Implementasi *Wireless Quality of Service* dengan Metode *Load Switching* Jaringan Seluler Menggunakan *Software Defined Network* untuk Meningkatkan *Network Reliability* pada Jaringan Dinamis

Yoga Bayu Aji Pranawa, Royyana Muslim Ijtihadie, dan Waskitho Wibisono
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: roy@if.its.ac.id, waswib@if.its.ac.id

***Abstrak*—Wireless Quality of Service (QOS) adalah salah satu dimensi mobilitas, yaitu sebuah metode yang digunakan untuk menjaga kualitas suatu jaringan nirkabel. QOS diperlukan sebagai sebuah metode untuk memenuhi kriteria pelayanan sistem bagi pengguna, yaitu confidentiality, integrity, dan availability. Beberapa aspek yang menjadi topik utama dalam QOS adalah failure and recovery mechanism, variable bandwidth, computing distribution, discovery mechanism, variable lantency, dan performance feedback. Wireless yang dibahas pada penelitian ini dititik beratkan pada jaringan seluler yang cenderung tidak reliable pada daerah tertentu. Oleh karena itu dibutuhkan sebuah mekanisme yang dapat mengatasi tidak stabilnya jaringan seluler tersebut. mplementasi mekanisme yang diterapkan pada penelitian ini adalah dengan menerapkan load switching pada jaringan seluler dengan menggunakan beberapa provider dan menerapkan teknologi Software Defined Network (SDN). Berdasarkan hasi uji coba dapat disimpulkan bahwa sistem yang dibuat pada penelitian ini dapat menerapkan wireless quality of service dan meningkatkan network reliability sebesar 65,29% dan 83,87% lebih baik untuk penggunaan tanpa waktu tunggu dan dengan waktu tunggu pada suatu jaringan dinamis.**

***Kata Kunci*—wireless quality of service, software defined network, load switching, jaringan seluler, OpenFlow.**

I. PENDAHULUAN

PERKEMBANGAN teknologi informasi (khususnya *internet*) yang sangat pesat dapat merubah pola hidup manusia dalam menjalani aktivitas sehari-hari. *Internet* dibutuhkan karena hampir semua aktivitas manusia pada era modern ini menuntut mobilisasi koneksi yang tinggi. Selain dituntut untuk mobilisasi koneksi yang tinggi, manusia pada era modern ini juga dituntut untuk *multitasking*, artinya dapat melakukan beberapa tugas atau aktivitas dalam waktu yang bersamaan. Oleh karena itu, dibutuhkan koneksi *internet* yang memadai untuk memenuhi kebutuhan mobilisasi dan *multitasking* yang dapat dinikmati kapanpun dan dimanapun meski sedang dalam perjalanan.

Salah satu infrastruktur yang digunakan agar dapat menikmati *internet* adalah melalui jaringan seluler (*Cellular Network*). Jaringan seluler menggunakan *Base Transceiver Station* (BTS) sebagai titik akses. Teknologi pada jaringan seluler yang berkembang saat ini adalah teknologi 4G (*Fourth Generation*). Teknologi 4G menggunakan *Long-Range Base Technologies* yang memungkinkan pengguna untuk melakukan akses kepada layanan yang berbeda. Layanan 4G akan memberikan akses data melalui beberapa teknologi berbasis IP dan menawarkan beberapa *bit rates* sampai dengan 50 Mbps [1]. Meskipun teknologi jaringan seluler yang sudah dikembangkan saat ini sudah mendukung kecepatan akses yang tinggi dan jangkauan yang lebih luas, perangkat komunikasi yang menggunakan jaringan seluler harus berada pada range BTS agar dapat berkomunikasi satu sama lain. Apabila sebuah perangkat seluler terlalu jauh dari *tower* BTS maka perangkat tersebut tidak dapat melakukan komunikasi secara maksimal.

Ketika sedang melakukan suatu perjalanan khususnya perjalanan darat (kereta api, bus, maupun kendaraan pribadi) seringkali terdapat suatu wilayah yang tidak dapat dijangkau oleh jaringan seluler. Hal ini menyebabkan pengguna tidak dapat mengakses suatu informasi melalui jaringan dan tidak memenuhi persyaratan *Quality of Service* (QOS). Agar memenuhi persyaratan *Quality of Service*, salah satu poin yang harus dipenuhi adalah suatu arsitektur jaringan tersebut dapat menangani *bandwidth* yang berubah-ubah (*variable bandwidth*) setiap saat [1]. Salah satu metode yang dapat digunakan untuk menangani hal ini adalah dengan melakukan *load switching* untuk pemilihan bandwidth tertinggi pada beberapa ISP menggunakan *Software Defined Network* (SDN) yang di implementasikan pada teknologi WLAN. *Load switching* yang dimaksud adalah pemilihan jalur aliran data dipilih berdasarkan kondisi *resource* yang terbaik dengan menggunakan parameter tertentu.

Software Defined Network (SDN) dipilih sebagai implementasi sistem karena SDN menyediakan fasilitas *Application Programming Interface* (API). API yang tersedia pada SDN dapat mengetahui statistik suatu perangkat dan memberikan kebebasan *programmer* untuk mengembangkan

sebuah aplikasi pada perangkat tersebut. Infrastruktur SDN juga memungkinkan sebuah arsitektur jaringan tidak terpaku pada perangkat dengan *vendor* tertentu, karena SDN menggunakan standar protokol OpenFlow untuk berkomunikasi dengan sebuah perangkat.

Makalah ini membahas tentang penerapan *load switching* jaringan seluler dengan menggunakan *software defined network* yang ditujukan untuk meningkatkan *network reliability* pada jaringan dinamis dan menerapkan aspek *wireless quality of service*. Struktur penulisan makalah ini diisusun sebagai berikut: bagian 2 membahas tentang tinjauan pustaka. Bagian 3 membahas tentang analisis dan perancangan arsitektur sistem yang digunakan pada makalah ini. Bagian 4 berisi implementasi sistem yang telah dirancang pada lingkungan pengembangan. Bagian 5 adalah uji coba yang telah dilakukan dengan melakukan perhitungan *delay time* dan *reliability*. Bagian 6 merupakan kesimpulan dari keseluruhan sistem yang telah dibuat.

II. TINJAUAN PUSTAKA

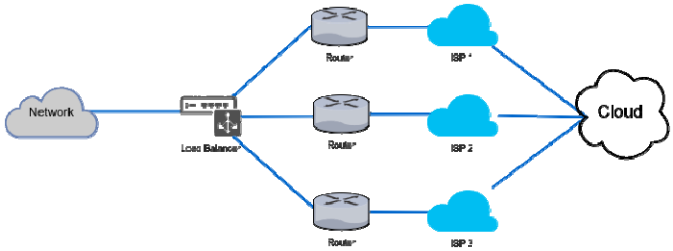
A. Wireless Quality of Service

Quality of Service (QOS) dalam konteks jaringan nirkabel (*wireless*) adalah salah satu dimensi mobilitas, yaitu suatu hal yang dapat membedakan aplikasi *mobile* berbeda dengan kondisinya ketika diam [1]. QOS diperlukan sebagai sebuah metode untuk memenuhi kriteria pelayanan sistem bagi pengguna, yaitu *confidentiality, integrity,* dan *availability*. Beberapa aspek yang menjadi topik utama dalam QOS adalah *failure and recovery mechanism, variable bandwidth, computing distribution, discovery mechanism, variable lantency,* dan *performance feedback*. Solusi yang diberikan berdasarkan problem dari beberapa aspek tersebut berbeda-beda tergantung dari arsitektur yang kita tentukan untuk mengimplementasikan sistem yang kita buat.

B. Load Switching

Load Switching adalah sebuah metode yang termasuk dalam metode *load balancing* yang berfungsi untuk mendistribusikan lalu lintas data diantara beberapa sumber daya *server* [2]. Sebuah perangkat atau aplikasi yang memiliki fungsi untuk menjalankan metode *load balancing* disebut dengan *load balancer*. *Load balancer* bertindak sebagai pengatur lalu lintas yang menerima permintaan lalu lintas yang datang. Kemudian *load balancer* meneruskannya kepada semua *server* yang mampu menangani permintaan tersebut dengan cara memaksimalkan kecepatan dan kapasitas dari pendayagunaan *server* dan memastikan bahwa tidak ada satupun *server* yang bekerja terlalu banyak *(overworked)* sehingga menyebabkan performa *server* menurun. Implementasi *load balancer* memiliki fungsi yang beragam, diantaranya adalah sebagai *load switching, traffic engineering,* dan *intelligent traffic switching*.

Algoritma yang digunakan pada metode *load balancing* sangat bervariasi, tergantung dengan kebutuhan sistem dan permasalahan yang akan ditangani. Algoritma *load balancing* yang sering digunakan pada makalah ini adalah penerapan dari algoritma *least connection*. Algoritma *least connection* yaitu sebuah metode pendistribusian lalu lintas data yang masuk dengan cara mengirimkan permintaan yang baru kepada *server* dengan koneksi aktif ke klien yang paling sedikit [4]. Namun pada makalan ini akan dimodifikasi dengan menggunakan koneksi aktif ke *server* yang paling tinggi. Sehingga koneksi yang didapatkan oleh *client* lebih stabil.



Gambar 1. Algoritma *internet service provider load switching*

C. Jaringan Seluler

Jaringan seluler *(Cellular Network)* adalah sebuah jaringan telekomunikasi yang menggunakan jaringan radio dan menggunakan beberapa pemancar yang berdaya rendah *(multiple low-power transmitter)* kurang dari 100 W [6]. Karena sistem pemancar yang digunakan memiliki daya yang rendah, maka jaringan seluler dibagi menjadi beberapa area. Setiap area pada jaringan seluler dibagi menjadi beberapa sel *(cell)*, setiap sel akan dilayani oleh sebuah antena dan akan dialokasikan pada frekuensi tertentu yang dilayani oleh *base station*. Sebuah *base station* terdiri dari *transmitter, receiver,* dan *control unit*.

Setiap *base station* yang berdekatan dapat berkomunikasi dan akan membentuk sebuah *pattern* yang menyelimuti area tersebut. Pattern yang digunakan biasanya membentuk *square pattern* atau *hexagonal pattern.*

D. Software Defined Network

Software Defined Network (SDN) merupakan sebuah arsitektur jaringan yang dinamis, dapat dikelola (*manageable*), hemat biaya, dan mudah beradaptasi dengan perubahan kondisi jaringan [7]. Beberapa komponen penting dalam *Software Defined Network* yang penting adalah *controller* dan protokol OpenFlow. *Controller* merupakan sebuah aplikasi yang menjadi titik strategis dalam *Software Defined Network*. *Controller* mengelola aliran kontrol pada *switch*/*router* melalui *southbound* APIs dan mengelola aplikasi beserta proses bisnis melalui *northbound* APIs untuk mengimplementasikan jaringan cerdas [9]. OpenDaylight *Controller* adalah salah satu *controller* Java *Virtual Machine* (JVM) software yang dapat berjalan di semua sistem operasi dan hardware selama mendukung Java [10]. OpenFlow adalah sebuah standar komunikasi pada jaringan yang mendefinisikan antara lapisan kontrol dan lapisan *forwarding* dari sebuah arsitektur SDN. OpenFlow memungkinkan akses langsung dan melakukan manipulasi dari *forwarding plane* pada sebuah perangkat jaringan seperti *switch* dan *router*, baik secara fisik maupun virtual (*hypervisor-based*) [11].

E. Dynamic Network Area

Dynamic network adalah suatu kondisi dimana jaringan yang diterima pada sebuah perangkat mengalami perubahan kenaikan atau penurunan secara signifikan ketika berada di suatu wilayah tertentu. *Dynamic network* biasanya ditemui ketika sebuah perangkat berpindah dari satu tempat ke tempat lainnya.

Salah satu *area* yang termasuk *area* paling buruk pada jaringan dinamis adalah *blank spot area*. *Blank spot* area atau dalam referensi lain disebut sebagai *dead zone*, *call drop*, dan lain-lain adalah sebuah *area* dimana jaringan seluler tidak dapat digunakan karena mengalami interferensi atau kondisi sinyal yang rendah (karena terlalu jauh dengan *tower* BTS) [6]. Akibat dari *blank spot area* ini adalah paket yang dikirimkan oleh *client* akan di *drop* oleh sistem, sehingga komunikasi dan pertukaran data tidak dapat dilakukan.

III. ANALISIS DAN PERANCANGAN SISTEM

A. Deskripsi Umum Sistem

Sistem yang dibangun pada makalah ini berupa sebuah aplikasi *load switching* yang diimplementasikan pada *software defined network (SDN)* dengan *controller* OpenDaylight. Aplikasi *load switching* yang dibuat diharapkan dapat beradaptasi dengan kondisi jaringan yang tidak stabil, khususnya jaringan seluler. Sehingga aplikasi dapat menyesuaikan dan meminimalisasi adanya gangguan ketika kondisi jaringan sedang tidak stabil.

Secara umum, aplikasi akan menggunakan 4 (empat) *modem* sebagai jalur koneksi. Berdasarkan 4 (empat) perangkat modem tersebut akan diambil koneksi terbaik berdasarkan kualitas *throughput* yang dihasilkan oleh masing-masing koneksi per detik. Masing-masing data *throughput* tersebut kemudian dikirimkan ke *controller* oleh *host router* untuk memodifikasi tabel *throughput port* (tx dan rx) pada tabel *port controller*. Setelah data *throughput* pada tabel *throughput port* pada *controller* dimodifikasi, maka aplikasi yang berada pada mesin *controller* akan memasukkan data tersebut kedalam *database* agar data tersimpan dan dapat dianalisis lebih lanjut.

Setelah data *throughput* dimasukkan kedalam *database* maka aplikasi akan menentukan jalur terbaik saat itu. Berikut ini merupakan rumus parameter yang dikirim oleh *host router*:

$$Parameter = \frac{Rx_{bps} + Tx_{bps}}{RTT} \quad (1)$$

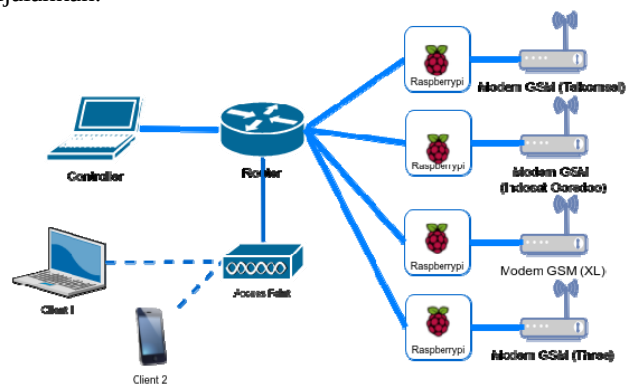
Keterangan:

Rx_{bps} = *Throughput* paket yang diterima per detik (*bytes*)

Tx_{bps} = *Throughput* paket yang dikirim per detik (*bytes*)

RTT = *Round Trip Time* (waktu yang dibutuhkan sebuah paket untuk kembali ke pengirim) (*miliseconds*)

kemudian aplikasi akan melakukan pembuatan *flow* (jalur data) yang akan digunakan oleh *client* ke *internet*. Proses tersebut akan berlangsung terus menerus selama aplikasi dijalankan.



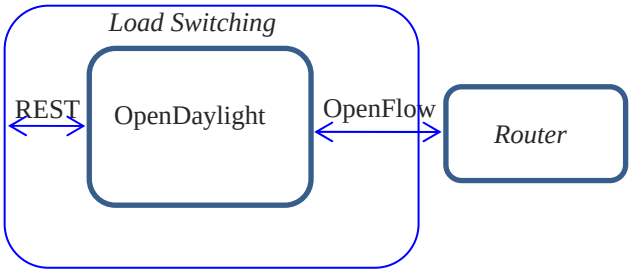
Gambar 2. Arsitektur sistem secara umum

Arsitektur sistem secara umum ditunjukkan pada gambar 2. Pada gambar 2 diatas dapat dilihat bahwa mesin *controller* akan menjalankan 2 (dua) aplikasi yaitu *controller* OpenDaylight dan aplikasi *load switching*. *Router* akan menjalankan protokol OpenFlow. Sedangkan *host router* akan menjalankan aplikasi *throughput* monitor dan melakukan pengiriman parameter *throughput*.

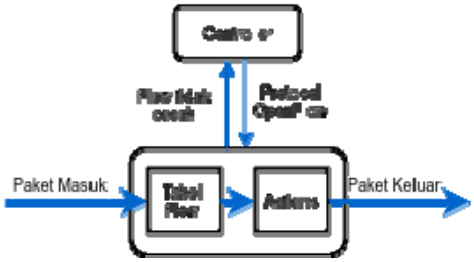
B. Arsitektur Sistem

Rancangan arsitektur sistem pada makalah ini dibuat dengan menggunakan protokol OpenFlow untuk berkomunikasi dengan *controller*. Protokol OpenFlow dirancang dengan memisahkan antara lapisan aplikasi (*application layer*), lapisan kontrol (*control layer*), dan lapisan infrastruktur *(infrastructure layer*). Arsitektur sistem OpenFlow yang dipakai menggunakan jalur *Secure Socket Layer* (SSL) untuk menghubungkan antara *controller* dengan OpenFlow *Enabled Switch*. Arsitektur yang digunakan ditunjukkan pada gambar 3 dibawah.

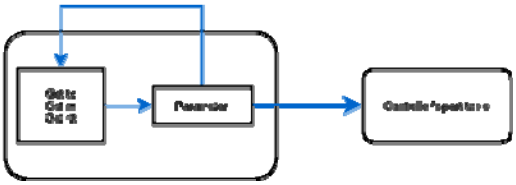
Alur dari proses sebuah pertukaran data dimulai dari masuknya suatu paket ke dalam OpenFlow *switch*. OpenFlow *switch* akan mengirimkan data paket tersebut kepada *controller* untuk melakukan pengecekan pada *flow table* terhadap sebuah paket, apakah sudah terdapat data paket yang *match* dengan paket yang baru masuk. Kemudian *controller* akan memberikan respon terhadap paket yang masuk untuk membuat suatu perlakuan terhadap paket tersebut, apakah di-*drop* atau dibuatkan sebuah *flow table* baru berdasarkan paket yang masuk tersebut. Secara garis besar, alur dari proses penanganan sebuah paket digambarkan pada gambar 4 dibawah.



Gambar 3. Arsitektur sistem dengan *software defined netowk*



Gambar 4. Proses penanganan sebuah paket pada OpenFlow



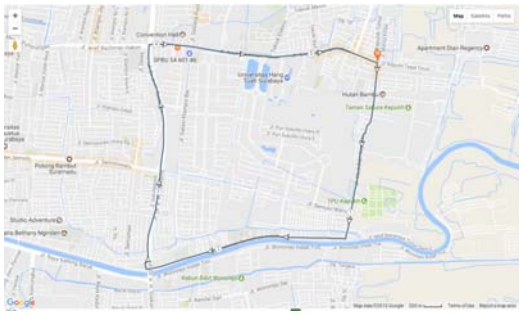
Gambar 5. Diagram sistem *host router*

Aplikasi yang berjalan pada *host router* secara periodik akan melakukan kalkulasi *throughput* dan mengirimkannya ke *host router* yang lain untuk melakukan modifikasi data *throughput port* pada lapisan kontrol. Sistem yang berjalan pada *host router* digambarkan pada gambar 5 diatas.

IV. IMPLEMENTASI

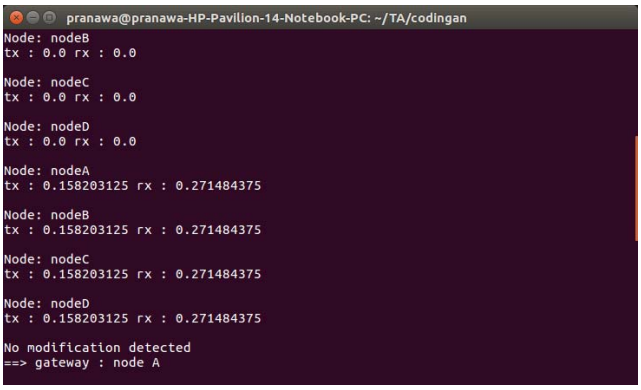
A. *Implementasi Rute*

Implementasi rute memiliki tujuan untuk menentukan daerah yang memiliki kualitas jaringan yang kurang stabil. Rute dipilih berdasarkan analisis *coverage*, analisis *download*, dan analisis *upload* pada suatu wilayah. Aplikasi yang digunakan sebagai acuan analisis adalah www.sensorly.com yang menyediakan data statistik jaringan seluer. Berdasarkan data didapat pada sensorly.com, maka ditentukan rute sebagai berikut:

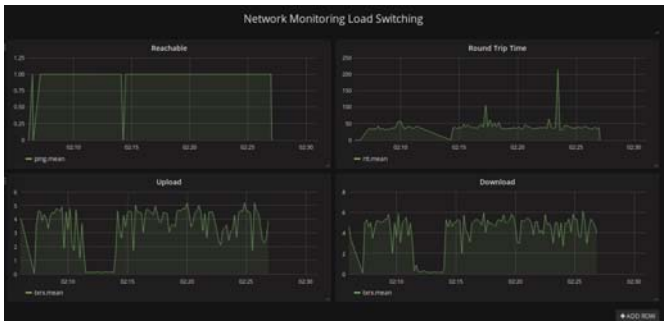


Gambar 6. Implementasi rute yang ditempuh untuk uji coba (diambil dari www.plotaroute.com)

B. *Implementasi Antarmuka*



Gambar 7. Tampilan antarmuka *load switching* pada *terminal*



Gambar 8. Tampilan antarmuka *network monitoring* pada *browser*

Implementasi antarmuka pada makalah ini menggunakan antarmuka *command line* pada *terminal* untuk antarmuka *load switching*, dan menggunakan antarmuka *web* untuk menampilkan grafik *network monitoring*. Tampilan antarmuka pada makalah ini ditunjukkan pada gambar 7 dan gambar 8 diatas.

V. UJI COBA DAN EVALUASI

Pengujian performa pada sistem ini dilakukan untuk mengetahui seberapa baik kualitas *throughput* yang didapat oleh pengguna sebelum dan sesudah menggunakan mekanisme *load switching* menggunakan *software defined network* dengan *controller* OpenDaylight. Uji coba performa yang dilakukan meliputi pengukuran *network reachable*, pengukuran *round trip time*, pengukuran pengunduhan berkas, dan pengukuran unggah berkas. Pada uji coba performa yang dilakukan akan dihitung jumlah *delay time* yaitu total setiap *delay time* yang dibutuhan oleh sistem untuk menaikkan kualitas jaringan ketika terjadi penurunan kualitas jaringan secara signifikan. Perhitungan *delay time* dan presentase *reliability* ditunjukkan pada persamaan berikut ini.

Selisih *delay time* tanpa waktu tunggu = 1224,5 - 425 = 799,5 detik

$$reliability\ tanpa\ waktu\ tunggu = \frac{799,5}{1224,5} x\ 100 = 65,29\ \%$$

Selisih *delay time* dengan waktu tunggu = 1224,5 - 197,5 = 1027 detik

$$reliability\ dengan\ waktu\ tunggu = \frac{1027}{1224,5} x\ 100 = 83,87\ \%$$

Berdasarkan hasil uji performa didapatkan data sebagai

berikut:

Tabel 1.
Tabel rata-rata jumlah *delay time* pada uji performa tanpa *load switching*

Uji Performa tanpa *Load Switching*	**Jumlah *Delay Time* (detik)**
Uji performa *network reachable*	1106
Uji performa *round trip time*	1232
Uji performa pengunduhan	1280
Uji performa pengunggahan	1280
Rata-rata	1224,5

Tabel 2.
Tabel rata-rata jumlah *delay time* pada uji performa menggunakan *load switching* tanpa waktu tunggu

Uji Performa menggunakan *Load Switching* tanpa Waktu Tunggu	**Jumlah *Delay Time* (detik)**
Uji performa *network reachable*	170
Uji performa *round trip time*	380
Uji performa pengunduhan	540
Uji performa pengunggahan	610
Rata-rata	425

Tabel 3.
Tabel rata-rata *delay time* pada uji performa menggunakan *load switching* dengan waktu tunggu

Uji Performa menggunakan *Load Switching* dengan Waktu Tunggu	**Jumlah *Delay Time* (detik)**
Uji performa *network reachable*	60
Uji performa *round trip time*	200
Uji performa pengunduhan	240
Uji performa pengunggahan	290
Rata-rata	197,5

Tabel 4.
Tabel perbandingan rata-rata jumlah *delay time*

Uji Performa	**Jumlah Rata-rata *Delay Time* (detik)**
Uji performa tanpa *load switching*	1224,5
Uji performa menggunakan *load switching* tanpa waktu tunggu	425
Uji performa menggunakan *load switching* dengan waktu tunggu	197,5

VI. KESIMPULAN

Implementasi *load switching* jaringan seluler menggunakan *software defined network* yang dilakukan telah berhasil diterapkan dengan baik. Implementasi *load switching* yang dibuat telah menerapkan aspek *wireless quality of service* dengan dibuktikan dengan penurunan *delay time* dan peningkatan *network reliability*.

Kualitas jaringan yang didapatkan menggunakan sistem *load switching* menggunakan *software defined network* memiliki pengurangan jumlah *delay time* rata-rata 799,5 detik apabila tanpa menggunakan waktu tunggu dan 1027 detik apabila menggunakan waktu tunggu. Sistem yang dibuat memiliki tingkat *reliability* sebesar 65,29% apabila tanpa menggunakan waktu tunggu dan 83,87% apabila menggunakan waktu tunggu.

DAFTAR PUSTAKA

[1] R. B'Far, "*Mobile Computing Principles*", Cambridge: Cambridge University Press, 2005.

[2] Kopparapu, Chandra, *load switching Servers, Firewalls, and Caches*, New York: New York, 2002.

[3] C. Ellrod, "Load Balancing – Round Robin," Citrix, 3 September 2010. [Online].Available:https://www.citrix.com/blogs/2010/09/03/load-balancing-round-robin/. [Diakses 6 Januari 2017].

[4] C. Ellrod, "Load Balancing – Least Connections," Citrix, 2 September 2010.[Online].Available:https://www.citrix.com/blogs/2010/09/02/load-balancing-least-connections/. [Diakses 6 Januari 2017].

[5] C. Ellrod, "Load Balancing – Hash Method," Citrix, 4 September 2010. [Online].Available:https://www.citrix.com/blogs/2010/09/04/load-balancing-hash-method/. [Diakses 6 Januari 2017].

[6] W. Stallings, "*Wireless Connections and Networking Second Edition*", New Jersey: Pearson Education, 2005.

[7] "*Software-Defined Networking (SDN) Definition*" , Open Networking Foundation.[online].Available:https://www.opennetworking.org/sdn-resources/sdn-definition. [diakses 6 Januari 2017].

[8] "*Wireless Local Area Network (WLAN)*", Techopedia. [Online].Available:https://www.techopedia.com/definition/5107/wireless-local-area-network-wlan. [diakses 6 Januari 2017].

[9] "*What are SDN Controllers (or SDN Controllers Platforms)?*", SDX Central.[Online].Available:https://www.sdxcentral.com/sdn/definitions/sdn-controllers/ [diakses 6 Januari 2017].

[10] "*OpenDaylight User Guide*", Linux Foundation. [Online]. Available:https://www.opendaylight.org/sites/opendaylight/files/bk-user-guide.pdf. [diakses 6 Januari 2017].

[11] "*OpenFlow*", Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow. [diakses 6 Januari 2017]