PERANCANGAN SISTEM MONITORING PADA SDN (*SOFTWARE DEFINED NETWORK*) BERBASIS *WEB* DENGAN MENGGUNAKAN PROTOKOL REST

DESIGN OF WEB-BASED MONITORING SYSTEM IN SDN (SOFTWARE DEFINED NETWORK) USING REST PROTOCOL

Depa Panjie Purnama[1], R. Rumani M.[2], Sofia Naning Hertiana[3]
[1,2]Prodi S1 Sistem Komputer, [3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik, Universitas Telkom
[1]depapp@students.telkomuniversity.ac.id, [2]rumani@telkomuniversity.ac.id, [3]sofiananing@telkomuniversity.ac.id

Abstrak

Software Defined Network (SDN) merupakan suatu paradigma yang merubah cara mengatur, mengontrol dan merancang jaringan. SDN membuat suatu jaringan dapat diprogram sesuai dengan kebutuhan yang ada. Salah satu protokol yang mendukung SDN yaitu OpenFlow. Pada OpenFlow ini, antara perangkat kontrol (*control plane*) dan perangkat penyalur paket data (*data plane*) dipisahkan. Perangkat kontrol tersebut dipusatkan pada sebuah *controller*. SDN masih dalam tahap pengembangan para peneliti sehingga masih terdapat beberapa fitur yang dibutuhkan kedepannya. Salah satu fitur yang menjadi kebutuhan dari SDN ini adalah monitoring. Monitoring bertujuan untuk memantau keadaan jaringan yang sudah mengimplementasikan SDN. Pada tugas akhir ini, penulis memberikan solusi atas permasalahan yang ada yaitu dengan membangun suatu aplikasi monitoring pada SDN berbasis *web*. Berdasarkan hasil dari pengujian *alpha*, aplikasi monitoring ini dapat berjalan dengan baik dan benar memonitoring jaringan sesuai yang diinginkan. Untuk pengujian *beta*, didapatkan skor *rating* sebesar 4.01 dengan *range* skor antara 1-5. Pada pengujian *response time*, waktu rata-rata yang dibutuhkan untuk menampilkan data yang diminta yaitu selama 0,0123 detik untuk pengujian jumlah *switch* yang berbeda-beda dan 0,0134 detik untuk pengujian pada jumlah *link* yang berbeda-beda.

Kata kunci: SDN, monitoring, *controller* Ryu

Abstract

Software Defined Network (SDN) is a paradigm that is changing the way regulate, control and design networks. SDN make a network can be programmed according to the existing needs. One protocol that supports SDN is OpenFlow. In OpenFlow, between the control device (control plane) and the forwarding device (data plane) are separated. The control device is on a controller. SDN is still in the development of the researchers, there are still some features that are required in the future. One of the features that become a necessity of SDN are monitoring, monitoring can monitor the state of the network that have already implemented SDN. In this thesis, the author gives a solution to the existing problems is to build an application on the SDN web-based monitoring. Based on the results of alpha testing, the monitoring application can work well. For the beta testing, the rating score is 4,01 from range score between 1-5. In response time testing, the average time to display the data requested is 0.0123 seconds for the test on different switches and 0.0134 seconds for the test on different links.

Keyword: SDN, monitoring, Ryu Controller

1. Pendahuluan

Jaringan komputer bukanlah sesuatu hal yang baru saat ini. Hampir disetiap perusahaan, sekolah dan kampus terdapat jaringan komputer untuk memperlancar arus informasi yang ada. Internet yang sekarang sudah banyak digunakan oleh hampir setiap orang merupakan suatu jaringan komputer dalam skala besar. Semakin berkembangnya pengguna jaringan komputer ini akan menimbulkan kompleksitas yang tinggi dalam hal manajemen jaringan, oleh karena itu dibutuhkan sebuah solusi agar *network administrator* dapat dengan mudah mengelola dan mengimplementasikan perangkat jaringan tersebut.

Solusi yang memungkinkan untuk permasalahan tersebut yaitu dengan melakukan pemisahan antara *control plane* dan *data plane*, dimana *control plane* akan diletakan secara terpusat pada sebuah *controller* [2]. Untuk mewujudkan hal tersebut, dibutuhkan sebuah *Application Program Interface* (API) untuk dapat mengkoneksikan seluruh perangkat jaringan kedalam sebuah *contoller* yang dapat di program sesuai dengan kebutuhan yang ada, dari hal tersebut sebuah paradigma baru pada dunia jaringan komputer muncul, yaitu *Software Defined Network* (SDN). Inti dari SDN sendiri yaitu memisahkan antara *control plane* dan *data plane* kedalam perangkat yang berbeda dan jaringan dapat diatur atau didefinisikan melalui sebuah *software*. Untuk dapat mengetahui informasi tentang *traffic* yang ada pada *controller* teresbut, dibutuhkan suatu sistem/aplikasi monitoring.

Pada penelitian sebelumnya yang berjudul "*Network Management and Performance Monitoring using Software Defined Networks*" [7], mengemukakan bahwa dibutuhkannya suatu sistem/aplikasi monitoring yang dapat menampilkan semua informasi yang ada dalam *controller*. Dalam penelitian tersebut, belum dilakukannya pembuatan sistem monitoring berbasis *web* sehingga pada tugas akhir ini diusulkan untuk melakukan pembuatan sistem monitoring pada SDN secara *real-time* dengan menggunakan *controller* Ryu.

2. Dasar Teori dan Perancangan

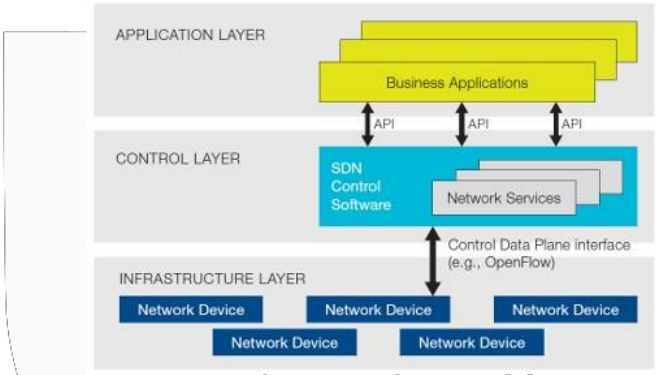
2.1. *Software Defined Network* (SDN)

Software Defined Networking (SDN) [3] merupakan suatu paradigma baru dalam dunia jaringan. SDN memisahkan antara *data plane* yang berfungsi sebagai penerus paket data dengan *control plane* yang berfungsi sebagai pengontrol perangkat. Hal ini sangat berbeda dengan jaringan konvensional yang masih banyak digunakan saat ini, dimana pada jaringan konvensional *control plane* (perangkat pengontrol) dan *data plane* (perangkat penerus paket data) disatukan dalam suatu perangkat jaringan yang sama seperti switch atau router.

2.1.1. Arsitektur dan Karakteristik SDN

SDN [5] memiliki arsitektur dan karakteristik tersendiri yang berbeda dengan jaringan konvensional, dimana arsitektur SDN terdiri dari tiga buah *layer* yang dapat diakses melalui API, yaitu:

- *Application Layer*, terdiri dari bisnis aplikasi *end-user* untuk layanan komunikasi SDN. *Application Layer* dan *Control Layer* dibatasi oleh sebuah *Northbound* API
- *Control Layer*, menyediakan fungsi kontrol terhadap perilaku *forwarding* di dalam jaringan melalui sebuah *open interface.*
- *Infrastruktur Layer*, terdiri dari elemen dan perangkat jaringan yang dapat menyediakan *packet switching* dan *forwarding*.



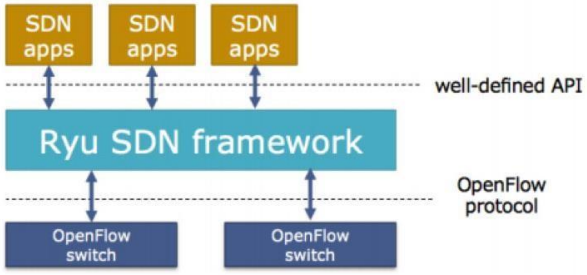
Gambar 1. Arsitektur SDN [5]

2.1.2. SDN *Controller*

Controller pada SDN yang memiliki perkembangan secara cepat yaitu controller yang berbasis OpenFlow. OpenFlow [4] merupakan standar pertama 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 (*hypervisor-based*). OpenFlow sendiri berfungsi sebagai *Southbound* API pada arsitektur SDN.

2.1.3. Ryu *Controller*

Ryu *Controller* [6] merupakan salah satu contoh dari *Controller* SDN yang ada. Ryu *Controller* sendiri mendukung berbagai protokol untuk mengelola perangkat jaringan, seperti OpenFlow, netconf dan OF-config. Untuk OpenFlow sendiri, Ryu *Controller* mendukung versi-versi yang ada yaitu versi 1.0, 1.2, 1.3, 1.4 dan 1.5. Ryu *Controller* sendiri menggunakan bahasa pemrograman Python.



Gambar 2. Arsitektur Ryu *Controller* [6]

2.2. Aplikasi Monitoring

Monitoring jaringan adalah salah satu fungsi dari *network management* yang berguna untuk menganalisa keadaan suatu jaringan. Tujuan dari monitoring jaringan adalah pengumpulan informasi yang berguna dari berbagai macam bagian dari jaringan sehingga jaringan tersebut dapat dikelola dan dikontrol dengan menggunakan informasi yang telah dikumpulkan tersebut. Pada SDN, monitoring dapat dilakukan dengan cara mengakses langsung *controller* dengan menggunakan *open* API yang disebut *Northbound* API. *Southbound* API berfungsi sebagai penghubung antara *Infrastructure Layer* dan *Control Layer*, salah satu contoh dari *Southbound* API ini adalah Protokol OpenFlow. Lalu selain *Southbound* API ada juga *Northbound* API yang berfungsi sebagai penghubung antara *Control Layer* dan *Application Layer*, salah satu contoh dari *Northbound* API yaitu Protokol REST [8]

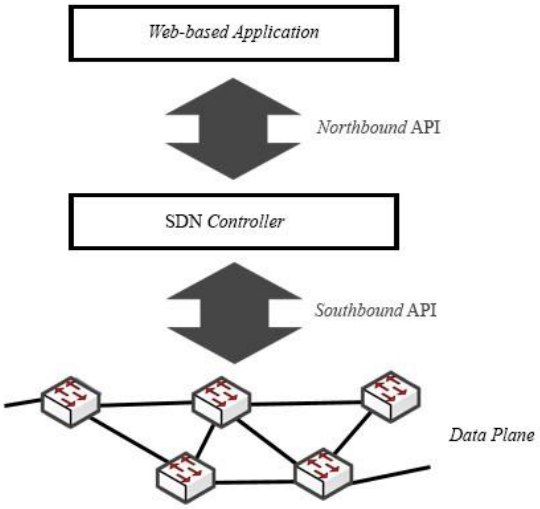
2.3. Mininet

Mininet [1] merupakan sebuah emulator jaringan yang dapat menggambarkan jaringan yang besar dengan hanya menggunakan sebuah PC/laptop. Mininet bersifat *open-source*, sehingga proyek yang telah dilakukan berupa *source code*, *scripts*, dan dokumentasi yang dapat dikembangkan oleh siapa saja. Mininet sendiri memiliki kelebihan sebagai berikut:

- *Prototype* jaringan yang telah dibuat dengan menggunakan emulator Mininet ini memiliki perilaku jaringan yang sama dengan jaringan yang sebenarnya
- Karena bersifat *open-source*, *prototype* yang telah dibuat dengan menggunakan Mininet ini dapat dikembangkan oleh semua orang secara bebas dan bisa juga dikembangkan bersama-sama.
- Dengan menggunakan Mininet, *prototype* jaringan yang dibuat dapat menggambarkan jaringan dengan jumlah *switch* yang banyak hanya dalam satu perangkat PC/laptop.

Selain memiliki kelebihan, adapula kekurangan yang dimiliki Mininet itu sendiri yaitu Mininet tidak dapat berjalan pada sistem operasi *non*-linux yang tidak mendukung switch OpenFlow.

2.4. Gambaran Umum Aplikasi



Gambar 3. Gambaran umum sistem

Pada Gambar 3 merupakan gambaran umum dari sistem monitoring. Sistem tersebut akan terdiri dari lima bagian. Kelima bagian tersebut diantaranya:

1. *Data Plane*

   Data plane yang merupakan perangkat penyalur paket data dalam jaringan yang mengimplementasikan SDN di dalamnya dibangun menggunakan emulator Mininet 2.2.1. dan menggunakan Open vSwitch 2.0.2. Pada bagian pengujian, bagian *data plane* ini akan menggunakan beberapa variasi, seperti perubahan node dan perubahan topologi.
2. *Southbound* API

   Southbound API merupakan penghubung antara *data plane* dan *control plane* pada jaringan yang mengimplementasikan SDN. *Southbound* API pada sistem monitoring ini menggunakan Protokol OpenFlow.
3. SDN *Controller*

   SDN *Controller* merupakan pusat dari perangkat kontrol yang ada pada jaringan tersebut. SDN *Controller* pada sistem monitoring ini menggunakan *controller* Ryu.
4. *Northbound* API

   Northbound API merupakan penghubung antara SDN *Controller* dan Aplikasi Web Monitoring pada sistem ini. *Northbound* API pada sistem ini menggunakan Protokol REST.
5. *Web-based Application*

   Web-based Application merupakan aplikasi web yang menampilkan keluaran dari sistem monitoring ini. Aplikasi web ini dibangun dengan menggunakan PHP versi 5.6.19

3. Pengujian

3.1. Parameter Pengujian

Pada sistem ini dilakukan pengujian pada aplikasi *web* dan *controller*. Pengujian yang dilakukan pada aplikasi *web* berupa pengujian *alpha* (pengujian *blackbox*) dan pengujian *beta* (*Mean Opinion Score* atau MOS). Pengujian yang dilakukan pada *controller* berupa pengujian *response time*.

3.2. Pengujian Aplikasi *Web*

3.2.1. Rencana Pengujian

Pengujian yang dilakukan pada aplikasi *web* adalah pengujian *alpha* dan pengujian *beta*. Pada pengujian *alpha* dilakukan pengujian *blackbox*, sedangkan pengujian *beta* menggunakan MOS untuk mendapatkan pendapat orang mengenai *user experience* ketika menggunakan aplikasi monitoring ini yang diukur dalam skor *rating*. *Range rating* skor antara 1-5, dimana skor 1 untuk sangat buruk dan skor 5 untuk sangat baik. Pengujian dilakukan dalam bentuk kuisioner, untuk total pertanyaan mengenai *user experience* ini diberikan tujuh buah pertanyaan. Sampel utama dari pengujian ini adalah seseorang yang mengerti tentang jaringan. Keluaran dari pengujian ini adalah skor *rating*. Rencana pengujian *alpha* aplikasi *web* dapat dilihat pada tabel 1

Tabel 1 Rencana pengujian *alpha* aplikasi *web*

Daftar Menu Uji	Detil Pengujian
Menu Input IP Controller	Memasukan alamat IP Controller
Menu Utama	Menampilkan menu topologi
	Menampilkan menu switches
	Menampilkan menu hosts
Menu Topologi	Menampilkan topologi jaringan
Menu Switches	Menampilkan informasi switch pada jaringan
	Menampilkan informasi flow pada jaringan
Menu Hosts	Menampilkan informasi host pada jaringan

3.2.2. Pengujian *Alpha*

Pengujian *alpha* pada sistem ini dilakukan dengan menggunakan pengujian *blackbox*. Pengujian *blackbox* dilakukan untuk memastikan aplikasi yang telah dibuat tidak mengalami *error* ketika digunakan dan di maksudkan untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak tersebut sesuai atau tidak dengan yang diharapkan. Berikut ini merupakan hasil dari pengujian

Tabel 2 Pengujian *blackbox* Menu Input IP *Controller*

Daftar Menu Uji	Prosedur Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Menu input IP *Controller*	Masuk ke aplikasi	Menampilkan halaman *input* IP *Controller*	Aplikasi menampilkan halaman *input* IP *Controller*	Sukses
	Mengisi alamat IP *Controller* pada *form* yang ada dengan nilai masukan benar	Menampilkan menu utama dan menampilkan jumlah *switch*	Aplikasi menampilkan menu utama dan menampilkan jumlah *switch*	Sukses

Tabel 3 Pengujian *blackbox* Menu Utama

Daftar Menu Uji	Prosedur Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Menu utama	Masuk ke aplikasi lalu memasukan alamat IP *Controller* dengan nilai masukan benar	Menampilkan halaman utama dengan menampilkan menu utama berupa menu topologi, *switches* dan *hosts*	Aplikasi menampilkan halaman utama dengan menampilkan menu utama berupa menu topologi, *switches* dan *hosts*	Sukses

Tabel 4 Pengujian *blackbox* Menu Topologi

Daftar Menu Uji	Prosedur Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Menu topologi	Masuk ke aplikasi lalu memasukan alamat IP *Controller* dengan nilai masukan benar dan memilih menu topologi	Menampilkan topologi jaringan yang dimonitoring dalam bentuk gambar	Aplikasi menampilkan topologi jaringan yang dimonitoring dalam bentuk gambar	Sukses

Tabel 5 Pengujian *blackbox* Menu *Switches*

Daftar Menu Uji	Prosedur Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Menu *switches*	Masuk ke aplikasi lalu memasukan alamat IP *Controller* dengan nilai masukan benar dan memilih menu *switches*	Menampilkan jumlah keseluruhan *switch* dalam jaringan	Aplikasi menampilkan jumlah keseluruhan *switch* dalam jaringan	Sukses
	Memilih DPID *switch* yang terdapat dalam daftar	Menampilkan informasi *switch* tertentu dengan DPID yang telah dipilih	Aplikasi menampilkan informasi *switch* tertentu dengan	Sukses

			DPID yang telah dipilih	
	Memilih menu *flows count* pada informasi *switch* yang telah ada	Menampilkan informasi *flows* pada *switch* tertentu dengan DPID yang telah dipilih	Aplikasi menampilkan informasi *flows* pada *switch* tertentu dengan DPID yang telah dipilih	Sukses

Tabel 6 Pengujian *blackbox* Menu *Hosts*

Daftar Menu Uji	Prosedur Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Menu *hosts*	Masuk ke aplikasi lalu memasukan alamat IP *Controller* dengan nilai masukan benar dan memilih menu *hosts*	Menampilkan informasi mengenai *host* yang ada dalam jaringan	Aplikasi menampilkan informasi mengenai *host* yang ada dalam jaringan	Sukses

3.2.3. Pengujian *Beta*

Pengujian *beta* yang dilakukan pada sistem ini merupakan pengujian MOS atau *Mean Opinion Score*. MOS merupakan suatu tes yang dilakukan untuk mendapatkan nilai rata-rata *rating* mengenai suatu sistem yang telah dibuat. Pengujian MOS dilakukan dengan cara memberikan kuisioner kepada responden mengenai sistem yang telah dibuat dan menilai sendiri tentang sistem tersebut, dalam hal ini pengujian yang dilakukan merupakan *user experience* ketika menggunakan aplikasi monitoring SDN. Pengujian ini dilakukan pada 30 responden. Berikut merupakan rekapitulasi hasil pengujian yang telah diperoleh.

Tabel 7 Hasil pengujian *beta*

Total soal	5
Skor terendah-tertinggi soal	1-5
Skor terendah-tertinggi responden	1-5
Total skor	601
Skor *rating*	4,01
Pertanyaan terlemah	Pertanyaan no. 2 (skor: 111)
Pertanyaan terkuat	Pertanyaan no. 5 (skor: 127)

Berdasarkan hasil pengujian yang telah dilakukan pada Tabel 7, didapatkan total skor 601 dari 750 dan skor *rating* sebesar 4,01 dari 5. Selain itu, didapatkan pula pertanyaan terlemah dan pertanyaan terkuat dengan skor masing-masing 111 dan 127. Pertanyaan terlemah yaitu tentang tampilan visualisasi topologi jaringan yang dimonitoring dengan skor sebesar 111 dan skor *rating* 3,70. Hal ini menunjukan bahwa para responden kurang menyukai tampilan visualisasi topologi yang ada. Sedangkan untuk pertanyaan terkuat yaitu tentang kecepatan aplikasi ketika menampilkan informasi mengenai jaringan yang dimonitoring yaitu dengan skor sebesar 127 dan skor *rating* 4,23. Hal ini menunjukan bahwa para responden menilai aplikasi monitoring sudah cepat dalam menampilkan informasi.

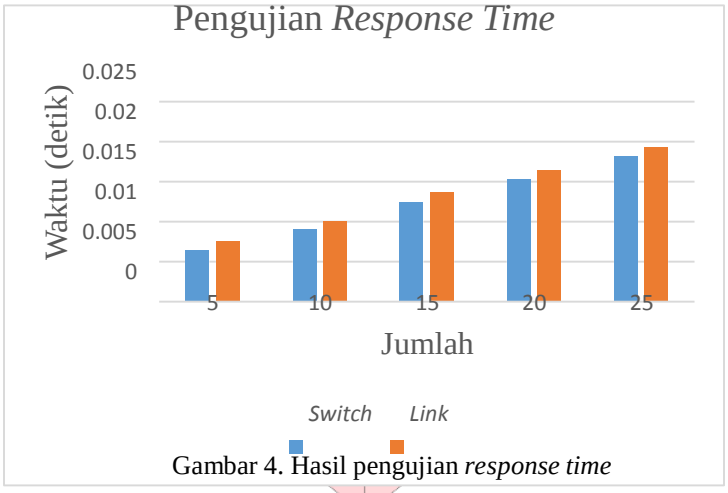
3.3. Pengujian *Controller*

3.3.1. Rencana Pengujian

Pengujian yang dilakukan pada *controller* merupakan pengujian *response time*. *Response time* merupakan waktu yang dibutuhkan untuk melakukan permintaan terhadap data yang diinginkan mulai dari pengguna melakukan *request* sampai pengguna mendapatkan data yang diinginkan. Pada pengujian response time ini, digunakan cURL sebagai alat pengukur waktu tersebut.

3.3.2. Pengujian

Pengujian yang dilakukan adalah pengujian terhadap jumlah *switch* dan *link* yang ada pada jaringan yang dimonitoring. Jumlah *switch* yang digunakan pada pengujian ini yaitu 5, 10, 15, 20 dan 25. Untuk jumlah *link* yang digunakan yaitu 5, 10, 15, 20 dan 25. Berikut ini merupakan hasil pengujian yang telah dilakukan.



Gambar 4. Hasil pengujian *response time*

Pada gambar 4 menunjukan hasil yang didapatkan dari pengujian yang dilakukan terhadap *response time*. Pada jumlah 5 *switch* didapatkan waktu selama 0,0064 detik, 10 *switch* selama 0,0091 detik, 15 *switch* selama 0,0124 detik, 20 *switch* selama 0,0153 detik dan 25 *switch* selama 0,0182 detik. Hal ini menunjukan bahwa semakin banyak *switch* yang digunakan dan dilakukan monitoring, maka waktu yang dibutuhkan untuk *response time* akan semakin bertambah juga. Untuk *link*, waktu yang diperlukan untuk jumlah 5 *link* adalah selama 0,0076 detik, 10 *link* selama 0,0101 detik, 15 *link* selama 0,0137 detik, 20 *link* selama 0,0164 detik dan 25 *link* selama 0,0193 detik. Dari percobaan tersebut dapat dilihat bahwa untuk percobaan *switch* dan *link* ketika jumlahnya ditambah, maka waktu yang dibutuhkan juga semakin bertambah. Untuk rata-rata waktu keseluruhan, percobaan *switch* memerlukan waktu selama 0,0123 detik dan percobaan *link* memerlukan waktu selama 0,0134 detik. Hal tersebut menunjukan bahwa ketika aplikasi melakukan monitoring yang menitikberatkan pada jumlah *switch*, maka aplikasi dapat lebih cepat menampilkan informasi yang diperlukan dibandingkan dengan ketika percobaan menitikberatkan pada jumlah *link*

4. Kesimpulan dan Saran

1. Kesimpulan

1. Berdasarkan pengujian *alpha* yaitu berupa pengujian *blackbox*, aplikasi monitoring dapat melakukan monitoring terhadap jaringan yang mengimplementasikan SDN didalamnya dengan baik dan benar sesuai yang diinginkan dengan ketepatan 100%.
2. Berdasarkan pengujian *beta* yaitu berupa pengujian MOS, skor *rating* yang didapatkan secara keseluruhan sebesar 4,01 dari skala 1-5 dan skor total sebesar 601 dari 750. Hal ini menunjukan bahwa aplikasi monitoring ini sudah dapat dengan mudah digunakan dan dipahami oleh penggunanya.
3. Berdasarkan pengujian *controller* yaitu pengujian *response time*, didapatkan waktu rata-rata yang dibutuhkan untuk menampilkan informasi sesuai yang diinginkan yaitu untuk percobaan pada jumlah *switch* adalah selama 0,0123 detik dan percobaan pada jumlah *link* adalah selama 0,0134 detik. Dengan selisih waktu rata-rata antara keduanya yaitu sebesar 0,0019 detik

2. Saran

1. Untuk pengembangan sistem monitoring pada SDN dapat dilakukan dengan melakukan penambahan parameter yang dimonitoring.
2. Pengembangan selanjutnya dapat dilakukan pada visualisasi topologi. Visualisasi topologi dapat dibuat lebih rinci ketika menampilkan topologi jaringan yang dimonitoring.
3. Desain dan tampilan aplikasi *web* dapat dibuat lebih menarik dan efisien agar dapat memudahkan penggunanya.

5. Daftar Pustaka

[1] B. Lantz, B. Heller and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," *Hotnes'10 ACM*, pp. 1-6, 2010.

[2] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh and J. v. d. Merwe, "The Case for Separating Routing from Routers," *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pp. 5-12, 2004.

[3] Open Networking Foundation, "Software-Defined Networking (SDN) Definition," [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition. [Accessed 12 May 2016].

[4] Open Networking Foundation, OpenFlow Switch Specification: Version 1.0.0 (Wire Protocol 0x01), 2009.

[5] Open Networking Foundation, Software Defined Networking: The New Norm for Networks, 2012.

[6] Ryu SDN Framework Community, "Ryu SDN Framework," [Online]. Available: https://osrg.github.io/ryu/. [Accessed 13 May 2016].

[7] V. S., R. P. Rustagi and K. N. B. Murthy, "Network Management and Performance Monitoring using Software Defined Networks," *Advanced Computing and Communications (ADCOM)*, pp. 29-31, 2014.

[8] W. Zhou, L. Li, M. Luo and W. Chou, "REST API Design Patterns for SDN Northbound API," *International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 358-365, 2014.