDN memiliki banyak kontroler seperti, NOX,

POX, Beacon, Floodlight, MuL, Maestro, Ryu dan lain-lain. Setiap kontroler memiliki perbedaan dalam kelebihan dan kekurangan sehingga mempengaruhi performansi jaringan.

Perbedaan pada kontroler ini dapat menentukan kegunaan masing-masing pada kontroler tersebut dan pengaruhnya terhadap performansi jaringan. Oleh karena itu pada tugas akhir ini akan melakukan analisis perbandingan terhadap dua kontroler yakni, Ryu dan POX menggunakan virtualisasi jaringan pada Mininet. Ryu dan POX yang merupakan kontroler berbasis python[3]. Analisis ini berdasarkan performansi untuk menentukan kontroler yang lebih baik dan akan dibandingkan kembali ke kontroler lainnya.

2. Dasar Teori



Gambar 2.1 Arsitektur Jaringan SDN

- Layer Aplikasi Berada pada lapis teratas, berkomunikasi dengan system via NorthBound API
- Layer Kontrol menyediakan fungsi kontrol terhadap perilaku *forwading* di dalam jaringan melalui sebuah antarmuka terbuka
- Layer infrastruktur merupakan bagian dari perangkat jaringan yang menyediakan *packet switching* dan *forwading*. Agar dapat menghubangkan *control plane* (layer control) dan *data plane* (layer infrastruktur) dapat menggunakan OpenFlow.

2.1. Protokol OpenFlow[2]

OpenFlow adalah suatu protokol terbuka yang pertama kali mendefinisikan antara perangkat *control plane* yaitu *controller*, dan perangkat *data plane* yaitu *switch* pada arsitektur SDN[6]. OpenFlow memungkinkan mengakses langsung dan manipulasi perangkat *forwarding plane* seperti *switch* dan *router*, baik fisik dan virtual (*hypervisor-based*)[10]. OpenFlow diimplementasi dari kedua sisi antarmuka baik dari sisi infrastruktur jaringan dan kontrol *software* SDN.

Sebuah OpenFlow *switch* terdiri dari *flow table* atau *grup table*. Setiap *flow table* dalam *switch* berisi satu *set flow entri*, yang terdiri dari *header field*, *counter*, dan *set of instructions* atau *actions*.

2.2. Kontoller Ryu[7]

RYU [13] adalah sebuah *framework* untuk aplikasi SDN. Pada RYU disediakan komponen perangkat API yang memudahkan untuk membuat sebuah manajemen jaringan atau aplikasi kontrol. Ryu mendukung berbagai protokol untuk mengelola perangkat jaringan, seperti OpenFlow, netcof, OF-config, dan lainya

RYU mendukung untuk OpenFlow 1.0, 1.2, 1.3, dan 1.4 . RYU sepenuhnya dikembangkan menggunakan bahasa pemerograman pyton semua *code* pada ryu bersifat *open source* di bawah lisensi Apache 2.0 .

2.3. Kontroler POX[6]

POX [12] adalah kontroler open source untuk mengembangkan aplikasi SDN. POX controller menyediakan cara yang efisien untuk mengimplementasikan protokol OpenFlow yang merupakan protokol komunikasi antara control plane dan data plane. POX dapat menjalankan aplikasi yang berbeda seperti hub, switch, load balancer, dan firewall. Alat capture paket Tcpdump bias digunakan untuk menangkap dan melihat paket yang mengalir di antaranya POX controller dan perangkat OpenFlow.

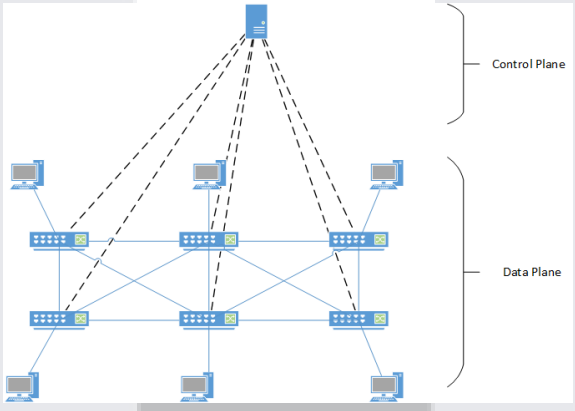
2.4. Mininet

Mininet adalah emulator jaringan yang menciptakan jaringan *virtual host*, switch, *controller*, dan *link*. Mininet merupakan emulator jaringan yang bersifat *open source* yang mendukung protocol OpenFlow untuk arsitektur SDN[7]. Tampilan pada mininet berupa *Command Language Interpreter* (CLI), hal tersebut memudahkan dalam melakukan skenario jaringan. Mininet merupakan emulator yang sudah teruji untuk mengimplementasikan konsep SDN[8]. Mininet dapat dijalankan dalam sebuah laptop yang meggunakan sistem operasi berbasis Linux[13]. Mininet memiliki kekurangan yaitu mininet tidak dapat berjalan pada sistem operasi non-linux yang tidak mendukung switch OpenFlow.

3. Pembahasan

3.1. Model Sistem Simulasi

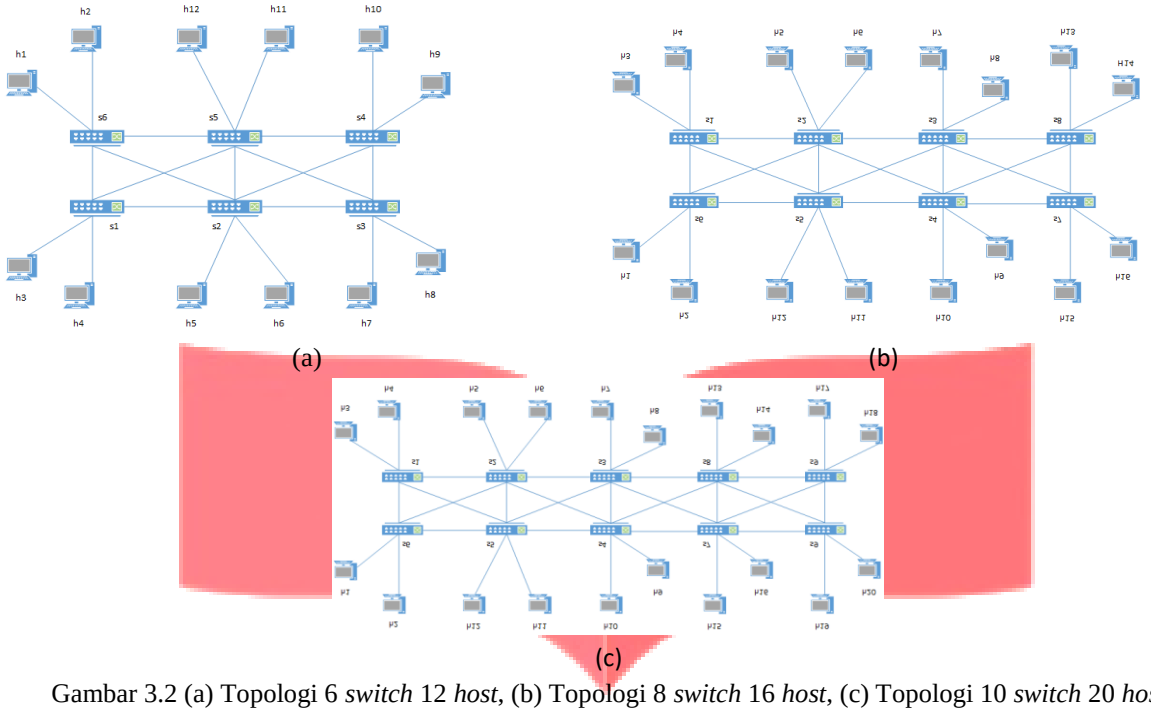
Sistem simulasi ini mensimulasikan suatu jaringan berbasiskan SDN yang terdiri dari controller dan data plane. Untuk controller yang digunakan adalah controller RYU dan POX yang didalamnya sudah terdapat routing menggunakan algoritma Johnson. Controller RYU dan POX pada simulasi ini menjalankan fungsi sebagai control plane yang bertugas untuk mengontrol aliran data, melakukan routing dan melakukan keseluruhan aturan dari SDN. Data plane dijalankan pada suatu emulator mininet. Pada mininet dirancang sebuah topology yang terdiri switch openflow dan host.



Gambar 3.1 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:



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 topology dengan jumlah switch 6, 8 dan 10. Hal ini dilakukan untuk mengetahui pengaruh jumlah switch terhadap kinerja routing. Pengujian QoS melibatkan parameter delay, jitter, troughput, dan packet loss. 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
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
3. VoIP menggunakan G.711 codec tanpa voice activation detection (VAD) sebanyak 100 pps yang berukuran paket 80 Bytes

Pada pengujian ini dilakuan variasi pengujian dengan menambahkan background trafik dengan bantun iperf. Terdapat 5 jenis backround trafik yang digunakan yaitu 0Mbps, 20Mbps, 40Mbps, 60Mbps, dan 80Mbps.

3.3. Hasil Pengujian

3.3.1. Pengujian *QoS*

Parameter yang digukan untuk pengukuran QoS pada subab ini adalah *delay, troughput, jitter, dan packet loss*. Pengujian QoS dilakukan dengan mengalirkan jenis layanan yaitu data, video, dan VoIP. Terdapat empat skenario yang digunakan pada pengujian ini yaitu pengujian tanpa *background traffic*, dan pengujian dengan *background traffic* 0Mbps, 20Mbps, 40Mbps, 60Mbps dan 80Mbps.

3.3.1.1 *Delay*

Secara umum hasil yang didapatkan dari pengujian *delay* pada kontroler POX dan Ryu dengan layanan data, VoIP, dan video tidak jauh berbeda. Bila dilakukan perbandingan pada standarisasi yang telah ditentukan ITU-T G.1010 dengan trafik berupa data, VoIP, dan video hasil yang didapatkan masih memenuhi standar untuk ketiga jenis trafik tersebut. Dari hasil pengujian menunjukan nilai delay pada kontoler POX dan Ryu tidak jauh berbeda, namun pada pengujian delay video terlihat POX mempunyai delay yang lebih besar dengan margin yang signifikan. Pada hasil 6 switch POX menunjukan nilai 1,49 detik berbanding dengan 0,07 detik kontroler Ryu. Hal ini menunjukan bahwa POX kesulitan dalam mengirim layanan video. Hasil yang didapat juga menunjukan bahwa ada peningkatan delay terhadap besar background traffic.

3.3.1.2 Jitter

Hasil yang didapatkan berdasrkan pengujian Jitter pada jaringan SDN dengan kontroler POX dan Ryu bila dilakukan perbandingan pada standarisasi yang telah ditentukan ITU-T G.1010 dengan layanan berupa data, VoIP, dan video yang dialirkan maka hasil yang didapatkan masih memenuhi standar untuk ketiga jenis layanan tersebut yaitu dengan nilai jitter masih dibawah 1ms. Pada pengujian jitter ini kembali terlihat kontroler POX kesulitan mengolah layanan video karena ukurannya, hal ini terlihat dari nilai jitter yang didapat.

3.3.1.3 Throughput

Hasil nilai pengujian throughtput layanan video menunjukan penurunan bitrate kedua kontroler Ryu dan POX ini. Hal ini disebabkan ukuran file video streaming lebih besar dari pada pengujian data dan VoIP sebelumnya.

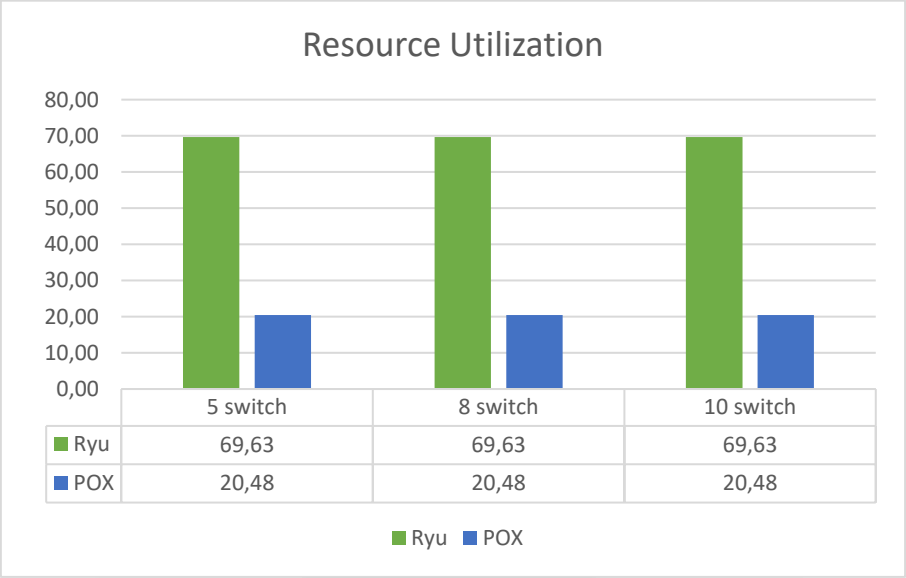
3.3.1.4 Packet Loss

Pada pengujian *packet los*s kontroler Ryu mempunyai packet loss % pada jenis layanan data, VoIP dan video . Sedangkan pada POX packet loss 0% hanya terjadi pada data dan voip. Hasil pengujian video mempunyai packet loss, menunjukan saat besar *background traffic* 0Mbps, 20Mbps, 40Mbps, 60Mbps, dan 80Mbps nilai *packet loss* meningkat seiring dengan penambahan besar background traffic, hal ini disebabkan besar *background traffic* sudah menyamai dengan besar *bandwith* pada *link*. Pada hasil ini juga dapat ditemukan kesulitan kontroler POX dalam mengolah layanan video.

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.



Gambar 3.3 Hasil pengujian *resource utilization*

Dari graf diatas terlihat hasil pengukuran konsumsi memori kontroler POX dan Ryu. Kedua kontroler tersebut tidak menunjukan perbedaan konsumsi memori pada penambahan jumlah switch yakni, 5 *switch*, 8 *switch* dan 10 *switch*. Secara umum konsumsi memory kontroler Ryu lebih besar dibandingan POX. Hal ini terlihat dari besarnya konsumsi memory sekitar 3 kali konsumsi POX.

4.1. Kesimpulan

Perbandingan performansi jaringan kinerja kontroler POX dan Ryu dapat terlihat dalam pengujian resource utilization dan pengukuran QoS jaringan. Hasil analisis efek kontroler POX dan Ryu terhadap performansi jaringan adalah sebagai berikut:

1. Hasil dari pengujian resource utilization menunjukan kontoler Ryu memakai 1,7% memory dari total 4GB memory virtual machine yang digunakan, atau sekitar 69,63 MB. Kontroler POX memakai 0,5% memory atau sekitar 20,48 MB.
2. Dari pengujian resource utilization menunjukan penambahan jumlah switch terhadap topologi yang telah dirancang tidak mempengaruhi pemakaian jumlah memory kontroler POX dan Ryu.
3. Hasil pengujian QoS kontroler POX dan Ryu pada topologi 5 switch, 8 switch dan 10 switch yang telah dirancang menunjukan nilai delay, jitter dan throughput yang didapatkan masih dalam rentang nilai standar ITU-T G.1010 yaitu delay kecil dari 15 s untuk data, kecil dari 150 ms untuk VoIP dan kecil dari 10 s untuk video.
4. Berdasarkan pengujian QoS, performa kontroler Ryu lebih unggul dibandingkan kontroler POX. Terutama terhadap layanan video streaming, hal ini dibuktikan dengan kontroler POX mendapatkan packet loss terkecil 7,7% dan terbesar 52% nilai ini tidak memenuhi standar ITU-T G.1010 untuk packet loss kecil dari 1%.

4.2. Saran

Saran yang diberikan untuk penelitian selanjutnya:

1. Melakukan analisa performansi pada kontroler selain Ryu dan POX seperti pyretic, NOX, beacon, dll
2. Melakukan analisa pengaruh kontroler terhadap jaringan SDN *real* tidak lagi menggunakan emulator.

DAFTAR PUSTAKA

[1] Pooja, Sood, M., "SDN and Mininet: Some Basic Concepts," *Int. J. Advanced Networking and Applications*, vol. Volume: 07, no. Issue 02, pp. 2690-2693, 2015.

[2] ROWSHANRAD, S., ABDI, V., KESHTGARI, M., "PERFORMANCE EVALUATION OF SDN CONTROLLERS: FLOODLIGHT AND OPENDAYLIGHT," *IIUM Engineering Journal*, vol. Volume 17, no. No. 2, pp. 57-57, 2016.

[3] Kaur, K., Kaur, S., Gupta, V., "Performance Analysis Of Python Based OpenFlow Controllers," *3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS)*, 2016.

[4] Ilhamsyah, M., R. Rumani M, Sofia Naning Hertiana, "PERANCANGAN SDN (SOFTWARE DEFINED NETWORK) DENGAN ALGORITMA JOHNSON MENGGUNAKAN EMULATOR MININET," Telkom University, Bandung, 2017.

[5] Anggara, S.M., Suhardi, "Pengujian Performa Kontroler Software-defined," INSTITUT TEKNOLOGI BANDUNG, Bandung, 2015.

[6] Borcoci, E., Badea, R., Obreja, S.G., Vochin M., "On Multi-controller Placement Optimization in Software Defined Networking based WANs," in *ICN 2015 : The Fourteenth International Conference on Networks*, Bucharest, 2015.

[7] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R., "On Controller Performance in Software-Defined Networks," *Big Switch Networks*, 2012.

[8] Sukhveer Kaur, Japinder Singh, Navtej Singh Ghumman, "Network Programmability Using POX Controller," Ferozepur, India , 2014.

[9] Bindhu, M., G., P.R., "Load Balancing and Congestion Control in Software Defined Networking using the Extended Johnson Algorithm for Data Centre," *International Journal of Applied Engineering*, vol. 10, 2015.

[10] Seitz, N., & NTIA/ITS, "ITU-T QoS Standards for IP-Based," *Networks. IEEE Communications Magazine.*, 2013.

[11] Botta, A., Donato, W. d., Dainotti, A., Avallone, S., & Pescape, A, "D-ITG 2.8.1 Manual," University of Napoli Federico II, Departement of Electrical Engineering and Information Technologies., Napoli, 2013.

[12] Ryu, "COMPONENT-BASED SOFTWARE DEFINED NETWORKING FRAMEWORK Build SDN Agilely," 2017. [Online]. Available: http://osrg.github.com/ryu/.

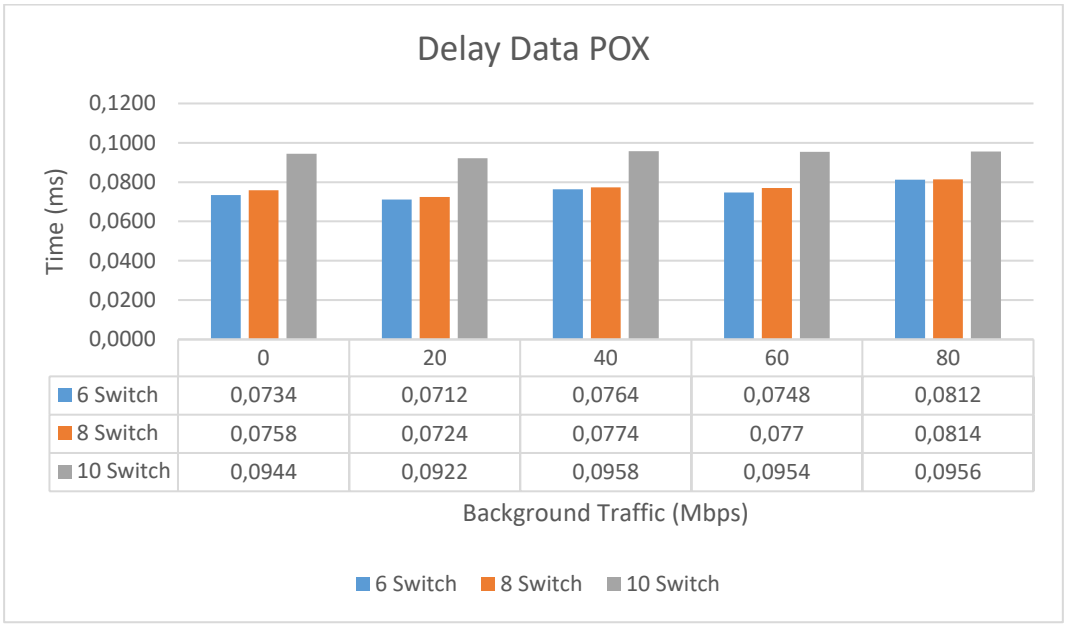
[13] Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., Smeliansky, R., "Advanced Study of SDN/OpenFlow controllers," CEE-SECR, Moscow, 2013.

[14] Valluvan, S., Manoranjitham, T., Nagarajan, V., "A STUDY ON SDN CONTROLLERS," *International Journal of Pharmacy & Technology*, vol. Vol. 8, no. No. 4, pp. 5234-5242, 2016.

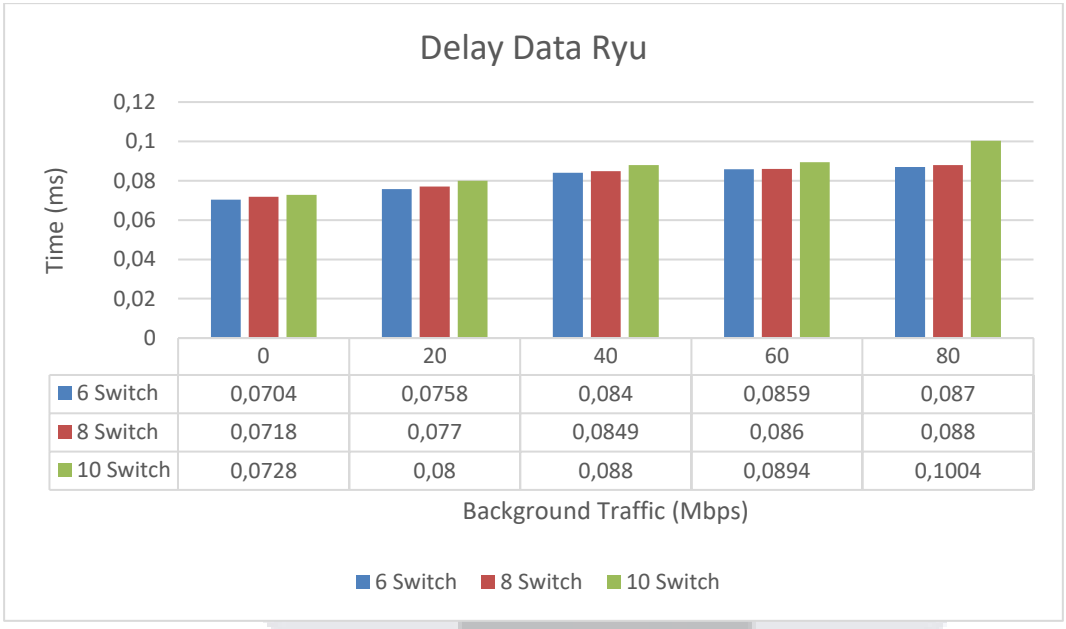
LAMPIRAN

Hasil pengujian QoS

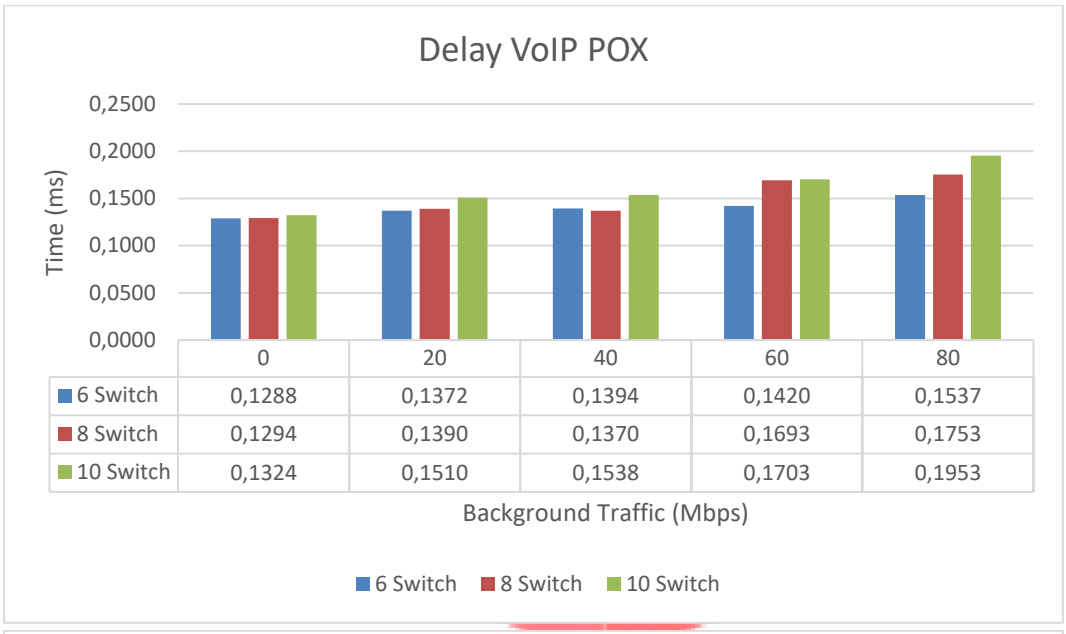
1. Delay



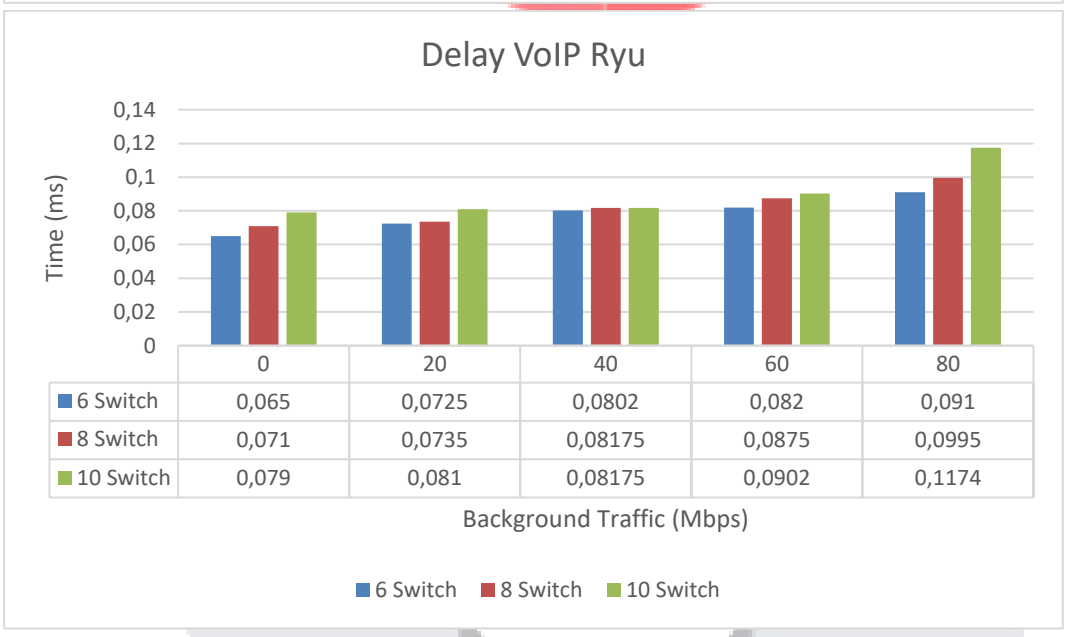
	0	20	40	60	80
6 Switch	0,0734	0,0712	0,0764	0,0748	0,0812
8 Switch	0,0758	0,0724	0,0774	0,077	0,0814
10 Switch	0,0944	0,0922	0,0958	0,0954	0,0956



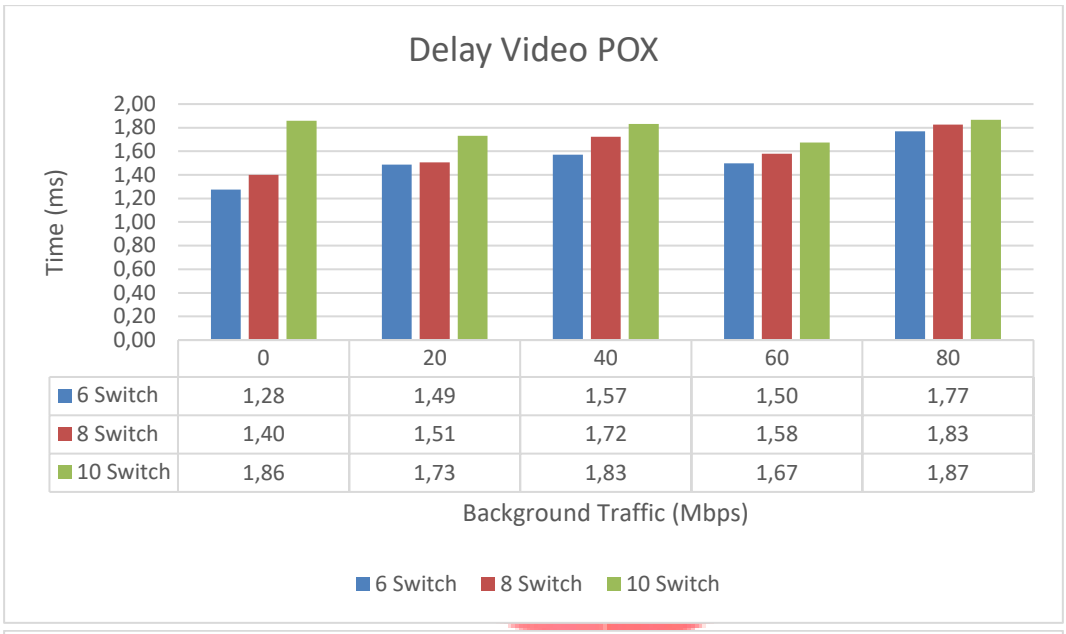
	0	20	40	60	80
6 Switch	0,0704	0,0758	0,084	0,0859	0,087
8 Switch	0,0718	0,077	0,0849	0,086	0,088
10 Switch	0,0728	0,08	0,088	0,0894	0,1004

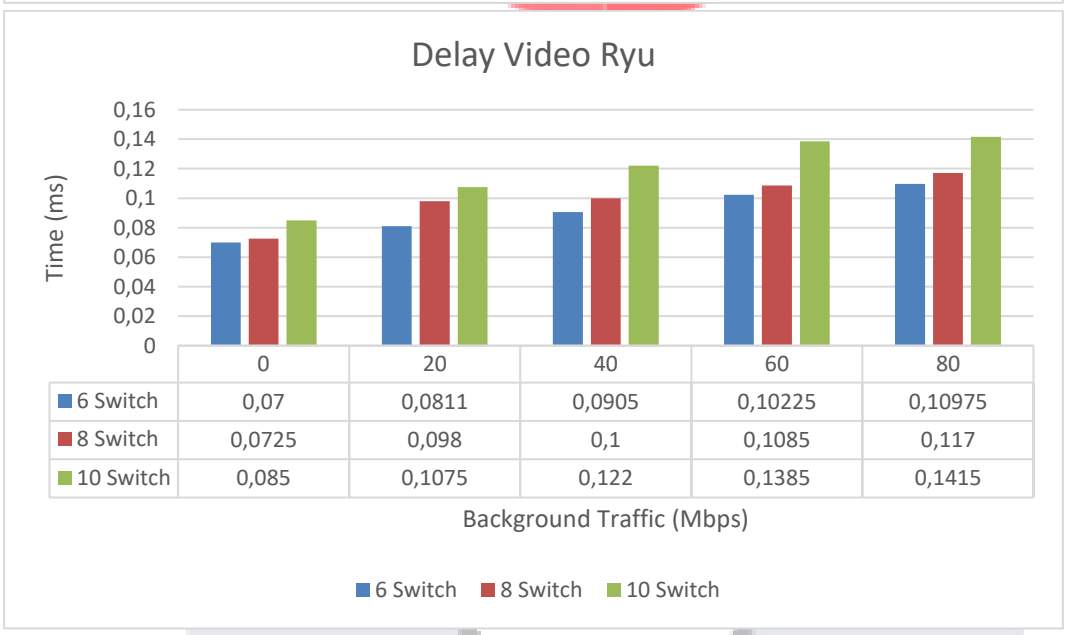


	0	20	40	60	80
6 Switch	0,1288	0,1372	0,1394	0,1420	0,1537
8 Switch	0,1294	0,1390	0,1370	0,1693	0,1753
10 Switch	0,1324	0,1510	0,1538	0,1703	0,1953

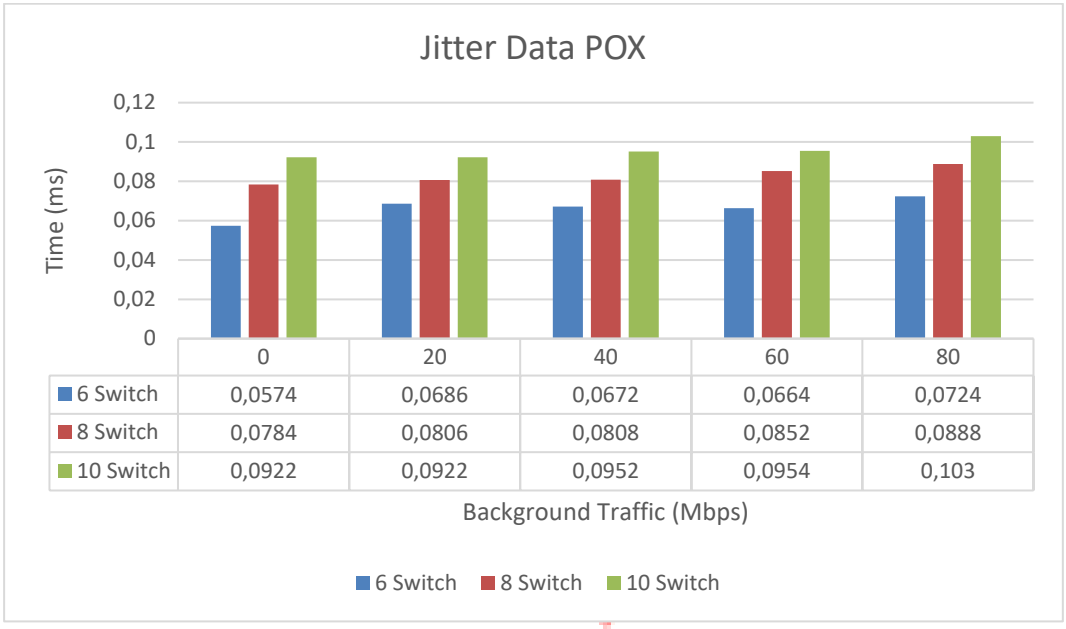


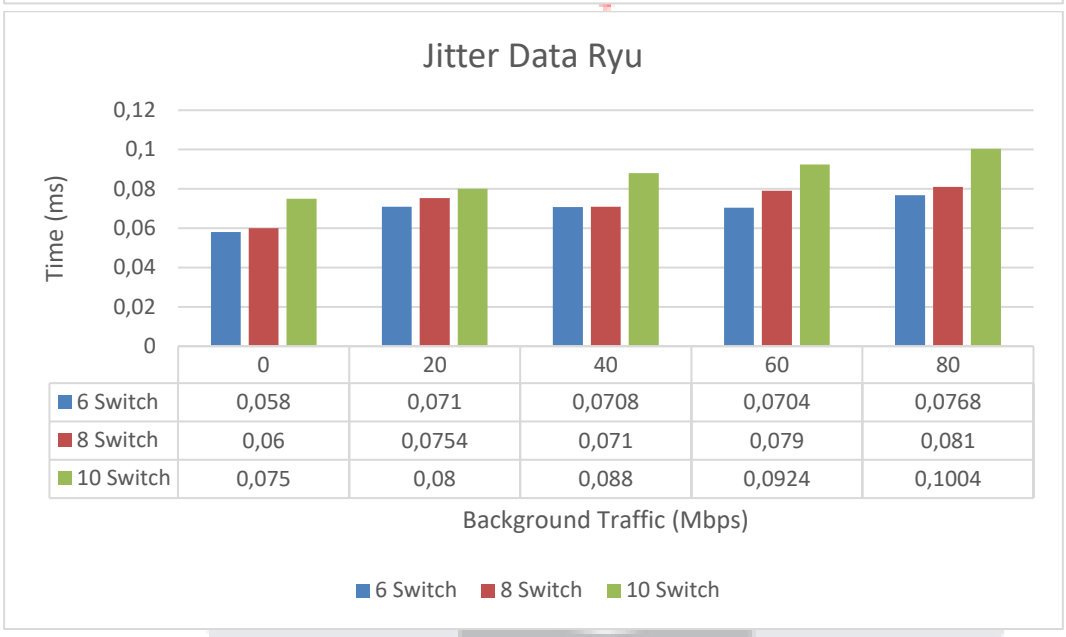
	0	20	40	60	80
6 Switch	0,065	0,0725	0,0802	0,082	0,091
8 Switch	0,071	0,0735	0,08175	0,0875	0,0995
10 Switch	0,079	0,081	0,08175	0,0902	0,1174

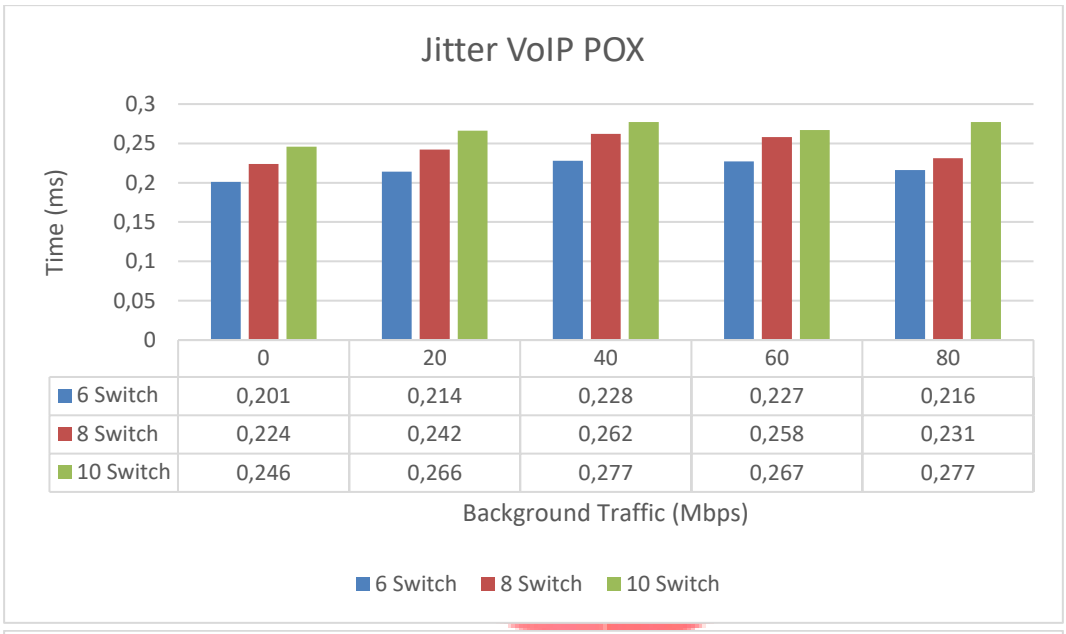

Delay Video POX
Time (ms)
2,00
1,80
1,60
1,40
1,20
1,00
0,80
0,60
0,40
0,20
0,00
0
20
40
60
80
6 Switch 1,28 1,49 1,57 1,50 1,77
8 Switch 1,40 1,51 1,72 1,58 1,83
10 Switch 1,86 1,73 1,83 1,67 1,87
Background Traffic (Mbps)
6 Switch
8 Switch
10 Switch

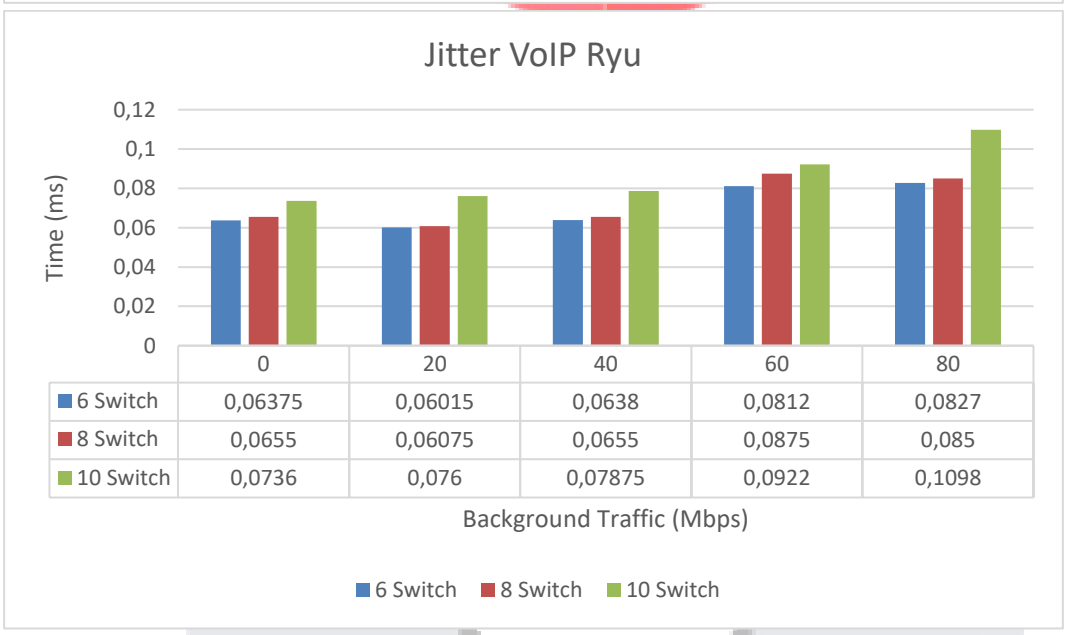

Delay Video Ryu
Time (ms)
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0
20
40
60
80
6 Switch 0,07 0,0811 0,0905 0,10225 0,10975
8 Switch 0,0725 0,098 0,1 0,1085 0,117
10 Switch 0,085 0,1075 0,122 0,1385 0,1415
Background Traffic (Mbps)
6 Switch
8 Switch
10 Switch

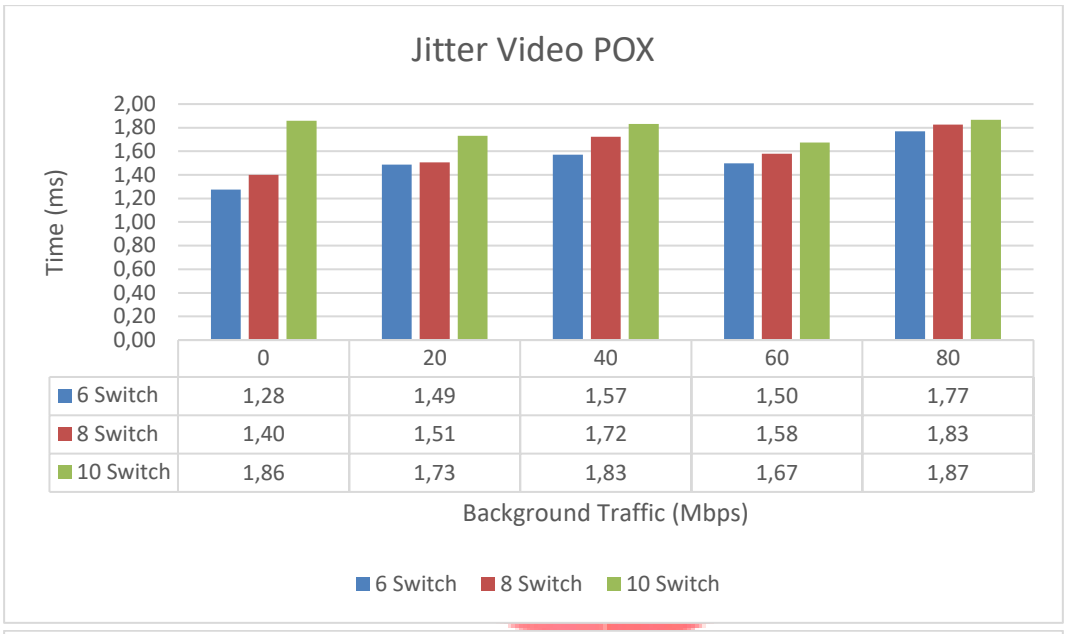
2. Jitter


Jitter Data POX
Time (ms)
0,12
0,1
0,08
0,06
0,04
0,02
0
0 20 40 60 80
6 Switch 0,0574 0,0686 0,0672 0,0664 0,0724
8 Switch 0,0784 0,0806 0,0808 0,0852 0,0888
10 Switch 0,0922 0,0922 0,0952 0,0954 0,103
Background Traffic (Mbps)
6 Switch 8 Switch 10 Switch

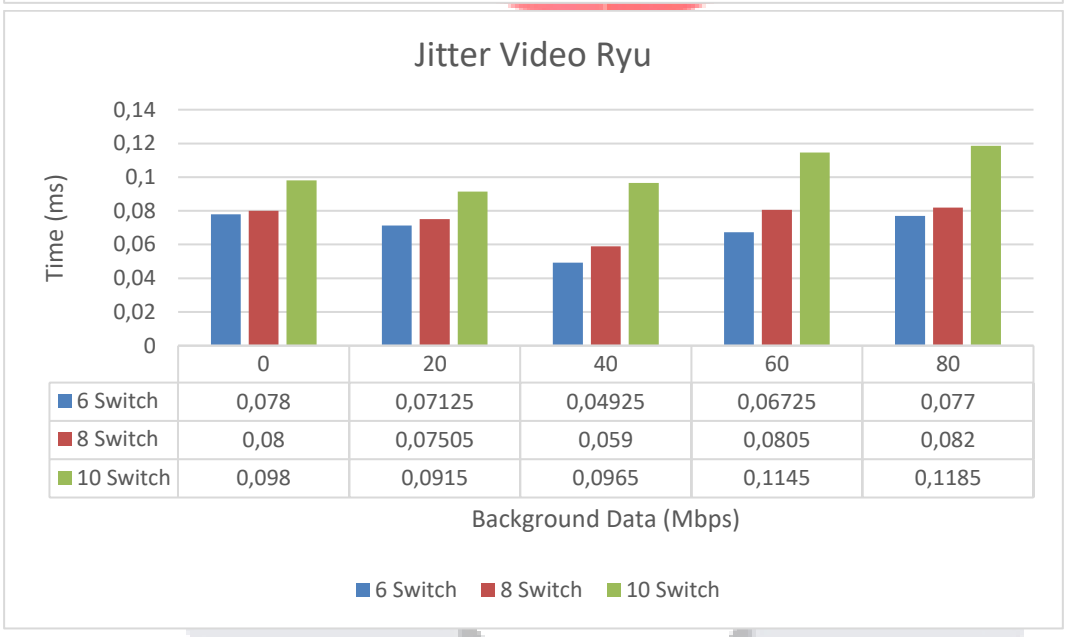

Jitter Data Ryu
Time (ms)
0,12
0,1
0,08
0,06
0,04
0,02
0
0 20 40 60 80
6 Switch 0,058 0,071 0,0708 0,0704 0,0768
8 Switch 0,06 0,0754 0,071 0,079 0,081
10 Switch 0,075 0,08 0,088 0,0924 0,1004
Background Traffic (Mbps)
6 Switch 8 Switch 10 Switch


Jitter VoIP POX
Time (ms)
0,3
0,25
0,2
0,15
0,1
0,05
0
0 20 40 60 80
6 Switch 0,201 0,214 0,228 0,227 0,216
8 Switch 0,224 0,242 0,262 0,258 0,231
10 Switch 0,246 0,266 0,277 0,267 0,277
Background Traffic (Mbps)
6 Switch 8 Switch 10 Switch


Jitter VoIP Ryu
Time (ms)
0,12
0,1
0,08
0,06
0,04
0,02
0
0 20 40 60 80
6 Switch 0,06375 0,06015 0,0638 0,0812 0,0827
8 Switch 0,0655 0,06075 0,0655 0,0875 0,085
10 Switch 0,0736 0,076 0,07875 0,0922 0,1098
Background Traffic (Mbps)
6 Switch 8 Switch 10 Switch

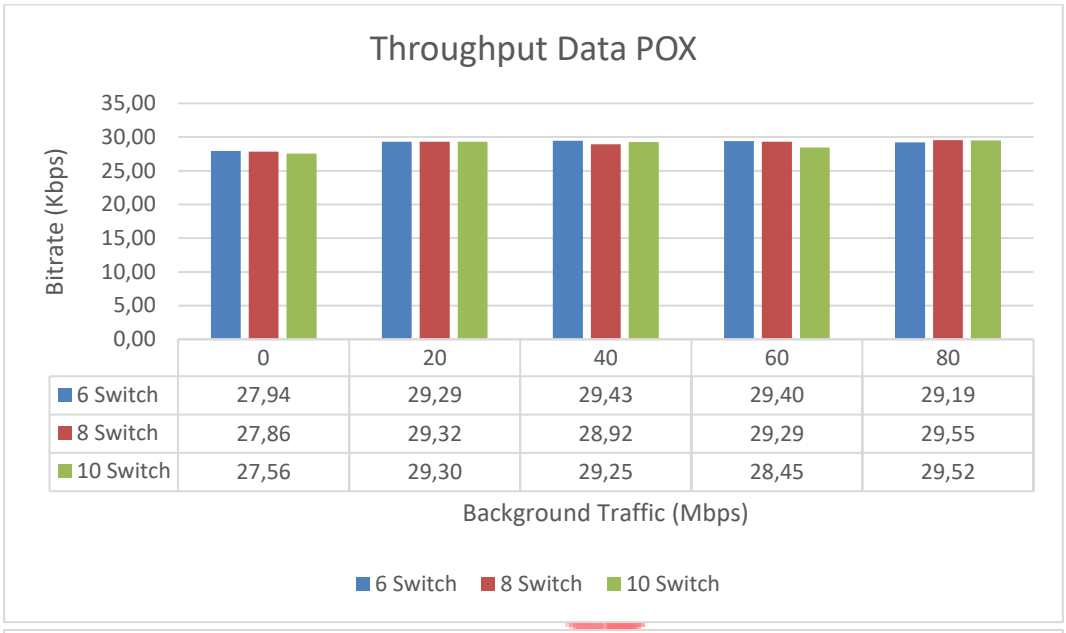


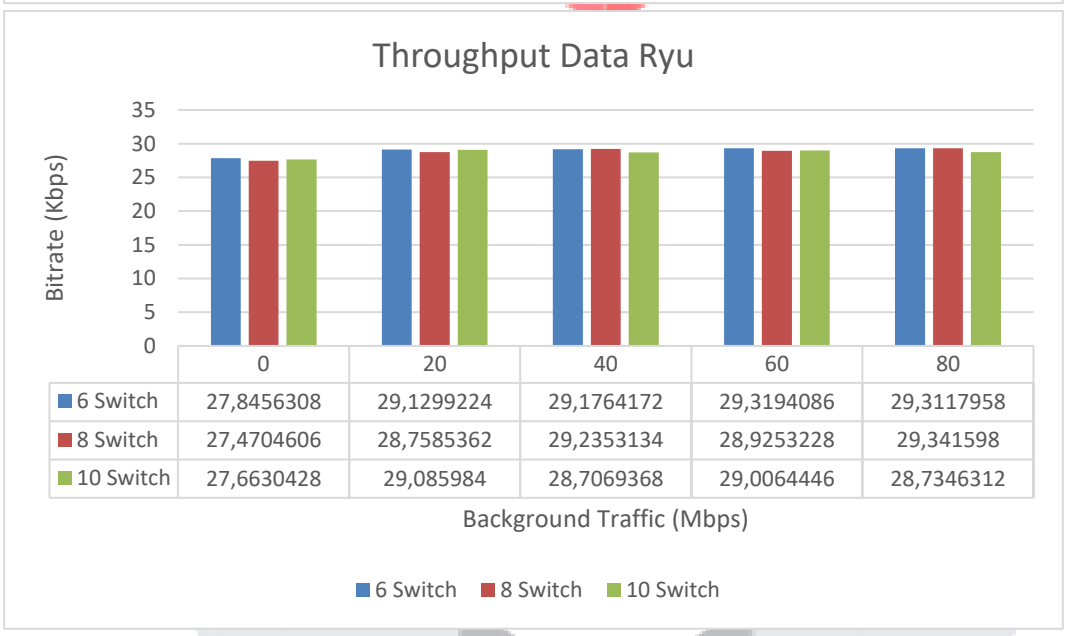
	0	20	40	60	80
6 Switch	1,28	1,49	1,57	1,50	1,77
8 Switch	1,40	1,51	1,72	1,58	1,83
10 Switch	1,86	1,73	1,83	1,67	1,87

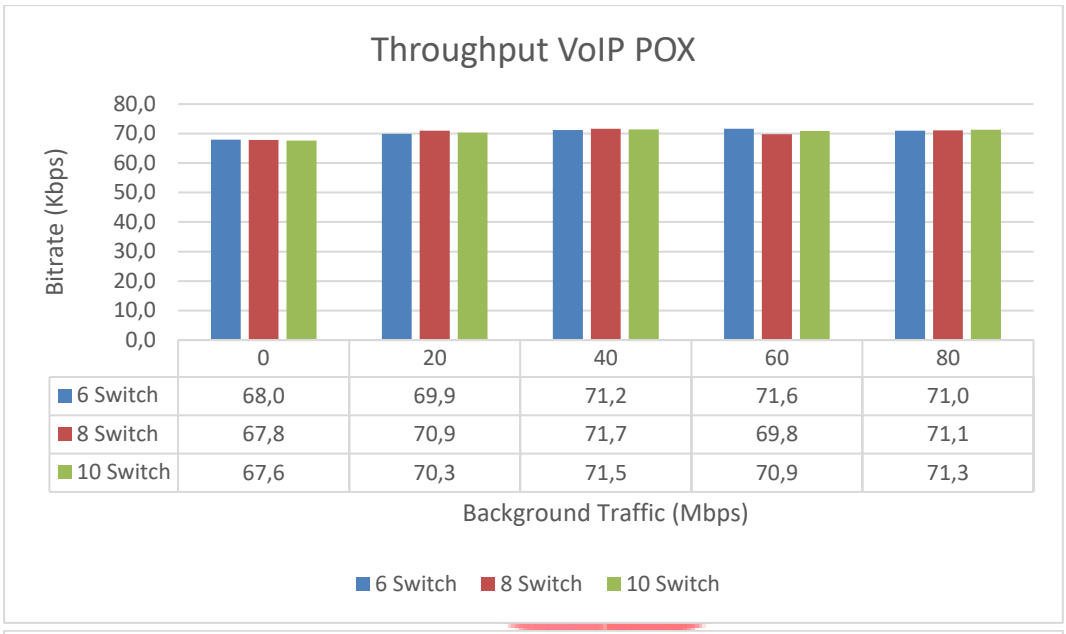


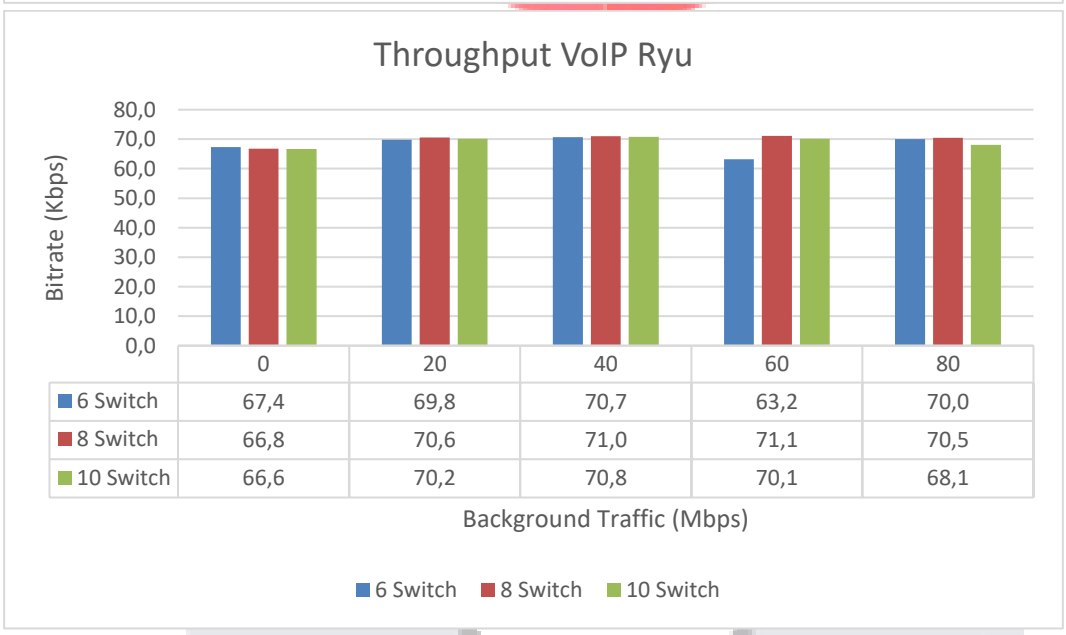
	0	20	40	60	80
6 Switch	0,078	0,07125	0,04925	0,06725	0,077
8 Switch	0,08	0,07505	0,059	0,0805	0,082
10 Switch	0,098	0,0915	0,0965	0,1145	0,1185

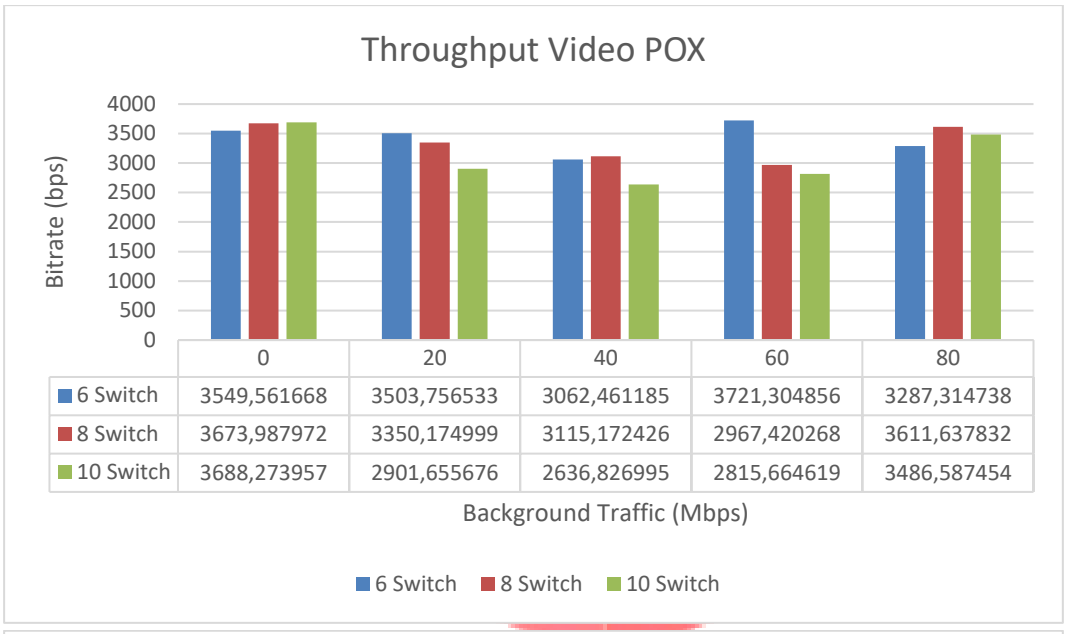
3. Throughput



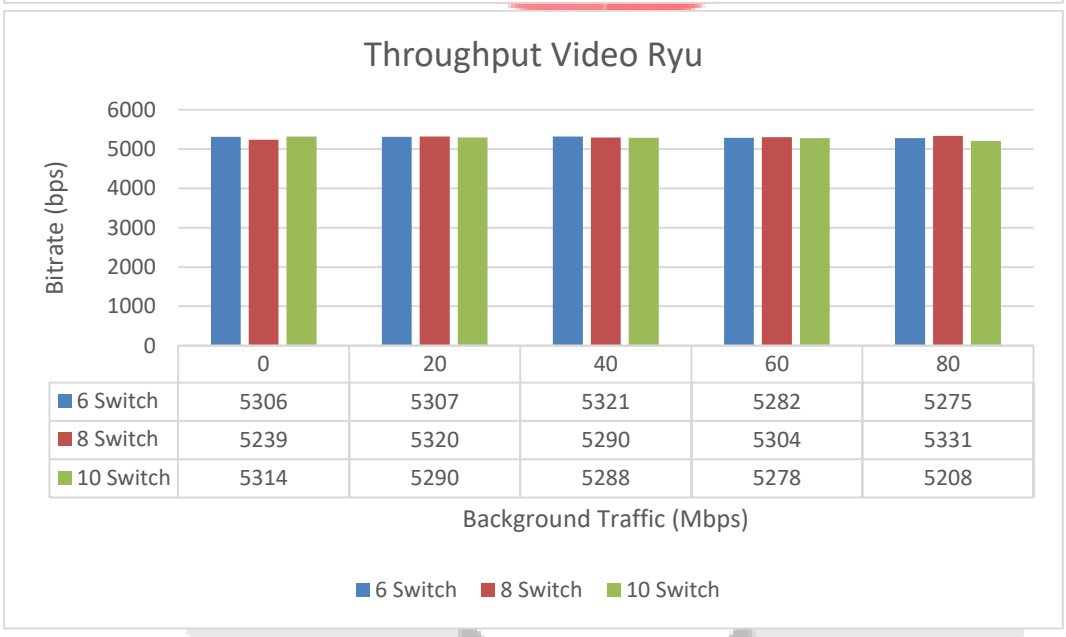



Throughput VoIP POX
Bitrate (Kbps)
80,0
70,0
60,0
50,0
40,0
30,0
20,0
10,0
0,0
0
20
40
60
80
6 Switch 68,0 69,9 71,2 71,6 71,0
8 Switch 67,8 70,9 71,7 69,8 71,1
10 Switch 67,6 70,3 71,5 70,9 71,3
Background Traffic (Mbps)
6 Switch
8 Switch
10 Switch


Throughput VoIP Ryu
Bitrate (Kbps)
80,0
70,0
60,0
50,0
40,0
30,0
20,0
10,0
0,0
0
20
40
60
80
6 Switch 67,4 69,8 70,7 63,2 70,0
8 Switch 66,8 70,6 71,0 71,1 70,5
10 Switch 66,6 70,2 70,8 70,1 68,1
Background Traffic (Mbps)
6 Switch
8 Switch
10 Switch



	0	20	40	60	80
6 Switch	3549,561668	3503,756533	3062,461185	3721,304856	3287,314738
8 Switch	3673,987972	3350,174999	3115,172426	2967,420268	3611,637832
10 Switch	3688,273957	2901,655676	2636,826995	2815,664619	3486,587454



	0	20	40	60	80
6 Switch	5306	5307	5321	5282	5275
8 Switch	5239	5320	5290	5304	5331
10 Switch	5314	5290	5288	5278	5208

4. Packet Loss