Journal of Data Analysis
Vol.3, No.1, June 2020, p. 36-49

Analisa *Quality of Service* pada Jaringan *Wireless* Berbasis *Software-Defined Network* dengan Protokol *Openflow* Menggunakan Floodlight Controller

Muhd. Iqbal[1*], Zulfan[2], Muhammad Arif Ramadhan[3]

[123]Departement of Infomatics, Universitas Syiah Kuala, Indonesia
E-mail: iqbal@unsyiah.ac.id[*]

Abstrak

Jaringan komputer merupakan sistem yang terdiri dari dua atau lebih perangkat komputer yang saling terhubung. Semakin banyak perangkat jaringan yang digunakan maka semakin lama waktu yang diperlukan untuk mengkonfigurasikan jaringan. Metode *Software-Defined Network* (SDN) dapat menjadi solusi untuk mengatasi hal tersebut. SDN dapat mengendalikan dan mengkonfigurasikan perangkat jaringan secara bersamaan dalam jumlah banyak. Untuk dapat menjalankan SDN dibutuhkan *controller* untuk mengatur jaringan. Pada penelitian ini, *controller* yang akan digunakan adalah Floodlight Controller dan untuk mengetahui kemampuan jaringan, *Quality of Service* (QoS) dari SDN akan diuji. Pertama, jaringan pada perangkat lunak Mininet akan dirancang terlebih dahulu. Selanjutnya dilakukan pengujian QoS dengan dan tanpa menggunakan metode *Hierarchical Token Bucket* (HTB). Kemudian, hasil dari pengujian akan dibandingkan dan dinilai menggunakan standar TIPHON. Dari hasil pengujian, dengan menggunakan metode HTB, jaringan yang dihasilkan lebih bagus dibandingkan tanpa metode HTB.

Informasi Artikel

Sejarah Artikel:
Dajukan 4 Okt 2019
Diterima 9 Agt 2020

Kata Kunci:
Jaringan computer
S*oftware-defined network*
Quality of Service
Hierarchical Token Bucket

Abstract

Computer network is a system consisting of two or more interconnected computer devices. The more network devices used, the longer it takes to configure the network. The Software-Defined Network (SDN) method can be a solution to overcome this. SDN can control and configure network devices simultaneously in large numbers. SDN can be run physically and virtually on Mininet software. To be able to run SDN, a controller is needed to run the network. In this study, the controller that will be used is the Floodlight Controller and to find out the network capabilities, Quality of Service (QoS) from the SDN will be tested. First, the network on the Mininet software was designed first. Then QoS testing is done with and without using the Hierarchical Token Bucket (HTB) method. Then, the results of the tests will be compared and assessed using the TIPHON standard. From the test results, using the HTB method, the resulting network is better than without the HTB method.

Keywords:
Computer Network
Software-defined Network
Quality of Service
Hierarchical Token Bucket

1. Pendahuluan

Jaringan komputer adalah sebuah sistem yang terdiri dari dua atau lebih perangkat komputer yang terhubung satu sama lain melalui media transmisi sehingga dapat saling berbagi informasi dan komunikasi satu sama lain dengan mudah. Seiring perkembangan waktu, perkembangan dari jaringan komputer sudah semakin meningkat sehingga menyebabkan jumlah perangkat yang terhubung dengan internet semakin pesat yang menyebabkan meningkatnya penggunaan perangkat jaringan. Dengan semakin banyak nya perangkat yang digunakan, maka semakin banyak juga perangkat yang harus dikonfigurasikan. Untuk memudahkan hal tersebut, metode *Software-Defined Network* (SDN) dapat digunakan.

SDN merupakan sebuah konsep jaringan baru yang dapat mengelola jaringan secara terpusat sehingga jaringan menjadi lebih cepat dan mudah untuk dikonfigurasikan. Perbedaan mendasar pada jaringan SDN dengan jaringan tradisional adalah penempatan fungsi *control* dan *forward*. Pada jaringan tradisional fungsi control dan forward ditempatkan pada device yang sama seperti *router*. Sedangkan, pada jaringan SDN fungsi *control* ditempatkan pada *software* terpusat (*controller*) dan fungsi *forward* ditempatkan pada suatu perangkat kosong berupa *switch* (*forwarding decive*). Sehingga, dengan menggunakan metode SDN dapat memudahkan kerja kita saat mengkonfigurasikan jaringan.

Untuk saat ini, jaringan SDN sudah dapat dijalankan secara fisik maupun virtual. Pada mesin virtual, SDN memiliki simulator khusus yang dibuat agar dapat menjalankan jaringan SDN itu sendiri. Simulator tersebut adalah Mininet.

Mininet adalah simulator yang digunakan untuk mensimulasikan kinerja dari SDN. Mininet dapat membangun jaringan berdasarkan kebutuhan yang diinginkan dari SDN seperti jaringan realistis, dan jaringan yang dibangun dengan Mininet dapat dijalankan atau diperintahkan dengan Floodlight Controller.

Floodlight Controller sendiri bekerja sebagai pengontrol jaringan agar jaringan tersebut dapat digunakan pada saat jaringan simulasikan. Terdapat banyak fitur yang didukung oleh Floodlight Controller salah satunya adalah *Quality of Service* (QoS). Pengujian dari QoS dapat dilakukan dengan Floodlight Controller pada simulator Mininet sehingga jaringan SDN dapat diukur kemampuannya. Adapun yang akan diukur pada pengujian QoS adalah *Packet Loss*, *Jitter*, dan *Throughput* dengan menggunakan menggunakan metode *Hierarchical Token Bucket* dan menggunakan standar ukur dari TIPHON.

2. Tinjauan Pustaka

2.1. Jaringan Komputer

Jaringan komputer adalah sebuah sistem yang didalamnya terdapat sekumpulan perangkat komputer yang saling terhubung yang bertujuan agar perangkat-perangkat tersebut saling terhubung. Perangkat-perangkat terkait dihubungkan agar dapat berbagi atau mengakses data-data atau informasi-informasi antar perangkat. Perangkat-perangkat tersebut dihubungkan dengan media transmisi seperti kabel dan perangkat keras lainnya seperti switch, router, dll. Dengan adanya jaringan komputer, pengguna dapat mengirimkan data-data ataupun informasi dengan mudah dan cepat. [1][2]

2.2. Software-defined Network (SDN)

SDN adalah suatu konsep baru yang memberikan kemudahan untuk mengelola perangkat *router* dan *switch* secara fleksibel. SDN dapat memberikan pengelolaan jaringan secara terpusat sehingga layanan menjadi lebih mudah dan cepat. SDN memisahkan fungsi *control* dan fungsi

forwarding, sehingga perangkat lebih mudah di kontrol. SDN memiliki 3 layer, yaitu *Application layer* yang berisi *layer* yang menyediakan *interface* antara aplikasi dan jaringan. Kemudian ada *Control layer*, yaitu *layer* yang berperan untuk mengendalikan sistem, pada layer ini fungsi *data plane* (*forwarding*) dan *control plane* (*controller*) dipisah. *Control plane* akan mengendalikan sistem dan *data plane* adalah sekumpulan perangkat yang dikendalikan oleh controller. Yang terakhir ada *Infrastructure layer* yaitu perangkat-perangkat networking yang berjalan sesuai keinginan dari controller. [3][4][5]

2.3. Openflow

Openflow adalah protokol utama dari jaringan SDN. *Openflow* menjadi jembatan yang menghubungkan antara *controller* yang secara fisik sudah di pisahkan dengan *data plane*. *Openflow* mengontrol switch agar bisa mengirimkan paket ketika melalui sebuah *switch*. Dengan menggunakan protokol *Openflow*, memungkinkan operator mengatasi *bandwidh* yang tinggi dan mengurangi kompleksitas manajemen jaringan. [6]

2.4. Floodlight Controller

Floodlight merupakan *controller Openflow* kelas *enterprise* dengan lisensi Apache dan berbasis Java. Hal ini didukung oleh komunitas pengembang termasuk sejumlah insinyur dari *Big Switch Networks*. Floodlight dirancang untuk bekerja dengan meningkatnya jumlah *switch*, *router*, *switch virtual*, dan jalur akses yang mendukung standar *Openflow*. Floodlight Controller sendiri bekerja sebagai penyedia jaringan untuk jaringan simulasi. Apabila Floodlight Controller tidak terdapat pada saat jaringan simulasi dijalankan, maka perangkat-perangkat jaringan tersebut tidak dapat dijalankan. [7]

2.5. Quality of Service (QoS)

QoS adalah sebuah metode pengukuran untuk mengetahui seberapa baik jaringan. QoS juga diciptakan untuk memberikan layanan berbeda-beda sesuai kebutuhan client. Dengan metode QoS, administrasi jaringan dapat memberikan layanan kepada *client* secara efisien sesuai kebutuhan. QoS bertujuan untuk membantu pengguna mendapatkan layanan jaringan dengan performa yang memuaskan. [8][9]

QoS didefinisikan sebagai sebuah mekanisme atau cara yang memungkinkan layanan dapat beroperasi sesuai dengan karakteristiknya masing-masing dalam jaringan *Internet Protocol* (IP). Analisis jaringan menggunakan QoS khususnya adalah *latency* dan *throughput* karena mampu memberikan analisis jaringan yang baik, dimana aspek ini yang sering digunakan didalam analisis jaringan. [10]

Tabel 1 Persentase dan nilai dari Quality of Services

Nilai	Persentase (%)	Indeks
3.8-4	95-100	Sangat Memuaskan
3-3.79	75-94.75	Memuaskan
2-2.99	50-74,75	Kurang Memuaskan
1-1.99	25-49,75	Buruk

2.5.1. Jitter

Jitter diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket. *Jitter* menunjukkan banyaknya variasi delay

pada taransmisi data di jaringan. Semakin besar nilai *jitter* akan mengakibatkan nilai QoS akan semakin turun.

Tabel 2 *Kategori dan nilai dari Jitter*

Kategori *Jitter*	*Jitter (ms)*	Indeks
Sangat Memuaskan	0	4
Memuaskan	0-75	3
Kurang Memuaskan	75-125	2
Buruk	125-225	1

Persamaan perhitungan *Jitter*:

$$Jitter = \frac{Total\ variasi\ delay}{Total\ packet\ yang\ diterima} \quad (1)$$

2.5.2. Packet Loss

Packet Loss adalah parameter dari QoS untuk mengetahui jumlah packet yang hilang. Biasanya penyebab terjadinya packet loss adalah adanya *noise* (sinyal-sinyal yang tidak diinginkan yang selalu ada dalam suatu sistem transmisi), *collision* (hilangnya data yang terjadi karena dua buah *device* mengirim data pada saat yang bersamaan) dan *congestion* (tabrakan yang akan menyebabkan terlambatnya data sampai) yang disebabkan oleh terjadinya antrian yang berlebihan dalam jaringan.

Tabel 3 Kategori dan nilai dari Packet Loss

Kategori Packet Loss	Packet Loss (%)	Indeks
Sangat Memuaskan	0	4
Memuaskan	3	3
Kurang Memuaskan	15	2
Buruk	>25	1

Persamaan perhitungan *Packet Loss*:

$$Packet\ Loss = \frac{(Packet\ data\ dikirim - Packet\ data\ diterima)\ x\ 100\%}{Packet\ data\ dikirim} \quad (2)$$

2.5.3. Throughput

Throughput merupakan kemampuan sebenarnya suatu jaringan dalam melakukan pengiriman data. Yaitu kecepatan (*rate*) *transfer* data efektif, yang diukur dalam *bps*. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama *interval* waktu tertentu dibagi oleh durasi *interval* waktu tersebut.

Tabel 4 Kategori dan nilai dari Throughput

Kategori Throughput	Throughput (%)	Indeks
Sangat Memuaskan	100	4
Memuaskan	75	3
Kurang Memuaskan	50	2
Buruk	<25	1

Persamaan perhitungan *Throughput*:

$$Throughput = \frac{\text{Paket data diterima}}{\text{Lama pengamatan}} \quad (3)$$

2.6. Denial of Service (DoS)

DoS adalah jenis serangan terhadap sebuah komputer atau server di dalam jaringan internet dengan cara menghabiskan sumber (resource) yang dimiliki oleh komputer tersebut sampai komputer tersebut tidak dapat menjalankan fungsinya dengan benar sehingga secara tidak langsung mencegah pengguna lain untuk memperoleh akses layanan dari komputer yang diserang tersebut.

Dalam sebuah serangan DoS, si penyerang akan mencoba untuk mencegah akses seorang pengguna terhadap sistem atau jaringan dengan menggunakan beberapa cara, yakni sebagai berikut:

- Membanjiri lalu lintas jaringan dengan banyak data sehingga lalu lintas jaringan yang datang dari pengguna yang terdaftar menjadi tidak dapat masuk ke dalam sistem jaringan. Teknik ini disebut sebagai traffic flooding.
- Membanjiri jaringan dengan banyak request terhadap sebuah layanan jaringan yang disediakan oleh sebuah host sehingga request yang datang dari pengguna terdaftar tidak dapat dilayani oleh layanan tersebut. Teknik ini disebut sebagai *request* flooding.
- Mengganggu komunikasi antara sebuah host dan kliennya yang terdaftar dengan menggunakan banyak cara, termasuk dengan mengubah informasi konfigurasi sistem atau bahkan perusakan fisik terhadap komponen dan server. [11]

2.7. Manajemen Bandwidth

Manajemen bandwidth merupakan teknik pengelolaan jaringan sebagai usaha untuk memberikan performa jaringan yang adil dan memuaskan. Manajemen bandwidth juga digunakan untuk memastikan bandwidth yang memadai untuk memenuhi kebutuhan traffic data dan informasi serta mencegah persaingan antara aplikasi. Manajemen bandwidth menjadi hal mutlak bagi jaringan multi layanan, semakin banyak dan bervariasinya aplikasi yang dapat dilayani oleh suatu jaringan akan berpengaruh pada penggunaan link dalam jaringan tersebut. Link-link yang ada harus mampu menangani kebutuhan pengguna akan aplikasi tesebut bahkan dalam keadaan kongesti sekalipun. [12]

Manajemen bandwidth dapat diartikan sebagai proses mengukur dan mengendalikan pertukaran informasi dalam jaringan komputer, sehingga dapat menghindari hal-hal yang tidak diinginkan yang berakibat pada network congestion dan penurunan kemampuan jaringan. Sebuah manajemen bandwidth yang baik harus dapat membuat dan menjaga aturan tentang ketersediaan koneksi (dalam hal ini internet). [13]

Untuk proses manajemen bandwidth dapat dilakukan dengan beberapa tipe queue, yaitu simple queue dan queue tree.

- *Simple Queue*

Simple Queue merupakan menu pada RouterOS untuk melakukan manajemen bandwidth untuk skenario jaringan yang sederhana. Untuk menggunakan *Simple Queue*, pekerjaan *packet classification* dan *marking packet* tidak wajib dilakukan. Meskipun demikian, *Simple Queue* sebenarnya juga bisa melakukan manajemen *bandwidth* terhadap packet-packet yang sudah di *marking*.

- *Queue Tree*

Queue Tree adalah konfigurasi *queue* yang bersifat *one way* (satu arah), ini berarti sebuah konfigurasi *queue* hanya akan mampu melakukan *queue* terhadap satu arah jenis traffic. Jika sebuah konfigurasi *queue* pada *Queue Tree* ditujukan untuk melakukan antrian terhadap *bandwidth download*, maka konfigurasi tersebut tidak akan melakukan antrian untuk *bandwidth upload*, demikian pula sebaliknya. Sehingga untuk melakukan *queue* terhadap *traffic upload* dan *download* dari sebuah komputer *client*, kita harus membuat 2 (dua) konfigurasi *queue*. Terdapat beberapa metode pada queue tree, diantaranya *Class-Based Queue* (CBQ) dan HTB. [14]

2.8. Hierarchical Token Bucket (HTB)

HTB adalah sebuah metode untuk mendistribusikan bandwidth secara fleksibel berbasis Linux. Dengan kata lain HTB digunakan untuk mengatur pembagian bandwidth. Pembagian ini dilakukan secara hirarki yang dibagi kedalam kelas-kelas untuk memudahkan pengaturan bandwidth. HTB salah satu teknik antrian yang bertujuan untuk menerapkan *link sharing*. Dalam konsep *link sharing*, jika suatu kelas meminta kurang dari jumlah service yang telah ditetapkan untuknya, sisa bandwidth akan di distribusikan ke kelas-kelas lain yang meminta service. [15]

HTB adalah metode manajemen *bandwidth* yang digunakan untuk membatasi akses menuju alamat IP tertentu tanpa menggangu traffic *bandwidth* pengguna lain. HTB merupakan teknik penjadwalan paket yang sering digunakan pada *router-router* berbasis Linux. HTB adalah salah satu teknik penjadwalan yang digunakan pada *queue tree*. [16]

2.9. Mininet

Mininet adalah sebuah emulator untuk membuat prototype jaringan berskala besar secara cepat pada sumber daya yang terbatas (seperti pada single komputer atau laptop maupun Virtual Machine). Mininet diciptakan dengan tujuan untuk mendukung riset di bidang SDN dan *Openflow*. Emulator Mininet memungkinkan kita untuk menjalankan sebuah kode secara interaktif di atas laptop atau di atas virtual hardware, tanpa harus memodifikasi kode tersebut. Artinya kode simulasi sama persis dengan kode pada real network environment.

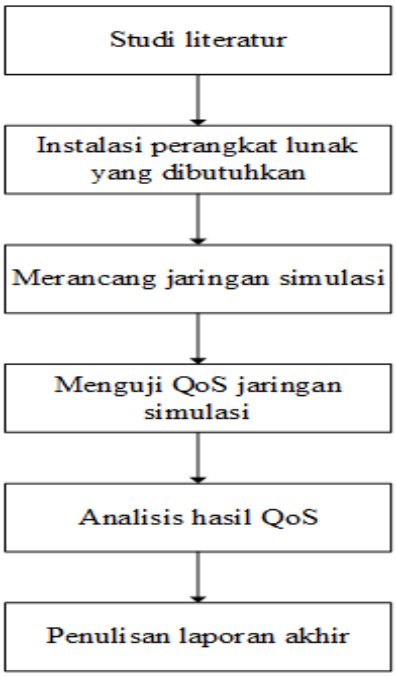
Mininet adalah solusi yang dianggap paling unggul dalam hal kemudahan penggunaan, performansi, akurasi, dan skalabilitas serta memiliki sertifikasi sebagai emulator untuk SDN dan *Openflow*. Ia mampu menyediakan lingkungan yang realistis dan nyaman (convenience) dengan harga yang murah (low cost). Kita dapat menggunakan alternatif lain seperti hardware test-bed untuk simulasi jaringan, yang mana dapat berjalan cukup kencang dan akurat, namun harganya mahal dan harus di-shared dengan pengguna lain. Begitu pula, kita dapat menggunakan simulator yang harganya murah, namun seringkali kode simulasi akan harus dimodifikasi lagi bila akan dijalankan di real network environment.

Mininet menggunakan pendekatan light-weight virtualization menggunakan fitur virtualisasi level OS mencakup proses-proses dan namespace jaringan, sehingga memungkinkan dilakukannya simulasi jaringan dengan skala sangat besar (hingga ratusan node). Mininet dapat menciptakan jaringan virtual yang realistis, menjalankan real kernel, switch dan kode aplikasi,

pada single machine (baik berupa physical machine, virtual machine, atau cloud). Mininet sangat berguna untuk pengembangan riset, pengajaran, serta penelitian. [17]

3. Metode Penelitian

Gambar 1 menunjukkan alur metode penelitian yang digunakan dalam penelitian ini.



Gambar 1 Tahapan penelitian

Terdapat beberapa tahapan dalam penelitian ini dengan penjelasan sebagai berikut.

1. Studi Literatur

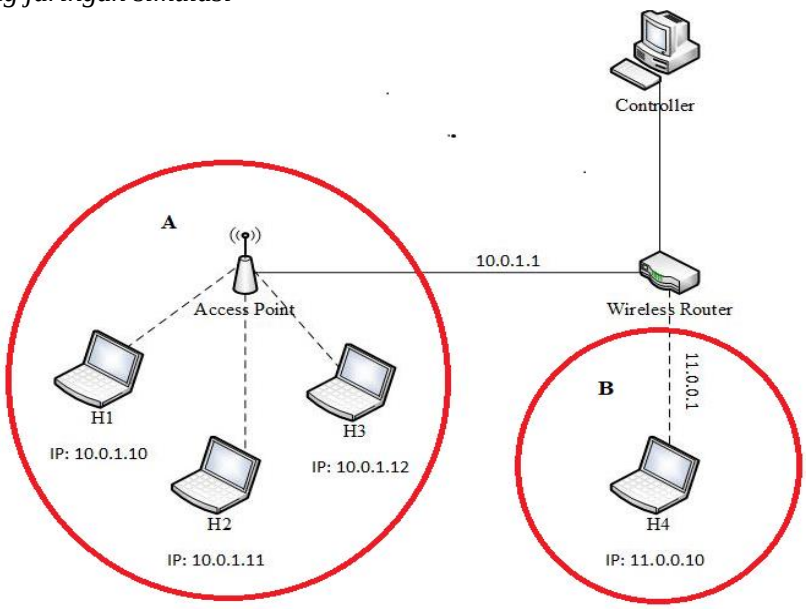
Studi literatur merupakan langkah awal untuk mempelajari dan mengkaji teori-teori yang berkaitan dengan SDN, *Openflow*, HTB, dll. Studi literatur ini dilakukan selama penelitian melalui buku, jurnal dan video dokumenter yang berkaitan dan di jadikan sebagai referensi penelitian ini.

2. Instalasi perangkat lunak yang dibutuhkan

Pada tahapan ini dilakukan instalasi perangkat lunak yang di butuhkan seperti VirtualBox, sistem operasi Linux Ubuntu, Mininet, Java SE Development Kit serta Floodlight Controller, dll.

VirtualBox digunakan untuk dapat menjalankan sistem operasi Linux Ubuntu secara virtual, yang mana Linux Ubuntu diperlukan sebagai sistem operasi yang akan digunakan untuk penelitian. Sementara itu, Mininet berguna untuk dapat membangun simulasi jaringan dari SDN. Untuk Java SE Development Kit digunakan untuk dapat menjalankan Floodlight Controller, dan Floodlight Controller berguna untuk dapat mengkontrol jaringan kepada perangkat-perangkat yang telah dibangun pada software mininet.

3. Merancang jaringan simulasi



Gambar 2 Topologi jaringan

Tahapan ini merupakan tahapan untuk merancang jaringan simulasi berdasarkan kebutuhan penelitian. Pada tahapan ini, akan dirancang sebuah jaringan dengan *bandwidth* sebesar 20Mbps. Jaringan akan dirancang dengan sebuah controller sebagai pengatur jaringan, yang dihubungkan dengan sebuah *router*, yang mana *router* juga digunakan sebagai pemisah antara jaringan A dengan jaringan B. Jaringan A adalah *host* yang akan diuji QoS-nya sementara jaringan B digunakan untuk membanjiri *traffic*. Pada jaringan A, terdiri sebuah *access point* dan tiga buah *host*. Sementara jaringan B terdiri dari sebuah *host*. Gambar dari topologi jaringan yang dirancang dapat dilihat pada gambar 2.

4. Menguji Quality of Service jaringan simulasi

Pada tahapan ini akan dilakukan pengujian QoS dari jaringan simulasi yang sudah di bangun dan akan dilakukan dengan dua tahap pengujian. Pada tahap pertama, semua *host* pada jaringan A akan dipenuhi *traffic* nya oleh jaringan B yang bertujuan untuk membuat *traffic* pada jaringan A penuh sehingga tidak bisa mengakses jaringan luar, atau bahasa lain dari pengujian ini adalah DoS, adapun DoS yang dilakukan adalah *SYN Flood* dan UDP *Flood* dengan menggunakan *script* berbasis PYTHON. Setelahnya akan dilakukan pengujian QoS untuk melihat kemapuan dari layanan jaringan simulasi tersebut.

Pada tahapan kedua, jaringan A pada simulasi akan dikonfigurasikan dengan metode HTB dimana metode ini akan mengatur batasan *bandwidth* pada jaringannya dengan maksimal *bandwidth* yaitu 20Mbps sementara minimal *bandwidth* sebesar 10Mbps, sehingga traffic jaringan A tidak akan berjalan dibawah nilai yang sudah diatur dengan sistem HTB. Metode HTB sendiri dimasukkan pada perangkat *access point* melalui controller.

Adapun pada pengujian QoS ini akan diuji 3 parameter pengujian. Yaitu pengujian *jitter*, *packet loss*, serta *throughput*. Untuk pengujian *jitter* dapat dilihat pada tabel 5 dibawah ini.

Tabel 5 Pengujian jitter dengan iperf3

	Host 1	*Host* 4
IP Adderss	10.0.1.10	11.0.0.10
CLI	iperf3 –c 11.0.0.10 -u	iperf3 –s

Tabel diatas menunjukkan cara pengujian jitter, dimana host 1 bertindak sebagai client dan sementara host 4 bertindak sebagai server.

Untuk pengujian *packet loss* dapat dilihat pada tabel 6 dibawah ini.

Tabel 6 Pengujian packet loss dengan ICMP

	Host 1	*Host* 4
IP Adderss	10.0.1.10	11.0.0.10
CLI	*Ping* –c (c) –s (s) 11.0.0.10	-

Tabel diatas menunujukka cara pengujian *packet loss*, dari *host* 1 ke *host* 4 dimana c merupakan banyaknya paket, sementara s adalah besar beban.

Untuk pengujian *throughput* dilakukan dengan dua tahapan, tahapan pertama diuji *throughput* dengan protokol TCP dan tahapan kedua diuji *throughput* dengan protokol UDP. Untuk pengujian *throughput* dapat dilihat pada tabel 7 dibawah ini.

Tabel 7 Pengujian *throughput* dengan protokol TCP iperf3

	Host 1	*Host* 4
IP Adderss	10.0.1.10	11.0.0.10
CLI	iperf3 –c 11.0.0.10	iperf3 –s

Tabel diatas menunjukkan cara pengujian *packet loss*, dimana *host* 1 bertindak sebagai *client* dan sementara *host* 4 bertindak sebagai *server*.

5. Analisis hasil quality of service

Pada tahapan ini, akan dilakukan analisis hasil pengujian QoS pada kedua tahapan pengujian. Dimana hasil dari kedua pengujian akan dibandingkan manakah hasil yang lebih baik dan manakah hasil yang lebih buruk.

6. Penulisan laporan akhir

Berdasarkan hasil dari analisi hasil QoS. Langkah akhir dari penelitian ini adalah penulisan dan penyusunan laporan akhir sebagai dokumentasi dari penelitian yang telah dilakukan.

4. Hasil dan Pembahasan

Pengujian QoS pada jaringan berbasis SDN dilakukan dengan dua tahapan. Yang pertama, pengujian dilakukan dengan cara seluruh jaringan B memenuhi *traffic* jaringan A dan akan diuji *jitter*, *packet loss*, serta *throughput* dari jaringan A akan tetapi metode HTB tidak dimasukkan pada jaringan A. Sementara pada tahapan kedua, pengujian dilakukan dengan cara metoda HTB dimasukkan kedalam jaringan A, dan kemudian jaringan B memenuhi *traffic* jaringan A, dan kemudian akan dilakukan uji yang sama yang dilakukan pada tahapan pertama. Pada saat

pengujian, juga di lakukan *Ping* dengan beban 64 *Bytes*, 100 *Bytes*, 1000 *Bytes*, dan 3000 *Bytes*, dan juga diulangi sebanyak tiga kali perulangan untuk mendapatkan hasil yang maksimal.

Berikut adalah keseluruhan hasil penelitian yang didapatkan dari pengujian QoS jaringan berbasis SDN.

4.1. Jitter

Jitter adalah perbedaan selang waktu kedatangn antar paket di terminal tujuan. Munculnya jitter disebabkan banyak faktor, salah satunya adalah peningkatan traffic jaringan secara tiba-tiba sehingga menyebabkan penyempitan bandwidth.

Tabel 8 Hasil rata-rata pengujian *jitter* pada *host* 1

Jitter (*ms*)			
Ping	Tanpa HTB, Tanpa UDP *Flood*	Tanpa HTB, UDP *Flood*	HTB, UDP *Flood*
Tanpa *Ping*	0	13.1	5
64 *Bytes*	0	13.1	6.3
100 *Bytes*	0	26.4	10.7
1000 *Bytes*	0	-1	11.1
3000 *Bytes*	0	-1	13.8

*-1 =Hasil tidak keluar

Tabel diatas merupakan hasil pengujian *jitter* pada *host* 1 yang hasilnya sudah dirata-ratakan terlebih dahulu. Dapat dilihat pada hasil diatas, menunjukkan bahwa nilai *jitter* pada jaringan simulasi ini terlihat sangat buruk pada saat dipenuhi *traffic*nya oleh UDP *Flood* yang jaringannya tidak disisipi oleh metode HTB. Dapat dilihat pada tabel, nilai pada percobaan dengan beban *Ping* sebesar 1000 *Bytes* dan 3000 *Bytes* hasil pengujian tidak keluar. Ini disebabkan pada saat pengujian tanpa metode HTB, *traffic* jaringan meningkat dan menyebabkan penyempitan *bandwidth* sehingga tidak dapat lagi membalas permintaan dari *host* 1.

Akan tetapi, pada saat metode disisipi dapat dilihat bahwasanya hasil *jitter* yang diperoleh sudah menurun, dan dapat bisa dikatan stabil. Ini dikarenakan metode HTB dapat mengatur *bandwidth* dari jaringan, sehingga tidak ada lagi terjadinya penyempitan *bandwidth* walaupun *traffic* dari jaringan sedang meningkat. Dapat dilihat pada tabel hasil pengujian, nilai *jitter* yang dihasilkan oleh metode HTB bernilai memuaskan dari standar TIPHON.

4.2. Packet Loss

Packet loss adalah jumlah paket yang hilang dalam suatu pengiriman paket data pada suatu jaringan. Terdapat beberapa penyebab dari *packet loss* diantaranya adalah adanya *noise* (sinyal-sinyal yang tidak diinginkan yang selalu ada dalam suatu sistem transmisi), *collision* (hilangnya data yang terjadi karena dua buah *device* mengirim data pada saat yang bersamaan) dan *congestion* (tabrakan yang akan menyebabkan terlambatnya data sampai) yang disebabkan oleh terjadinya antrian yang berlebihan dalam jaringan.

Tabel 9 Hasil rata-rata pengujian *packet loss* protocol TCP pada *host* 1

Packet Loss (%)			
Ping	Tanpa HTB, Tanpa SYN *Flood*	Tanpa HTB, SYN *Flood*	HTB, SYN *Flood*
64 *Bytes*	0	0	0
100 *Bytes*	0	0	0
1000 *Bytes*	0	0	0
3000 *Bytes*	0	0	0

Tabel diatas merupakan hasil pengujian *packet loss* pada *host* 1 yang hasilnya sudah dirata-ratakan terlebih dahulu. Dapat dilihat dari hasil pengujian diatas, bahwasanya untuk protokol TCP, dengan adanya dan tidaknya metode HTB tidak mempengaruhi pengiriman data dari jaringan, sehingga tidak adanya *packet loss* yang dihasilkan. Ini membuktikan bahwasanya untuk protokol TCP, jaringan ini sudah terbilang sangat baik karena tidak terdapatnya *noise*, *collision* dan *congestion* yang disebabkan oleh terjadinya antrian yang berlebihan dalam jaringan walaupun *traffic* dari jaringan sedang dipenuhi oleh *SYN Flood*.

Tabel 10 Hasil rata-rata pengujian *packet loss* protocol UDP pada *host* 1

Packet Loss (%)			
Ping	Tanpa HTB, Tanpa UDP *Flood*	Tanpa HTB, UDP *Flood*	HTB, UDP *Flood*
64 *Bytes*	0	30.6	0
100 *Bytes*	0	34.3	0.3
1000 *Bytes*	0	47.3	2.6
3000 *Bytes*	0	62.3	10.6

Tabel diatas merupakan hasil pengujian *packet loss* pada *host* 1 yang hasilnya sudah dirata-ratakan terlebih dahulu. Dapat dilihat dari hasil pengujian diatas, bahwasanya untuk protokol UDP, UDP *Flood* sangat mempengaruhi performa jaringan pada saat pengiriman data. Bisa dilihat pada kolom tanpa metode HTB, *packet loss* yang dihasilkan sangat besar, bahkan saat *Ping* dengan paket data sebesar 3000 *Bytes*, *packet loss* yang dihasilkan mencapai 62.3%. Ini dikarenakan terdapat banyaknya *noise*, *collision* dan *congestion* pada saat UDP *Flood* dijalankan. Akan tetapi, pada saat metode HTB disisipkan, besar *packet loss* menurun drastis. Untuk *Ping* dengan paket data sebesar 3000 *Bytes*, *packet loss* yang dihasilkan hanya 10.6%, ini menandakan pada saat metode HTB, UDP *Flood* dapat diredam oleh metode HTB sehingga *noise*, *collision* dan *congestion* pada jaringan dapat diminimalisir, sehingga *packet loss* menurun dan dapat dikategorikan dengan nilai memuaskan dari standar TIPHON.

4.3. Throughput

Throughput merupakan kemampuan sebenarnya suatu jaringan dalam melakukan pengiriman data. Yaitu kecepatan (*rate*) *transfer* data efektif, yang diukur dalam *bps*.

Tabel 11 Hasil rata-rata pengujian *throughput sender* protocol TCP pada *host* 1

Throughput Sender (%)			
Ping	Tanpa HTB, Tanpa SYN *Flood*	Tanpa HTB, SYN *Flood*	HTB, SYN *Flood*
Tanpa *Ping*	97.5	96.5	97
64 *Bytes*	97.5	96	97
100 *Bytes*	97.5	94.5	96
1000 *Bytes*	97.5	91.5	94.5
3000 *Bytes*	97.5	89.5	93.5

Tabel diatas merupakan hasil pengujian *throughput sender* pada *host* 1 yang hasilnya sudah dirata-ratakan terlebih dahulu. Dapat dilihat dari hasil pengujian diatas, dapat dilihat pada pengujian dengan *SYN Flood* tanpa menggunakan metode HTB pada *Ping* 3000 *Bytes* menghasilkan *throughput sender* sebesar 89.5%, yang menunjukkan penurunan kualitas, akan tetapi dapat dikatan tidak terlalu signifikan. Akan tetapi, saat menggunakan metode HTB, *throughput sender* yang dihasilkan terbilang stabil.

Tabel 12 Hasil rata-rata pengujian *throughput receiver* protocol TCP pada *host* 1

Throughput Receiver (%)			
Ping	Tanpa HTB, Tanpa SYN *Flood*	Tanpa HTB, SYN *Flood*	HTB, SYN *Flood*
Tanpa *Ping*	97	95	96
64 *Bytes*	97	95	95.5
100 *Bytes*	97	94.5	95.5
1000 *Bytes*	97	90.5	93
3000 *Bytes*	97	89	92.5

Tabel diatas merupakan hasil pengujian *throughput receiver* pada *host* 1 yang hasilnya sudah dirata-ratakan terlebih dahulu. Dapat dilihat dari hasil pengujian diatas, bahwasanya untuk protokol TCP yang dihasilkan sama stabilnya seperti seperti *throughput sender* yang menunjukkan bahwasanya jaringan tersebut sudah baik dan dapat dikategorikan memuaskan.

Tabel 13 Hasil rata-rata pengujian *throughput* protokol UDP pada *host* 1

Throughput (%)			
Ping	Tanpa HTB, Tanpa UDP *Flood*	Tanpa HTB, UDP *Flood*	HTB, UDP *Flood*
Tanpa *Ping*	5.2	4.9	5.2
64 *Bytes*	5.2	4.9	5.2
100 *Bytes*	5.2	4.9	5.2
1000 *Bytes*	5.2	1.25	5.2
3000 *Bytes*	5.2	0.75	5.2

Tabel diatas merupakan hasil pengujian *throughput* pada *host* 1 yang hasilnya sudah dirata-ratakan terlebih dahulu. Dapat dilihat dari hasil pengujian diatas, bahwasanya untuk protokol

UDP, bandwdith yang dihasilkan pada tanpa metode HTB hasil yang diberikan sangat aman kecil, bahkan pada percobaan dengan *Ping* sebesar 1000 *Bytes*, hasil *throughput*nya adalah 1.25%, bahkan pada *Ping* 3000 *Bytes* hasilnya hanya sebesar 0.75%. Dari hasil tersebut dapat disimpulkan bahwasanya pada saat UDP *Flood* mememenuhi *traffic* protokol UDP, serangan tersebut sangat berpengaruh pada layanan jaringan.

Akan tetapi, pada saat metode HTB dimasukkan, aktifitas jaringan kembali normal. Hal ini dapat dilihat pada tabel, walaupun pada saat penyerangan terjadi dengan tambahan *Ping* dengan beban sebesar 1000 *Bytes* dan 3000 *Bytes*, jaringan tetap berjalan lebih baik dibandingkan tanpa metode HTB. Hal ini menguatkan argumen bahwasanya dengan metode HTB, penyerangan pada protokol UDP dapat diatasi dengan mencoba untuk memberikan *bandwidth* yang cukup walaupun *traffic* pada jaringan sedang padat.

5. Kesimpulan

Dari hasil pengujian QoS pada *host* 1 yang dihasilkan, dapat disimpulkan bahwasanya metode HTB sangat membantu memaksimalkan jaringan simulasi ini. Pada saat HTB dimasukkan, besar *jitter* yang dihasilkan pada saat pengujian tidak ada yang melebihi 14ms, yang menunjukkan bahwasanya *jiiter* pada jaringan tersebut berada di kategori bagus dengan nilai indeks 3 dari standar TIPHON.

Pada pengujian *packet loss*, juga hasil yang didapatkan berkurang dengan drastis, sehingga pengiriman data jaringan dapat berjalan tanpa banyak hilangnya data. *Packet loss* yang dihasilkan setelah menambahkan metode HTB sudah bisa dibilang memuaskan, dari semua percobaan, hanya saat dengan *ping* sebesar 3000 *Bytes* yang hanya menghasilkan *packet loss* sebesar 10.6%. Jika seluruh parameter *ping* dirata-ratakan lagi, hasil *packet loss* yang dihasilkan hanya sebesar 3.3% dan mendapatkan kategori memuaskan, dan mendapatkan indeks sebesar 3.

Untuk hasil *throughput* yang dihasilkan juga menunjukkan bahwasanya metode HTB dapat memaksimalkan layanan jaringan. Dapat dilihat pada tabel hasil pengujian, besaran *throughput sender* dengan protokol TCP yang dihasilkan tidak ada yang berada dibawah 89%. Jika semua parameter *ping* dirata-ratakan, makanya akan mendapatkan *throughput sender* sebesar 95.6% dan mendapatkan kategori bagus. Sementara untuk *throughput receiver* yang dihasilkan juga terbilang bagus, dan apabila seluruh parameter *ping* dirata-ratakan akan menghasilkan *throughput receiver* sebesar 94.5%.

Sedangkan hasil dari *throughput* dengan protokol UDP, *throughput* yang dihasilkan dapat dibilang sangat buruk. Karena jika hasilnya dirata-ratakan maka akan menghasilkan *throughput* sebesar 5.2%. Jika dikategorikan maka *throughput* akan mendapatkan kategori jelek, dan mendapatkan indeks 1. Jika hasil seluruh indeks *throughput* dirata-ratakan, makan akan mendapatkan hasil indeks sebesar 2.3, dan dapat dikategorikan kurang memuaskan.

Jika seluruh hasil indeks dirata-ratakan, maka akan mendapatkan indeks sebesar 2.8, ini membuktikan bahwasanya dengan metode HTB, jaringan simulasi ini dapat dikategorikan jaringan yang kurang memuaskan. Ini dapat dilihat dari hasil semua pengujian yang menunjukkan bahwasanya terdapat penurunan perfomansi yang signifikan dari *throughput* pada protokol UDP yang hanya menghasilkan *throughput* sebesar 5.2%, sehingga mempengaruhi hasil keseluruhan pengujian QoS, walaupun hasil dari *jiiter*, *packet loss*, dan *throughput* pada protokol TCP yang dihasilkan sudah memuaskan.

Referensi

[1] Dicky H. 2012, Analisis Perancangan dan Implementasi Jaringan Komputer Lokal PT.Sukanda Djaya Yogyakarta, Tugas Mandiri & Seminar, Jurusan Teknik Informatika, FTI, IST AKPRIND, Yogyakarta

[2] Faiz, M. A. 2012. Sistem Jaringan Komunikasi Komputer. Jawa Tengah: Fakultas Teknologi Informasi.

[3] Astuto, B. A. Nunes, M. M.-N. 2014. *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*. IEEE *Communications Surveys & Tutorials*, *Volume*. 16, No. 3, *Third Quarter*.

[4] Kartadie, R. 2016. Mikrotik RB750 Routerboard Sebagai Alternatif Switch Openflow Software-Base. Yogyakarta: Sleman.

[5] Rosyid, N. R. 2017. Analisis Kinerja dan Karakteristik Arsitektur *Software-Defined Network* Berbasis OpenDaylight *Controller*. Yogyakarta: Universitas Gadjah Mada.

[6] Anam, K. dan Adrian, R. 2017. Analisis Performa Jaringan *Software-Defined Network* Berdasarkan Penggunaan *Cost* Pada Protokol *Routing Open Shortest Path First*.Yogyakarta: Universitas Gadjah Mada.

[7] Anggara, S., M., 2015. Pengujian Performa Kontroler SDN: POX dan Floodlight.Institut Teknologi Bandung: Bandung.

[8] Antodi, C., P., dkk., 2017, Penerapan *Quality of Service* Pada Jaringan Internet Menggunakan Metode *Hierarchical Token Bucket*. Jurnal Teknologi dan Sistem Komputer ISSN:23380403.

[9] Iskandar, I. dan Hidayat, A. 2015. Analisa *Quality of Service* (*QoS*) Jaringan Internet Kampus (Studi Kasus: UIN Suska Riau). Pekanbaru: UIN Sultan Syarif Kasim.

[10] Gani, A. 2010. Aplikasi Pengaruh *Quality of Service (QoS)* Video *Conference* Pada *Traffic* H.323 Dengan Menggunakan Metode *Differentiated Service* (*Diffserv*). Aceh: Universitas Syiah Kuala.

[11] Ramadhani, K.. dkk. 2013. Pendeteksian Dini Serangan UDP *Flood* Berdasarkan Anomali Perubahan *Traffic* Jaringan Berdasarkan *Cusum Algorithm*.

[12] Pamungkas, C.A. 2016. "Manajemen *Bandwidth* Menggunakan Mikrotik RouterBoard di Politeknik Indonusa Surakarta," INFORMA Politeknik Indonusa Surakarta.

[13] Septiawan, D. A. 2013. “Membangun Prioritasisasi Lalu Lintas Data (Internet) Menggunakan HTB Queueing Disciplines Pada Jaringan Lokal SMK N 1 Nanggulan”. Naskah Publikasi; Amikom Yogyakarta

[14] Towidjojo, R. 2014. “Mikrotik Kung Fu: Kitab 3 Manajemen *Bandwidth*”. Jasakom.

[15] Nugraha, M., & Utama, S., N., 2016, Implementasi Manajemen *Bandwidth* Dengan Disiplin Antrian *Hierarchical Token Bucket* (HTB) Pada Sistem Operasi Linux. Jurnal Ilmu Komputer dan Teknologi Informasi.ISSN : 1978-161X

[16] Soniya, Y.; Priyono, W. A., dan Ambarwati, R. 2013. Sistem Manajemen *Bandwidth* dengan Prioritas Alamat IP *Client*. Jurnal Penelitian Vol. 2.

[17] Karamjeet K., Singh, J., Ghumman, N. S. 2014. *Mininet as Software Defined NetworkingTesting Platform. International Conference on Communication, Computing & Systems* (ICCCS).