PERANCANGAN DAN ANALISIS JARINGAN VIRTUAL BERBASIS SOFTWARE-DEFINE NETWORKING (SDN)

DESIGN AND ANALYSIS OF VIRTUAL NETWORK BASED ON SOFTWARE-DEFINE NETWORKING (SDN)

Desianto Abdillah[1], Yuliant Sibaroni[2], Izzatul Ummah[3]

[1]Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[2] Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[3] Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[1]dei.jeg@gmail.com, [2]yuliant2000@yahoo.com, [3]izzatul.ummah@gmail.com

Abstrak

Pada saat ini perkembangan teknologi sangatlah pesat, tidak terkecuali pada jaringan komputer. Pada perkembangan teknologi ini muncul lah ide baru atau konsep baru yaitu *Software-Define Networking* (SDN). *Software-Define Networking* (SDN) adalah sebuah konsep pendekatan baru untuk mendesain, membangun dan mengelola jaringan komputer dengan memisahkan *control plane* dan *data plane*. Konsep utama pada SDN adalah sentralisasi jaringan dengan semua pengaturan berada pada *control plane*. Dalam SDN terdapat protokol yang paling menonjol yaitu OpenFlow. OpenFlow adalah sebuah protokol atau standar komunikasi antarmuka yang berada antara *control* dan *forwarding layer*. Pada tugas akhir ini akan disimulasikan jaringan SDN pada jaringan virtual. Simulasi jaringan virtual SDN ini menggunakan sebuah *tool* atau aplikasi yaitu Mininet. Mininet merupakan aplikasi yang berbasis *light-weight virtualization* yang dapat menciptakan jaringan virtual yang realistik, menjalankan real kernel, switch dan kode aplikasi. Hasil penelitian menunjukkan konsep jaringan SDN berjalan, mengukur kinerja dari jaringan SDN seperti *delay, jitter* dan *throughput* dengan beberapa skenario topologi yaitu 2 switch, 4 switch , 8 switch dan 16 switch.
Kata kunci : SDN , OpenFlow, Mininet, *delay, jitter, throughput*

Abstract

In this day, technological development is very rapid, not least on computer network. In these technological development has emerges a new idea or a concept that was Software-Define Networking (SDN). Software-Define Networking (SDN) is a new approach concept of design, build and manage computer network by seperating the control plane and data plane. The main concept of Software-Define Networking (SDN) is centralized network waith all regulation are in controll plane. SDN have a main protocol, that protocol is OpenFlow. OpenFlow is a standard communication interface define between controll and forwarding layer. In this final project will simulated SDN network on the virtual network. Simulation of virtual network SDN using a tool, that was Mininet. Mininet is tool based on ligh-weight virtualization application which could create a realistic virtual network running real kernel, switch and application code. This research is to understood how SDN network run, measuring performance of SDN network such as delay, jitter and throughput with a few topology scenarios like 2 switches, 4 switches, 8 switches and 16 switches.
Keywords: SDN, OpenFlow, Mininet, delay, jitter and throughput

1. Pendahuluan

Pada saat ini perkembangan teknologi informasi berkembang sangat pesat, tidak terkecuali pada jaringan komputer. Pada perkembangan teknologi ini muncul lah ide baru atau konsep baru yaitu *Software-Define Networking* (SDN). *Software-Define Networking* (SDN) adalah sebuah konsep pendekatan baru untuk mendesain, membangun dan mengelola jaringan komputer dengan memisahkan *control plane* dan *data plane*[4]. Konsep utama pada *Software-Define Networking* (SDN) adalah sentralisasi jaringan dengan semua pengaturan berada pada *control plane.* Konsep SDN ini sangat memudahkan operator atau administrator jaringan dalam mengelola jaringannya. SDN juga mampu memberikan solusi untuk permasalahan-permasalahan jaringan yang sekarang seperti sulitnya mengintegrasikan teknologi baru karena alasan perbedaan perangkat atau platform, kinerja yang buruk karena ada beberapa operasi yang berlebihan pada *protocl layer* dan sulitnya menyediakan layanan-layanan baru[10]. Konsep dari SDN sendiri dapat mempermudah dan mempercepat inovasi pada jaringan sehingga diharapkan muncul ide-ide baru yang lebih baik dan dapat di implementasikan.

Pada jaringan konvensional (non SDN) , perangkat lunak jaringan (firmware) dan router selama ini dibawah kendali perusahaan-perusahaan yang memproduksi perangkat tersebut. Namun dengan menggunakan SDN, membuat firware bisa diswitch dengan diakses dari jarak jauh, sehingga perangkat lunak pengguna bisa menggunan protokol terbuka seperti OpenFlow[14]. OpenFlow dapat

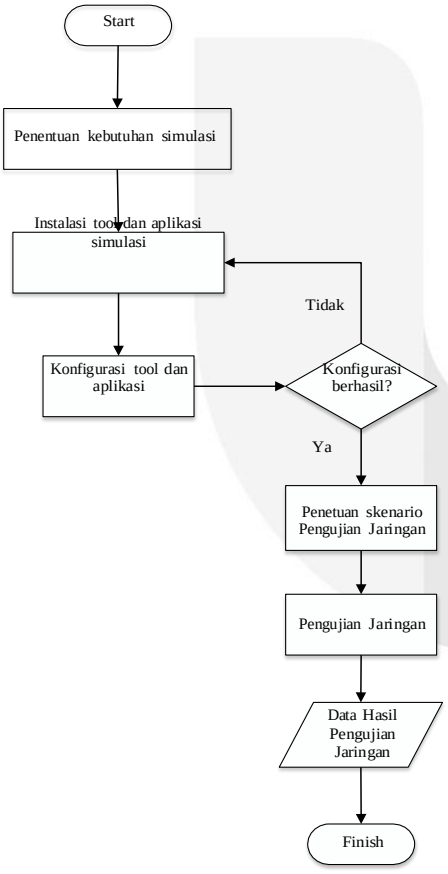
mengakses dan memanipulasi secara langsung *forwardign plane*/*data plane* dari perangkat-perangkat jaringan seperti Switch dan router, baik secara fisik maupun virtual[15].

Pada tugas akhir ini akan simulasikan jaringan virtual SDN berjalan pada paltform *Proxmox Virtual Environment* (PVE). *Proxmox Virtual Environment* (PVE) adalah sebuah *open-source environment* untuk virtualisasi server yang berbasis Linux distribusi Debian, mempunyai web *console* dan command-line tools. Dalam jaringan SDN diperlukan *controller*. *Controller* yang digunakan adalah POX, POX merupakan SDN *controller* yang *berbasis* Python dan dapat dijalankan diluar Mininet. Mininet merupakan aplikasi yang berbasis *light-weight virtualization* yang dapat menciptakan jaringan virtual yang realistik.

Melalui penelitian ini akan ukur kinerja jaringan dengan beberapa skenario rancangan topologi. Sehingga akan diketahui representasi jaringan SDN terhadap jumlah node dalam jaringan..

2. Metodologi

Secara umum, langkah - langkah dalam menyimulasikan jaringan SDN sebagai berikut :



Gambar 2-1: Gambaran Umum Sistem

2.1 Penentuan kebutuhan simulasi

Sesuai dengan konsep utama SDN, yaitu memisahkan antara *control plane* dan *data plane*, maka untuk *controller* SDN yang dalam hal ini adalah POX dipisah dengan Mininet yang akan berperan dalam menjalankan *data plane*.

2.2 Instalasi tool dan aplikasi simulasi

Instal Mininet dengan versi 2.2.1., Setelah itu instal paket protokol OpenFlow dengan versi 1.0.0., dan kemudian Install *controller* POX dengan versi/branch *dart* (0.3.0). Instalasi Mininet dan OpenFlow dilakukan pada server Mininet. Sedangkan *controller* POX diinstall pada server controllerSDN.

2.3 Konfigurasi tool dan aplikasi

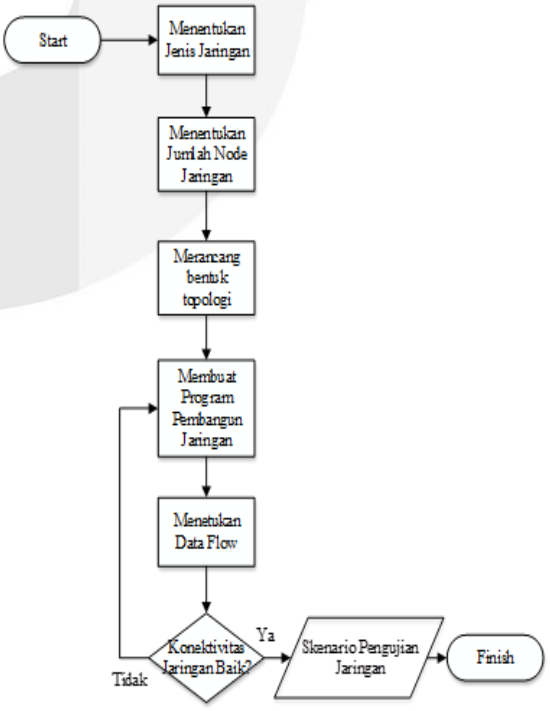
Konfigurasikan server Mininet agar dapat terhubung dengan server controllerSDN. Indeks konfigurasi sudah berjalan dengan baik, yaitu saat server Mininet menjalankan program/suatu topologi jaringan, pada server seharusnya terdeteksi MAC Address dari switch-switch yang dijalankan oleh server Mininet

2.4 Konfigurasi berhasil?

Seperti yang sudah dijelaskan pada diatas, pada server controllerSDN menginvoke controller POX. Sedangkan pada pihak server Mininet gunakan perintah *remote controller* yang artinya Mininet mengaktifkan *controller* namun bukan local *controller* Mininet. Apabila konfigurasi konvigurasi berhasil identitas-identias perangkat-perangkat jaringan seperti MAC Address terdeteksi oleh *controller* POX.

2.5 Penentuan skenario pengujian jaringan

Berikut merupakan tahapan menentukan skenario pengujian jaringan :



Gambar 2-5: *Flowchart* pengujian jaringan

2.5.1 Menentukan jenis jaringan

Topologi jaringan yang gunakan dalam simulasi jaringan ini adalah topologi *full mesh.* Topologi full mesh adalah topologi jaringan yang semua perangkat jaringan dalam hal ini node-node jaringan saling terhubung. Node dalam jaringan berupa komputer/host, switch dan *controller*. Dengan yang saling terhubung adalah antar switch dan *controller*, sedangkan host cukup terhubung dengan switch masing-masing. Topologi *full mesh* cukup kompleks untuk diukur kinerjanya, kemudian kapasitas *channel* komunikasi terjamin, karena memiliki hubungan yang lebih. Topologi *full mesh* relatif lebih mudah dalam *troubelshooting* dan sharing file lebih cepat karena memiliki jalur masing-masing

2.5.2 Menetukan jumlah node

Jumlah node yang digunakan adalah satu *controller* dengan variasi jumlah switch dan host. Variasi jumlah host semakin bertambah. Jumlah node disesuaikan dengan topologi *full mesh* dan yang terpenting dapat digunakan untuk mencari data yang dibutuhkan.

2.5.3 Menentukan bentuk topologi

Dalam perancangan bentuk topologi tools yang digunakan adalah *Visual Network Description* (VND). *Visual Network Description* (VND) adalah sebuah grafis antarmuka *user* untuk pembuatan skenario jaringan SDN. VND mendukung Mininet dan *controller* SDN bentuk topologi disesuaikan dengan jumlah node yang semakin berkembang.

2.5.4 Membuat program pembangun jaringan

Program yang dibuat dalam bahasa Python sesuai *controller* SDN dan program tersebut mendukung Mininet. Program dibuat sesuai rancangan topologi yang sudah dibuat. Program tersebut juga berisi konfigurasi routing statis. Routing statis merupakan metode routing yang penjalurannya dilakukan secara manual dengan mengisi setiap entri dalam *forwarding table*.

2.5.5 Menentukan data flow

Data flow merupakan aturan didalam jaringan yang menentukan pergerakan paket data yang akan dikirim. Data flow harus sesuai dengan routing statis yang dibuat, agar saat pengiriman paket data tidak mengalami kesalahan serta konektivitas jaringan tetap terjaga.

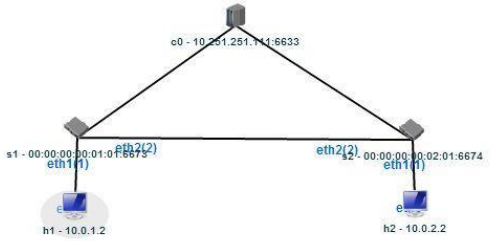
2.5.6 Konektivitas jaringan baik

Semua host di cek konektivitasnya agar dapat terbentuk komunikasi. Untuk mengecek konektivitas jaringan digunakan perintah *pingall*. Apabila terdapat sebagian host yang tidak dapat dijangkau/dihubungi, cek kembali program pembangun jaringannya, routing statis dan data flownya.

2.5.7 Skenario pengujian jaringan

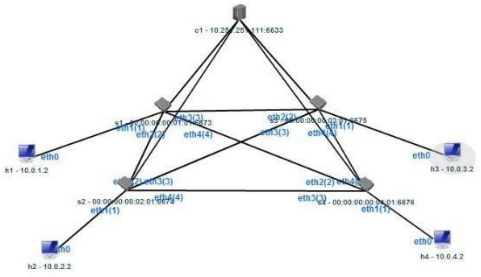
Didapatkan beberapa skenario untuk menguji jaringan. Skenario yang dibuat adalah Topologi 2 switch, Topologi 4 switch, topologi 8 switch dan topologi 16 switch dengan masing-masing maksimal bandwidth 1 GB/sec. Pemilihan jumlah switch yang bervariasi dan semakin banyak agar dapat diketahui pola dari tingkah laku jaringan SDN dan tingkat penanganan SDN terhadap jaringan yang semakin lama semakin besar dan kompleks. Berikut hasil skenario jaringan :

a. Topologi dua switch

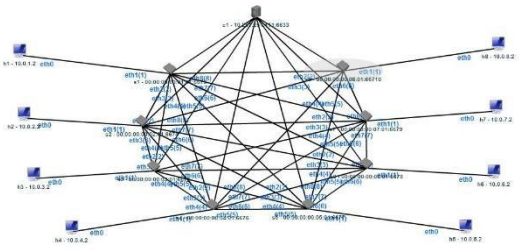


Gambar 2.5.7.a Topologi 2 switch

b. Topologi empat switch

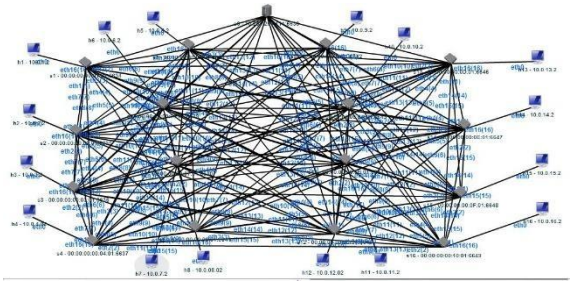


c. Topologi delapan switch



Gambar 2.5.7.c Topologi 8 switch

d. Topologi delapan switch



Gambar 2.5.7.d Topologi 16 switch

2.6 Pengujian jaringan

Konsep komunikasi jaringan SDN sangat digunakan dalam proses pengujian/pengukuran jaringan. Untuk menjalankan skenario/proses pengujian kinerja jaringan, digunakan tools/perintah *ping dan iperf*. Perintah ping digunakan untuk mencari nilai *delay* dan *jitter*, sedangkan perintah iperf digunakan untuk mengukur throughput bandwidth *throuhput* yang diukur melalui port TCP dan port UDP.

2.7 Hasil pengukuran

Hasil pengukuran adalah *delay, jitter* dan *throughput*. Ketiga data tersebut disajikan dalam bentuk tabel perskenario, kemudian analisis data tersebut dengan dibandingkan nilai *delay, jitter* dan *throughput* antar skenario.

3. Hasil dan Analisis

Hasil yang didapat disajikan dalam tabel dan juga dalam grafik agar lebih mudah dalam menganalisisnya. Hasil tersebut berupa data nilai *delay, jitter* dan *throughput.*

3.1 Data hasil pengujian jaringan

Tabel 3.1 Data semua skenario

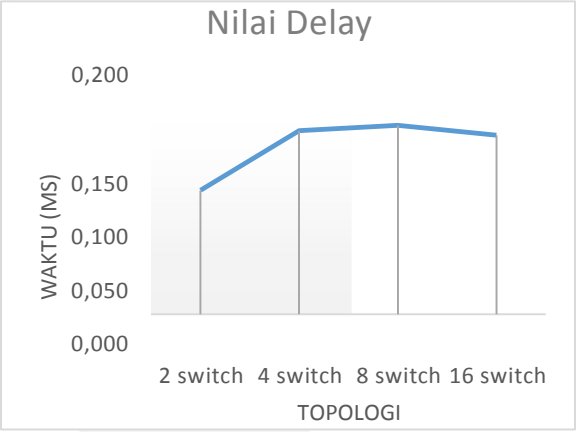
Skenario topologi	Delay (ms)	Jitter (ms)	Throuhput (Bits/sec)	
			TCP	UDP
2 switch	0,116	0,015	9,371	9,528
4 switch	0,171	0,111	9,339	9,534
8 switch	0,177	0,098	9,339	9,537
16 switch	0,167	0,154	9,347	9,537

Tabel diatas menunjukkan data semua nilay *delay, jitter* dan *throughput* dalam penelitian tugas akhir ini. Untuk nilai *delay* skenario topologi dua switch mempunyai waktu delay yang paling kecil, sedangkan skenario topologi delapan switch mempunyai waktu yang paling lama. Hal ini menunjukkan bahwa topologi dua switch membutuhkan waktu yang sedikit untuk bisa saling berkomunikasi dan topologi delapan switch membutuhkan waktu yang lama.

Nilai *jitter* sedikit berbeda dengan *delay*, pada topologi 16 switch mempunyai waktu *jitter* yang paling lama. Hal ini menunjukkan pada topologi 16 switch mempunyai waktu *delay* sangat bervariasi.

Untuk nilai throughput baik port TCP dan UDP antar topologi cenderung stabil dengan kisaran 9,371 – 9,537 bits/sec. Untuk throughput pada port UDP mempunyai nilai yang lebih tinggi dari nilai TCP. Hal ini menunjukkan bahwa untuk berapapun jumlah nodenya nilai throughput akan sama dan port UDP lebih cepat dari port TCP.

3.1.1 *Delay*

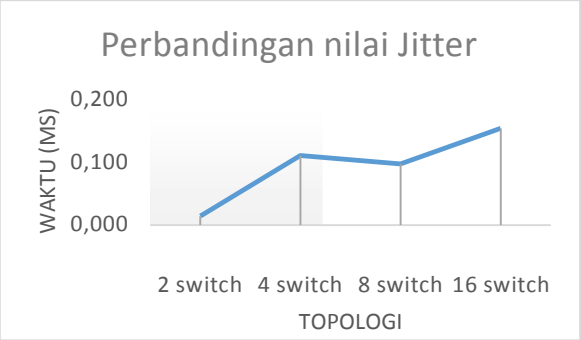


Gambar 3.1.1 Nilai *Delay*

Dapat dilihat dari grafik di atas bahwa pola keseluruhan nilai *delay* cenderung tetap, namun ada beberapa yang mengalami kenaikan dan penurunan. Kenaikan terjadi pada dua switch dengan empat switch dan empat switch dengan delapan switch. Hal ini terjadi karena pada saat penentuan data flow mengalami penambahan deskripsi seperti perangkat jaringan yang dituju dan port-port yang tersedia pada suatu switch pada jaringan tersebut. Sehingga menyebabkan waktu pengecekan alamat fisik dari perangkat jaringan dan port-port yang tersedia menjadi bertambah. Sedangkan penurunan nilai *delay* terjadi pada delapan switch dengan 16 switch, namun empat switch dengan 16 switch cenderung tetap. Hal ini terjadi karena waktu yang dibutuhkan pada saat pengecekan alamat fisik dari perangkat dan port-port yang tersedia cenderung sama karena sudah dideskripsikan pada data flow yang sebelumnya. Berdasarkan standard rekomendasi **ITU-T** yaitu 100

ms, nilai *delay* yang didapat masih dalam kategori baik karena memenuhi standard rekomendasi. Bahkan nilai *delay* paling tinggi yaitu pada topologi delapan switch yaitu 0,177 ms masih dibawah 1ms.

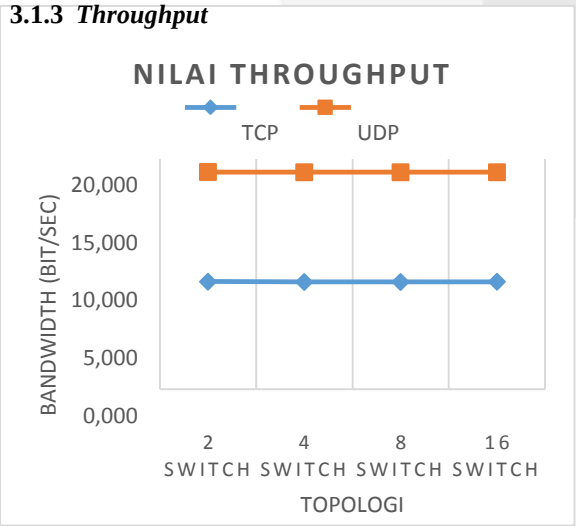
3.1.2 *Jitter*



Gambar 3.1.2 Nilai *Jitter*

Pada grafik diatas menunjukkan bahwa nilai *jitter* cenderung meningkat, dapat dilihat antara dua switch dengan empat switch mengalami peningkatan dan antara delapan switch dengan 16 switch juga mengalami. Hal ini menunjukkan kenaikan jumlah switch berbanding lurus dengan kenaikan nilai *jitter*. Namun ada suatu penurunan yang terjadi pada empat switch dengan delapan switch, yang terjadi karena pengaruh dari waktu komunikasi (nilai *delay*) itu sendiri. Sedangkan ada topologi 16 switch mempunyai nilai *jitter* paling tinggi, hal itu menunjukkan topologi tersebut waktu *delay* yang diperlukan untuk tiap-tiap host sangat bervariatif. Namun nilai-nilai *jitter* tersebut masih memenuhi standard rekomendasi **ITU – T** yaitu dibawah 50 ms, bahkan rata-rata dari nilai *jitter* diatas masih dibawah 1 ms.

3.1.3 *Throughput*



Gambar 3.1.3 Nilai *Throughput*

Dari grafik throughput diatas bisa diartikan, nilai throughput untuk port TCP cenderung stabil walau pada topologi dua switch lebih tinggi nilai throughputnya dari semua topologi dengan nilai 9,371. Hal ini dikarenakan pada topologi dua switch hanya terdapat dua host dengan salah satunya menjadi client dan salah satunya menjadi server yang menjadikan transfer data lebih cepat. Sedangkan port UDP nilai throuhputnya pun cenderung stabil dengan rentang nilainya berada pada 9,5. Namun dari semua topologi port UDP lah yang lebih cepat dengan rata-rata nilai throuhputnya 9,534 sedangkan rata-rata nilai throuhput dari port TCP adalah 9,349.

4. Kesimpulan

Setelah dilakukan beberapa pengujian, adapun berikut hal-hal yang dapat disimpulkan dari penelitian tugas akhir ini:

a. Topologi jaringan virtual berbasis SDN dapat dirancang dan dapat dijalankan dengan maksimal topologi 16 switch menggunakan sumber daya dengan spesifikasi RAM 32 GB dan CPU 16 Core.
b. Kinerja jaringan virtual berbasis SDN memiliki nilai delay yang tetap meskipun ada beberapa pengingkatan dan penurunan dengan jumlah switch semakin meningkat, nilai *jitter* cenderung meningkat sesuai dengan jumlah switch, namun niali *delay* dan *jitter* masih memenuhi standrad rekomendasi **ITU-T** dan untuk nilai throughput cenderung stabil baik untuk port TCP maupun UDP dengan port UDP lah yang lebih cepat.
c. Bentuk topologi dengan jumlah node yang telah dirancang dapat merepresentasikan bentuk topologi lainya dengan node yang lebih banyak.

5. Saran

Pada pengembangan selanjutnya untuk memperbaiki tugas akhir ini dapat dilakukan :

1. Menambah jumlah node-node jaringan atau menambah skenario pengujian jaringan.
2. Membandingan lebih dari satu jenis topologi, misalkan membandingkan topologi *tree* atau *fat-tree* dengan topologi *full mesh*.
3. Pengukuran atau pengujian jaringan lebih dispesifikasikan kedalam bidang-bidang jaringan, contoh (Quality of Service) QoS, *monitoring*, *traffic engineering* dll.

Daftar Pustaka:

[1] Kaur, Karamjeet., Singh, Japinder., dan Singh Ghumman, Navtej., 2014, "International Conference on Communication, Computing & Systems (ICCCS–2014) : *Mininet as Software Defined Networking Testing Platform*".

[2] Lantz, Bob., Heller, Brandon., McKeown, nick., 2010, *A Network in a Laptop : Rapid Prototyping for Software-Define Networks.*.
[3] Network Virtualization and Data Center Network, Assigment 3: Software-Define Network, 2013, System@ETHZurich.
[4] Mulyana, Eueng.2015.Buku Komunitas SDN-RG. Bandung : GitBook.
[5] Open Networkinh Foundation : Software-Defined Networking (SDN) Definition, tersedia: https://www.opennetworking.org/sdn-resources/sdn-definition (3 November 2014).
[6] Definition of Software-Define Networking (SDN), tersedia : https://www.sdncentral.com/resources/sdn/what-the-definition-of-software-defined-networking-sdn/ (21 November 2014)
[7] Software-Define Networking (SDN), tersedia : http://en.wikipedia.org/wiki/Software-defined_networking (21 November 2014).
[8] Ardiansyah (Ardi Sragen) : Jaringan OpenFlow, tersedia: http://ardisragen.net/jaringan-openflow.html (3 November 2014).
[9] Arora, Dushyat., Botero-Perez, Diego., *Live migration of an Entire Software-Defined Network.*
[10] Mininet, tersedia : http://mininet.org/ (2 November 2014)
[11] Feamster, Nick. Software Define Networking : *Active Network*
[12] Saputra, Ady. 2013. *Routing Statik dan Routing OSPF-like dengan openflow-mininet*. GitHub
[13] Samsono Hadi, Zen Samsono : *Performance & Monitoring Network*
[14] Adnantya, Fahry., Naning Hertiana, Sofia., Vidya Yovita, Leanna. 2015.Jurnal Ilmiah : *Simulasi dan analisis performansi protokol ruting EBGP pada SDN (Software Define Networking)*
[15] Singgih Wibowo.2009."Budi Daya Bawang".Penerbit Penebar Swadaya. Jakarta
[16] Sadewo, Bayu. SDN Sebagai Solusi Masa Depan Jaringan, tersedia : http://telsetnews.com/44549/sdn-sebagai-solusi-masa-depan-jaringan/ (5 Januari 2016)
[17] Open Network Foundation, OpenFlow, tersedia : https://www.opennetworking.org/sdn-resources/openflow (21 Agustus 2015)
[18] Open Networking Lab. POX Wiki, tersedia: https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-InstallingPOX (21 Agustus 2015)