

**ANALISIS PERBANDINGAN ANTARA JARINGAN KONVENSIONAL
DAN JARINGAN SOFTWARE DEFINED NETWORK YANG BERBASIS
OPENFLOW**

SKRIPSI



Oleh :

MUHAMMAD YAUMIL AGUS AWAL

H13114526

PROGRAM STUDI ILMU KOMPUTER DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

NOVEMBER 2018

**ANALISIS PERBANDINGAN ANTARA JARINGAN KONVENSIONAL
DAN JARINGAN SOFTWARE DEFINED NETWORK YANG BERBASIS
OPENFLOW**

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
pada Program Studi Ilmu Komputer Departemen Matematika Fakultas
Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Makassar

MUHAMMAD YAUMIL AGUS AWAL

H13114526

PROGRAM STUDI ILMU KOMPUTER DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

NOVEMBER 2018

LEMBAR PERYATAAN KEOTENTIKAN

Saya yang bertanda tangan di bawah ini menyatakan dengan sungguh-sungguh bahwa skripsi yang saya buat dengan judul:

**ANALISIS PERBANDINGAN ANTARA JARINGAN
KONVENSIONAL DAN JARINGAN SOFTWARE DEFINED
NETWORK YANG BERBASIS OPENFLOW**

adalah benar hasil karya saya sendiri bukan hasil plagiat dan belum pernah dipublikasikan dalam bentuk apapun

Makassar, 23 November 2018

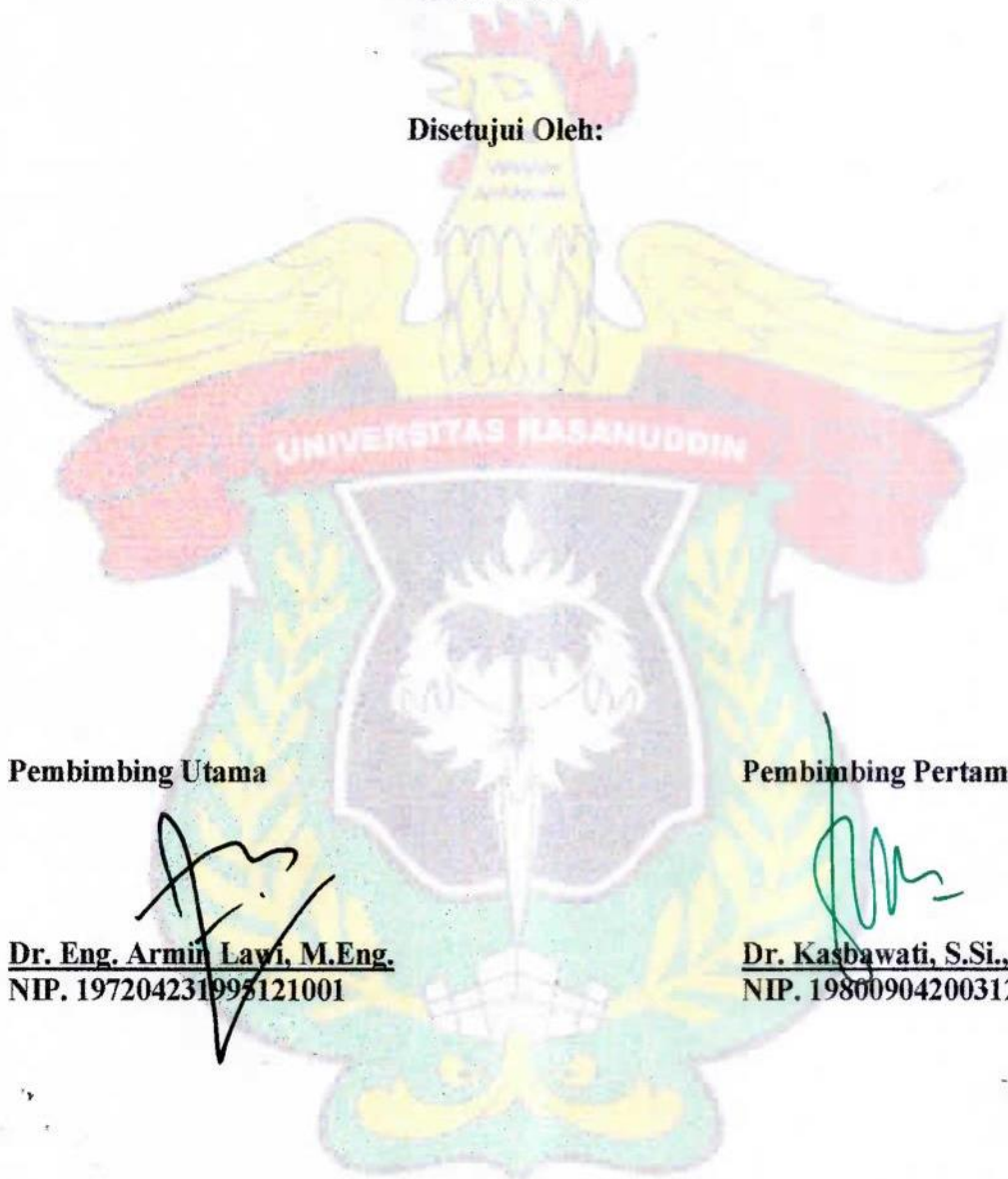


Muhammad Yaumil Agus Awal

NIM. H 131 14 526

**ANALISIS PERBANDINGAN ANTARA JARINGAN KONVENSIONAL
DAN JARINGAN SOFTWARE DEFINED NETWORK YANG BERBASIS
OPENFLOW**

Disetujui Oleh:



Pembimbing Utama

Dr. Eng. Armin Lawi, M.Eng.
NIP. 197204231993121001

Pembimbing Pertama

Dr. Kasbawati, S.Si., M.Si.
NIP. 198009042003122001

Pada Tanggal : 23 November 2018

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Muhammad Yaumil Agus Awal
NIM : H13114526
Program Studi : Ilmu Komputer
Judul Skripsi : Analisis Perbandingan Antara Jaringan
Konvensional dan Jaringan Software Defined
Network yang Berbasis Openflow

**Telah berhasil dipertahankan dihadapan dewan penguji dan diterima
sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar
Sarjana Komputer pada Program Studi Ilmu Komputer Fakultas
Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.**

DEWAN PENGUJI

Tanda Tangan

1. Ketua : Dr. Diaraya, M.Ak
2. Sekretaris : Dr. Loeky Haryanto, MS., M.Sc
3. Anggota : Dr. Eng. Armin Lawi, M.Eng
4. Anggota : Dr. Kasbawati, S.Si., M.Si.

(.....)

(.....)

(.....)

(.....)

Ditetapkan di : Makassar

Tanggal : 23 November 2018

KATA PENGANTAR

Segala puji penulis panjatkan kehadirat Allah SWT, yang senantiasa melimpahkan rahmat dan karunia-Nya. Shalawat dan salam senantiasa penulis kirimkan kepada Baginda Rasulullah SAW, yang telah mengajarkan kebenaran dan membimbing umat – umatnya ke arah yang benar. Rasa syukur yang tak terkira atas segala nikmat yang telah diberikan terutama nikmat kesehatan, kesempatan dan kemudahan yang dikaruniakan kepada penulis dalam menyelesaikan tugas akhir yang berjudul **Analisis Perbandingan Antara Jaringan Konvensional dan Jaringan Software Defined Network yang Berbasis Openflow** .

Penyusunan tugas akhir ini tentunya tidak lepas dari bantuan berbagai pihak baik moril maupun materil. Oleh karena itu, penulis menyampaikan ucapan terima kasih yang tulus dan penghargaan yang tak terhingga terhadap ayah dan ibu saya tercinta **Drs. Ishak, M.Pd.** dan **Dra. Hj. Rahmi Alwi** atas segala dukungan, doa, restu, nasehat dan motivasi yang tak henti-hentinya mereka berikan kepada penulis untuk menggapai cita-cita, serta untuk seluruh keluarga besar, terima kasih atas doanya.

Penghargaan yang tulus dan ucapan terima kasih dengan penuh keikhlasan juga penulis ucapkan kepada :

1. Ibu **Rektor Universitas Hasanuddin** beserta jajarannya, Bapak **Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam beserta jajarannya**, dan seluruh pihak birokrasi atas pengetahuan dan kemudahan-kemudahan yang diberikan, baik dalam bidang akademik maupun bidang kemahasiswaan.
2. Bapak Prof. **Dr. Amir Kamal Amir, M.Sc.**, selaku Ketua Jurusan Matematika, dan Bapak **Dr. Amran, S.Si., M.Si.**, selaku Sekretaris Jurusan, serta Bapak **Dr. Diaraya, M.Ak.**, selaku Kepala Program Studi Ilmu Komputer, selaku ketua penguji dan juga penasehat akademik saya (PA) yang telah memberikan banyak bantuan selama penulis menjalani pendidikan dan yang selalu memberikan motivasi dan saran kepada penulis. Terima kasih juga untuk

segenap jajaran Pegawai Akademik Jurusan Matematika atas bantuannya dalam pengurusan akademik selama ini.

3. Bapak **Dr. Eng. Armin Lawi, M.Eng**, selaku pembimbing utama, atas segala ilmu, nasehat, dan kesabaran dalam membimbing penulis serta meluangkan waktu di sela-sela rutinitas yang begitu padat hingga skripsi ini dirampungkan, dan Ibu **Dr. Kasbawati, S.Si., M.Si.** selaku pembimbing pertama, untuk segala ilmu, nasehat, dan kesabaran dalam membimbing dan mengarahkan penulis, serta bersedia meluangkan waktunya untuk mendampingi penulis sejak awal penyusunan hingga akhir perampungan skripsi ini.
4. Bapak **Dr. Loeky Haryanto, MS. M.Sc.**, selaku sekretaris tim penguji atas segala ilmu, nasehat, saran dan motivasi yang diberikan kepada penulis mulai dari perkuliahan hingga penyusunan skripsi ini.
5. Saudara-saudara **Keluarga Cemara (Dewi Ayu Hartina, Yolanda Gabyriela Ferandji, Nur Nilamyani, Nurul Hardiyanti, dan Fuad Fadhil Azzar)** yang telah menemani penulis selama perkuliahan, yang telah meluangkan waktu dan berbagi suka-duka dan kebersamaan selama menuntut ilmu.
6. Teman-teman seperjuangan **Ilmu Komputer 2014 (Nadya, Jo, Luki, Dila, Danti, Firda, Yayu, Nuhi, Aspar, Syam, Tio, Hajar, Darul, dll)** yang membantu dan memberi support penulis dalam penyusunan skripsi ini.
7. Adik-Adik **Ilmu Komputer 2015, 2016, 2017, dan 2018**
8. Rekan-rekan **KKN Internasional Jepang UNHAS Gelombang 96 (Fuad, Jo, Tio, Aspar, Iqbal, Fadhil, Mamet, Syam, Ola, Iyam, Yayu, Firda, Finka, Mine', Dani, Fifi, dan Dhita)** yang telah menjadi keluarga baru selama KKN dan menjadikan KKN sebagai momen yang membahagiakan.
9. Semua pihak yang telah banyak berpartisipasi, baik secara langsung maupun tidak langsung dalam penyusunan skripsi ini yang tak sempat penulis sebutkan satu per satu.

Semoga segala bantuan yang dengan tulus ditujukan kepada penulis mendapatkan balasan dari Allah SWT. Mudah-mudahan tulisan ini memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Semoga segala bantuan yang dengan tulus ditujukan kepada penulis mendapatkan balasan dari Allah SWT. Mudah-mudahan tulisan ini memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Makassar, 23 September 2018

A handwritten signature in black ink, appearing to read 'Muhammad Yaumil Agus Awal', written in a cursive style.

Muhammad Yaumil Agus Awal

**PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK
KEPENTINGAN AKADEMIK**

Sebagai civitas akademik Universitas Hasanuddin saya yang bertanda tangan di bawah ini:

Nama : Muhammad Yaumil Agus Awal

NIM : H 131 14 526

Program Studi : Ilmu Komputer

Departemen : Matematika

Fakultas : Matematika dan Ilmu Pengetahuan Alam

Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Prediktor Royalti Non-eksklusif (*Non-exclusive Royalty- Free Right*)** atas tugas akhir saya yang berjudul:

**“Analisis Perbandingan Antara Jaringan Konvensional dan Jaringan
Software Defined Network yang Berbasis Openflow”**

Beserta perangkat yang ada (jika diperlukan). Terkait dengan hal di atas, maka pihak universitas berhak menyimpan, mengalih-media/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di Makassar pada tanggal, 23 November 2018

Yang menyatakan,



Muhammad Yaumil Agus Awal

ABSTRAK

Software-Defined Network (SDN) menawarkan kemudahan dalam mengatur perangkat jaringan dengan hanya mengatur perangkat lunak kontrol pesawat sehingga dapat mengurangi kompleksitas konfigurasi jaringan. Namun, kemudahan dalam mengatur yang ditawarkan tidak selalu diikuti dengan peningkatan kinerja jaringan bila dibandingkan dengan arsitektur jaringan konvensional. Penelitian ini akan melakukan analisis perbandingan kinerja antara jaringan dengan arsitektur SDN (OpenFlow) dan arsitektur konvensional. Pengujian kinerja dilakukan berdasarkan simulasi jaringan dalam satu subnet menggunakan emulator Mininet.

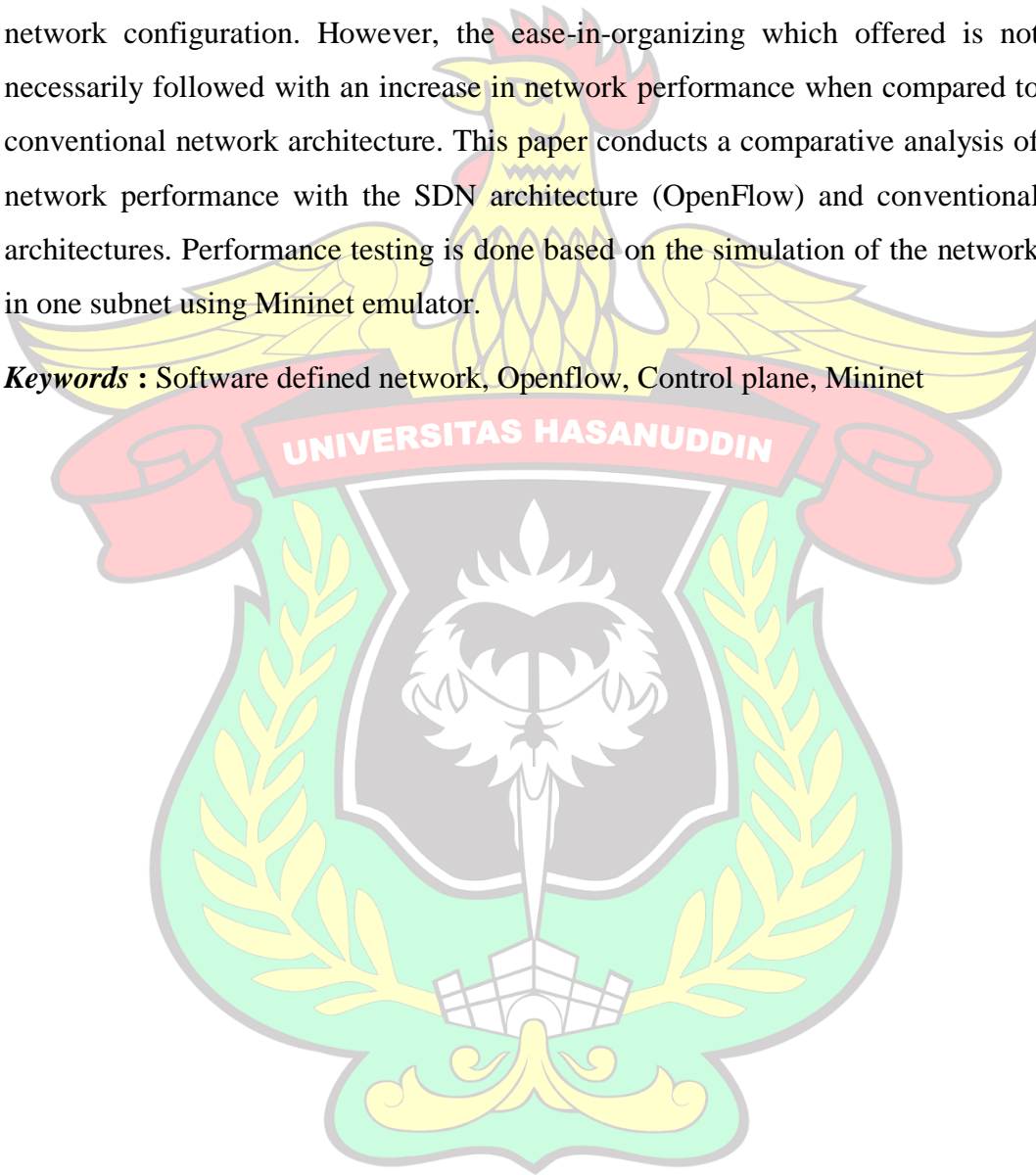
Kata Kunci : Software defined network, Openflow, Control plane, Mininet



ABSTRACT

Software-Defined Network (SDN) offers ease in organizing the network device with only set the control plane software so that it able to reduce the complexity of network configuration. However, the ease-in-organizing which offered is not necessarily followed with an increase in network performance when compared to conventional network architecture. This paper conducts a comparative analysis of network performance with the SDN architecture (OpenFlow) and conventional architectures. Performance testing is done based on the simulation of the network in one subnet using Mininet emulator.

Keywords : Software defined network, Openflow, Control plane, Mininet



DAFTAR ISI

HALAMAN JUDUL.....	ii
LEMBAR PERNYATAAN KEOTENTIKAN.....	iii
HALAMAN PENGESAHAN.....	iv
KATA PENGANTAR.....	v
DAFTAR ISI.....	xii
DAFTAR GAMBAR	xv
DAFTAR TABEL.....	xvi
BAB I.....	1
PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	2
1.4 Tujuan Penelitian.....	2
1.5 Manfaat Penelitian.....	2
1.6 Organisasi Skripsi.....	3
BAB II.....	4
TINJAUAN PUSTAKA.....	4
2.1 Landasan Teori	4
2.1.1 Jaringan komputer.....	4
2.1.2 <i>Software defined network</i>	4
2.1.3 <i>Openflow</i>	4
2.1.4 <i>Quality of Service (QoS)</i>	6
2.1.5 <i>Throughput</i>	7
2.1.6 <i>Latency</i>	7
2.1.7 <i>Jitter</i>	8
2.1.8 <i>Packet loss</i>	9
2.1.9 Perbedaan arsitektur jaringan SDN dan konvensional	9
2.1.10 Mininet.....	10
2.1.11 <i>Floodlight Controller</i>	11
2.2 Penelitian Terkait.....	11

2.3 Kerangka Konseptual	16
BAB III.....	17
METODE PENELITIAN.....	17
3.1 Waktu dan Tempat	17
3.2 Tahapan Penelitian	17
3.3 Sumber Data	18
3.4 Instrumen Penelitian.....	18
BAB IV	19
HASIL DAN PEMBAHASAN.....	19
4.1 Perancangan Sistem.....	19
4.1.1 Rancangan Topologi.....	19
4.2 Tahap instalasi dan Konfiurasi	22
4.2.1 <i>Controller</i> Floodlight.....	22
4.2.2 Emulator Mininet.....	22
4.3 Skenario 1 (Uji Latency)	23
4.3.1 Pengujian Latency Topologi SDN dan Konvensional Sederhana	23
4.3.2 Analisa	24
4.3.3 Pengujian Latency Topologi SDN dan Konvensional Rumit.....	26
4.3.4 Analisa	26
4.4 Uji <i>Throughput</i>	28
4.4.1 Pengujian <i>Throughput</i> Topologi SDN dan Konvensional Sederhana ..	28
4.4.2 Analisa	29
4.4.3 Pengujian <i>Throughput</i> Topologi SDN dan Konvensional Rumit.....	31
4.4.4 Analisa	31
4.5 Uji <i>Jitter</i> dan <i>Packet loss</i>	33
4.5.1 Pengujian <i>Jitter</i> dan <i>Packet Loss</i> Topologi SDN dan Konvensional Sederhana.....	33
4.5.2 Analisa	34
4.5.3 Pengujian <i>Jitter dan Packet Loss</i> Topologi SDN dan Konvensional Rumit	36
4.5.4 Analisa	36
BAB V.....	40
KESIMPULAN DAN SARAN.....	40

5.1 Kesimpulan.....	40
5.2 Saran	41
DAFTAR PUSTAKA	42

DAFTAR GAMBAR

Gambar 3 Kerangka Konseptual	16
Gambar 4 Tahapan Penelitian	17
Gambar 5 Topologi Jaringan SDN 1	19
Gambar 6 Topologi Jaringan Konvensional 1	20
Gambar 7 Topologi Jaringan SDN 2	20
Gambar 8 Topologi Jaringan SDN 3	21
Gambar 9 Topologi Jaringan Konvensional 2	21
Gambar 10 Uji Latency Arsitektur SDN	24
Gambar 11 Uji Latency Arsitektur Tradisional	24
Gambar 12 Hasil Uji Latency	25
Gambar 13 Hasil Uji Latency	27
Gambar 14 Uji Throughput Arsitektur SDN	29
Gambar 15 Uji Throughput Arsitektur Konvensional	29
Gambar 16 Hasil Uji Throughput	30
Gambar 17 Hasil Uji Throughput	32
Gambar 18 Uji Jitter dan Packet loss Arsitektur SDN	33
Gambar 19 Uji Jitter dan Packet loss Arsitektur Konvensional	34
Gambar 20 Hasil Uji Jitter	35
Gambar 21 Hasil Uji Packet loss	35
Gambar 22 Hasil Uji Jitter	37
Gambar 23 Hasil Uji Packet loss	38

DAFTAR TABEL

Tabel 1 Throughput.....	7
Tabel 2 Latency.....	8
Tabel 3 Jitter.....	8
Tabel 4 Packet loss.....	9
Tabel 5 Hasil Uji Latency	25
Tabel 6 Hasil Uji Latency	26
Tabel 7 Hasil Uji Throughput	30
Tabel 8 Hasil Uji Throughput	31
Tabel 9 Hasil Uji Jitter	34
Tabel 10 Hasil Uji Packet loss	35
Tabel 11 Hasil Uji Jitter	37
Tabel 12 Hasil Uji Packet loss	38

BAB I

PENDAHULUAN

1.1 Latar Belakang

Software-Define Networking (SDN) adalah sebuah pendekatan baru untuk merancang, membangun dan mengelola jaringan komputer dengan memisahkan *control plane* dan *data plane* berbeda dengan jaringan komputer saat ini yang merupakan jaringan konvensional dimana *control plane* dan *data plane* dipisahkan. Konsep utama SDN adalah sentralisasi jaringan dimana, dimana semua pengaturan berada pada *control plane*, terutama pada *data plane platform* yang menggunakan lapisan abstraksi untuk semua perangkat jaringan pada arsitektur SDN. Sehingga mudah dalam mengatur perangkat jaringan dengan hanya mengatur *software control plane* dan mampu mengurangi kompleksitas pada konfigurasi jaringan. Protokol yang paling menonjol pada SDN yaitu Openflow [1]

Openflow menjadi standar pertama sebagai interface dalam komunikasi antara *forwarding plane* dan *control plane* dalam arsitektur SDN dimana openflow dapat mengkonfigurasi perangkat jaringan dan memilih jalur yang optimal untuk aplikasi trafik dan *Controller* yang memungkinkan perangkat lunak untuk dijalankan pada berbagai jenis hardware. [2]

Namun dari kelebihan yang ditawarkan oleh jaringan SDN belum tentu diikuti oleh peningkatan performa jika dibandingkan dengan jaringan konvensional. Maka dari itu peneliti akan menganalisis pengaruh SDN terhadap performa jaringan dengan parameter yaitu *latency*, *throughput*, *Jitter*, dan *Packet loss* dan hasil akhirnya mengetahui perbandingan antara jaringan SDN dan jaringan konvensional. Jadi berdasarkan uraian masalah yang muncul maka peneliti mengambil judul “**Analisis Perbandingan antara Jaringan Konvensional dan Jaringan Software Defined Network yang Berbasis Openflow**”

1.2 Rumusan Masalah

Adapun yang menjadi rumusan masalah dalam penelitian tugas akhir ini adalah :

1. Bagaimana mensimulasikan SDN di mininet ?
2. Bagaimana menganalisis perbandingan performansi kinerja jaringan SDN dan konvensional ?

1.3 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah:

1. Sistem akan disimulasikan menggunakan mininet dengan *Floodlight* sebagai *Controller*.
2. Tidak mensimulasikan *background traffic* dan *multiple flow*.
3. Tidak memperhatikan performansi *Controller* yang digunakan.
4. Menggunakan *tools* ping dan iperf untuk men-*generate* aliran *traffic* pada jaringan.
5. Parameter yang akan diambil adalah *Throughput*, *latency*, *Jitter* dan *Packet loss*.

1.4 Tujuan Penelitian

Adapun tujuan yang ingin dicapai pada penelitian tugas akhir ini adalah:

1. Mensimulasikan SDN di mininet.
2. Menganalisis dan mengetahui perbandingan performansi antara jaringan SDN dan konvensional.

1.5 Manfaat Penelitian

Hasil penelitian ini diharapkan dapat berguna untuk mengetahui perbandingan antara jaringan SDN dan konvensional

1.6 Organisasi Skripsi

Sistematika penulisan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Bab ini membahas mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan manfaat penulisan, serta organisasi skripsi.

BAB II Tinjauan Pustaka

Bab ini membahas mengenai landasan teori dan konsep dasar yang mendasari pokok permasalahan dalam tulisan ini.

BAB III Metodologi Penelitian

Bab ini berisi waktu dan tempat penelitian, tahapan penelitian, rancangan sistem, sumber data, instrumen penelitian dan jadwal penelitian.

BAB IV Hasil dan Pembahasan

Bab ini berisi hasil dan pembahasan dari pokok permasalahan yang diangkat dalam tulisan skripsi ini yaitu perbandingan performansi jaringan SDN dan Konvensional

BAB V Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil dan pembahasan yang telah diperoleh dan dibahas pada bab II, serta berisi saran yang diharapkan dapat membantu membangun skripsi ini menjadi lebih baik.

BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Jaringan komputer

Menurut Jafar Noor Yudianto menyatakan bahwa Jaringan komputer ialah suatu sistem yang terdiri atas sebuah komputer-komputer yang didesain untuk bisa berbagi sumber daya (printer, CPU), berkomunikasi (surel, pesan instan), dan bisa mengakses informasi [3]

2.1.2 *Software defined network*

Software-defined *networking* (SDN) pendekatan jaringan komputer yang memungkinkan administrator jaringan memprogram, mengendalikan, mengubah, dan mengelola perilaku jaringan secara dinamis melalui antarmuka terbuka dan abstraksi fungsi tingkat rendah. SDN dimaksudkan untuk mengatasi kenyataan bahwa arsitektur statis jaringan tradisional tidak mendukung kebutuhan komputasi dan penyimpanan dinamis dan terukur dari lingkungan komputasi modern seperti pusat data. Hal ini dilakukan dengan memisahkan atau melepaskan sistem yang membuat keputusan tentang kemana lalu lintas dikirim (pengendali SDN, atau bidang kontrol) dari sistem yang mendasari lalu lintas ke tempat tujuan yang dipilih (bidang data). [4]

SDN umumnya terkait dengan protokol *Openflow* (untuk komunikasi jarak jauh dengan elemen pesawat jaringan untuk menentukan jalur paket jaringan di seluruh *switch* jaringan) sejak kemunculannya yang kedua di tahun 2011. [5]

2.1.3 *Openflow*

Openflow memungkinkan pengontrol jaringan untuk menentukan jalur paket jaringan di seluruh jaringan *switch*. Pengontrolnya berbeda dari *switch*. Pemisahan kontrol dari forwarding ini memungkinkan pengelolaan lalu lintas yang lebih canggih daripada yang layak dengan menggunakan daftar kontrol akses (Access Control List - ACLs) dan protokol *routing*. Selain itu, *openflow*

memungkinkan *switch* dari vendor yang berbeda - seringkali masing-masing dengan antarmuka proprietary dan bahasa scripting mereka sendiri - untuk dikelola dari jarak jauh dengan menggunakan satu protokol terbuka tunggal. Penemu protokol tersebut menganggap *openflow* sebagai enabler dari jaringan yang didefinisikan perangkat lunak (SDN). [4]

Protokol *Openflow*

Openflow adalah protokol yang berfungsi dalam menjembatani komunikasi antara *Data plane* Layer (router, *openflow switch*) dengan *Control plane* Layer (*Controller*) pada arsitektur SDN. *Openflow* memperbolehkan akses langsung dan manipulasi *data plane* (router, *openflow switch*). *Openflow* menerapkan konsep flow dalam mengidentifikasi trafik jaringan. *Openflow* merupakan kunci pada SDN dalam memanipulasi langsung *data plane* pada [6]

perangkat jaringan

Keuntungan *Openflow*

Protokol *openflow* memiliki beberapa keuntungan diantaranya sebagai berikut :

- a. Dapat mengendalikan jaringan secara terpusat pada berbagai vendor. Penggunaan *Controller* berbasis software pada SDN dapat mengendalikan berbagai *network devices* seperti *openflow switch*, router. Hal tersebut dapat diterapkan pada berbagai vendor, sehingga dapat mempercepat perkembangan jaringan, konfigurasi dan update perangkat secara otomatis.
- b. Dapat mengurangi kesalahan secara otomatis. *Openflow* berbasis SDN menawarkan jaringan dan manajemen framework yang otomatis dan fleksibel sehingga membuat perkembangan *tools* menjadi otomatis. *Tools* yang otomatis akan mengurangi biaya operasional, mengurangi human error, dan mendukung kemunculan IT as a *Service*. Dengan adanya SDN, aplikasi berbasis cloud dapat mengatur sistem yang cerdas, mengurangi biaya operasional sehingga meningkatkan bisnis yang cerdas.

- c. Dapat meningkatkan kehandalan dan keamanan jaringan. SDN menerapkan konfigurasi dan kebijakan tingkat tinggi, yang diterjemahkan melalui *openflow*. *Openflow* berbasis arsitektur SDN membutuhkan konfigurasi perangkat jaringan setiap waktu, kebijakan layanan dapat berubah, sehingga dapat mengurangi kesalahan konfigurasi jaringan dan kebijakan yang tidak konsisten. [7]

2.1.4 *Quality of Service (QoS)*

QoS merupakan teknik untuk mengelola *bandwidth*, *delay*, *Jitter*, dan *Packet loss* untuk aliran dalam jaringan. Tujuan dari mekanisme QoS adalah mempengaruhi setidaknya satu dari empat parameter dasar QoS yang telah ditentukan. Sedangkan menurut Ningsih dkk, *Quality of Service* adalah kemampuan sebuah jaringan untuk menyediakan layanan yang lebih baik lagi bagi layanan trafik yang melewatinya. QoS merupakan sebuah sistem arsitektur end-to-end dan bukan merupakan sebuah fitur yang dimiliki oleh jaringan. Tujuan dari QoS adalah untuk memenuhi kebutuhan-kebutuhan layanan yang berbeda, yang menggunakan infrastruktur yang sama.

Fungsi-fungsi QoS dijelaskan sebagai berikut:

1. Pengkelasan paket untuk menyediakan pelayanan yang berbeda-beda untuk kelas paket yang berbeda-beda.
2. Penanganan kemacetan (*congestion*) untuk memenuhi dan menangani kebutuhan layanan yang berbeda-beda.
3. Pengendalian lalu lintas paket untuk membatasi dan mengendalikan pengiriman paket-paket data.
4. Pensinyalan untuk mengendalikan fungsi-fungsi perangkat yang mendukung komunikasi di dalam jaringan IP. [8]

2.1.5 *Throughput*

Throughput yaitu kecepatan (rate) transfer data efektif, yang diukur dalam bps. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada destination selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. *Throughput* adalah *bandwidth* aktual yang terukur pada suatu ukuran waktu tertentu dalam suatu hari menggunakan rute jaringan yang spesifik ketika sedang men-download suatu file. Menurut ETSI besarnya *Throughput* dapat diklasifikasikan sebagai berikut:

Tabel 1 *Throughput*

Kategori <i>Throughput</i>	<i>Throughput</i>
Sangat Bagus	100%
Bagus	75%
Sedang	50%
Jelek	<25%

Nilai *Throughput* dinyatakan dalam rumus berikut:

$$Throughput = \frac{\text{Jumlah data diterima}}{\text{Waktu pengiriman}}$$

Throughput (Sumber: ETSI)

2.1.6 *Latency*

Delay (latency) adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama [21]. Menurut versi ETSI [14] besarnya *latency* dapat diklasifikasikan sebagai berikut :

Tabel 2 Latency

Kategori Latency	Besar Delay
Sangat Bagus	<150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Jelek	>450 ms

$$Delay\ rata - rata = \frac{Total\ delay}{Total\ paket\ yang\ diterima}$$

2.1.7 Jitter

Jitter didefinisikan sebagai variasi dari *delay* atau variasi waktu kedatangan paket. Banyak hal yang dapat menyebabkan *Jitter*, diantaranya adalah peningkatan trafik secara tiba tiba sehingga menyebabkan penyempitan *bandwidth* dan menimbulkan antrian. Selain itu, kecepatan terima dan kirim paket dari setiap node juga dapat menyebabkan *Jitter*. Menurut versi ETSI terdapat empat kategori penurunan performansi jaringan berdasarkan nilai peak *Jitter*, yaitu:

Tabel 3 *Jitter*

Kategori Degradasi	Peak <i>Jitter</i>
Sangat Bagus	0 ms
Bagus	0 s/d 75 ms
Sedang	75 s/d 125 ms
Jelek	125 s/d 225 ms

Nilai *Jitter* dapat dinyatakan dalam rumus berikut:

$$Jitter = \frac{Total\ variasi\ delay}{Total\ paket\ yang\ diterima}$$

Jitter (Sumber: ETSI)

2.1.8 *Packet loss*

Merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan . Di dalam implementasi jaringan IP, nilai *Packet loss* ini diharapkan mempunyai nilai yang minimum. Nilai *Packet loss* menurut versi ETSI adalah sebagai berikut:

Tabel 4 *Packet loss*

Kategori Degrasi	<i>Packet loss</i>
Sangat Bagus	0%
Bagus	3%
Sedang	15%
Jelek	25%

Nilai *Packet loss* dapat dinyatakan dalam rumus berikut:

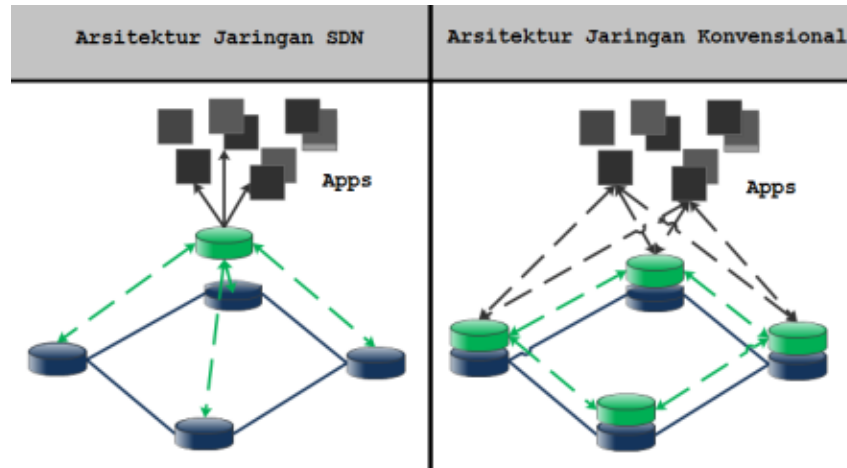
$$Packet\ Loss = \frac{\text{Paket dikirim} - \text{Paket diterima}}{\text{Total Paket yang diterima}}$$

Packet loss (Sumber: ETSI)

2.1.9 Perbedaan arsitektur jaringan SDN dan konvensional

Pada Gambar 1 mengilustrasikan perbedaan antara arsitektur jaringan SDN dan konvensional. Pada gambar di sisi kiri, menunjukkan arsitektur jaringan SDN dimana *control plane* dan *data plane* dipisahkan. Sedangkan pada gambar di sisi kanan, menunjukkan arsitektur jaringan konvensional dimana *control plane* dan *data plane* berada di tempat yang sama. Ini menunjukkan bahwa pada jaringan SDN, *control plane* menggunakan struktur tersentralisasi. Sedangkan pada jaringan konvensional, *control plane* menggunakan struktur terdistribusi.

Selain itu, perbedaan antara jaringan arsitektur SDN dan konvensional (TCP/IP) terletak pada arsitektur layer yang digunakan. Pada Gambar dibawah



Gambar 1 Perbedaan Arsitektur Jaringan SDN dan Jaringan Konvensional

terlihat bahwa jaringan dengan arsitektur SDN menggunakan 3 layer yang terdiri dari Application, Control dan Physical. Sedangkan pada arsitektur jaringan konvensional menggunakan 4 layer yang terdiri dari Application, Transport, Internet dan *Network*.

SDN	TCP/IP
Application	Application
Control Layer	Transport
	Internet
	Network Access
Physical	

Gambar 2 Perbedaan Layer Jaringan SDN dan Layer Jaringan Konvensional [22]

2.1.10 Mininet

Mininet adalah emulator jaringan yang menciptakan jaringan virtual host, *switch*, *Controller*, dan link. Mininet merupakan emulator jaringan yang bersifat open source yang mendukung protokol *openflow* untuk arsitektur SDN. Tampilan pada mininet berupa *Command Language Interpreter* (CLI), hal tersebut memudahkan dalam melakukan skenario jaringan. Mininet merupakan

emulator yang sudah teruji untuk mengimplementasikan konsep SDN. Mininet dapat dijalankan dalam sebuah laptop yang menggunakan sistem operasi berbasis Linux. Mininet memiliki kekurangan yaitu mininet tidak dapat berjalan pada sistem operasi non-linux yang tidak mendukung *switch openflow*. [1]

2.1.11 Floodlight Controller

Floodlight merupakan kontroler *openflow* kelas enterprise dengan lisensi Apache dan berbasis Java. Hal ini didukung oleh komunitas pengembang termasuk sejumlah insinyur dari *Big Switch Networks*. *Floodlight* dirancang untuk bekerja dengan meningkatnya jumlah *switch*, router, *switch* virtual, dan jalur akses yang mendukung standar *OpenFlow*. Beberapa fitur yang ditawarkan *Floodlight*:

- a. Menawarkan sistem modul beban yang membuatnya sederhana untuk dikembangkan.
- b. Mudah untuk mengatur dengan dependensi minimal
- c. Mendukung berbagai *switch openflow* virtual dan fisik.
- d. Dapat menangani *openflow* dan non-*openflow* jaringan campuran
- e. Dirancang untuk menjadi kinerja tinggi - adalah inti dari produk komersial dari *Big Switch Networks*.
- f. Dukungan untuk OpenStack (link) platform orkestrasi cloud. [9]

2.2 Penelitian Terkait

Pada sub bab ini dijelaskan beberapa penelitian yang terkait dengan judul dari penelitian penulis

- a. **Uji Performansi Algoritma Floyd-Warshall pada Jaringan *Software defined network* (SDN) Performance Analysis of Floyd-Warshall Algorithm on *Software defined network* (SDN)** (hsan Aris Saputra*, R. Rumani M., dan Sofia Naning Hertiana)

Penentuan rute pada sebuah jaringan *Software defined network* (SDN) merupakan salah satu contoh topik yang menarik untuk diteliti. Algoritma penentuan rute terpendek pada jaringan SDN sangatlah menentukan apakah jaringan SDN yang dibangun dengan algoritma tersebut sudah optimal. Salah satu algoritma penentuan rute terpendek yaitu algoritma Floyd-Warshall,

yang akan diuji coba dan dianalisis apakah sudah termasuk algoritma yang optimal pada jaringan SDN dengan membandingkan dengan standarisasi yang ada. Pengujian akan dilakukan dengan mengirimkan paket data, VoIP dan video dengan melihat overhead traffic dan QoS (*delay* dan *Packet loss*). Algoritma Floyd-Warshall akan digunakan pada pengontrol Ryu dan menggunakan Mininet sebagai emulator jaringan dengan topologi berbasis Abiline. Hasil simulasi dan pengujian algoritma Floyd-Warshall sebagai penentuan jalur terbaik dalam jaringan SDN, mendapatkan hasil yang memenuhi standarisasi. Nilai dari QoS yang didapat untuk *delay* masih berada pada nilai yang menjadi standar ITU-T G.1010. *Packet loss* yang dihasilkan semua jenis layanan sudah memenuhi standar ITU-T G.1010 yaitu 0% hingga saat pada jaringan diberikan background traffic melebihi kapasitas link yaitu pemberian sebesar 75 Mbps. Dalam pengujian waktu konvergensi didapatkan waktu dengan rata-rata nilai 17.71446 detik. Kemudian untuk overhead traffic menunjukkan bahwa perubahan overhead dipengaruhi oleh *Controller* update dan juga flow update, dimana ketika sering terjadinya *Controller* update dan flow update maka semakin besar juga overhead yang didapat. semakin besar juga overhead yang didapat.

- b. **Performance Analysis of Software-Defined Networking (SDN)** (Alexander Gelberger, Niv Yemini, Ran Giladi- 2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems)

Pendekatan Jaringan Perangkat Lunak-Defined (SDN) diperkenalkan sejak pertengahan 1990-an, tapi baru-baru ini Menjadi standar industri yang mapan. Banyak jaringan arsitektur dan sistem mengadopsi SDN, dan vendornya Memilih SDN sebagai alternatif untuk yang tetap, yang telah ditentukan, Dan tumpukan protokol yang fleksibel. SDN menawarkan fleksibilitas, dinamis, Dan fungsi programmable dari sistem jaringan, juga Seperti banyak keuntungan lain seperti kontrol terpusat, re- Kompleksitas, pengalaman pengguna yang lebih baik, dan dramatis, penurunan sistem jaringan dan biaya peralatan. Namun, Karakterisasi SDN dan kemampuan, serta beban kerja dari

trafik jaringan yang ditangani oleh sistem berbasis SDN, Tentukan tingkat keuntungan ini. Apalagi yang diaktifkan Kelancaran sistem berbasis SDN hadir dengan kinerja penalti. Desain dan kemampuan SDN yang mendasarinya Infrastruktur memengaruhi kinerja jaringan umum Tugas, dibandingkan dengan solusi khusus. Dalam tulisan ini kami Menganalisis dua isu: a) dampak SDN terhadap kinerja baku (Dalam hal *Throughput* dan *latency*) di bawah berbagai beban kerja, Dan b) apakah ada hukuman kinerja yang melekat untuk Infrastruktur SDN yang kompleks dan lebih fungsional. Hasil kami Menunjukkan bahwa SDN memang memiliki hukuman kinerja; namun, Itu tidak harus berhubungan dengan tingkat kompleksitas Infrastruktur SDN yang mendasarinya

- c. **Analysing the performance of the *Openflow* standard for Software defined networking using the OMNeT++ network simulator** (Ameen Banjar, Pakawat Pupatwibul, Robin Braun and Bruce Moulton, IEEE 2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE))

Software-defined *networking* (SDN) adalah relatif metode lanjutan untuk mengimplementasikan jaringan komunikasi. SDN memisahkan pengambil keputusan, yang disebut pesawat kontrol, yang menentukan di mana paket dikirim, dari yang mendasarinya infrastruktur, disebut *data plane*, yang meneruskan paket ke tujuan yang telah ditentukan. Standar SDN yang baru muncul standar *openflow*, yang mencakup protokol standar untuk komunikasi antara pesawat kontrol dan data pesawat. Studi ini menganalisis sejauh mana lokasi dari kontroler *openflow* mempengaruhi kinerja *openflow* jaringan. Analisis dilakukan dengan menggunakan OMNeT ++ INET kerangka acara diskrit simulator jaringan. Dengan menganalisa kunci metrik jaringan termasuk round-trip-time (RTT) dan data transfer rate (DTR), hasilnya menunjukkan lokasi dari pengendali memiliki efek yang dapat dibuktikan pada kinerja jaringan.

- d. ***Openflow: Enabling Innovation in Campus Networks*** (N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker

and J. Turner SIGCOMM Computer Communication, vol. 38, pp. 69-74, 2008.)

Paper ini mengusulkan *openflow*: cara bagi para periset untuk menjalankan protokol eksperimental di jaringan yang mereka gunakan every day *openflow* didasarkan pada *switch* Ethernet, dengan tabel aliran internal, dan antarmuka standar untuk ditambahkan dan menghapus entri aliran. Tujuan kami adalah untuk mendorong jaringan-meminta vendor untuk menambahkan *openflow* ke produk *switch* mereka penyebaran di kampus kampus tulang punggung dan kabel lemari. kami percaya bahwa *openflow* adalah kompromi pragmatis: pada satu tangan, memungkinkan peneliti untuk menjalankan eksperimen pada hetero- *switch* geneous dengan cara yang seragam pada line-rate dan dengan tinggi kepadatan pelabuhan; Sementara di sisi lain, vendor tidak perlu untuk mengekspos kerja internal *switch* mereka. Sebagai tambahan untuk memungkinkan peneliti mengevaluasi gagasan mereka di dunia nyata pengaturan yang ketat, *openflow* bisa menjadi kampus yang berguna komponen dalam testbeds berskala besar yang diusulkan seperti GENI. Dua bangunan di Stanford University akan segera menjalankan *openflow* jaringan, menggunakan *switch* Ethernet komersial dan router. kami akan bekerja untuk mendorong penyebaran di sekolah lain; dan kami mendorong Anda untuk mempertimbangkan penerapan *openflow* di jaringan universitas juga

e. Performances of Openflow Based Software-Defined Networks: An

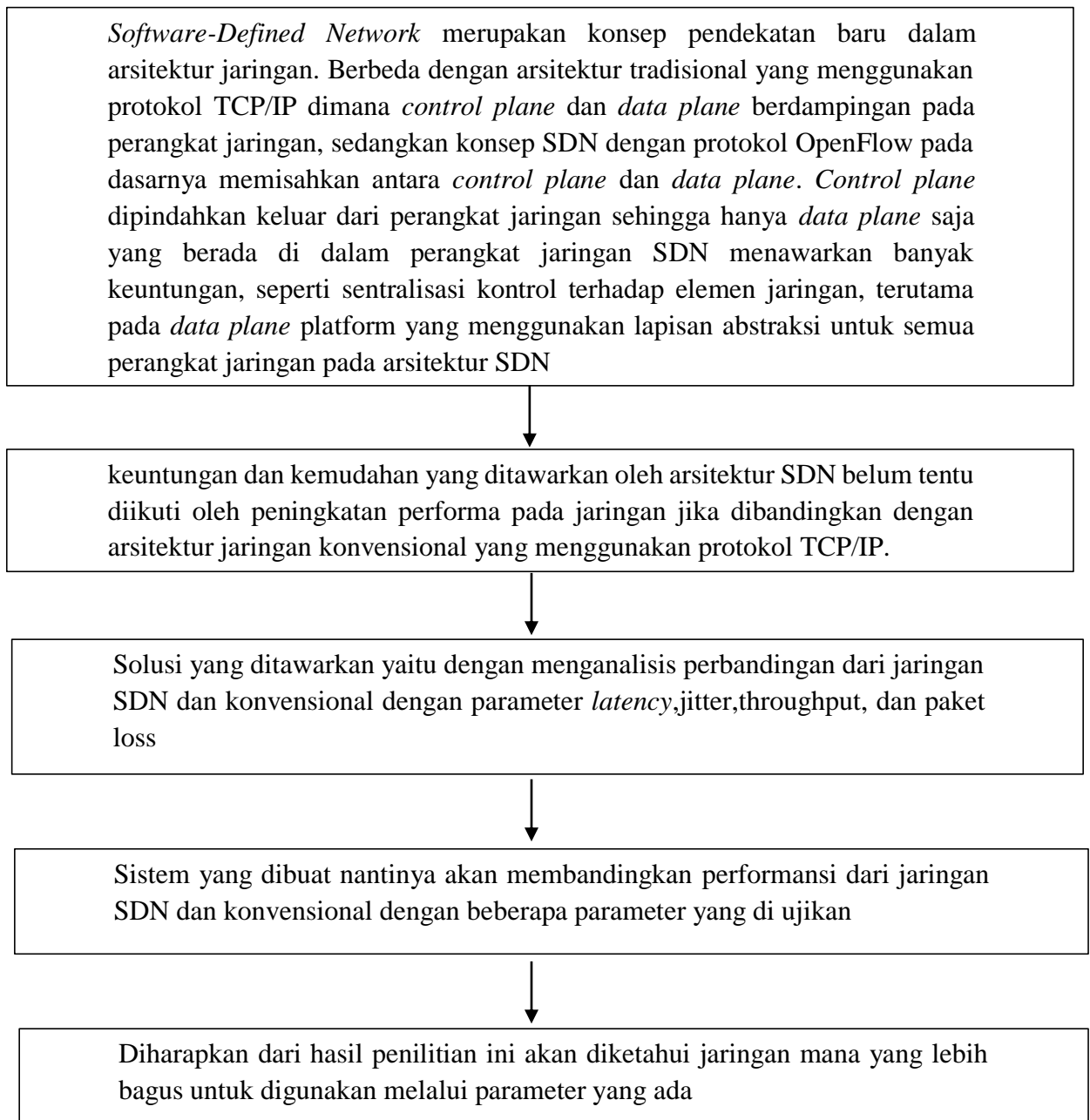
overview(F. Benamrane, M. Ben mamoun and R. Benain, Journal Of Networks, vol. 10, pp. 329-337, 2015.)

Software defined networking (SDN) telah menguat perhatian yang signifikan dari periset jaringan dan industry dalam beberapa tahun terakhir. Memang, konsep SDN banyak memberi keunggulan seperti programmability dan *easy management* dari jaringan. Namun, hal itu menimbulkan tantangan baru skalabilitas dan kinerja, pemahaman mendalam pertunjukan dan keterbatasan konsep SDN adalah prasyarat untuk penerapan dan penerapannya secara nyata jaringan. Dalam tulisan ini, kami bertujuan untuk menyajikannya di cara yang komprehensif, karya terpenting yang dipusatkan pertunjukan SDN.

Saat SDN memisahkan pesawat control dari data pesawat, kami pertama kali mempresentasikan hasil penelitian yang dilakukan untuk meningkatkan kinerja perangkat *data plane*, maka, kita berikan ikhtisar tentang berbagai solusi yang diusulkan untuk diperbaiki kinerja *Controller*. Kami juga memberikan ikhtisar tentang arsitektur pesawat kontrol baru-baru ini dengan beberapa pengendali yang telah diusulkan untuk memenuhi pertunjukan dan kendala skalabilitas. Akhirnya, kami menyajikan yang berbeda teknik dan alat yang digunakan dalam literatur untuk mengevaluasi kinerja jaringan yang didefinisikan perangkat lunak.

2.3 Kerangka Konseptual

Pada sub bab ini akan dijelaskan kerangka konseptual dari penelitian ini



Gambar 1 Kerangka Konseptual

BAB III

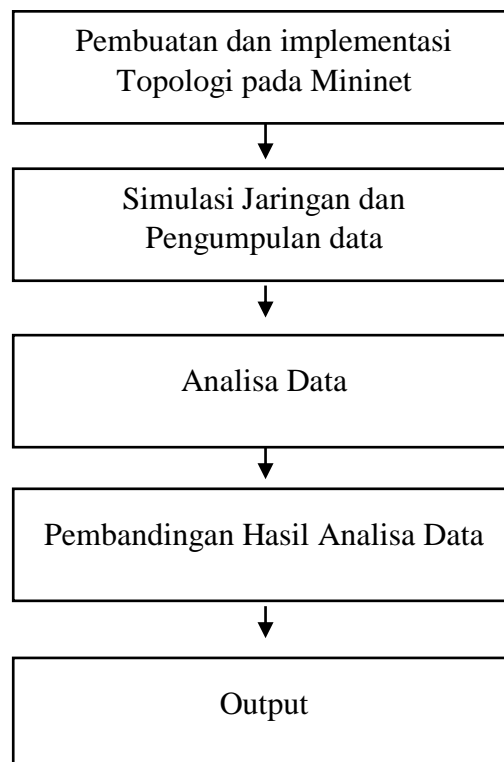
METODE PENELITIAN

3.1 Waktu dan Tempat

Penelitian ini dilaksanakan dari bulan Januari 2018 sampai dengan bulan April 2018. Lokasi penelitian dilakukan di Laboratorium Rekayasa Perangkat Lunak Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin.

3.2 Tahapan Penelitian

Pada sub bab ini kan ditampilkan alur dari tahapan penelitian



Gambar 2 Tahapan Penelitian

3.3 Sumber Data

Sumber data didapatkan dari hasil simulasi jaringan SDN dan Konvensional menggunakan mininet yang di ulangi sebanyak 25 kali percobaan untuk masing-masing topology.

3.4 Instrumen Penelitian

Instrumen yang digunakan dalam penelitian adalah Laptop dengan Processor 64 Bit AMD A10, dengan RAM sebesar 4 Gigabyte menggunakan sistem operasi Linux Ubuntu dengan emulator mininet.

BAB IV

HASIL DAN PEMBAHASAN

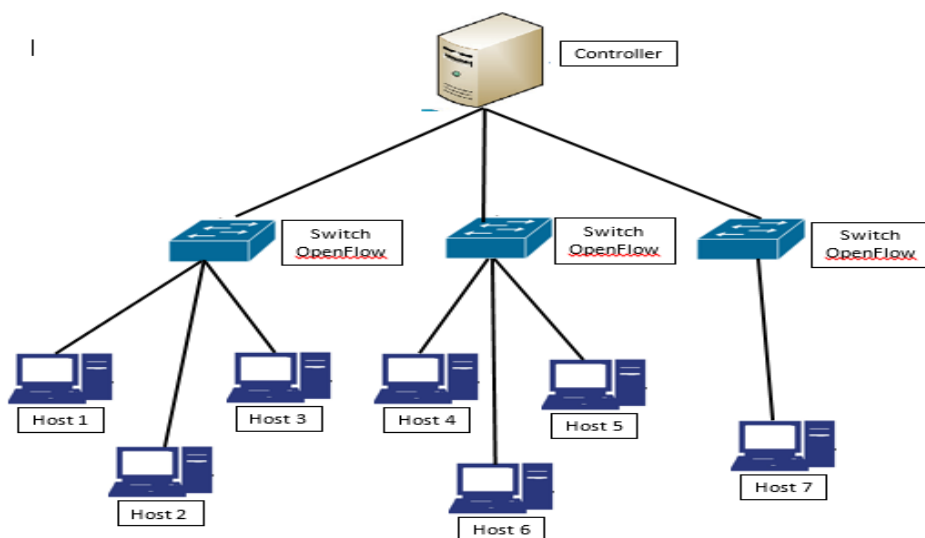
Pada bab ini dibahas mengenai hasil dan analisis dari pengujian untuk setiap skenario uji yang telah dirancang pada bab sebelumnya. Pada pengujian ini, dilakukan pengukuran terhadap empat parameter QoS yang berbeda yaitu latency, *throughput*, *Jitter* dan *Packet loss* bagi setiap arsitektur jaringan baik SDN maupun konvensional. Hasil pengujian dan analisa setiap skenario uji diuraikan sebagai berikut.

4.1 Perancangan Sistem

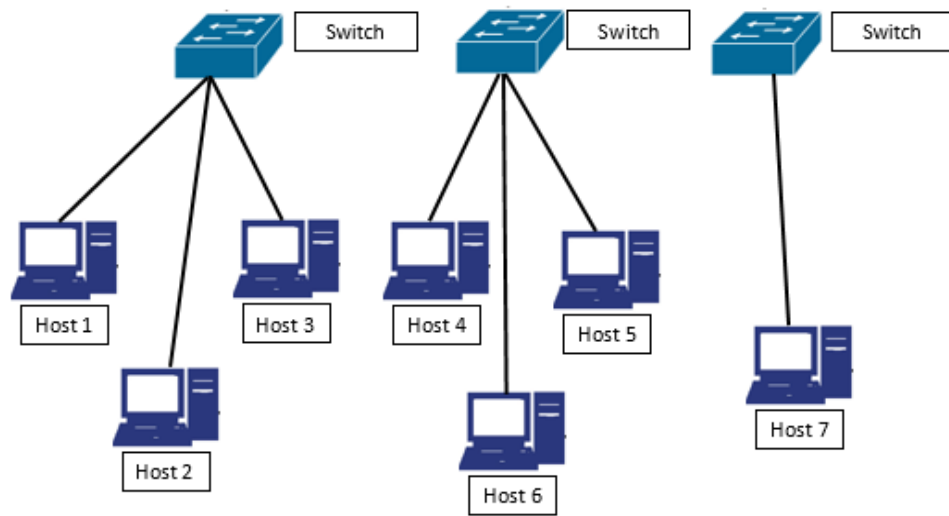
Sistem yang dibangun pada penelitian ini adalah berbasis pada simulasi menggunakan emulator Mininet. Untuk arsitektur SDN, *Controller* akan terkoneksi langsung dengan emulator Mininet yang didalamnya terdiri dari dua buah *switch* OpenFlow yang saling terkoneksi. Setiap *switch* OpenFlow akan terkoneksi masing-masing dengan dua buah *hosts*. Sedangkan untuk arsitektur konvensional, *Controller* tidak digunakan. Ruang lingkup sistem merepresentasikan jaringan yang masih berada dalam 1 subnet.

4.1.1 Rancangan Topologi

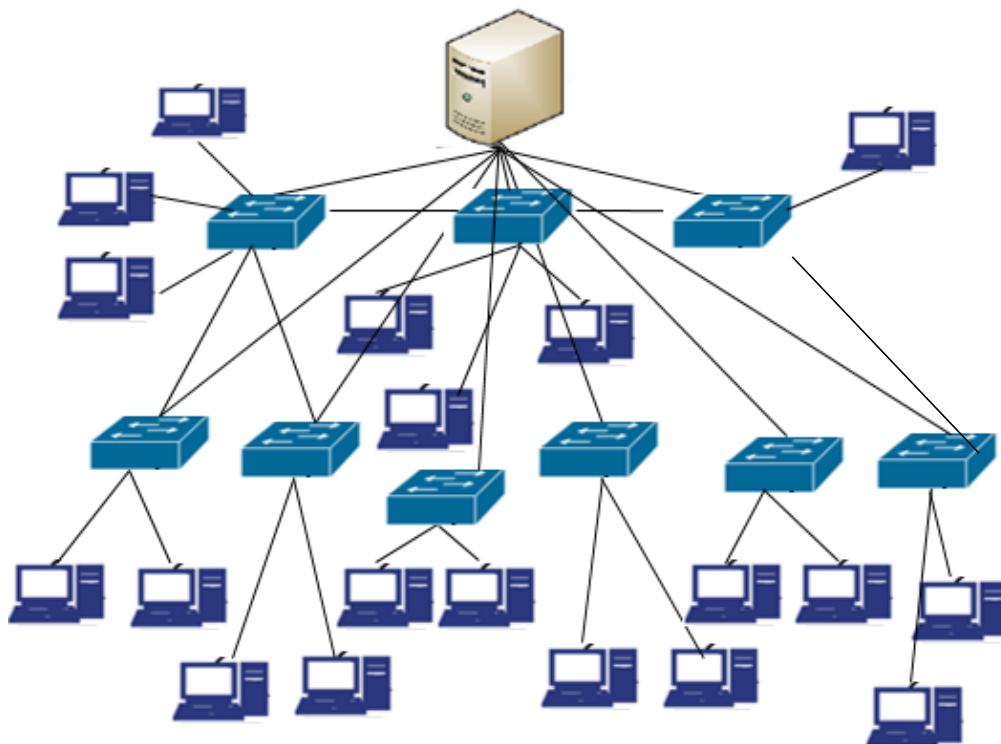
Rancangan topologi yang digunakan ada 5 macam yaitu :



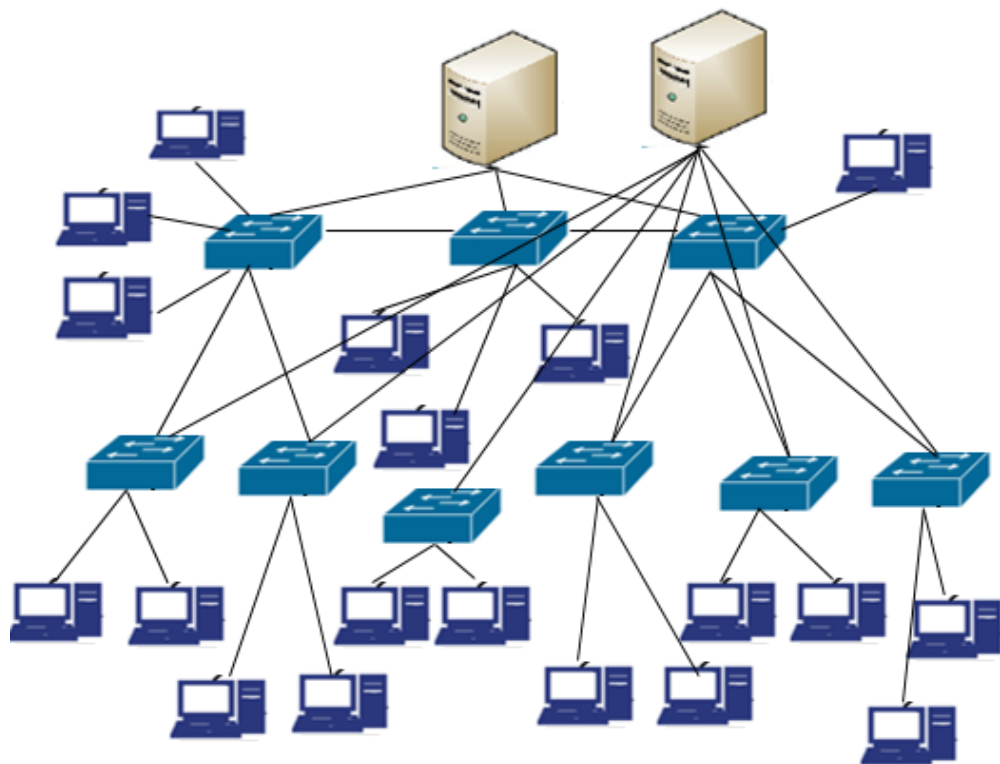
Gambar 3 Topologi Jaringan SDN 1



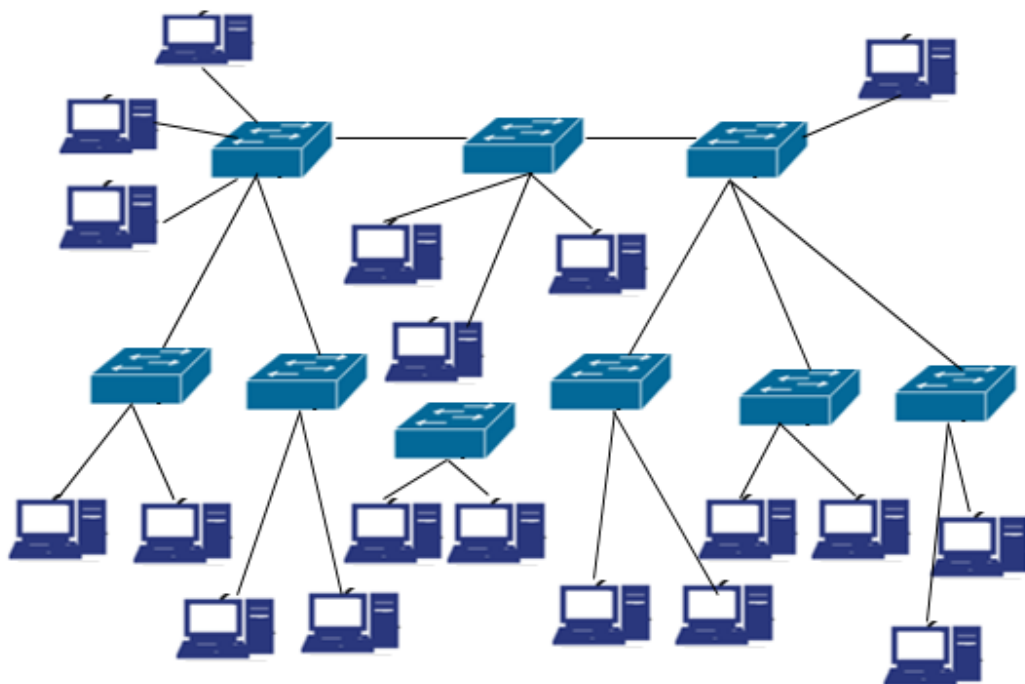
Gambar 4 Topologi Jaringan Konvensional 1



Gambar 5 Topologi Jaringan SDN 2



Gambar 6 Topologi Jaringan SDN 3



Gambar 7 Topologi Jaringan Konvensional 2

4.2 Tahap instalasi dan Konfiurasi

Pada tahap ini akan dilakukan instalasi dan konfigurasi dan jaringan yang akan dibuat

4.2.1 *Controller Floodlight*

Berikut merupakan langkah-langkah menginstall *Controller*

1. Unduh Floodlight dan *build* via terminal menggunakan perintah sebagai berikut

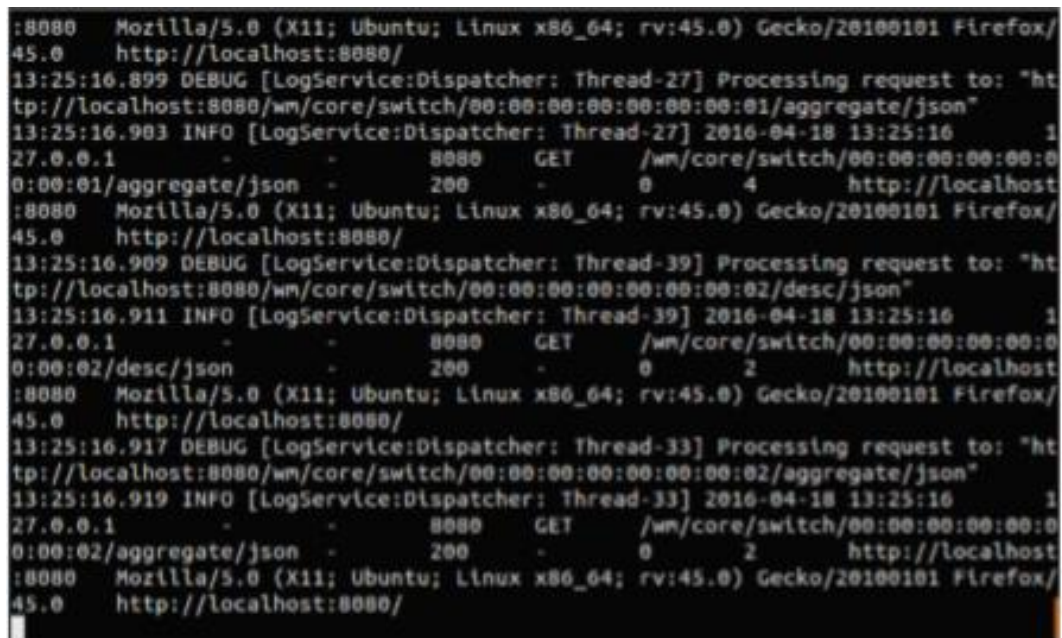
```
$ git clone git://github.com/floodlight/floodlight.git
$ cd floodlight
$ ant

$ sudo mkdir /var/lib/floodlight
$ sudo chmod 777 /var/lib/floodlight
```

2. Setelah proses instalasi dan *build* selesai, maka *Controller Floodlight* dapat dijalankan dengan menggunakan perintah berikut :

```
$ java -jar target/floodlight.jar
```

3. Jika Proses selesai maka terminal akan memunculkan tampilan sebagai berikut :



The screenshot shows a terminal window with the following output:

```
8080 Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0 http://localhost:8080/
13:25:16.899 DEBUG [LogService:Dispatcher: Thread-27] Processing request to: "http://localhost:8080/wm/core/switch/00:00:00:00:00:00:00:01/aggregate/json"
13:25:16.903 INFO [LogService:Dispatcher: Thread-27] 2016-04-18 13:25:16 1 27.0.0.1 - 8080 GET /wm/core/switch/00:00:00:00:00:00:00:01/aggregate/json - 200 - 0 4 http://localhost:8080/
8080 Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0 http://localhost:8080/
13:25:16.909 DEBUG [LogService:Dispatcher: Thread-39] Processing request to: "http://localhost:8080/wm/core/switch/00:00:00:00:00:00:00:02/desc/json"
13:25:16.911 INFO [LogService:Dispatcher: Thread-39] 2016-04-18 13:25:16 1 27.0.0.1 - 8080 GET /wm/core/switch/00:00:00:00:00:00:00:02/desc/json - 200 - 0 2 http://localhost:8080/
8080 Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0 http://localhost:8080/
13:25:16.917 DEBUG [LogService:Dispatcher: Thread-33] Processing request to: "http://localhost:8080/wm/core/switch/00:00:00:00:00:00:00:02/aggregate/json"
13:25:16.919 INFO [LogService:Dispatcher: Thread-33] 2016-04-18 13:25:16 1 27.0.0.1 - 8080 GET /wm/core/switch/00:00:00:00:00:00:00:02/aggregate/json - 200 - 0 2 http://localhost:8080/
8080 Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0 http://localhost:8080/
```

4.2.2 Emulator Mininet

1. Mengunduh mininet dari source github

```
$ git clone git://github.com/Mininet/Mininet
```

2. Melakukan proses instalisasi

```
$ Mininet/util/install.sh -a
```

3. Menjalankan emulator mininet

```
$ sudo mn
```

4. Secara langsung mininet akan menjalankan topologi default. Setelah itu dijalankan sintaks untuk mengubah topologi default

```
mn --custom SDNtopology.py --topo sdn --  
controller=remote,ip=192.168.9.1,port=6653 --mac
```

```
mn --custom nonSDNtopology.py --topo nonSDN --switch  
nonSDN --controller=none --mac
```

4.3 Skenario 1 (Uji Latency)

4.3.1 Pengujian Latency Topologi SDN dan Konvensional Sederhana

Pada skenario 1 ini dilakukan pengukuran terhadap latency dengan melakukan pengiriman paket ICMP dengan memperhatikan perubahan besaran paket yaitu 64, 128, 512, 1024, 2048, dan 4096 *bytes*. Pengujian dilakukan menggunakan *tools* ping dengan skema perintah diberikan dalam Gambar 6. Pada skema ini digunakan ukuran interval 10 dengan total paket yang dibangkitkan secara acak. Sedangkan untuk ukuran paket bergantung pada ukuran paket ICMP yang akan dikirimkan.

```
ping -s [ukuran paket] -i [interval] -c [total paket] [iptujuan]
```

```
72 bytes from 10.0.0.7: icmp_seq=7 ttl=64 time=0.096 ms
72 bytes from 10.0.0.7: icmp_seq=8 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=9 ttl=64 time=0.093 ms
72 bytes from 10.0.0.7: icmp_seq=10 ttl=64 time=0.087 ms
72 bytes from 10.0.0.7: icmp_seq=11 ttl=64 time=0.092 ms
72 bytes from 10.0.0.7: icmp_seq=12 ttl=64 time=0.093 ms
72 bytes from 10.0.0.7: icmp_seq=13 ttl=64 time=0.091 ms
72 bytes from 10.0.0.7: icmp_seq=14 ttl=64 time=0.089 ms
72 bytes from 10.0.0.7: icmp_seq=15 ttl=64 time=0.090 ms
72 bytes from 10.0.0.7: icmp_seq=16 ttl=64 time=0.091 ms
72 bytes from 10.0.0.7: icmp_seq=17 ttl=64 time=0.121 ms
72 bytes from 10.0.0.7: icmp_seq=18 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=19 ttl=64 time=0.091 ms
72 bytes from 10.0.0.7: icmp_seq=20 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=21 ttl=64 time=0.096 ms
72 bytes from 10.0.0.7: icmp_seq=22 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=23 ttl=64 time=0.104 ms
72 bytes from 10.0.0.7: icmp_seq=24 ttl=64 time=0.090 ms
72 bytes from 10.0.0.7: icmp_seq=25 ttl=64 time=0.096 ms

--- 10.0.0.7 ping statistics ---
25 packets transmitted, 25 received, 0% packet loss, time 48365ms
rtt min/avg/max/mdev = 0.082/3.958/96.335/18.856 ms
root@yaumil-VirtualBox:~#
```

Gambar 8 Uji Latency Arsitektur SDN

```
72 bytes from 10.0.0.7: icmp_seq=7 ttl=64 time=0.098 ms
72 bytes from 10.0.0.7: icmp_seq=8 ttl=64 time=0.106 ms
72 bytes from 10.0.0.7: icmp_seq=9 ttl=64 time=0.097 ms
72 bytes from 10.0.0.7: icmp_seq=10 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=11 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=12 ttl=64 time=0.096 ms
72 bytes from 10.0.0.7: icmp_seq=13 ttl=64 time=0.096 ms
72 bytes from 10.0.0.7: icmp_seq=14 ttl=64 time=0.096 ms
72 bytes from 10.0.0.7: icmp_seq=15 ttl=64 time=0.059 ms
72 bytes from 10.0.0.7: icmp_seq=16 ttl=64 time=0.095 ms
72 bytes from 10.0.0.7: icmp_seq=17 ttl=64 time=0.099 ms
72 bytes from 10.0.0.7: icmp_seq=18 ttl=64 time=0.096 ms
72 bytes from 10.0.0.7: icmp_seq=19 ttl=64 time=0.185 ms
72 bytes from 10.0.0.7: icmp_seq=20 ttl=64 time=0.097 ms
72 bytes from 10.0.0.7: icmp_seq=21 ttl=64 time=0.103 ms
72 bytes from 10.0.0.7: icmp_seq=22 ttl=64 time=0.090 ms
72 bytes from 10.0.0.7: icmp_seq=23 ttl=64 time=0.103 ms
72 bytes from 10.0.0.7: icmp_seq=24 ttl=64 time=0.122 ms
72 bytes from 10.0.0.7: icmp_seq=25 ttl=64 time=0.093 ms

--- 10.0.0.7 ping statistics ---
25 packets transmitted, 25 received, 0% packet loss, time 48377ms
rtt min/avg/max/mdev = 0.059/0.129/0.835/0.145 ms
root@yaumil-VirtualBox:~#
```

Gambar 9 Uji Latency Arsitektur Tradisional

4.3.2 Analisa

Dari hasil pengujian yang dilakukan, diperoleh nilai uji latency untuk arsitektur SDN dan konvensional yang diberikan dalam Tabel 5. Dari hasil tersebut diketahui bahwa nilai latency yang terbaik bagi arsitektur jaringan SDN maupun konvensional didapatkan ketika besar paket ICMP berukuran 128 *bytes*. Hal ini

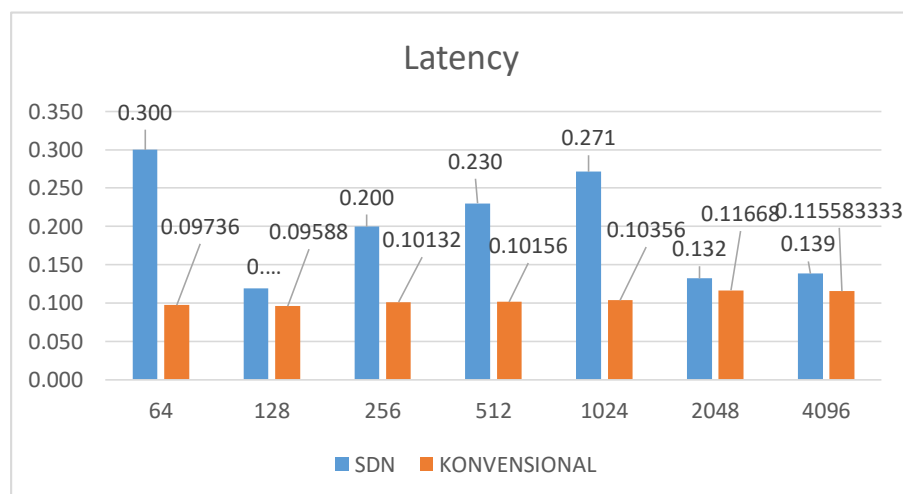
dikarenakan pada ukuran 128 *bytes* diperoleh nilai *latency* terkecil mengikuti rumusan berikut

$$Delay\ rata - rata = \frac{Total\ delay}{Total\ paket\ yang\ diterima}$$

Jika dilakukan perbandingan antara kedua arsitektur diperoleh hasil yang menunjukkan bahwa arsitektur konvensional masih lebih baik daripada arsitektur SDN dengan nilai delay rata-rata sebesar 0.09588. Hal ini disebabkan karena pada arsitektur SDN setiap keputusan untuk transmisi paket dilakukan oleh *Controller*. Hal ini karena jika paket *echo request* pertama membutuhkan waktu yang lebih untuk proses address resolution oleh *Controller* dan pada saat yang sama *Controller* memasukkan *flow entries* kedalam *flow tables* dari *switch* agar pada *forwarding* paket selanjutnya tanpa memerlukan proses resolving address lagi.

Tabel 5 Hasil Uji Latency

Ukuran (<i>byte</i>)	SDN	KONVENSIONAL
64	0.300	0.09736
128	0.119	0.09588
256	0.200	0.10132
512	0.230	0.10156
1024	0.271	0.10356
2048	0.132	0.11668
4096	0.139	0.115583333



Gambar 10 Hasil Uji Latency

Tingginya nilai latency yang disebabkan oleh komunikasi antar *Controller* dan *switch* terjadi secara 2 tahap yaitu paket *arrival* dan *forwarding table updates*. Pada tahap pertama, ketika menerima paket ICMP pertama. *Switch* mengirimkan pesan paket_in kepada *Controller* yang berisi metadata dan 128B pertama dari paket. Pada tahap kedua, *Controller* mengirimkan pesan *flow_mods* untuk memasukan atau memperbaharui *flow* entries ke dalam *flow* tables dari *switch*. Tahap ini dinamakan *inbound* dan *outbound latency*.

Sebagaimana dari hasil pengujian yang menunjukkan paket ICMP pertama dari setiap ukuran paket yang dikirimkan selalu menunjukkan nilai latency yang begitu besar bagi arsitektur jaringan SDN.

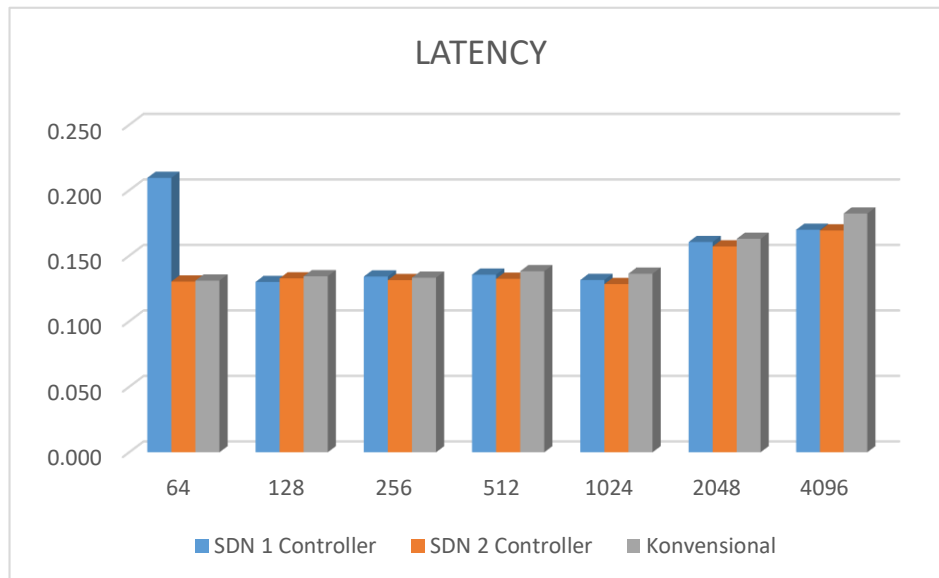
4.3.3 Pengujian Latency Topologi SDN dan Konvensional Rumit

Pada skenario ini dilakukan pengukuran terhadap latency dengan menggunakan 3 topologi yang lebih kompleks dibandingkan topologi sebelumnya dan salah satu topologi menggunakan 2 *Controller* untuk melihat seberapa besar perbedaan antara topologi menggunakan 1 *Controller*. Pengujian dilakukan dengan mengirimkan paket ICMP dengan memperhatikan perubahan besaran paket yaitu 64, 128, 512, 1024, 2048, dan 4096 *bytes*. Pengujian dilakukan menggunakan *tools* ping dengan skema perintah diberikan dalam Gambar 6. Pada skema ini digunakan ukuran interval 10 dengan total paket yang dibangkitkan secara acak. Sedangkan untuk ukuran paket bergantung pada ukuran paket ICMP yang akan dikirimkan.

4.3.4 Analisa

Tabel 6 Hasil Uji Latency

Ukuran (Byte)	SDN 1 <i>Controller</i>	SDN 2 <i>Controller</i>	Konvensional
64	0.289	0.130	0.131
128	0.130	0.133	0.134
256	0.134	0.131	0.133
512	0.136	0.132	0.138
1024	0.131	0.128	0.136
2048	0.160	0.157	0.163
4096	0.170	0.169	0.182



Gambar 11 Hasil Uji Latency

Dari hasil pengujian yang dilakukan, diperoleh nilai uji latency untuk arsitektur SDN 1 *Controller*, SDN 2 *Controller* dan konvensional yang diberikan dalam Tabel 6. Dari hasil tersebut diketahui bahwa nilai latency yang terbaik bagi arsitektur jaringan SDN didapatkan ketika besar paket ICMP berukuran 128 bytes dan 1024 bytes masing-masing untuk SDN 1 *Controller* dan SDN 2 *Controller*. Sedangkan untuk jaringan arsitektur konvensional nilai latency terbaik ketika paket besar paket ICMP berukuran 256 bytes. Secara keseluruhan nilai latency untuk jaringan SDN lebih baik daripada jaringan Konvensional. Dari hasil pengujian yang didapatkan, SDN 2 *Controller* memperoleh nilai latency yang paling terbaik. Hal ini disebabkan karena pada arsitektur SDN setiap keputusan untuk mentransmisi paket dilakukan oleh *controller* sehingga *switch* tidak berhak menentukan kemana paket yang akan diterimanya akan dikirimkan, melainkan harus menunggu keputusan dari *controller*, maka dengan menggunakan 2 *controller* yang setiap *controllernya* menaungi beberapa *switch* sehingga akan lebih cepat dalam melakukan pengiriman maupun penerimaan. *Controller* memerlukan waktu berlebih untuk melakukan address resolution karena pada saat yang sama *controller* memasukkan flow entries kedalam flow tables dari *switch* agar pada forwarding paket selanjutnya tidak dibutuhkan lagi proses resolving address, dengan

menggunakan 2 *controller* atau lebih maka akan mempersingkat waktu untuk proses address resolution maupun proses resolving address.

4.4 Uji *Throughput*

4.4.1 Pengujian *Throughput* Topologi SDN dan Konvensional

Sederhana

Pada scenario uji *throughput* ini, dilakukan terhadap *throughput* dengan melakukan pengiriman paket TCP dengan variasi *window size* sebesar 64KB, 128KB, 256KB, 512KB dan 1024KB. Pada koneksi TCP, *window size* menentukan jumlah maksimum data yang dapat berada dalam jaringan pada saat yang bersamaan. Variasi *window size* merupakan prosedur standar dalam proses tuning koneksi TCP untuk mendapatkan bandwidth atau *throughput* yang maksimum. Apabila ukuran *window size* terlalu kecil maka jaringan akan idle untuk waktu tertentu sehingga didapatkan performa jaringan yang buruk.

Pengujian menggunakan *tools* iperf untuk men- *generate* paket data TCP dengan menggunakan perintah sebagai berikut :

Iperf -c [ip server] -w [ukuran window] -t [timing] -i [interval]

Gambar 12 Uji *Throughput* Arsitektur SDN

```

root@yaumil-VirtualBox:~# iperf -c 10.0.0.7 -w 64k -t 10 -i 2
-----
Client connecting to 10.0.0.7, TCP port 5001
TCP window size: 128 KByte (WARNING: requested 64.0 KByte)
-----
[ 5] local 10.0.0.1 port 34358 connected with 10.0.0.7 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 5] 0.0- 2.0 sec   2.73 GBytes 11.7 Gbits/sec
[ 5] 2.0- 4.0 sec   2.86 GBytes 12.3 Gbits/sec
[ 5] 4.0- 6.0 sec   2.53 GBytes 10.9 Gbits/sec
[ 5] 6.0- 8.0 sec   2.80 GBytes 12.0 Gbits/sec
[ 5] 8.0-10.0 sec   3.16 GBytes 13.6 Gbits/sec
[ 5] 0.0-10.0 sec  14.1 GBytes 12.1 Gbits/sec
root@yaumil-VirtualBox:~# █

```

Gambar 13 Uji *Throughput* Arsitektur Konvensional

```

root@yaumil-VirtualBox:~# iperf -c 10.0.0.7 -w 64k -t 10 -i 2
-----
Client connecting to 10.0.0.7, TCP port 5001
TCP window size: 128 KByte (WARNING: requested 64.0 KByte)
-----
[ 5] local 10.0.0.1 port 34762 connected with 10.0.0.7 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 5] 0.0- 2.0 sec   3.71 GBytes 15.9 Gbits/sec
[ 5] 2.0- 4.0 sec   3.43 GBytes 14.7 Gbits/sec
[ 5] 4.0- 6.0 sec   2.40 GBytes 10.3 Gbits/sec
[ 5] 6.0- 8.0 sec   3.37 GBytes 14.5 Gbits/sec
[ 5] 8.0-10.0 sec   3.57 GBytes 15.3 Gbits/sec
[ 5] 0.0-10.0 sec  16.5 GBytes 14.2 Gbits/sec
root@yaumil-VirtualBox:~# █

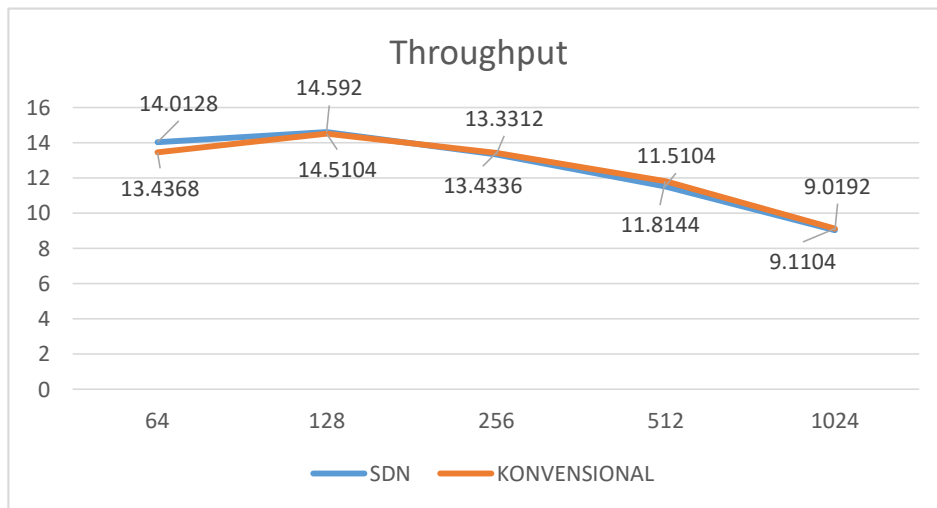
```

4.4.2 Analisa

Dari hasil pengujian sebanyak 25 kali pengambilan data, didapatkan rata-rata nilai *throughput* untuk setiap variasi *window size* seperti berikut :

Tabel 7 Hasil Uji Throughput

Window Size	Throughput (Gbps)	
	SDN	Konvensional
64	14.0128	13.4368
128	14.592	14.5104
256	13.3312	13.4336
512	11.5104	11.8144
1024	9.0192	9.1104



Gambar 14 Hasil Uji *Throughput*

Pada Gambar 14, dilihat bahwa besarnya *throughput* ikut berubah ketika ukuran *window size* berubah. Perubahan *throughput* tersebut tidak berbanding lurus dengan ukuran *window size*. Sehingga, ukuran *window size* yang besar belum tentu menghasilkan nilai *throughput* yang maksimum baik bagi jaringan dengan arsitektur SDN maupun konvensional.

TCP *window size* merupakan variabel selama koneksi berlangsung. Setiap *acknowledgment* terdiri dari *window advertisement* yang menandakan seberapa banyak *byte* data yang dapat diterima. Dengan satu *window size*, setiap segmen harus saling balas sebelum segmen lain ditransmisikan.

Jika dibandingkan, secara keseluruhan nilai *throughput* yang diperoleh dari setiap ukuran *window size* baik bagi arsitektur jaringan SDN maupun konvensional menunjukkan nilai sama baiknya. Hal ini ditunjukkan oleh rasio perbandingan nilai *throughput* yang hanya berkisar 0.01-0.1 % dari semua ukuran *window size* yang diujikan.

Dari hasil pengujian, bahwa rata-rata nilai *throughput* maksimum bagi arsitektur SDN maupun arsitektur konvensional didapatkan saat *window size* berukuran 128 KB.

4.4.3 Pengujian *Throughput* Topologi SDN dan Konvensional Rumit

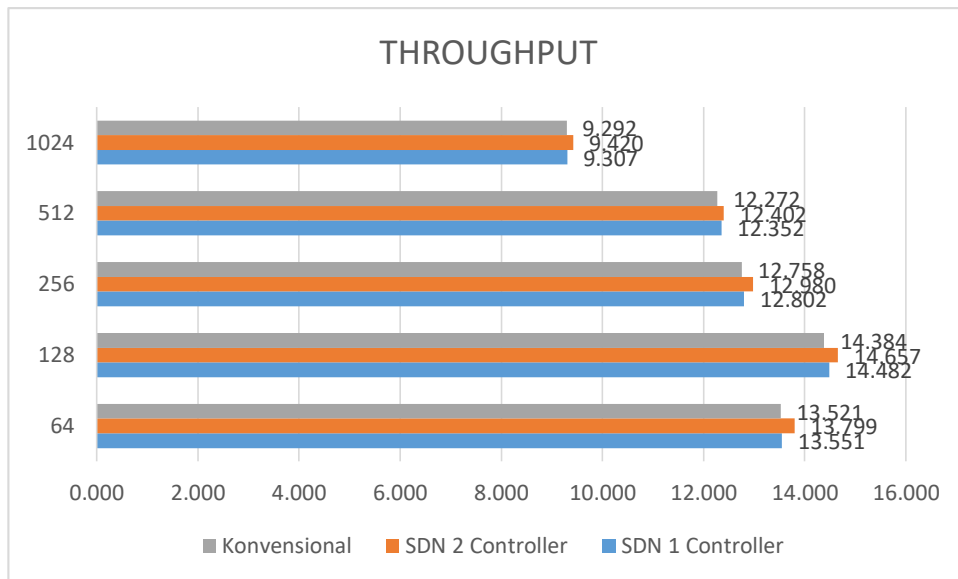
Pada skenario ini dilakukan pengukuran terhadap *Throughput* dengan menggunakan 3 topologi yang lebih kompleks dibandingkan topologi sebelumnya dan salah satu topologi menggunakan 2 *Controller* untuk melihat seberapa besar perbedaan antara topologi menggunakan 1 *Controller*. Pengujian dilakukan dengan mengirimkan paket TCP dengan variasi *window size* sebesar 64, 128, 512, dan 1024

4.4.4 Analisa

Dari hasil pengujian sebanyak 25 kali didapatkan hasil sebagai berikut :

Tabel 8 Hasil Uji *Throughput*

Window Size	SDN 1 <i>Controller</i>	SDN 2 <i>Controller</i>	Konvensional
64	13.551	13.799	13.521
128	14.482	14.657	14.384
256	12.802	12.980	12.758
512	12.352	12.402	12.272
1024	9.307	9.420	9.292



Gambar 15 Hasil Uji *Throughput*

Pada gambar 15 diketahui bahwa besarnya *throughput* ikut berubah ketika ukuran window size berubah, sehingga ukuran window size besar tidak berbanding lurus atau belum tentu menghasilkan *throughput* maksimum yang baik bagi jaringan SDN maupun konvensional. Window size menentukan seberapa banyak data yang akan di transmisikan pada suatu waktu. Semakin besar window size maka akan semakin banyak data yang di transmisikan.

Window size menentukan seberapa banyak data yang akan di transmisikan pada suatu waktu. Semakin besar *window size* maka semakin banyak data yang dapat di transmisikan.

Jika dilihat dan dibandingkan dengan 3 topologi yang diujikan nilai *throughput* terbaik yaitu topologi yang menggunakan arsitektur jaringan SDN yang menggunakan 2 *controller*. Dari hasil pengujian menunjukkan bahwa rata-rata nilai *throughput* maksimum bagi arsitektur jaringan SDN baik yang menggunakan 1 *controller* dan 2 *controller* dan juga bagi arsitektur konvensional di dapatkan pada saat window size berukuran 128KB.

4.5 Uji *Jitter* dan *Packet loss*

4.5.1 Pengujian *Jitter* dan *Packet Loss* Topologi SDN dan Konvensional Sederhana

Pada skenario ini, dilakukan pengujian terhadap *Jitter* dan *packetloss* dengan melakukan pengiriman paket UDP dengan variasi *buffer size* sebesar 64 KB, 128 KB, 256 KB, 512 KB dan 1024 KB selama periode 10 detik. Variasi ukuran *buffer size* UDP merupakan prosedur standar dalam proses *tunning* koneksi UDP untuk mendapatkan nilai *jitter* dan *packetloss* yang minimum.

Pengujian dilakukan menggunakan *tools* *iperf* untuk men- *generate* paket-paket UDP dengan menggunakan perintah-perintah sebagai berikut :

```
iperf -u [UDP test] -c [ip server] -w [ukuran buffer] -b  
[bandwidth] -t [timing] -I [interval]
```

```
root@yaumil-VirtualBox:~# iperf -u -c 10.0.0.7 -w 64k -b 10m -t 10 -i 2  
-----  
Client connecting to 10.0.0.7, UDP port 5001  
Sending 1470 byte datagrams, IPG target: 1176.00 us (kalman adjust)  
UDP buffer size: 128 KByte (WARNING: requested 64.0 KByte)  
-----  
[ 5] local 10.0.0.1 port 55373 connected with 10.0.0.7 port 5001  
[ ID] Interval      Transfer    Bandwidth  
[ 5] 0.0- 2.0 sec  2.38 MBytes 10.0 Mbits/sec  
[ 5] 2.0- 4.0 sec  2.38 MBytes 10.0 Mbits/sec  
[ 5] 4.0- 6.0 sec  2.39 MBytes 10.0 Mbits/sec  
[ 5] 6.0- 8.0 sec  2.38 MBytes 10.0 Mbits/sec  
[ 5] 8.0-10.0 sec  2.38 MBytes 10.0 Mbits/sec  
[ 5] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec  
[ 5] Sent 8505 datagrams  
[ 5] Server Report:  
[ 5] 0.0- 9.9 sec 11.9 MBytes 10.1 Mbits/sec 0.029 ms 24/ 8505 (0.28%)  
[ 5] 0.00-9.91 sec 50 datagrams received out-of-order  
root@yaumil-VirtualBox:~#
```

Gambar 16 Uji *Jitter* dan *Packet loss* Arsitektur SDN

```
[ 5] 6.0- 8.0 sec 0.00 Bytes 0.00 bits/sec
[ 5] 8.0-10.0 sec 0.00 Bytes 0.00 bits/sec
[ 5] 0.0-10.0 sec 1.44 KBytes 1.18 Kbits/sec
[ 5] Sent 1 datagrams
read failed: Connection refused
[ 5] WARNING: did not receive ack of last datagram after 2 tries.
root@yaumil-VirtualBox:~# iperf -u -c 10.0.0.7 -w 64k -b 10m -t 10 -i 2

-----
Client connecting to 10.0.0.7, UDP port 5001
Sending 1470 byte datagrams, IPG target: 1176.00 us (kalman adjust)
UDP buffer size: 128 KByte (WARNING: requested 64.0 KByte)
-----
[ 5] local 10.0.0.1 port 49873 connected with 10.0.0.7 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 5] 0.0- 2.0 sec  2.39 MBytes 10.0 Mbits/sec
[ 5] 2.0- 4.0 sec  2.38 MBytes 10.0 Mbits/sec
[ 5] 4.0- 6.0 sec  2.38 MBytes 10.0 Mbits/sec
[ 5] 6.0- 8.0 sec  2.38 MBytes 10.0 Mbits/sec
[ 5] 8.0-10.0 sec  2.38 MBytes 10.0 Mbits/sec
[ 5] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec
[ 5] Sent 8505 datagrams
[ 5] Server Report:
[ 5] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec 0.031 ms 0/ 8505 (0%)
root@yaumil-VirtualBox:~#
```

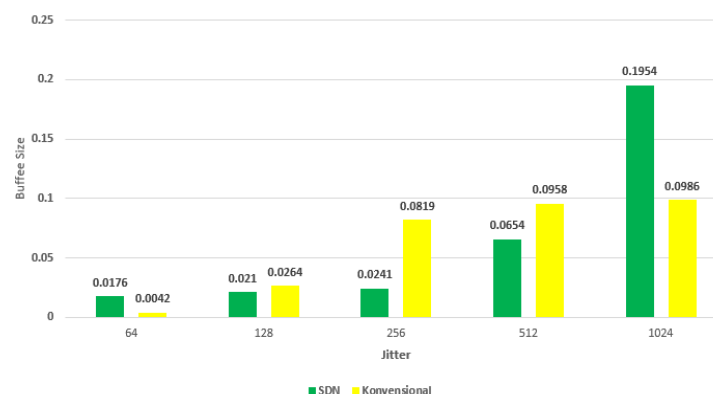
Gambar 17 Uji *Jitter* dan *Packet loss* Arsitektur Konvensional

4.5.2 Analisa

Dari hasil pengujian sebanyak 25 kali didapatkan hasil *Jitter* dan *Packet loss* sebagai berikut :

Tabel 9 Hasil Uji *Jitter*

Buffer Size	<i>Jitter</i> (ms)	
	SDN	Konvensional
64	0.0176	0.0042
128	0.0210	0.0264
256	0.0241	0.0819
512	0.0654	0.0958
1024	0.1954	0.0986



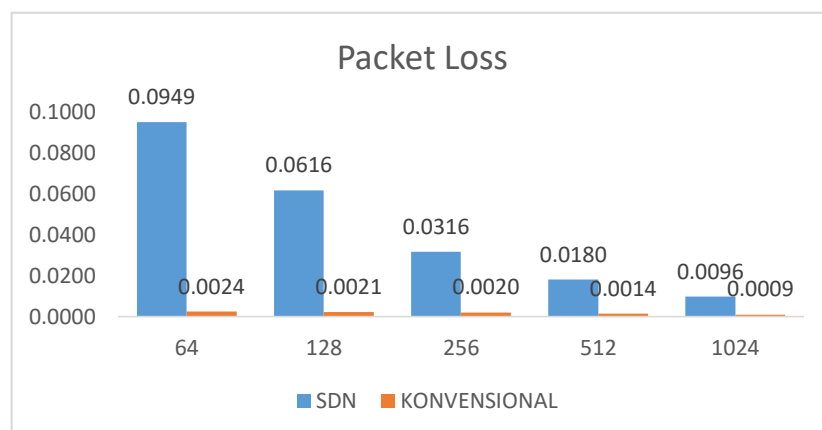
Gambar 18 Hasil Uji *Jitter*

Pada gambar 18 diatas, dapat dilihat bahwa nilai *Jitter* ikut berubah saat ukuran *buffer size* UDP berubah. Secara garis besar, perubahan *buffer size* berbanding lurus dengan nilai *Jitter* yang dihasilkan, baik bagi arsitektur jaringan SDN maupun konvensional. Hal ini ditunjukkan oleh nilai *Jitter* yang meningkat seiring dengan meningkatnya *buffer size* UDP yang ditransmisikan.

Jika dibandingkan, secara keseluruhan nilai *Jitter* yang diperoleh dari setiap ukuran buffer yang diujikan baik bagi arsitektur jaringan SDN maupun konvensional tidak menunjukkan nilai yang fluktuatif. Sehingga, nilai *Jitter* yang rendah didapatkan ketika *buffer size* terkecil yaitu 64 KB. Begitu pula sebaliknya, nilai *Jitter* ya tinggi diperoleh dengan *buffer size* yang terbesar yaitu 1024 KB.

Tabel 10 Hasil Uji Packet loss

Buffer Size	<i>Packet loss</i> (ms)	
	SDN	Konvensional
64	0.0949	0.0024
128	0.0616	0.0021
256	0.0316	0.0020
512	0.0180	0.0014
1024	0.0096	0.0009



Gambar 19 Hasil Uji *Packet loss*

Pada gambar 21, secara garis besar dapat dilihat bahwa semakin besar buffer size UDP yang digunakan ketika mentransmisikan paket akan menyebabkan semakin berkurangnya nilai *Packet loss* di dalam jaringan.

Semakin besar ukuran *buffer* yang digunakan ketika mentransmisikan data dalam suatu waktu, maka akan memperbesar ruang untuk mengakomodasi paket yang datang. Sehingga, dapat mengurangi nilai *Packet loss*. Sebaliknya, jika semakin kecil ukuran *buffer* maka akan menyebabkan semakin besarnya peluang terjadinya *Packet loss* di dalam jaringan karena *buffer* tidak memiliki ruang yang cukup mengakomodasi paket yang datang.

Dari hasil pengujian untuk setiap variasi ukuran buffer size UDP yang dikirimkan, jaringan dengan arsitektur SDN menunjukkan rasio *Packet loss* yang lebih besar dibandingkan dengan arsitektur jaringan konvensional untuk setiap ukuran *buffer size*. Hal ini dikarenakan oleh *Controller* SDN yang tidak dapat memproses semua paket yang masuk sebelum *buffer* pada *switch* mengalami *over run* dan mulai mendrop paket. Selain itu, hal ini juga bisa dimungkinkan oleh paket yang diteruskan sebelum semua *flow entries* terpasang di *path* jaringan sehingga menyebabkan paket akan *drop*.

4.5.3 Pengujian *Jitter* dan *Packet Loss* Topologi SDN dan Konvensional Rumit

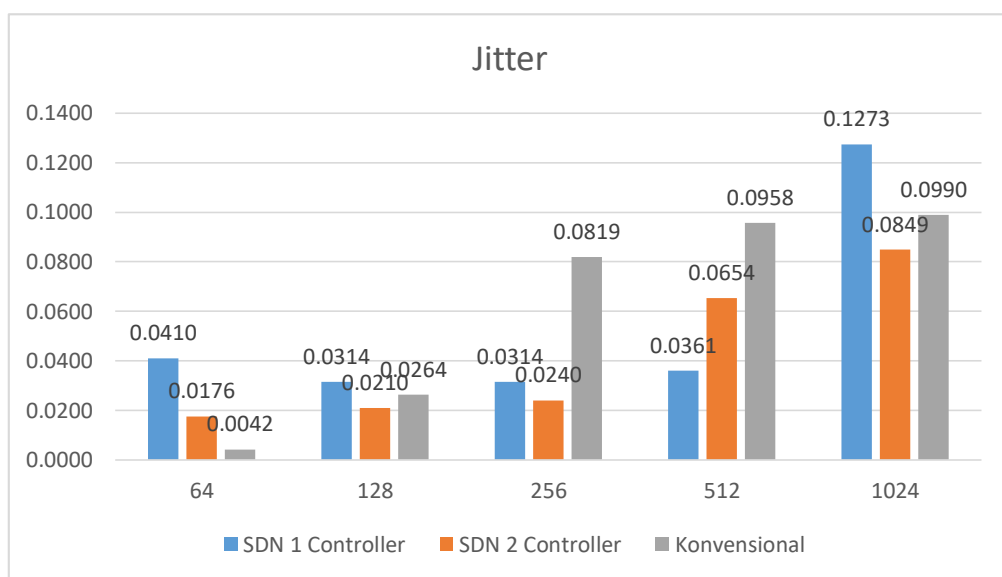
Pada skenario ini, dilakukan pengujian terhadap *Jitter* dan *packetloss* dengan melakukan pengiriman paket UDP dengan variasi *buffer size* sebesar 64 KB, 128 KB, 256 KB, 512 KB dan 1024 KB selama periode 10 detik. Variasi ukuran *buffer size* UDP merupakan prosedur standar dalam proses *tunning* koneksi UDP untuk mendapatkan nilai *jitter* dan *packetloss* yang minimum.

4.5.4 Analisa

Dari hasil pengujian sebanyak 25 kali didapatkan hasil *Jitter* dan *Packet loss* sebagai berikut :

Tabel 11 Hasil Uji Jitter

Buffer Size	SDN 1 Controller	SDN 2 Controller	Konvensional
64	0.0410	0.0176	0.0042
128	0.0314	0.0210	0.0264
256	0.0314	0.0240	0.0819
512	0.0361	0.0654	0.0958
1024	0.1273	0.0849	0.0990



Gambar 20 Hasil Uji Jitter

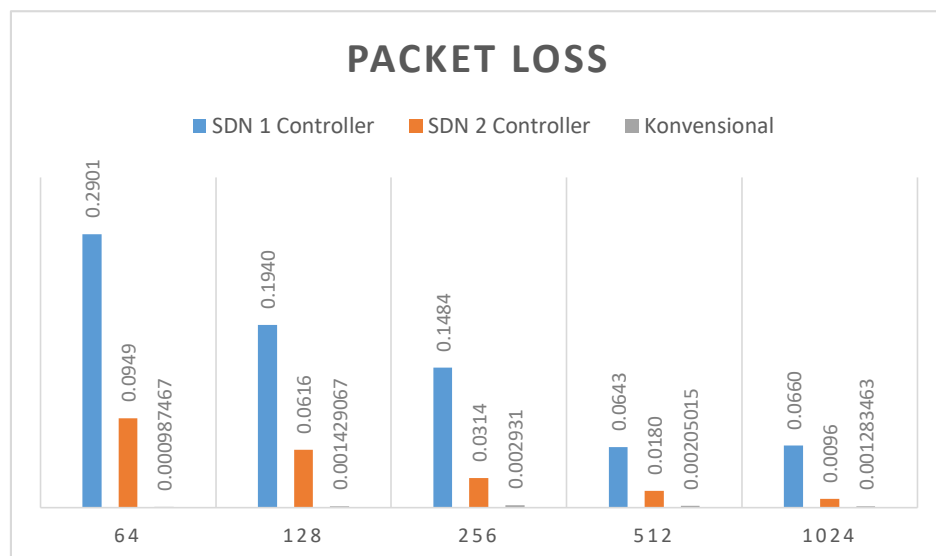
Perhitungan dilakukan *Jitter* oleh iperf dilakukan secara kontinu, dimana server akan menghitung waktu transit relatif (waktu penerimaan server – waktu pengiriman *client*) untuk setiap paket milik data yang sama. Tingginya nilai *Jitter* dalam jaringan mengindikasikan adanya antrian paket pada salah satu node yang dapat menimbulkan *congestion* (kemacetan). Antrian paket tersebut dapat menyebabkan waktu *transfer* paket tidak dapat diprediksi sehingga nilai *Jitter* meningkat.

Pada Gambar 21 dan 22 dilihat bahwa nilai *Jitter* meningkat apabila ukuran buffer size berubah dan jika dilihat nilai *Jitter* yang didapatkan tidak konstan atau tidak tetap baik untuk jaringan arsitektur SDN yang menggunakan 1 *controller*, jaringan arsitektur SDN yang menggunakan 2 *controller* maupun jaringan arsitektur konvensional. Sehingga nilai *Jitter* yang rendah didapatkan ketika ukuran buffer

size sebesar 64KB dan sebaliknya nilai *Jitter* yang terbesar pada saat buffer size sebesar 1024KB.

Tabel 12 Hasil Uji Packet loss

Buffer Size	SDN 1 Controller	SDN 2 Controller	Konvensional
64	0.2901	0.0949	0.000987467
128	0.1940	0.0616	0.001429067
256	0.1484	0.0314	0.002931
512	0.0643	0.0180	0.00205015
1024	0.0660	0.0096	0.001283463



Gambar 21 Hasil Uji *Packet loss*

Dilihat pada hasil uji untuk *Packet loss* semakin besar ukuran buffer size UDP maka semakin sedikit paket yang hilang. Semakin besar ukuran buffer size maka semakin besar pula ruang untuk menagkomodasi paket yang datang dan begitu pula sebaliknya makin kecil ukuran buffer maka akan semakin sedikit pula ruang untuk mengakomodasi paket yang datang.

Tingginya nilai *Packet loss* dapat disebabkan oleh berbagai factor seperti antrian yang berlebihan karena kemacetan didalam jaringan ataupun karena transmisi yang telah dikirim namun tidak pernah sampai pada jaringan yang dituju.

Dari hasil yang didapatkan jaringan arsitektur SDN yang menggunakan 1 *controller* mempunyai nilai *Packet loss* yang tinggi dibandingkan dengan arsitektur jaringan SDN yang menggunakan 2 *controller*, sedangkan untuk jaringan arsitektur konvensional mempunyai nilai *Packet loss* yang rendah dengan hampir mendekati angka 0 pada ukuran buffer size 1024KB. Hal ini dimungkinkan karena *controller* SDN tidak dapat memproses semua paket yang datang ketika *switch* mengalami over run sebelum buffer.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dari penelitian dan pengujian performansi jaringan arsitektur jaringan SDN dan konvensional yang telah dilakukan maka dapat ditarik beberapa poin-poin sebagai berikut :

1. Pengujian dengan mentransmisikan paket ICMP yang bertujuan untuk mengukur nilai *latency* bagi kedua arsitektur jaringan, menunjukkan bahwa adanya nilai *latency* tambahan yang cukup signifikan bagi arsitektur SDN dengan topologi sederhana tetapi apabila dibandingkan dengan arsitektur dengan menggunakan topologi yang lebih kompleks nilai *latency* untuk arsitektur SDN lebih rendah dibandingkan arsitektur konvensional untuk topologi sederhana maupun topologi yang lebih kompleks. Hal ini dikarenakan komunikasi antara *switch* dan *Controller* dalam menangani paket yang akan dikirimkan maupun di terima.
2. Transmisi paket TCP yang dikirimkan bertujuan menghitung nilai *throughput* bagi kedua arsitektur. Dari hasil pengujian menunjukkan bahwa nilai *throughput* untuk ke semua arsitektur jaringan baik itu arsitektur SDN dan konvensional dengan topologi sederhana maupun arsitektur jaringan SDN dan konvensional dengan topologi yang lebih kompleks menunjukkan nilai *throughput* yang sama baiknya. Hal ini terlihat dari tidak adanya perbedaan nilai yang hampir signifikan bagi nilai *throughput* yang didapat dari setiap nilai *window size* yang diujikan.
3. Pengujian dengan mentransmisikan paket UDP menghasilkan nilai *Jitter* dan *Packet loss* bagi ke semua arsitektur jaringan baik itu arsitektur SDN dan konvensional dengan topologi sederhana maupun arsitektur jaringan SDN dan konvensional dengan topologi yang lebih kompleks. Untuk parameter *Jitter* hasil pengujian tidak menunjukkan hasil yang fluktuatif, sehingga nilai *Jitter* berbanding lurus dengan ukuran buffer yang di transmisikan. Sedangkan untuk hasil nilai dari *Packet loss*, arsitektur SDN

baik itu pada arsitektur jaringan SDN yang menggunakan 1 dan 2 *contoller* pada topologi yang lebih kompleks menunjukkan nilai rasio yang lebih besar untuk setiap nilai buffer size dibandingkan dengan arsitektur jaringan konvensional. Hal ini disebabkan paket UDP yang tidak menunggu proses handshake terlebih dahulu melainkan paket datang secara sekaligus. Sehingga, *switch* dan *Controller* tidak siap untuk menangani setup *flow* table dari semua paket yang datang.

5.2 Saran

Beberapa saran dari penulis yang dapat dipertimbangkan untuk penelitian selanjutnya yaitu :

1. Menggunakan *Controller* dan emulator/simulator jaringan yang beda
2. Melibatkan lebih dari satu subnet jaringan
3. Menggunakan lebih dari dua *Controller* dengan topologi jaringan yang lebih kompleks.
4. Tipe jaringan yang dilibatkan tidak hanya *wired* saja, melainkan melibatkan jaringan *wireless* didalamnya.
5. Perlu dilakukan penelitian lebih lanjut mengenai performa dari *Controller* yang digunakan pada arsitektur jaringan SDN.

DAFTAR PUSTAKA

- [1] D. A. Izzatul Ummah, "Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking," *Ind. Journal on Computing Vol. 1, Issue. 1*, pp. 95-106, 2016.
- [2] R. A. Khoerul Anam, "Analisis Performa Jaringan Software Defined Network Berdasarkan Penggunaan Cost Pada Protokol Ruting Open Shortest Path First," *CITEE*, pp. 1-8, 2017.
- [3] J. N. Yudianto, 2007.
- [4] K. P. S. D. J. H. S. D. M. O. K. Evangelos Haleplidis, "Software-Defined Networking (SDN): Layers and Architecture Terminology," *Software-Defined Networking (SDN): Layers and Architecture Terminology*, pp. 2070-1721, 2015.
- [5] 12 12 2015. [Online]. Available: gartner.com.
- [6] "Open Networking Foundation," 04 November 2013. [Online].
- [7] A. Ahmad Heryanto, "SOFTWARE DEFINED NETWORK MENGGUNAKAN SIMULATOR MININET," *Konferensi Nasional Teknologi Informasi dan Aplikasi*, pp. 1-4, 2016.
- [8] A. H. Iwan Iskandar, "Analisa Quality of Service (QoS) Jaringan Internet Kampus," *Jurnal CoreIT, Vol.1*, pp. 68-69, 2015.
- [9] B. S. Rikie Kartadie, "UJI PERFORMA KONTROLER FLOODLIGHT DAN OPENDAYLIGHT," *Seminar Nasional Teknologi Informasi dan Multimedia 2015*, pp. 1-2, 2015.
- [10] N. Y. R. G. A. Gelberger, "Performance Analists of Software-Defined Network (SDN)," *MOdeling, Anlaysia & Simulation of Computer and Telecommunication System (MASCOTS)*, pp. 389-393, 2013.
- [11] P. P. R. B. a. B. M. A. Banjar, "Analysing the Performance of the Openflow Standard for Software-Defined Networking Using the OMNet++ Network Simulator," *Computer Aided System Engineering (APCASE)*, pp. 31-37, 2014.
- [12] M. B. m. a. R. B. F. Benamrane, "Performances of Openflow-Based Software defined networks: An overview," *Journal Of Networks vol. 10*, pp. 329-337, 2015.
- [13] U. .. D. D. a. I. Z. B. R. K. Jha, "Performance Analysis of Proposed Openflow Based Network Architecture Using Mininet," *Wireless Personal Communications*, pp. 1-16, 2015.

- [14] M. T., "Mininet Overview," 23 November 2017. [Online]. Available: <http://Mininet.org/overview/>.
- [15] "Project Floodlight," 9 November 2017. [Online]. Available: <http://www.projectFloodlight.org/Floodlight/>.
- [16] "Predicting SD-WAN Adoption," 23 November 2017. [Online]. Available: gartner.com.
- [17] N. L. I. R. G. G. Slavica Tomovic, "A new approach to dynamic routing in SDN networks," Proceedings of the 18th Mediterranean Electrotechnical Conference MELECON, pp. 18-20, 2016.
- [18] M. K. M. K. K. Xenofon Foukas, Software Defined Networking Concepts, Edinburgh: The University of Edinburgh, 2015, pp. 1-29.
- [19] ONF, "Software-Defined Networking: The New Norm for Networks," 5 Oktober 2015. [Online]. Available: <https://www.opennetworking.org/images/stories/downloads/sdnresources/whitepapers/wp-sdnnewnorm.pdf>.
- [20] M. Gates, Iperf User Docs, DAST, 2003.
- [21] T. S. a. R. F. I. Y. K. Ningsih, "Analisis Quality Of Service (QoS) pada Simulasi Jaringan Multiprotocol Label *Switching* Virtual Private Network (MPLS VPN)," JETri, vol. 3, pp. 33-48, 2004.
- [22] S. Alam, "Perbedaan Jaringan Tradisional dan Jaringan SDN," 21 November 2017. [Online]. Available: <https://vlog-net.blogspot.co.id/2016/09/perbedaan-jaringan-tradisional-dan.html>.
- [23] T. A. N. McKeown and G. P. L. P. J. R. S. S. a. J. T. H. Balakrishnan, "Openflow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communication Vol. 38*, pp. 69-74, 2008.

LAMPIRAN

Lampiran 1

Pseudocode Topologi Jaringan SDN

```
#!/usr/bin/phyton
```

```
from mininet.net import Mininet  
from mininet.node import Controller, RemoteController, OVSController  
from mininet.node import CPULimitedHost, Host, Node  
from mininet.node import OVSKernelSwitch, UserSwitch  
from mininet.node import IVSSwitch  
from mininet.cli import CLI  
from mininet.link import TCLink. Intf  
from subprocess import call
```

```
def myNetwork():
```

```
    net = Mininet( topo=None,  
                build=False,  
                ipBase='10.0.0.0/8')  
  
    info( '*** Adding Controller\n' )  
    c0=net.addController(name='c0',  
                        Controller=Controller,  
                        protocol='tcp',  
                        port=6633)
```

```
    info( '*** Add switches\n' )  
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)  
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
```

```
s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
```

```
info( '*** Add hosts\n')
```

```
h5 = net.addhost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)
```

```
h3 = net.addhost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
```

```
h1 = net.addhost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
```

```
h2 = net.addhost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
```

```
h4 = net.addhost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
```

```
h6 = net.addhost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)
```

```
h7 = net.addhost('h7', cls=Host, ip='10.0.0.7', defaultRoute=None)
```

```
info( '*** Add links\n')
```

```
net.addlinks(s2, s1)
```

```
net.addlinks(s1, h1)
```

```
net.addlinks(s2, h2)
```

```
net.addlinks(s1, h3)
```

```
net.addlinks(s2, h4)
```

```
net.addlinks(s1, h5)
```

```
net.addlinks(s2, h6)
```

```
net.addlinks(s2, s4)
```

```
net.addlinks(s4, h7)
```

```
info( '*** Starting network\n')
```

```
net.build()
```

```
info( '*** Starting Controllers\n')
```

```
for Controller in net.Controllers:
```

```
    Controller.start()
```

```
info( '*** Starting switches\n')
```

```
net.get('s2').start([c0])
```

```
net.get('s4').start([c0])
```

```
net.get('s1').start([c0])
```

```
info( '*** Post configure switches and hosts\n')
```

```
CLI(net)
```

```
net.stop()
```

```
if __name__ == '__main__':
```

```
    setloglevel( 'info' )
```

```
    mynetwork()
```

Lampiran 2

Pseudocode Topologi Jaringan Konvensional

```
#!/usr/bin/phyton
```

```
from mininet.net import Mininet
```

```
from mininet.node import Controller, RemoteController, OVSController
```

```
from mininet.node import CPULimitedHost, Host, Node
```

```
from mininet.node import OVSKernelSwitch, UserSwitch
```

```
from mininet.node import IVSSwitch
```

```
from mininet.cli import CLI
```

```
from mininet.link import TCLink. Intf
```

```
from subprocess import call
```

```
def myNetwork():
```

```

net = Mininet( topo=None,
               build=False,
               ipBase='10.0.0.0/8')

info( '*** Adding Controller\n' )
info( '*** Add switches\n' )

s1 = net.addSwitch('s1', cls=OVSKernelSwitch, failMode='standalone')
s2 = net.addSwitch('s2', cls=OVSKernelSwitch, failMode='standalone')
s3 = net.addSwitch('s3', cls=OVSKernelSwitch, failMode='standalone')

info( '*** Add hosts\n' )

h5 = net.addhost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)
h1 = net.addhost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
h2 = net.addhost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
h3 = net.addhost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
h4 = net.addhost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
h6 = net.addhost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)
h7 = net.addhost('h7', cls=Host, ip='10.0.0.7', defaultRoute=None)

info( '*** Add links\n' )

net.addlinks(s2, s3)
net.addlinks(s3, h7)
net.addlinks(s1, h3)
net.addlinks(s2, h4)
net.addlinks(s2, h5)
net.addlinks(s2, h6)
net.addlinks(s1, h1)
net.addlinks(s1, h2)

```



```
net.addlinks(s1, s2)
```

```
info( '*** Starting network\n')
```

```
net.build()
```

```
info( '*** Starting Controllers\n')
```

```
for Controller in net.Controllers:
```

```
    Controller.start()
```

```
info( '*** Starting switches\n')
```

```
net.get('s1').start([])
```

```
net.get('s3').start([])
```

```
net.get('s2').start([])
```

```
info( '*** Post configure switches and hosts\n')
```

```
CLI(net)
```

```
net.stop()
```

```
if __name__ == '__main__':
```

```
    setloglevel( 'info' )
```

```
    mynetwork()
```

Lampiran 3

Hasil Latency

SDN							
Uji ke -	64	128	256	512	1024	2048	4096
1	0.110	0.113	0.113	0.111	0.101	0.136	0.144
2	1.116	0.115	0.109	0.105	0.111	0.116	0.149
3	0.108	0.113	1.09	0.107	1.069	0.117	0.148
4	0.122	0.114	0.105	0.113	0.108	0.133	0.15
5	0.104	0.114	0.105	0.11	0.105	0.135	0.143
6	0.112	0.114	0.093	0.103	0.106	0.135	0.121
7	0.129	0.141	0.106	0.102	0.108	0.134	0.145
8	0.112	0.119	0.108	0.104	0.11	0.135	0.145
9	0.111	0.117	0.192	3.129	0.103	0.138	0.146
10	0.589	0.114	0.107	0.111	0.095	0.136	0.122
11	0.528	0.116	0.105	0.104	0.109	0.137	0.152
12	0.603	0.112	0.093	0.103	0.107	0.133	0.125
13	0.106	0.116	0.106	0.109	0.107	0.136	0.153
14	0.104	0.116	0.108	0.138	0.105	0.119	0.129
15	0.106	0.115	0.107	0.106	0.106	0.118	0.147
16	0.107	0.114	0.111	0.103	0.106	0.138	0.146
17	0.574	0.114	0.112	0.106	0.109	0.134	0.148
18	0.944	0.116	0.116	0.107	0.103	0.138	0.128
19	1.171	0.114	0.104	0.107	0.105	0.136	0.126
20	0.125	0.123	1.005	0.106	0.139	0.138	0.155
21	0.105	0.194	0.103	0.141	3.258	0.12	0.125
22	0.106	0.115	0.155	0.11	0.111	0.139	0.127
23	0.102	0.115	0.103	0.105	0.104	0.136	0.126
24	0.101	0.11	0.108	0.104	0.095	0.133	0.13
25	0.103	0.113	0.425	0.107	0.106	0.135	0.133
Rata- Rata	0.300	0.119	0.200	0.230	0.271	0.132	0.139

	KONVENSSIONAL						
Uji ke -	64	128	256	512	1024	2048	4096
1	0.095	0.108	0.104	0.111	0.106	0.111	0.119
2	0.108	0.093	0.092	0.112	0.106	0.112	0.112
3	0.09	0.107	0.102	0.109	0.112	0.112	0.12
4	0.091	0.104	0.091	0.107	0.09	0.111	0.117
5	0.092	0.095	0.092	0.089	0.093	0.112	0.112
6	0.089	0.102	0.104	0.108	0.097	0.116	0.114
7	0.093	0.105	0.105	0.11	0.169	0.111	0.114
8	0.09	0.107	0.105	0.091	0.105	0.114	0.115
9	0.108	0.089	0.086	0.091	0.089	0.112	0.114
10	0.093	0.093	0.104	0.104	0.105	0.118	0.114
11	0.089	0.104	0.109	0.109	0.094	0.115	0.112
12	0.104	0.091	0.107	0.09	0.122	0.114	0.119
13	0.103	0.09	0.089	0.096	0.096	0.118	0.115
14	0.094	0.091	0.094	0.094	0.09	0.118	0.118
15	0.091	0.093	0.106	0.093	0.091	0.115	0.117
16	0.093	0.089	0.103	0.109	0.108	0.113	0.114
17	0.103	0.091	0.102	0.104	0.108	0.112	0.115
18	0.107	0.093	0.102	0.091	0.11	0.13	0.114
19	0.103	0.091	0.103	0.104	0.11	0.114	0.113
20	0.091	0.09	0.102	0.105	0.092	0.114	0.114
21	0.105	0.09	0.107	0.09	0.111	0.133	0.128
22	0.093	0.091	0.106	0.106	0.093	0.131	0.112
23	0.103	0.092	0.103	0.107	0.109	0.131	0.115
24	0.09	0.106	0.109	0.107	0.091	0.115	0.117
25	0.116	0.092	0.106	0.102	0.092	0.115	
Rata-Rata	0.0973 6	0.0958 8	0.1013 2	0.1015 6	0.1035 6	0.1166 8	0.115583 3

SDN 1 Controller							
No.	64	128	256	512	1024	2048	4096
1	0.811	0.200	0.233	0.129	0.148	0.154	0.167
2	0.129	0.126	0.129	0.145	0.129	0.157	0.175
3	0.135	0.132	0.138	0.130	0.124	0.190	0.148
4	0.139	0.129	0.131	0.129	0.136	0.170	0.163
5	0.128	0.129	0.132	0.151	0.134	0.257	0.199
6	0.148	0.131	0.117	0.151	0.130	0.216	0.175
7	0.129	0.124	0.134	0.127	0.131	0.132	0.151
8	0.158	0.126	0.120	0.136	0.127	0.138	0.169
9	0.136	0.128	0.129	0.140	0.134	0.169	0.179
10	0.139	0.128	0.126	0.150	0.131	0.176	0.180
11	0.887	0.124	0.129	0.132	0.132	0.139	0.181
12	0.550	0.126	0.126	0.127	0.131	0.148	0.184
13	0.133	0.127	0.130	0.128	0.129	0.156	0.181
14	0.129	0.133	0.147	0.126	0.130	0.158	0.163
15	0.152	0.126	0.127	0.194	0.128	0.157	0.215
16	0.133	0.120	0.147	0.131	0.130	0.156	0.178
17	0.130	0.128	0.150	0.127	0.134	0.176	0.175
18	0.134	0.124	0.131	0.140	0.130	0.145	0.158
19	0.145	0.122	0.128	0.127	0.142	0.169	0.145
20	0.149	0.126	0.134	0.123	0.127	0.157	0.147
21	0.131	0.130	0.147	0.129	0.132	0.137	0.177
22	0.125	0.129	0.131	0.127	0.132	0.146	0.175
23	0.128	0.125	0.127	0.133	0.130	0.139	0.155
24	0.131	0.133	0.108	0.132	0.127	0.137	0.175
25	0.127	0.124	0.103	0.125	0.129	0.133	0.131
	0.209	0.130	0.134	0.136	0.131	0.160	0.170

SDN 2 Controller							
No.	64	128	256	512	1024	2048	4096
1	0.163	0.122	0.120	0.132	0.125	0.176	0.214
2	0.126	0.140	0.119	0.138	0.130	0.179	0.186
3	0.130	0.136	0.120	0.148	0.122	0.162	0.145
4	0.123	0.137	0.149	0.123	0.126	0.171	0.155
5	0.126	0.147	0.139	0.155	0.125	0.165	0.182
6	0.126	0.142	0.149	0.129	0.124	0.153	0.214
7	0.124	0.149	0.123	0.151	0.145	0.151	0.177
8	0.129	0.138	0.146	0.141	0.124	0.163	0.191
9	0.124	0.138	0.167	0.141	0.127	0.196	0.164
10	0.127	0.130	0.124	0.157	0.139	0.144	0.152
11	0.124	0.123	0.133	0.144	0.142	0.187	0.215
12	0.116	0.124	0.159	0.141	0.140	0.151	0.183
13	0.123	0.132	0.138	0.141	0.126	0.170	0.186
14	0.134	0.117	0.142	0.155	0.189	0.169	0.277
15	0.132	0.119	0.142	0.121	0.131	0.157	0.177
16	0.140	0.140	0.149	0.123	0.126	0.144	0.178
17	0.137	0.139	0.120	0.141	0.132	0.147	0.163
18	0.130	0.139	0.119	0.129	0.140	0.143	0.186
19	0.137	0.141	0.124	0.141	0.144	0.154	0.174
20	0.139	0.137	0.138	0.121	0.127	0.154	0.185
21	0.122	0.141	0.135	0.125	0.126	0.157	0.191
22	0.188	0.139	0.124	0.123	0.123	0.166	0.159
23	0.119	0.122	0.133	0.125	0.137	0.169	0.174
24	0.112	0.124	0.111	0.153	0.148	0.184	0.168
25	0.125	0.143	0.108	0.158	0.189	0.162	0.160
	0.131	0.134	0.133	0.138	0.136	0.163	0.182

Throughput

SDN					
Uji ke -	64	128	256	512	1024
1	13.52	13.04	13.52	11.84	9.04
2	15.04	15.12	13.84	12.32	8.96
3	14.64	13.12	13.84	10.8	9.52
4	13.92	15.04	11.12	11.2	9.52
5	14	15.2	13.84	11.68	8.88
6	14.48	14.56	14.16	11.92	9.12
7	14.4	15.12	13.6	10.64	9.12
8	14.48	14.72	13.44	12.16	8.96
9	14.48	14.24	12.32	12.32	8.72
10	14.16	14.56	13.68	11.84	9.04
11	14.4	14.56	14.08	11.68	9.44
12	14	15.12	13.76	11.2	9.2
13	13.2	15.36	13.76	11.44	9.12
14	13.2	14.88	13.6	11.52	8.96
15	14	14.8	13.28	11.44	9.28
16	13.6	15.12	13.04	11.36	9.28
17	13.92	14.72	13.2	11.28	9.12
18	14	14.4	13.2	11.76	8.72
19	13.76	14.64	12.32	11.12	8.88
20	14.32	14.72	13.04	11.92	9.04
21	13.84	14.16	13.76	11.52	9.12
22	13.68	14.32	13.84	11.52	8.88
23	13.76	14.8	13.28	11.44	8.88
24	13.68	14.48	13.2	9.76	7.56
25	13.84	14	12.56	12.08	9.12
Rata-Rata	14.0128	14.592	13.3312	11.5104	9.0192

KONVENSIONAL					
Uji ke -	64	128	256	512	1024
1	13.76	16.32	14.32	12.96	9.36
2	14.8	14.88	13.52	13.12	9.6
3	13.52	13.92	13.52	12	9.12
4	13.84	14.24	13.92	11.76	8.96
5	13.76	14.48	13.6	11.6	9.44
6	13.04	14.4	13.36	11.84	9.12
7	12	14.32	13.68	11.36	9.04
8	13.12	14.08	13.68	11.76	8.96
9	13.84	13.84	12.56	11.68	8.96
10	13.44	14.16	13.2	11.6	8.88
11	13.68	14.24	12.88	11.92	9.12
12	13.6	14.08	12.96	11.44	9.68
13	13.44	14.48	13.68	11.76	9.28
14	13.52	13.84	13.36	11.92	8.96
15	12.88	14.08	12.96	11.84	9.04
16	13.2	14.64	13.6	12.08	9.2
17	13.12	14.64	13.52	11.28	9.28
18	13.12	14.56	14	11.92	9.28
19	12.88	14	13.28	10.72	9.28
20	12.96	14.32	13.04	11.36	9.04
21	13.12	14.64	12.88	11.76	8.88
22	13.04	14.48	12.88	11.68	8.96
23	14.56	14.32	14.48	11.68	8.64
24	14.8	14	12.8	12.08	8.88
25	12.88	17.8	14.16	12.24	8.8
Rata-Rata	13.4368	14.5104	13.4336	11.8144	9.1104

SDN 1 Controller				
64	128	256	512	1024
13.368	13.264	10.544	12.632	9.248
9.600	15.184	12.856	12.832	9.056
14.304	14.992	12.680	10.160	9.088
14.096	14.696	12.416	12.320	9.056
13.040	14.376	13.296	12.728	9.144
13.096	15.184	13.584	12.688	9.136
14.176	12.720	13.304	12.672	9.312
14.424	13.048	13.056	12.824	9.248
13.800	12.816	12.328	12.800	9.048
14.584	14.640	12.144	12.704	9.528
13.480	14.744	12.824	12.000	9.456
13.824	15.392	13.016	12.632	9.392
12.448	14.912	13.232	13.024	9.480
14.008	15.200	13.256	12.744	9.376
13.000	15.696	13.424	12.136	9.456
13.040	14.424	12.312	11.488	9.184
14.152	15.888	12.128	12.048	9.416
13.680	14.992	12.392	12.568	9.376
14.224	14.336	12.328	12.368	9.400
13.712	13.840	13.400	12.192	9.424
13.936	14.960	12.784	11.648	9.376
13.680	13.680	13.256	12.488	9.568
13.848	14.288	12.808	12.656	9.024
13.280	14.584	13.456	11.728	9.312
13.976	14.192	13.216	12.712	9.560
13.551	14.482	12.802	12.352	9.307

SDN 2 Controller				
64	128	256	512	1024
12.112	15.272	12.688	12.144	9.480
12.184	14.880	13.160	13.168	8.800
12.392	13.760	12.080	12.336	9.472
13.112	7.512	12.336	12.760	9.480
12.048	16.064	12.456	12.040	9.440
14.248	15.848	12.672	12.352	9.496
13.968	15.968	13.024	12.496	9.480
13.600	15.768	12.576	11.456	9.288
13.904	15.424	12.232	12.080	9.424
13.832	15.632	12.248	12.024	9.520
14.056	15.704	12.584	12.048	8.888
10.912	15.496	12.832	12.296	8.816
13.944	14.448	12.552	12.904	8.992
13.808	15.296	14.080	12.296	9.536
14.376	15.696	12.512	12.296	9.384
13.584	14.664	12.624	12.504	9.288
14.368	13.984	12.496	12.528	9.272
13.984	13.784	12.736	11.752	9.504
14.288	13.680	12.800	12.456	9.552
13.832	13.800	12.808	12.280	9.480
13.760	13.504	13.288	11.872	9.288
13.960	13.120	13.024	12.368	9.488
13.544	13.984	12.912	12.448	8.920
14.040	13.232	13.520	12.032	8.840
14.160	13.080	12.720	11.872	9.176
13.521	14.384	12.758	12.272	9.292

Konvensional				
64	128	256	512	1024
14.000	13.584	14.176	12.376	8.800
13.440	15.160	13.832	12.080	9.488
14.488	14.888	14.360	12.240	9.448
14.312	14.488	13.112	12.272	8.800
13.904	15.488	13.616	12.792	9.552
13.560	14.800	14.112	11.488	8.744
13.592	14.912	12.816	13.192	9.144
14.376	14.984	12.936	13.392	9.600
14.992	14.728	12.864	12.816	10.120
13.176	15.000	13.008	13.168	9.272
13.272	15.016	13.040	13.224	9.488
13.280	14.624	13.560	12.336	9.440
13.760	14.176	13.128	12.264	8.976
13.216	14.144	13.032	13.216	9.024
13.784	13.528	12.504	13.216	9.224
13.648	14.776	12.672	13.344	9.792
13.560	14.840	12.528	13.272	10.056
13.904	13.800	12.080	12.336	10.144
13.016	14.232	12.472	11.560	10.176
14.192	15.160	12.712	11.264	9.880
14.128	15.120	12.632	11.632	9.472
13.920	15.152	12.624	11.752	9.160
14.032	14.568	12.256	11.280	9.448
14.352	14.408	12.128	11.808	9.440
13.080	14.848	12.312	11.736	8.800
13.799	14.657	12.980	12.402	9.420

Jitter

SDN					
Uji ke -	64	128	256	512	1024
1	0,0317	0,2203	0,0017	0,0017	0,298
2	0,0013	0,041	0,0123	0,002	0,334
3	0,0013	0,001	0,0023	0,0623	0,0017
4	0,0013	0,0203	0,0023	0,0017	0,0027
5	0,021	0,0033	0,0037	0,0727	0,213
6	0,0183	0,0037	0,001	0,135	0,0013
7	0,0013	0,002	0,0033	0,0317	0,006
8	0,023	0,0683	0,0423	0,0017	0,4397
9	0,0013	0,0547	0,059	0,142	0,238
10	0,002	0,002	0,027	0,1707	0,2747
11	0,002	0,0077	0,001	0,2383	0,2813
12	0,0033	0,0043	0,04	0,052	0,1427
13	0,003	0,004	0,0013	0,0217	0,001
14	0,0013	0,0023	0,0097	0,1487	0,4877
15	0,0873	0,027	0,0013	0,0013	0,242
16	0,0103	0,0017	0,0023	0,0777	0,1293
17	0,023	0,0023	0,0023	0,111	0,1457
18	0,001	0,0013	0,0017	0,0427	0,371
19	0,1397	0,0117	0,0547	0,0027	0,002
20	0,0017	0,0323	0,065	0,1417	0,1373
21	0,0013	0,002	0,0253	0,001	0,1413
22	0,0017	0,0027	0,0613	0,0337	0,004
23	0,0343	0,002	0,0297	0,1363	0,1643
24	0,001	0,0027	0,0703	0,0017	0,613
25	0,027	0,0033	0,0803	0,0023	0,1797
Rata-Rata	0,0176	0,021	0,0241	0,0654	0,1941

KONVENSIONAL					
Uji ke -	64	128	256	512	1024
1	0,0013	0,0033	0,0017	0,003	0,02
2	0,0013	0,0637	0,0017	0,2443	0,002
3	0,0027	0,1017	0,0017	0,01	0,0027
4	0,0017	0,0023	0,001	0,0943	0,267
5	0,001	0,0027	0,031	0,0027	0,3333
6	0,001	0,0017	0,0013	0,002	0,031
7	0,0017	0,002	0,0017	0,0037	0,0017
8	0,001	0,26	0,0023	0,0093	0,2913
9	0,0033	0,0173	0,2493	0,0017	0,204
10	0,0013	0,001	0,1633	0,0013	0,0023
11	0,0023	0,0063	0,291	0,2423	0,0027
12	0,0033	0,002	0,1063	0,1937	0,0693
13	0,0017	0,055	0,001	0,5327	0,0073
14	0,0013	0,002	0,0013	0,2603	0,002
15	0,0017	0,002	0,0023	0,002	0,0017
16	0,0023	0,1017	0,0013	0,001	0,0013
17	0,0027	0,002	0,0013	0,114	0,2033
18	0,0013	0,0223	0,002	0,1547	0,0027
19	0,0013	0,001	0,0013	0,0013	0,0123
20	0,0013	0,002	0,1693	0,0337	0,3707
21	0,0637	0,002	0,002	0,2847	0,0013
22	0,002	0,002	0,2703	0,002	0,4103
23	0,002	0,0013	0,2733	0,0013	0,0993
24	0,001	0,0013	0,4677	0,0017	0,123
25	0,0013	0,001	0,0017	0,1973	0,0013
Rata-Rata	0,0042	0,0264	0,0819	0,0958	0,0986

SDN 1 Controller					
	64	128	256	512	1024
1	0.0410	0.0476	0.3305	0.0026	0.4470
2	0.0410	0.0020	0.0615	0.0185	0.2010
3	0.0340	0.0020	0.0015	0.0035	0.0026
4	0.0420	0.0020	0.0305	0.0035	0.0041
5	0.0350	0.0315	0.0050	0.0056	0.1695
6	0.0420	0.0275	0.0056	0.0015	0.0020
7	0.0430	0.0020	0.0030	0.0050	0.0090
8	0.0480	0.0345	0.1025	0.0635	0.2096
9	0.0420	0.0020	0.0821	0.0885	0.2070
10	0.0460	0.0030	0.0030	0.0405	0.1121
11	0.0450	0.0030	0.0116	0.0015	0.1220
12	0.0460	0.0050	0.0065	0.0600	0.0641
13	0.0410	0.0045	0.0060	0.0020	0.0015
14	0.0380	0.0020	0.0035	0.0146	0.2816
15	0.0400	0.1310	0.0405	0.0020	0.0630
16	0.0470	0.0155	0.0026	0.0035	0.1940
17	0.0430	0.0345	0.0035	0.0035	0.1736
18	0.0015	0.0015	0.0020	0.0026	0.2565
19	0.2096	0.2096	0.0176	0.0821	0.0030
20	0.0026	0.0026	0.0485	0.0975	0.2060
21	0.0020	0.0020	0.0030	0.0380	0.0620
22	0.0026	0.0026	0.0041	0.0920	0.0060
23	0.0515	0.0515	0.0030	0.0446	0.0965
24	0.0015	0.0015	0.0041	0.1055	0.0195
25	0.0405	0.0405	0.0050	0.1205	0.2696
Rata-Rata	0.0410	0.0264	0.0314	0.0361	0.1273

SDN2 topo					
	64	128	256	512	1024
1	0.0317	0.2203	0.0017	0.0017	0.2980
2	0.0013	0.0410	0.0123	0.0020	0.1340
3	0.0013	0.0010	0.0023	0.0623	0.0017
4	0.0013	0.0203	0.0023	0.0017	0.0027
5	0.0210	0.0033	0.0037	0.0727	0.1130
6	0.0183	0.0037	0.0010	0.1350	0.0013
7	0.0013	0.0020	0.0033	0.0317	0.0060
8	0.0230	0.0683	0.0423	0.0017	0.1397
9	0.0013	0.0547	0.0590	0.1420	0.1380
10	0.0020	0.0020	0.0270	0.1707	0.0747
11	0.0020	0.0077	0.0010	0.2383	0.0813
12	0.0033	0.0043	0.0400	0.0520	0.0427
13	0.0030	0.0040	0.0013	0.0217	0.0010
14	0.0013	0.0023	0.0097	0.1487	0.1877
15	0.0873	0.0270	0.0013	0.0013	0.0420
16	0.0103	0.0017	0.0023	0.0777	0.1293
17	0.0230	0.0023	0.0023	0.1110	0.1157
18	0.0010	0.0013	0.0017	0.0427	0.1710
19	0.1397	0.0117	0.0547	0.0027	0.0020
20	0.0017	0.0323	0.0650	0.1417	0.1373
21	0.0013	0.0020	0.0253	0.0010	0.0413
22	0.0017	0.0027	0.0613	0.0337	0.0040
23	0.0343	0.0020	0.0297	0.1363	0.0643
24	0.0010	0.0027	0.0703	0.0017	0.0130
25	0.0270	0.0033	0.0803	0.0023	0.1797
Rata-Rata	0.0176	0.0210	0.0240	0.0654	0.0849

KONVENSIONAL					
	64	128	256	512	1024
1	0.0013	0.0033	0.0017	0.0030	0.0200
2	0.0013	0.0637	0.0017	0.2443	0.0020
3	0.0027	0.1017	0.0017	0.0100	0.0027
4	0.0017	0.0023	0.0010	0.0943	0.2670
5	0.0010	0.0027	0.0310	0.0027	0.3333
6	0.0010	0.0017	0.0013	0.0020	0.0310
7	0.0017	0.0020	0.0017	0.0037	0.0017
8	0.0010	0.2600	0.0023	0.0093	0.2913
9	0.0033	0.0173	0.2493	0.0017	0.2040
10	0.0013	0.0010	0.1633	0.0013	0.0023
11	0.0023	0.0063	0.2910	0.2423	0.0027
12	0.0033	0.0020	0.1063	0.1937	0.0693
13	0.0017	0.0550	0.0010	0.5327	0.0073
14	0.0013	0.0020	0.0013	0.2603	0.0020
15	0.0017	0.0020	0.0023	0.0020	0.0017
16	0.0023	0.1017	0.0013	0.0010	0.0013
17	0.0027	0.0020	0.0013	0.1140	0.2033
18	0.0013	0.0223	0.0020	0.1547	0.0027
19	0.0013	0.0010	0.0013	0.0013	0.0246
20	0.0013	0.0020	0.1693	0.0337	0.3707
21	0.0637	0.0020	0.0020	0.2847	0.0013
22	0.0020	0.0020	0.2703	0.0020	0.4103
23	0.0020	0.0013	0.2733	0.0013	0.0993
24	0.0010	0.0013	0.4677	0.0017	0.1230
25	0.0013	0.0010	0.0017	0.1973	0.0013
Rata-Rata	0.0042	0.0264	0.0819	0.0958	0.0990

Packet loss

SDN					
Uji ke -	64	128	256	512	1024
1	0.0820	0.0085	0.0352	0.0335	0.0054
2	0.1327	0.0366	0.0070	0.0337	0.0080
3	0.0850	0.0627	0.0387	0.0157	0.0100
4	0.0291	0.0261	0.0055	0.0257	0.0048
5	0.1517	0.0447	0.0213	0.0262	0.0158
6	0.0221	0.0780	0.0460	0.0156	0.0097
7	0.0069	0.0313	0.0173	0.0377	0.0068
8	0.0406	0.0127	0.0080	0.0174	0.0072
9	0.1652	0.0523	0.0126	0.0037	0.0038
10	0.1190	0.0173	0.0570	0.0089	0.0168
11	0.0483	0.1547	0.0443	0.0058	0.0253
12	0.0502	0.0393	0.0490	0.0190	0.0057
13	0.0547	0.2367	0.0238	0.0059	0.0014
14	0.1030	0.0208	0.0067	0.0060	0.0063
15	0.0560	0.0826	0.0474	0.0043	0.0060
16	0.1243	0.0737	0.0233	0.0151	0.0242
17	0.0730	0.0253	0.0314	0.0207	0.0115
18	0.0547	0.0341	0.0986	0.0095	0.0197
19	0.0287	0.2056	0.0286	0.0243	0.0069
20	0.0945	0.0907	0.0056	0.0214	0.0076
21	0.1687	0.0064	0.0523	0.0307	0.0086
22	0.1500	0.0767	0.0299	0.0246	0.0101
23	0.3683	0.0357	0.0380	0.0137	0.0080
24	0.1073	0.0330	0.0261	0.0136	0.0061
25	0.0570	0.0540	0.0354	0.0161	0.0049
Rata-Rata	0.09492	0.06158	0.03156	0.01795	0.00962

KONVENSSIONAL					
Uji ke -	64	128	256	512	1024
1	0.02322	0.00198	0.00033	0.00080	0.00111
2	0.00022	0.00639	0.00103	0.00170	0.00048
3	0.00166	0.00213	0.00114	0.00049	0.00139
4	0.0022	0.00291	0.00984	0.00309	0.00000
5	0.00065	0.00214	0.00144	0.00422	0.00168
6	0.0005	0.00569	0.0022	0.00013	0.00000
7	0.00306	0.00111	0.00224	0.00061	0.00084
8	0.00036	0.00126	0.00491	0.00026	0.00085
9	0.00021	0.00038	0.00166	0.00192	0.00120
10	0.00231	0.00122	0.00048	0.00168	0.00028
11	0.00086	0.00222	0.00125	0.00073	0.00000
12	0.00194	0.00101	0.00061	0.00115	0.00228
13	0.00174	0.00043	0.00111	0.00165	0.00093
14	0.000065	0.00114	0.00028	0.00307	0.00433
15	0.00046	0.00202	0.00121	0.00070	0.00029
16	0	0.00135	0.00134	0.00000	0.00112
17	0.00027	0.00061	0.00036	0.00049	0.00071
18	0.00142	0.00223	0.00203	0.00337	0.00060
19	0.00036	0.00317	0.00108	0.00253	0.00155
20	0.00274	0.00424	0.00134	0.00052	0.00142
21	0.00766	0.00059	0.00145	0.00102	0.00014
22	0.00148	0.00213	0.00405	0.00066	0.00000
23	0.00349	0.00237	0.00196	0.00151	0.00000
24	0.00207	0.0025	0.00111	0.00106	0.00046
25	0.00137	0.00014	0.0044	0.00130	0.00141
Rata-Rata	0.002413	0.00205	0.00195	0.00139	0.00092

SDN 1 Controller					
	64	128	256	512	1024
1	0.2367	0.1580	0.3800	0.0443	0.0523
2	0.3200	0.4100	0.4000	0.0377	0.0115
3	0.1567	0.1117	0.2367	0.0457	0.0742
4	0.1767	0.1700	0.1000	0.1420	0.0373
5	0.4333	0.3000	0.0763	0.0410	0.0245
6	0.3600	0.2933	0.1633	0.0793	0.1067
7	0.1510	0.3400	0.2533	0.0410	0.0813
8	0.1167	0.1850	0.1913	0.0720	0.1100
9	0.5667	0.3550	0.1077	0.1083	0.0126
10	0.3833	0.2600	0.1260	0.0567	0.0670
11	0.3700	0.1083	0.1350	0.0530	0.0537
12	0.1020	0.1633	0.1133	0.0570	0.0890
13	0.1310	0.1393	0.0990	0.0797	0.0837
14	0.1650	0.1253	0.1010	0.0507	0.0398
15	0.3310	0.1063	0.1047	0.0260	0.0580
16	0.1933	0.1867	0.1350	0.0613	0.0700
17	0.1343	0.2367	0.1533	0.0313	0.0449
18	0.2100	0.1667	0.0963	0.1363	0.0720
19	0.1643	0.2033	0.0993	0.0580	0.0947
20	0.4200	0.1120	0.0933	0.0717	0.0793
21	0.5900	0.1433	0.1040	0.0377	0.0650
22	0.6500	0.1057	0.0647	0.0797	0.0611
23	0.1490	0.1433	0.1570	0.0671	0.0654
24	0.1613	0.1667	0.1260	0.0597	0.0850
25	0.5800	0.1600	0.0923	0.0703	0.1100
Rata-Rata	0.2901	0.1940	0.1484	0.0634	0.0660

SDN 2 Controller					
	64	128	256	512	1024
1	0.0820	0.0085	0.0352	0.0335	0.0054
2	0.1327	0.0366	0.0070	0.0337	0.0080
3	0.0850	0.0627	0.0387	0.0157	0.0100
4	0.0291	0.0261	0.0055	0.0257	0.0048
5	0.1517	0.0447	0.0213	0.0262	0.0158
6	0.0221	0.0780	0.0460	0.0156	0.0097
7	0.0069	0.0313	0.0173	0.0377	0.0068
8	0.0406	0.0127	0.0080	0.0174	0.0072
9	0.1652	0.0523	0.0126	0.0037	0.0038
10	0.1190	0.0173	0.0570	0.0089	0.0168
11	0.0483	0.1547	0.0443	0.0058	0.0253
12	0.0502	0.0393	0.0490	0.0190	0.0057
13	0.0547	0.2367	0.0238	0.0059	0.0014
14	0.1030	0.0208	0.0067	0.0060	0.0063
15	0.0560	0.0826	0.0474	0.0043	0.0060
16	0.1243	0.0737	0.0233	0.0151	0.0242
17	0.0730	0.0253	0.0314	0.0207	0.0115
18	0.0547	0.0341	0.0986	0.0095	0.0197
19	0.0287	0.2056	0.0286	0.0243	0.0069
20	0.0945	0.0907	0.0056	0.0214	0.0076
21	0.1687	0.0064	0.0523	0.0307	0.0086
22	0.1500	0.0767	0.0299	0.0246	0.0101
23	0.3683	0.0357	0.0380	0.0137	0.0080
24	0.1073	0.0330	0.0261	0.0136	0.0061
25	0.0570	0.0540	0.0354	0.0161	0.0049
Rata-Rata	0.0949	0.0616	0.0316	0.0180	0.0096

KONVENSSIONAL					
	64	128	256	512	1024
1	0	0.00132	0.0005	0.00119	0.00166
2	0.00015	0.00426	0.00155	0.00255	0.00072
3	0.00111	0.00142	0.00171	0.00074	0.00209
4	0.00147	0.00194	0.01476	0.00463	0
5	0.00043	0.00143	0.00216	0.00633	0
6	0.00033	0.00379	0.0033	0.00019	0
7	0.00204	0.00074	0.00336	0.00091	0.00125
8	0.00024	0.00084	0.00737	0.00038	0.00127
9	0.00014	0.00025	0.00249	0.00289	0.0018
10	0.00154	0.00081	0.00072	0.00252	0.00043
11	0.00057	0.00148	0.00188	0.00109	0
12	0.00129	0.00067	0.00092	0.00172	0.00342
13	0.00116	0.00029	0.00167	0.00247	0.0014
14	0	0.00076	0.00042	0.00461	0.00649
15	0.00031	0.00135	0.00182	0.00105	0.00044
16	0	0.0009	0.00201	0	0.00168
17	0.00018	0.00041	0.00054	0	0.00106
18	0.00095	0.00149	0.00305	0.00506	0.0009
19	0.00024	0.00211	0.00162	0.0038	0.00232
20	0.00183	0.00283	0.00201	0.00078	0.00214
21	0.00511	0.00039	0.00218	0.00153	0.00021
22	0.00099	0.00142	0.00608	0.00098	0
23	0.00233	0.00158	0.00294	0.00227	0
24	0.00138	0.00167	0.00167	0.00159	0.00069
25	0.00091	0.00158	0.0066	0.00195	0.00211
Rata-Rata	0.0010	0.0014	0.0029	0.0021	0.0013