Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 7, Juli 2018, hlm. 2721-2729 http://j-ptiik.ub.ac.id

Perancangan dan Implementasi Kelas Virtual FILKOM Universitas Brawijaya dengan Memanfaatkan Teknologi WebRTC (*Web Real-Time Communication*)

Rahadiyan Yuniar Rahmanda[1], Eko Sakti Pramukantoro[2], Widhi Yahya[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]failuretechno@yahoo.com, [2]ekosakti@ub.ac.id, [3]widhi.yahya@ub.ac.id

Abstrak

Teknologi *streaming* maupun layanan *video conference* dapat dimanfaatkan untuk menerapkan teknologi *distance learning* seperti kelas virtual. Salah satu *platform* yang dapat digunakan dalam mengembangkan kelas virtual yakni WebRTC. Penelitian ini bermaksud untuk merancang dan mengimplementasikan sistem kelas virtual yang diterapkan untuk FILKOM UB dengan memanfaatkan WebRTC, yang menawarkan solusi akan metode pembelajaran jarak jauh yang dapat dilakukan oleh dosen dan mahasiswa kapanpun dan dimanapun melalui web secara *real-time*. 3 fitur utama kelas virtual yang telah dikembangkan yakni fitur *broadcast* yang memungkinkan dosen dapat melakukan proses *broadcasting* ke mahasiswa, fitur *share file* yang memungkinkan dosen mengirimkan file kepada mahasiswa, dan fitur *text-chat* yang memungkinkan dosen dan mahasiswa dapat saling mengirimkan pesan chat. Pada penelitian ini dilakukan pengujian fungsional yang menunjukkan bahwa semua kebutuhan sistem telah terpenuhi sesuai dengan analisis kebutuhan yang telah didefinisikan. Pada penelitian ini juga dilakukan pengujian kinerja jaringan sistem khususnya pada fitur *broadcast* untuk mengetahui kualitas layanan (QoS) sistem. Hasil pengujian menunjukkan nilai *delay* rata-rata tertinggi pada pengiriman audio sebesar 143,17 ms dan video sebesar 198,82 ms, sedangkan nilai *jitter* rata-rata tertinggi pada pengiriman audio sebesar 240,51 ms dan video sebesar 134,67 ms. Jumlah *packet loss* rata-rata pada pengiriman audio maupun video jauh di bawah 1 %.

Kata kunci: WebRTC, kelas virtual, *real-time*, *peer-to-peer*.

Abstract

Streaming and video conferencing technology can be utilized to implement distance learning technology such as virtual class. One of platforms that can be used in developing a virtual class is WebRTC. This research intends to design and implement virtual class system for FILKOM UB by utilizing WebRTC, which offers solutions to distance learning methods that lecturers and students can do whenever and wherever through the web in real-time. 3 main features of this virtual class are a broadcast feature that allow lecturer to do broadcasting to students, share file feature that allow lecturer to send files to students, and text-chat feature that allow lecturer and students can send chat messages. This research perform functionality testing and results that all of system requirements have been fulfilled in accordance with the analysis of system requirements. This research also perform system network performance testing especially on broadcast feature to know quality of service (QoS) from system. This test results show that highest average delay value on audio delivery is 143.17 ms and video is 198.82 ms. Highest average jitter value on audio delivery is 240.51 ms and video is 134.67 ms. Average packet loss on audio or video delivery is well below 1%.

Keywords: *WebRTC, virtual class, real-time, peer-to-peer*

1. PENDAHULUAN

Kelas virtual merupakan lingkungan belajar dan mengajar dimana peserta dapat berinteraksi, berkomunikasi, melihat dan mendiskusikan presentasi, dan terlibat dengan file pendukung pembelajaran pada satu grup dan semuanya dijalankan dalam lingkungan online. Kelas virtual sering menggunakan media atau aplikasi *video conference* yang memungkinkan beberapa

pengguna terhubung dan berkomunikasi satu sama lain pada waktu yang sama melalui internet, yang memungkinkan mengguna dari mana saja untuk berpartisipasi (Neeson, V. 2017).

Salah satu teknologi yang dapat digunakan untuk mengembangkan kelas virtual yakni WebRTC (*Web Real-Time Communication*). WebRTC merupakan teknologi atau platform komunikasi *real-time* yang dapat dijalankan antar penjelajah web tanpa menggunakan berbagai macam *plug-in* (Virag et al, 2014). WebRTC dapat melakukan komunikasi real-time seperti *teleconference* audio maupun video secara *peer-to-peer* dengan menggunakan penjelajah web tanpa adanya penginstalan plug-in ataupun membutuhkan perangkat lunak pihak ketiga dalam pengoperasiannya (Virag et al, 2014).

Banyak pengembang aplikasi yang memanfaatkan WebRTC untuk bereksperimen dan mengembangkan aplikasi *real-time* khususnya kelas virtual ataupun *e-learning*. Penelitian yang membahas pemanfaatan WebRTC sebagai media pembelajaran kelas virtual yakni dilakukan oleh Nattha Buasri, dkk (Buasri *et al*, 2014). Pada penelitian tersebut, peneliti membahas tentang kelas virtual interaktif berbasis web dengan memanfaatkan teknologi HTML5 dan WebRTC. Dalam penelitian tersebut peneliti mengembangkan kelas virtual interaktif yang memungkinkan pengguna khususnya pelajar dan juga pengajar dapat saling berkomunikasi menggunakan *video conference* dan juga dapat saling menggunakan fitur presentasi, *online whiteboard*, dan juga *screen sharing*. Kemudian terdapat penelitian oleh Karl Bissereth, dkk (Bissereth *et al*, 2014) yang memanfaatkan teknologi WebRTC menggunakan konfigurasi *mesh topology* sebagai modul interaktif koferensi video yang diimplementasikan dengan *platform e-learning*. Kemudian juga terdapat penelitian yang dilakukan oleh Amol Kokane, dkk (Kokanee *et al*, 2014) yang mengembangkan e-learning efektif dengan memanfaatkan WebRTC pada fitur Multimedia Chat.

Penelitian ini bermaksud untuk merancang dan mengimplementasikan sistem kelas virtual dengan memanfaatkan teknologi WebRTC yang dapat digunakan dan diterapkan bagi mahasiswa maupun dosen FILKOM (Fakultas Ilmu Komputer) Universitas Brawijaya. Kelas virtual dapat dimanfaatkan sebagai media pembelajaran jarak jauh yang interaktif dan menyerupai proses belajar mengajar pada kelas konvensional yang biasa dilakukan oleh mahasiswa dan dosen di kelas. Kelas virtual dengan memanfaatkan WebRTC diharapkan dapat digunakan sebagai alternatif proses belajar mengajar jarak jauh dalam melakukan proses belajar di luar kampus atau ketika proses belajar mengajar pada kelas konvensional terpaksa tidak bisa dilaksanakan. Diharapkan dengan kelas virtual ini proses belajar mengajar di luar kampus bisa dilaksanakan serta semua materi bisa tetap disampaikan tanpa mengurangi pemahaman mahasiswa akan materi yang diberikan oleh dosen. Kelas virtual yang dirancang dan dibangun pada penelitian ini memiliki fitur inti yakni fitur *broadcasting* audio dan video secara *real-time* yang dapat dilakukan oleh dosen untuk menyampaikan materi perkuliahan kepada mahasiswa, fitur *share file* yang menungkinkan dosen melakukan pengunggahan file pendukung perkuliahan dan dapat diunduh oleh mahasiswa, serta fitur *text-chat*, yang mengizinkan dosen dan seluruh mahasiswa dapat saling berkomunikasi satu sama lain melalui fitur ini. Sistem kelas virtual juga membutuhkan server node.js untuk menangani proses komunikasi *real-time* pada setiap fiturnya. Sistem yang telah dikembangkan nantinya akan diuji mengenai uji fungsional sistem untuk mengetahui apakah semua fitur-fitur pada kelas virtual dapat bekerja dengan baik sesuai dengan kebutuhan fungsional yang telah didefinisikan. Kemudian juga akan dilakukan pengujian kinerja jaringan sistem kelas virtual untuk mengetahui kualitas layanan atau QoS (*Quality of Service*) khususnya pada fitur *broadcast*. Parameter – parameter yang akan digunakan untuk mengetahui kualitas layanan sistem kelas virtual yakni *troughput*, *delay*, *jitter*, dan *packet loss*. Hasil dari pengujian akan dianalisa untuk mengetahui kualitas layanan dari sistem yang telah dikembangkan.

2. TINJAUAN PUSTAKA

2.1 WebRTC

WebRTC *(Web Real-Time Communication)* merupakan *framework open source* yang mengizinkan komunikasi secara *real-time* antara penjelajah web tanpa menggunakan berbagai macam *plug-in* (Virag *et al*, 2014). Komunikasi secara *real-time* pada WebRTC dapat diakses melalui Javascript API. WebRTC mengijinkan penjelajah web dapat melakukan pertukaran data

aplikasi dan juga melakukan performa *teleconferencing* audio/video secara *peer-to-peer*, tanpa melakukan penginstalan *plug-in* atau perangkat lunak pihak ketiga (Virag *et al*, 2014). WebRTC dirilis Google sebagai proyek *open source* yang telah distandarisasi oleh IETF (*Internet Engineering Task Force*) dan W3C (*World Wide Web Consortium*). Google, Mozilla dan Opera mendukung WebRTC dan terlibat dalam proses pengembangan WebRTC.

WebRTC bertujuan untuk menciptakan suatu aplikasi yang *Rich* dan *real-time* dengan kualitas tinggi. WebRTC dibangun untuk dapat dijalankan pada berbagai penjelajah web dan juga berbagai perangkat meliputi pada komputer personal, perangkat bergerak / *smartphone*, maupun pada *IoT device*, yang mana semua perangkat tersebut dapat saling berkomunikasi satu sama lain menggunakan protokol umum yang sudah ada (www.webrtc.org, 2016). Dengan adanya teknologi ini, pengguna dapat secara langsung berkomunikasi dan bertukar data melalui penjelajah web tanpa memikirkan penginstalan *plug-in* dan lain sebagainya. Teknologi ini sangat mempermudah dan menguntungkan pengguna, yang dapat dijalankan secara langsung kapanpun dan di manapun pengguna berada melalui beberapa peramban web dan juga beberapa *platform* yang berbeda.

2.2 Javascript

JavaScript merupakan bahasa pemrograman yang digunakan untuk menambah interaktivitas dan perilaku dalam suatu halaman web. Javascript juga digunakan untuk memanipulasi elemen pada halaman web, termasuk memanipulasi *style* bahkan penjelajah web itu sendiri (Robbins, J.N, 2012). JavaScrip dapat berkolaborasi dengan HTML dan CSS, sehingga dapat mebuat halaman web menjadi semakin optimal. JavaScript merupakan bahasa yang bersifat *client-side scripting*, yang berarti proses yang dijalankan pada sisi pengguna dan dijalankan secara lokal oleh penjelajah web (Robbins, J.N, 2012). Proses tersebut sangat efektif dan efisien karena proses yang dilakukan bersifat lokal sehingga tidak perlu melakukan *request* ke sisi server untuk menjalankan suatu perintah atau proses. Disamping itu server akan menjadi lebih ringan karena JavaScript dapat menghemat *bandwidth* dan beban dari server.

2.3 Node.js

Node.js merupakan *runtime environtment* yang digunakan untuk membangun aplikasi web yang bersifat *server-side* (Ben. W, 2013). Node.js bersifat *open-source* dan juga *cross-platform*, sehingga dapat digunakan dengan mudah dan dapat beroperasi pada banyak platform atau sistem operasi meliputi FreeBSD, Linux, Microsoft Windows, OS X, dan lain sebagainya.

Node.js menyediakan arsitektur yang bersifat *even-driven* dan *non-blocking I/O API*, sehingga membuat aplikasi menjadi ringan dan efisien (www.nodejs.org, 2016). Arsitektur berbasis *event-driven* yang dikombinasikan dengan arsitektur *non-blocking I/O* sangat cocok untuk membangun aplikasi dengan jumlah pengguna yang berjumlah jutaan. Komunikasi secara *real-time* akan ditangani oleh *event-driven* tanpa membutuhkan banyak memori, sehingga pengembang aplikasi dapat menghabiskan waktu lebih banyak untuk membangun fungsionalitas dari aplikasi (Orsini. L, 2013).

2.4 Quality of Service (QoS)

Quality of Service (QoS) didefinisikan oleh ITU (*International Telecommunications Union*) pada rekomendasi E.800 sebagai "Efek kolektif dari kinerja layanan yang menentukan derajat kepuasan pengguna terhadap suatu layanan".

2.4.1 Parameter QoS (*Quality of Service*)

Beberapa parameter QoS yang dibutuhkan untuk mengukur kinerja jaringan yang menentukan tingkat kepuasan pengguna layanan khususnya pada sistem pengiriman data audio dan video meliputi *throughput*, *delay*, *jitter*, dan *packet loss*.

1. Throughput

Throughput merupakan jumlah pesan sukses yang terkirim melalui kanal komunikasi. Data yang berhasil dikirimkan dapat dikirim melalui *physical link* atau *logical link*, atau dapat melewati node pada jaringan tertentu (Kaur, H. dan Kumar, R, 2017). *Throughput* biasanya diukur dalam satuan *bits per second* (bit/s atau bps).

2. Delay

Delay adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses

yang lama

3. Jitter

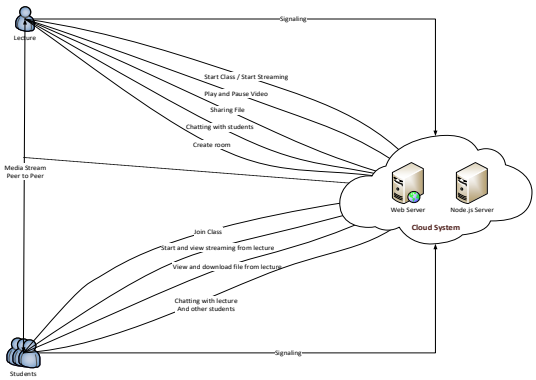
Jitter merupakan perbedaan selang waktu kedatangan antar paket di terminal tujuan. *Jitter* lazimnya disebut variasi *delay*, berhubungan erat dengan *delay* (*latency*), yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan (Yanto, 2013).

4. Packet Loss

Packet loss merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan (TIPHON, 2011). *Packet loss* terjadi ketika satu atau beberapa paket data yang melintasi jaringan gagal mencapai tujuan. *Packet loss* diukur sebagai persentase paket yang hilang dari total paket yang dikirim.

3. PERANCANGAN SISTEM

3.1 Perancangan Arsitektur Sistem



Gambar 1. Arsitektur Sistem

Gambar 1 menunjukkan rancangan arsitektur sistem kelas virtual interaktif FILKOM UB dengan memanfaatkan teknologi WebRTC. Rancangan ini menunjukkan apa saja yang dapat dilakukan oleh sistem kelas virtual dan seperti apa interaksi antar pengguna dengan server dalam melayani sistem kelas virtual.

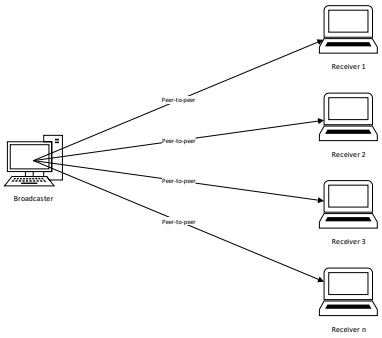
Pada sistem kelas virtual ini terdapat 2 pengguna yang dapat berhubungan dengan sistem, yakni dosen dan mahasiswa. Pada sistem kelas virtual, dosen bertugas sebagai *broadcaster*, dan mahasiswa merupakan *receiver* atau penerima siaran *broadcast* dari dosen yang disiarkan secara *live* atau *real-time*.

Pada sistem kelas virtual ini, terdapat beberapa fitur yang menunjang proses belajar mengajar secara online dan *real-time*. Pada sisi pengguna *broadcaster* atau dosen, dosen dapat membuat *room* kelas virtual tertentu dan dapat melakukan proses *broadcast* ke banyak mahasiswa yang telah bergabung dalam *room* yang telah dibuat oleh dosen. Kemudian pada saat proses *broadcast* berlangsung, dosen dapat melakukan *sharing file* atau mengirimkan berbagai macam file kepada mahasiswa. Dosen juga dapat melakukan *chatting* kepada semua mahasiswa yang bergabung atau *join* pada *room* dengan menanfaatkan fitur text-chat. Pada sisi pengguna *receiver* atau mahasiswa, mahasiswa dapat melakukan join atau bergabung pada suatu *room* yang telah dibuat oleh dosen dan dapat menerima atau menyaksikan siaran broadcast tersebut secara *real-time*. Kemudian mahasiswa juga dapat melihat dan mengunduh file yang telah dikirim oleh dosen. Mahasiswa juga dapat melalukan *chatting* kepada dosen dan juga kepada mahasiswa lain yang tergabung pada satu *room*.

Komunikasi *real-time* pada WebRTC menggunakan model komunikasi *peer-to-peer* dalam melakukan pertukaran audio, video dan data antar pengguna (*peers*). Model komunikasi secara *peer-to-peer* pada komunikasi *real-time* dibutuhkan proses *signaling* sebelum media dan data disalurkan antar *peers*. Proses *signaling* melakukan koordinasi antar *peers* meliputi melakukan pertukaran *session description* antar *peers*, meliputi pertukaran konfigurasi jaringan seperti alamat IP dan *port* yang digunakan untuk komunikasi, informasi konfigurasi media yang mendukung seperti *codec* apa yang didukung oleh setiap *peers*, dan lain sebagainya. Agar proses *signaling* dapat dilakukan, WebRTC memerlukan server *signaling* untuk melakukan pertukaran informasi tersebut. Server Node.js dengan menggunakan modul socket.io berperan dalam menangani proses tersebut. Pengaksesan WebRTC API dapat diakses melalui javascript API yang mana proses tersebut berjalan di sisi pengguna atau peramban web.

Model komunikasi pada fitur *broadcast* ini menggunakan model komunikasi *one-to-many*. Model komunikasi *one-to-many* yang diterapkan pada fitur *broadcast* menggunakan arsitektir atau topologi star dalam proses pengiriman data audio dan video. Arsitektur star ini akan mengirimkan aliran media dari 1 *broadcaster* kepada banyak *receiver*. *Broadcaster* (dosen) akan menyalurkan aliran data audio dan video ke *receiver* (mahasiswa) yang telah masuk ke dalam suatu *room* yang telah ditentukan. Arsiterktur yang digunakan akan menyalurkan

data dari *broadcaster* ke *receiver* secara langsung menggunakan komunikasi *peer-to-peer* (*direct peer-to-peer*). Gambar 2 menunjukkan topologi yang digunakan untuk pengiriman data audio-video pada fitur *broadcast*.



Gambar 2. Topologi Komunikasi fitur *Broadcast*

Server web dan server Node.js berada pada satu lingkungan *cloud* khususnya diimplementasikan dan di-*hosting* pada PaaS (*Platform as a Service*) untuk melayani sistem kelas virtual interaktif FILKOM UB agar dapat diakses secara online melalui jaringan internet.

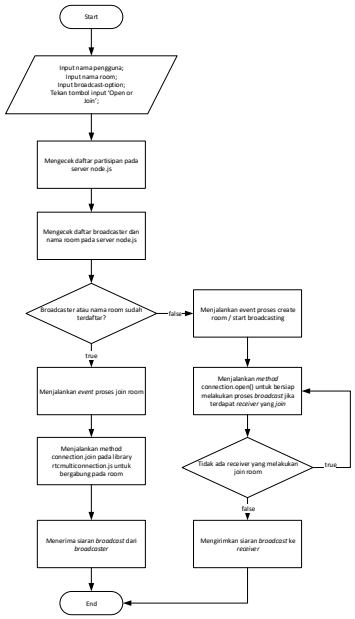
3.2 Perancangan Server Node.js

Server Node.js pada pengembangan aplikasi kelas virtual FILKOM UB berperan sebagai sistem *back-end*, yakni sebagai server yang menangani komunikasi *real-time* pada kelas virtual dan server Node.js ini juga berperan sebagai server HTTP atau server web.

Server Node.js akan melakukan proses *listening* untuk melayani pengguna kelas virtual agar aplikasi web kelas virtual selalu tersedia bagi pengguna ketika pengguna mengakses maupun menjalankan fitur-firur yang ada pada kelas virtual. Server Node.js juga berperan sebagai server web kelas virtual, yakni menyimpan dan menampilkan file-file halaman web, sehingga pengguna dapat mengakses halaman web kelas virtual. Server Node.js akan melakukan pembacaan file dan menampilkan file berupa halaman web ketika pengguna mengakses halaman web kelas virtual.

3.3 Perancangan Fitur *Broadcast*

Proses pada fitur *broadcast* diawali dengan menginputkan nama pengguna dan nama *room*. Untuk memulai proses *broadcasting*, pengguna harus menekan tombol ‘open or join’ untuk memulai proses *broadcasting*. Sebelum proses *broadcasting* berlangsung, sistem kelas virtual diawali dengan menjalankan *event* join-broadcast pada sisi server untuk menentukan peran pengguna tersebut, apakah berperan sebagai *broadcaster* atau sebagai receiver. Jika nama *room* yang diinputkan belum terdaftar, maka pengguna akan berperan sebagai *broadcaster* dan jika nama *room* yang diinputkan sudah terdaftar, maka pengguna akan berperan sebagai *receiver*. Jika pengguna tersebut adalah *broadcaster*, maka sistem pada sisi klien akan menjalankan *event* start-broadcasting dan siap untuk melakukan proses *broadcasting* jika ada *receiver* yang *join* ke dalam *room* tersebut. Jika peran pengguna tersebut adalah *receiver*, maka sistem akan menjalankan *event* join-broadcaster untuk menerima *broadcast* dari *broadcaster* yang dimaksud.

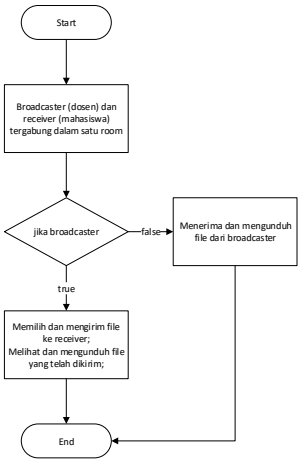


Gambar 3. Flowchart fitur *broadcast*

3.4 Perancangan Fitur Share File

Fitur share file mengizinkan *broadcaster* untuk melakukan pengiriman file kepada receiver yang tergabung atau *join* pada *room*. Sedangkan *receiver* dapat menerima dan mengunduh file yang telah dikirim oleh *broadcaster*. Proses pengiriman file diawali ketika *broadcaster* telah berhasil membuat *room* dan telah terdapat *receiver* yang join ke dalam room, maka *broadcaster* dapat memilih file pada media penyimpanan lokal yang ingin dikirimkan. Setelah memilih file yang dimaksud, maka sistem akan otomatis mengirimkan file tersebut dengan menjalankan *method* connection.send pada *library*

rtcmulticonnection.js dan receiver dapat menerima dan mengunduh file yang terkirim.



Gambar 4. Flowchart Fitur *share file*

3.5 Perancangan Fitur Text-chat

Broadcaster dan *receiver* yang tergabung pada satu *room* dapat melakukan komunikasi melalui pesan chat.

Gambar 5. Fitur *Text-Chat*

Broadcaster dan *receiver* yang berada pada satu *room* dapat saling mengirim pesan chat dengan menginputkan pesan teks pada form input yang telah disediakan dan menekan tombol Enter pada *keyboard* untuk mengirim pesan kepada seluruh partisipan yang tergabung dalam satu *room*.

4. PENGUJIAN

4.1 Pengujian Fungsional

Pengujian fungsional digunakan untuk menguji apakah fitur-fitur yang digunakan oleh sistem sudah berjalan dengan baik atau belum. Item-item yang telah dirancang sesuai daftar kebutuhan pada analisis kebutuhan merupakan acuan yang digunakan dalam melakukan pengujian fungsional. Pengujian dilakukan dengan menjalankan semua fitur-fitur yang ada pada sistem kelas virtual yang telah dikembangkan dan memastikan semua fitur yang telah dikembangkan dapat bekerja dan dapat digunakan dengan baik oleh pengguna.

4.1.1 Hasil Pengujian Fungsional

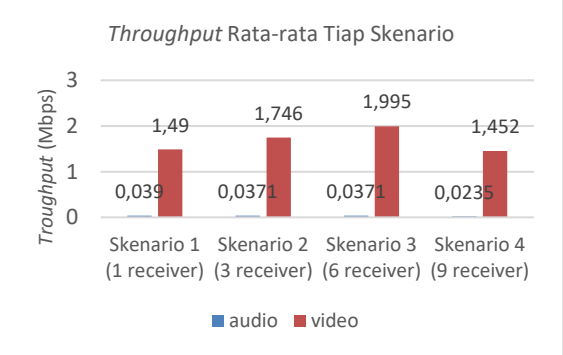
Tabel 1. Hasil Pengujian fungsional

No.	Kebutuhan Fungsional	Status Pengujian
1	Dosen dapat membuat *room* untuk memulai sesi kelas virtual	Terpenuhi
2	Dosen dapat mendapatkan akses ke kamera (misalnya *webcam* pada *notebook* atau kamera depan pada *smartphone*) dan mikrofon yang ada pada perangkat yang sedang digunakan untuk menangkap *media stream* sebelum melakukan proses *broadcast*	Terpenuhi
3	Dosen dapat melakukan *broadcast* ke mahasiswa saat ada mahasiswa melakukan *join* ke dalam *room* yang telah dibuat oleh dosen	Terpenuhi
4	Dosen dapat mengirimkan file atau membagikan file ke pada seluruh mahasiswa yang telah *join* ke dalam *room* yang telah dibuat oleh dosen yang bersangkutan	Terpenuhi
5	Dosen dapat melakukan berkomunikasi atau *chatting* kepada seluruh mahasiswa yang telah *join* ke dalam *room* melalui fitur chat	Terpenuhi
6	Mahasiswa dapat masuk atau *join* ke dalam *room* yang telah dibuat oleh dosen	Terpenuhi
7	Mahasiswa dapat menyaksikan *broadcast* dari dosen setelah masuk atau *join* ke dalam *room*	Terpenuhi
8	Mahasiswa dapat menerima file yang telah dibagikan oleh dosen dan mengunduh file tersebut	Terpenuhi
9	Mahasiswa dapat saling berkomunikasi kepada dosen dan seluruh mahasiswa lain yang telah *join* ke dalam *room* melalui fitur *chat*	Terpenuhi

4.2 Pengujian Kinerja Jaringan

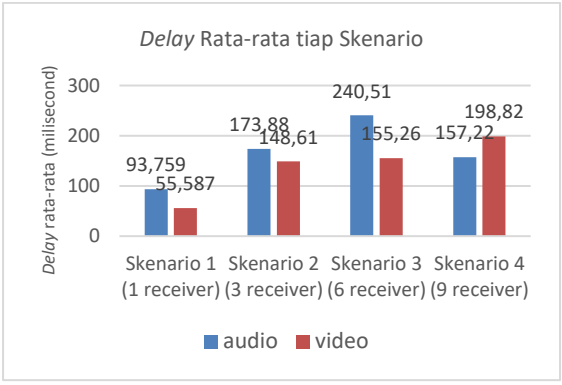
Pengujian sistem kelas virtual dilakukan dengan menguji kinerja jaringan sistem yang telah dikembangkan. Pengujian ini dilakukan untuk mengetahui kualitas layanan / *Quality of Service* (QoS) dari sistem. Terdapat 4 paremeter pengukuran yang diuji yaitu *throughput*, *delay*, *jitter* dan *packet loss*. Pada pengujian ini dilakukan dengan melakukan proses *broadcasting* dari *broadcaster* ke receiver dengan 4 skenario pengujian yang berbeda-beda pada jumlah *receiver*-nya. Jumlah *receiver* yang diuji yakni 1 *receiver*, 3 *receiver*, 6 *receiver* dan 9 *receiver*. Pengujian dilakukan dengan melakukan *capture* jaringan menggunakan *tools* RTC Event Log yang telah disediakan oleh peramban web Google Chrome untuk meng-*capture* jaringan dan aliran media pada WebRTC.

4.2.1 Hasil Pengujian Kinerja Jaringan



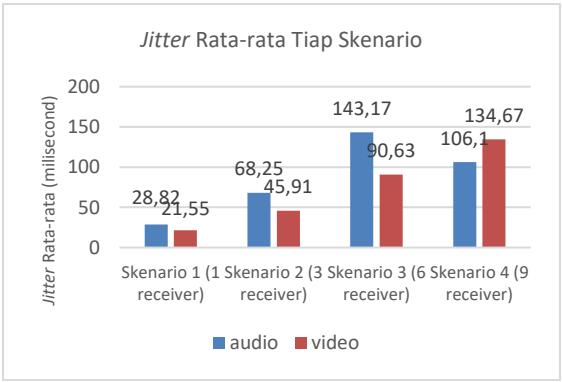
Gambar 6. Grafik *Throughput* Rata-rata Tiap Skenario

Throughput rata-rata tiap skenario dari skenario 1 higga skenario 4 didapat nilai *throughput* setiap *receiver* yang hampir sama. *Troughput* rata-rata yang didapat pada pengiriman audio berkisar antara 0.0235 Mbps hingga yang tertinggi 0.039 Mbps. Sedangkan *throughput* rata-rata yang didapat pada pengiriman video sebesar 1,452 Mbps hingga yang tertinggi 1,995 Mbps.



Gambar 7. Grafik *Delay* Rata-rata Tiap Skenario

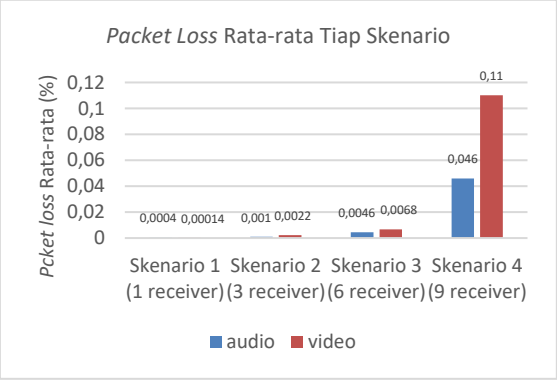
Dapat dilihat dalam grafik pada Gambar 7, *delay* rata-rata tiap skenario dari skenario 1 higga skenario 3 didapat nilai *delay* audio yang cenderung meningkat, tetapi nilai *delay* audio pada skenario 4 mengalami penurunan nilai *delay*. Sedangkan nilai *delay* pada pengiriman video, dari skenario 1 hingga skenario 4 nilainya cenderung meningkat. Dapat disimpulkan bahwa semakin banyak jumlah *receiver* yang menerima media stream yakni audio dan video dari broadcaster, maka semakin besar *delay* yang didapat. Pada Skenario 1 dengan jumlah 1 *receiver*, *delay* yang didapat dapat dikategorikan sangat bagus menurut standar TIPHON dengan rata-rata *delay* di bawah 150 ms. Sedangkan pada skenario 2, skenario 3 dan skenario 4, nilai *delay* dikategorikan masih bagus menurut standar TIPHON dengan nilai rata-rata *delay* di bawah 300 ms. Dapat disimpulkan bahwa dengan jumlah 1 *receiver* hingga 9 *receiver*, fitur broadcast pada kelas virtual dengan memanfaatkan teknologi WebRTC ini masih dalam kategori memenuhi atau dapat diterima oleh pengguna untuk komunikasi *real-time*.



Gambar 8. Grafik *Jitter* Rata-rata Tiap Skenario

Dapat dilihat dalam grafik pada Gambar 8, *jitter* rata-rata tiap skenario dari skenario 1 hingga skenario 4 didapat nilai jiiter yang cenderung meningkat. Dapat disimpulkan

bahwa semakin banyak jumlah *receiver* yang menerima media stream yakni audio dan video semakin besar jiiter atau variasi *delay* yang terjadi. Pada skenario 1 dengan jumlah 1 *receiver*, *jitter* yang didapat dapat dikategorikan bagus menurut standar TIPHON. Pada skenario 2 dengan jumlah 3 *receiver jitter* yang didapat juga dikategorikan bagus menurut standar TIPHON. Sedangkan pada skenario 3 didapat *jitter* yang sangat besar pada pengiriman audio, dengan rata-rata *jitter* audio sebesar 143,26 ms dan dikategorikan jelek menurut standar TIPHON, sedangkan pada pengiriman video nilai rata-rata *jitter* yang didapat sebesar 90,63 ms dan dikategorikan sedang menurut standar TIPHON. Sedangkan pada skenario 4 didapat *jitter* rata-rata pada pengiriman audio sebesar 106,1 ms dan dikategorikan sedang menurut standar TIPHON, sedangkan pada pengiriman video nilai rata-rata *jitter* yang didapat sebesar 137,67 ms dan dikategorikan jelek menurut standar TIPHON. Dapat disimpulkan bahwa nilai jiiter yang terjadi pada skenario 3 dengan jumlah 6 *receiver* dan skenario 4 dengan jumlah *receiver* dikategorikan jelek menurut standar TIPHON yang berarti mengurangi nilai kualitas layanan yang diharapkan.



Gambar 9. Grafik *Packet Loss* Rata-rata Tiap Skenario

Dapat dilihat dalam grafik pada Gambar 9, *packet loss* yang didapat pada skenario 1 hingga skenario 3 dengan jumlah masing-masing 1 *receiver*, 3 *receiver* dan 6 *receiver*, mengalami *packet loss* yang sangat sedikit. Terjadi kenaikan *packet loss* yang signifikan pada skenario 4 dengan jumlah 9 *receiver*, tetapi jumlah *packet loss* juga tergolong sangat sedikit dengan jumlah *packet loss* jauh di bawah 1 %. Nilai rata-rata *packet loss* dari semua skenario dikategorikan sangat bagus menurut standar TIPHON.

5. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem kelas virtual, maka dapat diambil kesimpulan sebagai berikut:

1. Perancangan Sistem Kelas Virtual FILKOM UB dengan memanfaatkan teknologi WebRTC dirancang dengan mendefinisikan kebutuhan fungsional sistem. Perancangan sistem juga dilakukan dengan merancang arsitektur sistem, topologi sistem pada fitur kelas virtual dan merancang fitur-fitur pada yang dibutuhkan oleh sistem kelas virtual.
2. Implementasi Sistem Kelas Virtual FILKOM UB dengan memanfaatkan teknologi WebRTC berhasil diimplementasikan. Tiga Fitur utama dari sistem kelas virtual telah berhasil diimplementasikan dengan memanfaatkan teknologi WebRTC dan menggunakan *library* rtcmulticonnection.js sebagai *library* WebRTC yang dapat membantu dalam pengimlementasian kelas virtual. Serta server node.js mampu melayani sistem kelas virtual, meliputi sebagai server untuk komunikasi pada fitur-fitur kelas virtual maupun sebagai server web kelas virtual.
3. Hasil dari pengujian fungsional didapat semua kebutuhan sistem yang telah didefinisikan pada analisis kebutuhan telah terpenuhi dan sistem kelas virtual dapat dijalankan dengan baik. Pengujian kinerja jaringan sistem kelas virtual dilakukan dengan mengukur kualitas layanan / *Quality of Service* (QoS) sistem khususnya pada fitur *broadcast* dengan model komunikasi *peer-to-peer*. Berikut hasil analisis pengujian yang telah dilakukan:
   a. *Throughput* rata-rata tiap skenario dari skenario 1 higga skenario 4 didapat nilai *throughput* setiap *receiver* yang hampir sama. *Troughput* rata-rata yang didapat pada pengiriman audio berkisar antara 0.0235 Mbps hingga yang tertinggi 0.039 Mbps. Sedangkan pada video sebesar 1,452 Mbps hingga yang tertinggi 1,995 Mbps.
   b. *Delay* rata-rata yang dari hasil analisis dapat dikategorikan sangat bagus menurut standar TIPHON dengan *delay* rata-rata pada pengiriman audio sebesar 93,75 ms dan pada pengiriman video sebesar 55,58 ms. Sedangkan pada pengujian dengan 3 *receiver*, 6 *receiver*,

dan 9 *receiver*, nilai *delay* rata-rata yang didapat dikategorikan bagus menurut standar TIPHON. Nilai *delay* rata-rata tertinggi pada pengiriman audio yakni sebesar 143,17 ms dan pada pengiriman video sebesar 198,82 ms.

c. *Jitter* rata-rata pada pengujian dengan jumlah 1 *receiver* dan 3 *receiver*, dapat dikategorikan bagus menurut standar TIPHON dengan nilai jiiter rata-rata pada pengiriman audio masing-masing sebesar 28,82 ms dan 68,25 ms, dan video sebesar 21,55 ms dan 45,91 ms. Nilai *jitter* pada pengujian dengan jumlah 6 *receiver* dan 9 *receiver*, dikategorikan jelek menurut standar TIPHON. Nilai *jitter* rata-rata tertinggi pada pengiriman audio yakni sebesar 240,51 ms dan video sebesar 134,67 ms.
d. *Packet loss* rata-rata yang didapat pada seluruh pengujian dikategorikan sangat bagus menurut standar TIPHON, dengan *packet loss* rata-rata jauh di bawah 1 %.

DAFTAR PUSTAKA

Bissereth, *et al*, 2014. *An Interactive Video Conferencing Module for e-Learning using WebRTC*. United States: Illinois State University.

Buasri, *et. al*. 2014. *Web-based Interactive Virtual Classroom using HTML5-based Technology*. Thailand: Mahidol University.

Kokanee, *et al*. 2014. *Effective E-learning Using 3D Virtual Tutors and WebRTC Based Multimedia Chat*. India: National Institute of Technology Karnataka.

Virag, *et al*. 2014. *Browser-Based Medical Visualization System*. Romania: University of Timisoara

WebRTC. 2016. WebRTC. [online] Tersedia di: https://webrtc.org/.

WebRTC. 2014. Architecture. [online] Tersedia di: https://webrtc.org/architecture/