ANALISIS PERFORMANSI *LOAD BALANCING* DENGAN METODE PEMILIHAN JALUR BEBAN TERKECIL PADA SDN (*SOFTWARE DEFINED NETWORK*)

ANALYSIS PERFORMANCE OF LOAD BALANCING USING PATH WITH THE SMALLEST LOAD SELECTION METHOD ON SDN (SOFTWARE DEFINED NETWORK)

Adi Arief Wicaksono, Sofia Naning Hertiana, dan Indrarini Dyah Irawati
Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom

adiarief@student.telkomuniversity.ac.id, sofiananing@telkomuniversity.ac.id, indrarini@telkomuniversity.ac.id

Abstrak

Pada jaringan berbasis *Software Defined Network* (SDN), jumlah pengguna mendorong peningkatan akan kebutuhan laju data untuk mengakses berbagai layanan. Pada sisi lain, jaringan SDN dengan banyaknya pengguna/user akan mengakibatkan link *overload* (penuh) yang akan mengakibatkan kemacetan bahkan *packet loss*. Oleh karena itu diperlukan suatu metode *Load Balancing* (penyeimbang beban) pada SDN yang akan mengatur *Load* (beban) pada lintasan dari *node* sumber ke *node* tujuan. Pada metode *Load Balancing* ini diharapkan mempunyai *quality of service* yang tetap terjaga dan dapat mengantisipasi adanya beban yang berlebihan pada salah satu link. Dalam Jurnal ini, disimulasikan penerapan dari penyeimbang beban pada jaringan SDN. Adapun *tools* yang akan digunakan, antara lain *Mininet* sebagai data planenya dan *Floodlight* sebagai kontrol planenya serta *iperf* dan D-ITG sebagai alat bantu untuk analisisnya. Topologi yang digunakan yaitu topologi *Abilene* dengan data yang dialirkan untuk pengujian berupa paket data dan video streaming.

Kata kunci : *Load Balancing*, SDN, *Floodlight*

Abstract

In the network-based Software Defined Network (SDN), the number of users will boost the data rate needs to access various services. On the other hand, SDN network with many users Will lead to a link over load (full) which will result in a collision or even packet loss. Therefore we need a load balancing method on SDN to set the load on the path from the source node to the destination node. On this load balancing method is expected to have a good quality of service and can anticípate their excessive strain on one of the links. In this Journal, simulated the implementation of load balancing on the SDN. The tools that will be used is Mininet as data plane, Floodlight as control plane along with iperf and D-ITG as tools for nalysis. The topology used is the Abilene topology with the data flowed for testing in the form of data and video streaming packeges.

Keywords: Load Balancing, SDN, Floodlight

1. Pendahuluan

Semakin hari penggunaan terhadap layanan internet semakin meningkat. Dalam satu waktu yang bersamaan memungkinkan banyak user sedang menggunakan layanan yang sama, hal tersebut menyebabkan beban pada jaringan meningkat, lebih parahnya lagi sebagian user tidak bisa mengakses layanan tersebut karena jaringan sibuk. Oleh karena itu, diperlukan suatu cara yang mampu menangani beban pada jaringan, salah satunya dengan *load balancing* (penyeimbang beban). Penyeimbang beban digunakan untuk mendistribusikan beban kerja di berbagai jalur untuk meningkatkan kehandalan jaringan dan mengoptimalkan penggunaan link [1].

Saat ini, paradigma jaringan yang sedang berkembang yaitu *Software Defined Network* (SDN). Jaringan SDN berbeda dengan jaringan tradisional, dimana pada konsep jaringan tradisional, *control plane* dan *data plane* berada pada satu perangkat yang sama sedangkan konsep dasar dari SDN yaitu memisahkan antara *control plane* dan *data plane* untuk membuatnya lebih fleksibel dan lebih mudah pengoptimalannya. Penerapan penyeimbang beban pada jaringan SDN melakukan pendekatan dimana pada setiap *switch* yang ada pada jaringan membagi trafik sesuai dengan aturan penanganan paket yang diberikan kontroler. Metode penyeimbang beban yang umum digunakan yaitu statis dan dinamis. Metode statis antara lain *round robin* dan *ratio*, sedangkan metode dinamis antara lain *least connection*, dan *fastest* [2]. Pada penelitian [3] diterapkan *least connection* pada topologi fat tree dengan parameter yang diuji yaitu *throughput* dan *latency* dengan data yang dialirkan berupa paket data.

Journal ini menggunakan topologi *backbone* yang umum digunakan yaitu topologi *Abilene*, dengan data yang dialirkan berupa paket data dan paket video *streaming*. Analisis performansi penyeimbang beban dilihat dari pemerataan beban pada setiap link, skenario waktu pemulihan link ketika ada link yang putus dan parameter *Quality of Service* (QoS) termasuk didalamnya *delay, jitter, throughput,* dan *packet loss ratio.*

2. Konsep Dasar

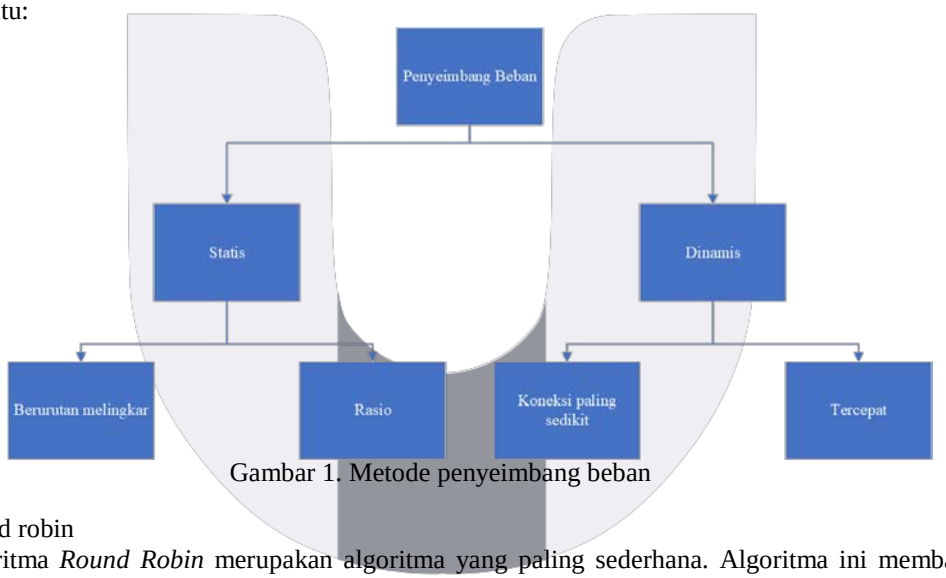
Pada bab ini menjelaskan teori dan konsep dasar yang berkaitan dengan simulasi dan perhitungan performansi dari penyeimbang beban pada jaringan SDN, yang antara lain konsep penyeimbang beban, arsitektur dan karakteristik dari SDN, protokol *OpenFlow*, kontroler *Floodlight*, emulator *Mininet*, dan QoS.

2.1 Penyeimbang Beban

Penyeimbang beban adalah metode pendistribusian beban diantara sejumlah *node* untuk mengoptimalkan pemanfaatan kemampuan komputasi dari setiap *node* dan mengurangi waktu respon rata-rata, ini setara dengan memaksimalkan *throughput* sistem [4]. Penyeimbang beban dapat digunakan untuk mengatur beban kerja pada *data center* dan jaringan *backbone*. Jalur protokol tradisional yang meneruskan trafik lalu lintas biasanya berdasarkan jalur terpendek, yang mungkin menyebabkan kemacetan yang disebabkan oleh link yang kelebihan beban. Penyeimbang beban merupakan solusi yang bagus untuk mengatasi kelemahan ini. Oleh karena itu, sangat penting untuk menerapkan penyeimbang beban di *data center* dan jaringan *backbone* yang memiliki lalu lintas dalam jumlah besar [1]. *Backbone* merupakan jalur transmisi yang lebih besar yang membawa kumpulan data dari jalur yang lebih kecil yang terkoneksi dengannya. Dalam tingkat lokal, *backbone* merupakan jalur atau kumpulan jalur dari *LAN* yang terhubung ke *WAN* atau antara *LAN* ke suatu tempat yang lain (contoh antar bangunan). Dalam tingkat internet atau *WAN*, *backbone* merupakan kumpulan jalur yang jaringan lokal atau regional terhubung ke jaringan interkoneksi jarak jauh, titik koneksi dikenal sebagai *node* jaringan atau *Telecommunication Data Switching Exchanges (DSEs). Data center* merupakan fasilitas yang terdiri dari jaringan komputer dan penyimpanan yang digunakan oleh organisasi untuk mengatur, memproses, menyimpan, dan menyebarluaskan data dalam jumlah besar.

2.2 Metode Penyeimbang Beban

Terdapat beberapa metode penyeimbang beban yang umum digunakan, seperti yang ditunjukkan pada gambar 2.1, yaitu:



Gambar 1. Metode penyeimbang beban

a. Round robin
   Algoritma *Round Robin* merupakan algoritma yang paling sederhana. Algoritma ini membagi beban secara bergiliran dan berurutan sehingga membentuk putaran.
b. Ratio
   Algoritma rasio merupakan sebuah parameter yang diberikan untuk masing-masing *node* yang akan di masukkan ke dalam sistem penyeimbang beban. Dari parameter rasio ini, akan dilakukan pembagian beban terhadap *node* yang diberikan rasio. *Node* dengan rasio terbesar diberikan beban besar, begitu pula sebaliknya.
c. Least Connection
   Algoritma ini mengarahkan penerimaan *request* dari jaringan kepada *node* dengan nomer koneksi aktif paling sedikit, algoritma ini sangat bagus untuk kelancaran distribusi ketika permintaan beban sangat banyak. Setiap satuan waktu tertentu algoritma ini akan mengecek jumlah koneksi yang sedang aktif pada masing-masing *node*. *Node* yang mempunyai jumlah koneksi yang paling sedikit yang akan menjadi urutan pertama.
d. Fastest

Algoritma ini melakukan pembagian beban dengan mengutamakan *node* yang memiliki respon paling cepat. *N*ode di dalam jaringan yang memiliki respon paling cepat merupakan *node* yang akan mengambil beban pada saat permintaan masuk.

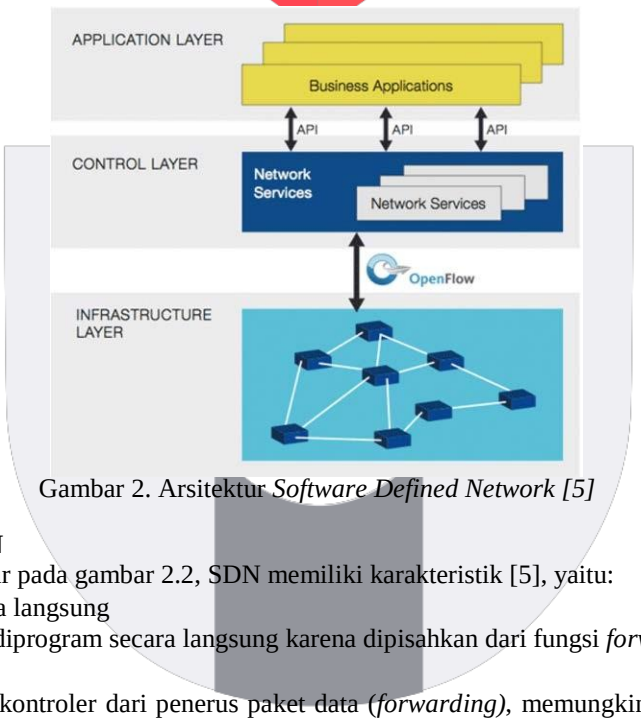
2.2 *Software Defined Network*

Software Defined Network (SDN) merupakan sebuah arsitektur baru yang dinamis, mudah diatur, biaya yang efektif, dan mudah diadaptasi sehingga ideal untuk *bandwidth* yang tinggi [5]. Dalam jaringan konvensional, *control plane* dan *data plane* berada pada satu perangkat jaringan yang sama sedangkan pada SDN konsep itu berubah dimana *control plane* dan *data plane* berada pada perangkat yang berbeda. Pemisahan ini memungkinkan untuk menerapkan *control plane* pada entitas eksternal, yang disebut kontroler [6], dan *data plane* yang termasuk pada *forwarding* paket disebut *switch* [7]. Dengan pemisahan kontrol jaringan dan *forwarding plane*, menjadikan SDN dapat diprogram sesuai keinginan.

2.2.1 Arsitektur SDN

Arsitektur SDN terdiri dari tiga buah lapisan yang dapat diakses melalui *Application Programmable Interface (API)* [8], seperti yang ditunjukkan pada gambar 2, yaitu:

a. Lapisan Aplikasi
   Terdiri dari *Business Applications end-user* untuk layanan komunikasi SDN. Batas antara lapisan aplikasi dan lapisan kontrol adalah *northbound* API.
b. Lapisan Kontrol
   Menyediakan fungsi kontrol terhadap perilaku *forwarding* di dalam jaringan melalui sebuah antarmuka terbuka.
c. Lapisan Infrastruktur
   Terdiri dari elemen dan perangkat jaringan yang dapat menyediakan *switching* paket dan *forwarding*.



Gambar 2. Arsitektur *Software Defined Network [5]*

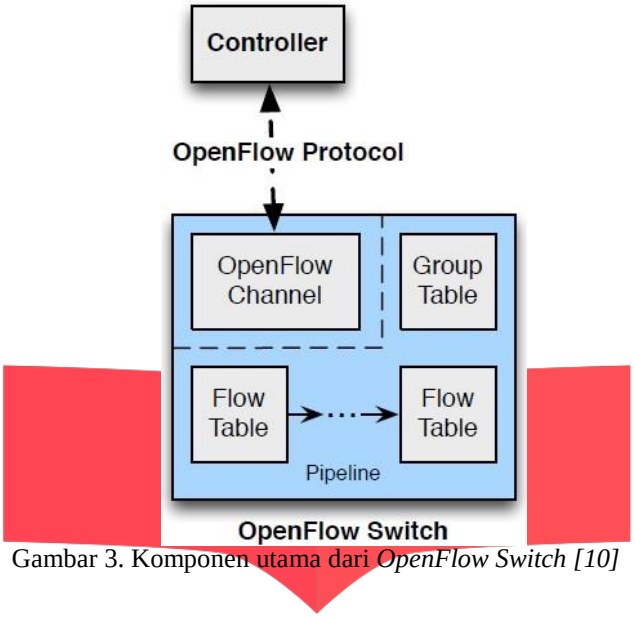
2.2.2 Karakteristik SDN

Berdasar arsitektur pada gambar 2.2, SDN memiliki karakteristik [5], yaitu:

a. Diprogram secara langsung
   Kontroler dapat diprogram secara langsung karena dipisahkan dari fungsi *forwarding*.
b. Lincah
   Pengabstraksian kontroler dari penerus paket data (*forwarding*), memungkinkan admin jaringan lebih dinamis dalam penyesuaian arus lalu lintas pada jaringan yang luas untuk memenuhi perubahan kebutuhan.
c. Dikelola secara terpusat
   Mengontrol jaringan terpusat pada perangkat lunak berbasis kontroler SDN yang memungkinkan mengatur seluruh jaringan.
d. Dikelola secara terprogram
   SDN menjadikan pengelola jaringan untuk mengkonfigurasi, mengelola, mengamankan, dan mengoptimalkan sumber daya jaringan dengan sangat cepat melalui program SDN yang dinamis.
e. Berbasis standar terbuka dan vendor yang netral
   SDN mengimplementasikan standar terbuka, sehingga SDN menyederhanakan desain jaringan dan pengoprasiannya karena instruksi disediakan oleh kontroler SDN bukan dari perangkat vendor atau protokol tertentu.

2.3 Protokol *OpenFlow*

OpenFlow adalah standar pertama antarmuka komunikasi yang ditetapkan antara lapisan kontrol dan lapisan *forwarding* pada arsitektur SDN. *OpenFlow* memungkinkan akses langsung atau manipulasi dari lapisan *forwarding* berupa perangkat jaringan seperti *switch* dan *router*, baik fisik dan virtual (*hypervisor-based*) [9]. Komponen utama dari *OpenFlow* ditunjukkan pada gambar 3.



Gambar 3. Komponen utama dari *OpenFlow Switch [10]*

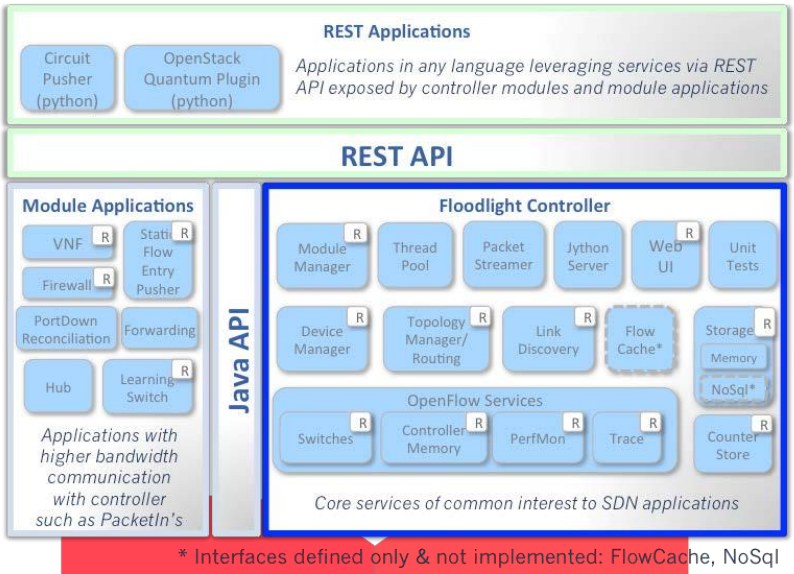
Dengan protokol *OpenFlow*, kontroler dapat menambahkan, memperbarui dan menghapus *flow-entry* yang berada pada *flow table*. Setiap *flow table* yang berada di *switch* berisi sekumpulan dari *flow-entry*, yang terdiri dari:

a. *Match fields*
   Terdiri dari *ingress port* dan *header* dari paket
b. *Priority*
   Pencocokan paket berdasarkan nilai yang lebih tinggi
c. *Counters*
   Untuk memperbarui pencocokan paket
d. *Instructions*
   Untuk memodifikasi *actions set*
e. *Timeouts*
   Lamanya waktu atau *idle time* untuk *flow*
f. *Cookie*
   Nilai data yang dipilih oleh kontroler, yang akan dipakai kontroler untuk menyaring *statistic flow*, modifikasi *flow*, dan penghapusan *flow*.

2.4 Kontroler

Kontroler merupakan aplikasi dalam SDN yang mengelola kontrol aliran untuk memungkinkan jaringan pintar. Kontroler SDN didasarkan pada protokol, seperti *OpenFlow*, yang memungkinkan server memberi tahu *switch* tempat mengirim paket [11]. Contoh kontroler yang terdapat pada SDN yaitu *Floodlight*, *Ryu*, *OpenDayLight*, NOX/POX, ONOS, *Beacon*, *Open vSwitch*, *OpenContrail*, dan lain-lain.

Floodlight merupakan kontroler *OpenFlow* dan sebuah rangkain dari aplikasi-aplikasi yang dibangun diatas kontroler *Foodlight*. Kontroler *Floodlight* mengerjakan satu set fungsi umum untuk mengontrol dan menanyakan jaringan *OpenFlow*, sedangkan aplikasi diatasnya mengerjakan fitur yang berbeda untuk memecahkan kebutuhan pengguna yang berbeda melalui jaringan [12]. Arsitektur dari kontroler *Floodlight* ditunjukkan pada gambar 4.



Gambar 4. Arsitektur *Floodlight [12]*

2.5 Emulator

Emulator merupakan perangkat keras atau perangkat lunak yang memungkinkan suatu sistem komputer (disebut *host*) untuk berjalan seperti sistem komputer lain (disebut *guest*). Contoh emulator untuk SDN antara lain *Mininet*, dan *OFNet*.

Mininet merupakan emulator jaringan yang dapat membuat jaringan virtual host, *switch*, kontroler, dan link. *Host Mininet* menjalankan perangkat lunak standar *linux*, dan *switch* yang telah mendukung *OpenFlow* dan SDN. *Mininet* bersifat *open source*, sehingga proyek yang telah dilakukan berupa *source code*, *scripts*, dan dokumentasi yang dapat dikembangkan oleh siapapun [13].

2.6 *Quality of Service*

Quality of Service (QoS) merupakan parameter-parameter minimal yang harus dimiliki oleh suatu layanan. Standarisasi *QoS* ditunjukkan pada table 2.2. Parameter *QoS* antara lain meliputi:

Tabel 1. Referensi standarisasi *QoS* ITU.T G.1010 *[14]*

Parameter performansi	Data	Video
Delay	< 15s *preffered* < 60s *acceptable* *With amount of data 10KB - 10MB*	< 10s *With amount of data 16-384 kbit/s*
Jitter	NA	< 1 ms
Throughput	NA	NA
Packet loss	0%	< 1%

2.6.1 *Delay*

Delay merupakan perbedaan waktu di antara waktu kirim dengan waktu terima sebuah paket. One-*way delay* atau *latency* merupakan waktu rata-rata pengiriman setiap paket dalam perjalanan dari satu titik kirim ke satu titik terima [15].

$$\quad (2.1)$$

Keterangan:
= *delay* paket ke-i
= waktu paket ke-i dikirimkan
= waktu paket ke-i diterima

2.6.2 *Jitter*

Jitter merupakan rata-rata deviasi dari selisih *delay* antar paket yang dikirimkan. *Jitter* disebut juga variasi *delay* dalam jaringan [15].

$$\quad (2.2)$$

Keterangan:

= *delay* paket ke-i dan paket ke-(i-1)
= *Jitter* rata-rata

2.6.3 *Throughput*

Throughput merupakan 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 [15].

(2.3)

Keterangan:

= Jumlah *bit* yang sukses diterima
= total waktu pengiriman

2.6.4 *Packet Loss Ratio*

Packet Loss Ratio (PLR) merupakan perbandingan antara jumlah paket sukses yang diterima oleh *user* dengan jumlah paket yang dikirimkan. *PLR* dinyatakan dalam bentuk persen (%) [15].

$$= \frac{-}{} \times 100\% \quad (2.4)$$

Keterangan:
= *Packet Loss Ratio*
= jumlah paket yang dikirimkan

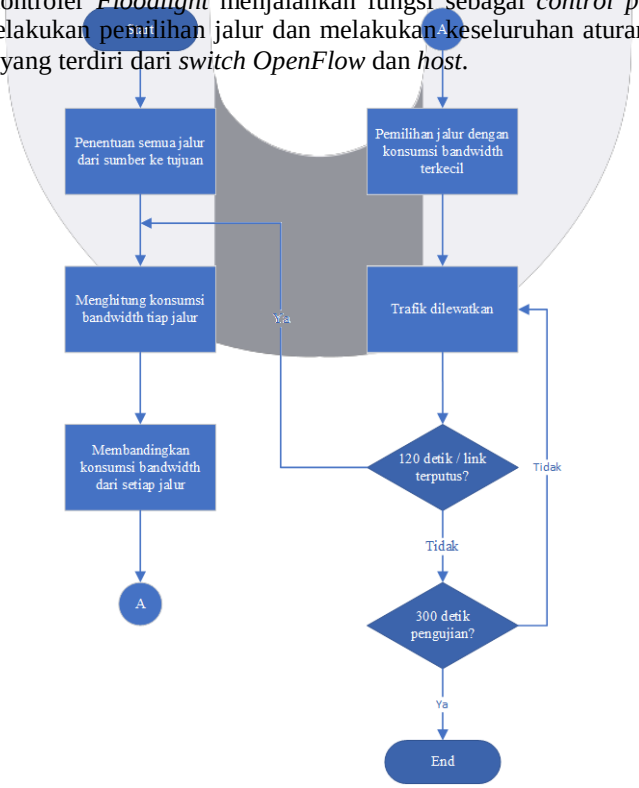
= jumlah paket yang sukses diterima

3. Perancangan Konfigurasi Simulasi

Pada bab ini menjelaskan perancangan sistem dalam simulasi performansi penyeimbang beban pada jaringan SDN, yang antara lain pemodelan sistem, perancangan simulasi termasuk didalamnya perancangan topologi, perangkat simulasi, dan skenario pengujian.

3.1 Pemodelan Sistem

Pemodelan sistem pada gambar 5 mensimulasikan suatu jaringan berbasis SDN yang terdiri dari kontroler dimana dalam simulasi ini menggunakan kontroler *Floodlight* dan *data plane* dimana dalam simulasi ini dijalankan pada emulator *Mininet*. Kontroler *Floodlight* menjalankan fungsi sebagai *control plane* yang bertugas untuk mengontrol aliran data, melakukan pemilihan jalur dan melakukan keseluruhan aturan dari SDN. Pada *Mininet* dirancang sebuah topologi yang terdiri dari *switch OpenFlow* dan *host*.



Gambar 5. Pemodelan Sistem

3.2 Perancangan Topologi

Pada simulasi, akan dilakukan penyeimbang beban dengan menggunakan topologi *Abilene* dengan 12 *switch* dan 12 *host* seperti yang ditunjukkan pada gambar 6. Topologi pada simulasi ini dibuat pada emulator *Mininet*. Pada *Mininet* di buat arsip dalam bahasa pemrograman *python* yang didalamnya didefinisikan beberapa hal yang sesuai dengan topologi yang digunakan, seperti jumlah *switch, host,* link antara *switch* dengan *switch,* link antara *switch* dengan *host,* alamat *ip host*, dan nilai *bandwidth* dan *delay* disetiap *link* antara *switch* dengan *switch*.

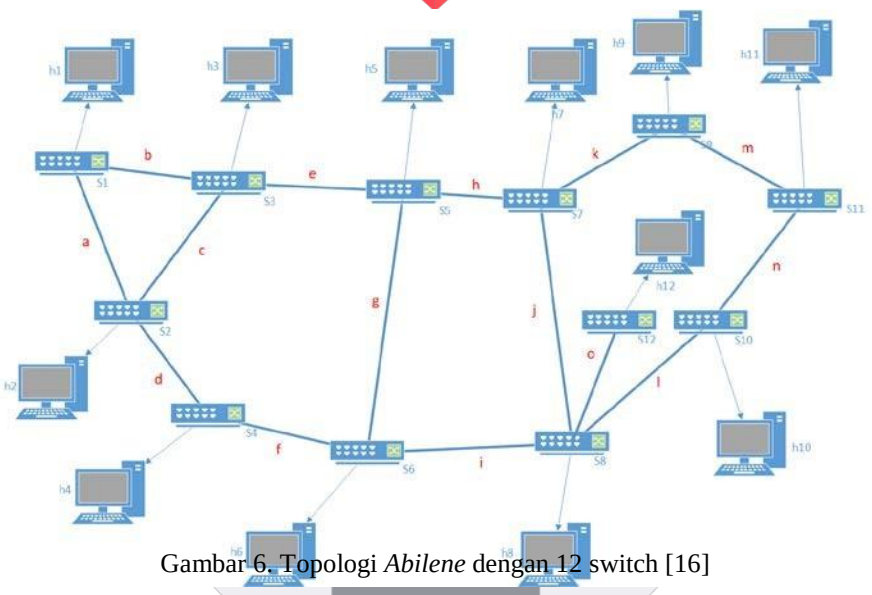
Pada simulasi dirancang topologi seperti pada gambar 6. dengan IP *host* 10.0.0.x dengan x adalah nomor urut dari *host.* Adapun spesifikasi topologi yang akan dirancang ditunjukkan pada tabel 2 sebagai berikut:

Tabel 2. Spesifikasi topologi

Jumlah *switch*	Jumlah *host*	Jumlah link antar *switch*	*Bandwith* link
12	12	1	100

Tabel 3. *Background Traffic* tiap link

Background Traffic	Link
75 Mbps	a, b, d, e, f, g, h, i, j, o
25 Mbps	c, l, k, m, n



Gambar 6. Topologi *Abilene* dengan 12 switch [16]

3.3 Skenario Pengujian

Untuk mengetahui kinerja dari simulasi dirancang, maka dilakukan dengan pengujian sebagai berikut:

a. Pemerataan Beban
   Pengujian pemerataan beban dilakukan untuk mengetahui kondisi setiap link dari penggunaan penyeimbang beban. Pengukuran dilakukan saat dikirimnya paket hingga pengujian berakhir. Nilai yang dihitung merupakan nilai konsumsi *bandwidth* pada setiap link yang didapat dari hasil pengolahan data selama pengujian yang dilakukan setiap jeda 0.5 detik.

b. *Quality of Service*
   Pengujian QoS dilakukan untuk mengetahui pengaruh jumlah *switch* terhadap kinerja penyeimbang beban. Parameter yang diuji dalam QoS pada simulasi ini antara lain *delay, throughput,* dan *packet loss*. Dalam Jurnal ini, *delay* yang dimaksud adalah *one-way delay* atau *latency*, dan *packet loss* yang dimaksud adalah *packet loss* dalam satu kali perjalanan paket (*one-way path*).

c. Waktu Pemulihan Link

Pengujian waktu pemulihan link dilakukan untuk mengetahui waktu yang dibutuhkan dari terputusnya link sampai dengan mendapatkan link baru. Pemutusan link pada pengujian dilakukan pada detik ke 180, link yang diputus merupakan link yang sedang digunakan/dilewatkan paket.

Percobaan dilakukan sebanyak 10 kali selama 300 detik dengan membandingkan jaringan tanpa penyeimbang beban dengan jaringan yang menggunakan penyeimbang beban, selain itu dilakukan dengan 2 skenario pengujian, yaitu skenario tanpa pemutusan link dan skenario dengan pemutusan link. Dalam pengujian ini dibangkitkan dua jenis trafik [17], yaitu:

1. Trafik Data UDP dengan *inter-departure time (IDT)* konstan sebesar 280 pps dengan ukuruan paket 8192 bytes. Sehingga membutuhkan *bandwidth* sekitar 17,5 Mbit/s.
2. *Video streaming* HDR dengan resolusi 1080p [18] memiliki 60 *frame* (satu paket per *frame*) per detik, ukuran paket terdistribusi normal dengan ukuran paket rata-rata 33792 bytes dan standar deviasi 6114 bytes, sehingga membutuhkan *bandwidth* sekitar 15,47 Mbit/s.

Selain pemberian tiga jenis trafik tersebut, dibangkitkan juga *background traffic* yang bervariasi (25 Mbps dan 75 Mbps) dengan bantuan *iperf*. Percobaan ini dilakukan untuk mengetahui kinerja penyeimbang beban terutama pada pemilihan jalur dari trafik yang padat.

4. Analisis dan Hasil

Pada bab ini dijelaskan mengenai hasil pengujian yang telah dilakukan. Terdapat tiga parameter yang diukur, yaitu konsumsi *bandwidth* tiap link, *Quality of Service*, dan waktu pemulihan link.

4.1 Konsumsi *bandwidth tiap link*

Gambar 7. Pemerataan beban pengujian paket data

Pada gambar 7 menunjukkan perbandingan tiap link pada pengujian paket data dengan penyeimbang beban dengan tanpa penyeimbang beban. Pada link c, k, l, m, n, dan o memiliki nilai yang hampir sama, karena pada link o baik dengan dan tanpa penyeimbang beban keduanya melewati link tersebut untuk sampai ke host tujuan, sedangkan pada link c, k, l, m, n merupakan link yang tidak dilewati keduanya sehingga hanya *background* trafik saja yang terdapat pada link tersebut. Pada link a, b, d, e, f, g, h, i, j, o merupakan link yang dilewati pada pengujian dengan penyeimbang beban secara bergantian, sedangkan link a, d, f, i, o merupakan link yang dilewati oleh pengujian tanpa penyeimbang beban. Link b, e, g, dan i pada 120 detik pertama merupakan link yang paling sering dilewati pada pengujian dengan penyeimbang beban, sedangkan link a, d, dan f pada 120 detik berikutnya merupakan link yang paling sering dilewati, link b, e, h, dan j pada detik 240 hingga berakhirnya pengujian yaitu pada detik 300 merupakan link yang paling sering dilewati pada pengujian dengan penyeimbang beban. Pada pengujian dengan penyeimbang beban, trafik dilewatkan ke beberapa jalur secara bergantian setiap 120 detik, sehingga pada satu

jalur tidak terbebani oleh trafik terus menerus, dan beban dapat disebar ke link yang lain sehingga pemerataan beban akan lebih baik dibandingkan pengujian tanpa penyeimbang beban.

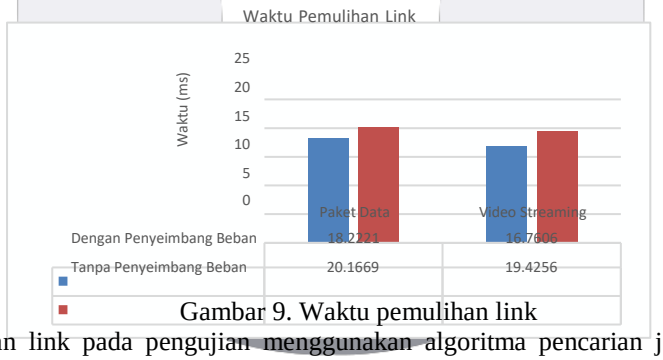
4.2 *Quality of Service*

Pada bagian ini ditunjukkan hasil pengukuran dan analisis hasil pengujian yang telah dilakukan. Parameter yang dianalisis untuk melihat performansi jaringan ini yaitu *delay*, *jitter*, *throughput*, dan *packet loss* dari kedua jenis layanan yaitu data, dan video *streaming*. Tanda (-) merupakan pengujian dengan pemutusan link.

Gambar 8. *Delay*, *jitter*, *throughput*, dan *packet loss*

Dalam grafik tersebut menunjukkan bahwa jaringan dengan penyeimbang beban lebih baik daripada jaringan tanpa penyeimbang beban. Jaringan dengan penyeimbang beban memiliki nilai rata-rata *delay* yang lebih baik, dan memiliki rata-rata *packet loss* yang lebih kecil pada skenario pemutusan link pada paket data.

4.3 Waktu Pemulihan link



Gambar 9. Waktu pemulihan link

Waktu pemulihan link pada pengujian menggunakan algoritma pencarian jalur yang sama, namun pada jaringan dengan penyeimbang beban memiliki nalai rata-rata waktu pemulihan link yang lebih baik dibanding tanpa penyeimbang beban.

5. Kesimpulan

Jurnal ini melakukan analisis simulasi terhadap performansi perbandingan penyeimbang beban dengan tanpa penyeimbang beban pada topologi *Abilene* 12 *switch*. Hasil simulasi dalam Jurnal ini menunjukkan bahwa jaringan dengan penyeimbang beban memiliki nilai rata-rata *delay* 26,55% lebih baik dibanding tanpa penyeimbang beban pada skenario tanpa pemutusan link pada pengujian paket data, pada skenario dengan pemutusan link, jaringan dengan penyeimbang beban memiliki nilai rata-rata *delay* 40,79%, begitu pula dengan pengujian video *streaming* 9,97% lebih baik, dan 39,91% lebih baik pada skenario pemutusan link. Nilai rata-rata *jitter* dan *throughput* tidak jauh berbeda karena data yang dialirkan sama. Nilai rata-rata *packet loss* pada pengujian paket data dengan skenario pemutusan link memiliki nilai yang lebih baik. Hasil dari jurnail ini juga diharapkan dapat dijadikan referensi dalam skenario maupun pemodelan sistem yang lain.

Daftar Pustaka:

[1] X. C. Y. R. Z. Z. G. W. Jun Li, “An Effective Path Load Balancing Mechanism Based on SDN,” *IEEE*, pp. 527-533, 2014.

[2] M. E. Mustafa, "Load Balancing Algorithms Round-Robin(RR), Least-Connection, and Least Loaded Efficiency," vol. 1, pp. 25-29, 2017.

[3] nayanseth, April 2016. [Online]. Available: https://github.com/nayanseth/sdn-loadbalancing. [Accessed May 2016].

[4] D. M. E. Mustafa, "Load Balancing Algorithms Round-Robin (RR), Least Connection, and Least Loaded Efficiency," 2017.

[5] ONF, “Open Networking Foundation,” [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition. [Diakses 19 November 2015].

[6] R. B. E. M. A. P. Paulo Fonseca, "A replication component for resilient OpenFlow-based networking," *IEEE*, 2012.

[7] R. P. R. K. M. Veen [illegible] using Software Defiened Networks," *IEEE*, 201[illegible]

[8] Open Networking Fou[illegible]Networks, 2012.

[9] ONF, “Open Netw[illegible].opennetworking.org/sdn-resources/openflow. [Diakses 19 Novem[illegible]

[10] Open Network Foundation, OpenFlow Switch Sp[illegible]cation Version 1.3.4 (Protocol version 0x04), 2014.

[11] M.Rouse,"techtarget,"November2012.[Online].Available: https://searchsdn.techtarget.com/definition/SDN-controller-software-defined-networking-controller. [Accessed 23 October 2018].

[12] kc.wang@bigswitch.c[illegible]"Floodlight "23October2012.[Online].Available: https://floodlight.atlas[illegible]ay/floodlightcontroller[illegible]cessed 4 November 2016].

[13] “Mininet,” [Online]. A[illegible]ninet.org/overview/. [illegible]ber 2015].

[14] N. Neal Seltz, “ITU-T[illegible] IP-Based Networks.[illegible]ations Magazine, 2003.

[15] W. d. D. A. D. S. A. A[illegible] "D-ITG 2.8.1 Manu[illegible]Mputer for Interaction and CommunicationS) Gro[illegible]

[16] A. B. S. I. J. M. E. I[illegible]vati, "Missing Interne[illegible]uction using Compressive Sampling," *Internatio[illegible]mmunication Networ[illegible] Security (IJCNIS)*, vol. 9, 2017.

[17] D. E. A. P. G. V. Stefa[illegible]actical Demonst[illegible]raffic Generator," in *IMSA*, Kauai (USA), 2004.

[18] Google,"https://support.g[illegible] https://support.google.com/[illegible]/ 2019].