PERANCANGAN DAN ANALISIS *SOFTWARE DEFINED NETWORK* PADA JARINGAN LAN : PENERAPAN DAN ANALISIS METODE PENJALURAN *PATH CALCULATING* MENGGUNAKAN ALGORITMA DIJKSTRA

DESIGN AND ANALYSIS SOFTWARE DEFINED NETWORKING FOR LAN NETWORK : APPLICATION AND ANALYSIS ROUTING METHOD PATH CALCULATING USING DIJKSTRA'S ALGORITHM

Brayan Anggita Linuwih[1], Agus Virgono[2], Budhi Irawan[2]

[123] Prodi S1 Teknik Komputer, Fakultas Teknik Elektro, Universitas Telkom

[1]brayananggita@students.telkomuniversity.ac.id [2]avirgono@telkomuniversity.ac.id,
[3]budhiirawan@telkomuniveristy.co.id,

Abstrak

Kebutuhan akan teknologi perangkat jaringan di berbagai perusahaan mengalami perkembangan yang sangat pesat. Performansi jaringan, penambahan konfigurasi yang semakin kompleks dan besar, bagian kontrol jaringan pun akan semakin rumit, tidak fleksibel dan sulit untuk diatur. *Software Defined Networking* (SDN) adalah sebuah pradigma jaringan di mana *control plane* terpisah dari *forwarding plane*. SDN diharapkan mampu menjalankan metode- metode yang terdapat pada jaringan konvensional seperti IP *forwarding, routing*.

Dalam tugas akhir ini dilakukan pembuktian penerapan layanan routing menggunakan metode *path calculating* algoritma dijkstra sebagai rekayasa kontrol penjaluran pada jaringan LAN berbasis *Software-Defined Network* dan menganalisa perbandingan kinerja metode pemilihan jalur terbaik dalam lalulintas jaringan pada jaringan LAN berbasis *Software-Defined Network* dengan jaringan konvensional yang juga diterapkan algoritma dijkstra dengan arsitektur yang sama namun tidak memiliki perangkat *control plane*. Pembuktian dilakukan dengan melakukan emulasi jaringan sebuah yang terdiri dari 11 buah *switch* yang saling terhubung dengan perangkat control plane sebagai pengendali sebuah jaringan.

Hasil pengujian performansi penerapan algoritma dijkstra berbasis jaringan SDN menunjukkan bahwa nilai dari keempat parameter QoS masih berada pada nilai yang menjadi standar ITU-T G.1010 serta memiliki nilai *delay* dan *jitter* yang lebih baik dibandingkan penerapan OSPF berbasis jaringan konvensional. Nilai QoS untuk UDP pada layanan data, VoIP masing-masing bervariasi dan berada di kisaran (5.17 – 145.81) ms untuk *delay*, (0.05 - 0.93) ms untuk *jitter*, (13.22 – 73.41) kbps untuk *throughput*, 8,05 % - 33,81 % untuk *packet loss* di semua skenario yang telah dibuat. Adapun nilai QoS untuk TCP pada layanan data masing-masing bervariasi dan berada di kisaran (10.32 – 20.21) ms untuk *delay*, (0.5 - 3.02) ms untuk *jitter*, (38.12 – 38.27) kbps untuk *throughput* di semua skenario yang telah dibuat.

Kata kunci : ***ALGORITMA DIJKSTRA, SDN, SNHx, OSPF***

Abstract

The need for technology network devices in a variety of companies experiencing rapid growth. Network performance, additional configuration of increasingly complex and large parts of the network control would be more complicated, inflexible and difficult to manage. Software Defined Networking (SDN) is a paradigm where the network control plane is separated from the forwarding plane. SDN is expected to run methods contained in conventional networks such as IP forwarding, routing.

In this final project verification application routing services using the path calculating algorithms dijkstra as engineering controls routing in the network LAN-based Software-Defined Network and analyzing the comparative performance of the method of selecting the best path in the network traffic on the LAN network based Software-Defined Network with conventional networks which also applied the algorithm dijkstra with the same architecture but does not have the control plane. Verification was conducted by a network emulation which consists of 11 pieces of switches that are connected to the control plane as a network controller.

Application performance testing results dijkstra algorithm based SDN network shows that the value of the four QoS parameters remain on the value of the standard ITU-T G.1010 and has a value of delay and jitter better than conventional network-based OSPF implementation. Value QoS for UDP on data services, VoIP each varies and is in the range (5.17 – 145.81) ms for the delay, (0.05 - 0.93)ms for jitter, (13.22 – 73.41) kbps for throughput, 8,05 % - 33,81 % for packet loss in all scenarios that have been made. The value of QoS for TCP on data services and each varies in the range of (10.32 – 20.21) ms for the delay, (0.5 - 3.02) ms for jitter, (38.12 – 38.27) kbps throughput in all scenarios that have been made.

Keywords: Dijkstra's Algorithm, SDN, SNHx, OSPF

1. Pendahuluan

1.1. Latar Belakang

Jaringan konvensional menggunakan algoritma khusus yang diterapkan pada perangkat yang didedikasikan untuk mengontrol dan memantau aliran data dalam jaringan, mengelola penjaluran dan menentukan bagaimana perangkat yang berbeda dapat saling berhubungan dalam jaringan. Secara umum algoritma *routing* dan seperangkat peraturan diimplementasikan dalam komponen perangkat keras khusus seperti *Application Specific Integrated Circuits* (ASICs) [1]. ACISc dirancang untuk melakukan operasi tertentu, contoh sederhana yaitu Paket forwarding. Pada penerimaan paket oleh perangkat routing, router menggunakan seperangkat aturan yang tertanam dalam *firmware* untuk menemukan perangkat tujuan serta menentukan penjaluran untuk paket tersebut. Pada umumnya paket data disampaikan kepada tujuan yang sama ditangani dengan cara yang sama. Operasi tersebut biasanya terdapat pada perangkat routing yang murah. Untuk perangkat routing yang lebih mahal dapat memperlakukan berbagai jenis paket dalam berbagai tata krama berdasarkan kondisinya. Sebagai contoh, *router* Cisco memungkinkan pengguna untuk menandai prioritas arus yang berbeda. Dengan demikian kita dapat mengatur ukuran antrian dan metode penjaluran paket di setiap *router* secara langsung. Pengaturan router secara langsung memungkinkan kontrol prioritas dan penjaluran menjadi lebih efisien terhadap kemacetan lalu lintas jaringan [2].

Sebuah solusi untuk masalah tersebut adalah pelaksanaan aturan penanganan data modul perangkat lunak pada embedding system dalam hardware. Metode ini memungkinkan administrator jaringan untuk memiliki kontrol atas lalu lintas jaringan dan karena itu memiliki potensi besar untuk lebih meningkatkan kinerja jaringan dalam hal efisiensi penggunaan sumber informasi jaringan. Gagasan seperti ini didefinisikan dalam teknologi inovatif, disebut *Software-Defined Networking* (SDN) [5]. Teknologi SDN memanfaatkan kemampuan untuk membagi data plane dari control plane di *router* dan *switch* [6]. *Control plane* dapat mengirimkan perintah ke perangkat keras *data plane* (*router* atau *switch*) [7]. Paradigma ini memberikan pandangan dari seluruh jaringan, sehingga dapat membantu melakukan perubahan secara global tanpa konfigurasi perangkat-sentris pada setiap unit *hardware* [8].

Dalam tugas akhir ini dilakukan pembuktian penerapan layanan *routing* menggunakan metode *path calculating* algoritma dijkstra sebagai rekayasa kontrol penjaluran pada jaringan LAN berbasis *Software-Defined Network* dan menganalisa perbandingan kinerja metode pemilihan jalur terbaik dalam lalulintas jaringan pada jaringan LAN berbasis *Software-Defined Network* dengan jaringan konvensional yang juga diterapkan algoritma dijkstra dengan arsitektur yang sama namun tidak memiliki perangkat *control plane*. Pembuktian dilakukan dengan melakukan emulasi jaringan sebuah yang terdiri dari 11 buah *switch* yang saling terhubung dengan perangkat *control plane* sebagai pengendali sebuah jaringan.

1.2. Perumusan Masalah

Dengan kondisi dan ruang lingkup yang akan ditentukan, masalah yang akan dibahas pada penelitian Tugas Akhir ini adalah:

1. Dalam membuktikan layanan routing pada emulasi jaringan LAN berbasis *software-defined network* perlu dibuat fitur untuk membangun layanan *routing* yaitu dengan menerapkan *path calculating* menggunakan algoritma dijkstra sebagai rekayasa *control* penjaluran pada jaringan LAN berbasis *Software-Defined Network* pada perangkat *control plane*.
2. Dalam jaringan LAN berbasis *Software-Defined Network* dibutuhkan informasi tentang topologi yang bertujuan untuk memantau kondisi jaringan yang nantinya akan dilakukan kalkulasi rute sesuai kepadatan jaringan.
3. Untuk melakukan analisa perbandingan kinerja penjaluran diperlukan perancangan jaringan LAN konvensional menggunakan protokol penjaluran OSPF sebagai penerapan prinsip kerja algoritma Dijksta didalamnya dengan mengemulasikan jaringan tersebut dalam aplikasi GNS3.
4. Untuk menentukan perbandingan performansi penjaluran pada kedua teknologi tersebut diperlukan pengukuran pada *bandwidth, delay, jitter, troughput, packet loss* dan trace route pada kedua teknologi tersebut.

1.3. Tujuan

Tujuan dari penelitian Tugas Akhir ini adalah:

1. Menerapkan aturan penjaluran algoritma dijkstra pada jaringan berbasis *Software-Defined Network* untuk mengetahui nilai *end to end latency* .
2. Mendapatkan nilai *latency traceroute* untuk menganalisa kinerja penjaluran paket pada jaringan LAN berbasis *Software-Defined Network* .
3. Mendapatkan nilai QoS pada jaringan untuk membandingkan kinerja penjaluran paket metode *path calculating* algoritma dijkstra pada jaringan LAN berbasis *Software-Defined Network* dan jaringan konvensional yang telah diterapkan algoritma dijkstra didalamnya.

1.4. Batasan Masalah

Adapun batasan masalah pada Tugas Akhir ini adalah sebagai berikut :

1. Dalam membuktikan layanan *routing* pada emulasi jaringan LAN berbasis *software-defined network* perlu dibuat fitur untuk membangun layanan *routing* yaitu dengan menerapkan *path calculating* menggunakan algoritma dijkstra sebagai rekayasa control penjaluran pada jaringan LAN berbasis *Software-Defined Network* pada perangkat *control plane.*
2. Dalam jaringan LAN berbasis *Software-Defined Network* dibutuhkan informasi tentang topologi yang bertujuan untuk memantau kondisi jaringan yang nantinya akan dilakukan kalkulasi rute sesuai kepadatan jaringan.
3. Untuk melakukan analisa perbandingan kinerja penjaluran diperlukan perancangan jaringan LAN konvensional menggunakan protokol penjaluran OSPF sebagai penerapan prinsip kerja algoritma Dijksta didalamnya dengan mengemulasikan jaringan tersebut dalam aplikasi GNS3.
4. Untuk menentukan perbandingan performansi penjaluran pada kedua teknologi tersebut diperlukan pengukuran pada *bandwidth, delay, jitter, troughput, packet loss dan trace route* pada kedua teknologi tersebut.

1.4 Metodologi Penyelesaian Masalah

Langkah yang dilakukan dalam proses pengerjaan Tugas Akhir adalah :

a. Studi literature : Pencarian informasi yang bersumber dari buku,media dan diskusi yang bertujuan menunjang selesainya tugas akhir ini
b. Perancangan dan implementasi alat : Melakukan perancangan sistem kerja alat sesuai dengan parameter yang diinginkan dan mengaplikasinya.
c. Analisa sistem : Mengamati hasil dari sistem yang dikerjakan dan menganalisis serta menyimpulkan masalah yang ada.
d. Penarikan kesimpulan : Dari keseluruhan tahapan yang telah dilakukan diatas ditambah dengan masukan dari dosen pembimbing maka dapat diambil kesimpulan dari hasil yang telah dilakukan.

2. Dasar Teori

2.1. *Software Defined Networking*

Software-defined networking (SDN) adalah sebuah konsep pendekatan jaringan komputer dimana sistem pengontrol dari arus data dipisahkan dari perangkat kerasnya. Awal mula terciptanya teknologi Software-defined networking dimulai tidak lama setelah Sun Microsystems merilis Java pada tahun 1995 [11] [12] [13], namun pada saat itu belum cukup membangunkan para periset untuk mengembangkan teknologi tersebut. Baru pada tahun 2008 *Software defined networking* ini dikembangkan di UC Berkeley and Stanford University [14]. Dan kemudian teknologi tersebut mulai dipromosikan oleh Open Networking Foundation yang didirikan pada tahun 2011 untuk memperkenalkan teknologi SDN dan OpenFlow.

Pada teknologi jaringan konvensional, sistem pembuat keputusan kemana arus data dikirimkan dibuat menyatu dengan perangkat kerasnya. Namun di dalam teknologi SDN memiliki dua karakteristik. Pertama, SDN memisah antara *control plane* dari *data plane*. Kedua, SDN menggabungkan *control plane* setiap perangkat menjadi sebuah kontroler yang berbasiskan programmeable software. Sehinga sebuah kontroler tersebut dapat mengontrol banyak perangkat dalam sebuah *data plane*. Di dalam SDN sebuah jaringan tersentralisasi dalam sebuah kontroler yang berbasiskan *software* yang dapat memelihara jaringan secara keseluruhan, Sehingga dapat mempermudah dalam mendesain dan mengoperasikan jaringan karena hanya melalui sebuah *logical point*. Berikut adalah keunggulan *Software-defined networking* :

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

2.2 Algoritma Dijkstra

Pada algoritma Dijkstra, ditampilkan dalam Algoritma 1, mempertahankan simpul dari grafik V di dua set memisah dan melengkapi simpul $S \subseteq V$ dan $Q \subseteq V$; S, diawali dengan $\emptyset$ (baris 2), termasuk semua simpul yang jalur terpendek dari s telah ditentukan. Q adalah prioritas antrian dinisialisasi oleh semua $x \in V$ (Jalur 4). Q

mengetik dengan dist [x], yang merupakan panjang lintasan terpendek saat ini diketahui dari s ke x ∈ Q. dist[] yang diinisialisasi (baris 1-2) suchthatfor x ∈ V \ s dist [x] = ∞ sementara dist [s] = 0. S di inisialisasi ∅.

Algoritma Dijkstra

***Input*: G=(*V*, *E*)**
***Output*: *d*[|*V*|]**

```
1  (∀x ≠ s) dist[x]=+∞ //Initialize dist[]
2  dist[s]=0
3  S = ∅
4  Q = V // Keyed by dist[].
5  while Q = ∅ do
6        u = extract_min(Q)
7        S = S ∪{u}
8        foreach vertex v ∈ Adj(u) do
9              dist[v]=min(dist[v],dist[u]+w(u,v))
10             //"Relax" operation.
```

Gambar 2.1 Algoritma *Djikstra* [13]

Pada setiap siklus, algoritma Dijkstra mengekstrak *vertex* u ∈ Q dengan minimal dist [] di Q. Kemudian, untuk setiap v tetangga u set dist [v] = min (dist [v], dist [u] + w (u , v)) (thisiscalledthe bersantai operasi). u kemudian ditambahkan ke S. Algoritma Dijkstra mempertahankan invarian bahwa setiap kali u dipilih dari Q maka Pastilah itu dist [u] = δ (u), di mana δ (x) adalah jalur terpendek dari s ke x. Pembuktian adalah dengan induksi. Awalnya hal ini berlaku untuk s. Kemudian, setiap kali u dipilih, karena itu node dengan minimal dist [] di Q berikut bahwa dist [u] harus sama dengan delta (u), jika tidak, kita mencapai kontradiksi. Hal ini penting untuk dicatat bahwa Algoritma Dijkstra dalam bentuk dasarnya seperti biasanya muncul dalam buku ini dirancang untuk menemukan jalan terpendek untuk semua simpul dari grafik.

2.3. Protokol Routing OSPF (*Open Shortest Path First*)

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

Teknologi yang digunakan oleh routing protokol ini adalah teknologi link-state yang memang didesain untuk bekerja dengan sangat mangkus dalam proses pembaruan informasi rute. Prinsip perutean link-statesangat sederhana. Sebagai pengganti dalam menghitung rute terbaik dengan cara terdistribusi, semua perute mempunyai peta jaringan dan menghitung semua rute terbaik dari peta ini. Peta jaringan tersebut disimpan dalam sebuah basis data dan setiap recorddalam basis data tersebut menyatakan sebuah pautan (link) dalam jaringan. Record-recordtersebut dikirimkan oleh perute yang terhubung langsung dengan masing-masing pautan. Karena setiap perute perlu memiliki peta jaringan yang menggambarkan kondisi terakhir topologi jaringan yang lengkap, setiap perubahan dalam jaringan harus diikuti oleh perubahan dalam basis data link-stateyang terletak di setiap perute. Perubahan status pautan yang dideteksi perute akan mengubah basis data link-state perute tersebut, kemudian perute mengirimkan perubahan tersebut ke perute - perute yang lain.

2.4. Mininet

Mininet [7] adalah emulator jaringan SDN yang dapat mensimulasikan kinerja antara end-host, switch, router, controller, dan link dalam sebuah kernel Linux. Mininet merupakan sebuah sistem virtualisasi yang dapat menggambarkan jaringan yang besar dengan hanya menggunakan sebuah laptop. Mininet bersifat open source, sehingga proyek yang telah dilakukan berupa source code, scripts, dan dokumentasi yang dapat dikembangkan oleh siapapun.

2.5. QoS (Quality of Service)

Quality of Service (QoS) adalah efek kolektif dari kinerja layanan yang menentukan derajat kepuasan seorang pengguna terhadapt sebuah layanan.Ada banyak parameter yang menjadi acuan untuk QoS di antaranya yaitu *throughput, jitter,* dan *packet loss.*

2.5.1. *Latency (maximum packet delay)*

Delay [10] adalah perbedaan waktu di antara waktu kirim dengan waktu terima sebuah paket. Dalam tugas akhir ini *delay* yang dimaksudkan adalah *one way delay*, yaitu waktu rata – rata pengiriman setiap paket dalam perjalanan dari satu titik kirim ke satu titik terima.

2.5.2.*Jitter*

Jitter [10] adalah variasi *delay* yang diakibatkan oleh panjang antrian dalam suatu pengolahan data dan *reassemble* paket data di akhir pengiriman akibat kegagalan sebelumnya. *Jitter* merupakan variasi selisih dari setiap *delay* dengan *delay* selanjutnya. Dalam tugas akhir ini *jitter* yang dimaksud adalah nilai dari *one-way delay variation*.

2.5.3.*Troughput*

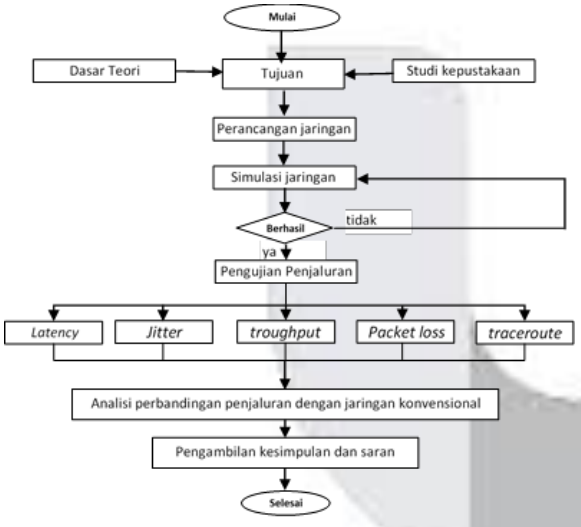
Throughput [10] adalah jumlah bit yang sukses diterima dari satu terminal tertentu di dalam sebuah jaringan, dari suatu titik jaringan ke titik jaringan lainnya dibandingkan dengan total waktu pengiriman.

3. Perancangan & Implementasi Sistem

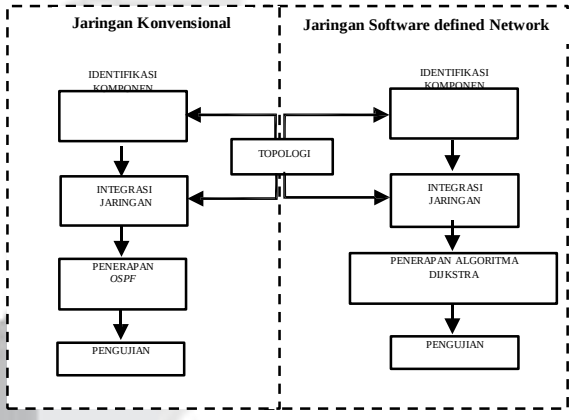
3.1 Model Perancangan Sistem

Secara garis besar sistem tersebut ditujukan untuk menerapkan aturan penjaluran pada jaringan berbasis *Software defined network* menggunakan algoritma *dijkstra*. Dan selanjutnya membandingkan dua teknologi penjaluran yaitu teknologi jaringan berbasis router (konvensional) dan teknologi berbasis *Software defined network*. Jaringan di simulasikan dalam 2 bentuk sesuai basis teknologi yang akan diujikan.

Perancangan jaringan dimulai dengan menentukan topologi yang akan digunakan, penerapan topologi pada emulator berbasis jaringan konvensional dan jaringan Software defined network, pengintegrasian komponen dalam topologi dan penerapan aturan penjaluran yaitu berbasis algoritma dijkstra. Selanjutnya penerapan skenario sebagai bahan uji dan analisa penjaluran.



Gambar 3.1 *FlowChart* Deskripsi umum Sistem



Gambar 3.2 Diagram perbandingan routing Jaringan konvensional dan jaringan SDN

Pada Tugas Akhir ini, topologi jaringan diimplementasikan pada program emulasi jaringan GNS3 untuk jaringan konvensional dan miniNet untuk jaringan *Software define Network*. Implementasi dan konfigurasi jaringan berupa integrasi komponen dan aturan penjaluran algoritma dijkstra disesuaikan berdasarkan basis teknologi jaringan. Untuk memferifikasi apakah sistem dapat berjalan maka di lakukan pengujian dari setiap *host* sumber ke *host* tujuan.

3.2 Identifikasi Kebutuhan Sistem

3.2.1 Spesifikasi Perangkat Lunak

Untuk melakukan simulasi tugas akhir ini dibutuhkan perangkat keras berupa sebuah *laptop* spesifikasi berikut ini:

Spesifikasi	***Laptop***
Processor	Intel(R) Core(TM) i3 CPU M370 2.40GHz
RAM	6 GB (6,14 GB *useable)*
Operating System	Ubuntu 14.04 LTS 64 bit
Fungsi	*Control plane* : Ryu Controller & simulasi jaringan

Selain perangkat keras ada perangkat lunak yang dibutuhkan dalam simulasi tugas akhir ini yaitu sebagai berikut:

- Iperf, digunakan untuk membangkitkan trafik
- Distributed Internet Traffic Generator (D-ITG), digunakan untuk membangkitkan trafik

- Wireshark, digunakan untuk capture trafik

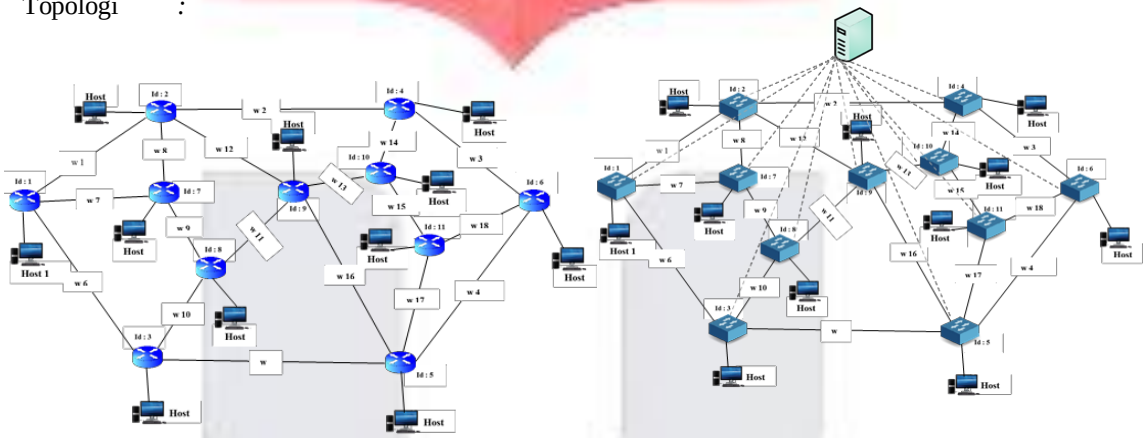
Laptop digunakan sebagai *controller* dengan pemasangan Ryu *virtual environment* di dalamnya. Kemudian PC digunakan sebagai *data plane* dengan memasang simulator Mininet. Selain itu, di dalam PC juga dipasang D- ITG dan Iperf sebagai pembangkit trafik, serta Wireshark untuk perekaman paket. Pemisahan *controller* dan *data plane* pada komputer yang berbeda ini untuk mendapatkan hasil yang maksimal dari setiap simulasi yang dilakukan. *Laptop* hanya bekerja sebagai *controller* dan PC hanya bekerja menjalankan Mininet sebagai simulasi *data plane*.

3.3 Perancangan

3.3.1 Perancangan topologi

Dalam perbandingan metode penjaluran untuk kedua teknologi jarigan digunakan sebuah desain topologi jaringan dimana dalam implementasi topologi ada sejumlah perbedaan, yaitu pada penggunaan perangkat penjaluran. Pada jaringan konvensional perangkat penjaluran menggunakan *router* karena *control plane* dan *data plane* menyatu. Sedangkan jaringan berbasis *Software defined Network* perangkat penjaluran yang digunakan menggunakan *Switch Open flow* sebagai *data plane* dan *controller* sebagai *control plane.* Topologi yang akan di implementasilan yaitu dengan spesifikasi sebagai berikut :

a. Jumlah Router : 11 buah (berlaku untuk jaringan konvensional)
b. Jumlah Switch Openflow : 11 buah (berlaku untuk jaringan *SDN*)
c. *Controller* : 1 buah (berlaku untuk jaringan *SDN*)
d. Jumlah Host : 11 buah
e. Jumlah *link* : 18
f. *Weight (cost)* : *static* sesuai skenario.
g. Topologi :



Gambar 3.3 Desain Topologi pada Teknologi Konvensional dan Teknologi SDN

3.4. Pengujian

3.4.2.Rencana Pengujian

Terdapat dua jenis pengujian pada tugas akhir ini, yaitu pengujian awal menggunakan PING dan *traceroute* pada dan pengujian Parameter QoS yang diukur dalam pengujian yaitu *delay*, *jitter*, *packet loss*, dan *throughput*.

a. Sistematika pengujian
berikut ini adalah rincian sistematika pengungujian dari pengujian awal dan pengujian QoS :

- Pengujian Awal pada Pengujian penjaluran dilaksanakan dengan melakukan pengiriman paket ICMP sebesar 84 kilobyte dan tidak di berikan backround traffic. Pengujian ditentukan node sumber ke node tujuan dipilih secara acak. Mewakili setiap router dengan 3 buah skenario. Untuk pengujian traceroute dilakukan dengan melakukan pengujian pada h1 sebagai node sumber dan h6 sebagai node sumber. Untuk mengetahui kinerja penjalurkan dilakukan percobaan menggunakan 3 buah skenario yang telah di tentukan. Dan akan di uji nilai end to end latency dan pemilihan jalur dari setiap skenario.
- Pengujian QoS dilakukan menggunakan topologi custom. Dipilih h6 untuk dijadikan node tujuan pengiriman, kemudian h1 sebagai node sumber. Percobaan dilakukan sebanyak 30 kali. Pengambilan data juga diambil pada jaringan konvensional yang nantinya dijadikan bahan perbandingan dengan jaringan Software defined Network dari segi QoS.

Dalam pengukuran ini dibangkitkan trafik [2] UDP dan TCP, yaitu:

a. Trafik Data dengan *inter-departure time* (IDT) konstan sebesar 100 pps dengan ukuruan paket yang terdistribusi *poisson* μ = 48 bytes, sehingga membutuhkan *bandwidth* 38,4 kbps

b. VoIP dengan G.711 codec tanpa voice activation detection (VAD) sebanyak 100 pps dengan ukuran paket 80 bytes, sehingga membutuhkan bandwidth 70,4 kbps.
c. Selain pemberian tiga jenis trafik tersebut secara bersamaan, dibangkitkan pula background traffic yang bervariasi (25 Mbps, 50 Mbps, 75 Mbps, dan 100 Mbps) menggunakan Iperf. Percobaan ini dilakukan untuk mengetahui pengaruh kepadatan trafik terhadap pemilihan jalur dengan melihat kualitas dari pengiriman trafik data yang sampai di penerima.

5. Pengujian dan Analisis

Analisa Perbandingan Kinerja Penjaluran Berdasarkan Pengujian.

Berdasarkan hasil pengujian yang telah dilakukan pada tugas akhir ini, langkah akhir pada bab IV adalah membandingkan kinerja penjaluran pada jaringan berbasis *Software Defined Networking* dengan jaringan berbasis konvensional. Nilai yang didapat dari hasil pengujian selanjutnya di bandingkan dengan standarisasi QoS ITU-T G. 1010 *End-user multimedia QoS Categories.* Berikut standard yang akan digunakan :

Tabel 5.1: Referensi Standarisasi QoS ITU.T G.1010 [10]

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

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

*Tabel 5.2 :*Perbandingan hasil pengujian

	SDN	KONVENSIONAL	ITU-T G1010	
			SDN	Konv.
Latency ping	9 – 13.795 ms	40 – 67,45 ms	-	-
Latency traceroute	13 – 32.356 ms	50 – 95.333 ms	-	-
Delay UDP	5.17 – 135.57 ms	204 - 503 ms	*preffered*	*preffered*
Delay TCP	10.32 - 20.12 ms	421 - 959 ms	*preffered*	*preffered*
Delay VoIP	5.11 – 145.81 ms	162 - 641 ms	*preffered*	*acceptable*
Jitter UDP	0.23 - 0.93 ms	31.29 - 88.3 ms	NA	NA
Jitter TCP	0.5 - 3.02 ms	10.21 - 29.28 ms	NA	NA
Jitter VoIP	0.05 - 0.52 ms	31.45 - 48.12 ms	< 1 ms	> 1 ms
Troughput UDP	13,22 - 35.53 kbps	10,49 - 30,48 kbps	NA	NA
Troughput TCP	38,12 – 38,27 kbps	38,11 – 38,39 kbps	NA	NA
Troughput VoIP	48,35 – 73,41 kbps	45,78 – 72,28 kbps	NA	NA
Packet Loss UDP	24,81 % - 30,58 %	27,67 % - 33,81 %	>0% untuk *bandwidth* 75 – 100 mbps	>3% untuk *bandwidth* 75 – 100 mbps
Packet Loss VoIP	8,05 % - 17,29 %	17,28 % - 22,94 %	>0% untuk *bandwidth* 75 – 100 mbps	>3% untuk *bandwidth* 75 – 100 mbps

Nilai *latency* pada pengujian awal yaitu pengujian pengiriman paket ICMP dan *traceroute* menunjukkan bahwa algoritma dijkstra yang diterapkan pada jaringan berbasis *Software Defined Networking* lebih kecil dibanding jaringan OSPF pada jaringan berbasis konvensional. Hal tersebut terbukti karena pada teknologi SDN, perangkat *controller* melakukan kalkulasi dan menentukan jalur untuk setiap *route* pengiriman. Selain itu apabila terdapat perubahan kondisi jaringan, *controller* dapat menyesuakian jalur dan memperbaharui daftar *route*.

Berdasarkan standar ITU-T G. 1010 *End-user multimedia QoS Categories*, *delay* baik data UDP, data TCP dan VoIP berada dalam standar tersebut yaitu *preffered* (< 15 detik) dan *acceptable* (< 400 ms) untuk *delay* pada VoIP jaringan berbasis konvensional . Namun terdapat perbedaan cukup signifikan karena pemberian *background traffic* untuk setiap sknario. Nilai pengujian *delay* UDP, TCP dan VoIP membuktikan bahwa *delay* pada jaringan berbasis SDN lebih baik. Karena semakin kecil nilai *delay*, menentukan ketercapaian data yang dikirimkan dari sumber ke tujuan . Nilai *jitter* pada teknologi berbasis SDN menunjukkan hasil yeng lebih kecil dibanding nilai *jitter* pada jaringan berbasis konvensional. *Jitter* berhubungan erat dengan *latency*. Jika *latency* adalah seberapa lama sebuah paket data sampai dari suatu drvice ke device lain, maka *jitter* adalah seberapa variasinya waktu tempuh tersebut. Semakin kecil nilai *jitter* menunjukkan bahwa jaringan tersebut stabil, sedangkan nilai *jitter* yang besar menunjukkan jaringan tersebut tidak stabil atau penuh ketika terjadi perpindahan data .

Pada hasil pengukuran *troughput* menunjukkan bahwa kedua jaringan menunjukkan hasil cukup baik pada pengiriman data TCP, hal tersebut dikarenakan sifat TCP yang *connection-oriented* dan *reliable.* Sedangkan pada pengiriman UDP nilai *troughput* pada jaringan berbasis SDN sedikit lebih baik dibandingkan jaringan berbasis

konvensional membuktikan bahwa sifat UDP yang *connectionless* dan *unreliable*, artinya tidak ada jaminan paket sampai.

Dapat disimpulkan bahwa algoritma dijkstra pada jaringan berbasis SDN lebih baik daripada OSPF pada jaringan berbasis konvensional. Hal tersebut terbukti dikarenakan pada jaringan SDN menggunakan sistem tersentral yaitu pemisahan antara *control plane* dan *data plane*. Pada saat SNHX-IP yang berbasis RYU *controller* dijalankan, sistem tersebut langsung mengkalkulasi jalur dari setiap node berdasarkan bobot sisi pada topologi yang di pasang. Apabila terjadi perubahan bobot atau kepadatan jaringan maka *controller* akan secara otomatis mengitung dan mengubah jalur sehingga dapat mempengaruhi nilai QoS pada jaringan tersebut.

6. Kesimpulan dan Saran

6.1. Kesimpulan

1. Layanan *routing* dengan metode *path calculating* algoritma dijkstra pada jaringan berbasis *Software-Defined Network* dapat dijalankan dengan nilai *latency* ping tanpa *backround traffic* sekitar 9 – 13.795 ms dan berada pada standar ITU-T G.1010 karena berada dibawah 250 ms.
2. Jaringan menggunakan dijkstra memiliki *latency traceroute* dengan nilai antara 13 – 32.356 ms melewati 3 – 6 *node* untuk mencapai *host* tujuan. sistem tersebut langsung mengkalkulasi jalur dari setiap node berdasarkan bobot sisi pada topologi yang di pasang. Apabila terjadi perubahan bobot atau kepadatan jaringan maka *controller* akan secara otomatis mengitung dan mengubah jalur sehingga mempu menekan nilai latency.
3. Jaringan yang menggunakan dijkstra lebih baik dibandingkan dengan jaringan yang menggunakan OSPF. Seluruh hasil pengujian membuktikan bahwa teknologi SDN lebih baik, latency ping sebesar 9 – 13.795 ms, latency traceroute sebesar 13 – 32.356 ms, delay UDP, TCP, dan VoIP berturut-turut sebagai berikut 5.17 – 131.57 ms, 10.32 - 20.12 ms, 5.18 - 5.58 ms. Nilai jitter UDP, TCP, dan VoIP berturut-turut sebagai berikut 0.23 - 0.93 ms, 0.5 - 3.2 ms, 5.11 – 145.81 ms. Nilai troughput UDP, TCP, dan VoIP berturut-turut 13,22 - 35.53 ms, 38,12 – 38,27 ms, 48,35 – 73,41 ms, Nilai Packet Loss UDP, VoIP berturut-turut 24,81 % - 30,58 % dan 8,05 % - 17,29 %. Hal tersebut terjadi karena pada jaringan SDN menggunakan sistem terdistribusi yaitu pemisahan antara control plane dan data plane pada jaringan dengan membagi beban kerja sehingga mampu mempengaruhi nilai QoS.

6.2. Saran

Saran yang dapat diajukan untuk penelitian selanjutnya adalah:

1. Diharapkan kedepanya dapat menambahkan kalkulasi bobot pada node atau *node weight* agar mendapatkan hasil yang lebih baik.
2. Implementasi secara real dengan menggunakan jaringan *fisik*.

DAFTAR PUSTAKA

[1] S. Ortiz, “Software-defined networking: On the verge of a breakthrough?” Computer, vol. 46, no. 7, pp. 10–12, Jul. 2013.

[2] Fei Hu, Qi Hao, and Ke Bao, “A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation”, IEEE communication surveys & tutorials, vol. 16, no. 4, fourth quarter 2014

[3] H. Kim and N. Feamster, “Improving network management with software defined networking, ” IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, Feb. 2013.

[4] S. Agarwal, M. Kodialam, and T. V. Lakshman, “Traffic engineering in software defined networks,” in Proc. IEEE INFOCOM, Apr. 2013, pp. 2211–2219.

[5] S. Fang, Y. Yu, C. H. Foh, and K. M. M. Aung, “A loss-free multipathing solution for data center network using software-defined networking approach,” IEEE Trans. Magn., vol. 49, no. 6, pp. 2723–2730, Jun. 2013.\

[6] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking, ” IEEE Commun. Mag., vol. 51, no. 2, pp. 136–141, Feb. 2013.

[7] S. Daset al., “Packet and circuit network convergence with OpenFlow,”inProc. Opt. Fiber Commun. Conf. Expo., Mar. 2010, pp. 1–3.

[8] Naqvi, H. A. (2015, August). *MPLS in SNHx - a Networking Application using RYU SDN Framework*.

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

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

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

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