ANALISIS PERFORMA KOMUNIKASI *VIDEO CONFERENCE* PADA WEBRTC DENGAN MENGGUNAKAN *FRAMEWORK* RTCMULTICONNECTION

Lestari Sintika Syarah[1)], Adhitya Bhawiyuga, S.Kom, M.S [2)], Aswin Suharsono, S.T, M.T [3)]
[1)] *Mahasiswa*, [2)] *Dosen Pembimbing*, [3)] *Dosen Pembimbing*
Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Brawijaya
Jalan Veteran Malang 65145, Indonesia
Email: lestarisintikasyarah@gmail.com [1)], adhitya.bhawiyuga@googlemail.com [2)], aswin@ub.ac.id [3)]

ABSTRAK

WebRTC adalah sekumpulan standar dari *World Wide Web Consortium* (WC3) dan *Internet Engineering Task* Force (IETF), WebRTC dapat melakukan proses komunikasi secara *real-time* pada *browser* yang telah mendukung fitur WebRTC. WebRTC dapat digunakan untuk layanan komunikasi *Video conference* dengan jumlah lebih dari dua pengguna. Pada penelitian ini webRTC yang dibangun menggunakan *framework* RTCMultiConnection dan node.js sebagai media signalingnya. Tujuan dari penelitian ini untuk mengetahui performansi komunikasi *video conference* pada WebRTC dengan melihat nilai parameter *delay*, *jitter*, *throughput* dan *packet loss* sebagai parameter uji. Hasil yang diperoleh dari penelitian komunikasi *video conference* pada WebRTC dengan menggunakan *framework* RTCMultiConnection menunjukkan bahwa semakin banyak jumlah *client* maka semakin meningkat nilai *delay, jitter, throughput* dan *packet loss* yang diperoleh. Dalam proses komunikasi dengan menggunakan minimum *bandwidth* sebesar 58 kbps dapat disimpulkan bahwa nilai parameter *delay* dan *jitter* dapat dikategorikan pada kategori baik menurut versi TIPHON, namun pada nilai parameter *throughput* dapat dikategorikan pada kategori kurang baik dan nilai *packets loss* masuk pada kategori sedang menurut versi TIPHON.

Kata Kunci: WebRTC, RTCMultiConnection, *Video conference*, Node.Js, TIPHON.

ABSTRACT

WebRTC is standards from the World Wide Web Consortium (WC3) and Internet Engineering Task Force (IETF), WebRTC can be process real-time communication in Browser has supports WebRTC. WebRTC can be used for video conference communication services with more than users. In this research, WebRTC was built using RTCMultiConnection framework and node.js as media signaling. The purpose of this research is to determine the performance of the WebRTC video conference communication with seeing the value parameter delay, jitter, throughput and packet loss as test parameters. Results from the research of communication video conference on WebRTC using the framework RTCMultiConnection show that the more the number of clients it has increased the value of delay, jitter, throughput and packet loss. In the process of communication using a minimum bandwidth of 58 kbps can be concluded that the value of the parameter index delay and jitter can be categorized in good category by category version TIPHON, but on the value of the index throughput can be categorized in the unfavorable category and the index value of packets loss is entered in the category according to TIPHON version.

Keywords: WebRTC, RTCMultiConnectin, *Video conference*, Node.js,TIPHON.

1. PENDAHULUAN

1.1 Latar Belakang

Salah satu teknologi komunikasi yang sekarang banyak digunakan adalah komunikasi yang dilakukan melalui layanan internet. Pemanfaatan teknologi komunikasi melalui jaringan internet menjadi komunikasi yang banyak diminati dan dibutuhkan oleh semua kalangan masyarakat. Hal ini dapat berlaku untuk perusahaan, kampus dan lain-lain (Aulia, 2006). Layanan komunikasi saat ini tidak hanya komunikasi melalui suara saja, melainkan terdapat layanan komunikasi yang dapat menggantikan tatap muka pada setiap pengguna, seperti layanan *video conference*

yang sangat memudahkan pengguna untuk saling berinteraksi satu sama lain.

Video conference yaitu salah satu aplikasi multimedia yang memungkinkan komunikasi suara dan gambar yang bersifat *real time*. Bentuk aplikasi ini adalah percakapan lewat video dan audio antar pengguna secara langsung dan diharapkan dapat menggantikan fungsi tatap muka (Nurdiansyah, 2013). Sebelumnya untuk melakukan komunikasi *video conference* pengguna diharuskan memasang *plugin,* seperti pada layanan aplikasi *skype*. Saat ini telah dikembangkan suatu teknologi yang dapat digunakan untuk melakukan komunikasi, yaitu WebRTC. Teknologi tersebut berjalan pada *browser* yang bersifat *realtime* dan *reliable*.

Web Real Time Communication (WebRTC) merupakan sekumpulan standar dari *World Wide Web Consortium* (WC3) dan *Internet Engineering Task* Force (IETF), dimana WebRTC dapat melakukan proses komunikasi secara *real-time* pada *browser* yang telah mendukung fitur WebRTC. WebRTC memanfaatkan fitur dari HTML5 dan Javascript tanpa *plugin* tambahan, sehingga memungkinkan untuk mengaktifkan audio dan kamera yang terdapat pada perangkat keras. (Phil Edholm E. at.). WebRTC dapat digunakan untuk layanan komunikasi *Video conference* dengan jumlah lebih dari dua pengguna dalam melakukan proses komunikasi.

WebRTC akan merevolusi komunikasi web, namun teknologi ini masih dalam pengembangan dan standarisasi proses [Taheri, 2015]. Oleh karena itu diperlukan lebih banyak penelitian yang harus dilakukan untuk melihat kinerja dari WebRTC. Seperti dari proses pembentukan koneksi, media komunikasi dan data *channel*. Parameter yang perlu diteliti dalam implementasi WebRTC adalah pengaruh jumlah *client* WebRTC dan besar *bandwidth* dalam satu sesi *video conference* saat proses komunikasi sedang berlangsung.

Dari permasalahan diatas penulis melakukan analisis performansi terhadap layanan *video conference* WebRTC dengan menggunakan *framework* RTCMulticonnection. Penelitian ini terfokus pada proses komunikasi ketika sedang berlangsung, sehingga penulis dapat mengetahui *Quality of Service* pada layanan WebRTC. Parameter yang digunakan untuk mengetahui baik atau buruk dari layanan WebRTC adalah *delay, jitter, throughput* dan *packet loss,* kemudian hasil dari parameter penelitian dibandingkan dengan standar versi TIPHON.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang akan diangkat, maka rumusan masalah dikhususkan pada:

1. Bagaimana hasil dan analisis parameter QoS pada komunikasi *video conference* WebRTC dengan jumlah *client* dan *bandwidth* yang berbeda?
2. Apakah nilai *delay, jitter, throughput dan packets loss* pada komunikasi *video conference* WebRTC memenuhi standar komunikasi yang baik menurut standar TIPHON?

1.3 Tujuan

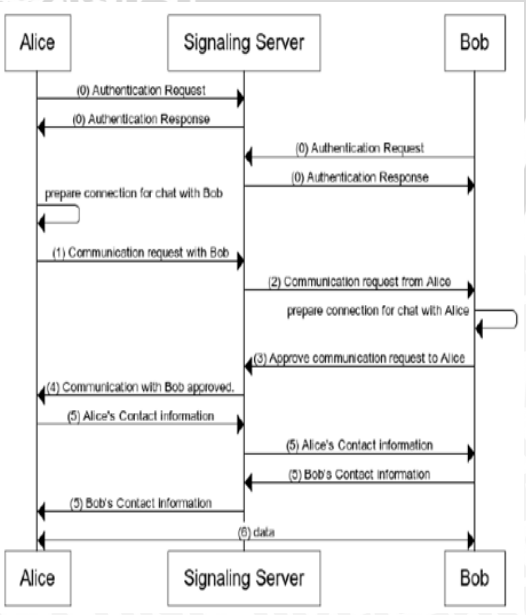
Tujuan dari penelitian ini adalah untuk mengetahui performansi dari komunikasi *video conference* pada WebRTC dengan jumlah *client* dan variasi *bandwith* yang berbeda.

2. LANDASAN KEPUSTAKAAN

2.1 API WebRTC

1. Media Stream**:** Salah satu API yang menyediakan layanan untuk dapat memperoleh *stream* audio dan video yang didapatkan dari kamera dan mikrofon tanpa memerlukan *plugin.*
2. RTCPeersConnection: Menyediakan layanan untuk membentuk sebuah koneksi diantara *peers* dan *stream* audio atau video.
3. RTCDataChannel: Menyediakan layanan untuk melakukan proses komunikasi data, terdiri dari sebuah API mirip dengan *WebSockets* untuk transfer data diantara peers

2.2 MEKANISME WebRTC



Gambar 2.1 Meknisme WebRTC

(1) Alice memulai koneksi dengan mengirimkan permintaan ke signaling server untuk melakukan komunikasi dengan Bob, pada proses ini Alice memberikan informasi tentang deskripsi komunikasi yang digunakan, isi dari session deskripsi tersebut berupa codec media yang digunakan.
(2) Infomasi diatas akan dikirimkan ke server signaling dan setelah koneksi berlangsung maka informasi tersebut digunakan sebagai otentikasi antar kedua klien yang bertujuan untuk memastikan bahwa Alice dan Bob dapat melakukan komunikasi.
(3) Signaling server mengotentikasi pesan dari Alice dan meneruskannya ke Bob.
(4) Bob dapat melakukan proses atau menolak permintaan komunikasi dari Alice. Jika Bob menyetujui permintaan dari Alice, maka Bob membuka saluran komunikasinya dan mengirimkan respon ke server signaling.
(5) Server signaling meneruskan pesan dari Bob ke Alice.
(6) Misalkan Bob menyetujui permintaan dari Alice , maka mereka langsung melakukan pertukaran informasi komunikasi, termasuk IP yang digunakan untuk melakukan komunikasi, jenis komunikasi (TCP/UDP), dan informasi traversal seperti NAT (IP / Port Binding).
(7) Kedua klien memulai direct komunikasi antar peer.

2.3 UDP

UDP disebut protokol connectionless, protokol transport yang tidak dapat dihandalkan, UDP adalah protokol yang sangat sederhana UDP merupakan mekanisme pengiriman datagram dari satu aplikasi ke aplikasi lain. UDP berfungsi untuk meyisipkan *field number port* sumber dan tujuan untuk layanan *multiplexing*. Tidak ada umpan balik untuk mengontrol tingkat infromasi yang dimiliki pesan

UDP digunakan untuk multimedia *streaming* yang sangat memberikan toleransi kehilangan segment cukup baik dan yang sangat tidak sensitive terhadap kerusakan atau kehilangan segment. UDP merupakan transport yang agresif dan selalu mencoba untuk mengambil *bandwidth* yang tersedia.

2.4 *Framework* RTCMultiConnection

RTCMultiConnection merupakan sebuah WebRTC JavaScript *framework* yang berjalan diatas API RTCPeerConnection untuk digunakan sebagai skenario *multisession* dibangun dari layanan WebRTC. Didalam framework ini telah banyak disediakan method-method yang dibutuhkan dalam perancangan komunikasi *video conference* pada layanan WebRTC. MultiConnection memungkinkan dua atau lebih pengguna untuk saling berkomunikasi secara langsung *browser* satu dengan *browser* yang lainnya.

2.5 *Quality of Service (QoS)*

Quality of Service (QoS) di definisikan sebagai sebuah mekanisme atau cara yang memungkinkan layanan dapat beroperasi sesuai dengan karakteristiknya masing-masing dalam jaringan IP.

Parameter yang lazim dijadikan referensi umum untuk mengamati kerja jaringan, diantarnya adalah *delay* dan *jitter.*

- *Delay*

Delay adalah waktu yang dibutuhkan untuk mengirimkan data dari sumber ke tujuan. Adanya variasi waktu tunda dalam transmisi layanan aplikasi internet menimbulkan masalah tersendiri, yaitu paket-paket yang datang terlambat yang dapat mengganggu data yang diterima oleh pengguna

- *Jitter*

Jitter merupakan variasi *delay* yang terjadi akibat adanya selisih waktu atau interval antara kedatangan paket di penerima

- *Throughput*

Throughput didefinisikan sebagai jumlah paket yang diterima di sisi penerima dengan benar tiap satuan waktu

- *Packet Loss*

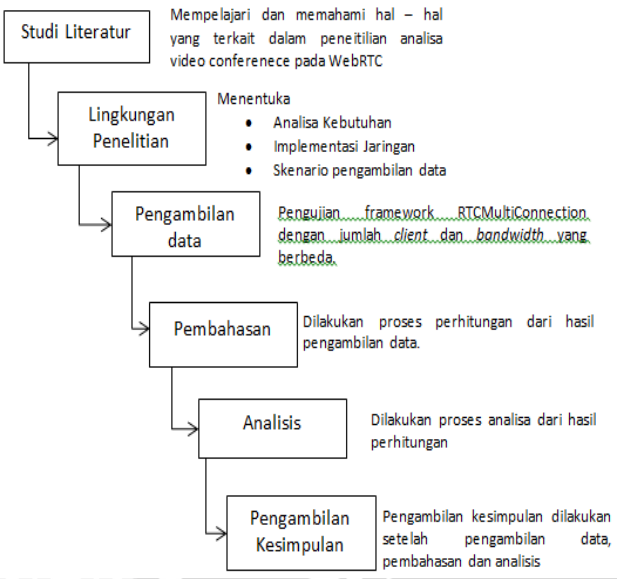
Packet Loss merupakan jumlah paket yang hilang dibandingkan dengan paket yang diterima (Tanenbaum, 2003). Paket yang hilang dapat disebabkan oleh beberapa kemungkinan, antara lain adalah:

a. Terjadinya *over load* dalam jaringan.
b. Tabrakan atau tumbukan dalam jaringan.
c. *Error* yang terjadi pada media fisik.
d. Pengiriman data pada waktu yang bersamaan dengan menggunakan sebuah saluran secara bersama-sama.

Semakin kecil nilai *packet loss* dalam suatu jaringan maka semakin baik pula kinerja yang dimiliki jaringan tersebut. *Packet loss* biasanya dinyatakan dalam ukuran prosentasi (%) untuk mencari besar *packet loss* dalam komunikasi. Biasanya *packet loss* yang dapat ditolerir adalah sebesar 10% (Mehta dkk, 2001).

3. METODOLOGI

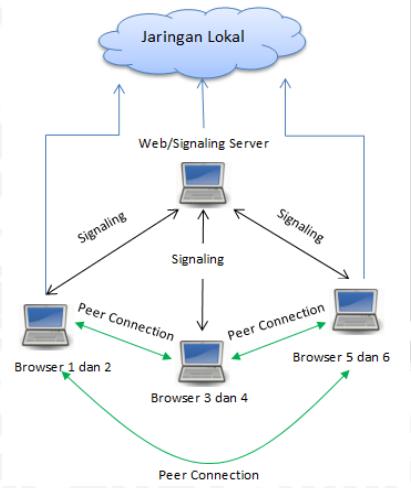
Penelitian ini merupakan penelitian non-implementatif analitik. Alur metode penelitian dapat dilihat pada Gambar 3.1



Gambar 3.1 Alur Penelitian

Metodologi penelitian diawali dengan studi pendahuluan yang terdiri dari studi literatur, kemudian proses penerapan lingkungan penelitian yang terdiri dari proses pembangunan jaringan layanan *video conference* pada WebRTC, proses pembatasan *bandwidth* menggunakan *software* Netlimiter, selanjutnya adalah metode pengumpulan data dan proses mendapatkan data pada saat melakukan suatu skenario. Hasil pengujian diolah terlebih dahulu yang dapat dijadikan pada proses analisa. Metode analisa data dilakukan untuk mengetahui hasil analisa performasi pembentukan layanan *video conference* melalui serangkaian proses pengujian yang dilakukan sehingga menghasilkan satu kesimpulan yang dapat ditarik sebagai rangkuman penelitian.

3.1 Perancangan Sistem

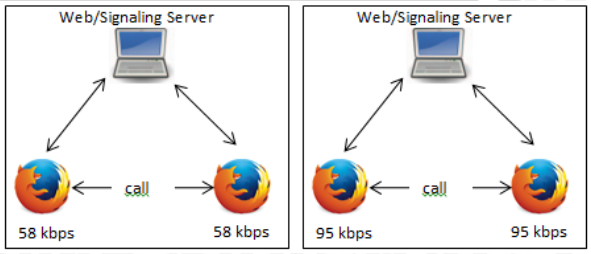


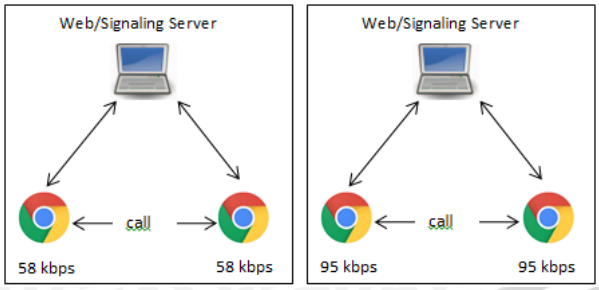
Gambar 3.2 Perancangan Sistem

Gambar 3.2 merupakan perancangan sistem pada *framework* RTCMultiConnection mekanisme komunikasi yang terjadi adalah *client* A dan *client* B mengakses halaman Web yang telah dibuat, dengan memanggil IP_Server:9001 (port), lalu setelah kedua *client* mengakses halaman *video conference,* kemudian salah satu *client* dapat melakukan "Open Room" atau membuka suatu komunikasi, dengan memasukan kode atau nama unik untuk berbagi dengan *client* lain, kemudian *client* A dapat melakukan proses akses getusermedia (akses webcam dan mikrofon), apabila telah diperbolehkan mengakses media maka akan mucul hasil kamera, untuk *client* yang ingin bergabung dengan layanan *video conference* maka *client* lainnya bisa memasukan kode yang sama dengan *client* A, untuk layanan WebRTC penulis menggunakan *framework* RTCMultiConnection sebagai interface client, dan node.js sebagai media signaling yang sudah terdapat modul socket.io.

3.2 Skenario Pengambilan Data

Pengumpulan data pengujian dilakukan dengan melakukan proses komunikasi menggunakan *framework* RTCMultiConnection dengan variasi *bandwidth* berbeda, pengujian komunikasi *video conference* akan dilakukan dengan melakukan komunikasi antar *client* selama empat menit dengan tiga kali pengujian untuk setiap kondisi *bandwidth*. Ketika proses komunikasi berlangsung akan dilakukan pengambilan data menggunakan Wireshark. Dari hasil *capture* wireshark tersebut dapat dianalisa nilai dari masing-masing parameter, pengambilan data kualitas *video conference* akan dilakukan selama pengujian komunikasi sedang berlangsung. Terdapat delapan skenario pengujian yang terbagi dalam dua kondisi *bandwidth* dan jumlan *client* yang berbeda dan pada tiap skenario akan dilakukan tiga kali pengujian. Gambar 3.3 menunjukan skema panggilan pada Mozilla Firefex dan Google Chrome untuk dua *client*.





Gambar 3.3 Skema *Video Conference* Dua *Client*

Tabel 3.1 Skenario Pengambilan Data

Browser	Jumlah *Client*	*Bandwidth*
Mozilla Firefox	Dua	58 kbps
		95 kbps
	Tiga	58 kbps
		95 kbps
	Lima	58 kbps
		95 kbps
	Enam	58 kbps
		95 kbps
Google Chrome	Dua	58 kbps
		95 kbps
	Tiga	58 kbps
		95 kbps
	Lima	58 kbps
		95 kbps
	Enam	58 kbps
		95 kbps

Setelah proses pengambilan data, kemudian dilakukan proses perhitungan untuk mengetahui hasil dari parameter *throughput*, *delay*, *packet loss* dan *jitter*.

4. Lingkup Penelitian

4.1 Konfigurasi Server

Diperlukannya modul node.js sebagai proses *signaling*, pada penelitian ini dibutuhkan node.js pada server dimana server tersebut berada dalam OS ubuntu 14.04. langkah-langkah untuk menginstall node.js

```
#curl -sL
https://deb.nodesource.com/setup |
sudo bash -
#apt-get install -y nodejs
https://github.com/priologic/easyrtc.git
```

Setelah node.js selesai terinstal langkah selanjutnya adalah instalasi *framework* RTCMultiConnection pada server.

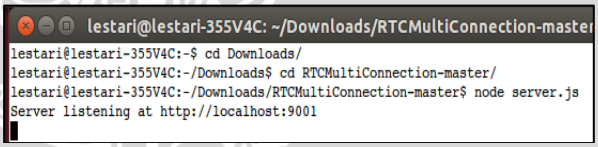
```
$sudo su
#cd
/Document/Program/RTCMultiConnect
ion
#npm install
```

Setelah *framework* easyrtc selesai terinstal, kemudian jalankan server RTCMultiConnection.

Setelah semua selesai diseting dan dikonfigurasi, sekarang tahap untuk menghidupkan *signaling* dari server, untuk menjalankan *signaling* agar webrtc tersebut dapat diakses oleh pengguna sebagai berikut

```
#cd
/Document/Program/RTCMultiConnectio
n
#node server.js
```

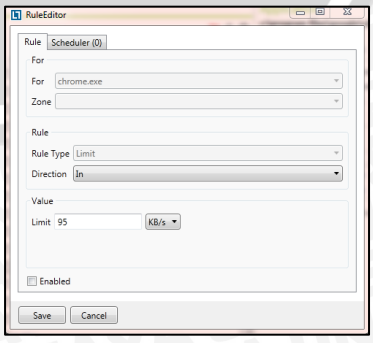
Maka proses signaling akan berjalan dan server menunggu pengguna yang akan mengakses webrtc *video conference* pada IP address server dengan port 9001, tampilan dari server node.js dapat dilihat pada gambar 4.1.



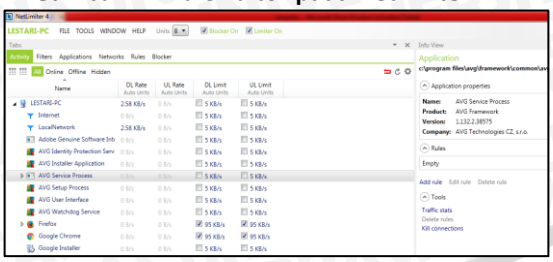
Gambar 4.1 Server RTCMultiConnection

4.2 Proses Pembatasan *bandwidth*

Untuk proses pembatasan *bandwidth* penulis menggunakan *software* netLimiter, *software* ini akan membatasi *bandwidth* yang diinginkan pada setiap aplikasi, untuk *setting bandwidth* kita dapat merubahnya pada menu *Rule Editor* terdapat nilai limit yang akan di *setting* kemudian *save*, maka *bandwidth* yang digunakan tidak akan membatasi *bandwidth* yang telah ditentukan. Gambar 4.2 dan Gambar 4.3 menunjukan proses konfigurasi pada Netlimiter.



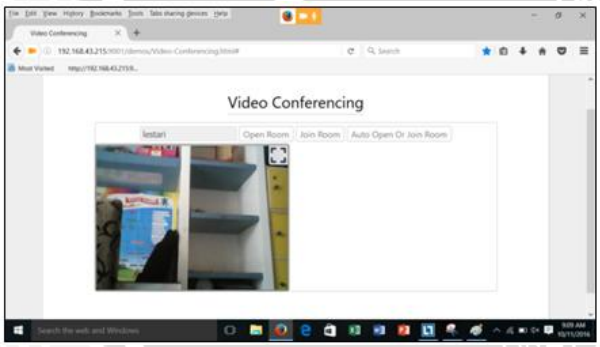
Gambar 4.2 *Rule Editor* pada Netlimiter



Gambar 4.3 Proses Pembatasan *Bandwidth*

4.2 Proses Komunikasi *client*

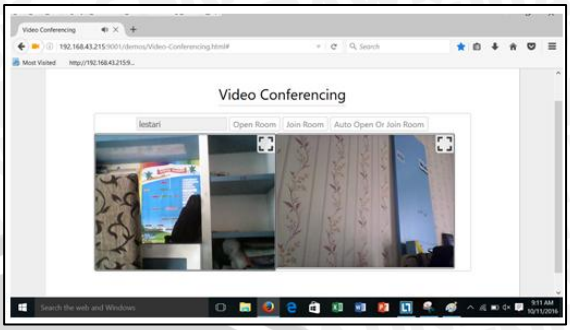
client mengakses halaman *video conference* masukan kode pada field text *yang* digunakan sebagai kode untuk bisa terhubung antara *client* A dan *client* B , misalkan disini *client* A memberikan kode dan melakukan proses *Open Room* berarti *client* A membuka komunikasi terhadap *client* lain dan akan mucul tampilan untuk mengakses kamera dan mikrofon seperti pada Gambar 4.4.



Gambar 4.4 Akses Kamera

Setelah *client* mengijinkan untuk mengakses kamera dan mikrofon maka akan otomatis muncul kamera yang digunakan oleh perangkat keras itu sendiri.

Kemudian *client* A dapat memberitahu *client* B kode yang digunakan dan *client* B dapat melakukan proses *Join Room* dengan menggunalan kode yang telah diberikan *client* A maka kedua *client* akan saling terhubung untuk melakukan komunikasi *video conference* seperti yang terlihat pada Gambar 4.5 *Client* A dan *client* B sudah dapat saling terhubung satu sama lain.



Gambar 4.5 Komunikasi WebRTC

5. Pengambilan Data

Pada pengujian *bandwidth* 58 kbps penulis melakukan tiga kali percobaan dengan waktu sekitar empat menit. Dibawah ini merupakan filter yang digunakan pada wireshark

```
Ip.addr == 192.168.43.3 && ip.addrs==
192.168.43.113 && udp.port == 50992
&& udp.port == 62195
```

Pada penelitian ini terfokus pada protokol UDP karena penelitian ini memfokuskan pada proses pertukaran data *mediastream* dan *peers* pada saat komunikasi telah terhubung satu sama lain. Dalam proses pertukaran data *Framework* RTCMultiConnection berjalan dengan memanfaatkan protokol UDP. Pengambilan data untuk kualitas *video conference* diambil dari dua sumber yaitu *client* dengan IP 192.168.43.3 menuju *client* dengan IP 192.168.43.113 dengan nomor port 50992 dan 62195.

Traffic	Captured	Displayed	Displayed %	Marked	Marked %
Packets	31453	30813	97.965%	0	0.000%
Between first and last packet	231.313 sec	231.313 sec			
Avg. packets/sec	135.976	133.209			
Avg. packet size	457 bytes	462 bytes			
Bytes	14386803	14225617	98.880%	0	0.000%
Avg. bytes/sec	62196.296	61499.465			
Avg. MBit/sec	0.498	0.492			

Gambar 5.1 *Summary* Wireshark

Untuk mengetahui paket data yang didapatkan setelah proses *filtering* maka dapat dilihat pada sub menu *summary pada Wireshark* dari sini penulis dapat menghitung parameter dari setiap percobaan yang dilakukan seperti yang terlihat pada Gambar 5.1.

Untuk langkah selanjutnya dilakukan proses perhitungan parameter.

A. *Delay*

Dari *capture* data yang telah dilakukan dengan wireshark maka didapatkan rata-rata *delay*.

$$Delay\ Rata-rata = \frac{Total\ Delay}{Total\ Paket\ yang\ diterima}$$

$$Delay\ Rata-rata = \frac{241{,}578}{39932} = 0{,}006\ \text{sec} = 6{,}050\ ms$$

Tabel 6.1 *Delay* pada Pengujian ke-1

Delay	
Total *Delay*	241,578
Total Paket yang diterima	39932
rata - rata *delay* (s)	0,006
rata - rata *delay* (ms)	6,050

total *delay* (*second*) didapatkan dari hasil penjumlahan pengiriman paket data yang terdapat pada tabel "*time delta from previous displayed frame*" pada wireshark. Tabel tersebut adalah waktu kedatangan pada tiap paket data waktu, kemudian untuk total paket data yang diterima didapatkan pada *summary* wireshark pada baris *packets* dan kolom *displayed*. Kemudian setelah mendapatkan nilai tersebut maka dapat dilakukan proses perhitungan menggunakan persamaan 2.1 maka didapatkan rata-rata *delay* dari hasil perhitungan persamaan tersebut.

B. *Jitter*

Dari *capture* data yang telah dilakukan dengan wireshark maka didapatkan nilai *jitter.*

$$Jitter = \frac{Total\ Variasi\ Delay}{Total\ Paket\ yang\ diterima}$$

Total variasi *delay* diperoleh dari penjumlahan :

$$delay2 - delay1 \ + \ delay3 - delay2 \ + \cdots + (delay\ n - delay\ (n-1)$$

Untuk menghitung nilai *jitter* penulis juga melakukan tiga kali percobaan selama empat menit. Berikut nilai *jitter* yang penulis dapatkan dari tiga kali percobaan.

Tabel 6.2 *Jitter* pada Pengujian ke-1

Jitter	
Total variasi *delay*	0,000055
Total paket yang diterima	39932
Nilai *Jitter* (s)	0,000000
Nilai *Jitter* (ms)	0,000001

total variasi *delay* (*second*) didapatkan dengan menjumlahkan keseluruhan selisih *delay* yang ada antara paket satu dengan paket yang lainnya kemudian dibagi dengan total paket yang diterima.

C. *Throughput*

Dari *capture* data yang telah dilakukan dengan wireshark maka didapatkan nilai *troughput*.

$$Throughput = \frac{Paket\ data\ yang\ diterima}{lama\ pengamatan}$$

$$Throughput = \frac{14727978\ bytes}{241{,}577\ sec} = 60965{,}8\ \text{bps} = 60{,}966\ \text{kbps}$$

Tabel 6.3 *Throughput* pada Pengujian ke-1

Throughput	
paket yang dikirim (bytes)	14727978
lama pengamatan (s)	241,577
Nilai *Throughput* (bps)	60965,8
Throughput (kbps)	60,966

D. *Packet Loss*

Untuk menghitung nilai *packet loss* penulis juga melakukan tiga kali percobaan selama empat menit. Berikut nilai *packets loss* yang penulis dapatkan dari tiga kali percobaan. Untuk mengetahui *paket loss* yang terjadi pada komunikasi WebRTC.

$$Paket\ loss = \frac{paket\ data\ yang\ dikirim\ -\ paket\ yang\ diterima}{Paket\ data\ yang\ dikirim} x\ 100\ \%$$

$$Paket\ loss = \frac{40206 - 39932}{40206} x\ 100\ \% = 0{,}68\ \%$$

Tabel 6.4 *packet loss* pada Pengujian ke-1

Packet Loss	
Paket yang dikirim	40206
paket yang diterima	39932
nilai *packet loss*	0,68

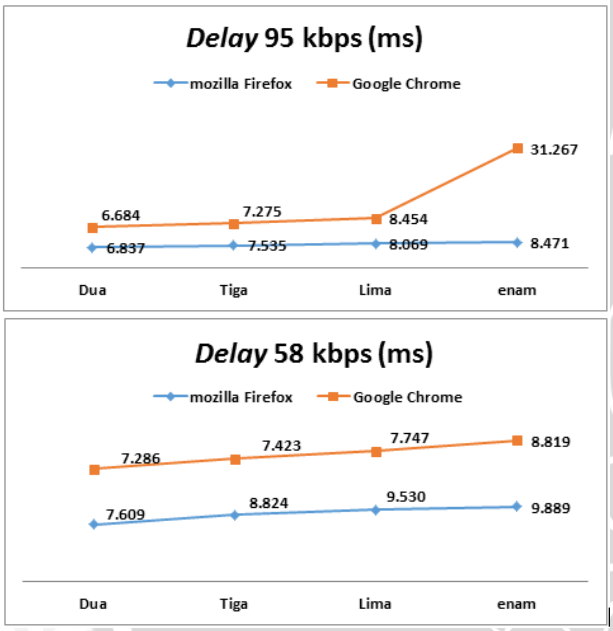
6. Hasil dan Pembahasan

Tabel 6.5 Nilai *Delay*

58 kbps		
Jumlah *Client*	mozilla Firefox (ms)	Google Chrome (ms)
Dua	7,609	7,286
Tiga	8,824	7,423
Lima	9,530	7,747
Enam	9,889	8,819

95 kbps		
Jumlah *Client*	mozilla Firefox (ms)	Google Chrome (ms)
Dua	6,837	6,684
Tiga	7,535	7,275
Lima	8,069	8,454
Enam	8,471	31,267

Nilai *delay* komunikasi *video conference* pada kondisi *bandwidth* 58 kbps pada Mozilla Firefox cenderung lebih tinggi dibandingkan pada *browser* Google Chrome dan semakin banyak banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *delay,* apabila dibandingkan dengan *delay* pada standarisasi TIPHON nilai *delay* komunikasi *video conference* masuk kedalam kategori sangat baik, dikarenakan nilai rata-rata *delay* pada komunikasi *video conference* kurang dari 150 ms.



Gambar 6.1 Grafik Nilai *Delay*

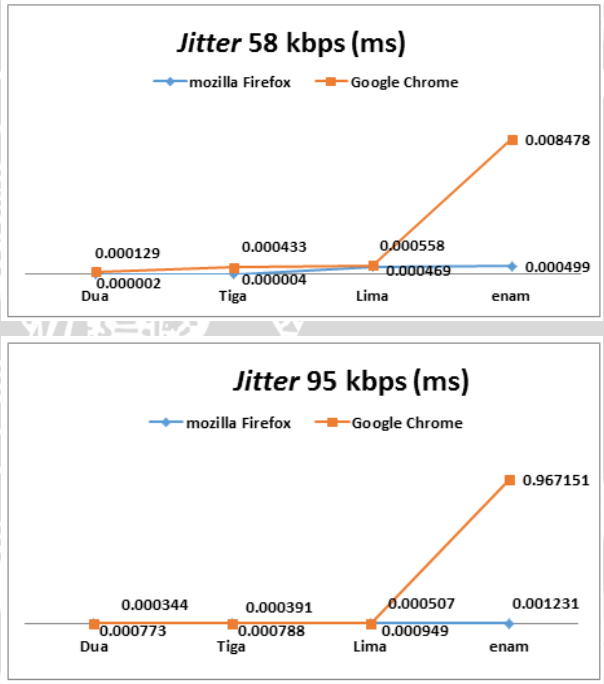
Sedangkan nilai *delay* komunikasi *video conference* pada kondisi *bandwidth* 95 kbps pada Mozilla Firefox cenderung lebih tinggi dibandingkan pada *browser* Google Chrome terkecuali pada kondisi jumlah *client* sebanyak enam, pada saat kondisi ini nilai delay meningkat drastis disebabkan banyak paket yang hilang sehingga banyak paket yang tidak diterima, apabila dibandingkan dengan *delay* pada standarisasi TIPHON nilai *delay* komunikasi *video conference* masuk kedalam kategori baik, dikarenakan nilai rata-rata *delay* pada komunikasi *video conference* kurang dari 150 ms.

Tabel 6.6 Nilai *jitter*

58 kbps		
Jumlah Client	**mozilla Firefox (ms)**	**Google Chrome (ms)**
Dua	0,000002	0,000129
Tiga	0,000004	0,000433
Lima	0,000469	0,000558
Enam	0,000499	0,008478

95 kbps		
Jumlah Client	**mozilla Firefox (ms)**	**Google Chrome (ms)**
Dua	0,000773	0,000344
Tiga	0,000788	0,000391
Lima	0,000949	0,000507
Enam	0,001231	0,967151

Nilai *jitter* terlihat bahwa nilai *jitter* komunikasi *video conference* pada kondisi *bandwidth* 58 kbps pada Mozilla Firefox cenderung lebih tinggi dibandingkan pada *browser* Google Chrome dan semakin banyak banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *jitter,* apabila dibandingkan dengan *jitter* pada standarisasi TIPHON nilai *jitter* komunikasi *video conference* masuk kedalam kategori baik, dikarenakan nilai rata-rata *jitter* pada komunikasi *video conference* kurang dari 75 ms.



Gambar 6.2 Grafik Nilai *jitter*

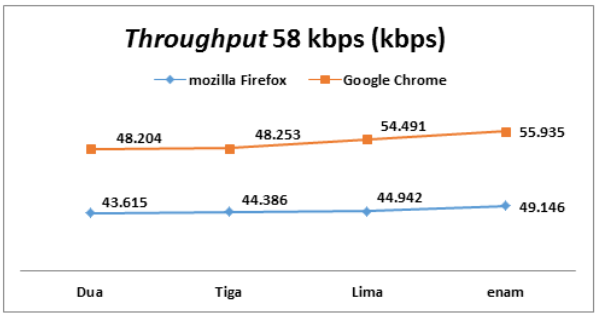
Nilai *jitter* komunikasi *video conference* pada kondisi *bandwidth* 95 kbps pada Mozilla Firefox cenderung lebih tinggi dibandingkan pada *browser* Google Chrome dan semakin banyak banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *jitter*.

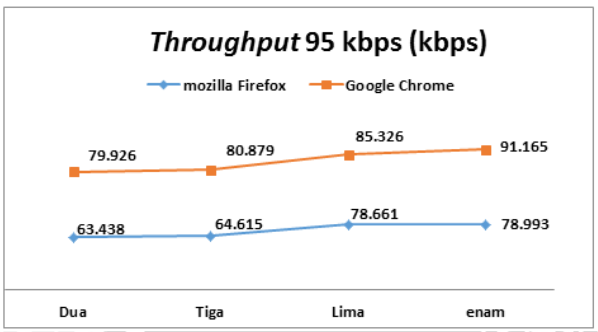
Tabel 6.7 Nilai *Throughput*

58 kbps		
Jumlah Client	**mozilla Firefox (kbps)**	**Google Chrome (kbps)**
Dua	43,615	48,204
Tiga	44,386	48,253
Lima	44,942	54,491
Enam	49,146	55,935

95 kbps		
Jumlah Client	mozilla Firefox (ms)	Google Chrome (ms)
Dua	63,438	79,926
Tiga	64,615	80,879
Lima	78,661	85,326
Enam	78,993	91,165

Nilai *throughput* komunikasi *video conference* pada kondisi *bandwidth* 58 kbps pada Google Chrome cenderung lebih tinggi dibandingkan pada *browser* Mozilla Firefox dan semakin banyak banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *throughput* yang dihasilkan.





Gambar 6.3 Grafik Nilai *Throughput*

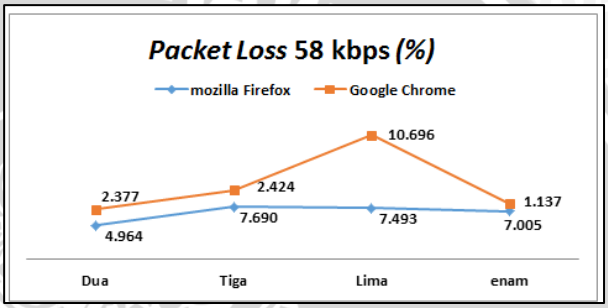
Nilai *throughput* komunikasi *video conference* pada kondisi *bandwidth* 95 kbps pada Google Chrome cenderung lebih tinggi dibandingkan pada *browser* Mozilla Firefox dan semakin banyak banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *throughput* yang dihasilkan.

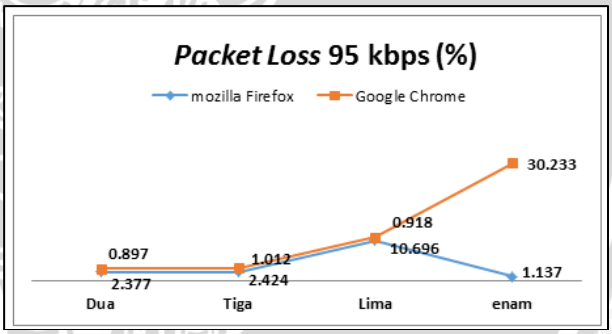
Tabel 6.8 Nilai *Packet Loss*

58 kbps		
Jumlah Client	mozilla Firefox (%)	Google Chrome (%)
Dua	4,964	7,493
Tiga	7,690	2,424
Lima	7,493	10,696
Enam	7,493	1,137

95 kbps		
Jumlah Client	mozilla Firefox (%)	Google Chrome (%)
Dua	2,586	0,897
Tiga	0,942	1,012
Lima	2,621	0,918
Enam	6,898	30,233

Nilai *packet loss* komunikasi *video conference* pada kondisi *bandwidth* 58 kbps pada Mozilla Firefox cenderung lebih tinggi dibandingkan pada *browser packet loss*, apabila dibandingkan dengan *packet loss* pada standarisasi TIPHON nilai *packet loss* komunikasi *video conference* masuk kedalam kategori baik, dikarenakan nilai rata-rata *packet loss* pada komunikasi *video conference* kurang dari 10 %.





Gambar 6.4 Grafik *Packet Loss*

Nilai *packet loss* komunikasi *video conference* pada kondisi *bandwidth* 95 kbps pada Mozilla Firefox cenderung lebih tinggi dibandingkan pada *browser packet loss*, akan tetapi pada saat jumlah *client* enam pada Google Chrome mengalami kenaikan yang sangat tinggi hal ini disebabkan pada pengiriman UDP tidak terdapat *control congesti* yang berakibat paket-paket akan dikirimkan tanpa perlu dibatasi saat kapasitas *queue* sedang mengalami kepadatan.

7. Penutup

a. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Terdapat empat parameter *quality of service* yang telah diuji pada jumlah *client* dan *bandwidth* yang berbeda
   - *Delay* dan *jitter*

     Dapat disimpulkan bahwa nilai *delay* dan *jitter* dari pengujian menghasilkan bahwa semakin banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *delay* dan *jitter.* Pada saat jumlah *client* meningkat maka Kecepatan terima dan pengiriman paket dari setiap node menyebabkan *jitter* semakin meningkat juga.

   - *Throughput*

     Nilai *throughput* komunikasi *video conference* pada kondisi *bandwidth* 58 dan 95 kbps, semakin banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *throughput* yang dihasilkan.

   - *Packets Loss*

     Nilai *packet loss* komunikasi *video conference* pada kondisi *bandwidth* 58 dan 95 kbps, semakin banyak jumlah *client* pada saat komunikasi berlangsung maka semakin tinggi nilai *packet loss* yang dihasilkan. Faktor yang menyebabkan terjadinya *packets loss* karena pada pengiriman UDP tidak terdapat *control congesti* yang berakibat paket-paket dikirimkan tanpa dibatasi saat kapasitas *queue* sedang mengalami kepadatan.

2. Dari hasil parameter *delay*, *jitter*, *throughput* dan *packet loss* pada komunikasi *video conference* WebRTC dengan menggunakan *framework* RTCMultiConnection, dapat disimpulkan bahwa nilai parameter *delay* dan *jitter* dapat dikategorikan pada kategori baik menurut versi TIPHON, namun pada nilai parameter *throughput* dapat dikategorikan pada kategori kurang baik dan nilai *packets loss* masuk pada kategori sedang menurut versi TIPHON.

DAFTAR PUSTAKA

Agassi, Mohammad, Vicky. 2017. Analisis Skalabilitas Dan Quality of Service komunikasi Audio Call Pada WebRTC dengan menggunakan Framework EasyRTC. Universitas Brawijaya.

Aulia, Nur, Hendra, 2006. *Perancangan MAC untuk layanan video conference sebagai alat bantu perkuliahan.* STT Telkom Bandung. www.scribd.com. Diakses pada tanggal 02 Februari 2016.

Bidelman, Eric, 2012. Capturing Audio & Video in HTML5. Tersedia di : < http://www.html5rocks.com/en/tutorials/getusermedia/intro/> [Diakses 30 Januari 2016>.

Phil Edholm, E. Brent Kelly, Ph.D., Matt Krebs, November 2014, "*WebRTC Ecosystem "Overview" A Description of the Ecosystem Framework*", pedholm@pkeconsulting.com, bkelly@kelcor.com, mattkrebs@kelcor.com.

Priaksa, Indradito, Boby. *Implementasi dan Analisis Performansi Mobicents Sebagai Server Aplikasi pada Arsitektur IP Multimedia Subsistem untuk Layanan Video Conference.* Universitas Telkom Bandung.

Primadasa, I Gede, Putu, Bagus. 2011. Kompresi Video Conference Degan Standar H-263 Dan H261. Universitas Udayana

Nurdiansyah Deby Cahya, 2013. *Implementasi Video Conference Pada Jaringan Hsupa (High SpeedUplink Packet Access) Dengan MediaIpv6 Menggunakan Simulator Opnet Modeler V.14.5*. Universitas Brawijaya.

Rhinow, Florian, Veloso, Pablo, Porto, Barrett Stephen & Nuollain, O, Eamonn. *P2P Live Video Streaming in WebRTC.* School pf Computer Scienceand Statisitcs, Trinity College Dublin.

Sepasthika, Rizki. 2015. Analisis Kinerja Real-Time Transport protokol dan Real-Time Transport control protokol pada VoIP menggunakan Codec GSM & G711. Universitas Brawijaya.

Taheri, Sajjad, Beni, Laleh, Aghababaie, Veidenbaum, V., Alexander & Nicolau Alexandru, 2015. *WebRTCBench : A Benchmark for Performansi Assessment of WebRTC Implementation*. Dept. of Computer Science, UC Irvine, Irvine, CA.

Tiphon. "*Telecommunication and Internet Protocol Harmonization Over Network (TIPHON) General Aspec of Quality of Service (QoS)*", DTR/TIPHON-05006 (cb0010cs.PDF).1999.

WebRTC. WebRTC Architecture. Tersedia di : https://webrtc.org/ [Diakses 01 Februari 2016].