IMPLEMENTASI RIP PADA JARINGAN BERBASIS *SOFTWARE DEFINED NETWORK* (SDN)

IMPLEMENTATION RIP ROUTING ON SOFTWARE DEFINED NETWORK

Ivan Hidayah[1], Indrarini Dyah[1], Yuli Sun Hariyani[1]
[1]Prodi D3 Teknik Telekomunikasi, Fakultas Ilmu Terapan, Universitas Telkom
ivanhidayah90@gmail.com, indrarini@telkomuniversity.ac.id, yulisun@tass.telkomuniversity.ac.id

Abstrak

Pada jaringan konvensional dahulu, seorang *Network Engineer* harus mengkonfigurasi secara individual pada setiap *intermediate device* yang ada pada infrastrukrur jaringan. Kemudian muncullah *Software Defined Network* (SDN) yang melakukan pemisahan secara eksplisit antara *control plane* dan *data plane* dengan menggunakan protokol *OpenFlow* sebagai protokol komunikasi antara 2 fungsi tersebut, dimana pada jaringan konvensional fungsi tersebut berada dalam satu perangkat. *Routing Information Protocol* v2 (RIPv2) merupakan protokol routing yang menggunakan algoritma *distance-vector* (*Bellman Ford*). Maka pada proyek akhir ini dilakukan simulasi dan implementasi untuk mengetahui kinerja protokol *routing* RIPv2 pada jaringan SDN skala kecil menggunakan *emulator* mininet sebagai simulasi dan TP-Link WR-1043ND v2 sebagai pengimplementasiannya. Dengan menggunakan *controller* POX dan mengaplikasikan *routing* konvensional *RouteFlow* yang dikonfigurasi dengan protokol *routing* RIPv2 yang kemudian dianalisis berdasarkan parameter *throughput, delay, jitter, packet loss, dan convergence time*.

Kata Kunci: Software Defined Network, RIPv2, RouteFlow, tp-link wr-1043nd v2

Abstract

In the first conventional network, a network engineer must to individually configure every intermediate device on network infrastructure. Then came a Software Defined Network which do explicitly separate between control plane and data plane used OpenFlow as a communication protocol between those 2 function, where on conventional network those function is on one device. Routing Information Protocol v2 (RIPv2) is a routing protokol which used distance-vector algorithm (Bellman Ford). Therefore in this final project do the simulation and implementation to know how routing protocol RIPv2 work on small scale SDN network using mininet emulator as a simulation and TP-Link WR-1043ND v2 as a implementation. By using POX controller dan applied conventional routing RouteFlow configured with routing protocol RIPv2 which are then analyzed based on some parameters like throughput, delay, jitter, packet loss, and convergence time.

Keywords: Software Defined Network, RIPv2, RouteFlow, tp-link wr-1043nd v2

1. Pendahuluan

Jaringan konvensional terus berkembang seiring dengan semakin kompleksitas sebuah jaringan tersebut. Terdapat beberapa algoritma *routing* yang telah digunakan pada sebuah infrastruktur jaringan, salah satunya adalah algoritma *Bellman-Ford*. Masalah yang ditimbulkan adalah apabila seseorang ingin bereksperimen dengan beberapa teknik *load-balancing* baru atau *routing protocol* baru dia tidak akan dapat melakukan hal itu Karena sistem operasi dan fitur sudah langsung *embedded* pada perangkat tersebut dan perangkat tidak mendukung teknik dan protokol tersebut [8]. Solusinya adalah pemisahan *control plane* dan *data plane* pada perangkat jaringan komputer seperti *Router* dan *Switch* yang memungkinkan untuk memprogram perangkat tersebut sesuai dengan uang diinginkan secara terpusat [8]. Paradigma ini memberikan pandangan dari seluruh jaringan [1]. Paradigma tersebut bernama *Software Defined Network* (SDN). *Routing Information Protocol* (RIP) merupakan sebuah protokol *routing* dinamis telah dikembangkan sehingga tercipta RIPv2. Kinerja dari RIPv2 ini masih dikategorikan rendah dari segi konvergensi dan skalabilitas.

Sebelumnya telah dilakukan penelitian tentang implementasi SDN menggunakan RYU *Controller* dan *Open vSwitch* namun penelitian tersebut hanya menggunakan protokol *routing static*. Selain itu juga dilakukan penelitian tentang simulasi jaringan SDN menggunakan *RouteFlow* dan protokol *routing* RIPv2, namun pada penelitian tersebut belum dilakukan implementasinya. Oleh Karena itu penelitian SDN yang dilakukan ini melakukan sebuah implementasi jaringan SDN yang menggunakan *RouteFlow* sebagai *controller*, RIPv2 sebagai protokol *routing dinamis*, dan TP-Link WR-1043ND v2 sebagai perangkat *forwarding* dan mengukur nilai QoS berupa throughput, delay, jitter, packet loss, dan convergence time.

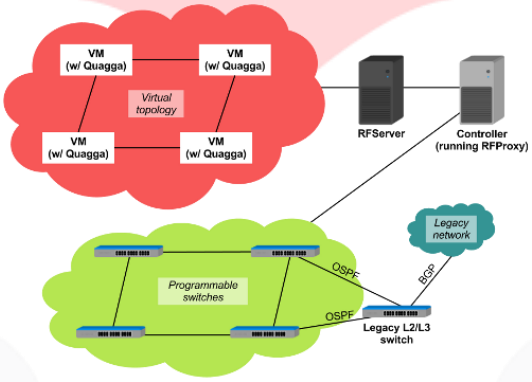
2. Dasar Teori

2.1 OpenFlow

OpenFlow adalah *interface* standar komunikasi pertama yang ditetapkan antara *control layer* dan *forwading layer* dari suatu arsitektur SDN [2]. Standar ini mendefinikan bagaimana koneksi antara *controller* dan *datapath* dapat dibentuk, menyediakan satu set struktur untuk mendefinisikan perilaku *datapath* terutama serangkaian pencocokan untuk mengklasifikasikan paket, dan serangkaian tindakan untuk dilakukan ketika sebuah paket cocok [4].

2.2 RouteFlow

RouteFlow merupakan proyek *open source* yang menyediakan layanan *virtualized ip routing* melalui perangkat *OpenFlow*. *RouteFlow* tersusun dari *OpenFlow controller apllication* (RFProxy), *RouteFlow server* (RFServer), dan jaringan virtual yang menghubungkan peralatan fisik dan menjalankan *IP routing engines* [5]. *Routing engine* pada *RouteFlow* dijalankan pada oleh Quagga [7].



Gambar 1 Skenario RouteFlow [4]

RouteFlow core terdiri oleh tiga aplikasi dasar: RFClient, RFServer dan RFProxy [6].

1. RFClient berjalan sebagai daemon di *Virtual Machine* (VM), mendeteksi perubahan dalam ARP Linux dan tabel *routing*. informasi *routing* dikirim ke RFServer ketika ada pembaruan.
2. RFServer adalah *standalone application* yang terhubung ke RFClient dan mengelola RFClient *instances* yang *running* di VMs. RFServer membuat *mapping* antara RFClient *instances*, *interfaces*, *switch* yang sesuai dan *port*. Menghubungkan ke RFProxy untuk mengkonfigurasi *flow mapped* dari *virtual environment*.
3. RFProxy adalah sebuah aplikasi (untuk POX dan *controller* lainnya) yang bertanggung jawab atas interaksi dengan *switch OpenFlow* (diidentifikasi oleh *datapaths*) melalui protokol *OpenFlow*. RFProxy mengikuti instruksi dari RFServer dan memberitahukan sesuatu yang terjadi dalam jaringan

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* [3]. 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 *clients* [7].

2.4 *Routing Information Protocol* (RIP)

Routing information protocol (RIP) adalah salah satu jenis protokol *routing* yang efektif dalam skala jaringan kecil dan menggunakan protokol *routing* distance vector sebagai referensi dalam penentuan jalurnya. Memiliki karakteristik utama sebagai berikut [9] :

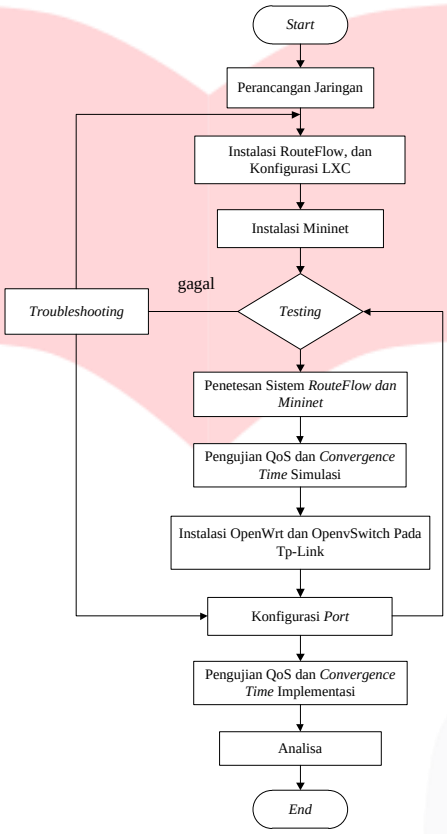
1. Perhitungan *hop* digunakan sebagai *metric* untuk memilih jalur.
2. Jika jumlah *hop* pada suatu jaringan lebih dari 15 *hop*, maka RIP tidak dapat menampung rute pada jaringan tersebut.
3. *Routing Update* di *broadcast* atau *multicast* setiap 30 detik, secara *default*.

RIP memiliki beberapa *timer* yang berkaitan dengan table *routing* :

1. *Update timer* (setiap 30 detik). *Timer* ini untuk pengiriman *response message* yang digunakan pada *update* tabel *routing*
2. *Timeout timer* ((180 detik) dimulai ketika suatu rute didirikan dan setiap kali menerima *update message* untuk rute tersebut). Setelah berakhirnya batas waktu dari *timer* ini, rute tersebut menjadi *invalid*.
3. *Garbage-collection timer* ((120 detik) dimulai ketika berakhirnya batas waktu dari *timeout timer* atau ketika *metric* untuk suatu rute diatur ke 16).

3. Perancangan Sistem

3.1 Metode Penelitian

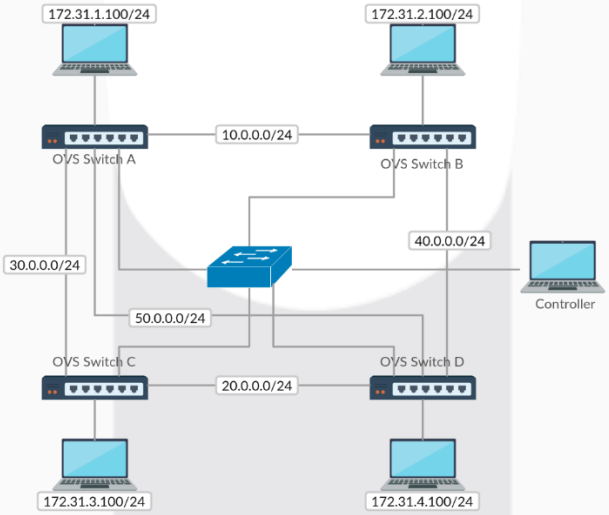


Gambar 2 *Flowchart* Metode Penelitian

3.2 Perancangan Jaringan

Pada tahap ini penulis melakukan perancangan jaringan berupa pembuatan topologi jaringan. Desain topologi jaringan pada proyek akhir ini menggunakan satu buah laptop sebagai *controller*, empat buah TP-Link sebagai *switch* dan arus data *forwarding*, dan 2 buah PC atau Laptop sebagai host.

Topologi jaringan yang digunakan pada proyek akhir ini adalah seperti gambar dibawah ini:



Gambar 3 Topologi Jaringan

3.3 Perancangan Sistem Kontrol

Tahap ini merupakan tahapan dalam merancang dan membuat kontrol jaringan atau *controller* jaringan berbasis SDN. RouteFlow dipasang pada sistem operasi Ubuntu 12.04. Terdapat folder rfvm dan file-file lain didalam folder rftest yang mendukung *RouteFlow environment*. Penulis memodifikasi beberapa file dan membuat file *ripd.conf* untuk konfigurasi *routing* pada *RouteFlow* tersebut.

3.4 Perancangan Forwarding

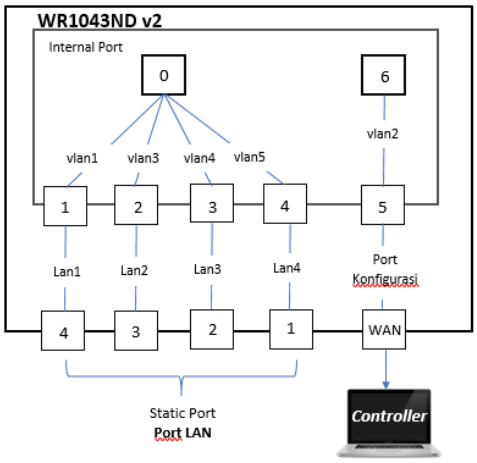
Pada tahap perancangan *forwarding* yang dilakukan adalah merancang dan membuat sistem perutean pada jaringan berbasis SDN. Pada metode simulasi topologi *OpenFlow switch* konfigurasi dilakukan pada mininet. Sedangkan pada metode implementasi dibagi menjadi dua bagian yaitu proses instalasi dan konfigurasi port.

3.4.1 Instalasi OpenWrt dan *Open vSwitch*

Pada proyek akhir ini menggunakan perangkat *wireless router* TP-Link WR1043ND v2. Perangkat tersebut di install OpenWrt dan *Open vSwitch*. OpenWrt yang digunakan adalah OpenWrt versi 15.05 Chaos Calmer yang di *build* dengan melakukan *compile* dan menanamkan *Open v*Switch versi 2.3.90.

3.4.2 Konfigurasi *Port*

Port pada perangkat wireless router TP-Link WR1043ND v2 dikonfigurasi dengan memisahkan port yang terhubung ke controller, port untuk konfigurasi, dan port untuk Local Area Network. Berikut ini adalah *schematic* dan konfigurasi *port* pada TP-Link WR1043ND v2 menggunakan OpenWrt 15.05 Chaos Calmer:

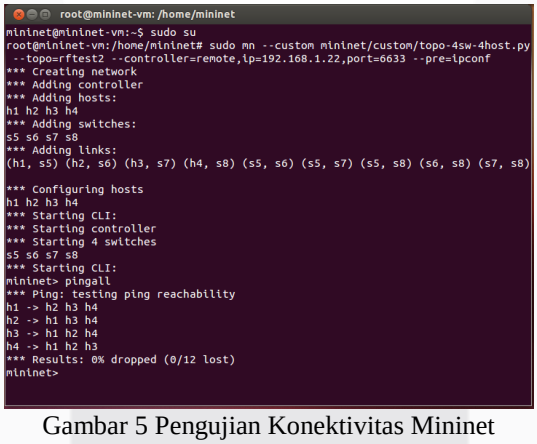


Gambar 4 Konfigurasi Port

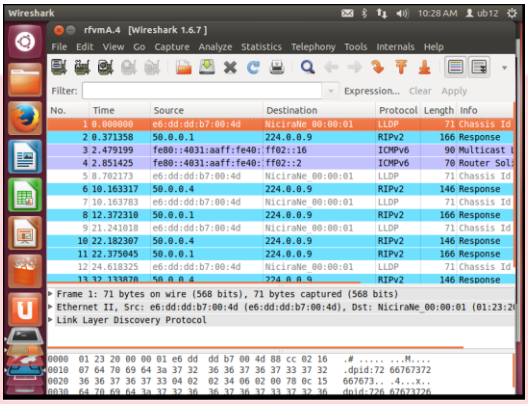
4 Hasil dan Analisa

4.1 Pengujian Simulasi

Pada proyek akhir ini, pengujian simulasi mencakup beberapa aspek diantaranya, konektivitas antara mininet dengan *RouteFlow* dan protokol *routing* RIP.



Gambar 5 Pengujian Konektivitas Mininet



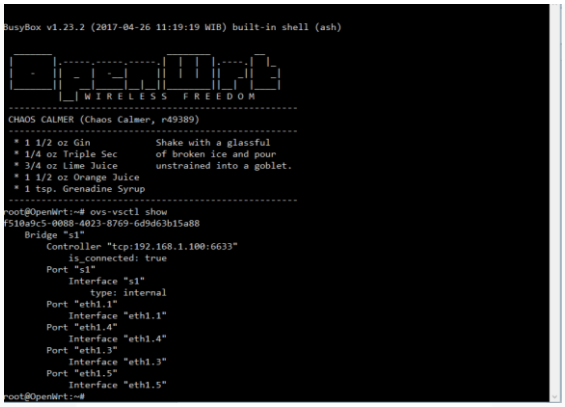
Gambar 6 Pengujian Protokol Routing

4.2 Pengujian Implementasi

Pengujian implementasi bertujuan untuk mengetahui performansi dan fungsionalitas perangkat. Berikut penjelasannya:

4.2.1 Pengujian Switch Forwarding

Pengujian ini dilakukan pada switch TP-Link WR1043ND v2 menggunakan OpenWrt 15.05 Chaos Calmer. *Switch* dan *controller* sudah terhubung dan terintegrasi yang dibuktikan oleh *is_connected:true* dengan perintah *ovs-vsctl show* pada TP-Link seperti gambar berikut:



Gambar 7 Switch dan Controller sudah terintegrasi

4.2.2 Pengujian Fungsionalitas

Untuk bisa membuktikan *routing* bekerja dengan baik dan benar, maka dilakukan pengecekan koneksi dengan melakukan ping *host-to-host* antara h1 dan h4 seperti gambar dibawah ini:

Gambar 8 Hasil Ping h1 ke h4

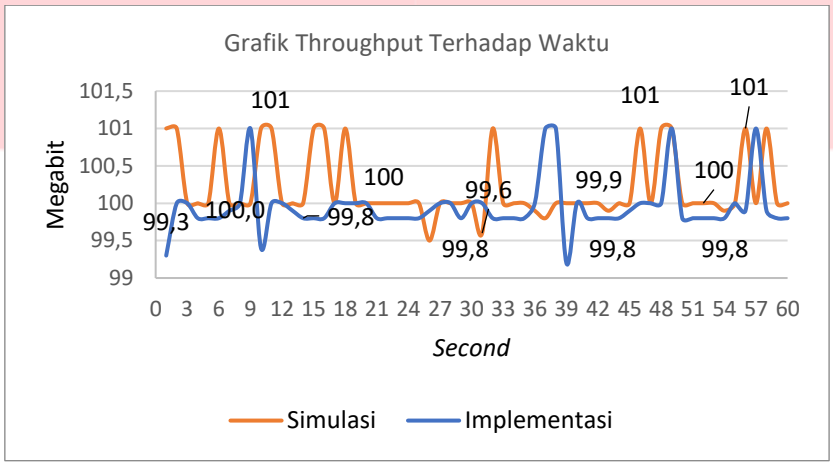
4.3 Pengukuran Performansi

Pengukuran ini menggunakan *tool iperf*. Pengujian ini dilakukan pada h1 ke h4 yang sudah saling terhubung dan dapat berkomunikasi menggunakan protokol *routing* RIP dan menggunakan *measurement software*

berupa *iperf* dengan metode *server-side* (h4) pada pengirim dan *client-side* (h1) pada penerima dan dilakukan pada simulasi dan implementasi dengan waktu 60 detik.

4.3.1 Throughput

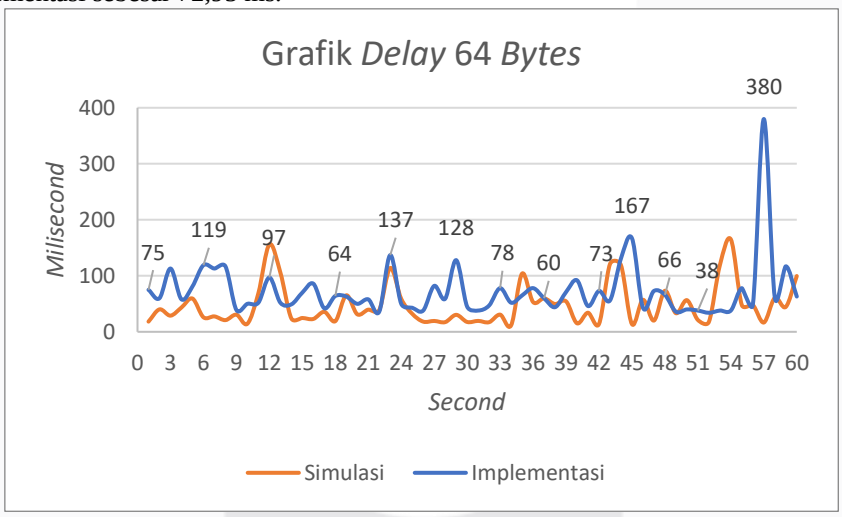
Pengukuran *throughput* ini bertujuan untuk mengetahui kemampuan jaringan dalam mentransmisikan data secara aktual. Dari pengukuran tersebut diperoleh hasil rata-rata *throughput* simulasi adalah 100.21 Mbps, dan hasil rata-rata *throughput* implementasi adalah 99,94 Mbps.



Gambar 9 Grafik Throughput Terhadap Waktu

4.3.2 Delay

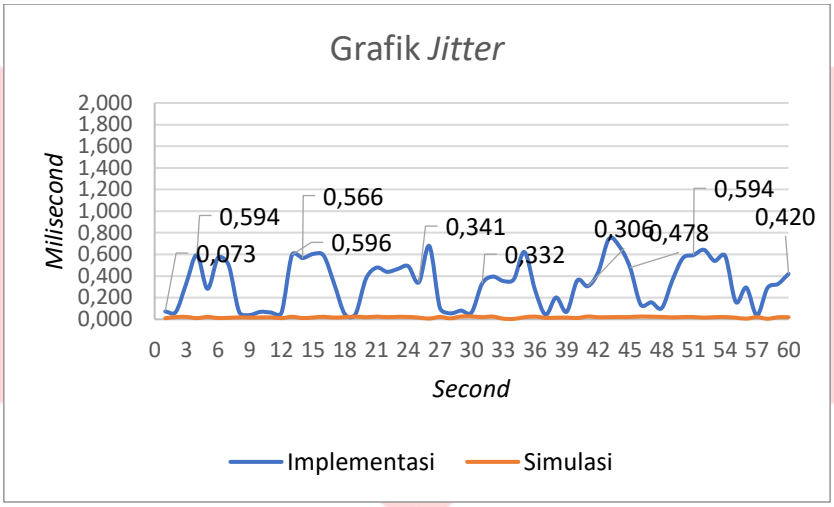
Pengukuran *delay* ini dilakukan dengan menggunakan protokol *ping* dengan beban sebesar 64 *bytes* dalam waktu 60 detik, sehingga mendapatkan nilai rata-rata *delay* pada simulasi sebesar 47,43 ms dan nilai rata-rata *delay* pada implementasi sebesar 72,95 ms.



Gambar 10 Grafik Delay 64 Bytes

4.3.3 Jitter

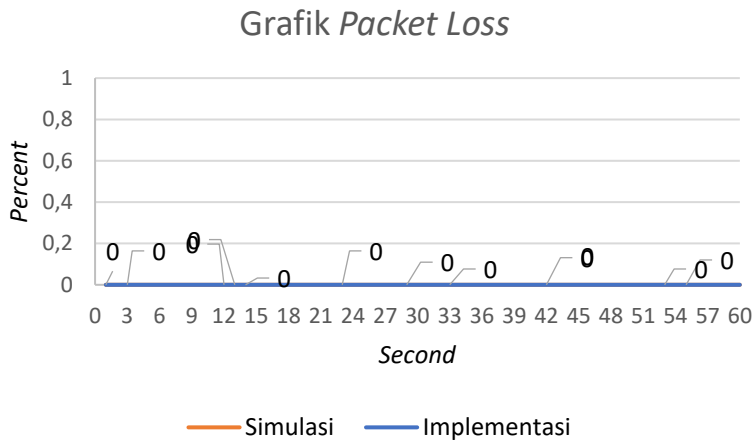
Pengukuran *jitter* dilakukan dalam waktu 60 detik sehingga mendapatkan nilai rata-rata *jitter* pada simulasi yaitu 0,018 ms, dan nilai rata-rata *jitter* pada implementasi yaitu 0,332 ms.



Gambar 11 Grafik Jitter

4.3.4 Packet Loss

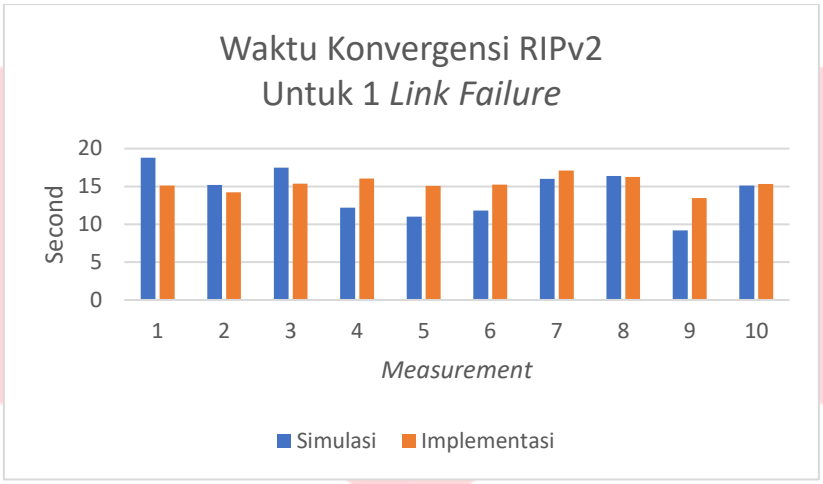
Pengukuran *Packet Loss* dilakukan dalam waktu 60 detik sehingga mendapatkan nilai rata-rata *packet loss* pada simulasi yaitu 0% dan begitu pula nilai rata-rata pada implementasi yaitu 0%.



Gambar 12 Grafik Packet Loss

4.3.5 Convergence Time

Pada proyek akhir ini dilakukan pengukuran *convergence time* yang bertujuan untuk mengetaui waktu yang dibutuhkan oleh *router* untuk mencapai keadaan konvergensi. Berikut ini merupakan tabel *convergence time* pada jaringan komputer berbasis SDN pada metode simulasi dan implementasi dengan melakukan 10 kali percobaan:



Gambar 13 Grafik Convergence Time RIPv2 Untuk 1 Link Failure

5 Kesimpulan

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

1. Protokol routing RIPv2 dapat disimulasikan dan diimplementasikan pada jaringan berbasis SDN dengan adanya komponen pendukung arsitektur utama yaitu *RouteFlow*.
2. Protokol routing RIPv2 memilih jalur yang memiliki hop terkecil pada saat mengirimkan paket. Dimana pada proyek akhir ini sudah sesuai dengan teori tersebut.
3. Simulasi dan implementasi jaringan SDN telah selesai dibuat dan diuji performansinya menggunakan parameter throughput, delay, jitter, dan packet loss dengan hasil berikut:
   a. Simulasi:
      - Rata-rata *throughput* dari hasil Proyek Akhir ini adalah 100,21 Mbps
      - Rata-rata *delay* dari hasil Proyek Akhir ini adalah 47,43 ms
      - Rata-rata *jitter* dari hasil Proyek Akhir ini adalah 0,018 ms
      - Rata-rata *packet loss* dari hasil Proyek Akhir ini adalah 0%
      - Rata-rata *convergence time* dari hasil Proyek Akhir ini adalah 14,31 detik
   b. Implementasi
      - Rata-rata *throughput* dari hasil Proyek Akhir ini adalah 99,94 Mbps
      - Rata-rata *delay* dari hasil Proyek Akhir ini adalah 72,95 ms
      - Rata-rata *jitter* dari hasil Proyek Akhir ini adalah 0,332 ms
      - Rata-rata *packet loss* dari hasil Proyek Akhir ini adalah 0,0%
      - Rata-rata *convergence time* dari hasil Proyek Akhir ini adalah 15,32 detik

Daftar Pustaka

[1] Naqvi, H. A. 2015. MPLS in SNHx - a Networking Application using RYU SDN Framework.
[2] Open Networking Foundation, [online] https://www.opennetworking.org/sdn- resources/openflow. Diakses pada tanggal 20 September 2016.
[3] P. Goransson and C. Black. 2014. “Software Defined Network : A Comprehensive Approach”. New York : Morgan Kaufmann.
[4] Quora, 2014. http://www.quora.com/How-do-Bellman-Ford-Algorithm-and-Dijkstras-algorithm-work-differently-in-routing-protocols#, Tanggal akses : 20 September 2016
[5] RouteFlow, [online] http://cpqd.github.io/RouteFlow/. Diakses pada tanggal 15 November 2016.
[6] RouteFlow, [online] https://github.com/CPqD/RouteFlow. Diakses pada tanggal 15 November 2016.
[7] SDX Central, [online] https://www.sdxcentral.com/sdn/definitions/sdn-controllers/. Diakses pada tanggal 15 November 2016.
[8] Sitohang, Dwinson. 2015. Implementasi Load-Balancing dengan Metode Round Robin dalam software Defined Networking (SDN) Menggunakan Controller POX. Medan : Skripsi Universitas Sumatera Utara
[9] Cisco Networking Academy. 2009. “CCNA Exploration Course Booklet : Routing Protocols and Concepts Version 4.0,”. California : Cisco Press