Analisis Efek Penggunaan Kontroler Floodlight dan OpenDaylight Pada Performansi Jaringan SDN

Analysis of Effect of Controller Floodlight and OpenDaylight On Software Defined Network

TUGAS AKHIR

Disusun sebagai syarat untuk memperoleh gelar Sarjana Teknik

pada Program Studi S1 Sistem Komputer

Universitas Telkom

Oleh

Krisandi Agusto

1104120081

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

BANDUNG

2018

	UNIVERSITAS TELKOM	No. Dokumen	TEL_U-AK-FEK-WD1-UAK-FMP-001/006
	Jl. Telekomunikasi No. 1 Ters. Buah Batu Bandung 40257	No. Revisi	00
	FORMULIR LEMBAR PENGESAHAN TUGAS AKHIR	Berlaku Efektif	04 Mei 2015

LEMBAR PENGESAHAN
TUGAS AKHIR

Analisis Efek Penggunaan Kontroler Floodlight dan OpenDaylight Pada Performansi Jaringan SDN

Analysis of Effect of Controller Floodlight and OpenDaylight On Software Defined Network

Disusun oleh :

KRISANDI AGUSTO

1104120081

Telah disetujui dan disahkan sebagai Tugas Akhir

Program S1 Sistem Komputer Fakultas Teknik Elektro Universitas Telkom

Bandung, April 2018

Disahkan oleh:

Pembimbing 1	Pembimbing 2
Ir. Agus Virgono., M.T.	**Drs. Ir. R. Rumani M., Bc.TT., M.Sc.**
NIP : 93660083-1	**NIP : 174710052-6**

LEMBAR PERNYATAAN ORISINALITAS

NAMA : Krisandi Agusto

NIM : 1104120081

ALAMAT : Hilia Banda , Kab. Solok, Kec. Kubung, Sumatera Barat

No. Tlp/HP : 0085668347417

E-mail : krisandia.ka@gmail.com

Menyatakan bahwa Tugas Akhir II ini merupakan karya orisinil saya sendiri dengan judul :

Analisis Efek Penggunaan Kontroler Floodlight dan OpenDaylight Pada Performansi Jaringan SDN

Analysis of Effect of Controller Floodlight and OpenDaylight On Software Defined Network

Atas pernyataan ini, saya siap menanggung resiko / sanksi yang dijatuhkan kepada saya apabila kemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, Juli 2018

Krisandi Agusto

1104120081

ABSTRAK

Pada jaringan tradisional, penggunaan antara *control plane* dan *forwarding plane* menjadi satu. Desain jaringan seperti ini dianggap kurang fleksibel dalam mengkontrol dan mengatur jaringan tersebut. Ada suatu sistem jaringan yang dianggap lebih efektif dari pada jaringan konvensional yang sudah ada yang disebut dengan *Software Defined Network* (SDN). SDN berfungsi untuk memisahkan secara eksplisit antara *control plane* dengan *data plane* supaya terjadi pemanajemenan jaringan yang lebih bagus. Dengan adanya sistematika ini pengelolaan jaringan akan lebih baik dan lebih fleksibel melalui kontroler yang bersifat *programmable.*

Ada banyak jenis kontroler dengan menggunakan bahasa pemrograman berbeda. Beberapa kontroler ada yang menggunakan bahasa pemrograman java dan sebagian menggunakan bahasa pemrograman python. Pada pengujian ini, kontroler yang digunakan adalah kontroler dengan bahasa pemrograman java. Adapun kontroler yang digunakan yaitu kontroler Floodlight tanpa algoritma Johnson, Foodlight dengan algoritma Johnson dan OpenDaylight. Ketiga model kontroler ini akan diuji performanya menggunakan parameter QoS dengan standarisasi ITU-T G.1010.

Hasil penelitian yang didapat kontroler Floodlight tanpa algoritma Johnson lebih unggul dibandingkan kedua kontroler lainnya dari untuk layanan data dan VoIP. Namun untuk pengiriman layanan video, kontroler OpenDaylight lebih baik dari yang lainnya. Karena hanya kontroler OpenDaylight yang bisa mengirimkan paket video. Untuk pengujian *resource utilization*, konsumsi memori tertinggi dimiliki oleh kontroler OpenDaylight. Berdasarkan standarisasi ITU-T G.1010 kontroler Floodlight tanpa algoritma Johnson, Floodlight menggunakan algoritma Johnson dan OpenDaylight hanya memenuhi standarisasi *delay* untuk layanan data saja.

Kata Kunci: Kontroller, Kinerja, Floodlight, OpenDaylight, QoS, *Resource Utilization*

ABSTRACT

In traditional networks, the use between control plane and forwarding plane becomes one. Network design like this is considered to be less flexible in controlling and managing the network. There is a network system that is considered more effective than existing conventional networks called Software Defined Network (SDN). SDN serves to separate explicitly between the control plane and the data plane so that there is better network management. With this systematics network management will be better and more flexible through programmable controllers.

There are many types of controllers using different programming languages. Some controllers use the Java programming language and some use the Python programming language. In this test, the controller used is a controller with a java programming language. The controllers used are Floodlight controllers without Johnson algorithm, Foodlight with Johnson algorithm and OpenDaylight. These three controller models will be tested for their performance using QoS parameters with ITU-T G.1010 standardization.

The results obtained by the Floodlight controller without the Johnson algorithm are superior to the other two controllers for data and VoIP services. But for sending video services, OpenDaylight controllers are better than others. Because only OpenDaylight controllers can send video packages. For testing resource utilization, the highest memory consumption is owned by the OpenDaylight controller. Based on the ITU-T G.1010 standardization of Floodlight controllers without the Johnson algorithm, Floodlight using Johnson and OpenDaylight algorithms only meets the delay standard for data services only.

Keywords: Controller, Performance, Floodlight, OpenDaylight, QoS, Resource Utilization

KATA PENGANTAR

Assalamualaikum wr. wb.

Alhamdulillah saya limpahkan atas segala rahmat Allah SWT yang sudah diberikan kepada kita semua, terkhususnya saya sebagai penulis yang selalu diberikan kelancaran oleh Allah SWT dalam berbagai hal termasuk juga dalam penyelesaian Tugas Akhir. Adapun judul yang penulis ambil untuk dijadikan Tugas Akhir ini adalah "Analisis Efek Penggunaan Kontroler Floodlight dan OpenDaylight Pada Performansi Jaringan SDN"

Tugas Akhir ini disusun untuk sebagai syarat akhir untuk menyelesaikan studi dan mendapatkan gelar Sarjana prodi Teknik Komputer pada Fakultas Teknik Elektro di Universitas Telkom. Dalam menuntaskan penyusunan Tugas Akhir ini, tentu banyak juga aspek-aspek yang mendukung semua ini untuk segera terselesaikan. Dengan kerendahan hati penulis ingin mengucapkan terima kasih kepada :

1. Allah SWT atas segala karunia dan takdir yang sudah diberikan dan kepada penulis dan kepada Nabi Muhammad SAW yang menjadi teladan penulis dalam melakukan tindakan.
2. Bapak Ir. Agus Virgono., M.T. dan Drs. Ir. R. Rumani M., Bc.TT., M.Sc. selaku Dosen Pembimbing Tugas Akhir ini yang sudah membimbing kami dalam menuntaskan Tugas Akhir ini dengan sebaik-baiknya.
3. Kedua orang tua, uda dan adik-adik, keluarga besar dari papa dan mama, keluarga di Warung Tercinta dan Ading yang sudah mendo'akan selalu, memotivasi selalu, menasihati selalu dan memberikan kasih sayang.
4. Romi Afan selaku teman sekelompok Tugas Akhir atas kerja samanya dan kekompakkannya.
5. Teman-teman SK-36-04, USBM, Biospin Lab dan untuk semua orang yang sudah mengharapkan kebaikan kepada penulis yang tidak mungkin bisa dituliskan satu persatu.

Akhir kata penulis menyadari bahwa dalam tulisan Tugas Akhir ini memiliki kekurangan. Penulis menginginkan kritik dan saran untuk kedepannya dari pembaca. Sekian,

Wassalamualaikum wr wb.

Bandung, Agustus 2018

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN i
LEMBAR PERNYATAAN ORISINALITAS ii
ABSTRAK iii
ABSTRACT iv
KATA PENGANTAR v
DAFTAR ISI vii
DAFTAR GAMBAR ix
DAFTAR TABEL xi
BAB 1 PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 1
1.3. Tujuan 2
1.4. Batasan Masalah 2
1.5. Metodologi Penyelesaian Masalah 2
1.6. Sistem Penulisan 3
BAB 2 DASAR TEORI 4
2.1. Software Defined Network 4
2.1.1. Arsitektur dan Karakteristik SDN 5
2.1.2. Protokol OpenFlow 6
2.1.3. Perutean Pada Openflow 7
2.1.4. Kontroler OpenFlow 8
2.2. Kontoller Floodlight 9
2.3. Kontroler OpenDaylight 9
2.4. Mininet 9
2.5. Routing Pada OpenFlow 10
2.6. Algoritma Shorthest Path 10
2.6.1. Algoritma Johnson 11
2.7. Topologi Mesh 12
2.8. Parameter Uji 13
2.8.1. QoS (Quality of Service) 13
2.8.2. Resource Utilitazion 15

BAB 3 PERANCANGAN KONFIGURASI SIMULASI 16
3.1. Gambaran Sistem 16
3.2. Model Sistem Simulasi 17
3.3. Perangkat Simulasi 17
3.4. Perancangan Topologi Pada Mininet 18
3.5. Implementasi Algoritma Johnson Pada *Controller* Floodlight dan OpenDaylight 21
3.6. Skenario Pengujian 22
3.6.1. Pengujian Quality of Service 22
3.6.2. Pengujian Resource Utilization 23
BAB 4 PENGUJIAN DAN ANALISIS 24
4.1. Pengujian QoS (Quality of Services) 24
4.1.1. Delay 24
4.1.2. Jitter 29
4.1.3. Throughput 32
4.1.4. Packet Loss 36
4.2. Pengujian Resource Utilization 38
BAB 5 KESIMPULAN DAN SARAN 41
5.1. Kesimpulan 41
5.2. Saran 42
DAFTAR PUSTAKA 43

DAFTAR GAMBAR

Gambar 2.1 Konsep SDN ……....4
Gambar 2.2 Arsitektur SDN……...5
Gambar 2.3 Topologi Mesh…….…...14
Gambar 3.1 Blok Perancangan Diagram…. ……..……..…...18
Gambar 3.2 Flowchart Perancangan Sistem ……..…….……...19
Gambar 3.3 Model Sistem Simulasi …….…… 19
Gambar 3.4 Topologi menggunakan 6 switch, 12 host……..22
Gambar 3.5 Topologi 8 switch, 16 host…….…...…….23
Gambar 3.6 Topologi 10 switch, 20 host……...23
Gambar 3.7 Alur Algoritma Johnson…….24
Gambar 4.1 Delay Data Floodlight Tanpa Algoritma Johnson……..24
Gambar 4.2 Delay Data Floodlight Dengan Algoritma Johnson…….……...25
Gambar 4.3 Delay Data OpenDaylight Tanpa Algoritma Johnson……25
Gambar 4.4 Delay VoIP Floodlight Tanpa Algoritma Johnson…….……26
Gambar 4.5 Delay VoIP Floodlight Dengan Algoritma Johnson ……..……26
Gambar 4.6 Delay VoIP OpenDaylight Tanpa Algoritma Johnson ……..……27
Gambar 4.7 Delay Video OpenDaylight Tanpa Algoritma Johnson……..……27
Gambar 4.8 Jitter Data Floodlight Tanpa Algoritma Johnson…….…28
Gambar 4.9 Jitter Data Floodlight Dengan Algoritma Johnson …….…..…28
Gambar 4.10 Jitter Data OpenDaylight Tanpa Algoritma Johnson…….…29
Gambar 4.11 Jitter VoIP Floodlight Tanpa Algoritma Johnson…….……..…29
Gambar 4.12 Jitter VoIP Floodlight Dengan Algoritma Johnson ……..…30
Gambar 4.13 Jitter VoIP OpenDaylight Dengan Algoritma Johnson ……..…..30
Gambar 4.14 Jitter Video OpenDaylight Dengan Algoritma Johnson ……..…31
Gambar 4.15 Thoughput Data Floodlight Tanpa Algoritma Johnson…….31
Gambar 4.16 Throughput Data Floodlight Dengan Algoritma Johnson…….32
Gambar 4.17 Throughput Data OpenDaylight Tanpa Algoritma Johnson……..…32
Gambar 4.18 Throughput Data Floodlight Dengan Algoritma Johnson……..…33
Gambar 4.19 Throughput VoIP Floodlight Tanpa Algoritma Johnson...……..….33
Gambar 4.20 Throughput VoIP Floodlight Dengan Algoritma Johnson …..….....34
Gambar 4.21 Throughput VoIP OpenDaylight Tanpa Algoritma Johnson ……34

Gambar 4.22 Packet Loss Video Floodlight Tanpa Algoritma Johnson ……35
Gambar 4.23 Packet Loss Video Floodlight Dengan Algoritma Johnson ……..…35
Gambar 4.24 Packet Loss Video OpenDaylight Tanpa Algoritma Johnson……38
Gambar 4.25 Resource Utilization Floodlight Tanpa Algoritma Johnson..……39
Gambar 4.26 Resource Utilization Floodlight Dengan Algoritma Johnson ……..39
Gambar 4.27 Resource Utilization OpenDaylight Tanpa Algoritma Johnson……40

DAFTAR TABEL

Tabel 2.1 Standarisasi QoS ITU-T G.1010 ……..……17

Tabel 3.1 Spesifikasi Perangkat Keras……...21

Tabel 3.2 Detail beban topologi…….,,,,……24

Tabel 3.5 Skenario pengujian QoS……26

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Software Define Network (SDN) merupakan paradigma baru pada teknologi jaringan yang memisahkan *control plane* dan *forwarding plane* pada jaringan[1]. *Control plane* dipisahkan dari *forwarding plane* dan dipindahkan menuju *server* terpusat yang disebut dengan kontroler. Peran utama kontroler adalah mengatur *rules* atau membuat keputusan, dan peran utama dari *control plane* adalah secara langsung meneruskan paket berdasarkan *rules* yang datang dari kontroler[5]. Paradigma seperti ini membuat jaringan lebih efektif dan efisien.

Peran kontroler sangat penting pada jaringan SDN. SDN pada umumnya menggunakan suatu protokol yang disebut dengan protokol OpenFlow. Masih banyak lagi protokol yang ada selain OpenFlow. Kontroler pada jaringan SDN yang didukung oleh protokol OpenFlow ada banyak macamnya. Beberapa jenis kontroler tersebut diantaranya adalah POX, NOX, Ryu, Beacon, Maestro, Floodlight, OpenDaylight dan sebagainya. Setiap kontroler diatas memiliki kelamahan dan kelebihannya masing-masing. Kelemahan dan kelebihannya itu bisa dilihat dari berbagai segi, salah satunya yaitu dari segi performansinya.

Ada beberapa aspek yang mempengaruhi performansi kontrol. Bahasa pemrograman, topologi dan *environment* simulasi juga mempengaruhi bagaimana performansi kontroler tersebut. Hal inilah yang akan diujikan untuk mengetahui bagaimana efek penggunaan kontroler pada jaringan. Penelitian dilakukan dengan kondisi tanpa menambahkan algoritma ke kontroler dan ketika ditambahkan algoritma pada jenis topologi yang dimodifikasi dengan *background traffic* yang berbeda-beda.

1.2. Rumusan Masalah

Rumusan masalah dalam pembuatan Tugas Akhir ini adalah seperti yang dijelaskan berikut :

1. Bagaimana efek pengunaan kontroler Floodlight dan OpenDaylight terhadap skalabilitas, fleksibilitas dan efisiensi.

2. Bagaimana efek penggunaan kontroler Floodlight dan OpenDaylight segi QoS (*quality of service*) yang dihasilkan dengan nilai standarisasi ITU-T G.1010

1.3. Tujuan

Tujuan dari tugas akhir ini terdapat poin-point berikut :

1. Menganalisis efek penggunaan kontroler Floodlight dan OpenDaylight terhadap skalabilitas, fleksibilitias dan efisiensi.
2. Menganalisis efek penggunaan kontroler Floodlight dan OpenDaylight segi QoS (*quality of service*) yang dihasilkan dengan nilai standarisasi ITU-T G.1010

1.4. Batasan Masalah

Tugas Akhir ini mempunyai batasan masalah yaitu :

1. Menggunakan protokol OpenFlow.
2. Menggunakan mininet sebagai emulator.
3. Topologi yang digunakan adalah topologi mesh full yang dimodifikasi.
4. Parameter dalam analisis QoS menggunakan standarisasi ITU-T yaitu G.1010.
5. Penelitian dilakukan hanya pada model ini.

1.5. Metodologi Penyelesaian Masalah

1. Studi Literatur

   Mengindentifikasi masalah melalui studi kepustakaan dengan literature yang ada seperti paper dan buku yang berkaitan dengan kompleksitas, SDN, algoritma *shortest-path*, parameter parameter yang bersangkutan dan teori pendukung lainnya.

2. Implementasi

   Pada tahap ini dilakukan perancangan topologi dan *controller* sesuai dengan karakteristik SDN dengan algoritma routing yang telah ditentukan.

3. Pengujian Alat dan Aplikasi

   Melakukan pengujian terhadap sistem yang telah dirancang dengan skenario dan parameter uji yang telah ditentukan.

4. Analisa Hasil Pengujian

   Tahap ini dilakukan analisis dari hasil pengujian terhadap simulasi yang telah dibuat dengan parameter uji sesuai yang telah ditentukan serta melakukan analisis pada hasil yang didapat.

5. Penyusunan Laporan Tugas Akhir

1.6. Sistem Penulisan

Tugas akhir ini dibagi dalam beberapa topik bahasan yang disusun secara sistematis dan terdiri dari:

- BAB 1 PENDAHULUAN

  Pada bab ini membahas tentang hal-hal yang mendasari penelitian ini serta membahas mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penyelesaian masalah, dan sistematika penulisan.

- BAB 2 LANDASAN TEORI

  Bab ini menjelaskan mengenai teori-teori dasar yang digunakan untuk menunjang penelitian seperti penjelasan tentang SDN, RYU, algoritma shorthest-path dan topologi yang digunakan pada simulasi ini, dan paramater uji.

- BAB 3 PERANCANGAN DAN IMPLEMENTASI

  Bagian dari bab ini adalah perancangan sistem, kebutuhan sistem, implementasi, dan skenario uji.

- BAB 4 PENGUJIAN DAN ANALISIS

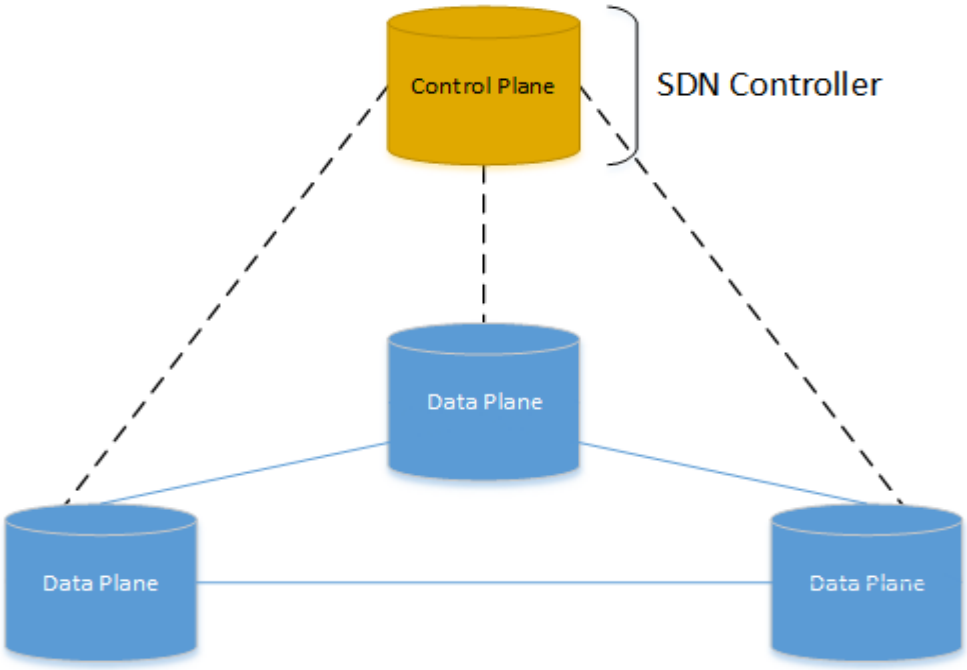
  Bab ini menjelaskan tentang pengujian dan analisa hasil pengujian yang telah dilakukan sesuai paramater uji yang telah ditentukan.

- BAB 5 KESIMPULAN DAN SARAN

  Bab ini menjelaskan kesimpulan dari hasil penelitian yang telah dilakukan serta saran untuk pengembangan selanjutnya.

BAB 2
DASAR TEORI

2.1. Software Defined Network



Gambar 2.1 Konsep SDN

Software Defined Network (SDN) dianggap sebagai salah satu cara yang memiliki kemampuan untuk mengubah bentuk jaringan tradisional yang sudah ada menjadi lebih fleksibel. SDN melakukan pendekatan melalui pemisahan secara fisik antara *control plane* dengan *forwarding plane*.. *Forwarding plane* membawa banyak paket data untuk diteruskan dari suatu *port* menuju *port* lainnya dengan mengikuti rules yang sudah di *program* kedalam perangkat keras. *Forwarding plane* melakukan pengiriman paket dengan cepat dan efisien. *Control plane* menambahkan karakteristik kepada perangkat dan menentukan logika pemrograman pada *forwarding plane*. Komunikasi antara *control plane* dan *forwarding plane* ini melalui suatu saluran yang aman, yang disebut dengan protocol OpenFlow[4].

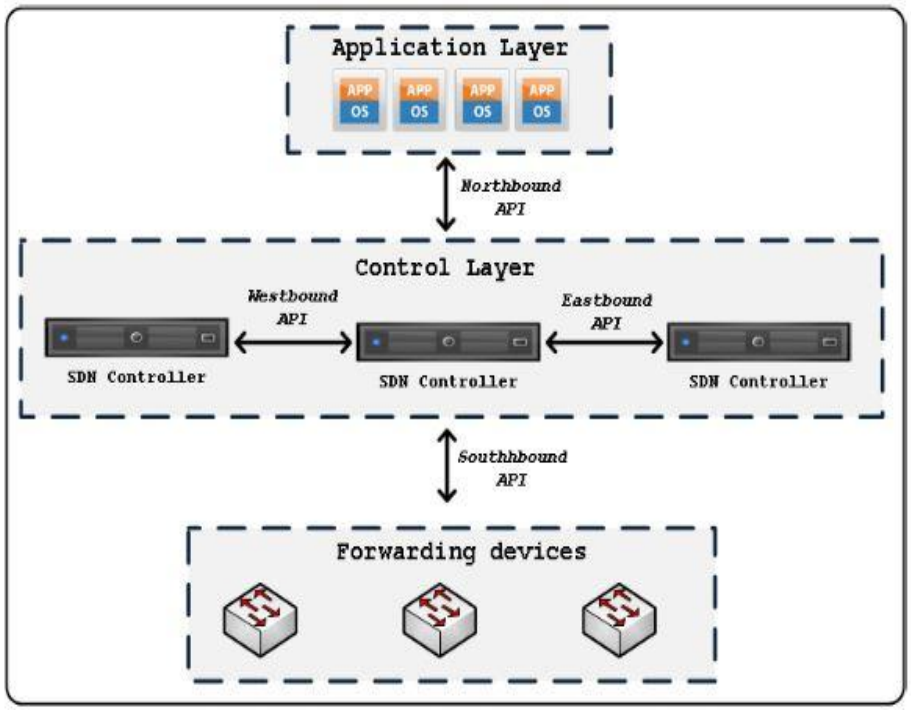
Seperti yang diketahui pendekatan pada jaringan SDN, terjadi pemisahan secara fisik antara *control plane* dengan *forwarding plane* dimana control plane

bertugas untuk mengendalikan banyak perangkat. Ada suatu organisasi yang bertugas untuk mengembangkan dan mengatur standarisasi tentang SDN. Nama organisasi tersebut adalah *Open Networking Foundation* (ONF). Mereka mengatakan bahwa SDN merupakan arstektur yang dinamis, mudah dikelola, hemat biaya, dan bisa deprogram yang membuatnya ideal untuk jaringan yang memiliki bandwith tinggi dan jaringan zaman sekarang[4].

SDN memiliki kelebihan seperti berikut[4]:

1. Pemisahan antara *control plane* dan *forwarding plane*
2. Pemusatan kontroler pada jaringan
3. Antarmuka yang terbuka antara perangkat yang ada pada *control plane* dan *forwarding plane*.
4. Kemampuan untuk deprogram pada jaringan menggunakan aplikasi lainnya.

2.1.1. Arsitektur dan Karakteristik SDN



Gambar 2.2 Arsitekture SDN

Jaringan SDN secara umum dibagi menjadi tiga bagian[1], yaitu:

1. Lapis Aplikasi, merupakan bidang pengelolaan dimana pemrograman jaringan sedang berlangsung

2. Lapis Kontrol, berfungsi sebagai pusat pengendalian yang memiliki *flow table / rules* yang berguna untuk penentuan keputusan pengiriman aliran paket data
3. Lapis Infrastruktur, lapis yang berisikan perangkat jaringan yang dihubungkan dengan Lapis Kontrol melalui Southbound API. Agar komunikasi melalui Southbound API selalu bersifat *programmable* maka digunanakan protokol OpenFlow.

Berdasarkan Arsitektur diatas SDN memiliki karakteristik sebagai berikut:

1. *Logically centralized intelligencei* (Kecerdasan terpusat secara logik), alam sebuah arsitektur SDN, pengendalian jaringan terdistribusi menggunakan sebuah antarmuka terstandarisasi yaitu OpenFlow. Dengan kecerdasan jaringan terpusat, pengambilan keputusan berdasarkan pandangan jaringan dari secara keseluruhan.
2. *Programmability*, jaringan SDN dikendalikan oleh fungsi dari perangkat lunak, yang mungkin disediakan oleh vendor atau operator jaringan.
3. *Abstraction*, di dalam jaringan SDN, aplikasi yang menggunakan layanan SDN dipisahkan dari teknologi jaringan yang mendasarinya. Perangkat jaringannya pun dipisahkan dari *layer* kontrol SDN untuk memastikan kemudahan dalam penggunaannya kembali di masa yang akan datang dalam layanan jaringan dan menempatkannya dalam *layer* kontrol.

2.1.2. Protokol OpenFlow

OpenFlow adalah standar komunikasi antarmuka pertama yang mendefinisikan antara *control plane* dengan *forwarding plane* pada arsitektur jaringan SDN[2]. OpenFlow memungkinkan mengakses langsung dan manipulasi perangkat *forwarding plane* seperti *switch* dan *router*, baik fisik dan virtual (*hypervisor-based*)[10]. OpenFlow diimplementasi dari kedua sisi antarmuka baik dari sisi infrastruktur jaringan dan kontrol *software* SDN. Sebuah OpenFlow *switch* terdiri dari *flow table* atau *group*

table. Setiap *flow table* dalam *switch* berisi satu *set flow entri,* yang terdiri dari *header field*, *counter,* dan *set of instructions* atau *actions*. Untuk menjalankan fungsinya, protokol *OpenFlow* membutuhkan tiga hal berikut, yaitu:

1. Kontroler (*controller*), berfungsi sebagai *server OpenFlow* yang berperan sebagai *control plane*
2. Penerus (*forwarder*), berfungsi sebagai perangkat yang dijalankan sebagai *forwarding plane*
3. *OpenFlow flow table,* tabel penerusan yang tidak tergantung pada jenis layanan. Setiap *flow table* mengandung kolom kecocokan dan tindakan terkait. *Forwarder* mencocokkan paket dengan bidang tertentu dalam flow table dan melakukan tindakan spesifik yang terkait dengan bidang pada paket yang cocok.

   Masing-masing paket yang memasuki *switch* melewati satu atau lebih *flow table*. Setiap flow table berisi entri yang terdiri dari enam komponen, seperti:

   a. *Field matching*, digunakan untuk mencocokkan paket sesuai dengan nilai di field
   b. *Priority*, data prioritas relatif dari entri tabel
   c. *Counters*, digunakan untuk pencocokan. Menggunakan perhitungan jumlah paket ditambah *bit error*
   d. *Instructions*, langkah-langkah dalam kasus korespondensi
   e. *Timeouts*, waktu tidak aktif maksimum dari suatu aliran
   f. *Cookie*, dapat digunakan oleh kontroler untuk menyaring statistik aliran, perubahan arus dan penghapusan arus yang tidak digunakan selama pemrosesan paket
   g. *Header Fields*, memudahkan pencocokan paket
   h. *Actions*, tindakan untuk diterapkan ke paket yang sesuai.

2.1.3. Perutean Pada Openflow

Perbedaan arsitektur antara jaringan konvensional dengan jaringan SDN menjelaskan bahwa setiap layer pada masing-masing jenis jaringan memiliki perbedaan yang sangat signifikan. Arsitektur SDN memiliki

southbound API yang didefinisikan oleh protocol openflow yang memungkinkan interaksi antara control plane dengan *forwarding plan*, dan *northbound API* yang menyajikan antarmuka (*interface)* abstraksi jaringan untuk aplikasi dan *management system* pada layer atas. Serta pangambilan keputusan terletak pada sebuah controller terpusat[14]. Berikut gambaran umum arsitektur perutean pada jaringan SDN.

Control plane akan menggunakan tabel *routing* untuk membuat tabel forwarding yang digunakan oleh *forwarding plane*. Tabel forwarding dikirim ke *forwarding plane* sebagai bagian dari sistem operasi perangkat. Jadi ketika sebuah frame Ethernet tiba pada *interface switch, forwarding plane* kemudian meneruskannya ke *port output*.

OpenFlow mendefinisikan standar untuk mengirimkan aturan aliran perangkat jaringan. Sehingga *control plane* dapat menambahkannya ke tabel forwarding untuk forwarding plane. Aturan *flow* ini berisi kolom seperti MAC sumber & tujuan, IP Sumber & tujuan, sumber dan TCP tujuan, VLAN, QoS, tag MPLS dan banyak lagi. Aturan aliran kemudian ditambahkan ke tabel forwarding yang ada di perangkat jaringan. Tabel forwarding adalah yang semua router dan switch gunakan untuk mengirimkan frame dan paket ke port tujuan. Sehingga Controller mendapatkan kemampuan baru dan kontrol yang benar-benar menyeluruh terhadap lalu lintas mengalir. Oleh karena itu, OpenFlow bukan tentang routing atau switching, namun openflow tentang forwarding.

2.1.4. Kontroler OpenFlow

Merupakan aplikasi pada SDN yang bertugas mengelola *flow control* untuk mengaktifkan kecerdasan jaringan[2]. Kontroler SDN sama seperti protokol OpenFlow yang mana memungkinkan *software* bisa dijalankan di berbagai macam *hardware*, jadi tidak terikat pada *vendor* dan bersifat *opensource* sehingga bisa dikembangkan siapa saja. Banyak jenis kontroler openflow yang sering kita dengar, seperti : NOX, POX, Ryu, OpenDaylight, Floodlight, Pyretic dan semacamnya.

2.2. Kontoller Floodlight

Kontroller Floodlight merupakan controller dengan kelas *enterprise*, memiliki lisensi Apache, dan memilik Bahasa pemrograman berbasis java yang dikembangkan oleh Big Switch Network. Floodlight didukung oleh *feature* OpenFlow yang membuatnya bisa mengatur aliran data pada lingkungan SDN[9].

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.*

2.3. Kontroler OpenDaylight

Proyek OpenDaylight terbentuk atas konsorsium Linux Foundation. Kontroler ini didukung oleh standar *northbound* API, jadi tidak hanya didukung dengan protokol *southbound* API seperti OpenFlow, I2RS dan NETCONF yang bisa diprogram. Beberapa fitur kontroler ini adalah Java-based (OSGI), modular, *pluggable* dan juga didukung oleh banyak protokol *southbound*. OpenDaylight sering dimanfaatkan dalam jenis jaringan berskala besar[8].

2.4. Mininet

Mininet merupakan suatu emulator yang memperbolehkan penggunanya secara berulang-ulang untuk membuat prototipe suatu jaringan dalam skala besar menggunakan satu komputer[3]. Fitur pada mininet memperbolehkan pembuatan, komunikasi, kustom dan membagikan prototipe jaringan secara cepat dalam bentuk mekanisme virtualisasi. Topologi awal pada mininet memiliki OpenFlow Kernel Switch yang menghubungkan dua host dengan sebuah kontroler openflow[8].

2.5. Routing Pada OpenFlow

Pada arsitektur SDN, terdapat southbound API yang memiliki protokol OpenFlow didalamnya agar memungkinkan interaksi antara *control plane* dan *forwarding plane*. Northbound API menyediakan *interface* abstraksi jaringan untuk mengelola sistem pada layer tersebut. Pengambilan keputusan akan ditentukan oleh controller yang bersifat secara terpusat[11].

Cara kerja *routing* pada SDN yaitu *control plane* akan menggunakan *table routing* untuk membuat *table forwarding* yang digunakan *forwarding plane*. *Table forwarding* dikirim ke *forwarding plane* sebagai bagian dari sistem operasi perangkat. Jadi ketika sebuah *frame ethernet* tiba pada *interface switch*, *forwarding plane* kemudian meneruskannya ke *port output*.

OpenFlow mendefinisikan standar untuk mengirimkan aturan aliran perangkat jaringan sehingga *control plane* dapat menambahkannya ke *table forwarding* untuk *forwarding plane*. Aturan *flow* ini berisi kolom seperti MAC sumber & tujuan, IP Sumber & tujuan, sumber dan TCP tujuan, VLAN, QoS, tag MPLS dan banyak lagi. Aturan *flow* kemudian ditambahkan ke *table forwarding* yang ada di perangkat jaringan. *Table forwarding* adalah yang semua *router* dan *switch* gunakan untuk mengirimkan *frame* dan paket ke port tujuan. Sehingga controller mendapatkan kemampuan baru dan kontrol yang benar-benar menyeluruh terhadap lalu lintas mengalir. Oleh karena itu, OpenFlow bukan tentang *routing* atau *switching*, namun openflow tentang *forwarding*.

2.6. Algoritma Shorthest Path

Algoritma *shorthest path* adalah algoritma untuk mencari jalur terpendek dari suatu titik ke titik tertentu. Ada beberapa macam jenis *shortest path*, antara lain[5]:

1. Lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*).
2. Lintasan terpendek antara semua pasangan simpul (*all pair shortest path*).
3. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*).

4. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Terdapat banyak algoritma untuk menghituk jalur terpendek seperti algoritma Dijkstra, algoritma Floyd-Warshall, algoritma Bellman-Ford, dan algoritma Johnson. Algoritma *shortest* juga digunakan pada protokol *routing* sebagai algoritma routing seperti Algoritma dijkstra pada protokol *routing* OSPF dan IS-IS dan algoritma Bellman-Ford pada protokol *routing* RIP.

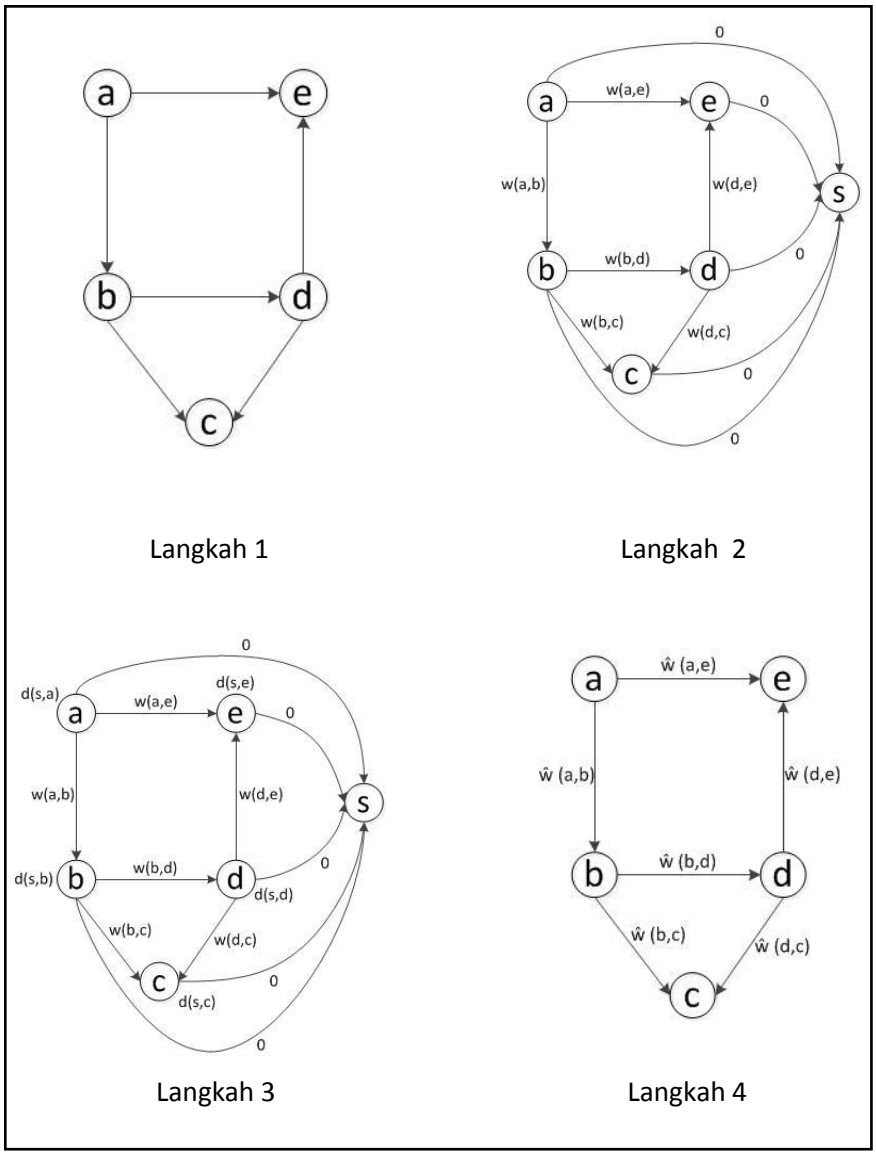
2.6.1. Algoritma Johnson

Algoritma Johnson adalah algoritma untuk menyelesaikan *shortest path problem* dengan orientasi mencari jarak terdekat sekaligus bobot paling sedikit. Algoritma Johnson merupakan perpaduan anatara algoritma Bellman Ford dan Algoritma Dijkstra.

Penyelesaian Algoritma Johnson adalah mengonstruksi graf baru dengan menambahkan titik baru (s) pada graf dan memberi bobot sisi yang keluar dari titik baru tersebut dengan angka 0 seperti Gambar 2.3(a). Langkah selanjutnya adalah mencari jarak terpendek d(s, dari titik s kesemua titik lain, jarak terpendek tersebut digunakan untuk mengubah bobot sisi bernilai positif ŵ(u,v) dengan cara[2] :

$$\widehat{\omega}(u,v) = \omega(u.v) + d(s,u) - d(s,v) \text{ [2.1]}$$

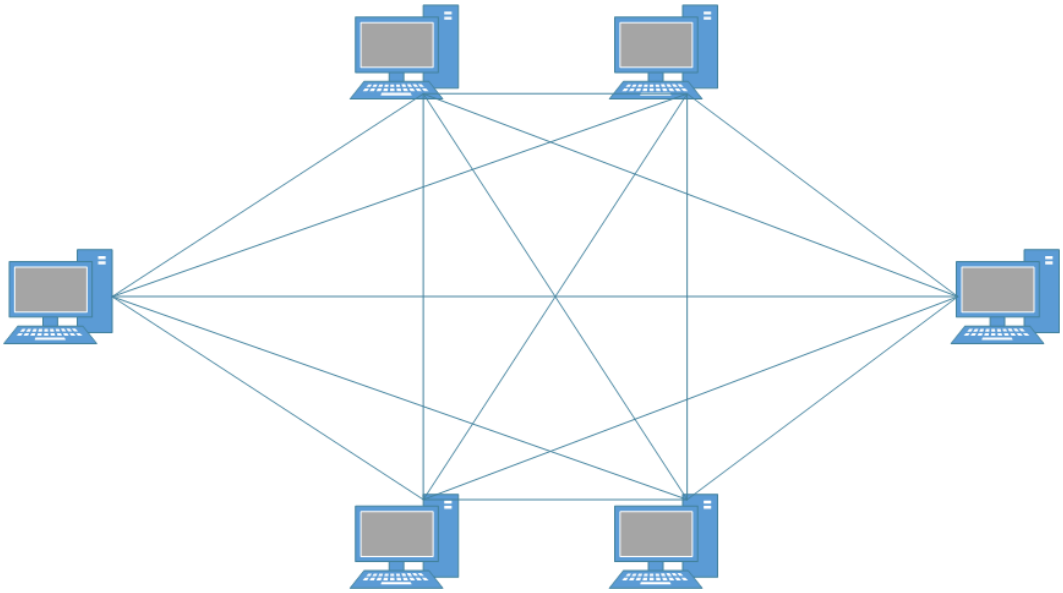
Setelah itu titik baru (s) tadi dihapus dan algoritma dijkstra digunakan untuk mencari jalur terpendek dari node sumber ke setiap node lain pada graph baru yang sudah diubah bobotnya menjadi positif. Ilustrasi langkah-langkah pencarian jalur terpendek dengan algoritma Johnson dapat dilihat pada Gambar 2.3



Gambar 2.3 Langkah-langkah pencarian algoritma Johnson

2.7. Topologi Mesh

Topologi mesh merupakan topologi jaringan komputer yang menghubungkan semua komputer secara penuh, topologi ini adalah topologi yang paling komplek dibanding dengan topologi jaringan lainnya[7]. Topologi jenis ini banyak digunakan oleh penyedia layanan internet (ISP). Konsep dari topologi ini adalah setiap komputer dalam jaringan saling terhubung satu sama lain sehingga jika terjadi kerusakan pada salah satu komputer tidak berpengaruh pada komputer lain atau berpengaruh pada jaringan.



Gambar 2.4 Topologi Mesh

Jenis koneksi pada topologi jaringan mesh terdiri dari 2 jenis, kedua topologi mesh tersebut meliputi:

- Topologi Mesh *Fully Connected* mempunyai ciri utama dimana setiap komputer dalam jaringan saling terhubung satu sama lain secara penuh. Sebagai contoh jika ada 5 komputer dalam jaringan tersebut maka satu komputer akan terhubung ke 4 komputer lainnya.
- Topologi mesh *partial connected* topologi jenis ini memiliki ciri yaitu setiap komputer dalam jaringan tersebut tidak semua komputer akan terhubung dengan komputer lainnya sehingga ada beberapa komputer yang saling terhubung satu sama lain dan beberapa komputer tidak saling berhubungan.

2.8. Parameter Uji

Untuk menganalisis performansi kontroler SDN sebelum dan sesudah di dan sebelum ditambahkan algoritma Johnson maka digunakan parameter uji yaitu QoS dan *resource utilization.*

2.8.1. QoS (Quality of Service)

QoS merupakan terminologi yang digunakan untuk mendefinisikan kemampuan suatu jaringan untuk menyediakan tingkat jaminan layanan

yang berbeda-beda. Melalui Q0S, seorang *network administrator* dapat memberikan prioritas trafik tertentu[3].

Parameter-parameter dalam QoS secara umum adalah:

- *Delay*
- *Jitter*
- *Packet Loss·*
- *Throughput*

Tabel 2.1 Standarisasi QoS ITU-T G.1010

Parameter performansi	*Data*	*VoIP*	*Video*
One Way Delay (latency)	*Preffered < 15 s; Acceptable < 60 s Nb : Amount of Data 10 kB – 10MB*	*Prefered < 150 ms; acceptable < 400 ms*	*< 10 s*
Jittter	*NA*	*< 1 ms*	*NA*
Throughput	*NA*	*4 – 64 kbps*	*16 – 384 kbps*
Packet loss	*0%*	*< 3%*	*< 1%*

2.8.1.1. Delay

Dikenal juga dengan nama *latency*, yang merupakan jumlah waktu yang dibutuhkan untuk mengirim data dari sumber ke tujuan dalam suatu jaringan[9]. Persamaan:

$$di = (Ri - Si) \text{ .. [2.2]}$$

Keterangan:

di : *delay* paket ke-*i*

R : waktu paket ke-*i* dikirimkan

S : waktu paket ke-*i* diterima

2.8.1.2. Jitter

Jitter merupakan variasi nilai dari *delay*. Besarnya nilai *jitter* biasanya dipengaruhi oleh variasi beban trafik dan besarnya tumbukan antar paket (*congestion*) yang ada dalam jaringan tersebut[9]. Nilai beban trafik berbanding lurus dengan nilai *congestion*.

Persamaan :

$$\text{Di} = (R_i - S_i) - (R_{i-1} - S_{i-1})$$

$$\text{Di} = (R_i - S_{i-1}) - (R_i - S_{i-1})$$

$$\text{Averange Jitter} = \frac{\sum_i^n |D_i|}{n} \quad [2.3]$$

2.8.1.3. *Packet Loss*

Packet loss merepresentasikan jumlah paket yang gagal terkirim sampai tujuan[9]. Persamaan:

Packet Loss Ratio = number of paket loss/ number packet sent x 100% [2.3]

2.8.1.4. *Throughput*

Throughput adalah kemampuan jaringan untuk membawa volume data lebih dari satu unit waktu[9]. Volume data yang sukses dikirim dari satu titik ke titik lainnya akan dibandingkan dengan jumlah waktu yang dibutuhkan selama pengiriman (total).

Persamaan :

$$\textit{Throughput} = \frac{P_{received}}{T_{total}} \quad [2.4]$$

Keterangan :

$P_{received}$ = paket yang sukses diterima

T_{total} = total waktu pengiriman

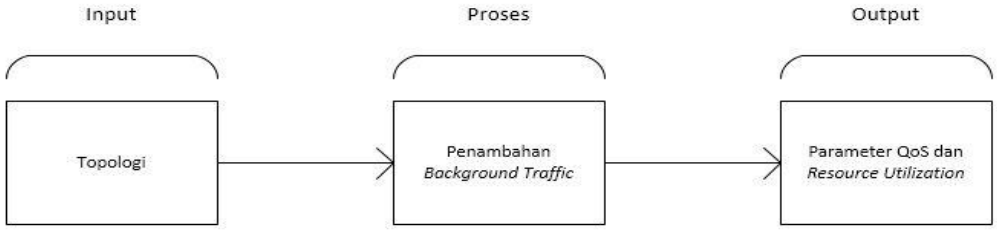
2.8.2. Resource Utilitazion

Pengujian *resource utilization* untuk mengetahui konsumsi memori *controller* selama *controller* dijalankan. Pengukuran dilakukan saat *controller* mulai dijalankan hingga jaringan berada dalam kondisi stabil. Untuk mengetahui konsumsi memori dari *controller* dapat menggunakan aplikasi *monitoring* yang terdapat pada sistem operasi Ubuntu

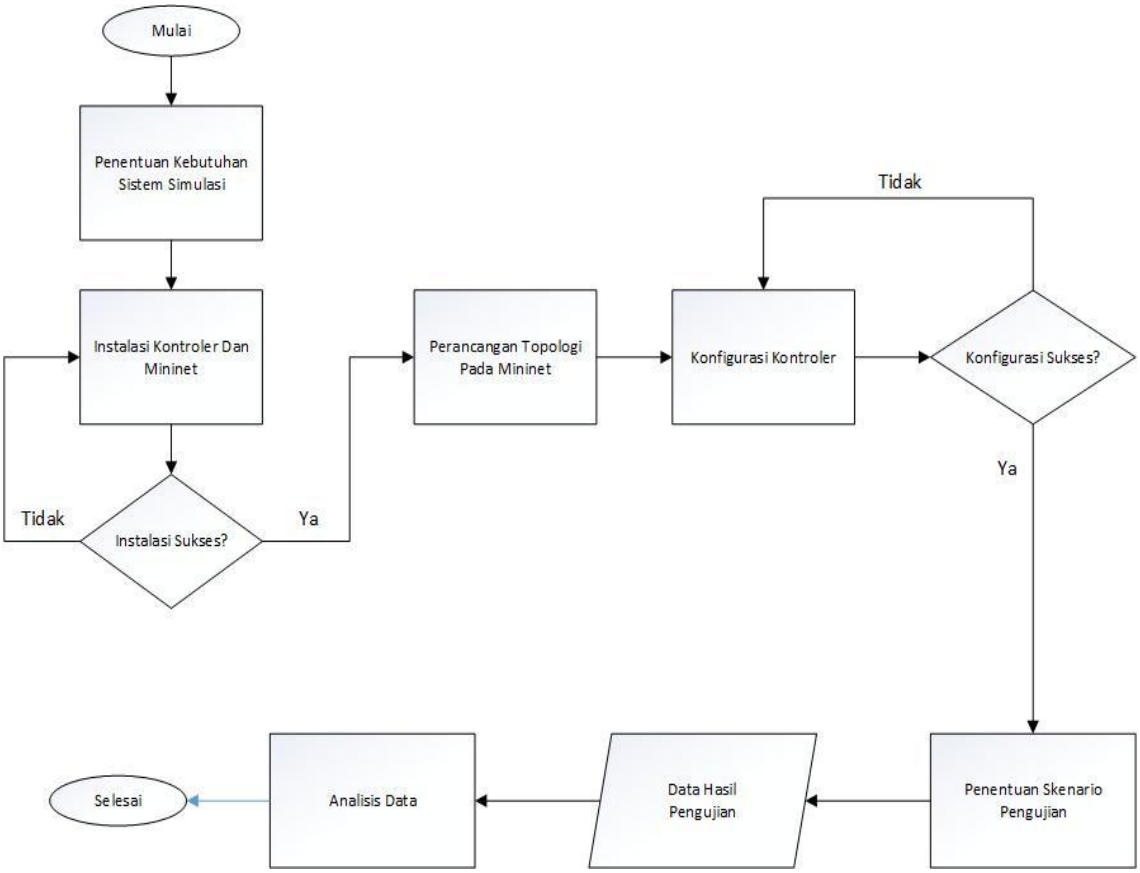
BAB 3
PERANCANGAN KONFIGURASI SIMULASI

3.1. Gambaran Sistem

Pada tugas akhir ini, pembuatan topologi dibuat pada emulator mininet. Dan dilakukan penambahan *background traffic* secara berkala menggunakan aplikasi Iperf. Lalu ditambah dengan aplikasi D-ITG yang berfungsi untuk merekam informasi lalu lintas data, yang nantinya berguna untuk analisis performasi kontroler Floodlight dan OpenDaylight.

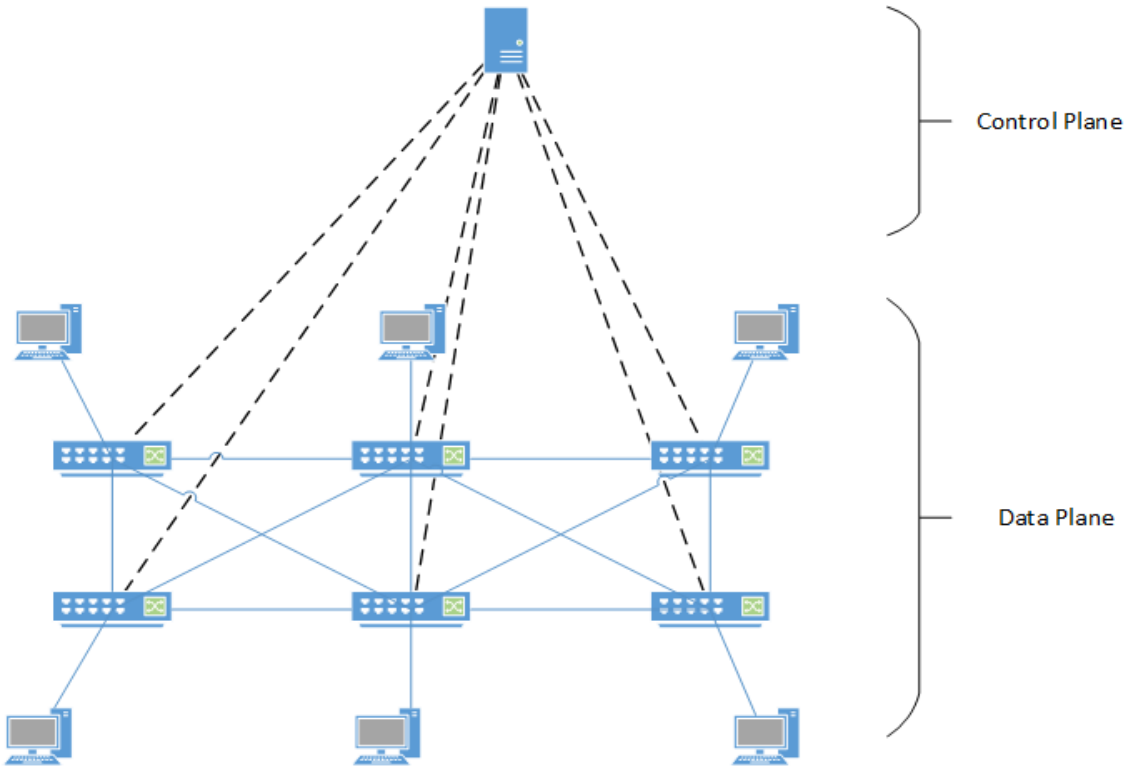


Gambar 3.1 Blok Diagram Perancangan



Gambar 3.2 Flowchart Perancangan Sistem

3.2. Model Sistem Simulasi



Gambar 3.3 Model Sistem Simulasi

Pada simulasi ini akan dilakukan pengujian pada jaringan SDN dan melibatkan dua kontroler SDN yaitu Floodlight dan OpenDaylight. Simulasi juga akan dilakukan dua kali pengujian, yaitu ketika tanpa algoritma Johnson dan ketika sudah ditambahkan algoritma johnson. Kontroler menjalankan fungsi sebagai *control plane* yang bertugas untuk mengontrol aliran data, melakukan routing dan melakukan keseluruhan aturan dari SDN. Pembuatan model topologi seperti gambar diatas karena kompleksitas jaringannya. Terdapat banyak loop, jumlah switch dan jumlah host yang banyak untuk dilakukan pada pengujian ini. Pemodelan topolgi diatas merupakan pemodelan topologi mesh.

3.3. Perangkat Simulasi

Pada simulasi dan analisis tugas akhir ini akan digunakan perangkat keras berupa satu unit Komputer/PC dengan spesifikasi yang dijelaskan pada table berikut:

Tabel 3.1 Spesifikasi Perangkat Keras

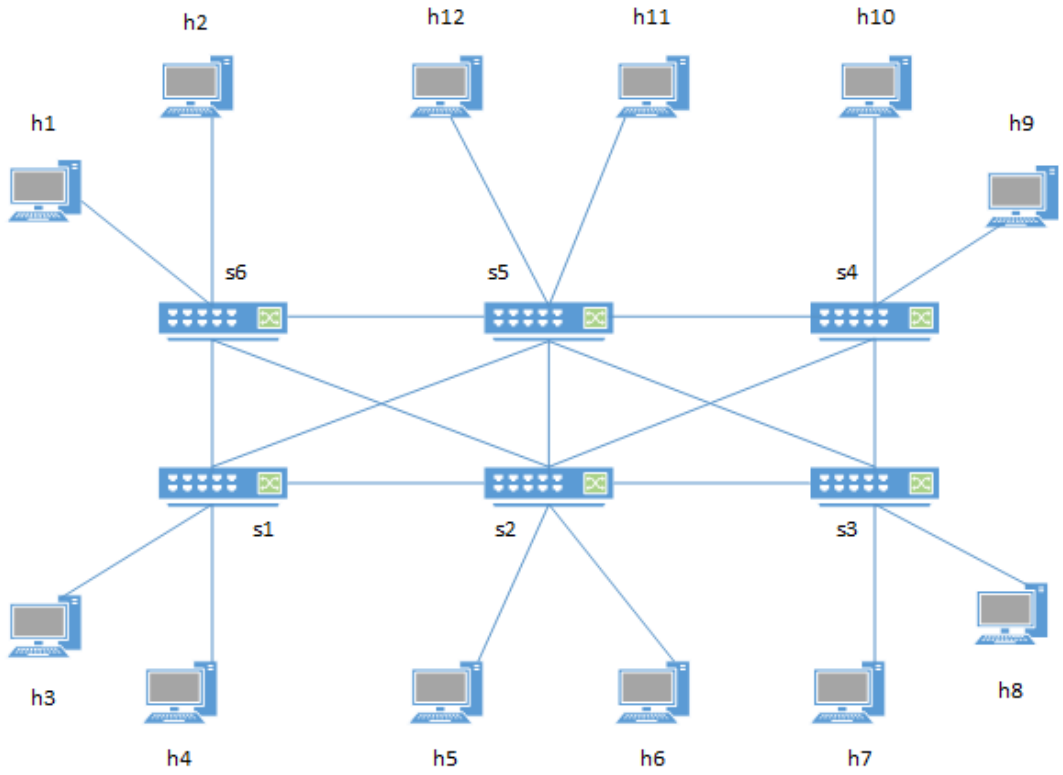
Spesifikasi	Komputer
Processor	Intel(R) Core(TM) i5-750 CPU @ 2.67GHZ
RAM	6 GB
Operating System	Ubuntu 15.04 LTS 64 bit
Fungsi	*Control plane* : Floodlight, OpenDaylgiht *Forwarding plane* : Switch OpenFlow dan host pada Mininet

Adapun aplikasi lain untuk melengkapi kebutuhan simulasi ini yaitu:

- *Distributed Internet Traffic Generator* (D-ITG) digunakan untuk menghasilkan trafikvdan mereplika beban kerja aplikasi internet secara akurat. Selain itu D-ITG juga berfungsi sebagai alat pengukuran mampu mengukur metrik kinerja yang paling umum misalnya *throughput, delay, jitter, paket loss*
- Iperf yang berguna untuk membangkitkan nilai *background traffic* secara berkala.

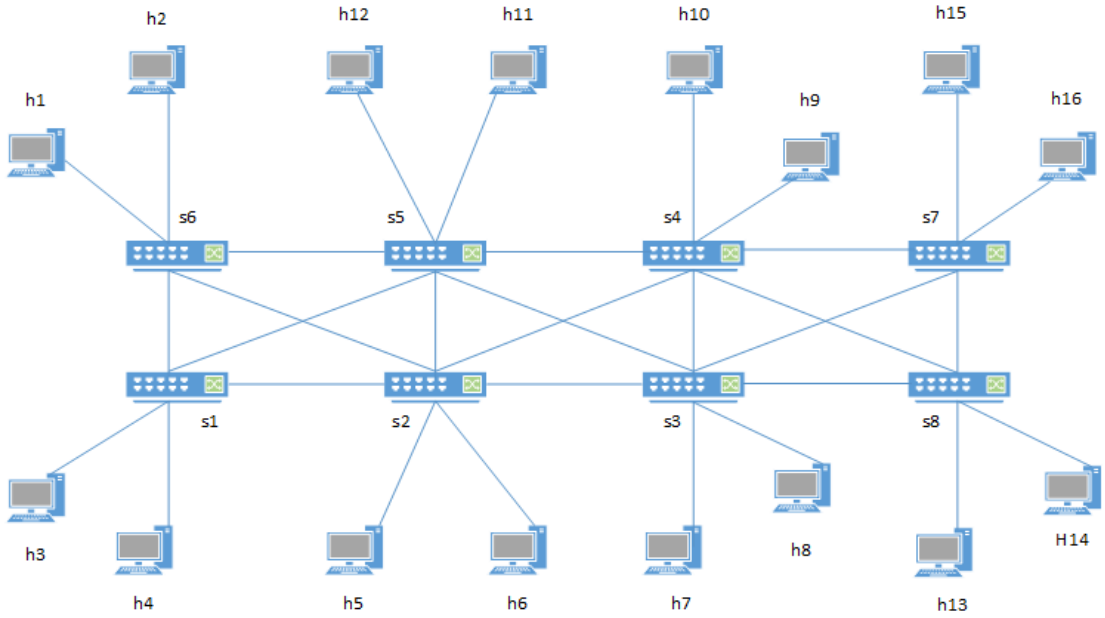
3.4. Perancangan Topologi Pada Mininet

Topologi pada simulasi ini dibuat menggunakan salah satu fitur dari mininet yaitu miniedit. Miniedit sangat membantu dalam perancangan pembuatan topologi ini karena memiliki *interface* yang mudah dipahami. *User* hanya merancang model topologi sesuai dengan yang diinginkan, lalu memberikan nilai pada perangkat-perangkat yang ada didalam rancangan topologi.

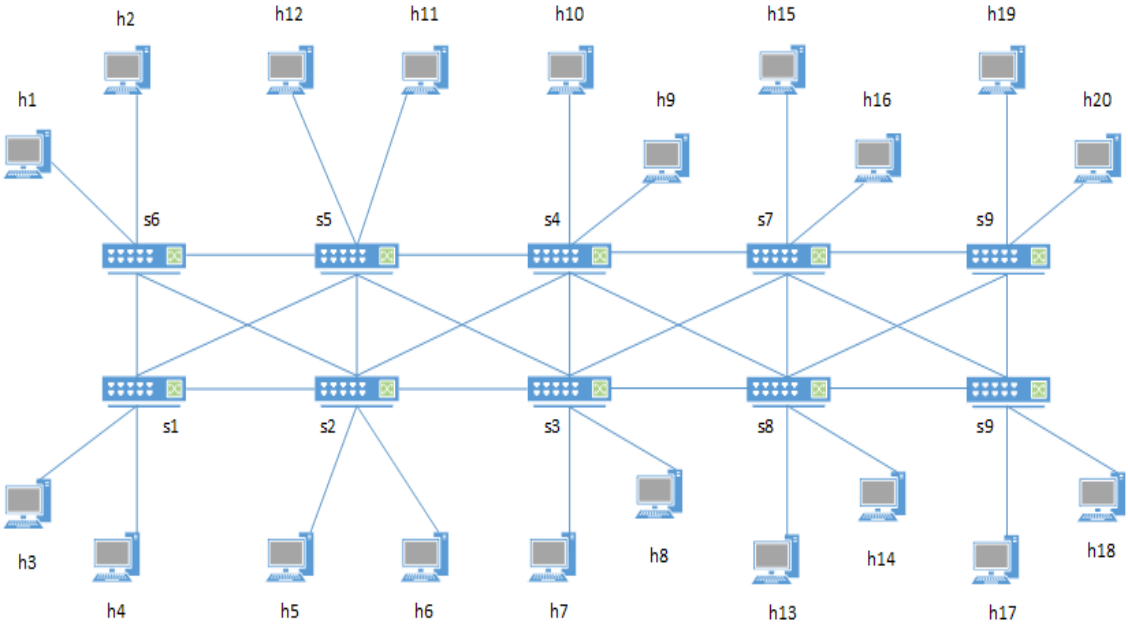


Gambar 3.4 Topologi menggunakan 6 switch, 12 host

Topologi yang digunakan diatas adalah topologi mesh *fully connected* yang terdiri dari 6 *switch* dan 12 *host*. Pembuatan topologi mesh seperti modifikasi diatas dianggap bisa menjadi topologi yang kompleks untuk dijadikan pengujian parameter QoS untuk dua kontroler diatas karena memiliki hop dan link yang banyak dibandingkan penggunaan model topologi jaringan lainnya. Ditambag pada model jaringan ini terdapat banyak looping.



Gambar 3.5 Topologi 8 switch, 16 host



Gambar 3.6 Topologi 10 switch, 20 host

Penambahan jumlah switch dan host membuat penentuan rute jaringan semakin kompleks, karena pertambahan jumlah hop dan link diatas. Penambahan jumlah switch dan host yang dilakukan secara berkala nantinya akan memperlihatkan karakteristik jaringan tersebut.

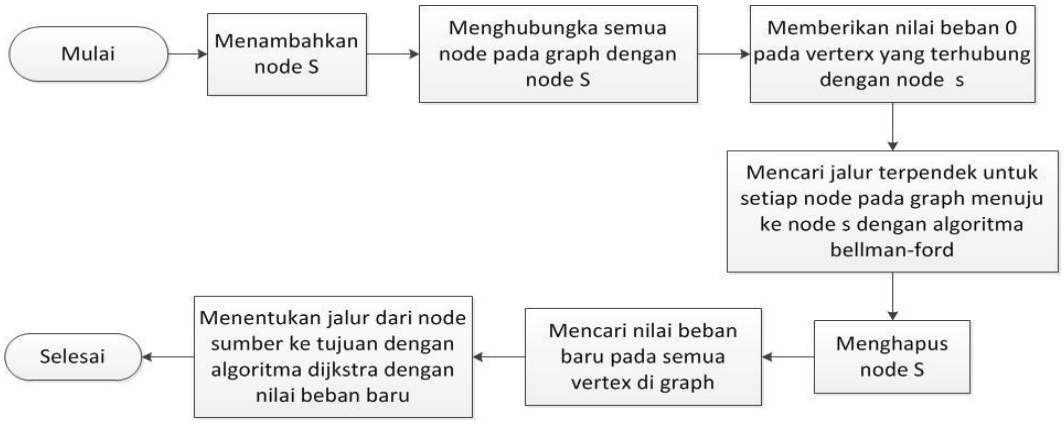
Pada simulasi ini dirancang tiga macam topologi dengan jumlah host dan switch yang berbeda sesuai dengan gambar diatas. Adapun keterangan dari tiap topologi tersebut adalah :

Tabel 3.2 Detail beban topologi

Jumlah switch	Jumlah host	Jumlah link antar swicth	Bandwith link	Delay link	Metric
6	12	9	0Mbps-100Mbps	1ms	(bandwith+delay)*256
8	16	11			
10	20	17			

3.5. Implementasi Algoritma Johnson Pada *Controller* Floodlight dan OpenDaylight

Implementasi Algoritma Johnson pada kontroler Floodlight adalah dengan cara menambahkan *package* baru yang bernama MultiPathRouting pada *package* bawaan kontroler floodlight sendiri. Didalamnya terdapat *class* bernama LinkWithCost.java yang bisa disisipkan algoritma johnson dengan bantuan *library* NetworkX. Agar package ini bisa dijalankan, tambahkan `net.floodlightcontroller.multipathrouting.MultiPathRouting` kedalam modul kontroler floodlight dan ke dalam file konfigurasi properti kontroler floodlight.



Gambar 3.6 Alur Algoritma Johnson

3.6. Skenario Pengujian

Pada pengujian ini akan digunakan dua parameter pengujian yaitu QoS dan *Resouce Utilization* untuk mengetahui bagaimana kinerja jaringan SDN tersebut.

3.6.1. Pengujian Quality of Service

Tujuan pengujian ini adalah untuk menganalisa kinerja kontroler yang mengacu pada nilai standarisasi ITU-T G.1010 yang sudah dicantumkan pada tabel 2.1 yang menggunakan parameter *delay*, *jitter*, *throughpu*t dan *packet loss*.

Pengujian *quality of service* menggunakan topologi dengan jumlah switch 6, 8 dan 10. Hal ini dilakukan untuk mengetahui pengaruh penambahan jumlah switch terhadap kinerja routing. Percobaan akan dilakukan sebanyak 30 kali dengan membangkitkan trafik UDP. Trafik UDP berupa data, video streaming, dan Voip dihasilkan dari D-ITG dengan spesifikasi sebagai berikut:

1. Trafik Data dengan spesifikasi inter-departure time (IDT) konstan = 100 pps , ukuran paket yang terdistribusi poisson dengan μ = 38,4 Bytes dengan waktu 20000milisecond
2. Video streaming 24 frame (satu paket per frame) per sekon, ukuran paket yang terdistribusi normal dengan μ = 27791 Bytes dan σ^2 = 6254 Bytes, dan membutuhkan throughput sekitar 5,336 Mbps.
3. VoIP menggunakan G.711 codec tanpa voice activation detection (VAD) sebanyak 100 pps yang berukuran paket 80 Bytes, dan membutuhkan throughput sekitar 70,4 Kbps.

Pada pengujian ini dilakukan variasi pengujian dengan menambahkan background trafik dengan bantun iperf. Terdapat 5 jenis backround trafik yang digunakan yaitu 0Mbps, 20Mbps, 40Mbps, 60Mbps, dan 80Mbps. Pengujian ini dilakukan tidak menggunakan jaringan GigabitEthernet karena skala jaringan yang digunakan masih kecil. Terlebih untuk pengujian menggunakan GigabitEthernet memerlukan spesifikasi computer *high-end*

Tabel 3.5 Skenario pengujian QoS

Jumlah Switch	Node sumber	Node Tujuan
6	h1	h9
8	h1	h13
10	h1	h17

Nilai dari parameter QoS yang didapat akan dibandingkan dengan nilai yang telah ditetapkan oleh badan standarisasi ITU-T. sehingga dapat diketahui nilai dari parameter QoS tersebut sudah memenuhi standarisasi atau belum.

3.6.2. Pengujian Resource Utilization

Pengujian ini dilakukan bertujuan untuk memperoleh informasi tentang konsumsi memori oleh kontroler pada saat kontroler sedang dijalankan. Informasi konsumsi memori ini akan diperlihatkan dalam ukuran MegaBytes (MB).

Pengujian resource utilization dilakukan ketika jaringan dalam kondisi steady state. Lalu untuk memeriksa konsumsi memori yang digunakan oleh kontroler tersebut bisa dilakukan dengan mengetikkan command top pada terminal ubuntu. Temukan proses kontroler yang sedang berjalan dan nantinya akan diperlihatkan konsumsi memori dalam bentuk persen. Untuk mengetahui jumlah konsumsi memori dalam MB, maka dibutuhkan perhitungan sebagai berikut:

%memori x total ram

Keterangan:

%memori: konsumsi memori dalam persen

total ram: jumlah ram yang dialokasikan untuk *operating system*

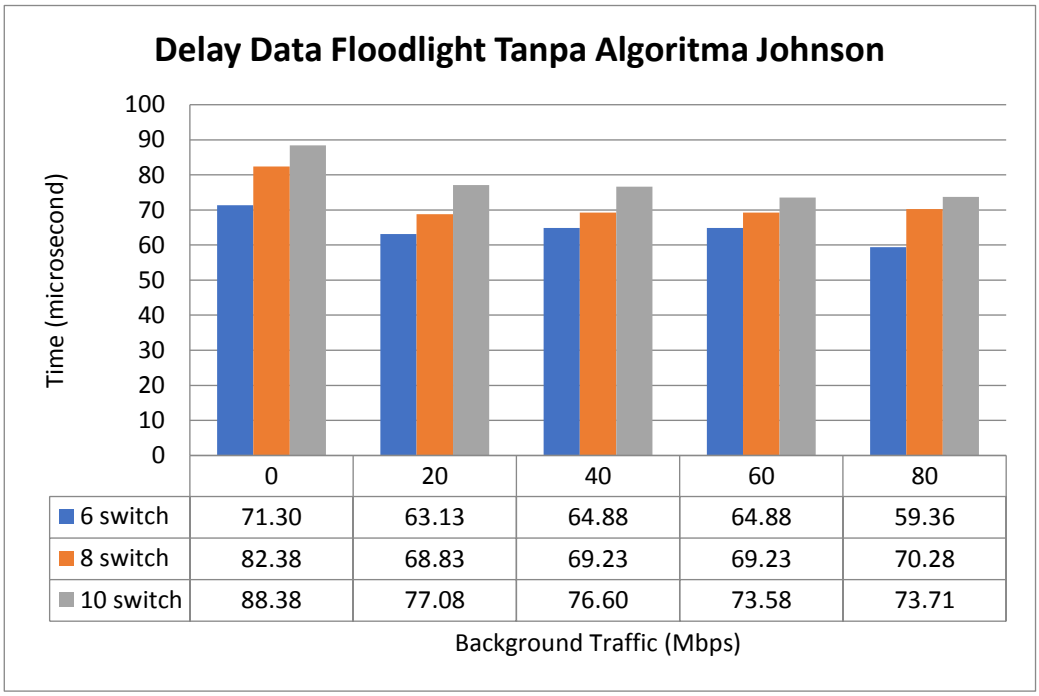
BAB 4
PENGUJIAN DAN ANALISIS

4.1. Pengujian QoS (Quality of Services)

Terdapat empat parameter yang digunakan dalam pengujian nilai QoS pada tugas akhir ini yaitu *delay*, *throughput*, *jitter*, *dan packet loss*. Pengujian ini dilakukan dengan cara pengiriman yang menggunakan layanan berupa data, VoIP dan video. Skenario pengujian yang digunakan pada penelitian ini yaitu menggunakan *background traffic* sebesar 0.1Mbps, 20 mMbps, 40Mbps, 60Mbps dan 80Mbps.

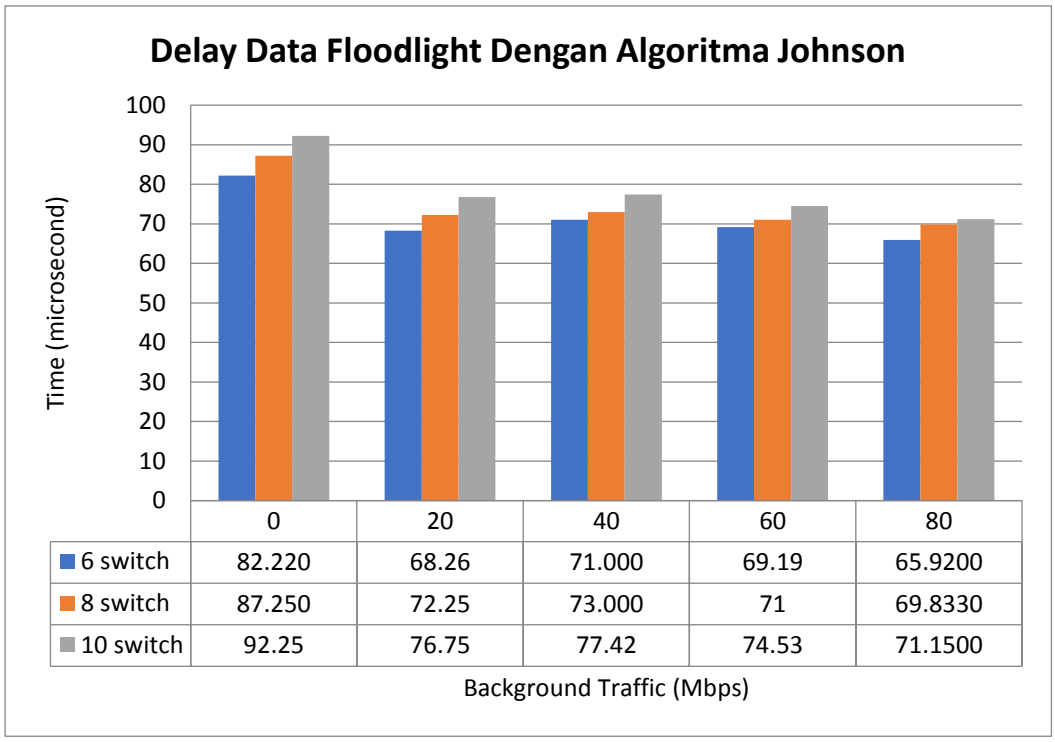
4.1.1. Delay

Delay merupakan total waktu yang dilalui suatu paket dari pengirim ke penerima melalui sebuah jaringan. *Delay* yang diukur pada pengujian ini adalah *one-way-delay*. Ada dua scenario. Sebelum dioptimalisasi dan sudah di optimalisasi.

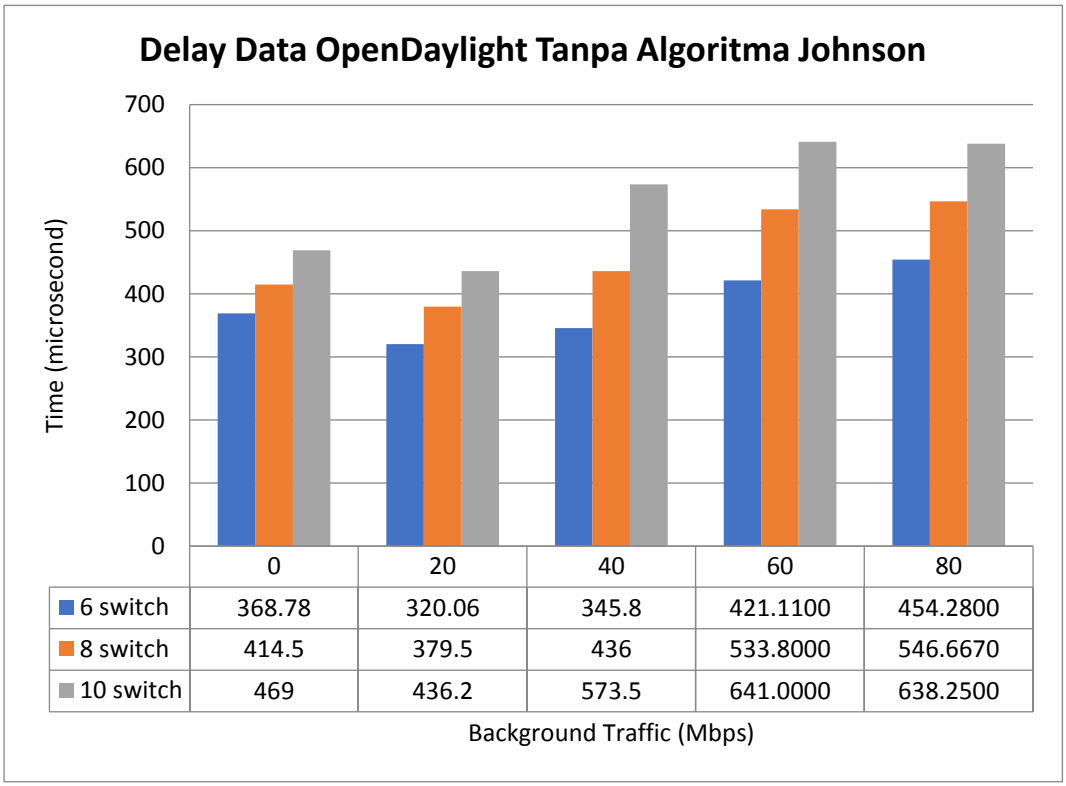


	0	20	40	60	80
6 switch	71.30	63.13	64.88	64.88	59.36
8 switch	82.38	68.83	69.23	69.23	70.28
10 switch	88.38	77.08	76.60	73.58	73.71

Gambar 4.1 Delay Data Floodlight Tanpa Algoritma Johnson

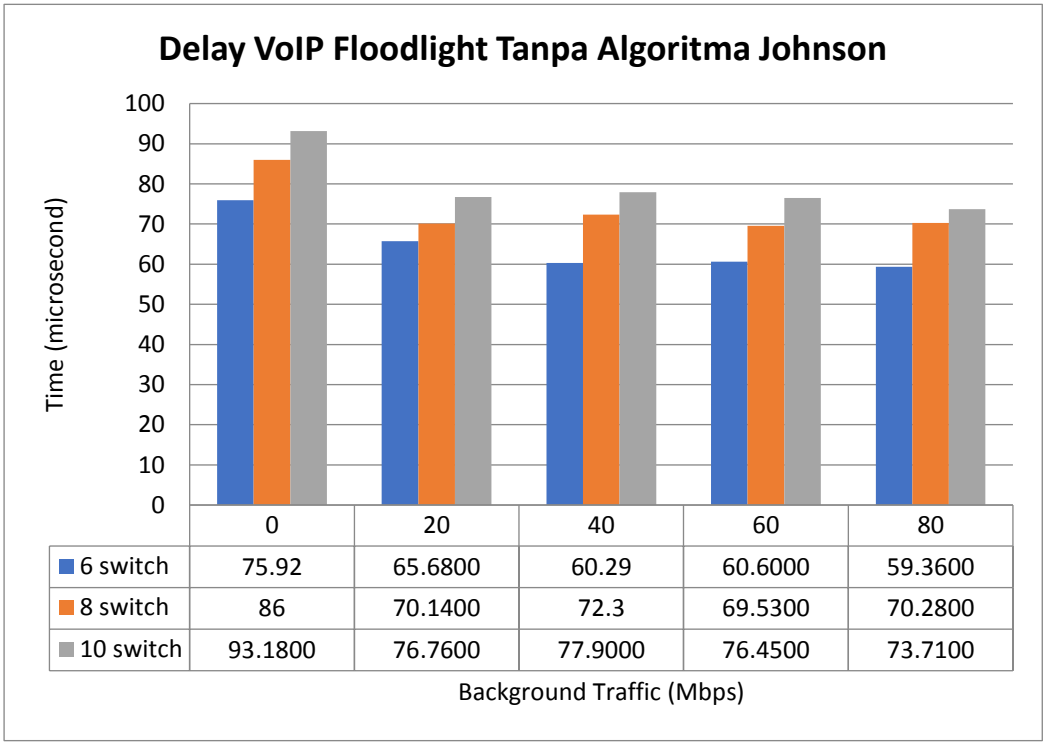


Gambar 4.2 Delay Data Floodlight Dengan Algoritma Johnson

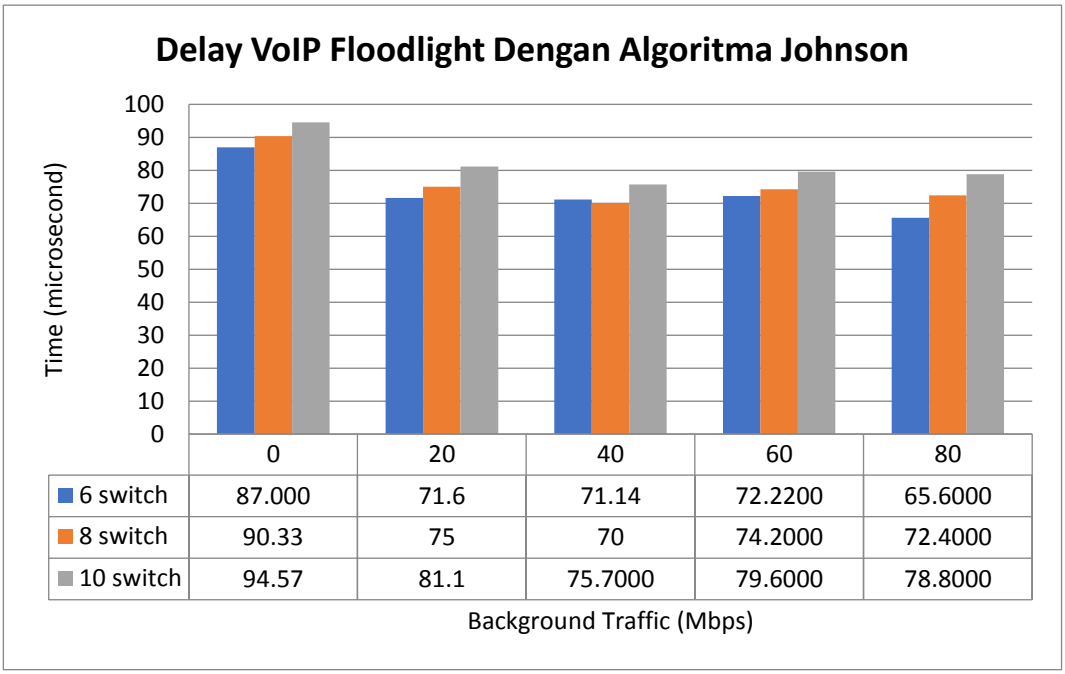


Gambar 4.3 Delay Data OpenDaylight Tanpa Algoritma Johnson

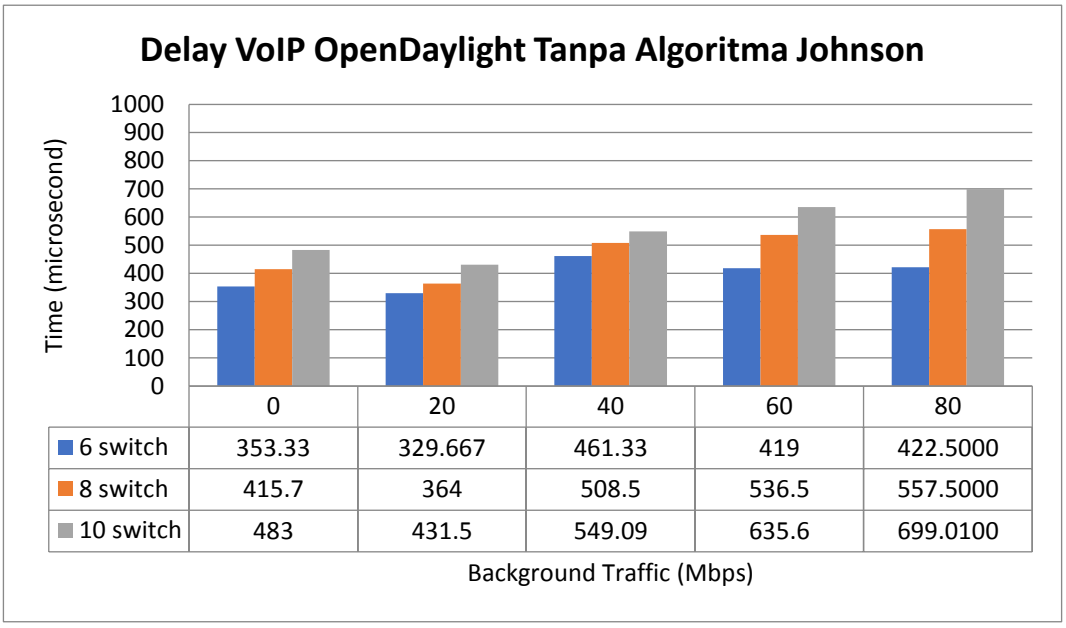
Berdasarkan hasil diatas, secara umum kontroler floodlight tanpa algoritma johnson cenderung mengalami sedikit penurunan *delay* ketika jumlah *background traffic* ditambahkan . Walau terjadi sedikit fluktuasi dari bandwith 0Mbps ke 20Mbps, namun ini memperlihatkan karakteriskiknya. Hal yang sama tidak berlaku untuk kontroler floodlight dengan algoritma johnson. Pada hasil pengujian *delay* menggunakan layanan data diatas, peningkatan jumlah *switch* mempengaruhi nilai *delay* data. Untuk controller OpenDaylight kenaikan jumlah *switch* dan trafik berbanding lurus dengan delay datanya. Setiap terjadi penambahan *switch*, nilai *delay* meningkat dan ketika terjadi kenaikan nilai trafik *delay*-nya cendrung naik. Perubahan yang agak signifikan terlihat ketika kontroler floodlight berada pada trafik 60Mbps ke 80Mbps. Itu terjadi kaerna bandwith link sudah mendekati ambang. Jika kita menggunakan referensi standarisasi QoS ITU.T G.1010, tiga kontroler diatas memenuhi standarnya, karena nilai *delay*-nya tidak lebih dari *60 second.* Nilai *delay* terkecil dimiliki oleh kontroler floodlight tanpa algoritma johnson.



Gambar 4.4 Delay VoIP Floodlight Tanpa Algoritma Johnson



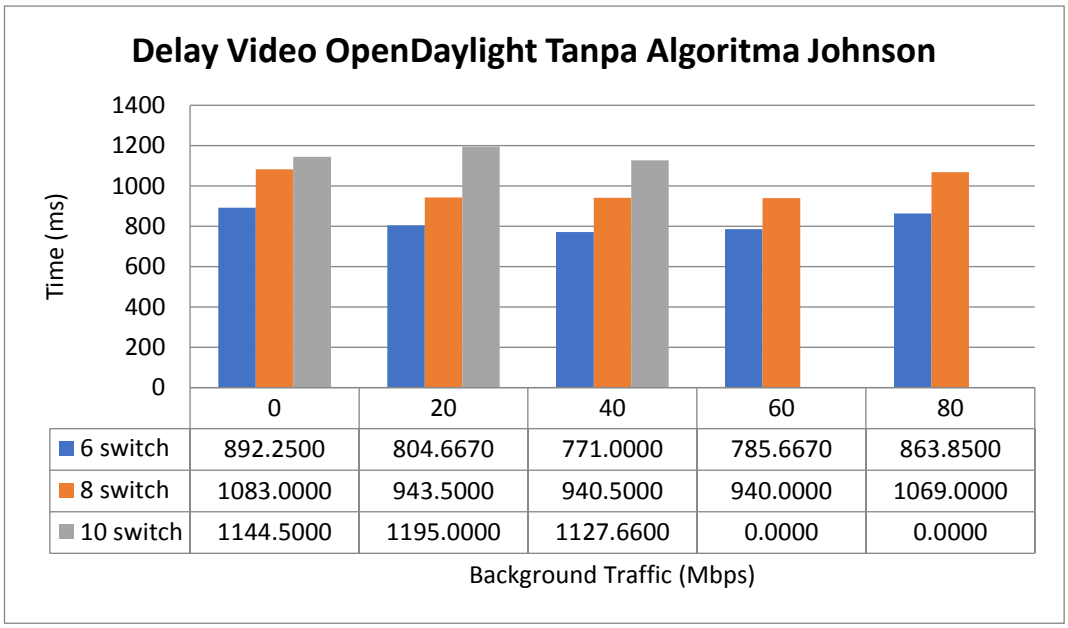
Gambar 4.5 Delay VoIP Floodlight Dengan Algoritma Johnson



Gambar 4.6 Delay Video OepnDaylight Tanpa Algoritma Johnson

Untuk layanan VoIP, kontroler floodlight tanpa algoritma johnson mengalami kenaikan *delay* ketika jumlah *host* nertambah. Semakin bertambah jumlah *host*, *delay*-nya akan naik. Untuk perubahan nilai background traffic tidak terlalu mempengaruhi *delay*-nya. Begitu juga yang terjadi pada kontroler floodlight ketika ditambahkan algoritma johnson dan OpenDaylight tanpa algoritma johnson.

Terjadi kenaikan *delay* ketika jumlah *host* bertambah. Kontroler OpenDaylight mengalami cendrung mengalami peningkatan *delay* ketika *background traffic*-nya meningkat. Berdasarkan standarisasi QoS ITU.T G.1010. kontroler OpenDaylight tidak memenuhi standar ini. Karena *delay*-nya ada yang menyentuh nilai diatas 400ms.



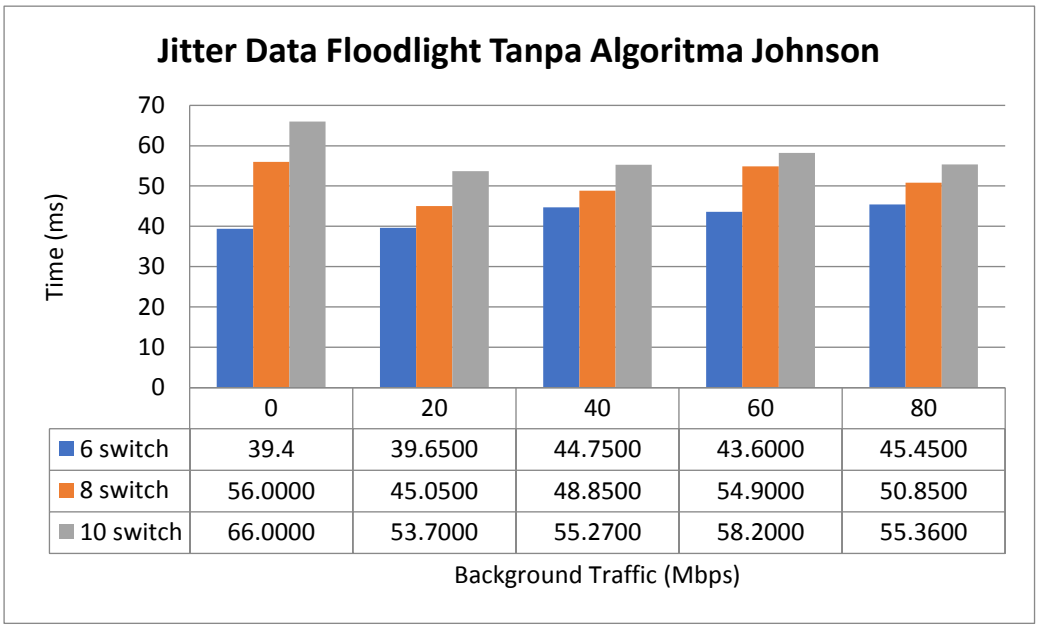
	0	20	40	60	80
6 switch	892.2500	804.6670	771.0000	785.6670	863.8500
8 switch	1083.0000	943.5000	940.5000	940.0000	1069.0000
10 switch	1144.5000	1195.0000	1127.6600	0.0000	0.0000

Gambar 4.7 Delay Video OpenDaylight Tanpa Algoritma Johnson

Pengujian *delay* pada layanan video, hanya bisa dilakukan pada kontroler OpenDaylight. Karena floodlight tidak bisa mengirim packet dalam jumlah besar. Ini bug yang masih belum diperbaiki oleh developer kontroler floodlight sekarang. Berdasarkan hasil pengujian, penambahan jumlah *host* berdampak pada kenaikan nilai *delay*. Jika jumlah *host* bertambah, maka nilai *delay* akan naik. Sorotan khusus pada kontroler OpenDaylight ketika memiliki *background traffic* 60Mbps dan 80Mbps. Topologi dengan model 10 *switch* tidak bisa mengirimkan data video karena mengalami *hang* pada PC yang digunakan. Ini terjadi karena *background traffic* yang sudah mendekati bandwith ambang pada link dan jumlah *switch* yang banyak dengan ukuran data yang besar. Jadi terjadilah keantrian dalam mengirim data, selama antrian juga akan memakan resource lebih banyak dan menyebabkan PC menjadi *hang* dan menyebabkan pengiriman data tidak bisa diteruskan.

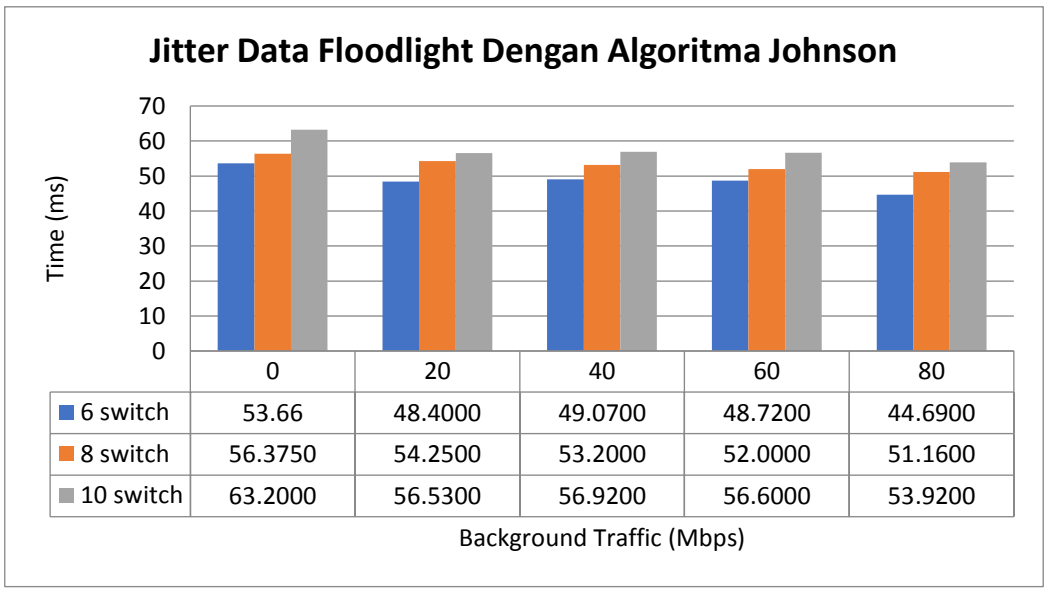
4.1.2. Jitter

Jitter merupakan variasi *delay* antar paket yang terjadi pada jaringan. Besarnya nilai jitter akan sangat dipengaruhi oleh variasi beban trafik dan banyaknya antrian yang terjadi.



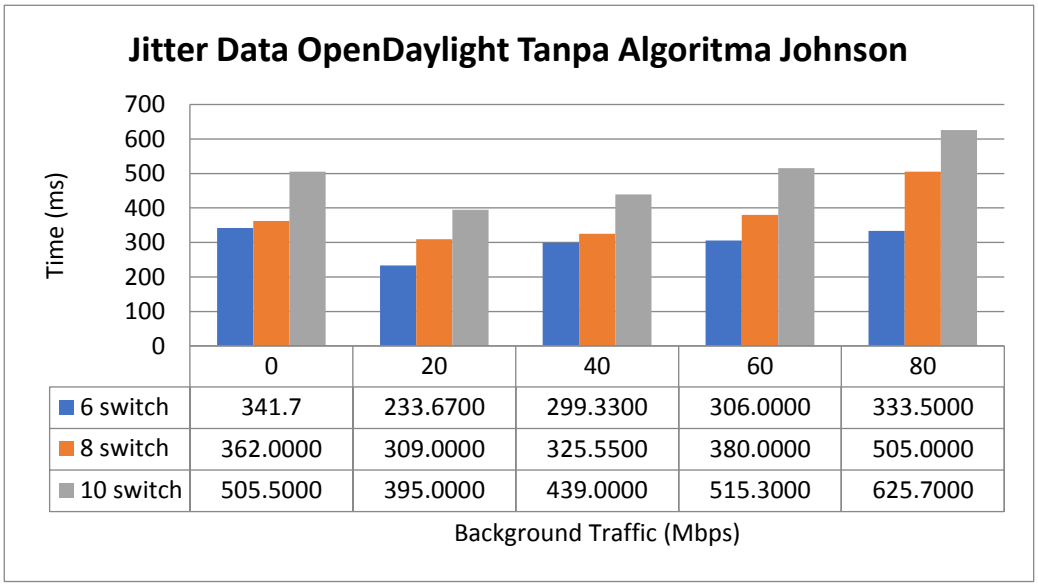
	0	20	40	60	80
6 switch	39.4	39.6500	44.7500	43.6000	45.4500
8 switch	56.0000	45.0500	48.8500	54.9000	50.8500
10 switch	66.0000	53.7000	55.2700	58.2000	55.3600

Gambar 4.8 Jitter Floodlight Tanpa Algoritma Johnson



	0	20	40	60	80
6 switch	53.66	48.4000	49.0700	48.7200	44.6900
8 switch	56.3750	54.2500	53.2000	52.0000	51.1600
10 switch	63.2000	56.5300	56.9200	56.6000	53.9200

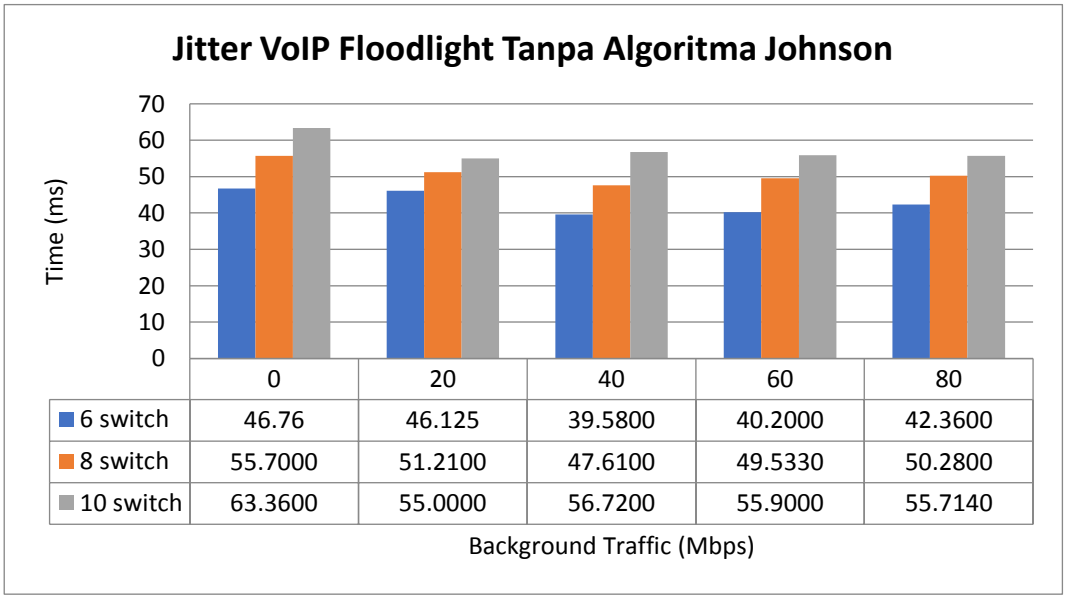
Gambar 4.9 Jitter Data Floodlight Dengan Algoritma Johnson



	0	20	40	60	80
6 switch	341.7	233.6700	299.3300	306.0000	333.5000
8 switch	362.0000	309.0000	325.5500	380.0000	505.0000
10 switch	505.5000	395.0000	439.0000	515.3000	625.7000

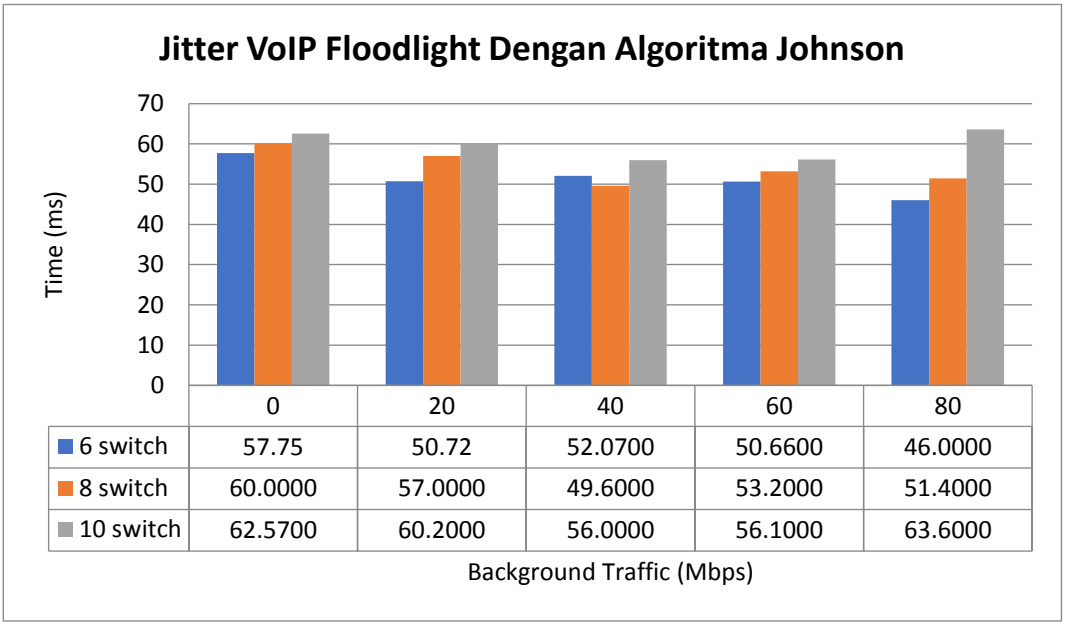
Gambar 4.10 Jitter Data OpenDaylight Tanpa Algoritma Johnson

Berdasarkan hasil pengujian diatas, nilai *jitter* pada layanan data meningkat seiring bertambahnya jumlah *host*. Perubahan *jitter* tiap perubahan *background traffic*-nya tidak terlalu signifikan. Namun nilai *jitter* terkecil dipegang oleh kontroler floodlight tanpa algoritma johnson. Berdasarkan standarisasi QoS ITU.T G.1010. ketiga kontroler ini tidak memenuhi standar. Karena *jitter*-nya melebihi angka 1ms.

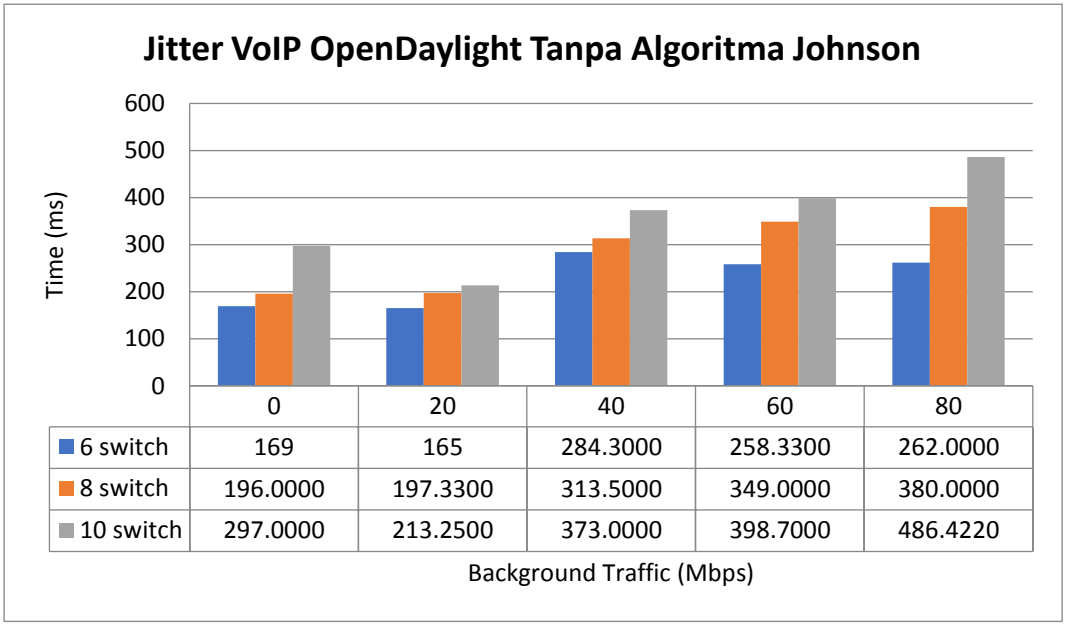


	0	20	40	60	80
6 switch	46.76	46.125	39.5800	40.2000	42.3600
8 switch	55.7000	51.2100	47.6100	49.5330	50.2800
10 switch	63.3600	55.0000	56.7200	55.9000	55.7140

Gambar 4.11 Jitter Data Floodlight Tanpa Algoritma Johnson

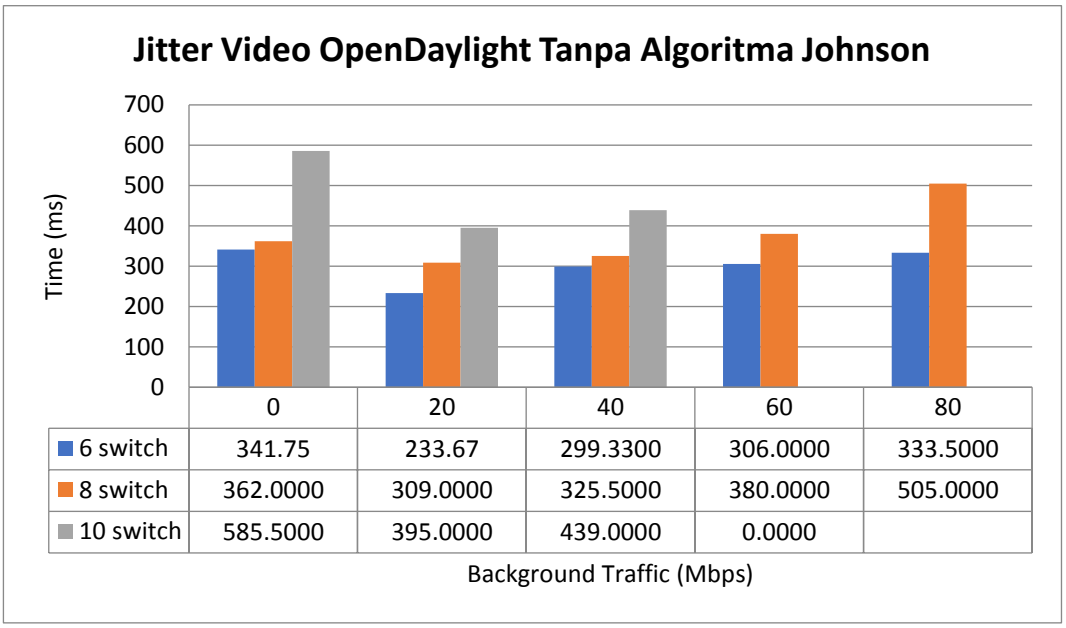


Gambar 4.12 Jitter VoIP Floodlight Dengan Algoritma Johnson



Gambar 4.13 Jitter VoIP OpenDayllight Tanpa Algoritma Johnson

Pada pengujian *jitter* untuk layanan VoIP, secara umum ketiga model kontroler mengalamami kenaikan nilai jitter saat terjadi penambahan jumlah host. Jika dilihat dari standarisasi QoS ITU.T G.1010, tidak ada satupun kontroler yang memenuhi standarisasi tersebut. Karena nilai jitternya lebih dari 1ms. Kontroler floodlight tanpa algoritma johnson lebih baik dalam nilai jitternya karena kontroler ini memiliki nilai jitter terkecil diantara dua kontroler lainnya.

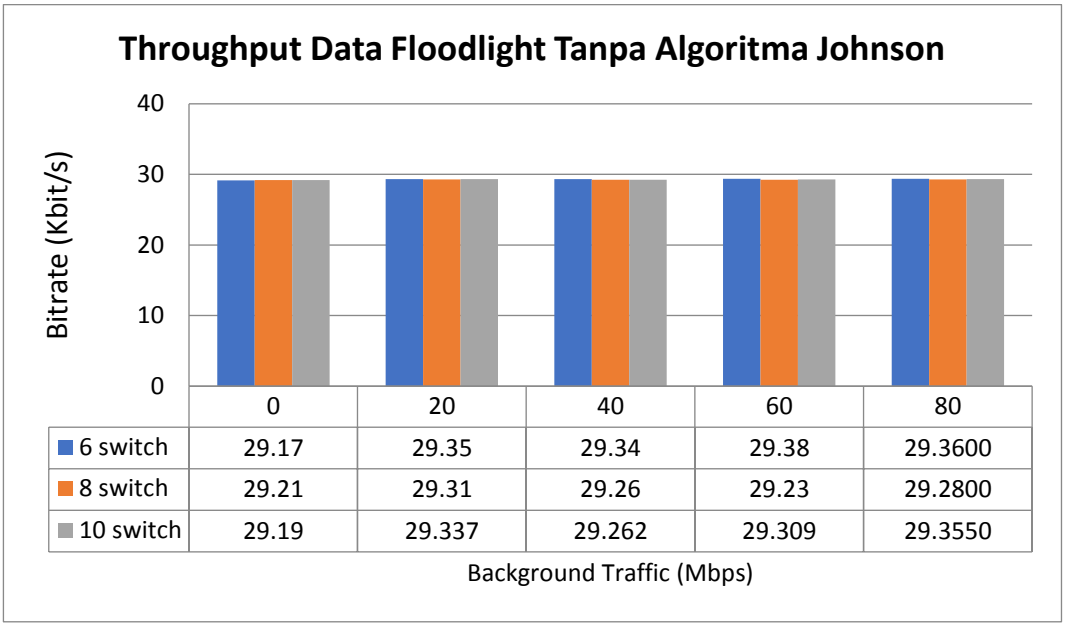


Gambar 4.14 Jitter Video OpenDaylight Tanpa Algoritma Johnson

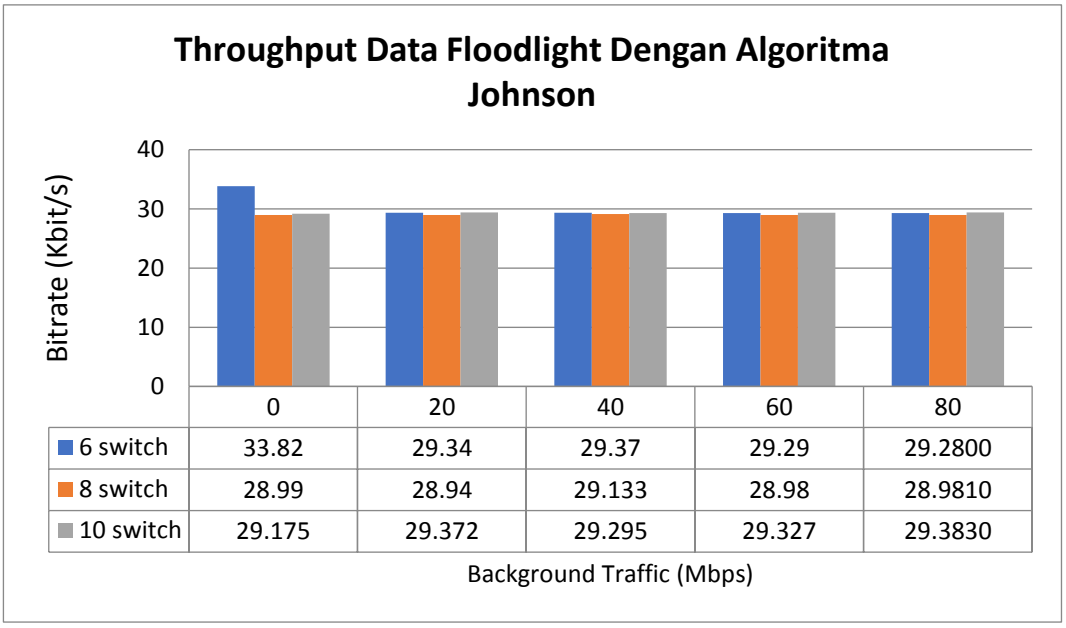
Untuk pengujian jitter pada layanan video hanya bisa dilakukan pada kontroler opendaylight, karena kontroler floodlight tidak bisa meneruskan paket dengan ukuran besar. Jika diteruskan, nilai keterangan jitternya akan memiliki keluaran *nan*. Ini bug yang belum diperbaiki oleh *developer* floodlight. Untuk kontroler opendaylight perubahan jumlah *host* mempengaruhi kenaikan nilai *jitter*. Semakin bertambah jumlah host maka jitternya akan bertambah. Sama seperti pada layanan VoIP, terjadi *hang* pada komputer percobaan saat dilakukan pengujian pada model topologi 10 switch dengan *bandwith* 60Mbps dan 80Mbps. Ini terjadi karena *background traffic* yang sudah mendekati bandwith ambang pada link dan jumlah *switch* yang banyak dengan ukuran data yang besar. Jadi terjadilah keantrian dalam mengirim data, selama antrian juga akan memakan resource dan menyebabkan PC menjadi *hang* dan menyebabkan pengiriman data tidak bisa diteruskan.

4.1.3. Throughput

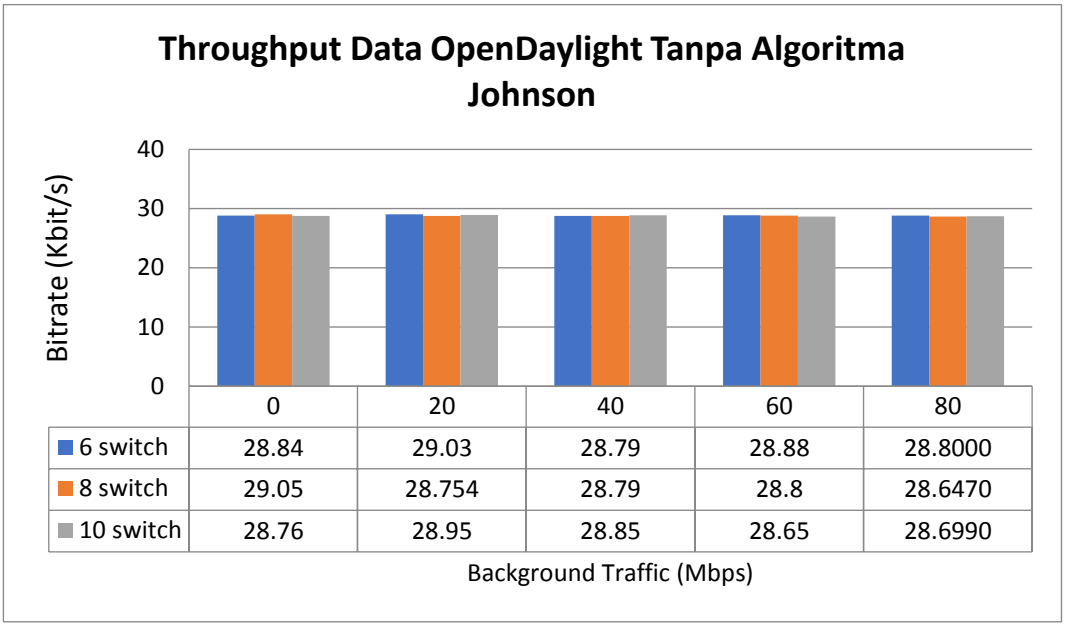
Throughput adalah perbandingan jumlah bit paket yang sukses diterima dengan total waktu pengiriman.



Gambar 4.15 Throughput Data Floodlight Tanpa Algoritma Johnson

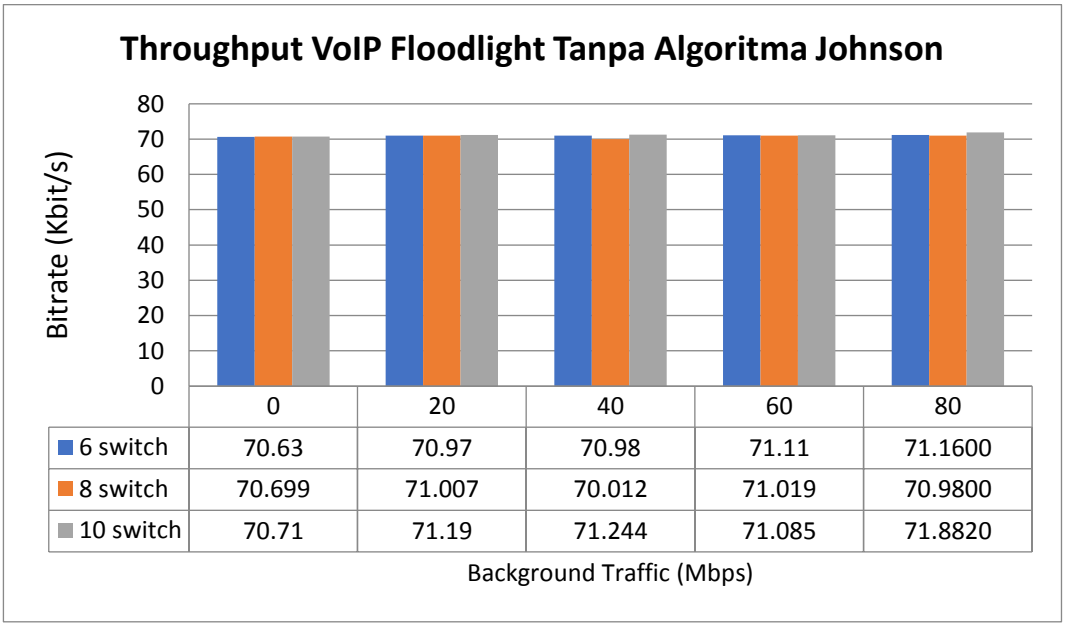


Gambar 4.16 Throughput Data Floodlight Dengan Algoritma Johnson

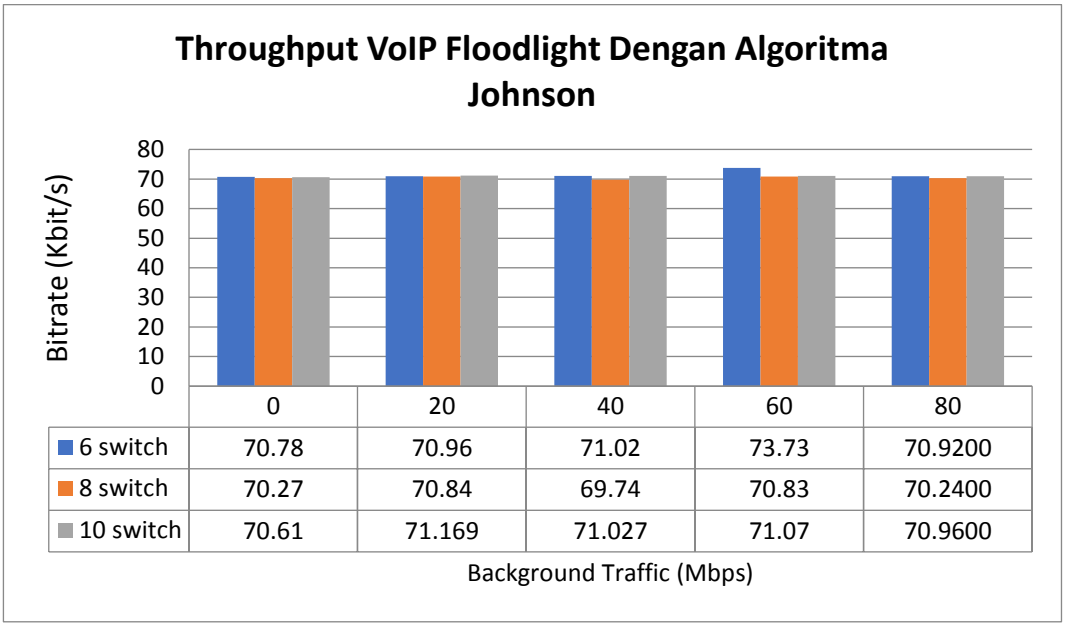


Gambar 4.17 Throughput Data OpenDaylight Tanpa Algoritma Johnson

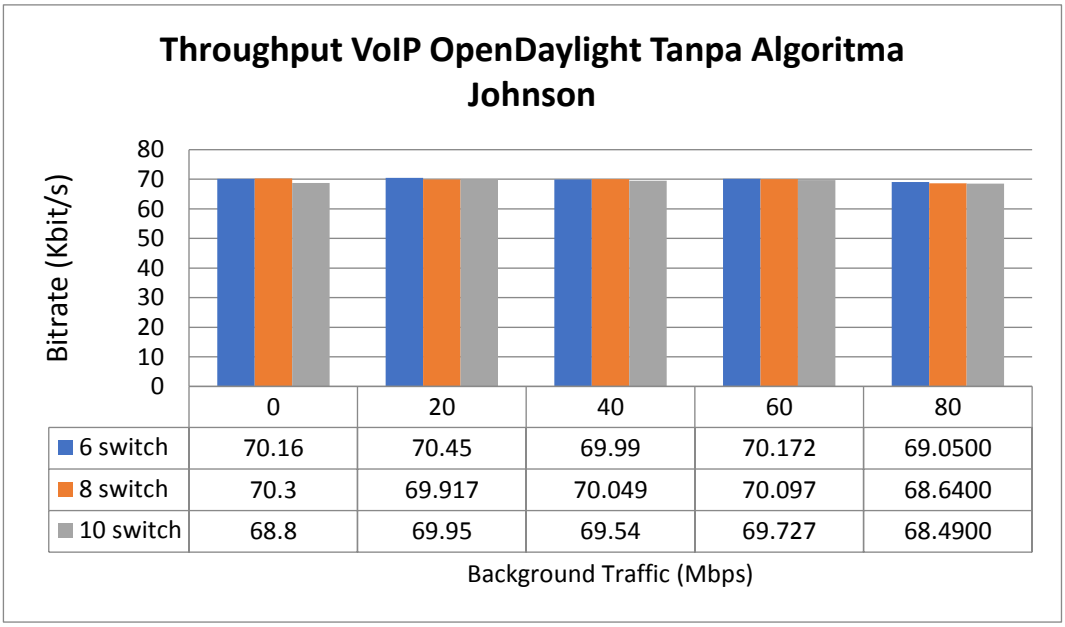
Perubahan nilai *throughput* untuk layanan data pada ketiga kontroler tersebut masih konstan. Secara umum, perubahan jumlah *host* dan kenaikan *background traffic* tidak mempengaruhi nilai thorughputnya. Hal ini dikarenakan besar *background traffic* masih dibawah besar maksimal yang disediakan *link*. Nilai throughput ketiga kontroler diatas masih berkisaran antara 28-29Mbps.



Gambar 4.18 Throughput VoIP Floodlight Tanpa Algoritma Johnson



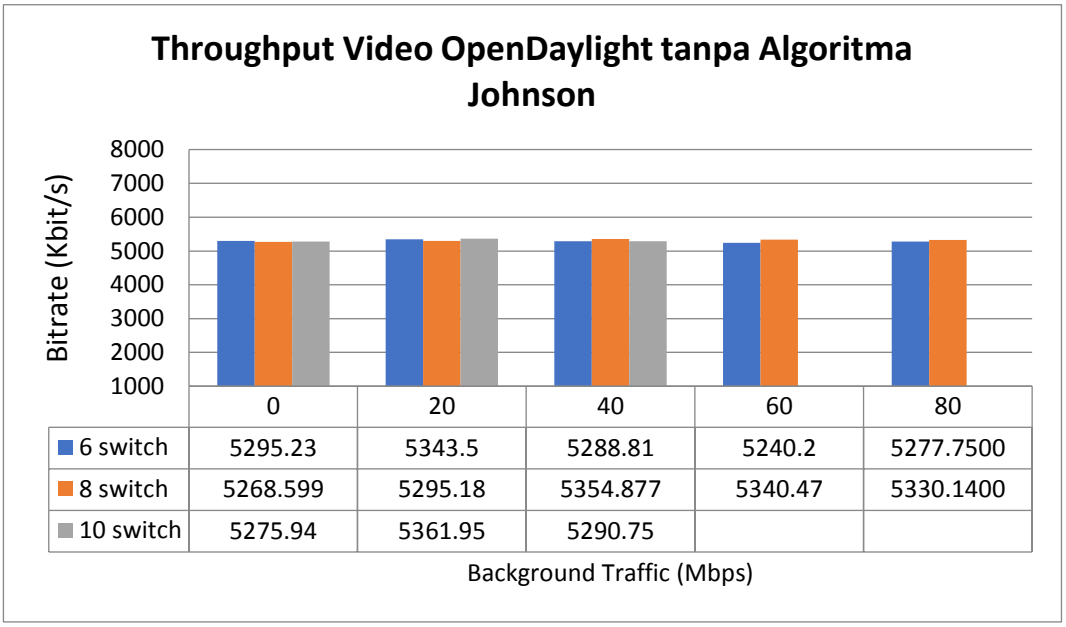
Gambar 4.19 Throughput VoIP Floodlight Tanpa Algoritma Johnson



Gambar 4.20 Throughput VoIP Floodlight Tanpa Algoritma Johnson

Untuk perubahan nilai *throughput* pada layanan VoIP di ketiga kontroler tersebut masih konstan. Secara umum, perubahan jumlah *host* dan kenaikan *background traffic* tidak mempengaruhi nilai thorughputnya. Hal ini dikarenakan besar *background traffic* masih dibawah besar maksimal yang disediakan *link.*

Berdasarkan standarisasi QoS ITU.T G.1010 tidak ada kontroler yang memenuhi standar tersebut karena nilai throughput-nya lebih dari 64kbps.

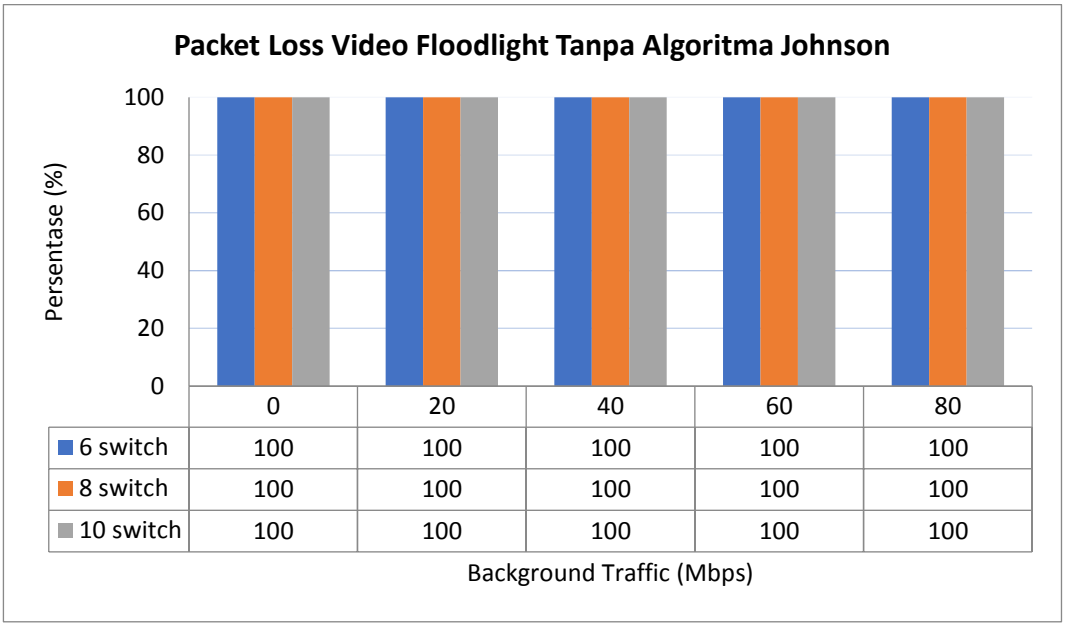


Gambar 4.21 Throughput Video OepnDaylight Tanpa Algoritma Johnson

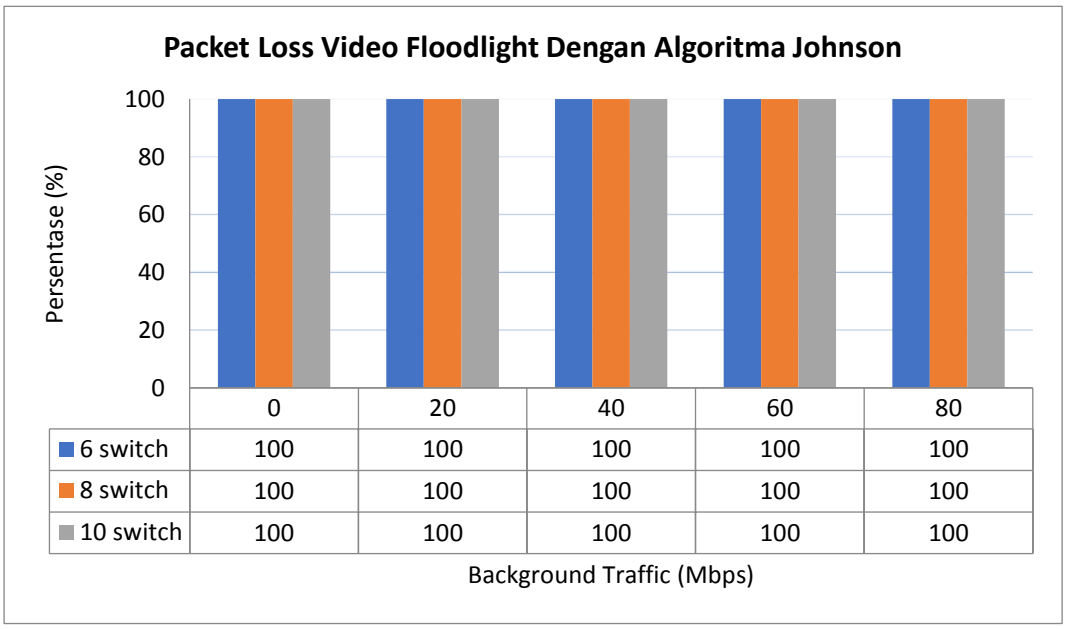
Berdasarkan dengan hasil penelitian data diatas, nilai *throughput* untuk layanan video bersifat konstan dengan *background* traffic 60-80Mbps. Hal ini terjadi karena keterbatasan alat pengujian dan itu terjadi pada bandwith 60-80Mbps. Hal ini juga disebabkan oleh nilai *background traffic* yang tinggi. Namun jika dilihat dari *background traffic* 0-40Mbps nilai *throughput* masih terbilang konstan. Jika yang ditinjau hanya hanya sampai *background traffic* 40Mbps, maka berdasarkan standarisasi QoS ITU.T G.1010 kontroler opendaylight tidak memenuhi standarnya yaitu nilai *throughput*-nya berkisar diangka 5Mbps dan melebihi standar seharusnya yang bernilai 384kbps.

4.1.4. Packet Loss

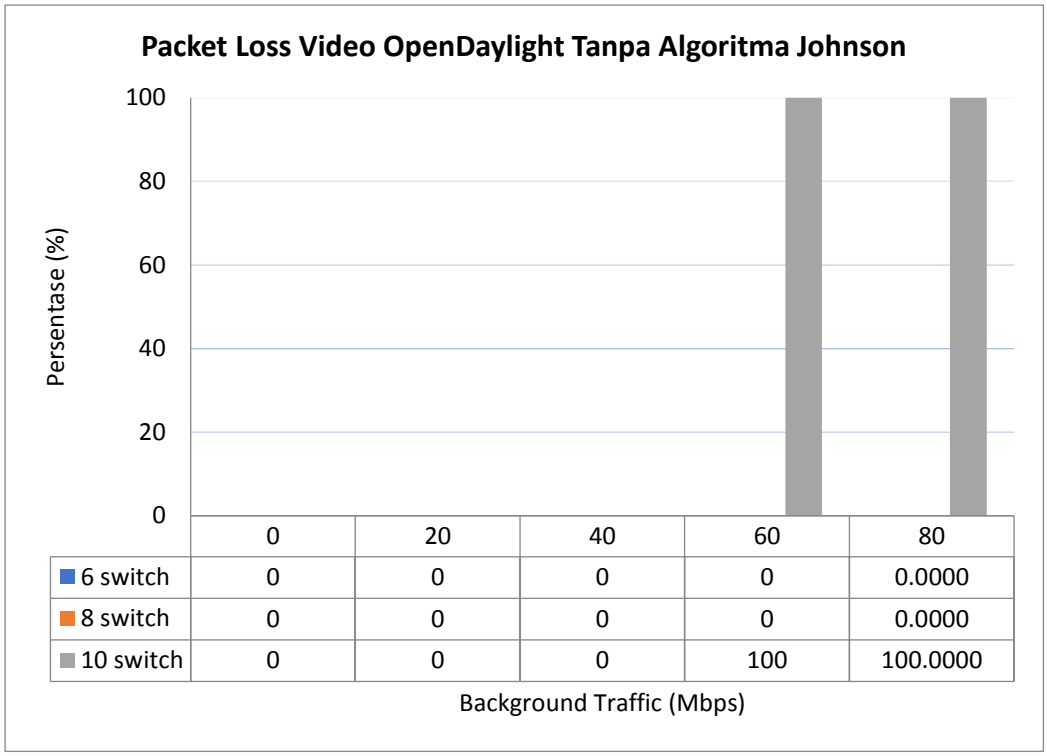
Packet loss ratio adalah persentase perbandingan antara jumlah paket yang gagal diterima oleh *user* dengan jumlah paket yang dikirimkan. Adapun terdapat packet loss adalah pada pengujian berikut:



Gambar 4.22 Packet Loss Video Floodlight Tanpa Algoritma Johnson



Gambar 4.23 Packet Loss Video Floodlight Menggunakan Algoritma Johnson



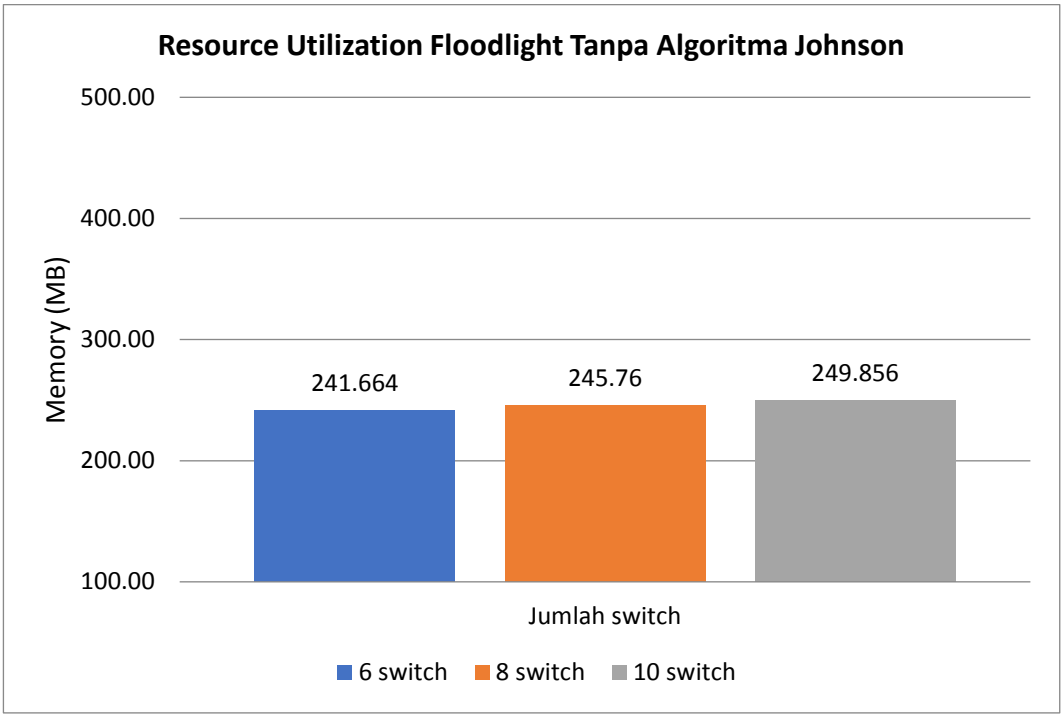
	0	20	40	60	80
6 switch	0	0	0	0	0.0000
8 switch	0	0	0	0	0.0000
10 switch	0	0	0	100	100.0000

Gambar 4.24 Packet Loss Video OpenDaylight Tanpa Algoritma Johnson

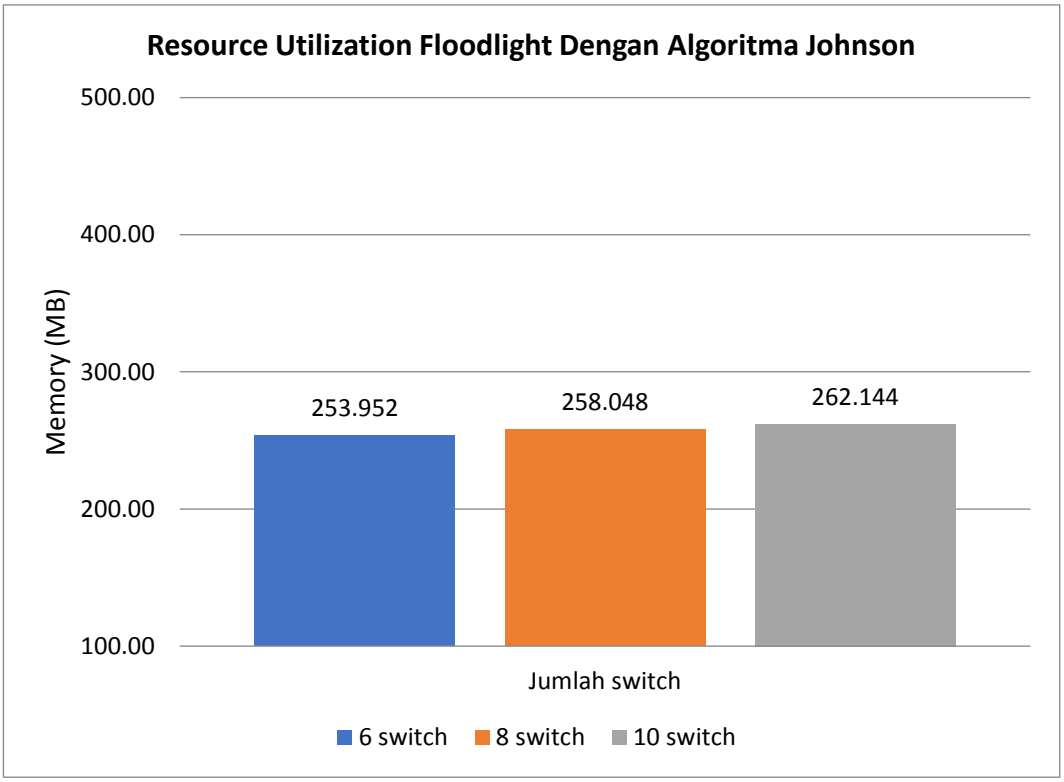
Pada grafik diatas diperlihatkan hasil penelitian packet loss terjadi untuk jenis data video. Untuk kontroler floodlight, baik yang menggunakan algortima johnson atau tanpa algortima johnson tidak bisa mengirimkan layanan berupa video. Ini merupakan *bug* yang belum diperbaiki oleh *developer* kontroler floodlight. Untuk kontroler opendaylight terdapat *packet loss* pada switch berjumlah 10 dengan background traffic 60-80Mbps. Seperti yang sudah dijelaskan sebelumnya, ini terjadi karena keterbatasan *environment*.

4.2. Pengujian Resource Utilization

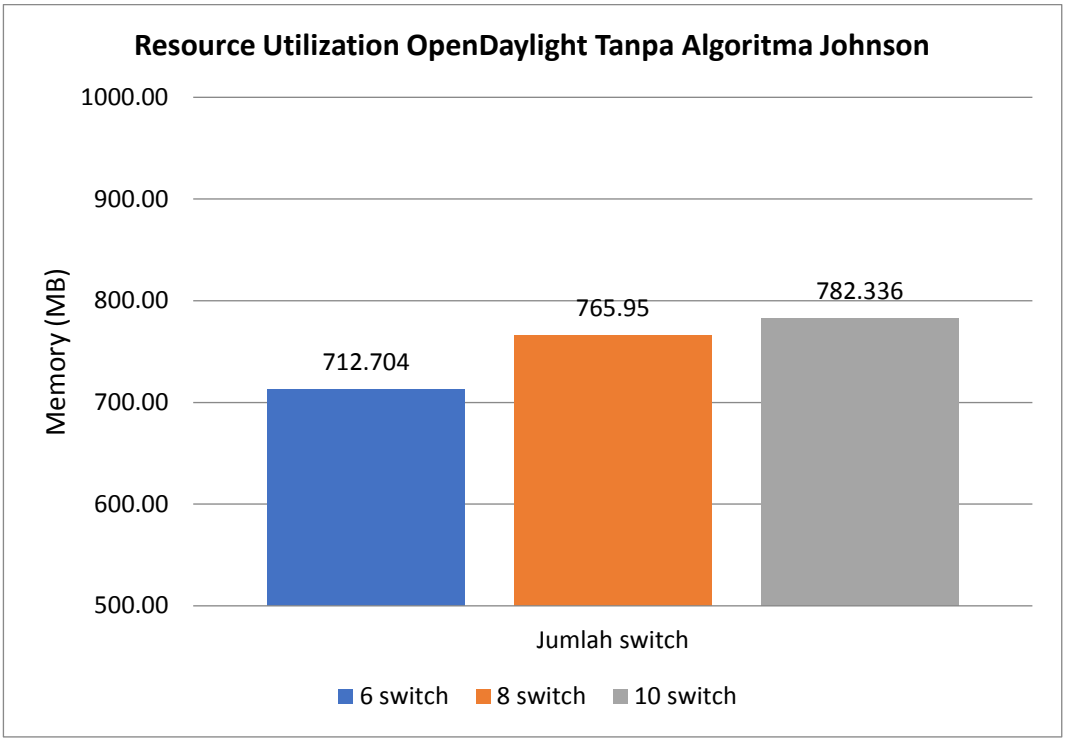
Tujuan dilakukannya pengujian *resource utilization* adalah untuk mengetuhui seberapa besar konsumsi memori yang digunakan oleh kontroler ketika belum memiliki Algoritma Johson dan ketika sudah memiliki Algrotima Johnson didalamnya.



Gambar 4.25 Resource Utilization Floodlight Tanpa Algoritma Johnson



Gambar 4.26 Resource Utilization Floodlight Dengan Algoritma Johnson



Gambar 4.27 Resource Utilization OpenDaylight Tanpa Algoritma Johnson

Berdasarkan grafik diatas, secara umum terjadi peningkatan konsumsi memori pada masing-masing kontroler pada setiap peningkatan jumlah switch. Kontroler OpenDaylight lebih terbilang paling banyak memakan memori sebesar 782.336MB pada jumlah switch 10

.

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berikut kesimpulan dari hasil penelitian tugas akhir ini:

1. Pada pengujian performansi kontroler Floodlight tanpa algoritma Johnson, Floodlight dengan algoritma Johnson dan OpenDaylight tanpa algoritma Johnson mempelihatkan perubahan jumlah *switch* dari 6 *switch*, 8 *switch* dan 10 *switch* berbanding lurus dengan perubahan nilai QoS terutama pada nilai *delay* dan *jitter*. Kontroler Floodlight tanpa algoritma Johnson lebih unggul dibandingkan kedua kontroler lainnya dari untuk layanan data dan VoIP. Ini terlihat dari jumlah *delay* pada layanan data dan VoIP. Adapun nilai *delay* maksimal pada model kontroler ini adalah 88.38ms. Namun untuk pengiriman layanan video, kontroler OpenDaylight lebih baik dari yang lainnya. Karena hanya kontroler OpenDaylight yang bisa mengirimkan paket video. Adapun nilai *delay* maksimal untuk pengiriman layanan video adalah 1195ms.
2. Untuk peningkatan *background traffic* yang berkisar dari 0-80Mbps, secara umum kontroler OpenDaylight memiliki perubahan nilai delay dan jitter yang berbanding lurus sesuai kenaikkan background traffic-nya untuk semua layanan. Perubahan nilai *delay* dan *jitter* untuk semua layanan sering terjadi pada *background traffic* yang bernilai di 60-80Mbps. Pada kontroler floodlight tidak terjadi perubahan seperti ini.
3. Untuk hasil pengujian *resource utilization*, konsumsi memori tertinggi dimiliki oleh kontroler OpenDaylight. OpenDaylight mengkonsumsi memori sebanyak 782MB (19.1%) dari total memori *virtual machine* yaitu4094MB. Lalu diikuti oleh Floodlight dengan algoritma Johnson sebanyak 262MB (6.4%) dan terakhir Floodlight tanpa algoritma johnson sebanyak 245MB (6%).
4. Berdasarkan standarisasi ITU-T G.1010 kontroler Floodlight tanpa algoritma Johnson, Floodlight menggunakan algoritma Johnson dan OpenDaylight hanya memenuhi standarisasi *delay* untuk layanan data saja.

Untuk nilai *jitter* dan *throughput*-nya tidak memenuhi standar disemua layanan. Adapun nilai *delay* maksimal untuk layanan data pada tiga kontroler tersebut yaitu 88ms, 82ms dan 636ms dimana itu belum melebihi 60s.

5.2. Saran

Saran yang diberikan untuk penelitian selanjutnya:

1. Menggunakan algoritma lain dan metode lain agar menemukan hasil QoS yang lebih baik pada kontroler floodlight.
2. Melakukan penelitian lebih lanjut pada kontroler OpenDaylight agar nilai QoS nya lebih baik dan menemukan cara bagaimana cara menambahkan algoritma lain pada kontroler OpenDaylight.

DAFTAR PUSTAKA

[1] Siddharth Valluvan,T.Manoranjitham, V.Nagarajan , "A STUDY ON SDN CONTROLLERS," *Siddharth Valluvan*et al. /International Journal of Pharmacy & Technology* , p. 9, 2016.

[2] Shie-Yuan Wang, Hung-Wei Chiu, Chih-Liang Chou, "Comparisons of SDN OpenFlow Controllers over EstiNet: Ryu vs. NOX," *The International Symposium on Advances in Software Defined Networks,* p. 6, 2015.

[3] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient SDN Controllers," *IEEE Conference on Network Softwarization (NetSoft),* p. 6, 2015.

[4] Mr. Sachin Ashok Vanjari, Dr. R. B. Ingle, P.G. Student, Dean Second Shift, "Network Traffic Control Using Distributed Control Plane of Software Defined Networks," *I nternational Journal of Computer Science Trends and Technology (IJCST) – Volume 3 Issue 5,* p. 5, 2015.

[5] FAID Souad, Mohamed MOUGHIT, Noureddine IDBOUFKER , "OpenFlow Controllers Performance Evaluation," *International Journal of Emerging Research in Management &Technology ISSN: 2278-9359 (Volume-5, Issue-5),* p. 7, 2016.

[6] KaramjeetKaur, VipinGupta, "Performance Analysis Of Python Based OpenFlow Controllers," *International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences,* p. 4, 2016.

[7] SHIVA ROWSHANRAD, VAJIHE ABDI, MANIJEH KESHTGARI, "PERFORMANCE EVALUATION OF SDN CONTROLLERS: FLOODLIGHT AND OPENDAYLIGHT," *IIUM Engineering Journal, Vol. 17,,* p. 11, 2016.

[8] Pooja, Manu Sood, "SDN and Mininet: Some Basic Concepts," *Int. J. Advanced Networking and Applications* , p. 4, 2015.

[9] Selcuk Yazar, Eldem Uchar, "POX CONTROLLER PERFORMANCE FOR OPENFLOW NETWORKS," *Management and Education Vol. 1,* p. 10, 2013.

[10] Sugam Agarwal, Murali Kodialam, T. V. Lakshman, "Traffic Engineering in Software Defined Networks," *Proceedings IEEE INFOCOM,* p. 9, 2013.

[11] M. Sisov, Building a Sofware-Defined Networking System with OpenDaylight Controller, Helsinki: Helsinki Metropolia University of Applied Sciences, 2016.

[12] Seitz, N., NTIA/ITS,, "ITU-T QoS Standards for IP-Based Networks," *IEEE Communication Magazine,* p. 6, 2003.

[13] Bruce Hartpence, Rossi Rosario, "Software Defined Networking for Systems and Network Administration Programs," *The USENIX Journal of Education in System Administration,* p. 15, 2016.

[14] Wawan Apriadi, R. Rumani, Sofia Naning Hertiana, "SIMULATION AND PERFORMANCE ANALYSIS OF SOFTWARE DEFINE NETWORK (SDN) USING BELLMAN FORD ALGORITHM AS ROUTING," p. 18, 2016.