ISSN 2355-3286

Rancang Bangun Mekanisme Quality of Service terhadap Protokol RTP dan SIP pada Arsitektur Openflow

Richard Alvianto[1], Samuel Hutagalung[2], Franciscus Ati Halim[3]
Program Studi Teknik Komputer, Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
samuel.hutagalung@umn.ac.id

Diterima 7 Mei 2019
Disetujui 24 Juni 2019

***Abstrak*— Pada beberapa tahun terakhir, angka dari penggunaan *Voice Over Internet Protocol* (VoIP) terus meningkat. Di sisi lain masih terdapat suatu kendala, seperti penggunaan *bandwidth* yang tidak terbagi rata sesuai dengan kebutuhan masing-masing paket. Hal ini dapat menjadi suatu masalah dikarenakan paket VoIP membutuhkan *delay, jitter,* dan *packet loss* seminimal mungkin untuk menjamin kualitas suara yang dihasilkan. Penelitian ini menggunakan mekanisme *Quality of Service* (QoS) untuk memberikan prioritas terhadap paket *Real-time Transport Protocol* (RTP) dan *Session Initiation Protocol* (SIP) agar kualitas VoIP tetap terjaga. Pengujian dalam penelitian ini dilakukan pada emulator mininet dengan menganalisis beberapa parameter QoS. Pada skenario pengujian, jaringan akan menampung paket hingga 100 Mbps untuk menciptakan kondisi lalu lintas paket yang padat. Jenis paket yang dikirimkan dalam pengujian terdiri dari RTP, STP, dan data *dummy*. Berdasarkan hasil pengujian, ketika menggunakan mekanisme QoS terbukti nilai *delay, jitter,* dan *packet loss* dapat berkurang dan memenuhi standar ITU-T G.1010. Sedangkan pada kondisi tidak menerapkan QoS, paket RTP dan SIP memperoleh nilai *delay, jitter, packet loss* yang lebih tinggi dan tidak memenuhi standar dari ITU-T G.1010.**

Kata Kunci — Delay; Jitter; Over Internet Procol; Packet Loss; Quality of Service; Real-time Transport Protocol; Session Initiation Protocol; Voice.

I. PENDAHULUAN

Perkembangan teknologi yang pesat membawa perubahaan pada dunia telekomunikasi. Pada mulanya, tarif telepon konvensional relatif mahal baik untuk dalam maupun luar negeri. Namun dengan kemajuan teknologi, kini telepon dapat melalui jaringan Internet Protocol (IP) yang biasanya disebut dengan Voice Over Internet Protocol (VoIP).

VoIP memanfaatkan infrastruktur berkomunikasi dengan mengubah sinyal analog menjadi sinyal digital dan dialirkan melalui jaringan [1]. Trafik VoIP dibagi menjadi dua jenis transmisi jaringan yaitu *signaling* dan pengiriman suara atau *video*. Protokol *signaling* yang digunakan pada umumnya ialah *Session Initiation Protocol* (SIP), sedangkan protokol yang berfungsi untuk mengirim suara secara *real time*adalah *Real-time Transport Protocol* (RTP). Kedua protokol berbasis User Datagram Protocol (UDP) yang lebih mementingkan kecepatan suatu paket sampai tujuan daripada kehandalan untuk memastikan suatu paket dikirim dan diterima, dengan karakteristik UDP yang *connectionless* dengan mengirim paket tanpa melalui proses negosiasi antara pengirim dan penerima paket, dan tidak dapat memastikan apakah suatu paket diterima secara berurut [2],[3].

Trafik VoIP transmisi pada satu medium dalam jaringan, tentu akan menghadapi beberapa permasalahan, seperti trafik kelebihan beban, dan berbagai jenis paket yang saling mendahului sehingga tidak dapat diprediksi paket mana yang mendapatkan bandwidth lebih besar.

Padatnya trafik dalam sebuah jaringan menyebabkan waktu yang dibutuhkan untuk mengirim paket sampai ke tujuan menjadi lebih lama dan kemacetan transmisi paket mengakibatkan paket dibuang. Ini memberikan dampak buruk pada paket VoIP yang sensitif, apabila dalam kondisi sedang berkomunikasi dengan VoIP, yang terjadi adalah terdapat suara yang hilang dan kualitas suara yang buruk, sehingga mengganggu kenyaman pengguna VoIP.

Dengan menggunakan arsitektur OpenFlow dapat memisahkan antara *dataplane* dan *controlplane*, yang memungkinkan untuk pemrograman perilaku *switch* atau *router* sesuai dengan yang diinginkan secara terpusat dari kontroller, sehingga tidak hanya terpaku dengan fitur-fitur yang disediakan oleh vendor, dan memungkinkan fleksibilitas dalam implementasi fitur yang diinginkan dan dalam jaringan berskala besar lebih mudah untuk melakukan pemeliharaan perangkat tersebut.

Tujuan dari penelitian ini adalah untuk membangun sebuah mekanisme QoS yang diperuntukkan untuk menjaga *delay*, *jitter*, dan *packet loss* dari RTP maupun SIP agar tetap sesuai dengan standard kebutuhan.

II. TINJAUAN PUSTAKA

A. *Quality of Service (QoS)*

Quality of Service merupakan kemampuan untuk memberikan layanan yang lebih baik lalu lintas jaringan dengan menyediakan *bandwidth*, *jitter* dan *delay* yang terkendali [4]. Kinerja dari jaringan tidak terlepas dari beberapa akibat seperti *packet loss*, *delay*, *jitter* yang dapat memberikan efek yang cukup berpengaruh untuk banyak aplikasi seperti VoIP.

Dalam pengukuran nilai dari parameter QoS ini merujuk pada standarisasi dari ITU-T.G.1010 mengenai batas nilai yang telah ditentukan dalam rangka menjamin kualitas suatu layanan dapat diterima maupun dirasakan baik oleh pengguna, berikut tabel batas nilai dari standar ITU-T. G.1010.

Tabel 1. Standar QoS ITU-T.G1010

Parameter	Voice over IP (VoIP)
Delay	< 150ms *preferred*
Jitter	< 1ms
Packet loss	< 3%

Parameter-parameter QoS terdiri dari:

- *Throughput* merupakan total jumlah kedatangan paket yang sukses diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu, throughput biasanya dikenal dengan bandwidth.
- *Packet loss* dapat digambarkan dalam suatu kondisi yang menunjukan total jumlah paket yang hilang dapat terjadi disebabkan oleh kepadatan trafik, kongesti dalam jaringan atau terjadi overflow pada buffer.
- *Delay (latency)* adalah waktu yang dibutuhkan paket untuk menempuh jarak dari asal menuju tujuan. Delay dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga proses yang lama.
- *Jitter* adalah variasi paket masuk, hal yang mengakibatkan terjadinya variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket diakhir perjalanan jitter.

B. Real-time Transport Protocol (RTP)

Merupakan sebuah standarisasi paket yang mengirimkan suara dan video pada jaringan. RTP umumnya dikonfigurasi menggunakan port antara 16834 – 32767, RTP biasanya digunakan sebagai salah pondasi penting untuk komunikasi *Voice over Internet Protocol* (VoIP) [5]. Protokol RTP pada lapisan aplikasi yang berperan untuk mentransmisikan paket secara *realtime* dan besar ukuran paket bergantung pada penggunaan *codec* dan RTP digunakan berpasangan UDP dengan tujuan dapat mengirim paket secara *real-time*.

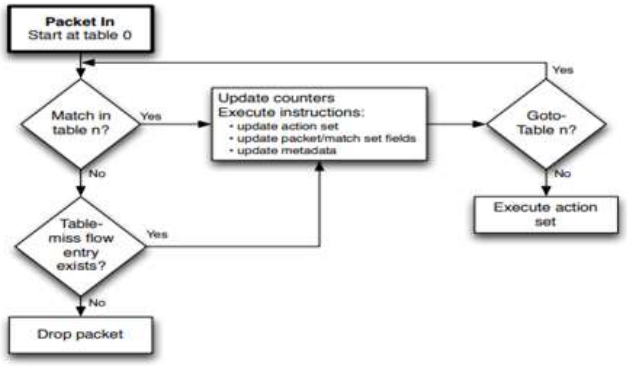
C. Session Initial Protocol

Merupakan salah satu standardisasi pensinyalan dan pengontrolan sesi yang dikembangkan oleh IETF, protokol yang berada pada lapisan aplikasi yang berfungsi untuk proses membangun, memodifikasi dan mengakhiri suatu sesi komunikasi multimedia. SIP dapat mengontrol sinyal untuk jaringan IP [2]. Protokol SIP bukan merupakan protokol *transport* yang membawa paket data suara maupun video, SIP berjalan diatas lapisan UDP yang menggunakan *port* 5060.

User Agent secara logis terbagi menjadi dua yaitu *User Agent Client* (UAC) sebagai entitas yang memulai dan mengirimkan SIP *request*, sedangkan *User Agent Server* (UAS) berperan dalam menerima dan menjawab dari SIP *request*. Setiap SIP *message* terdiri dari permintaan pesan atau tanggapan pesan dari UAS maupun UAC [6].

D. OpenFlow

OpenFlow adalah salah satu *open* standar komunikasi protokol yang mampu melakukan pemisahan antara *control plane* dan *data plane* dari sebuah perangkat jaringan, serta untuk mampu menciptakan komunikasi yang baik antara *control plane* dan *data plane*[7]. *Flowtable* pada *OpenFlow* merupakan tabel yang digunakan untuk mengidentifikasi proses yang dilakukan oleh *switch* terhadap setiap paket yang diterima.



Gambar 1. OpenFlow diagram alur paket.

Berikut adalah proses aliran paket pada *switch OpenFlow*[8]:

- Setiap paket masuk di *switch*, pada bagian header paket diperiksa berdasarkan flow table.
- Jika di temukan sesuai dengan entri *flow table*, maka akan menerapkan instruksi terkait berdasarkan aliran paket.
- Jika tidak sesuai dengan *flow table*, maka paket akan di arahkan ke entri *table miss*. Yang dimana table-miss adalah entri yang menentukan set instruksi yang akan di terapkan terhadap paket yang masuk ketika tidak ada yang cocok dengan *flow table*. Instruksi terdiri dari membuang paket

(*dropping*), mengirim paket pada semua interface dan meneruskan paket ke *controller*.

E. Mininet

Mininet adalah emulator jaringan yang menciptakan prototipe jaringan *host virtual*, *switch*, pengendali dan tautan. *Host* di *mininet* menjalankan perangkat lunak jaringan *Linux* standar dan *switch* mendukung *OpenFlow* untuk kustom yang sangat fleksibel. *Mininet* mendukung untuk penelitian, pengembangan, pembelajaran, prototipe, pengujian maupun *debugging*, dan tugas lain yang dapat kelebihan dari memiliki jaringan eksperimental yang lengkap pada laptop atau PC lainnya [9].

F. Ryu Controller

Ryu adalah *controller* yang kompatibel pada arsitektur *OpenFlow* yang dirancang untuk meningkatkan kecepatan jaringan dengan membuatnya mudah untuk mengelola dan beradaptasi dengan trafik jaringan, Secara umum kontroller adalah otak dari OpenFlow, yang mengkomunikasikan informasi ke *Switch* dan *Router* dengan *southbound* API dan logika bisnis dengan *northbound* API [10].

G. Hierarki Token Bucket (HTB)

HTB merupakan kelas yang berbasis *queue discipline* (qdisc), yang mempunyai kemampuan mengantri paket sesuai urutan waktu yang tertulis dalam sebuah algoritma, secara konseptual HTB yaitu sejumlah *token bucket* yang disusun dalam suatu hirarki. Secara sederhana, alogritma ini menawarkan kemudahan pemakaian dengan teknik peminjaman dan implementasi pembagian trafik yang lebih akurat [11].
Dengan metode HTB mampu untuk melakukan pembagian trafik yang lebih akurat dan teknik untuk melakukan antrian, dengan menggunakan antrian HTB dapat memberikan fasilitas dengan membatasi trafik pada setiap level ataupun klasifikasinya, sehingga bandwidth yang sedang tidak digunakan dapat digunakan oleh klasifikasi lainnya yang lebih rendah.

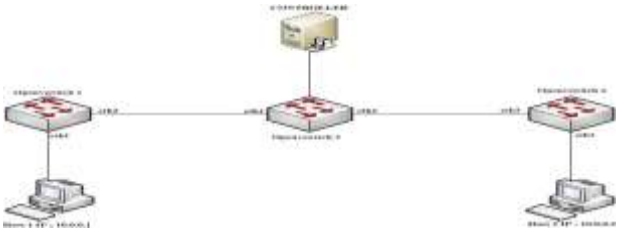
Parameter yang penyusunan dalam antrian HTB yaitu [11]:

- *Rate* merupakan parameter untuk menentukan batas bandwidth maksimum yang dapat digunakan oleh setiap kelas, bila bandwidth melebihi dari nilai rate, maka paket data akan di drop.
- *Ceil* merupakan parameter untuk menentukan peminjaman bandwidth antar kelas, peminjaman bandwidth dilakukan kelas paling bawah ke kelas di atasnya, teknik ini biasa di sebut dengan link sharing.

III. PERANCANGAN SISTEM

A. Perancangan Topologi

Pada penelitian ini, merancang topologi dengan *host* sebanyak dua unit, *Openvswitch* sebanyak tiga unit, *controller* satu unit. semua perangkat tersebut disimulasi di dalam emulator mininet, dengan topologi jaringan yang akan dirancang.



Gambar 2. Perancangan topologi.

Berikut komponen dari topologi yang akan dikembangkan:

- *Openvswitch* menjadi alat untuk menghubungkan antar perangkat dalam topologi dan juga tiap *Openvswitch* menggunakan protokol *OpenFlow* untuk berkomunikasi dengan kontroller.
- Kontroller akan berperan sebagai otak yang mengatur *openvswitch* yang bekerja sehingga kontroller yang bertanggung jawab untuk mengontrol *switch* dengan menggunakan protokol *OpenFlow*, dan juga perangkat lunak yaitu ryu kontroller. Komponen ini menggunakan Alamat *IP* tetap untuk komponen ini yaitu 127.0.0.1 dan port 6633.
- Host merupakan *client* maupun *server* yang dapat mengirim hingga menerima paket RTP dan SIP melalui *switch*.
- *Link* adalah *Ethernet* fisik dengan kecepatan 100Mbps sebanyak enam buah

B. Diagram alur klasifikasi paket masuk.

Pada proses klasifikasi setiap paket masuk pada *switch* yang terhubung *host*, berdasarkan topologi yang telah dipaparkan, terdapat tiga unit *switch* yang dimana di antara ketiga *switch* yang terhubung langsung dengan *host* terdapat dua unit dan perangkat tersebut disimulasikan dengan emulator *mininet.*

Gambar 3. Diagram alur *Flowtable.*

Dalam penelitian ini fokus terhadap protokol RTP hanya menggunakan port 30000 dan SIP dengan port 5060 yang dimana kedua protokol merupakan paket UDP sehingga pada klasifikasi paket untuk memetakan jenis paket UDP maupun TCP dengan menambahkan atau mengubah nilai pada bagian DSCP, yang dimana untuk setiap paket RTP dan SIP diberikan nilai 46 pada bagian DSCP sedangkan selain dari RTP dan SIP akan diberi nilai 0 pada bagian DSCP, kemudian berdasarkan kedua nilai untuk menentukan suatu paket masuk dalam antrian prioritas tinggi atau antrian prioritas rendah.

C. Perancangan jaringan dalam mininet

Proses pembuatan topologi hingga simulasi selesai dan semua proses yang ditulis dengan pemrograman *python* di *Mininet*.

Gambar 4. Diagram alur membuat topologi di Mininet.

Pada proses awal dengan memanggil *library* yang akan digunakan untuk membuat topologi, setelah pemanggilan *library* dapat melakukan inisialisasi kontroller, *host*, *switch*. Pada kontroller dapat diberi nama, protokol *OpenFlow*13 yang akan digunakan, IP *address* dan *port number*, pada komponen *host* dan *switch* juga dapat diberikan parameter seperti IP *address* yang akan digunakan, MAC *address*, maupun *Link* untuk menghubungan antar komponen. Setelah semua komponen sudah siap sehingga sudah dapat melakukan proses pembuatan topologi secara keseluruhan, dan ketika proses pembuatan topologi dapat berjalan tanpa ada masalah, maka topologi sudah berjalan dengan semua perangkat seperti kontroller, *switch* maupun *host* sudah dapat memulai simulasi dengan *command line interface* (CLI) dan ketika ingin mengakhiri simulasi dapat menggunakan perintah "*exit*" pada CLI, maka dengan otomatis mininet akan menghapus semua komponen dan topologi dengan bersih sehingga tidak tersisa-sisa yang dapat berpengaruh dengan pembuatan topologi yang baru.

IV. Implementasi dan Pengujian Sistem

A. *Implementasi*

Implementasi sistem yang merupakan lanjutan dari perancangan sistem. Rancang Bangun Mekanisme *Quality of Service* Terhadap Protokol RTP dan SIP Pada Arsitektur *OpenFlow*. Kontroller yang digunakan adalah Ryu kontroller dirancang dengan menggunakan Bahasa python dan metode yang digunakan untuk mengatur proses suatu paket dalam antrian dengan hierarki *token bucket* yang dapat membagi *bandwidth* secara akurat untuk masing-masing antrian berdasarkan prioritas.

B. *Menjalankan Ryu dan membangun topologi dengan mininet.*

Menjalankan Ryu kontroller pada terminal CLI linux dengan perintah "PYTHONPATH=. ./bin ryu-manager ryu.app.SkripsiQoS" maka Ryu kontroller akan mengeksekusi kode aplikasi dengan Bahasa python sehingga kontroller sudah dapat berjalan dan menunggu topologi yang terdapat kontroller yang akan berkomunikasi dengan Ryu yang telah berjalan.

```
root@sdnhubvm:/home/ubuntu/ryu# PYTHONPATH=. ./bin/ryu-manager ryu.app.SkripsiQoS
loading app ryu.app.SkripsiQoS
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.app.SkripsiQoS of SimpleSwitch13
```

Gambar 5. Menjalankan aplikasi Ryu kontroller.

Perangkat lunak Mininet akan digunakan untuk membuat topologi yang akan di simulasikan, dengan menjalankan perintah "sudo python skripsi_topo.py", dalam topologi yang dibuat untuk masing-masing *interfaces* yang di miliki *switch* telah disisipkan konfigurasi untuk QoS HTB yang akan membagi *bandwidth* dengan maksimal kecepatan 100Mbps yang terdapat sub antrian nol dengan batas maksimal memperoleh bandwidth 100Mbps sedangkan untuk sub antrian satu dengan batas minimal bandwidth yang diperoleh adalah 70Mbps dan batas maksimal adalah 100Mbps di setiap interfaces *switch*. Setelah menjalankan *command* tersebut maka secara otomatis masuk di *Mininet* sehingga topologi sudah dapat disimulasikan.

```
root@sdnhubvm:/home/ubuntu/ryu/ryu/app# python skripsi_topo.py
mininet>
```

Gambar 6. Membuat topologi dengan kode python.

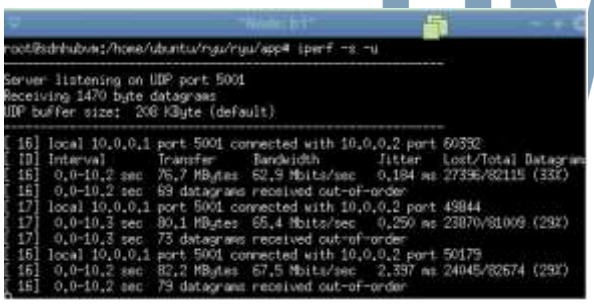
C. Skenario pengujian QoS

Pengujian jaringan yang menerapkan mekanisme QoS dan jaringan yang tidak menerapkan QoS, semua skenario pengujian jaringan dijalankan pada *mininet* dan pembangkit trafik yang digunakan adalah D-ITG serta merekam nilai yang menjadi parameter yang diukur oleh QoS seperti *delay*, *jitter*, dan *packet loss*. Skenario pengujian dengan menjalankan secara bersamaan 3 jenis trafik yang terdiri dari paket udp dengan port acak, paket RTP dan paket SIP selama 60 detik dari salah satu *node* sebagai penerima paket adalah h1 sedangkan untuk pengirim paket adalah h2, dan dilibatkan juga *background traffic* menggunakan *iperf* dengan kecepatan konstan 100Mbps dengan pengambilan data berulang-ulang sebanyak 5 kali.

D. Pengujian Troughput

Pengujian *troughput* ini diuji menggunakan *iperf* dengan h1 sebagai penerima paket dan h2 sebagai pengirim paket, proses ujicoba *troughput* untuk mengukur dari antrian tidak prioritas (*queue*:0) dan antrian prioritas (*queue*:1), dengan skenario pertama dengan mengukur masing-masing antrian dengan kecepatan 100Mbps dari pengirim paket selama 10 detik dan diuji sebanyak 3 kali. Skenario kedua adalah dua jenis trafik mengirim secara bersamaan kecepatan 100Mbps selama 10 detik dan diuji sebanyak 3 kali, pada dua jenis trafik terdiri dari paket udp dengan tujuan port asal yang akan masuk dalam antrian tidak prioritas dan paket udp dengan tujuan port RTP atau SIP yang masuk dalam antrian prioritas.

Percobaan pada skenario pertama hasil yang diperoleh untuk antrian non dan prioritas sebagai berikut :



Gambar 7. Pengujian skenario untuk antrian non prioritas

Gambar 8. Pengujian skenario untuk antrian prioritas

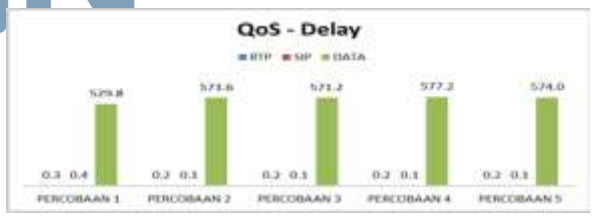
Dalam kondisi jaringan yang di simulasikan pada *mininet* setiap perangkat pada *interfaces* dengan kecepatan 100Mbps, tetapi *throughput* yang dapat hasilkan seperti Gambar 8, dengan kecepatan rata-rata 60 Mbps hingga 75 Mbps.

E. Pengujian Delay

Pada pengujian ini dilakukan dengan mengukur nilai *delay* pada jaringan tanpa QoS dan pada jaringan yang menerapkan QoS dilakukan dengan skenario yang telah dijelaskan pada bagian skenario pengujian QoS.

Gambar 9. Hasil uji delay pada jaringan tanpa QoS.

Hasil pengujian terlihat bahwa pengukuran nilai *delay* di jaringan tanpa QoS, ketiga jenis trafik menunjukan nilai *delay* yang diperoleh kurang lebih mendekati sama, jika dengan trafik variasi banyak tentu nilai *delay* yang dihasil terus meningkat sehingga berdampak terhadap paket RTP maupun SIP dan dapat menyebabkan pengguna VoIP terganggu dengan keterlambatan paket sampai tujuan karena *delay* yang terlalu tinggi, dengan mengacu pada standar yang direkomendasi ITU-T.G.1010 dengan skenario pengujian kondisi jaringan *delay* mencapai 249 ms yang tidak memenuhi rekomendasi kurang dari 150 ms.



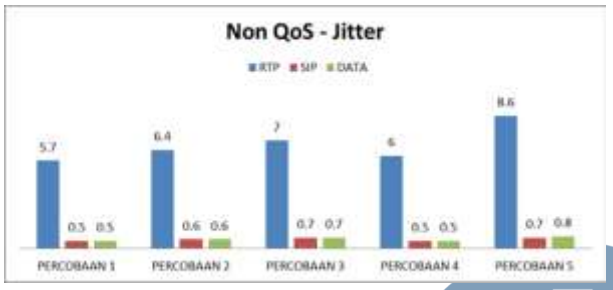
Gambar 10. Hasil uji delay pada jaringan diterapkan QoS.

Berdasarkan hasil pengujian diatas, terlihat bahwa pengukuran nilai *delay* pada jaringan yang menerapkan QoS menunjukan bahwa dengan trafik dengan protokol RTP dan SIP tidak mengalami *delay* yang berarti dengan kondisi terdapat trafik background yang berjalan sebesar 100 Mbps, karena dengan antrian prioritas yang dapat menjamin bahwa paket yang masuk dalam antrian tersebut, tidak bersaing dengan paket lain di antrian non prioritas, jika dengan kondisi bertambah variasi trafik yang bukan merupakan trafik VoIP. Nilai *delay* dari trafik VoIP juga masih dalam kondisi *delay* yang kecil,

berbeda dengan kondisi jika jaringan tanpa QoS yang tidak ada jaminan *bandwidth* untuk trafik VoIP. Sehingga dengan hasil pengujian seperti Gambar 10 untuk trafik VoIP memenuhi standar ITU-T.G.1010 dengan nilai kurang dari 150ms.

F. Pengujian Jitter

Pada pengujian ini dilakukan dengan mengukur nilai *jitter* pada jaringan tanpa QoS dan pada jaringan yang menerapkan QoS dilakukan dengan skenario yang telah dijelaskan pada bagian skenario pengujian QoS.



Gambar 11. Hasil uji jitter pada jaringan tanpa QoS.

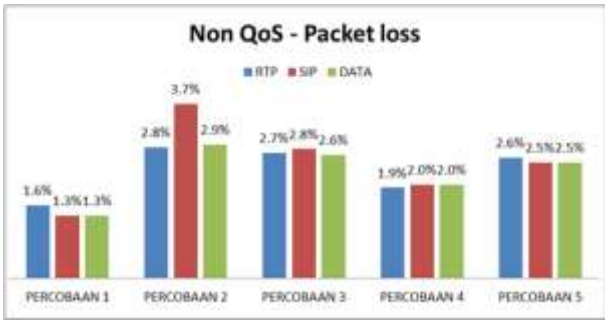
Hasil pengukuran terlihat bahwa nilai *jitter* pada trafik RTP memperoleh nilai yang bervariasi mulai dari 5,7 ms hingga 8,6 ms dan trafik SIP dan trafik data relatif lebih kecil. Besarnya nilai *jitter* dapat dipengaruhi oleh beban trafik dan kongesti antar paket dalam jaringan, dengan nilai *jitter* pada trafik RTP tidak memenuhi standar ITU-T.G.1010 dengan batas kurang dari 1 ms sedangkan untuk trafik SIP masih dalam kondisi memenuhi standar.

Gambar 12. Hasil uji jitter pada jaringan diterapkan QoS

Dengan hasil pengukuran seperti Gambar 12, nilai *jitter* pada trafik RTP lebih terkendali dibandingan dengan pengujian sebelumnya pada jaringan tanpa QoS pada Gambar 4.13, nilai *jitter* yang dihasil pada pengujian di jaringan QoS pada trafik RTP mencapai 0,25 ms dan pada trafik SIP sebesar 0,06 ms. Tentu dengan nilai *jitter* yang terkendali dapat menjaga kualitas VoIP dengan baik dan dari hasil pengukuran diatas dengan skenario yang diuji pada trafik RTP dan SIP memenuhi batas aman yang direkomendasikan oleh ITU-T.G.1010 yaitu kurang dari 1 ms.

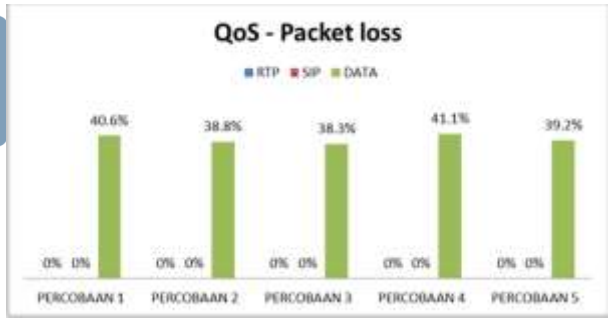
G. Pengujian Packet loss

Pada pengujian ini dilakukan dengan mengukur nilai *packet loss* pada jaringan tanpa QoS dan pada jaringan yang menerapkan QoS dilakukan dengan skenario yang telah dijelaskan pada bagian skenario pengujian QoS.



Gambar 13. Hasil uji packet loss pada jaringan tanpa QoS.

Dengan hasil pengujian dilakukan terlihat bahwa pada jaringan tanpa QoS dalam beberapa percobaan terjadi *packet loss* pada semua trafik mencapai 3,7%. *Packet loss* terjadi dikarenakan kondisi dari beberapa paket tidak terkirim dari *host* sumber menuju *host* tujuan karena terlalu banyak antrian didalam jaringan tersebut, dengan nilai *packet loss* untuk trafik RTP dan SIP terdapat beberapa percobaan masih memenuhi standar ITU-T.G.1010 dengan nilai batas rekomendasi kurang dari 3% *packet loss*, tetapi masih memungkinkan untuk terjadi *packet loss* yang lebih besar seperti pada percobaan kedua terdapat salah satu trafik melebihi dari 3% *packet loss.*



Gambar 14. Hasil uji packet loss pada jaringan QoS.

Pada hasil pengujian yang terlihat bahwa nilai *packet loss* untuk trafik RTP dan SIP dalam percobaan sebanyak lima kali terbukti mampu menjaga trafik RTP maupun SIP terjadi *packet loss*, sedangkan pada trafik data terjadi *packet loss* yang cukup besar mencapai 41,1% dikarenakan ada *background* trafik yang berjalan dalam antrian sama dengan trafik data, dari hasil pengujian untuk trafik RTP dan SIP masih dalam kondisi batas aman standar dari ITU-T.G.1010 dengan *packet loss* yang kurang dari 3%.

V. KESIMPULAN

Hasil pengukuran *troughput* pada jaringan yang disimulasikan dengan mininet hanya mampu mencapai kecepatan rata-rata 65 Mbps dengan masing-masing *interfaces switch* dibatasi dengan 100 Mbps.

Hasil pengukuran parameter delay terbukti memenuhi standar ITU-T G.1010 dalam jaringan yang menerapkan QoS.Sedangkan jaringan tanpa QoS tidak memenuhi standar karena delay melebihi 150 ms, hal ini memberikan pengaruh yang cukup signifikan terhadap trafik RTP maupun SIP.

Untuk hasil pengukuran dari parameter jitter pada jaringan tanpa QoS juga tidak dapat memenuhi standar ITU-T G.1010 untuk trafik RTP karena jitter melebihi batas dari standar yaitu 1 ms, sedangkan pada jaringan yang menerapkan QoS memenuhi standar ITU-T G.1010 untuk semua trafik RTP dan SIP dengan nilai jitter lebih kecil dari 1 ms.

Kemudian hasil pengukuran dari parameter packet loss pada jaringan tanpa QoS dan pada jaringan dengan QoS memenuhi standar ITU-T G.1010 dengan batas packet loss kurang dari 3%, tetapi dengan QoS yang diterapkan dapat memberikan jaminan packet loss yang lebih kecil sampai tidak terjadi packet loss.

DAFTAR PUSTAKA

[1] P. Ketut Sudiarta, G. Sukadarmika, P. Ketut Sudiarta, and G. Sukadarmika, "Penerapan Teknologi VoIP Mengoptimalkan Penggunaan Jaringan," *Majalah Ilmiah Teknik Elektro*, vol. 8, Dec. 2009.

[2] K. K. Tam and H. L. Goh, "Session Initiation Protocol," 2002, pp. 1310–1314.

[3] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne, "RTP: A Transport Protocol for Real-Time Applications." [Online]. Available: https://tools.ietf.org/html/rfc3550#page-13. [Accessed: 12-Jul-2018].

[4] Allied Telesis, "Quality of Services and the mechanisms QoS uses to manage congestion." .

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," p. 89.

[6] "What is SIP?" [Online]. Available: https://askozia.com/voip/what-is-sip/. [Accessed: 13-Jul-2018].

[7] ONF, "Software-Defined Networking : The New Norm for Networks." [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. [Accessed: 19-Mar-2018].

[8] Open Networking Foundation, "OpenFlow Switch Specification." [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf.

[9] "Mininet Overview - Mininet." [Online]. Available: http://mininet.org/overview/. [Accessed: 19-Mar-2018].

[10] "What is Ryu Controller?," *SDxCentral*. [Online]. Available: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/. [Accessed: 19-Mar-2018].

[11] J. L. Valenzuela, A. Monleon, I. San Esteban, M. Portoles, and O. Sallent, "A hierarchical token bucket algorithm to enhance QoS in IEEE 802.11:proposal, implementation and evaluation," 2004, vol. 4, pp. 2659–2662.