Analisis Peforma Multimedia *Streaming* Menggunakan *Clustering Controller* Pada *Software Defined Network*

Fikri Ubaidillah[1], I Made Suartana[2]

[1,2] Teknik Informatika, Jurusan Teknik Informatika, Fakultas Teknik
Universitas Negeri Surabaya
[1]fikri.17051204061@mhs.unesa.ac.id
[2]imadesuartana@unesa.ac.id

Abstrak— Pesatnya penggunaan aplikasi multimedia juga perlu diiringi peningkatan infrastruktur untuk meningkatkan kualitas layanan yang diberikan. Untuk menyediakan QoS aplikasi multimedia *streaming*, arsitektur baru organisasi jaringan komputer diusulkan, yang disebut *software defined network* (SDN). Teknologi SDN menawarkan pendekatan terpusat untuk manajemen *dataflow* di jaringan komputer, berdasarkan pemisahan *data plane* dan *control plane*. Kelebihan SDN dibanding dengan jaringan konvensional yaitu, *programmable controller* yang mengatur jaringan secara terpusat. Beberapa *controller* yang ada. Pada penelitian ini dilakukan analisis peforma Multimedia Streaming pada arsitektur SDN dengan digunakan konsep *clustering-controller* yang nantinya akan dibandingkan dengan konsep *single-controller*, dengan digunakan 3 controller, 1 sebagai master controller dan 2 sebagai slave controller, dengan begitu beban jaringan dapat dibagi pada 3 controller. Uji coba dilakukan simulasi dengan *mininet* sebagai *emulator* topologi jaringan dari SDN dan digunakan *controller* Opendaylight dengan dilakukan Multimedia Streaming dari hasil pengujian akan dilihat dari *end-to-end* QoS dengan melihat parameter *throughput, packet loss, delay, dan jitter*. Untuk skenario uji digunakan topologi dengan kualitas video 360p dan 720p dengan jumlah host, 3 host dan juga 6 host. Disetiap skenario dilakukan uji sebanyak 2 kali kemudian diambil rata-rata hasilnya. Dari pengujian yang dilakukan didapatkan hasil untuk parameter *throughput* pada konsep *clustering-controller* mendapatkan nilai yang lebih besar dari konsep *single-controller* yang berarti *clustering-controller* lebih baik, untuk parameter *packet loss* didapatkan hasil untuk konsep *clustering controller* mendapat nilai yang lebih kecil dari konsep *single-controller* yang berarti *clustering-controller* lebih baik, untuk paramtere *delay* mendapatkan hasil dari konsep *clustering-controller* mendapatkan nilai yang lebih kecil disetiap uji dari konsep *single-controller* yang berarti *clustering lebih baik*, untuk parameter *jitter* juga mendapatkan hasil untuk konsep *clustering-controller* mendapat nilai yang lebih kecil dari konsep *single-controller* disetiap ujinya. Dari semua skenario pengujian yang telah dilakukan, maka dapat dihasilkan bahwa penggunaan Multimedia Streaming dalam pengujian *Broadcast* streaming atau *Live* streaming untuk diterapkan pada teknologi *clustering-controller* pada arsitektur *Software Defined Network* bisa memberikan performa jaringan yang lebih baik dari penerapan teknologi *single-controller* dengan semua parameter pengujian yang telah dilakukan.

Kata Kunci—SDN, Multimedia Streaming, Clustering-Controller, Mininet, Opendaylight.

I. PENDAHULUAN

Pesatnya pengguna aplikasi Multimedia juga perlu diiringi peningkatan infrastruktur untuk meningkatkan kualitas layanan yang akan diberikan. Apalagi sekarang ini layanan multimedia sering dijumpai dalam berbagai bidang salah satunya bidang pendidikan, dalam bidang Pendidikan multimedia streaming digunakan untuk menunjang proses belajar mengajar misalnya: Aplikasi Multimedia Interaktif untuk proses belajar mengajar [5]. Untuk mendapatkan QoS di dalam multimedia streaming dibutuhkan arsitektur jaringan yang baik, pada saat ini telah muncul konsep teknologi arsitektur jaringan baru yaitu SDN (*Software Defined Network*), dengan digunakan SDN kualitas layanan yang disediakan oleh multimedia streaming dapat meningkat[3]. *Controller* menjadi faktor terpenting dalam peforma sebuah jaringan SDN baik ataupun buruk, dikarenakan apabila *controller* mengalamai penurunan peforma atau *error* jaringan dibawahnya juga pasti mengalami kendala, sehingga dibutuhkannya *multi-controller* yang saling terhubung sehingga jika salah satu *controller* terganggu *controller* yang lain akan menjadi *backup* sehingga jaringan masih bisa tetap berjalan [4].

Untuk menyediakan QoS aplikasi multimedia *streaming*, arsitektur baru organisasi jaringan komputer diusulkan, yang disebut *software defined network* (SDN). Teknologi SDN menawarkan pendekatan terpusat untuk manajemen *dataflow* di jaringan komputer, berdasarkan pemisahan *data plane* dan *control plane* [1]. *Programmable controller* menjadi kelebihan SDN dari pada jaringan konvensional, dimana jaringan diatur oleh *controller* secara terpusat. Beberapa *controller* yang ada Seperti adalah Floodlight, Maestro, RYU, Opendaylight, POX, dan ONOS [2].

Pada penelitian yang telah dilakukan oleh [3] yaitu Analisis Perancangan Multimedia Streaming Berbasis Software Defined Network. Pada penelitian ini, peneliti membuat uji coba Multimedia Streaming dengan digunakan SDN sebagai arsitektur jaringannya. Ada beberapa skenario uji coba dengan penggunaan jumlah host yang berbeda. Switch dan Host dengan througput tautan/link penghubung 10 Mbps di antara switch. 2 aliran video (240p dan 480p) dikirim bersamaan melewati jaringan. Jaringan dimulai dengan 4 host, 8 host, dan 12 host dengan 2 aliran/flow, 4 aliran/flow, dan 6 aliran/flow secara simultan dengan digunakan emulator Mininet dengan POX openflow controller. Hasil uji coba awal dilakukan streaming video pada jaringan, aplikasi VLC digunakan sebagai media atau streaming server dan sekaligus sebagai pengguna. RTSP (Realtime Transport Streaming Protocol) digunakan sebagai Protokol transport. Hasil Percobaan dilakukan untuk mengamati hasil simulasi sesuai parameter yaitu, Throughput, dan Delay. digunakan dua aliran video yang digunakan dalam format mp4 dan berukuran 720x480p dan 360x240p di semua simulasi. Durasi video 720x480p adalah 27 detik, dan video 360x240p adalah 31 detik. Menghasilkan kesimpulan penambahan jumlah host dengan topologi jaringan

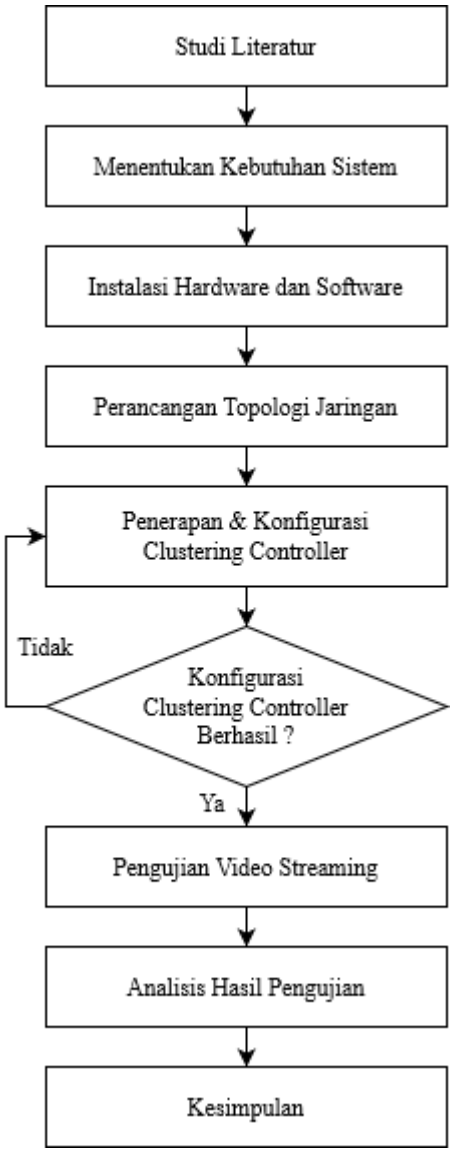
yang sama menyebabkan nilai throughput yang sesemakin kecil. Nilai Througput dapat ditingkatkan dengan menerapkan QoS pada jaringan. Penambahan jumlah host dengan topologi jaringan yang sama menyebabkan nilai delay yang sesemakin besar. Nilai delay dapat dikurangi dengan menerapkan QoS pada jaringan[3].

Pada penelitian selanjutnya dilakukan oleh [4] yaitu tentang Performa Clustering Controller pada Arsitektur Software Defined Network. Peneliti mencoba mengetahui peforma SDN jika digunakan *multi-controller* yang saling terhubung atau bisa disebut *Clustering Controller* . Uji coba dilakukan dengan membuat topologi *clustering-controller* yang terdiri dari 3 *controller*, 1 *controller* sebagai utama dan 2 *controller* sebagai *controller* pembantu dengan 9 *switch* yang masing-masingnya terhubung dengan *master controller* dan setiap *switch* membawahi 10 *host*, pengujian bertujuan untuk melihat hasil dari *throughput, delay, jitter* dan *packet loss* dengan mengirim paket UDP dengan berjumah 25 paket yang variasi ukuran paket dimulai dari 100Mb, 1Gb, 5Gb, 10Gb, dan 15Gb. Paket dikirimkan selama waktu 10 detik kemudian pengujian dilakukan 10 kali disetiap variasi ukuran paketnya. Dari hasil pengujian *throughput* yang dihasilkan dari *clustering controller* lebih besar dari pada digunakan *controller* tanpa *clustering*, sedangkan untuk *delay, jitter,* dan *packet loss* digunakan *clustering controller* lebih cepat dan besar dari *controller* tanpa *clustering*. Hasil dari penelitian ini yaitu performa *controller* tanpa *clustering* lebih buruk daripada *clustering controller* [4].

Penelitian ini penulis ingin menganalisis tentang peforma multimedia streaming digunakan *clustering controller* pada *software defined network*. Penelitian ini akan digunakan simulasi pengujian, dengan digunakan *mininet* sebagai *emulator* SDN dan untuk *clustering controller* peneliti digunakan *Opendaylight* sebagai *controller*. Untuk simulasi video di multimedia streaming salah satu *host* akan menjadi server yang akan dilakukan streaming video, kemudian beberapa *host* yang menjadi client akan mengakses video yang ada di dalam *host server*. Untuk kualitas video digunakan 480p dan 720p. Peforma multimedia streaming digunakan *clustering controller* ini akan dilihat dari parameter QoS *throughput, delay, jitter* dan *packet loss*. Dengan ini diharapkan dapat meningkatkan peforma multimedia streaming pada *software defined network*.

II. METODE PENELITIAN

Alur penelitian dalam Analisis Peforma Multimedia *Streaming* digunakan *clustering-controller* pada SDN, dikembangkan digunakan alur atau tahapan-tahapan simulasi seperti pada Gambar. 1.Seperti yang dijelaskan pada Gambar.1 Studi literatur terkait penelitian dilakukan untuk mengumpulkan bahan dan teori yang mendukung dalam penelitian ini dan mempelajari penelitian sebelumnya terkait peforma *clustering controller* pada arsitektur SDN dan Analisis perancangan multimedia *streaming* berbasis SDN.



Gambar. 1 Alur Penelitian

Tahapan penelitian selanjutnya adalah mengkombinasikan keduanya. Permasalahan pada penelitian ini adalah bagaimana peforma jika dilakukannya multimedia streaming digunakan *clustering controller* pada jaringan SDN. Selanjutnya menentukan bahan-bahan perangkat yang akan digunakan dalam penelitian ini baik perangkat keras maupun perangkat lunak. Perangkat keras yang digunakan yaitu empat *virtual* komputer (satu digunakan untuk menjalankan simulasi jaringan dan tiga digunakan sebagai *controller*). Perangkat lunak yang digunakan yaitu system operasi Ubuntu, *emulator mininet* sebagai *software* emulasi topologi jaringan pada SDN, Opendaylight *controller*, *Wireshark Network Analizer Tools* untuk pengujian parameter *Quality of Service* (QoS) di penelitian ini, VLC sebagai media atau video streaming. Tahapan selanjutnya yaitu dilakukan instalasi *hardware* dan *software* dengan benar. Selanjutnya membuat perancangan topologi jaringan yang akan digunakan untuk penelitian ini, Setelah itu dilakukan implementasi dan konfigurasi dari *clustering controller* pada tiga *controller* yang sudah di buat.

Setelah berhasil dilakukan penerapan dan konfigurasi *clustering controller* selanjutnya dilakukan pengujian digunakan multimedia *streaming* digunakan VLC sebagai *software* multimedia *streaming* untuk pengujian peforma digunakan *Quality of Service* (QoS) dengan parameter pengujian diantaranya *throughput, packet loss, delay,* dan *jitter* digunakan standar TIPHON. Dari pengujian dan analisis data tersebut akan ditarik kesimpulan dari penelitian yang sudah dikerjakan.

A. *Arsitektur Sistem*

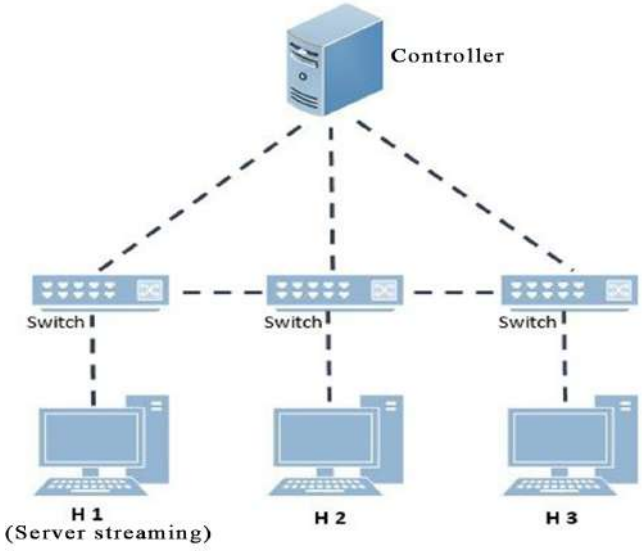
Pada arsitektur *single-controller*, jaringan terpusat pada satu *controller.* Sedangkan untuk arsitektur *clustering-controller, controller* satu dengan *controller* lainnya saling terhubung dan bertukar data untuk menangani beban jaringan. Pada penelitian ini digunakan tiga *controller* dikarenakan pada controller Opendaylight minimal *clustering-controller* harus digunakan tiga buah *controller,* karena apabila kurang dari tiga controller fitur clustering masih bisa berjalan tapi apabila salah satu controller mengalami down atau gangguan maka controller yang lainya tidak akan berjalan[6], untuk ilustrasi bisa dilihat pada Gambar.3.

Untuk bisa dilakukan *clustering controller* di Opendaylight penerapannya berjalan digunakan arsitektur *akka.conf*. Akka memiliki fungsi sebagai jembatan untuk komunikasi data antar controller berdasarkan konfigurasi pada script *akka.conf* di Opendaylight, adapun juga *module-shards.conf* untuk mengenalkan nama atau *alias* setiap node dari *controller*.

B. *Perancangan Topologi*

Empat jenis topologi digunakan untuk penelitian MultiMedia Streaming yang dibangun digunakan Mininet yakni topologi jaringan *single-controller* 3 *host* dan 4 *host*, *clustering-controller* 3 *host* dan 4 *host*. Berikut ini topologi jaringannya.

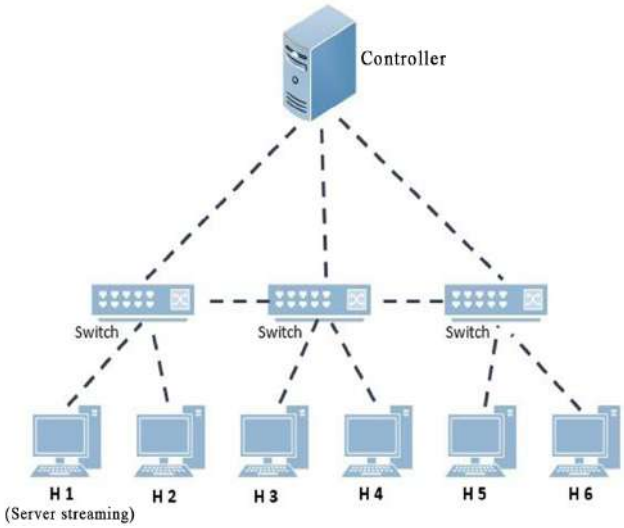
1) Topologi *Single-Controller* 3 host:



Gambar. 2 Topologi *single-controller* 3 *host*

Pada Gambar. 2 merupakan topologi *single-controller* 3 *host*. Terdiri dari 1 *controller,* 3 *switch* yang terhubung dengan *controller*, masing-masing *swtich* saling terhubung dan setiap *switch* ada 1 *host* yang terhubung, untuk pengujian H1 sebagai *server* sedangkan H2 dan H3 sebagai *client*.

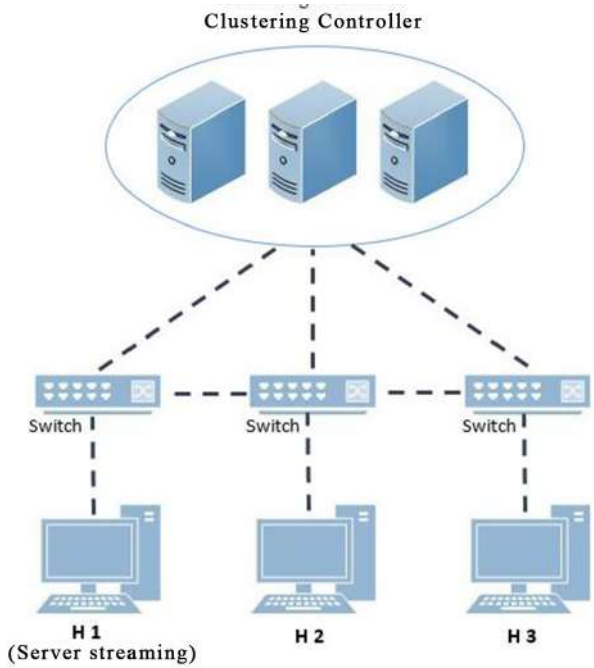
2) Topologi *Single-Controller* 6 host:



Gambar. 3 Topologi *single-controller* 6 *host*

Pada Gambar. 3 merupakan topologi *single-controller* 6 *host*. Terdiri dari 1 *controller*, 3 *switch* yang terhubung dengan *controller*, masing-masing *swtich* saling terhubung dan setiap *switch* ada 2 *host* yang terhubung, untuk pengujian H1 sebagai *server* sedangkan H2, H3, H4, H5 dan H6 sebagai *client*.

3) Topologi *Clustering-Controller* 3 host:

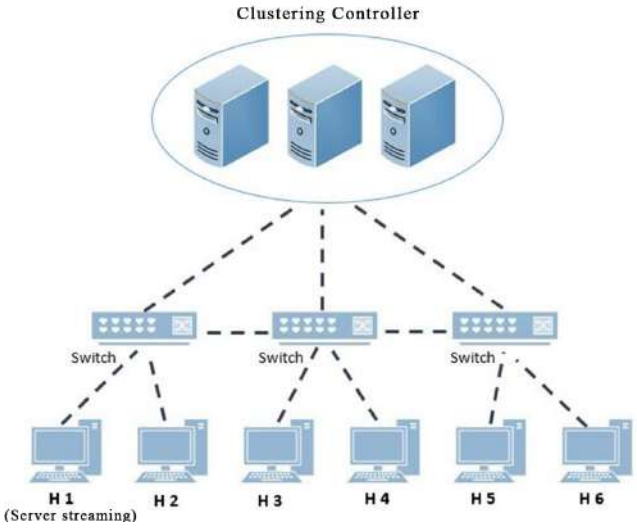


Gambar. 4 Topologi *clustering-controller* 3 *host*

Pada Gambar. 4 merupakan topologi *clustering-controller* 3 *host*. Terdiri dari 3 *controller* yang saling terhubung 1 sebagai *controller* utama dan 2 sebagai *controller* pembantu, 3 *switch*

yang terhubung dengan *controller* utama, masing-masing *swtich* saling terhubung dan setiap *switch* ada 1 *host* yang terhubung, untuk pengujian H1 sebagai *server* sedangkan H2 dan H3 sebagai *client*.

4) Topologi *Clustering-Controller* 6 host:



Gambar. 5 Topologi *clustering-controller* 3 *host*

Pada Gambar. 5 merupakan topologi *clustering-controller* 3 *host*, Terdiri dari 3 *controller* yang saling terhubung. 1 sebagai *controller* utama dan 2 sebagai *controller* pembantu, 3 *switch* yang terhubung dengan *controller* utama, masing-masing *swtich* saling terhubung dan setiap *switch* ada 2 *host* yang terhubung, untuk pengujian H1 sebagai *server* sedangkan H2, H3, H4, H5 dan H6 sebagai *client*.

C. *Skenario Pengujian*

Dalam pengujian ini dilakukan Multimedia Streaming pada topologi yang sudah disiapkan, ada 4 uji coba yang akan dilakukan untuk dapat melihat peforma Multimedia Streaming dalam arsitektur jaringan SDN. Untuk mendapatkan data dari dilakukan Multimedia Streaming kita digunakan *wireshark*. Untuk dapat mendapatkan hasil peforma yang akan diujikan dapat dilihat dari parameter yang akan ditentukan, adapaun parameter yang akan digunakan yaitu *throughput*, *packet loss*, *delay*, dan *jitter*. Berikut penjelasan dari parameter yang akan digunakan:

1) *Throughput*: *Throughput* adalah nilai kecepatan pengiriman data dengan satuan bit per second (bps) yang dihitung berdasarkan keseluruhan packet yang berhasil diterima pada jangka waktu tertentu lalu dibagi dengan jangka waktu tersebut[7] Rumus untuk pengukuran *throughput* adalah sebagai berikut:

$$Throughput = \frac{Jumlah\ data\ yang\ dikirim}{waktu\ pengiriman\ data} \quad (1)$$

2) *Packet Loss: Packet loss* adalah parameter yang menunjukkan jumlah keseluruhan paket yang tidak terkirim atau hilang. Paket yang tidak terkirim atau hilang bisa terjadi karena *collision* dan *congestion* pada suatu jaringan. Rumus untuk pengukuran *packet loss* adalah sebagai berikut[8]:

$$PL = \left(\frac{paket\ dikirim - paket\ diterima}{paket\ data\ dikirim}\right) \times 100\% \quad (2)$$

Adapun standar *Packet Loss* menurut TIPHON [9] dapat dilihat pada Tabel I.

TABEL I
STANDAR TIPHON *PACKET LOSS*

Kategori *Packet Loss*	*Packet Loss*
Sangat Bagus	$0 - 2\%$
Bagus	$3 - 14\%$
Sedang	$15 - 24\%$
Jelek	$> 25\%$

3) *Delay* adalah waktu yang diperlukan data untuk melintasi jarak dari asal ke tujuan. *Delay* bisa terpengaruh oleh jarak, media fisik, kongesti atau juga proses yang lama. Rumus untuk pengukuran *delay* adalah sebagai berikut[8]:

$$Delay = \frac{total\ delay}{total\ paket\ yang\ diterima} \quad (3)$$

Adapun standar *Delay* menurut TIPHON [9] dapat dilihat pada Tabel II.

TABEL II
STANDAR TIPHON *DELAY*

Kategori *Delay*	*Delay*
Sangat Bagus	$< 150\ ms$
Bagus	$150\ ms\ s/d\ 300\ ms$
Sedang	$300\ ms\ s/d\ 450\ ms$
Jelek	$> 450\ ms$

4) *Jitter: Jitter* bisa diartikan sebagai gangguan pada komunikasi digital maupun analog yang dipicu oleh berubahnya sinyal karena referensi posisi waktu. *Jitter* dapat mengakibatkan hilangnya data, terutama pada pengiriman data dengan kecepatan tinggi. Variasi *jitter* biasanya berkaitan erat dengan *delay*, *delay* menunjukkan variasi yang cukup besar dalam transmisi data pada jaringan. Rumus untuk pengukuran *jitter* adalah sebagai berikut[8]:

$$Jitter = \frac{total\ variasi\ delay}{total\ paket\ yang\ diterima} \quad (4)$$

Adapun standar *Jitter* menurut TIPHON [9] dapat dilihat pada Tabel III.

TABEL III
STANDAR TIPHON *JITTER*

Kategori *Jitter*	*Jitter*
Sangat Bagus	$0\ ms$
Bagus	$0\ ms\ s/d\ 75\ ms$

Sedang	$75\ ms\ s/d\ 125\ ms$
Jelek	$125\ ms\ s/d\ 225\ ms$

Pengujian Multimedia Streaming dilakukan dengan cara *host* yang ada pada topologi yang sudah dibuat, 1 host akan buat sebagai *server* yang nantinya akan dilakukan streming video, untuk videonya ada 2 macam yaitu 360p dan 720p setelah itu server akan membuka akses agar video dapat diakses atau dilihat pada *client* digunakan *via* RTP Transport Stream jadi *server* akan mendaftarkan *ip client* dengan *port* 5004, kemudian *host* yang lainnya akan menjadi *client* yang nantinya akan mengakses video streaming yang dilakukan *server*. Jadi masing-masing *client* harus dilakukan *Open Network Stream via* RTP pada *port* 5004 agar dapat terhubung, untuk streaming bersifat *broadcast* streaming atau *live streaming* sehingga *server* dan *client* akan menampilkan gambar dan video secara bersamaan diwaktu itu juga. Sebagai media Multimedia Streaming digunakan VLC Media Player. Untuk ujicoba dilakukan di setiap topologi yang sudah dibuat termasuk topologi 3 host dan juga 6 host, ujicoba setiap topologi dilakukan sebanyak 2 kali kemudian akan diambil rata-rata dari hasil ujicoba tersebut. Hasil ujicoba akan dibandingkan antara skenario *single-controller* dan juga *clustering controller*.

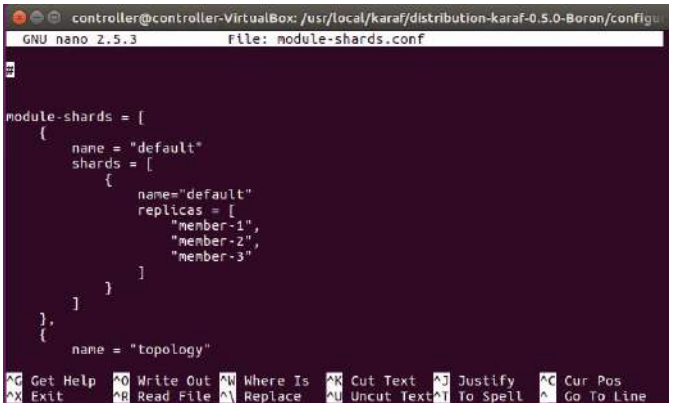
III. Hasil dan Pembahasan

A. *Hasil Implementasi*

Pada Gambar. 6 merupakan *script* dari *akka.conf* . *Akka.conf* merupakan penerapan berjalannya *Clustering Controller* di Opendaylight, scrip *akka.conf* disetting di masing-masing *controller* yang digunakan untuk mengkonfigurasi *clustering-controller* dan untuk penjelasannya *hostname* adalah *ip address* dari *controller* tersebut setiap *controller* memiliki *ip address* yang berbeda, kemudian untuk *port* 2550 merupakan *port* yang disediakan di Opendaylight yang digunakan untuk konfigurasi *clustering-controller*, selanjutnya *seed-nodes* merupakan alamat *ip address* dari semua *controller* yang digunakan yang menghubungkan antara *controller*, untuk baris 1 pada *seed-nodes* merupakan *ip address* dari *controller* yang sedang dilakukan konfigurasi dan untuk baris 2 dan 3 adalah *ip address* dari *controller* yang lainnya, kemudian *roles* adalah inisial dari masing-masing *controller,* untuk *roles* harus berbeda untuk setiap *controller*. Setelah selesai dilakukan konfigurasi semua *controller* dapat terhubung satu sama lain dan juga dapat bertukar data.

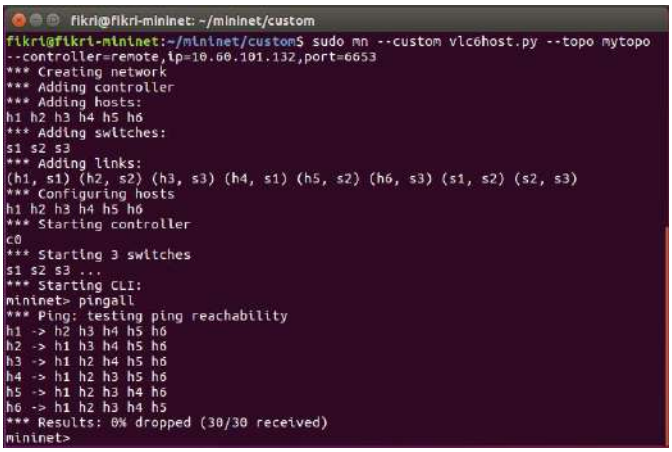
Gambar. 6 *Script akka.conf*

Pada Gambar. 7 merupakan *script* dari *module-shared*, untuk default dari *script module-shared* replicas yang berisi *roles* dari setiap *controller* cuma terdapat 1 yaitu "member-1", jadi untuk bisa menjalankan *clustering-controller* kita tambahkan anggota disetiap *replicas* yang ada sesuai dengan *roles* yang dibuat di *script akka.conf*.



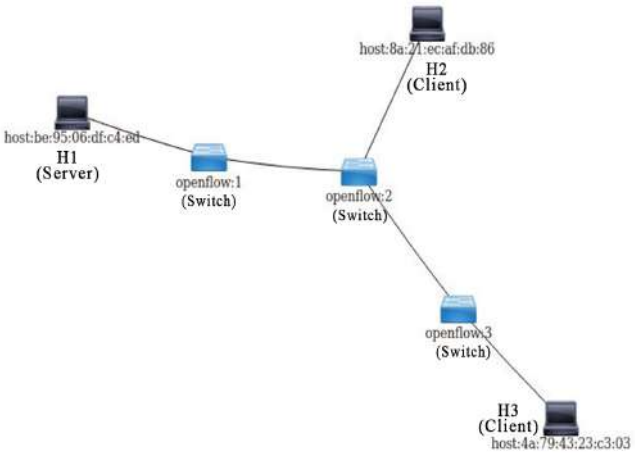
Gambar. 7 *Script module-shared.conf*

Setelah dilakukan setting pada *akka.conf* dan juga *module-shared*, dilakukannya pengecekkan untuk konfigurasi *clustering-controller* berhasil atau tidak, untuk pengecekkan dilakukan dengan menjalankan topologi yang sudah dibuat dimininet kemudian disambungkan dengan *controller* utama yang ada di *clustering-controller* seperti Gambar. 9. Setelah itu dilakukan *ping* kesemua host, jika host tersambung satu sama lain berarti menandakan *clustering-controller* sudah berjalan.

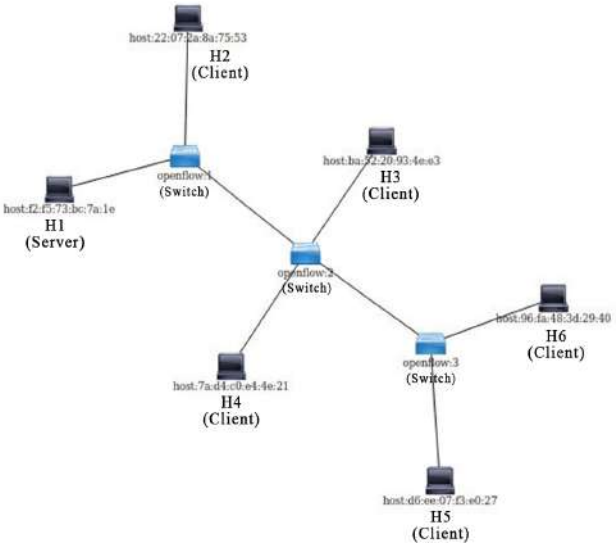


Gambar. 9 mininet terhubung dengan *clustering-controller*

Hasil Implementasi Topologi bisa dilihat pada Gambar. 10 dan Gambar. 11. Untuk gambar implementasi topologi mininet tidak dapat terlihat mirip dengan gambar rancangan topologi diatas dikarenakan dari controller Opendaylight sudah mempunyai *page* untuk dapat melihat topologi yang sudah dibuat di mininet, dan mempunyai default ketika topologi ditampilkan *controller* pada topologi tidak akan kelihatan. Untuk uji coba dilakukan dengan skenario yang sudah dibuat, dengan penggunaan host yang berbeda 3 host dan 6 host, ukuran kualitas video 360p dan 720p, skenario dijalankan pada arsitektur *single-controller* dan juga *clustering-controller*. Setiap skenario dilakukan uji sebanyak 2 kali dan diambil rata-ratanya.



Gambar. 10 Hasil Implementasi Topologi 3 host



Gambar. 11 Hasil Implementasi Topologi 6 host

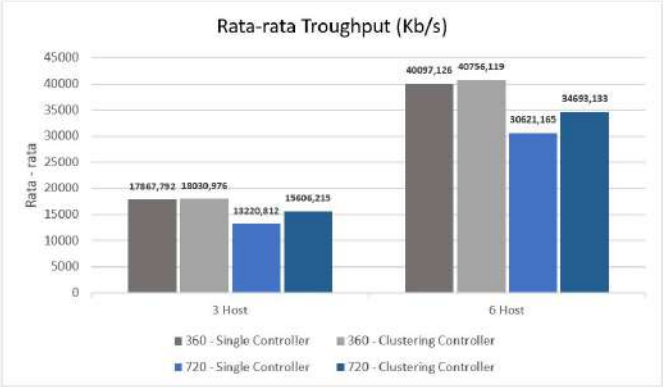
Pada Gambar. 12 adalah proses uji coba streaming, dalam uji coba streaming penulis digunakan aplikasi VLC Media Player untuk proses *Broadcast* streaming atau *Live* Multimedia Streaming juga sebagai media streaming server dan sekaligus host. Untuk terhubung antara server dengan host, digunakan protokol RTP dengan port 5004. Sehingga untuk host yang akan dilakukan streaming melalui server harus dilakukan *Open Network Stream via* RTP pada port 5004 agar dapat terhubung dan dapat mengakses atau melihat video yang sedang diputar pada server, sedangkan untuk server ketika dilakukan streaming video juga harus mendaftarkan *ip* host yang digunakan melalui protokol RTP, untuk streaming bersifat *broadcast* streaming atau *live* streaming sehingga server dan host akan menampilkan gambar video yang sama.

Gambar. 12 uji coba streaming

B. *Hasil Pengujian Throughput*

Pada Gambar. 13 adalah grafik perbandingan hasil uji coba *throughput*. Dilihat dari grafik parameter *throughput* hasil uji coba *throughput* dari 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-*

controller. Untuk lebih detail data hasil pengujian parameter *throughput* bisa dilihat pada Tabel IV.



Gambar. 13 Grafik hasil pengujian *throughput*

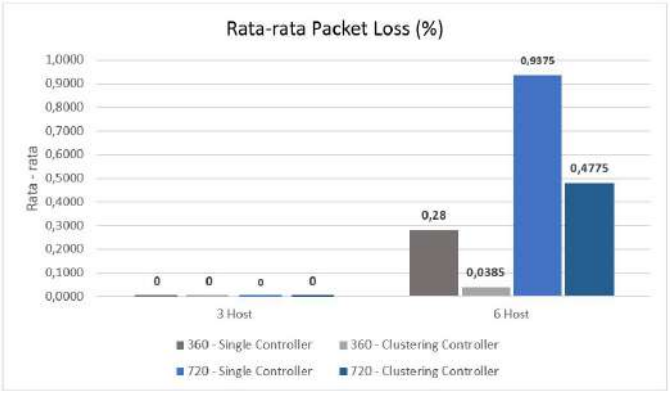
TABEL IV
HASIL PENGUJIAN *THROUGHPUT*

Throughput (Kb/s)				
Jumlah Host	**Single Controller**		**Clustering Controller**	
	360p	**720p**	**360p**	**720p**
3	17867,792	13220,812	18030,976	15606,215
6	40097,126	30621,165	40756,119	34693,133

Setelah didapatkan data tersebut maka langsung dilakukan perhitungan untuk parameter *throughput*. Dalam pengujian peforma pada parameter *throughput*, dilakukan perbandingan uji 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Dapat dilihat di Tabel IV bahwa nilai *throughput* pada arsitektur *clustering-controller* lebih besar dibandingkan pada arsitektur *single-controller* disemua ujinya dikarenakan pada *clustering-controller* mempunyai 3 *controller* yang saling berhubungan sehingga dapat meringankan beban dibanding dengan *single-controller*.

C. *Hasil Pengujian Packet Loss*

Pada Gambar. 14 adalah grafik perbandingan hasil pengujian *packet loss*. Dilihat dari grafik pengujian *packet loss* hasil pengujian dari 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Untuk lebih detail hasil pengujian parameter *packet loss* bisa dilihat pada Tabel V.



Gambar. 14 Grafik hasil pengujian *packet loss*

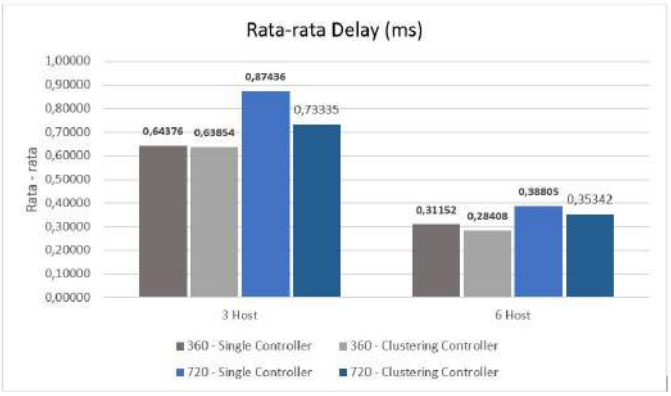
TABEL V
HASIL PENGUJIAN *PACKET LOSS*

Packet Loss (%)				
Jumlah Host	**Single Controller**		**Clustering Controller**	
	360p	**720p**	**360p**	**720p**
3	0	0	0	0
6	0,28	0,9375	0,0385	0,4775

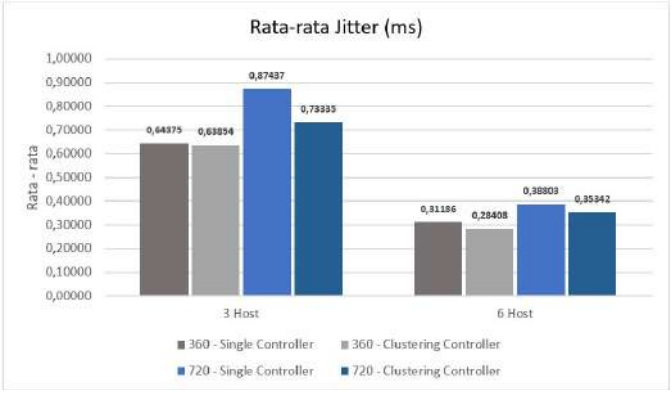
Setelah didapatkan data tersebut maka langsung dilakukan perhitungan untuk parameter *packet loss*. Dalam pengujian peforma pada parameter *packet loss*, dilakukan perbandingan uji 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Dapat dilihat di Tabel V bahwa nilai *packet loss* pada semua skenario di 3 host pada arsitektur *clustering-controller* maupun *single-controller* hasilnya 0% dikarenakan pengaruh di jumlah host dapat mengurangi *packet loss* , dan pada skenario di 6 host dengan kualitas dari video 360p maupun 720p nilai *packet loss* pada *clustering-controller* lebih kecil dibandingkan pada arsitektur *single-controller* dikarenakan pada *clustering-controller* mempunyai 3 *controller* yang saling berhubungan sehingga dapat meringankan beban dibanding dengan *single-controller*.

D. *Hasil Pengujian Delay*

Pada Gambar. 15 adalah grafik perbandingan hasil pengujian *delay*. Dilihat dari grafik pengujian *delay* dari 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Untuk lebih detail data hasil pengujian parameter *delay* bisa dilihat pada Tabel VI.



Gambar. 15 Grafik hasil pengujian *delay*



Gambar. 16 Grafik hasil pengujian *jitter*

TABEL VI
HASIL PENGUJIAN *DELAY*

Dellay (ms)				
Jumlah Host	Single Controller		Clustering Controller	
	360p	720p	360p	720p
3	0,64376	0,87436	0,63854	0,73335
6	0,31152	0,38805	0,28408	0,35342

Setelah didapatkan data tersebut maka langsung dilakukan perhitungan untuk parameter *delay*. Dalam pengujian peforma pada parameter *delay*, dilakukan perbandingan uji 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Dapat dilihat di Tabel VI bahwa nilai *delay* pada semua skenario di 3 host dengan kualitas dari video 360p maupun 720p nilai *delay* pada *clustering-controller* lebih kecil dibandingkan pada arsitektur *single-controller* begitupun juga pada skenario di 6 host dengan kualitas dari video 360p maupun 720p nilai *delay* pada *clustering-controller* lebih kecil dibandingkan pada arsitektur *single-controller* dikarenakan pada *clustering-controller* mempunyai 3 *controller* yang saling berhubungan sehingga dapat meringankan beban dibanding dengan *single-controller* sehingga dapat mengurangi waktu *delay* disetiap skenario 3 host maupun 6 host.

E. *Hasil Pengujian Jitter*

Pada Gambar. 16 adalah grafik perbandingan hasil pengujian *jitter*. Dilihat dari grafik pengujian *jitter* dari 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Untuk lebih detail data hasil pengujian parameter *jitter* bisa dilihat pada Tabel VII.

TABEL VII
HASIL PENGUJIAN *JITTER*

Jitter (ms)				
Jumlah Host	Single Controller		Clustering Controller	
	360p	720p	360p	720p
3	0,64375	0,87437	0,63854	0,73335
6	0,31186	0,38803	0,28408	0,35342

Setelah didapatkan data tersebut maka langsung dilakukan perhitungan untuk parameter *jitter*. Dalam pengujian peforma pada parameter *jitter*, dilakukan perbandingan uji 360p-3host *single-controller* dengan 360p-3host *clustering-controller*, 360p-6host *single-controller* dengan 360p-6host *clustering-controller*, 720p-3host *single-controller* dengan 720p-3host *clustering-controller*, dan 720p-6host *single-controller* dengan 720p-6host *clustering-controller*. Dapat dilihat di Tabel VII bahwa nilai *jitter* pada semua skenario di 3 host dengan kualitas dari video 360p maupun 720p nilai *jitter* pada *clustering-controller* lebih kecil dibandingkan pada arsitektur *single-controller* begitupun juga pada skenario di 6 host dengan kualitas dari video 360p maupun 720p nilai *jitter* pada *clustering-controller* lebih kecil dibandingkan pada arsitektur *single-controller* dikarenakan pada *clustering-controller* mempunyai 3 *controller* yang saling berhubungan sehingga dapat meringankan beban dibanding dengan *single-controller* sehingga dapat mengurangi *jitter* disetiap skenario 3 host maupun 6 host.

IV. KESIMPULAN

Setelah dilakukan penelitian ini didapatkan hasil bahwa pada parameter *throughput* dalam penggunaan Multimedia Streaming pada SDN, untuk hasil pada *clustering-controller* mendapatkan nilai yang lebih besar dari hasil pada *single-controller* disetiap uji dengan perbedaan kualitas video 360p dan 720p maupun perbedaan jumlah host antara 3 host dengan 6 host. Untuk parameter *packet loss* hasil yang didapat dari Multimedia Streaming pada *clustering-controller* mendapat nilai *packet loss* yang lebih kecil atau lebih baik dibanding dengan *single-controller* meskipun pada skenario uji 3 host hasilnya sama 0% antara *clustering-controller* dengan *single-controller*, tetapi untuk skenario uji yang 6 host dapat dilihat perbedaan dari nilai *packet loss* bahwa *clustering-controller* mendapat hasil yang lebih baik. Pada parameter *delay* dan *jitter*

hasil dari pengujian topologi dan skenario yang dibuat menunjukkan bahwa *clustering-controller* mendapatkan hasil yang lebih kecil dibanding dengan *single-controller*, bisa dilihat ada perbedaan 3 host dengan 6 host pada hasil dari parameter *delay* dan *jitter*, untuk nilai dari 6 host lebih kecil dibanding yang 3 host di setiap ujinya baik itu *clustering-controller* maupun *single-controller*.

Dari semua skenario pengujian yang telah dilakukan, maka dapat disimpulkan bahwa penggunaan Multimedia Streaming dalam pengujian *Broadcast* streaming atau *Live* streaming untuk diterapkan pada teknologi *clustering-controller* pada arsitektur *Software Defined Network* bisa memberikan performa jaringan yang lebih baik dari penerapan teknologi *single-controller* dengan semua parameter pengujian yang telah dilakukan.

UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kepada Allah SWT atas rahmat dan karunia-Nya serta memberikan kemudahan dan kelancaran sehingga dapat menyelesaikan penelitian ini dengan baik. Terimakasih juga penulis sampaikan kepada kedua orangtua dan keluarga yang terus memberikan semangat dan doa. Dan tak lupa terimakasih juga penulis sampaikan kepada dosen pembimbing yang sudah membimbing penelitian ini dari awal sampai akhir. Terakhir, terimakasih juga penulis sampaikan kepada seluruh pihak yang terlibat dalam penyelesaian penelitian ini.

REFERENSI

[1] Koryachko, V., Perepelkin, D., Ivanchikova, M., Byshov, V., & Tsyganov, I. (2017, June). Analysis of QoS metrics in software defined networks. In *2017 6th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1-5). IEEE.

[2] Putra, M. W., Pramukantoro, E. S., & Yahya, W. (2018). Analisis Perbandingan Performansi Kontroler Floodlight, Maestro, RYU, POX Dan ONOS Dalam Arsitektur Software Defined Network (SDN). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN*, *2548*, 964X.

[3] Suartana, I. M., & Prapanca, A. (2020, December). Analisis Perancangan Multimedia Streaming Berbasis Software Defined Network. In *Prosiding Seminar Nasional Informatika Bela Negara* (Vol. 1, pp. 76-81).

[4] Anggraini, M. A. N., & Suartana, I. M. (2020). Performa Clustering Controller pada Arsitektur Software Defined Network. *Journal of Informatics and Computer Science (JINACS)*, *2*(1).

[5] Cantos, L. C., Izquierdo, J. L., & Cantos, E. C. (2016). Interactive multimedia application for teaching and learning in Analytical Geometry. *IEEE Latin America Transactions*, *14*(7), 3461-3466.

[6] Opendaylight Documentation, "Deployment Considerations : Multiple Node Opendaylight Clustering," Opendaylight, - - 2018. [Online]. Tersedia: https://nexus.opendaylight.org /content/sites/site/org.opendaylight.docs/master/userguide/manuals/us erguide/bk-user-guide/content/_deployment_consider ations.html. [Diakses pada 19 Juni 2020].

[7] R. Wulandari, "Analisis QoS (Quality of Service) Pada Jaringan Internet (Studi Kasus : UPT Loka Uji Teknik Penambangan Jampang Kulon - LIPI)," Jurnal Teknik Informatika dan Sistem Informasi, vol. 2, no. 2, pp. 162-172, 2016.

[8] Purwahid, Muhammad., & Triloka, Joko. 2019. Analisis Quality of Service (QOS) Jaringan Internet Untuk Mendukung Rencana Strategis Infrastruktur Jaringan Komputer Di SMK N I Sukadana. *JTKSI*, Vol. 02, No. 03 September 2019. Hal. 100-109.

[9] Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS) TR 101 329 V2.1.1, 1999.