IMPLEMENTASI PROTOKOL ROUTING OSPF PADA SOFTWARE DEFINED NETWORK BERBASIS ROUTEFLOW

IMPLEMENTATION OF OSPF ROUTING PROTOCOL FOR SOFTWARE DEFINED NETWORK BASE ON ROUTEFLOW

Ayu Irmawati[1], Indrarini Dyah Irawati, S.T., M.T.[2], Yuli Sun Hariyani, S.T., M.T.[3]

[123]Prodi D3 Teknik Telekomunikasi, Fakultas Ilmu Terapan, Universitas Telkom

[1]Ayuirmawati@students.telkomuniversity.ac.id [2]Indrarini@tass.telkomuniversity.ac.id

[3]yulisun@tass.telkomuniversity.ac.id

Abstrak

Internet merupakan salah satu teknologi sistem jaringan komputer yang sedang berkembang pesat saat ini. *Routing* merupakan suatu mekanisme konfigurasi yang sangat dibutuhkan dalam membangun sebuah jaringan internet, karena *routing* merupakan bagian utama dalam memberikan suatu performansi dalam jaringan. Performansi jaringan, penambahan konfigurasi yang semakin kompleks dan besar, bagian kontrol pun akan semakin rumit, tidak fleksibel dan sulit untuk diatur. *Software Defined Network* (SDN) adalah sebuah paradigma jaringan dimana *control plane* terpisah dengan *data plane*, sehingga mempermudah kita untuk melakukan konfigurasi di sisi *control plane*. Dengan adanya SDN diharapkan dapat menjalankan metode-metode yang terdapat pada jaringan konvensional seperti IP *forwarding* dan *routing*.

Dalam proyek akhir ini dilakukan penerapan layanan *routing Open Shortest Path First* (OSPF) berbasis *RouteFlow* pada jaringan *Software Defined Network*. Serta mengetahui jalur yang digunakan untuk mentransmisikan paket data dari pengirim ke penerima. Serta dilakukan skenario pemutusan dua jalur pada jaringan untuk membuktikan kinerja *routing* OSPF. Pembuktian dilakukan dengan melakukan emulasi dan implementasi jaringan, yang terdiri dari 4 *switch* yang saling terhubung dengan perangkat *control plane* sebagai pengendali sebuah jaringan.

Hasil pengujian performansi penerapan *routing* OSPF pada jaringan SDN berbasis *RouteFlow* menunjukkan bahwa nilai *convergece time* 4.2 detik untuk simulasi dan 4 detik untuk implementasi. Nilai QoS pada simulasi yaitu, 99.8 *Mbps* untuk *troughput*, 27.43 *ms* untuk *delay*, 0.008 *ms* untuk *jitter*, dan 0.0375% untuk *packet loss*. Nilai QoS pada implementasi yaitu, 99.98 *Mbps* untuk *troughput*, 35.7 *ms* untuk *delay*, 1.015 *ms* untuk *jitter*, dan 0.0956% untuk *packet loss*.

Kata Kunci: QoS, *RouteFlow*, SDN, OSPF

Abstrac

The Internet is one of the technology of computer network system which is rapidly growing. Routing is a mechanism which configuration is needed in building a network of internet. Routing, as a part which gives a network performance. The network performance, additional configuration of increasingly complex and large part of the network control would be more complicated, inflexible and difficult to manage. Software Defined Network (SDN) is a paradigm where the control plane of network that separated from the data plane, so that

make us easier to do configuration on control plane. SDN is expected to run methods contained in conventional networks such as IP forwarding and routing.

In this final project, applying routing service using Open Shortest Path First (OSPF) based on RouteFlow for Software Defined Network. Knowing the path used for transmitting packets from sender to receiver. On the scenario done two termination links on a network to prove routing OSPF performance. Authentication is done by doing emulation and implementation the network, consisting of 4 switches are connected with the control plane.

Application performance testing result routing OSPF for SDN base on RouteFlow shows that the value of convergence time are 4.2 seconds for simulation and 4 seconds for implementation. The value QoS for simulation that are, 99.8 Mbps for troughput, 27.43 ms for delay, 0.008 ms for jitter, and 0.0375 % for packet loss. The value QoS for implementation that are, 99.98 Mbps for troughput, 35.7 ms for delay, 1.015 ms for jitter, and 0.0956% for packet loss.

Keywords : QoS, RouteFlow, SDN, OSPF

1. Pendahuluan

1.1 Latar Belakang

Kondisi jaringan yang mulai jenuh menjadi alasan mulai banyaknya penelitian dan percobaan *platform Software Defined Network* (SDN) yang merupakan salah satu evolusi teknologi jaringan sesuai dengan tuntutan yang berkembang. Dibandingkan dengan jaringan konvensional, *Software Defined Networking* (SDN) memberikan kemudahan kepada pengguna dalam mengembangkan aplikasi pengontrol jaringan dengan memisahkan fungsi *data plane* dari *control plane* [2].

Pemisahan *data plane* dan *control-plane* pada perangkat jaringan komputer seperti *Router* dan *Switch* memungkinkan untuk memprogram perangkat tersebut sesuai dengan yang diinginkan secara terpusat. Pemisahan inilah yang mendasari terbentuknya paradigma baru dalam jaringan komputer yang disebut *Software Defined Networking* (SDN) (US: Open Networking Foundation. 2013). *Platform controller* menyediakan *Application Programming Interfaces* (APIs) sehingga memudahkan dalam mengimplementasikan fitur dan layanan dalam jaringan komputer. Pada *Software Defined Networking* (SDN), *controller* terpusat mengkonfigurasi *forwarding tabel* (*flow-table*) *Switch* yang bertanggung jawab untuk meneruskan aliran paket komunikasi [8].

Berdasarkan dari penelitian sebelumnya tentang perbandingan kinerja penjaluran paket dengan metode *path calculating* algoritma dijkstra (Brayan Anggita Linuwih, 2016) [6], dan implementasi *routing static* pada jaringan SDN menggunakan *Ryu Controller* dan OpenvSwitch (Nuruzzamanirridha., 2016) [7]. Penulis ingin mengembangkan dengan menerapkan *routing* OSPF dan penambahan skenario yang belum terimplementasi pada penelitian tersebut.

Kehebatan teknologi SDN dalam jaringan komputer dianggap menarik oleh penulis, sehingga tertarik untuk mengimplementasikan *routing OSPF* berbasis *RouteFlow*. Dalam implementasi, diperlukan 4 buah *switch* yang telah *support* teknologi SDN dan 4 *host* sebagai *source* dan *destination* dalam percobaan pengiriman paket *Internet Control Message Protocol* (ICMP).

1.2 Tujuan

Adapun tujuan Proyek Akhir dari Implementasi Protokol *Routing* OSPF pada Jaringan SDN berbasis *RouteFlow* sebagai berikut:

a. Dapat melakukan simulasi *routing* OSPF berbasis *RouteFlow* pada jaringan *Software Defined Network.*
b. Dapat mengetahui *flow-table* pada jaringan.
c. Dapat mengimplementasi jaringan *Software Defined Network* yang dapat melakukan fungsi *routing* OSPF.
d. Mendapatkan nilai *convergence time* dan QoS (*throughput, delay, jitter* dan packet *loss*) pada jaringan.
e. Dapat menghasilkan modul pembelajaran *Software Defined Network*.

1.3 Rumusan Masalah

Berdasarkan deskripsi latar belakang, maka dapat dirumuskan beberapa masalah dalam Proyek Akhir ini yaitu:

a. Bagaimana cara menggunakan POX *controller* dan Mininet dalam emulasi jaringan berbasi *Software Defined Network* ?
b. Bagaimana cara membentuk topologi sesuai dengan skenario pada Mininet ?
c. Bagaimana cara menerapkan *routing* OSPF berbasis *RouteFlow* yang digunakan pada jaringan SDN ?
d. Bagaimana cara menggunakan *tool* iperf untuk mengukur QoS pada jaringan?
e. Bagaimana cara konfigurasi OpenvSwitch dan TP-Link agar dapat digunakan untuk implementasi jaringan *Software Defined Network*?

1.4 Metode Penelitian

Metedologi yang digunakan pada penulisan Proyek Akhir ini sebagai berikut:

a. Studi literatur : pencarian informasi yang terkait bersumber dari buku, media, jurnal dan diskusi yang bertujuan menunjang selesainya Proyek Akhir ini.
b. Perancangan dan Implementasi alat : melakukan perancangan dan pengimplementasian sistem sesuai dengan parameter yang diinginkan.
c. Analisa sistem : mengamati hasil dari sistem yang dikerjakan sesuai dengan skenario yang telah ditetapkan serta menyimpulkan masalah yang ada.
d. Penarikan kesimpulan : dari seluruh tahapan yang telah dilakukan diatas ditambah dengan masukan dari dosen pembimbing maka dapat diambil kesimpulan dari hasil yang telah dilakukan.

2. Dasar Teori

2.1 *Software Defined Network*

Software defined networking (SDN) adalah sebuah konsep pendekatan jaringan komputer dimana sistem pengontrol dari arus data dipisahkan dari perangkat kerasnya. Awal mula terciptanya teknologi *Software defined networking* dimulai tidak lama setelah Sun Microsystems merilis Java pada tahun 1995 [1] [3] [4], namun pada saat itu belum cukup membangunkan para periset untuk mengembangkan teknologi tersebut. Baru pada tahun 2008 *Software defined networking* ini dikembangkan di UC Berkeley and Stanford University. Dan kemudian teknologi tersebut mulai dipromosikan *oleh Open Networking Foundation* yang didirikan pada tahun 2011 untuk memperkenalkan teknologi SDN dan *OpenFlow*.Di dalam teknologi SDN memiliki dua karakteristik. Pertama,

SDN memisah antara *control plane* dari *data plane*. Kedua, SDN menggabungkan *control plane* setiap perangkat menjadi sebuah kontroler yang berbasiskan *programmeable software.* Sehinga sebuah kontroler tersebut dapat mengontrol banyak perangkat dalam sebuah *data plane*. Di dalam SDN sebuah jaringan tersentralisasi dalam sebuah kontroler yang berbasiskan *software* yang dapat memelihara jaringan secara keseluruhan, Sehingga dapat mempermudah dalam mendesain dan mengoperasikan jaringan karena hanya melalui sebuah *logical point*. Berikut adalah keunggulan *Software defined networking*:

a. Manajemen terpusat dan kontrol perangkat jaringan dari beberapa vendor.
b. Peningkatan otomatisasi dan manajemen dengan menggunakan API.
c. Inovasi cepat melalui kemampuan untuk memberikan kemampuan jaringan baru dan jasa tanpa perlu mengkonfigurasi perangkat individu atau menunggu penjual rilis.
d. *Programmability* oleh operator, perusahaan, vendor perangkat lunak independen, dan pengguna (bukan hanya produsen peralatan) menggunakan pemrograman umum lingkungan, yang memberikan semua pihak peluang baru untuk mendorong pendapatan dan diferensiasi.
e. Peningkatan kehandalan jaringan dan keamanan sebagai akibat dari pengelolaan jaringan terpusat dan manajemen otomatis dari perangkat jaringan, penegakan kebijakan yang seragam, dan meminimalisir kesalahan konfigurasi yang lebih sedikit.
f. Kemampuan kontrol jaringan akan lebih rinci dengan menerapkan dengan kebijakan di tingkat sesi, pengguna, perangkat, dan aplikasi secara komprehensif.
g. Penyajian antarmuka yang lebih baik sebagai aplikasi manajemen jaringan terpusat.

2.2 *RouteFlow*

RouteFlow terbentuk atas penggabungan proyek *OpenFlow* dengan *routing engine* Quagga [5]. Sistem ini terdiri dari *controller OpenFlow* (RFProxy), RFClient dan *Independent Server* (RFServer). Tujuan utama dibuatnya *RouteFlow* adalah menerapkan virtualisasi IP *routing* secara terpusat, dengan memisahkan fungsi *control-plane* dan *data-plane*. Penelitian ini memanfaatkan *RouteFlow* sebagai sistem yang berjalan pada *control-plane.*

Arsitektur *RouteFlow* terdiri dari :

1. RFServer adalah *standalone application* yang memegang kendali pusat kontrol jaringan. RFServer mengatur *Virtual Machine* (VMware) yang berjalan pada RFClient dan mengatur *logic process* (seperti *event processing*, VM mapping, *resource management*).
2. RFProxy adalah *controller* POX yang bertugas meneruskan kebijakan protokol (misalnya *update route*, konfigurasi datapath) dari RFServer ke *data-plane*.
3. RFClient adalah *daemon* pada VMware yang bertugas mendeteksi perubahan informasi *routing* dan memberitahukannya ke RFServer. Tugas tersebut dikomunikasikan dengan *routing engine* (Quagga).

2.3 Quagga

Quagga adalah suatu *routing engine* yang dapat menjalankan protokol *routing* konvensional seperti RIP, OSPF, BGP, dan lain-lain. Quagga yang tujuannya secara umum implementasi *routing* bersifat *open source* yang cocok untuk digunakan di SDN *environtment* [9]. Penerapan quagga dapat digunakan untuk Unix *platform* khususnya FreeBSD, Linux, Solaris, dan NetBSD. Arsitektur Quagga terdiri dari *core* daemon , zebra, yang

bertindak sebagai layer abstraksi untuk Unix kernel yang mendasari dan menyajikan Zserv API diatas unix atau TCP *stream* untuk Quagga *client*. Pada Zserv *client* ini biasanya menerapkan protokol *routing* dan menyampaikan *update routing* ke zebra daemon.

2.4 OpenWrt[12]

OpenWrt adalah sebuah sistem operasi *embedded* berbasis Linux kernel untuk perangkat *embedded* yang berfungsi sebagai perutean trafik jaringan. Komponen utama OpenWrt adalah Linux kernel, util-linux, uClibc, dan busybox. Semua komponen sudah dioptimalkan agar cocok dengan *wireless router* yang mempunyai kapasitas RAM dan media penyimpanan yang terbatas.

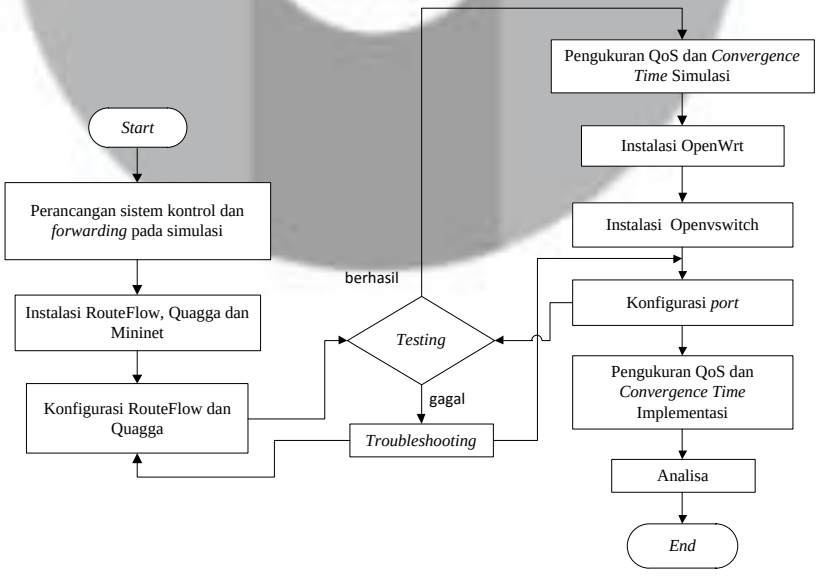
Konfigurasi OpenWrt dapat dilakukan menggunakan *command-line* atau secara GUI menggunakan interface Web. Selain itu, ada sekitar 3500 *packages*, salah satunya adalah Open vSwitch yang tersedia sebagai aplikasi tambahan dan dapat di install menggunakan *opkg packages management system*.

2.5 Protokol Routing OSPF (*Open Shortest Path First*)

Open Shortest Path First (OSPF) adalah protokol yang memanfaatkan algoritma Dijkstra dalam pemilihan *rute* terbaiknya. OSPF merupakan sebuah protokol perutean berjenis IGP (*interior gateway protocol*) yang hanya dapat bekerja dalam jaringan internal suatu organisasi atau perusahaan [10]. Jaringan internal maksudnya adalah jaringan di mana pengguna masih memiliki hak untuk menggunakan, mengatur, dan memodifikasinya, atau dengan kata lain, pengguna masih memiliki hak administrasi terhadap jaringan tersebut. Jika pengguna sudah tidak memiliki hak untuk menggunakan dan mengaturnya, maka jaringan tersebut dapat dikategorikan sebagai jaringan eksternal. Selain itu, OSPF juga merupakan protokol perutean yang berstandar terbuka. Maksudnya, protokol perutean ini bukan ciptaan dari vendor mana pun, sehingga siapapun dapat menggunakannya, perangkat mana pun dapat cocok dengannya, dan di mana pun protokol perutean ini dapat diimplementasikan.

3. Perancangan Sistem

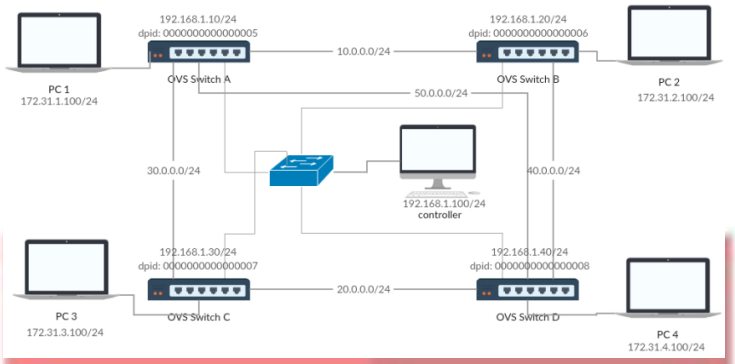
3.1 Metode Penelitian



Gambar 3.1 *Flowchart* Metode Penelitian

3.2 Desain Topologi Jaringan

Desain topologi jaringan pada Proyek Akhir ini menggunakan Laptop sebagai kontroler, 4 buah TP-Link sebagai *switch* arus data *forwarding*, dan 4 buah Laptop sebagai *host*. Topologi jaringan yang digunakan pada Proyek Akhir ini adalah seperti gambar dibawah ini



Gambar 3.2 Desain Topologi Jaringan

3.3 Skenario Pengujian

Terdapat tiga jenis pengujian pada Proyek Akhir ini, yaitu pengujian awal menggunakan PING dan *traceroute*, pengujian *convergence time* dan pengujian parameter QoS (*troughput*, *delay*, *jitter* dan *packet loss*). Adapun skenario pengujian sebagai berikut:

a. Mengukur QoS (*troughput*, *delay*, *jitter* dan *packet loss)* pada jaringan menggunakan protokol UDP pada simulasi dan implementasi dengan *background traffic* 100Mbps.
b. Pengukuran *Delay* dilakukan dengan pengiriman paket *ping* (64 *bytes*) dari h1 ke h4.
c. Melihat jalur yang dilalui paket data dari h1 ke h4 (atau sebaliknya) saat *link* tidak diputus dan saat diputus (*link* yang diputus *switch* A ke *switch* D dan *switch* D ke *switch* B).
d. Mencari nilai *convergence time* dan *troughput* menggunakan protokol TCP, jika *link switch* A ke *switch* D diputus.

4. Pengujian dan Analisa Sistem

4.1 Analisa Kinerja Jaringan Berdasarkan Pengujian

Berdasarkan hasil pengujian yang telah dilakukan pada Proyek Akhir ini, langkah terakhir dari bab IV adalah membandingkan nilai hasil pengujian dengan standarisasi QoS ITU-T G 1010 *End-user multimedia* QoS *Categories*. Berikut standar yang digunakan:

Tabel 4.1 Referensi QoS ITU-T G. 1010[11]

No.	Parameter Performansi	Data
1	*One way delay (latency)*	Preffered < 15 s; Acceptable < 60 s Nb: *Amount of Data* 10kB – 10MB
2	*Jitter*	NA
3	*Troughput*	NA
4	*Packet Loss*	0%

Pengujian berhasil dilakukan dengan mendapatkan semua nilai yang diinginkan. Berikut ini rangkuman seluruh hasil pengujian yang telah dilakukan.

Tabel 4.2 Perbandingan Hasil Pengujian

No.	Parameter	Hasil Pengukuran		ITU-T G. 1010
		Simulasi	Implementasi	
1.	*Troughput* - Saat *link* tidak diputus (UDP) - Saat *link* diputus (TCP)	 99.8 *Mbps* 473.78 *Mbps*	 99.985 *Mbps* 392.48 *Mbps*	 NA NA
2.	*Delay*	27.43 *ms*	35.7 *ms*	-
3	*Jitter* (UDP)	0.008 *ms*	1.015 *ms*	NA
4	*Packet Loss* (UDP)	0.0375%	0.0956%	> 0% dan < 3% untuk *bandwidth* 75 – 100 Mbps
5	*Convergence Time*	4.2 s	4 s	-

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil dari Proyek Akhir yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Implementasi dan simulasi jaringan SDN yang memisahkan fungsi kontrol dengan fungsi *forwarding* dapat dilakukan dengan menggunakan *RouteFlow*.
2. Hasil pengujian performansi penerapan *routing* OSPF berbasis *RouteFlow* pada *Software Defined Network* menunjukkan bahawa nilai dari keempat parameter QoS (*troughput, jitter, delay* dan *packet loss*) masih berada pada nilai yang menjadi standar ITU-T G.1010.
3. Implementasi dan simulasi jaringan *software defined network* berbasis *RoteFlow* telah selesai dibuat dan di uji performansinya menggunakan parameter *convergence time, troughput, delay, jitter* dan *packet loss* dengan hasil sebagai berikut:
   a. Rata-rata *convergence time* implementasi dari hasil Proyek Akhir ini adalah 4*s*, dan simulasi adalah 4.2*s*.
   b. Rata-rata *troughput* implementasi dari hasil Proyek Akhir ini adalah 99.985 Mbit/sec, dan *troughput* simulasi adalah 99.8 *Mbit/sec*.
   c. Rata-rata *troughput* implementasi saat jalur *direct switch* A ke *switch* D diputus dari hasil Proyek Akhir ini adalah 392.48 *Mbit/sec*, dan simulasi adalah 473.78 *Mbit/sec*.
   d. Rata-rata *delay* implementasi dari hasil Proyek Akhir ini adalah 35.7 *ms*, dan simulasi adalah 27.43 *ms.*
   e. Rata-rata *jitter* implementasi dari hasil Proyek Akhir ini adalah 1.015 *ms*, dan simulasi adalah 0.008 *ms.*
   f. Rata-rata *packet loss* implementasi dari hasil Proyek Akhir ini adalah 0.0956 % dan simulasi adalah 0.0375 %.

5.2 Saran

Saran yang dapat diusulkan pada Proyek Akhir ini adalah:

1. Mengembangkan jaringan SDN berbasis *RouteFlow* menggunakan *controller* lain seperti OpenDaylight dan NOX.
2. Mengembangkan jaringan SDN yang menitikberatkan pada *controller* dan SDN *application*.
3. Membangun jaringan SDN berbasis *RoueFlow* skala besar dengan mengukur kemampuan *node* yang dapat ditampung SDN *controller*.

DAFTAR PUSTAKA

[1] Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1987. Data Structures and Algorithms. Addison-Wesley.

[2] Cisco, "SDN," 5 April 2014. [*Online*]. Diakses pada 22 Mei 2017 *Available*: https://www.cisco.com/web/ANZ/ciscolive/attend/hot_topics/sdn.html.

[3] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. Introduction to Algorithms. Cambridge, Massachusetts: MIT Press. 2nd edition.

[4] Felner, Ariel. 2011. "Position Paper: Dijkstra's Algorithm versus Uniform Cost Search oraCaseAgainst Dijkstra's Algorithm" . The Fourth International Symposium on Combinatorial Search (SoCS-2011).

[5] J. Stringer, C. Owen. 2013. "RouteMod: A Flexible Approach to Route Propagation".

[6] Linuwih, Brayan Anggita. 2016. "Penerapan dan Analisis Metode Penjaluran *Path Calculating* Menggunakan Algoritma Dijkstra".

[7] Nuruzzamanirridha, Mohammad. 2016. "Implementasi jaringan komputer berbasis *Software Defined Network* menggunakan Ryu *Controller* dan *Openvswitch*"

[8] Open Network Foundation.(2012). SoftwareDefined Networking: The New Norm for Networks.

[9] Open Networking Foundation. 2013. Software Defined Networking Security Considerations in the Data Center developer Works cn linux cluster. United States.

[10] P. Goransson and C. Black. 2014. "Software Defined Network : A Comprehensive Approach". New York : Morgan Kaufmann.

[11] Seitz, N., & NTIA/ITS. (2003). ITU-T QoS Standards for IP-Based Networks. IEEE Communications Magazine.

[12] "What is OpenWrt?." [*Online*]. Diakses pada 25 Mei 2017 *Available*: https://openwrt.org/