ANALISIS PERFORMA KONTROLER *RYU* DAN *PYRETIC* SEBAGAI KOMPONEN UTAMA PADA ARSITEKTUR *SOFTWARE DEFINED NETWORK*

Wicky Permadi Puji Asmara [1], Rakhmadhany Primananda, S.T, M.Kom [2], Sabriansyah Rizqika Akbar, S.T, M.Eng [3]

[1,2.3]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya

Jl. Veteran No. 8 Malang, Informatika, Gedung A FILKOM-UB

Email : wickyprmd@gmail.com[1], rakhmadhany@yahoo.com[2], sabrian@ub.ac.id[3]

ABSTRAK

Konsep jaringan *Software Defined Network* (SDN) adalah melakukan pemisahan antara *control plane* dan *data plane*. *data plane* berfungsi menentukan kemana paket akan diteruskan, sedangkan *control plane* mempunyai fungsionalitas infrastruktur jaringan seperti *routing* dan *switching*. Penelitian ini melakukan pengujian performa terhadap kontroler *Ryu* dan *Pyretic* dengan menggunakan Iperf yang bertujuan untuk mengetahui *throughput*, kapasitas *bandwidth*, mengetahui jumlah *jitter*, banyaknya *packet loss*, serta pemakaian CPU dan RAM pada jaringan SDN. Penelitian ini juga menggunakan algoritma *Non-weighted Dijkstra*. Berdasarkan rata-rata total skenario, nilai rata-rata *throughput* kontroler *Pyretic* mencapai 619,83 MBps lebih tinggi dibanding kontroler *Ryu* yang mencapai 454,99 MBps. Nilai rata-rata *bandwidth* pada *Pyretic* mencapai 5,13 Gbps lebih tinggi dibanding *Ryu* yang mencapai 3,78 Gbps. Nilai rata-rata *jitter* pada kontroler *Pyretic* mencapai 0,020 ms lebih rendah dibanding kontroler *Ryu* yang mencapai 0,027 ms. Nilai rata-rata *paket loss Pyretic* mencapai 0,334 % lebih rendah dibanding *Ryu* yang mencapai 0,631 %. Rata-rata pemakaian CPU pada kontroler *Pyretic* lebih rendah dibanding kontroler *Ryu*. ketika *idle Pyretic* butuh rata-rata 8,3 %, sedangkan *Ryu* butuh rata-rata 13,3 %. Ketika dilakukan pengujian QoS, *Pyretic* butuh rata-rata 58,6 % sedangkan *Ryu* membutuhkan rata-rata 64,2 %. Dan rata-rata pemakaian RAM pada kontroler *Pyretic* mencapai 12,7 % lebih rendah dibanding *Ryu* yang mencapai 13,6 %. Hasil analisis pengujian tersebut menunjukkan bahwa kontroler *Pyretic* lebih unggul dibanding kontroler *Ryu* dalam seluruh paramater uji. Dalam penelitian ini, artinya kontroler yang paling tepat digunakan adalah *Pyretic*.

Kata kunci : *Software Defined Network, Ryu*, *Pyretic, Iperf, throughput, bandwidth, jitter, packet loss*, CPU, RAM

ABSTRACT

The concept of Software Defined Network network is the separation between the control plane and the data plane. Data plane serves to determine where the packet will be forwarded, and the control plane has network functionality such as routing and switching. This research measures the performance of the two controllers, Ryu and Pyretic. It uses Iperf to determine throughput, bandwidth capacity, determine the amount of jitter, packet loss percentage, also the amount of CPU and RAM usage on the SDN. This research also applies Non-weighted Dijkstra algorithm.Based on the average of total scenario, the average Pyretic controller value of throughput reached 619,83 MBps, higher than Ryu controller which reached 454,99 MBps. The average of Pyretic bandwidth value reached 5,13 Gbps higher than the 3,78 Gbps of Ryu. The average value of jitter on the Pyretic controller reached 0,020 ms lower than Ryu controller which reached 0.027 ms. The average value of Pyretic packet loss reached 0.334 % lower than Ryu, which reached 0,631 %. The average of Pyretic CPU consumption is lower than the Ryu. when idle, Pyretic average reached 8,3 %, while Ryu average reached 13,3 %. When testing the QoS, Pyretic average reached 58,6 %, while Ryu average reached 64,2 %. And the average RAM usage on Pyretic reached 12,7 %, lower than Ryu which reached 13,6 %. The results of the analysis show that Pyretic controller is the better than Ryu controller in all QoS parameters test. In this research, it means that the most suitable controller to use is Pyretic.

Keywords: *Software Defined Network, Ryu, Pyretic, Iperf, throughput, bandwidth, jitter, packet loss, CPU, RAM*

1. PENDAHULUAN

Beberapa tahun terakhir banyak inovasi dibidang teknologi komputer yang dikembangkan. Berbagai macam teknologi seperti *smartphone, notebook,* konsol *online games* dikembangkan dengan mengakses infrastruktur jaringan yang ada. Tetapi jaringan utama saat ini belum mengalami perubahan. Infrastruktur jaringan tersebut adalah jaringan tradisional yang menggunakan algoritma khusus pada perangkat jaringan seperti *router* dan *switch* untuk mengelola dan

memelihara jaringan (Hu, 2014). Sekitar seperempat abad terakhir perangkat jaringan memiliki dua jenis pengolahan yaitu *data plane* dan *control plane* (Silver, 2013). Perangkat jaringan dapat mencapai 200, 2000 atau 20.000 dalam sebuah jaringan.

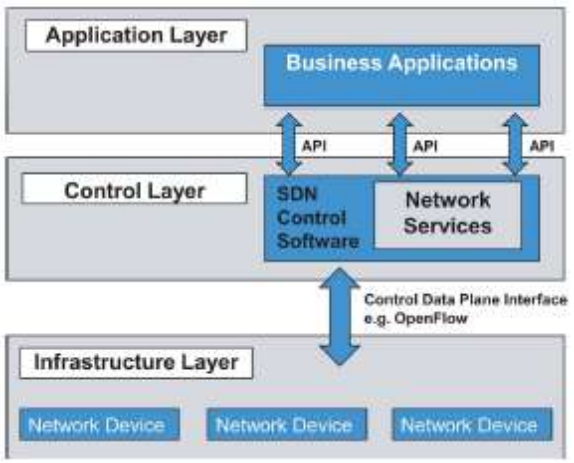
Permasalahan jaringan tradisional yang telah disebutkan membutuhkan solusi yang adaptif terhadap perkembangan kebutuhan jaringan, sehingga solusi yang ditawarkan cenderung pada pengendalian data menggunakan perangkat lunak daripada meningkatkan sumber daya perangkat keras jaringan. Peneliti mewujudkan hal tersebut dengan paradigma *Software Defined Network* (SDN). Pada SDN, fungsionalitas infrastruktur jaringan seperti *routing* dan *switching* dikendalikan oleh perangkat lunak sehingga hal ini lebih fleksibel dan adaptif terhadap perkembangan kebutuhan jaringan dibandingkan mekanisme jaringan tradisional yang membutuhkan manajemen *console* pada ratusan atau ribuan perangkat (Underdahl, 2015).

Konsep jaringan *Software Defined Network* adalah melakukan pemisahan antara *control plane* dan *data plane*. *Data plane* berfungsi menentukan kemana paket akan diteruskan, sedangkan *control plane* menjalankan fungsionalitas infrastruktur jaringan seperti *routing* dan *switching*. *control plane* diimplementasikan dalam bentuk perangkat lunak (Silver, 2013). Implementasi *control plane* menggunakan kontroler. Namun pemilihan kontroler yang tepat menjadi masalah dalam penerapan jaringan tersebut.

Kontroler yang digunakan dalam penelitian ini adalah *Ryu* dan *Pyretic* karena menurut Rao pada tahun 2015, *Ryu* mendukung dua dari *use case* paling penting yaitu *service insertion and chaining,* dan *transport networks*. Sedangkan *Pyretic* memiliki persamaan terhadap *Ryu* dari segi bahasa pemrograman python, dukungan *rest api*, dan dukungan *platform* sendiri. Untuk mendapatkan hasil perbandingan kedua kontroler, penelitian menggunakan parameter QoS sebagai parameter uji. Hal tersebut dipilih karena QoS mengacu pada kemampuan jaringan untuk memberikan layanan yang baik pada lalu lintas jaringan dari berbagai teknologi (Caudhary, 2012). Hal itu menjadi fokus penelitian ini yaitu mengukur performa *throughput, bandwidth, jitter, dan packet loss* pada kontroler *Ryu* dan *Pyretic*. Dan untuk menambah parameter pengujian kontroler ditambahkan parameter pemakaian CPU dan RAM untuk mengetahui pengaruh kontroler terhadap perangkat keras yang digunakan. Untuk menguji performa QoS serta pemakaian CPU dan RAM pada kedua kontroler tersebut diperlukan alat uji yang tepat. Iperf digunakan untuk menguji seluruh parameter QoS karena Iperf mendukung *tuning* dari berbagai parameter QoS yang terkait dengan *bandwidth, jitter,* dan *packet loss*. Untuk melakukan pengujian ini peneliti menggunakan simulator Mininet agar dapat menerapkan jaringan SDN tersebut. Dengan melakukan pengujian performa *QoS* serta pemakaian CPU dan RAM pada kontroler *Ryu* dan *Pyretic,* diharapkan penelitian ini dapat membantu pemilihan kontroler yang tepat.

2. *SOFTWARE DEFINED NETWORK*

Konsep jaringan SDN adalah melakukan pemisahan antara *Control Plane* dan *Data Plane*, dimana *Data Plane* tetap berada pada perangkat *Networking*, sedang *Control Plane* berada pada sebuah entitas terpisah bernama "Kontroler" yang akan menentukan perilaku jaringan dengan cara memungkinkan *Data Plane* untuk di program sehingga terbentuklah istilah *Software Defined Network* (SDN) yang mendefinisikan jaringan (Mendonca, 2013).



Gambar 1 Arsitektur *Software Defined Network*

Arsitektur SDN dapat dilihat sebagai tiga lapis bidang seperti pada gambar 1, meliputi :

- **Infrastruktur** (data-plane / infrastructure layer): terdiri dari elemen jaringan yg dapat mengatur SDN *Datapath* sesuai dengan instruksi yang diberikan melalui *Control-Data-Plane Interface* (CDPI)
- **Kontrol** (*control plane / layer*): entitas kontrol (*SDN Controller*) mentranslasikan kebutuhan aplikasi dengan infrastruktur dengan memberikan instruksi yg sesuai untuk SDN *Datapath* serta memberikan informasi yg relevan dan dibutuhkan oleh SDN *Application*
- **Aplikasi** (*application plane / layer*): berada pada lapis teratas, berkomunikasi dengan sistem via *NorthBound Interface* (NBI)

2.1 *RYU*

Ryu adalah sebuah *framework* dari *Software Defined Network* (SDN) yang berbasis komponen yang diimplementasikan sepenuhnya dalam *Python*. Seperti kontroler SDN lainnya, *Ryu* juga menyediakan komponen perangkat lunak dengan API yang dapat dikembangkan oleh *developer* untuk membuat manajemen jaringan dan aplikasi kontrol baru. Tujuan dari *Ryu* adalah untuk mengembangkan sebuah sistem operasi untuk SDN yang memiliki kualitas cukup tinggi untuk digunakan dalam jaringan besar (Al-Somaidai, 2014).

2.2 *PYRETIC*

Pyretic diperkenalkan sebagai bahasa pemrograman *Software Defined netwok* (Reich, 2013). Hal ini memungkinkan *programmer* untuk berfokus pada bagaimana menentukan *policy* jaringan. *Policy* ini bertindak sebagai sebuah fungsi atau *method* (Reich, 2013). *Programmer Pyretic* juga dapat membuat *dynamic policies* yang perilakunya akan berubah sewaktu-waktu. Singkatnya, *Pyretic* memungkinkan programmer SDN untuk membuat aplikasi jaringan modular (Jiang, 2014).

2.3 ALGORITMA *NON-WEIGHTED DIJKSTRA*

Algortima *Non-weighted Dijkstra* adalah algoritma *Dijkstra* yang tanpa menggunakan bobot atau *weight* dalam pengaplikasiannya, atau penggunaan nilai pada *edge* bernilai 1. Artinya Algortima *Non-weighted Dijkstra* mencari jalur dengan menghitung jumlah hop atau pasangan node awal dan node tujuan (Jiang, 2014). Algoritma tersebut dapat dilihat pada tabel 2.

Tabel 2 Algoritma *Non-weighted Dijkstra*

```
1   function Dijkstra(Graph, source):
2     create vertex set Q
3     for each vertex v in Graph:
4       dist[v] ← INFINITY
5       prev[v] ← UNDEFINED
6       add v to Q
7     dist[source] ← 0
8     while Q is not empty:
9       u ← vertex in Q with min dist[u]
10      remove u from Q
11      for each neighbor v of u:
12        if dist[u] + 1 < dist[v]:
13        dist[v] ← dist[u] + 1
14          prev[v] ← u
15    return dist[], prev[]
```

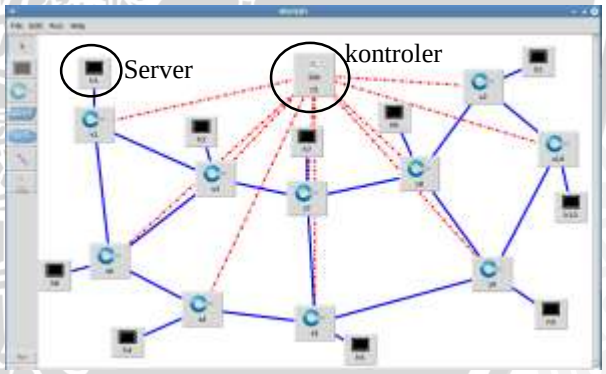
3. KONFIGURASI KONTROLER DAN TOPOLOGI

Konfigurasi kontroler pada kedua kontroler seperti tabel 1 ditentukan agar dapat digunakan pada tahap pengujian. Konfigurasi tersebut meliputi tipe yang digunakan kontroler, alamat IP, *port,* dan aplikasi yang digunakan masing-masing kontroler.

Tabel 1 Konfigurasi kontroler *Ryu* dan *Pyretic*

Parameter	Konfigurasi
Tipe Kontroler	*Remote Controller*
Alamat IP	127.0.0.1
Port yang digunakan	6633
Aplikasi (Algoritma)	*Non-weighted Dijkstra*

Pengujian performa kedua kontroler menggunakan mininet. Agar minnet dapat berjalan dengan baik diperlukan sebuah topologi. Penelitian ini menggunakan topologi yang sama pada kedua kontroler agar mendapatkan hasil yang akurat. Topologi yang digunakan adalah topologi yang menggunakan 10 *host* dan 10 buah *switch* yang dihubungkan dengan kontroler (c0). 10 *host* yang digunakan akan dibagi menjadi 1 *host* sebagai *server* dan 9 *host* menjadi *client.* Topologi yang digunakan dapat dilihat seperti gambar 2.



Gambar 2 Topologi pengujian

4. SKENARIO PENGUJIAN

Skenario pengujian dibagi menjadi tiga tahap dan memiliki beberapa skenario yang berbeda pada tiap tahap. Tahapan-tahapan tersebut yakni :

- **Tahapan pertama** : Pengujian terhadap parameter *throughput* dan *bandwidth* menggunakan protokol TCP. Tahap ini memiliki tiga skenario dan masing masing dilakukan sebanyak lima kali pengujian. Dan tiga skenario tersebut adalah :
  - **Skenario 1** : Pengujian *throughput* dan *bandwidth* dengan opsi perintah Iperf

dengan ukuran *window* standar yaitu 85,3 KB.
    - **Skenario 2** : Pengujian *throughput* dan *bandwidth* dengan opsi perintah Iperf dengan ukuran *window* lebih besar dari standar yaitu 170 KB.
    - **Skenario 3** : Pengujian *throughput* dan *bandwidth* dengan opsi perintah Iperf dengan ukuran *window* lebih kecil dari standar yaitu 42 KB.
- **Tahapan kedua** : Pengujian terhadap parameter *jitter* dan *packet loss* menggunakan parameter UDP. Tahap ini memiliki dua skenario dan masing masing dilakukan sebanyak lima kali pengujian. Dan dua skenario tersebut adalah :
    - **Skenario 1** : Pengujian *jitter* dan *packet loss* dengan menggunakan parameter batas *bandwidth* 100 Mbps
    - **Skenario 2** : Pengujian *jitter* dan *packet loss* dengan menggunakan parameter batas *bandwidth* 1024 Mbps
- **Tahapan ketiga** : Pengujian ini melakukan perekaman aktivitas CPU dan RAM yang dilakukan pada tahap pertama. Pengujian ini melakukan perekaman CPU dan RAM ketika kontroler sedang dinyalakan. Pengujian dilakukan selama satu menit dengan sepuluh detik pertama kontroler dalam keadaan *idle* dilanjutkan empat puluh detik selanjutnya pengujian parameter QoS dan sepuluh detik terakhir keadaan *idle* setelah pengujian parameter QoS.

5. ANALISIS JALUR *CLIENT* KE *SERVER*

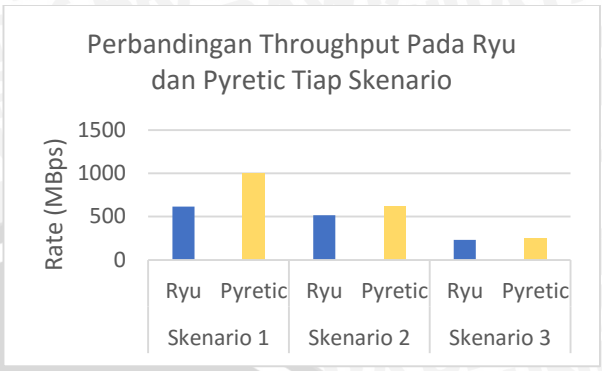
Jalur *client* ke *server* adalah jalur yang ditempuh dalam pengujian QoS, CPU dan RAM pada tiap *host*. Dari hasil pengujian didapatkan jalur yang terpilih. Jalur tersebut merupakan jalur yang ditempuh dari *host* sampai ke *server* seperti pada tabel 3 berikut:

Tabel 3 Jalur *client* ke *server*

H2	H3	H4	H5	H6
2-8-7-3-1	3-1	4-6-1	5-7-3-1	6-1
H7	**H8**	**H9**	**H10**	
7-3-1	8-7-3-1	9-5-7-3-1	10-9-5-7-3-1	

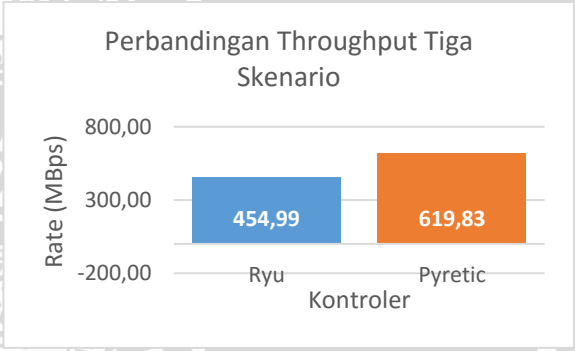
5.1 ANALISIS PERBANDINGAN *THROUGHPUT* PADA *RYU* DAN *PYRETIC*

Hasil pengujian terhadap *throughput* menghasilkan nilai rata-rata *throughput* pada 3 skenrio pengujian. Perbandingan 3 skenario pengujian *throughput* seperti gambar 3.



Gambar 3 Perbandingan *throughput* pada tiap skenario

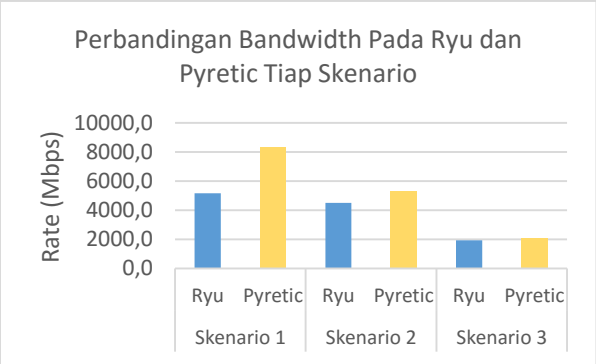
Berdasarkan gambar 3 hasil perbandingan 3 parameter menunjukkan skenario 1 yang memiliki nilai rata-rata *throughput* tertinggi. Skenario 1 menggunakan ukuran *TCP window* sebesar 85,3 KB, sedangkan pada skenario 2 sebesar 170KB, dan pada skenario 3 sebesar 42 KB. Jika membandingkan *throughput* secara keseluruhan skenario, maka hasil perbandingan *throughput* dapat dilihat seperti gambar 4.



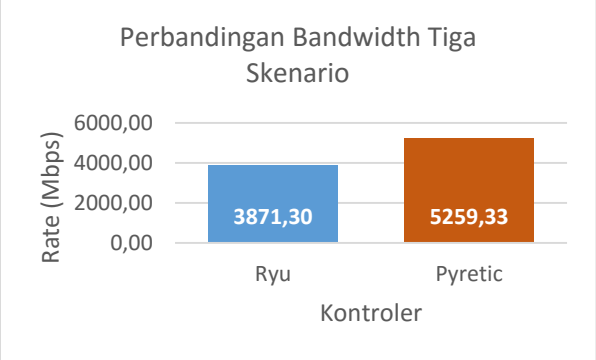
Gambar 4 Perbandingan *throughput*

Hasil secara keseluruhan rata-rata *throughput* pada kontroler *Pyretic* lebih tinggi dibanding dengan nilai *throuhput* kontroler *Ryu*. Nilai rata-rata *throughput Pyretic* mencapai 619,83 MBps lebih besar dibanding kontroler *Ryu* yang hanya 454,99 MBps. Tinggi rata-rata *throughput* dipengaruhi oleh ukuran *TCP window* yang digunakan dalam pengujian. Selain itu nilai rata-rata *throughput* juga dipengaruhi pada jalur transmisi yang dilalui. Jika jalur semakin banyak *switch* yang dilalui maka nilai *throughput* akan semakin rendah. Perbedaan performa kontroler terjadi karena perbedaan versi protokol *OpenFLow.*

5.2 ANALISIS PERBANDINGAN *BANDWIDTH* PADA *RYU* DAN *PYRETIC*



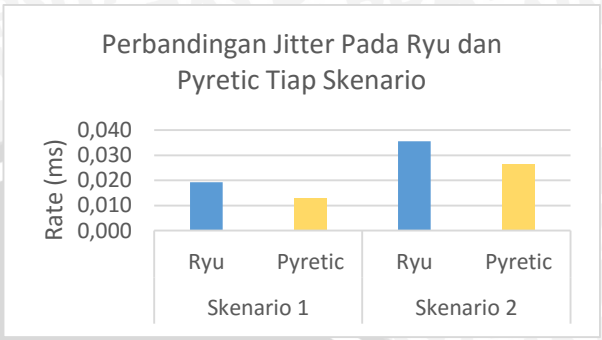
Gambar 5 Perbandingan *bandwidth* tiap skenario



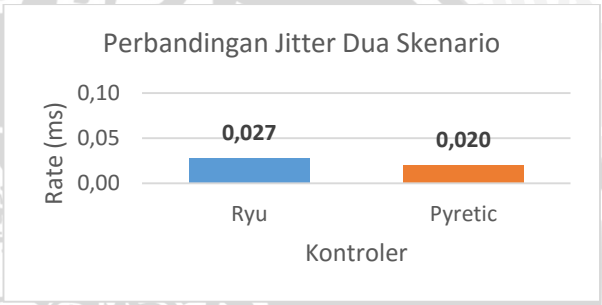
Gambar 6 Perbandingan *bandwidth*

Berdasarkan gambar 5, hasil perbandingan 3 parameter menunjukkan skenario 1 yang memiliki nilai rata-rata *bandwidth* tertinggi. Pada gambar 6 secara keseluruhan rata-rata *bandwidth* pada kontroler *Pyretic* lebih tinggi dibanding dengan nilai *throuhput* pada kontroler *Ryu*. Nilai rata-rata *bandwidth Pyretic* mencapai 5259,33 Mbps juga lebih besar dibanding kontroler *Ryu* yang hanya 3871,3 Mbps. Nilai *bandwidth* juga pada kedua kontroler juga terpengaruh pada jalur transmisi yang dilalui sama seperti *throughput*.

5.3 ANALISIS PERBANDINGAN *JITTER* PADA *RYU* DAN *PYRETIC*



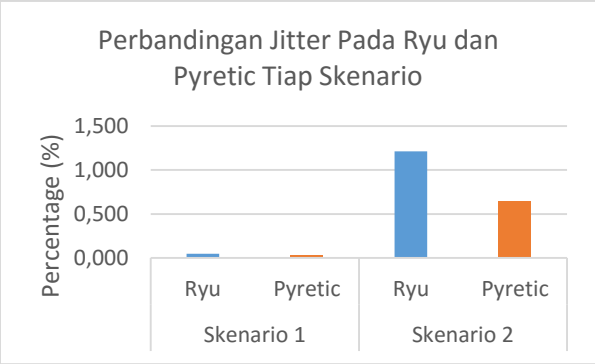
Gambar 7 Perbandingan *jitter* tiap skenario



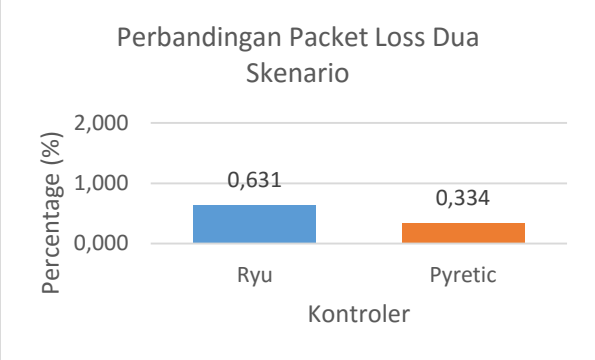
Gambar 8 Perbandingan *jitter*

Bedasarkan gambar 7 perbandingan dari dua skenario menunjukkan bahwa skenario 2 memiliki nilai rata-rata jitter lebih tinggi . Berdasarkan gambar 8 hasil secara keseluruhan bahwa nilai *jitter* pada kedua kontroler termasuk sangat bagus dalam kategori degradasi. Nilai rata-rata *jitter* pada kontroler *Pyretic* lebih rendah daripada nilai *jitter* pada *Ryu*. Nilai rata-rata *jitter Pyretic* lebih rendah dengan waktu 0,020 ms sedangkan kontroler *Ryu* mempunyai rata-rata waktu 0,027 ms. Selain batas *bandwidth* yang ditentukan yakni 100 Mbps dan 1024 Mbps, hal ini juga dipengaruhi ukuran *buffer* UDP dalam pengujian. Karena ukuran *buffer UDP* sebesar 208 KB maka jumlah *packet loss* masih memiliki persentase yang sangat bagus.

5.4 ANALISIS PERBANDINGAN *PACKET LOSS* PADA *RYU* DAN *PYRETIC*



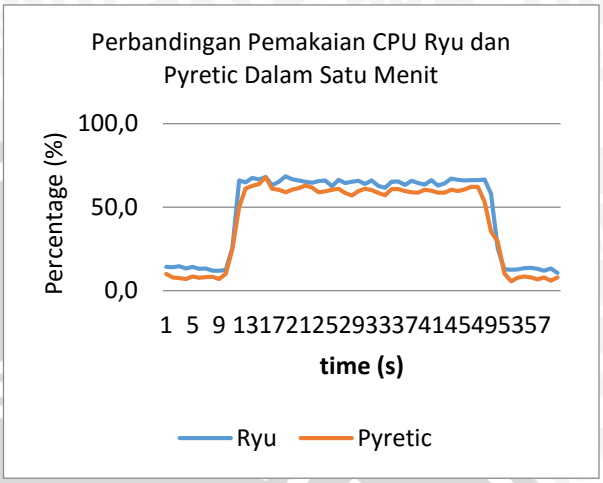
Gambar 9 Perbandingan *packet loss* tiap skenario



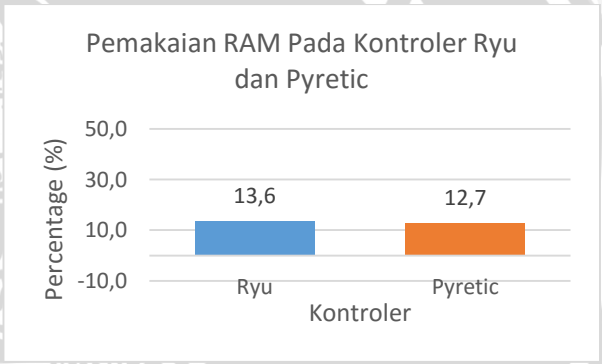
Gambar 10 Perbandingan *packet loss*

Berdasarkan gambar 9 hasil perbandingan 2 skenario menunjukkan skenario 2 yang memiliki nilai rata-rata *packet loss* lebih tinggi. Secara keseluruhan bahwa nilai *packet loss* pada kedua kontroler termasuk sangat bagus dalam kategori degradasi seperti pada gambar 4.8. berdasarkan gambar 10 nilai rata-rata *packet loss Pyretic* juga lebih rendah dengan persentase sebesar 0,334 % sedangkan kontroler *Ryu* mempunyai prosentase sebesar 0,631 %. Selain batas *bandwidth* yang ditentukan yakni 100 Mbps, hal ini juga dipengaruhi kesetabilan jaringan internet yang tersedia dan ukuran *buffer* UDP dalam pengujian. Karena ukuran *buffer UDP* yang cukup besar sebesar 208 KB dengan jumlah transfer 120 MB selama 10 detik. Maka *jumlah packet loss* masih memiliki persentase yang sangat bagus.

5.5 ANALISIS PERBANDINGAN PEMAKAIAN CPU DAN RAM PADA *RYU* DAN *PYRETIC*



Gambar 11 Perbandingan pemakaian CPU



Gambar 12 Perbandingan pemakaian RAM

Berdasarkan pada gambar 11 hasil secara keseluruhan pemakaian CPU pada kontroler *Ryu* lebih rendah dibanding dengan pemakaian CPU pada kontroler *Pyretic*. Persentase pemakaian CPU pada kontroler *Ryu* lebih rendah baik dalam keadaan *idle* maupun pengujian QoS. rata-rata pemakaian CPU pada kontroler *Ryu* sebesar 13,3 % dan 8,3 % pada kontroler *Pyretic*. Hasil pengamatan pemakaian CPU pada 40 detik selanjutnya menunjukkan bahwa pemakaian CPU pada kontroler *Ryu* mencapai rata-rata 64,2 % lebih rendah daripada kontroler *Pyretic* yang rata-ratanya mencapai 58,6 %. Hal ini berarti penggunaan kontroler *Pyretic* membutuhkan *resourse hardware* yang lebih tinggi.

Secara keseluruhan pemakaian RAM pada kedua kontroler relatif rendah. Namun kontroler *Ryu* memiliki jumlah pemakaian lebih besar dibanding dengan pemakaian pada kontroler *Pyretic*. Berdasarkan gambar 12 hasil perbandingan menunjukkan bahwa rata-rata pemakaian RAM kontroler *Ryu* mencapai 13,6 % sedangkan kontroler *Pyretic* mencapai 12,7 %.

Pemakaian RAM ini merupakan jumlah beban yang diterima sistem ketika sedang menjalankan pengujian terhadap QoS

6. KESIMPULAN

Hasil Rata-rata nilai *throughput* pada kontroler *Pyretic* lebih tinggi dibanding dengan nilai *throuhput* pada kontroler *Ryu*. Nilai rata-rata *throughput Pyretic* mencapai 619,83 MBps lebih besar dibanding kontroler *Ryu* yang hanya 454,99 MBps. Nilai rata-rata *throughput* pada kedua kontroler terpengaruh pada jalur transmisi yang dilalui. Jika jalur transmisi menuju *server* semakin panjang atau semakin banyak switch maka nilai *throughput* akan semakin rendah. Hasil analisis *bandwidth* menunjukkan bahwa kontroler *Pyretic* juga memiliki nilai rata-rata *bandwidth* yang lebih besar dibanding dengan kontroler *Ryu.* Nilai rata-rata *bandwidth Pyretic* mencapai 5259,33 Mbps juga lebih besar dibanding kontroler *Ryu* yang hanya 3871,3 Mbps. Nilai rata-rata *bandwidth* juga terpengaruh dengan jarak jalur yang dilalui. Semakin pandang jalur yang dilalui maka semakin rendah nilai rata-rata *bandwidth* yang diperoleh.

Rata-rata nilai *jitter* pada kontroler *Pyretic* lebih tinggi dibanding dengan kontroler *Ryu*. Nilai rata-rata *jitter Pyretic* lebih rendah dengan waktu 0,020 ms sedangkan kontroler *Ryu* mempunyai rata-rata waktu 0,027 ms. Selain batas *bandwidth* yang ditentukan yakni 100 Mbps dan 1024 Mbps, hal ini juga dipengaruhi ukuran *buffer* UDP dalam pengujian. Karena ukuran *buffer UDP* sebesar 208 KB maka jumlah *packet loss* masih memiliki persentase yang sangat bagus. Nilai rata-rata *packet loss* pada kontroler *Pyretic* lebih rendah daripada nilai *packet loss* pada *Ryu*. Nilai rata-rata *packet loss Pyretic* juga lebih rendah dengan persentase sebesar 0,334 % sedangkan kontroler *Ryu* mempunyai prosentase sebesar 0,631 %. Selain batas *bandwidth* yang ditentukan yakni 100 Mbps, hal ini juga dipengaruhi kesetabilan jaringan internet yang tersedia dan ukuran *buffer* UDP dalam pengujian. Karena ukuran *buffer UDP* yang cukup besar sebesar 208 KB dengan jumlah transfer 120 MB selama 10 detik. Maka *jumlah packet loss* masih memiliki persentase yang sangat bagus.

Persentase pemakaian CPU pada kontroler *Ryu* lebih rendah baik dalam keadaan *idle* maupun pengujian QoS. rata-rata pemakaian CPU pada kontroler *Ryu* sebesar 13,3 % dan 8,3 % pada kontroler *Pyretic*. Hasil pengamatan pemakaian CPU pada 40 detik selanjutnya menunjukkan bahwa pemakaian CPU pada kontroler *Ryu* mencapai rata-rata 64,2% lebih rendah daripada kontroler *Pyretic* yang rata-ratanya mencapai 58,6%. Sedangkan rata-rata pemakaian RAM kontroler *Ryu* mencapai 13,6% sedangkan kontroler *Pyretic* mencapai 12,7%.Hal ini berarti penggunaan kontroler *Pyretic* membutuhkan *resourse hardware* yang lebih tinggi.

REFERENSI

Al-Somaidai, M. B. & Yahya, E. B., 2014. Survey of software components to emulate *OpenFlow*. *American Journal of Software Engineering and Applications,* Volume 3, pp. 74-78.

Anggara, S. M., 2015. *Pengujian Performa Kontroler Software-defined Network (SDN): POX dan Floodlight.* Bandung: Institut Teknologi Bandung.

]Ashton, M., 2014. *Ten Things to Look For in an SDN Controller.*

Chaudhary, D. D., & Waghmare, L. M., 2012. *Quality of service analysis in wireless sensor network by controlling end-to-end delay*. pp. 703-708. IEEE.

ETSI, 1999. *Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS).* V2.1.1 penyunt. Valbonne: European Telecommunications Standards Institute.

Hu, F., Hao, Q. & Bao, K., 2014. *A Survey on Software-defined Network and OpenFlow: From Concept to Implementation.* Alabama: University of Alabama.

Iperf., *iPerf - The ultimate speed test tool for TCP, UDP and SCTP.* [online] tersedia di : < https://iperf.fr/iperf-doc.php> [diakses pada 8 April 2016]

Jiang, J. R., Huang, H. W. & Chen, S. Y., 2014. *Extending Dijkstra's Shortest Path Algorithm for Software Defined Network.* Taiwan: National Central University.

Kim, H. & Feamster, N., 2013. *Improving Network Management with Software Defined Networking.* Georgia: Georgia Institue of Technology.

Mendonca, M. et al., 2013. *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks.*

Pratama, P. N., 2015. *ANALISIS PERFORMANSI KONTROLER FLOODLIGHT DAN BEACON SEBAGAI KOMPONEN UTAMA PADA ARSITEKTUR SOFTWARE DEFINED NETWORK.* Malang: Universitas Brawijaya.

Rao, S., 2015. *SDN Series Part Eight: Comparison Of Open Source SDN Controllers.*The New Stack.

Reich, J. et al., 2013. *Modular SDN Programming with Pyretic.*

Ross, K. & Kurose, J., 2013. *Computer Networking : a top-down approach.* 6 penyunt. New Jersey: Pearson Education, Inc.

Seehorn, A., 2016. *What is the Difference Between CPU Usage & RAM?*. [online] tersedia di : <https://www.techwalla.com/articles/what-is-the-difference-between-cpu-usage-ram> [diakses pada 28 November 2016]

Silver, J., 2013. *SDN 101: What It Is, Why It Matters, and How to Do It.* [online] tersedia di : <http://blogs.cisco.com/ciscoit/sdn-101-what-it-is-why-it-matters-and-how-to-do-it> [diakses pada 8 April 2016]

Underdahl, B. & Kinghorn, G., 2015. *Software Defined Networking For Dummies.* New Jersey: John Wiley & Sons, Inc.

Wang, S. Y., Chiu, H. W. & Chou, C. L., 2015. *Comparisons of SDN OpenFlow Controllers over EstiNet: Ryu vs. NOX.* Taiwan: National Chiao Tung University.

[WIB-14] Wibowo, M., 2014. *Analisis dan Implementasi Quality Of Service (Qos) Menggunakan Ipcop Di Smk Muhammadiyah Imogiri*. Yogyakarta: AMIKOM YOGYAKARTA