Pengembangan Media Proxy untuk Mendukung Komunikasi *Real Time* Berbasis *Web* (WebRTC)

Iqra Aswad[1] , Muhammad Niswar[*1], Amil Ahmad Ilham[1]
[1]Departemen Teknik Elektro, Fakultas Teknik, Universitas Hasanuddin
Jl. Poros Malino Km.6, Bontomarannu, Gowa, Sulawesi Selatan, 92171, Indonesia
*Email: niswar@unhas.ac.id

Abstrak

Aplikasi komunikasi *real time* berbasis *web* (webRTC) cukup populer saat ini karena memberikan kemudahan bagi para pengguna dalam melakukan pertukaran data berupa gambar, suara maupun video antar *client* secara *peer-to-peer* tanpa adanya plugin tambahan seperti *flash player* pada browser. Akan tetapi aplikasi webRTC sendiri memiliki keterbatasan pada jumlah *client* yang dapat terhubung dalam satu waktu terkhusus pada aplikasi berupa *video conference*. Hal ini dikarenakan setiap *client* yang terhubung akan mengkonsumsi *bandwidth*, RAM, serta *processor* yang cukup tinggi seiring dengan peningkatan jumlah *client* (*peer*). Tujuan dari penelitian ini adalah untuk menghasilkan sebuah infrastruktur yang memanfaatkan media server untuk mengatasi kendala - kendala pada aplikasi webRTC serta dapat diterapkan di lingkungan Universitas Hasanuddin. Dalam penelitian ini dikembangkan sebuah algoritma yang dapat digunakan untuk membagi beban kerja dari aplikasi webRTC kedalam beberapa media server sehingga aplikasi webRTC dapat digunakan untuk skala yang lebih besar. Selain itu algoritma tersebut digunakan untuk mengoptimalkan penggunaan beberapa media server di lingkungan Universitas Hasanuddin. Hasil dari penelitian ini adalah dikembangkannya sebuah aplikasi webRTC berupa *video conference* yang menerapkan algoritma tersebut serta pembuatan model infrastruktur berbasis media server.

Abstract

Real time web-based communications applications (webRTC) are popular nowadays because they make it easy for users to exchange data likeimages, sounds and videos between peer-to-peer (clients) without additional plugins such as flash players in the browser. However, the webRTC application itself has limitations on the number of clients that can connect in a particular time in the application of video conferencing. This is because each connected client will consume bandwidth, RAM, and a high enough processor along with an increase in the number of clients (peer). The purpose of this research is to produce an infrastructure that utilizes the media server to overcome the constraints of the webRTC application and can be applied in the environment of Hasanuddin University. In this study developed an algorithm that can be used to divide the workload of webRTC applications into multiple media servers so that webRTC applications can be used for larger scale. In addition, the algorithm is used to optimize the use of several media servers within the Hasanuddin University. The result of this research is the development of a WebRTC application in the form of video conference that implement the algorithm as well as the making of infrastructure model based on media server.

Kata-kunci: *webRTC,* kurento, *video conference*

.

1. Pendahuluan

Aplikasi berbasis online yang sering menjadi fokus penelitian saat ini adalah aplikasi komunikasi *real time* berbasis *web* atau sering disebut dengan webRTC. WebRTC memungkinkan *web* browser melakukan komunikasi *real time* tanpa menggunakan plugin tambahan serta penginstalan layaknya aplikasi *real time* pada umumnya. Aplikasi ini merupakan sebuah API (*Application Programming Interface*) yang didesain agar dua buah browser atau lebih dapat melakukan *video chat*, *voice chat*, *live editor* dan *file sharing* dengan menggunakan koneksi langsung dari dua browser. Aplikasi yang sedang dikembangkan sejak 2011 ini diharapkan dapat menjadi aplikasi standar oleh semua browser sehingga dapat memudahkan konektivitas. Para pengembang berharap dengan hadirnya webRTC ini akan mengurangi penginstallan sistem yang dapat berujung kepada *error*, dan malfungsi serta membantu tiap-tiap individu yang tidak familiar dengan teknologi untuk menggunakannya dengan mudah.

Dalam pengimplementasiannya, aplikasi-aplikasi pembelajaran jarak jauh yang dibangun

dengan menggunakan webRTC terkhusus aplikasi yang tidak menggunakan sebuah media server sering mengalami kendala – kendala seperti adanya keterbatasan jumlah *client* atau partisipan yang dikarenakan oleh kemampuan dari komputer *client* yang terbatas. Hal ini dikarenakan seluruh pekerjaan dilakukan secara langsung oleh komputer *client* mulai dari proses tukar menukar data antara *client* hingga proses *decoding* dan *encoding* video / audio.

Adapun penelitian yang telah dilakukan sebelumnya antara lain penelitian dengan judul judul Kurento : *a media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC* [1]. Penelitian ini memperkenalkan sebuah media server yang berbasis *open source* bernama Kurento dengan mengkombinasikan aplikasi SIP/HTTP berbasis *signaling* dan sebuah infrastruktur media server yang dibuat dengan menggunakan GStreamer. Teknologi yang dihadirkan dapat menunjang pengiriman multimedia secara *real time* melalui protokol dan format yang berbeda. Selain itu teknologi yang dihadirkan juga menawarkan berbagai fasilitas seperti *media mixing*, *transcoding*, dan *filtering*. Dengan adanya fasilitas tersebut, WebRTC tidak hanya menawarkan aplikasi *peer-to-peer* semata.

Penelitian kedua yang berjudul *Catalysing the Success of WebRTC for the Provision of Advanced Multimedia Real-Time Communication Services* [2]. Penelitian ini mendeskripsikan bagaimana sebuah API yang sederhana dapat digunakan untuk memanfaatkan berbagai fungsi dari Kurento.

Penelitian ketiga yang berjudul *Models for Multimedia Conference between Browser based on WebRTC* [3]. Penelitian ini menggunakan 2 jenis aplikasi konferensi berbasis WebRTC yang ditujukan secara khusus untuk konferensi dalam skala kecil maupun skala besar. Aplikasi yang dikembangkan dalam penelitian ini menggunakan SID / DSP yang bersifat primitif.

Penelitian keempat yang berjudul *A P2P-MCU Approach to Multi-Party Video Conference with WebRTC* [4]. Fokus penelitian yang dilakukan adalah mengembangkan aplikasi *video conference* dimana setiap *peer* dapat bertindak sebagai MCU atau pusat distribusi *data stream*. Setiap *peer* / komputer maupun *gadget* yang terhubung dengan aplikasi *video conference* tersebut akan diberikan *score* sesuai dengan kondisi *network*, maupun CPU yang ada. *Peer* yang terpilih akan dijadikan sebagai MCU (*Media Control Unit*), dan komunikasi akan terpusat kepada *peer* tersebut. Apabila *peer* tersebut keluar dari aplikasi *video conference*, maka akan ditunjuk kandidat baru untuk menggantikan *peer* tersebut.

Penelitian kelima yang berjudul *A Comparison of QoS Parameters of WebRTC Videoconference with Conference Bridge Placed in Private and Public Cloud* [5]. Dalam penelitian ini dilakukan analisa parameter QoS trafik WebRTC untuk memusatkan sistem *video conference* yang digunakan untuk pemakaian bersama. Hasilnya menunjukkan bahwa *conference bridge* dalam suatu sistem cloudmerupakan solusi yang baik untuk WebRTC dan transimisi tambahan dari data telemetrik tidak mempengaruhi parameter QoS dari *media stream* WebRTC.

Penelitian keenam yang berjudul *Hyper-linked communications: WebRTC enabled asynchronous collaboration* [6]. Penelitian ini mendeskripsikan desain, implementasi dan evaluasi platform komunikasi hyper-media yang menargetkan platform *web* dan memanfaatkan teknologi WebRTC. Hasil penelitian tersebut menunjukkan bahwa tes pemakaian oleh 20 *user* yang mengevaluasi prototipe mereka secara positif 100% menganggapnya inovatif dan 95% me-rekomendasikan penggunaannya.

Penelitian ketujuh yang berjudul *Congestion Control for WebRTC: Standardization Status and Open Issues* [7]. Penelitian ini meninjau status dari kebutuhan standar yang mengambil tempat di grup kerja RMCAT. Mereka mendiskusikan pilihan yang dilibatkan dalam desain control kemacetan media, algoritma yang diusulkan dan isu yang dianggap masih terbuka.

Penelitian kedelapan yang berjudul *Experimental Evaluation of Simulcast for WebRTC* [8]. Dalam penelitian tersebut dilakukan pemeriksaan terhadap persyaratan komputasi dan *bandwith* baik dari server maupun *client* serta

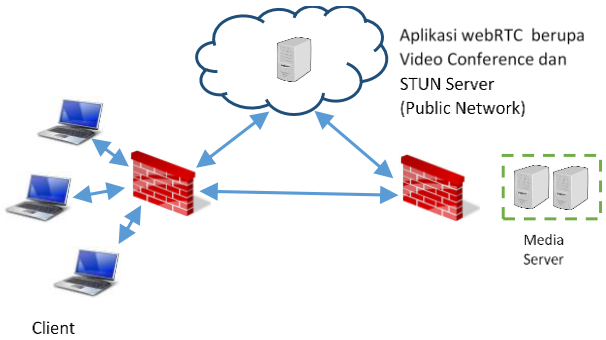
kualitas *video end-to-end*. Hasilnya menunjukkan adanya keuntungan secara signifikan bagi *receiver* dan infrastruktur, dengan sedikit pinalti terhadap kualitas gambar dan penggunaan sumber daya bagi pengirim.

Penelitian kesembilan yang berjudul *Kurento: The Swiss Army Knife of WebRTC Media Servers* [9]. Dalam penelitian tersebut di perkenalkan Kurento, sebagai media server WebRTC yang *open source* dan satu set API *client* yang dimaksudkan untuk memudahkan pengembangan aplikasi dengan kemampuan media yang kaya untuk platform *web* dan *smartphone*.

Penelitian kesepuluh yang berjudul *Demonstration of a P2P assisted video streaming with WebRTC and network coding* [10]. Dalam penelitian tersebut ditunjukkan sebuah aplikasi yang menggunakan WebRTC untuk melakukan koneksi langsung antar browser. Hasil penelitian tersebut menunjukkan bahwa dengan menggunakan WebRTC, VOD P2P dapat dilakukan dan bahkan mengalahkan pengaturan konvensional *client-server* dengan *throughput* sebesar 70%. Selain itu, dengan menerapkan pengkodean jaringan, kinerja jaringan akan semakin meningkat.

Dengan memanfaatkan Kurento yang merupakan hasil dari penelitian pertama serta penelitian – penelitian lain yang berkaitan dengan webRTC, penelitian ini bertujuan untuk mengembangkan media proxy (Kurento Media Server) untuk skala yang lebih besar. Dimana dalam penelitian ini dikembangkan sebuah algoritma yang dapat digunakan untuk mengatur beban kerja diantara beberapa media server kurento terkhusus media server yang terpasang di Universitas Hasanuddin nantinya. Diharapkan dengan menggunakan algoritma tersebut keterbatasan *client* yang menjadi masalah utama aplikasi webRTC serta keterbatasan infrastruktur yang ada di Universitas Hasanuddin dapat teratasi.

2. Perancangan infrastruktur aplikasi webRTC dengan menggunakan media server kurento



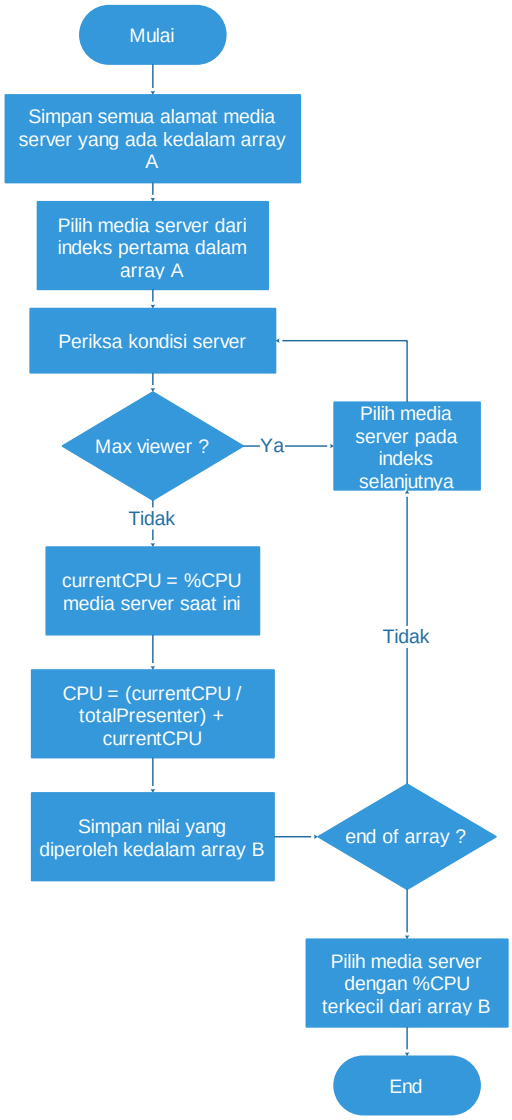
Gambar 1. Desain infrastruktur aplikasi webRTC menggunakan media server

Penelitian yang dilakukan berdasarkan kondisi jaringan di Universitas Hasanuddin saat ini. Dimana setiap fakultas maupun laboratorium terpisah oleh NAT. Sehingga dalam penelitian ini digunakan beberapa server yang bertugas sebagai media server dan sebuah server yang bertugas sebagai STUN sekaligus tempat berjalannya aplikasi webRTC. STUN dan webRTC berada pada server yang sama karena keduanya menggunakan resource yang cukup kecil. Hal ini dikarenakan setelah terjadi komunikasi di antara *client* dan media server, pengiriman media *streaming* tidak akan terjadi antara *client* dan webRTC tetapi antara *client* dan media server. STUN dan aplikasi webRTC membutuhkan sebuah alamat publik sehingga aplikasi webRTC dipastikan dapat diakses oleh *user* yang mungkin berada pada *private network* yang berbeda – beda. Media server dan *client* dapat berada pada *network* yang berbeda karena keduanya akan menggunakan STUN yang telah disediakan sebelumnya. STUN akan me-*reply* alamat ip maupun *port* UDP yang terbaca dari luar NAT sehingga nantinya dapat digunakan untuk saling berkomunikasi. Pada media server akan dipasang *web service* yang nantinya akan digunakan oleh aplikasi webRTC untuk memonitor media server sehingga aplikasi webRTC tersebut dapat menentukan kapan akan menggunakan media server yang tersedia. *Web service* dan aplikasi webRTC yang dijalankan menggunakan bahasa pemrograman javascript, html dan dijalankan dengan menggunakan node.js. Selain perancangan infrastruktur, Penelitian ini juga

difokuskan pada pembuatan aplikasi berbasis webRTC berupa *video conference* guna menunjang media server yang digunakan.

3. Prototipe aplikasi *video conference*

Video conference yang dibuat tidak menggunakan Kurento *group-call* API yang disediakan oleh pengembang Kurento. Hal ini dikarenakan API tersebut hanya didesain untuk menggunakan satu media server Kurento saja. Karena keterbatasan tersebut maka pada penelitian ini dikembangkan pula aplikasi *video conference* yang menerapkan konsep aplikasi *broadcasting* milik Kurento (*one-2-many*). Pada aplikasi *one*-2-*many* yang disajikan oleh pengembang Kurento, terdapat beberapa *peer* / komputer dimana salah satu komputer bertindak sebagai presenter dan komputer lainnya bertindak sebagai *viewer*. Komputer yang bertindak sebagai presenter hanya akan membroadcast media yang dimilikinya. Dan komputer yang bertindak sebagai *viewer* hanya akan melihat media yang dibroadcast oleh presenter. Setiap presenter dan *viewer* akan berada pada *instance* yang sama dan berada pada *pipeline* yang sama di media server Kurento. *Video conference* yang dibangun menggabungkan fitur broadcast yang dimiliki oleh aplikasi tersebut. Dimana setiap *user* yang terhubung pada aplikasi *video conference* akan bertindak sebagai presenter yang akan membroadcast media miliknya (video dan audio) sehingga dapat ditangkap oleh *client* lain yang bertindak sebagai *viewer*. Selain sebagai presenter, *user* tersebut juga bertindak sebagai *viewer* yang akan melihat media yang dikirim oleh *user* lain yang juga bertindak sebagai presenter. Konsep pembuatan *instance* per presenter oleh aplikasi *broadcasting* milik Kurento dianggap sesuai dengan penelitian ini karena setiap presenter akan memiliki saluran komunikasi (*pipeline*) yang berbeda-beda. Hal ini cukup berbeda dengan konsep yang digunakan oleh Kurento *group-call* API dimana pembuatan *pipeline* dilakukan perkelas (*room)* sehingga pembagian pembuatan *pipeline* pada media server yang berbeda – beda menjadi lebih sulit. Selain itu diterapkan pula algoritma yang akan menentukan mekanisme pemilihan media server sebelum sebuah *pipeline* dibuat seperti pada Gambar ke 2 dibawah. Hal ini ditujukan agar beban yang mungkin terjadi pada media server dapat terbagi secara merata pada media – media server yang tersedia.



Gambar 2. Flowchart penyeleksian kandidat media server

4. Hasil dan Pembahasan

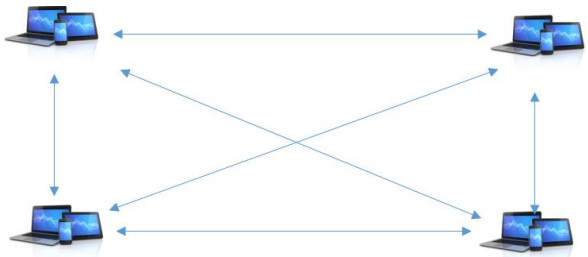
Untuk melakukan pengujian terhadap penelitian yang dilakukan, maka berikut digunakan aplikasi webRTC dengan membandingkan kinerja yang diperoleh saat menggunakan media server dan saat tidak menggunakan media server. Pengujian dilakukan dengan melakukan log terhadap peningkatan penggunaan *bandwidth processor*, maupun

memory dari *client* yang digunakan saat melakukan penambahan jumlah *user* / *participant*. Berikut ini hasil dari pengujian yang telah dilakukan :

4.1. Analisis *video conference* berbasis webRTC tanpa menggunakan media server

Pengujian pertama dilakukan dengan menggunakan sebuah server yang bertindak sebagai *signaling* sekaligus STUN. Adapun perlakuan yang diberlakukan terhadap aplikasi tersebut adalah dengan memberikan batasan – batasan media yang akan dikirim maupun diterima seperti *framerate* yang digunakan oleh webRTC sebesar 15 fps, serta lebar maksimal media sebesar 340px.

Pengujian dilakukan dengan menambahkan *participant / user* satu demi satu ke aplikasi *video conference* yang disediakan.Model komunikasi yang dilakukan untuk pengujian pertama menggunakan topologi *mesh* seperti yang ditunjukkan pada Gambar 3 dibawah ini.



Gambar 3. Model topologi *mesh* pada aplikasi *video conference*

Berdasarkan penelitian yang dilakukan diperoleh hasil pengukuran pada Tabel 1 berikut:

Tabel 1. Tabel peningkatan beban kerja CPU (%)

No	Jumlah *Participant*	CPU *Used* (x)	$x_n - x_{n-1}$
1	2	6.65%	-
2	3	12.94%	6.29%
3	4	18.67%	5.73%
4	5	26.29%	7.62%
5	6	33.82%	7.53%
6	7	43.69%	9.87%
7	8	47.26%	3.57%
8	9	55.55%	8.29%
9	10	64.71%	9.16%

Berdasarkan Tabel diatas, maka diperoleh rata – rata peningkatan penggunaan CPU untuk setiap penambahan 1 *participant / user* sebesar :

$$X = \frac{6.29\% + 5.73\% + 7.62\% + 7.53\% + 9.87\% + 3.57\% + 8.29\% + 9.16\%}{8} = \pm 7.2\%$$

Adapun besar peningkatan penggunaan memori oleh aplikasi *video conference* untuk jumlah *user* yang berbeda dapat dilihat pada Tabel 2 berikut :

Tabel 2. Tabel peningkatan jumlah kebutuhan *memory* (bytes)

No	Jumlah *Participant*	*Memory Used* (m)	$m_n - m_{n-1}$
1	2	40143	-
2	3	50028	9885
3	4	59667	9639
4	5	69811	10144
5	6	79280	9469
6	7	88911	9631
7	8	98668	9757
8	9	109360	10692
9	10	119951	10591

Berdasarkan Tabel diatas, dapat disimpulkan rata – rata peningkatan memori yang digunakan untuk setiap penambahan *user* sebesar ± 9976 KB.

$$m = \frac{9885 + 9639 + 10144 + 9469 + 9631 + 9757 + 10692 + 10591}{8} = \pm 9976\ KB$$

Peningkatan kebutuhan *bandwidth* untuk setiap penambahan *user* diperlihatkan pada Tabel 3 dibawah :

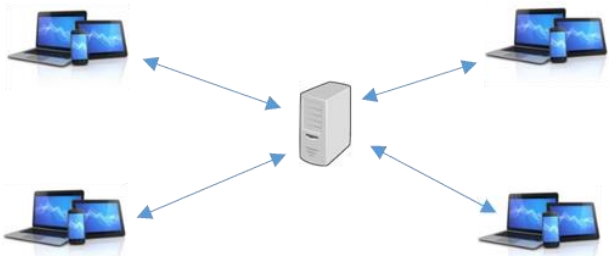
Tabel 3. Tabel peningkatan kebutuhan *bandwidth*(bps)

No	Jumlah *Participant*	bitsSent PerSecond (bps)	bitsReceived PerSecond (bps)
1	2	572540	535636
2	3	1009455	1002808
3	4	1425179	1601384
4	5	2152680	2190642
5	6	2757710	2623765
6	7	3038434	2983911
7	8	3537813	3452398
8	9	4114360	4275704
9	10	4585664	4837820

4.2. Analisis *video conference* berbasis webRTC menggunakan media server

Pengujian selanjutnya dilakukan dengan menggunakan infrastruktur media server yang telah dibuat sebelumnya. Perlakuan yang dilakukan sama dengan pengujian sebelumnya, dimana dilakukan pembatasan berupa lebar maksimal media sebesar 340px ,*framerate* sebesar 15fps. Pengaturan ini dilakukan dengan memodifikasi file kurento-utils.js.

Model komunikasi pada pengujian kedua dapat dilihat pada Gambar dibawah ini :



Gambar 4. Topologi aplikasi *video conference* berbasis *client-server*

Dengan menggunakan batasan yang telah ditentukan sebelumnya, diperoleh data yang menunjukkan peningkatan beban CPU seperti pada Tabel 4 dibawah :

Tabel 4. Tabel peningkatan penggunaan CPU

No	Jumlah *Participant*	CPU *Used* (x)	$x_n - x_{n-1}$
1	2	6.5%	-
2	3	9.4%	2.9%
3	4	11%	1.6%
4	5	13.02%	2.02%
5	6	15.25%	2.23%
6	7	17.71%	2.46%
7	8	17.81%	0.1%
8	9	18.06%	0.25%
9	10	20.18%	2.12%

Berdasarkan Tabel diatas, maka diperoleh rata – rata peningkatan penggunaan CPU untuk setiap penambahan 1 *participant* / *user* sebesar :

$$X = \frac{2.9\% + 1.6\% + 2.02\% + 2.23\% + 2.46\% + 0.1\% + 0.25\% + 2.12\%}{8} = \pm 1.71\%$$

Adapun besar peningkatan penggunaan memori oleh aplikasi *video conference* untuk jumlah *user* yang berbeda dapat dilihat pada Tabel berikut :

Tabel 5. Tabel peningkatan jumlah kebutuhan *memory* (*bytes*)

No	Jumlah *Participant*	*Memory Used* (m)	$m_n - m_{n-1}$
1	2	47145	-
2	3	50982	3837
3	4	57132	6150
4	5	62174	5042
5	6	66980	4806
6	7	70830	3850
7	8	74883	4053
8	9	79548	4665
9	10	83828	4280

Berdasarkan Tabel diatas, dapat disimpulkan rata – rata peningkatan memori yang digunakan untuk setiap penambahan *user* sebesar ± 4585 KB.

$$m = \frac{3837 + 6150 + 5042 + 4806 + 3850 + 4053 + 4665 + 4280}{8} = \pm 4585\ KB$$

Peningkatan kebutuhan *bandwidth* untuk setiap penambahan *user* diperlihatkan pada Tabel 6 dibawah :

Tabel 6. Tabel peningkatan kebutuhan *bandwidth*(bps)

No	Jumlah *Participant*	bitsSent PerSecond (bps)	bitsReceived PerSecond (bps)
1	2	517551	452328
2	3	452420	1142850
3	4	463827	1608481
4	5	585455	2137822
5	6	582077	2612936
6	7	496145	3301510
7	8	568138	3514259
8	9	499425	4008544
9	10	438657	4687501

5. Kesimpulan

Dari penelitian yang telah dilakukan, maka diperoleh kesimpulan :

1. Penggunaan media server dalam lingkup Universitas Hasanuddin merupakan pilihan yang tepat. Hal ini dikarenakan beban CPU serta penggunaan memori yang cukup besar oleh *client*, dapat berkurang. Selain itu dengan menggunakan media server, keterbatasan *bandwidth* yang ada di Indonesia dapat sedikit teratasi. *Gadget – gadget* yang semakin berkembang dengan pesat pun dapat merasakan manfaat webRTC karena beban kerja CPU dapat berkurang secara drastis.
2. Pemilihan desain infrastruktur media server yang tepat dapat mengurangi beban kerja server secara langsung.
3. Pengembangan aplikasi webRTC berbasis media server untuk *device* yang lebih kecil seperti Raspberry Pi dapat terealisasi dengan menerapkan pembagian kerja terhadap beberapa media server.
4. Pemilihan Kurento dengan fitur yang kaya dapat menjadi solusi untuk pengembangan penelitian selanjutnya.

Referensi

[1] Luis L., Miguel P., & Raúl B. (2013). Kurento : a media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC. Institute of Electrical and Electronics Engineers (IEEE).

[2] Luis L., Miguel P., Raúl B., Francisco J., & Jose A. (2013). Catalysing the Success of WebRTC for the Provision of Advanced Multimedia Real-Time Communication Services. Institute of Electrical and Electronics Engineers (IEEE).

[3] Wajdi E. (2013). Models for Multimedia Conference between Browser based on WebRTC. Sixth International Workshop on Selected Topics in Mobile and Wireless Computing.

[4] Kwok-Fai., Ching MY., Liu Y., Cai T., Li L., & Chou W. (2014). A P2P-MCU Approach to Multi-Party Video Conference with WebRTC. International Journal of Future Computer and Communication, 3(5)

[5] Chodorek RR., Chodorek A., Rzym G., & Wadja K. A Comparison of QoS Parameters of WebRTC Videoconference with Conference Bridge Placed in Private and Public Cloud. Institute of Electrical and Electronics Engineers (IEEE).

[6] Rocha H., & Pereira RL.(2017). Hyper-linked communications: WebRTC enabled asynchronous collaboration. Institute of Electrical and Electronics Engineers (IEEE).

[7] Cicco LD., Carlucci G., & Mascolo S.(2017). Congestion Control for WebRTC: Standardization Status and Open Issues. Institute of Electrical and Electronics Engineers (IEEE).

[8] Grozev B., Politis G., Ivov E., Noel T., & Singh V.(2017). Experimental Evaluation of Simulcast for WebRTC. Institute of Electrical and Electronics Engineers (IEEE).

[9] Garcia B., Lopez-Fernandez L., Gallego M., & Gortazar F.(2017). Kurento: The Swiss Army Knife of WebRTC Media Servers. Institute of Electrical and Electronics Engineers (IEEE).

[10] Braun PJ., Ekler P., & Fitzek FHP.(2017). Demonstration of a P2P assisted video streaming with WebRTC and network coding. Institute of Electrical and Electronics Engineers (IEEE).