ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

PENGEMBANGAN MEKANISME ADAPTIF LIVE STREAMING MULTIMEDIA PEER-TO-PEER BERBASIS WEBRTC UNTUK KONDISI JARINGAN DINAMIS

DIMAS RANGGA F
5111100096

Dosen Pembimbing I
WASKITHO WIBISONO, S.Kom., M.Eng., Ph.D.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2015

FINAL PROJECT – KI141502

DEVELOPMENT ADAPTIVE LIVE STREAMING MULTIMEDIA PEER-TO-PEER MECHANISM BASED ON WEBRTC FOR DYNAMIC NETWORK CONDITIONS

DIMAS RANGGA F
5111100096

Supervisor
WASKITHO WIBISONO, S.Kom., M.Eng., Ph.D.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
Sepuluh Nopember Institute of Technology
Surabaya, 2015

LEMBAR PENGESAHAN

PENGEMBANGAN MEKANISME ADAPTIF LIVE STREAMING MULTIMEDIA PEER-TO-PEER BERBASIS WEBRTC UNTUK KONDISI JARINGAN DINAMIS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
Pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
DIMAS RANGGA F
NRP: 5111 100 096

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Waskito Wibisono, S.Kom., M.Eng., Ph.D. .....
NIP: 19741022 200003 1 001 (Pembimbing 1)

SURABAYA

JUNI 2015

PENGEMBANGAN MEKANISME ADAPTIF LIVE STREAMING MULTIMEDIA PEER-TO-PEER BERBASIS WEBRTC UNTUK KONDISI JARINGAN DINAMIS

Nama Mahasiswa	**: Dimas Rangga F**
NRP	**: 5111 100 096**
Jurusan	**: Teknik Informatika FTIF ITS**
Dosen Pembimbing 1	**: Waskito Wibisono, S.Kom., M.Eng., Ph.D.**

Abstrak

Komunikasi secara *real time* dengan multimedia yaitu video dan audio telah umum di masyarakat. Dengan komunikasi tersebut mereka dapat secara langsung berkomunikasi dengan mengirimkan video dan audio yang dapat menunjukkan keberadaan seseorang atau kegiatan yang sedang dilakukan. Tantangan yang ada sekarang ini yaitu bagaimana mengatur kualitas pengiriman multimedia tersebut sesuai dengan kondisi *traffic* jaringan internet. Oleh karena itu untuk mengatasi permasalahan dilakukan pengembangan mekanisme adaptif streaming multimedia berdasarkan kondisi jaringan yang ada.

Pada tugas akhir ini, dilakukan pengembangan adaptif live streaming multimedia peer-to-peer berdasarkan kondisi jaringan yang ada. Kondisi jaringan yang dijadikan sebagai acuan yaitu *jitter*, *packet-loss*, dan *bandwidth*. Ketiga acuan tersebut digunakan untuk menentukan konstrain multimedia seperti resolusi dari video streaming. Resolusi video yang telah ditentukan diharapkan tidak mengurangi kualitas layanan multimedia streaming.

Hasil uji coba terhadap adaptif multimedia streaming menunjukkan bahwa kualitas video yaitu PSNR dan frame rate

lebih baik dibandingkan dengan tidak menggunakan metode adaptif multimedia streaming.

Kata kunci: Adaptif, jaringan dinamis, live streaming , peer-to-peer, webrtc

DEVELOPMENT ADAPTIVE LIVE STREAMING MULTIMEDIA PEER-TO-PEER MECHANISM BASED ON WEBRTC FOR DYNAMIC NETWORK CONDITIONS

Student's Name : Dimas Rangga F
Student's ID : 5111 100 096
Department : Informatics Engineering, FTIF-ITS
First Advisor : Waskito Wibisono, S.Kom., M.Eng., Ph.D.

Abstract

Real time multimedia communication with video and audio have been common in the world. With that communications they can directly communicate with sending video and audio that can indicate the location of a person or person activities. The current challenge is how to organize delivery quality of multimedia compatible with the conditions of the Internet traffic network. Therefore, to overcome the problems should be develop of adaptive mechanisms multimedia streaming based on network conditions that exist.

At this final project, develop of adaptive live streaming multimedia peer-to-peer based on the existing network conditions. Network conditions that serve as a references is jitter, packet-loss and bandwidth. The third reference is used to determine the constraints of multimedia such as resolution of the streaming video. Video resolution that has been determined was expected not to reduce the quality of streaming multimedia services.

Results of tests on adaptive streaming multimedia show that the video quality and frame rate that PSNR is better than not using adaptive streaming multimedia.

Keywords: Adaptive, dynamic network, live streaming, peer-to-peer, webrtc

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Pengembangan Mekanisme Adaptif Live Streaming Multimedia Peer-to-Peer Berbasis WebRTC untuk Kondisi Jaringan Dinamis".

Harapan dari penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata bagi kampus Teknik Informatika, ITS, dan bangsa Indonesia.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT yang telah memberikan kesehatan dan kemampuan kepada penulis untuk menyelesaikan Tugas Akhir ini.
2. Kedua orang tua penulis yaitu Bapak Soemartono dan Ibu Denny Yuliani yang tak henti-hentinya memberikan semangat dan kasih sayang serta selalu memberikan doa kepada penulis.
3. Saudara kandung penulis yaitu Okky Savitri Febiyani dan Dhio Sandi Yulianto yang telah memberikan motivasi agar penulis lulus tepat waktu.
4. Bapak Waskito Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen pembimbing penulis yang selalu meluangkan waktu dan memberikan kepercayaan, dukungan, nasihat, serta semangat kepada penulis.
5. Ibu Nanik Suciati, S.Kom., M.Kom., Dr.Eng. selaku ketua jurusan Teknik Informatika ITS.
6. Ibu Umi Laili Yuhana, S.Kom., M.Sc. selaku dosen wali penulis yang telah membimbing penulis selama kuliah di Teknik Informatika ITS.

7. Inka Ayu Permanasari yang selalu setia menemani dan memberikan dukungan serta motivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
8. Teman-teman admin Laboratorium AJK yaitu Harum, Vivi, Romen, Uyung, Samidh, serta admin lain yang penulis tidak dapat menulis satu per satu yang selalu membantu dan memberikan kecerian selama pengerjaan Tugas Akhir ini.
9. Teman-teman Kabinet HMTC Bersahabat yang selalu memberikan semangat, motivasi dan kepedulian kepada penulis sehingga dapat menyelesaikan Tugas Akhir ini.
10. Teman-teman konski yaitu Komang, Tebe, Vendo, Anom, Akbar, Pak De, Madi, Ujek yang selalu menjadi sahabat penulis selama kuliah di ITS.
11. Teman-teman mahasiswa Teknik Informatika ITS angkatan 2011 yang selama ini berjuang bersama-sama selama empat tahun dan telah menjadi keluarga penulis di Teknik Informatika ITS.
12. Juga kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penulis untuk menyelesaikan Tugas Akhir ini.

Kesempurnaan tentu masih jauh tercapai pada Tugas Akhir ini, sehingga penulis mengharapkan saran dan kritik yang membangun dari pembaca untuk perbaikan ke depan. Semoga Tugas Akhir ini dapat bermanfaat bagi perkembangan ilmu pengetahuan dan bagi semua pihak.

Surabaya, Juni 2015

Dimas Rangga F

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam Tugas Akhir ini yang meliputi latar belakang, rumusan masalah, tujuan dan manfaat pembuatan Tugas Akhir, serta metodologi dan sistematika penulisan buku Tugas Akhir ini.

1.1.Latar Belakang

Komunikasi secara *real time* dengan multimedia yaitu video dan audio telah umum di masyarakat. Dengan komunikasi tersebut mereka dapat secara langsung berkomunikasi dengan mengirimkan video dan audio yang dapat menunjukkan keberadaan seseorang atau kegiatan yang sedang dilakukan. Komunikasi tersebut memudahkan seseorang untuk bertatap muka dengan tidak perlu bertemu orang yang akan diajak berkomunikasi, tetapi hanya cukup menggunakan media seperti smartphone atau komputer. Contoh aplikasi yang dapat berkomunikasi secara *real time* dengan multimedia yaitu Skype[1], Facetime[2], dll.

Kualitas layanan atau yang disebut dengan *Quality Of Service* (QoS) pada komunikasi video dan audio merupakan bagian terpenting dari sistem multimedia terdistribusi, karena dengan adanya parameter kualitas layanan tersebut, kita dapat menentukan nilai yang merepresentasikan kualitas layanan yang baik, tetapi hal tersebut tidaklah mutlak selama manusia yang melihatnya terlihat baik. Kualitas dari suatu layanan dibedakan menjadi dua yaitu QoS berdasarkan end-user dan QoS berdasarkan jaringan. QoS berdasarkan jaringan adalah bagaimana suatu aplikasi bisa mengatur pengiriman multimedia dibutuhkan agar pengguna tetap bisa mengirimkan video dan audio secara *real time* dan

[1] Program penyedia layanan komunikasi multimedia dalam smartphone dan komputer (http://www.skype.com/en/)

[2] Program penyedia layanan komunikasi multimedia dikhususkan bagi pengguna Apple (https://www.apple.com/ios/facetime)

mengantisipasi adanya delay yang terlalu tinggi antar paket yang dikirimkan [1].

Tantangan yang ada sekarang ini yaitu bagaimana mengatur kualitas pengiriman multimedia tersebut sesuai dengan kondisi *traffic* jaringan internet. Seperti yang diketahui, kondisi jaringan internet tidak selalu lancar. Banyak sekali hambatan yang menyebabkan kondisi jaringan tidak stabil. Untuk komunikasi *real time* membutuhkan pengaturan terhadap multimedia sesuai dengan kondisi jaringan agar mengantisipasi adanya delay yang terlalu lama sehingga mengurangi kualitas dari layanan komunikasi tersebut.

Kondisi jaringan dinamis adalah keadaan suatu koneksi jaringan yang berubah-ubah sesuai dengan kondisi *traffic* jaringan. Keadaan yang berubah-ubah tersebut yang menyebabkan pengiriman paket pada jaringan memerlukan waktu yang berbeda-beda. Kondisi tersebut juga dapat mempengaruhi komunikasi multimedia pada jaringan. Berbagai hambatan yang mempengaruhi komunikasi multimedia pada jaringan antara lain*jitter, packet-loss* dll. *Jitter* adalah variasi dari delay atau selisih dari delay yang pertama dengan delay selanjutnya. Besarnya nilai *jitter* akan sangat dipengaruhi oleh variasi beban *traffic* dan besarnya tumbukan antar paket [2]. *Jitter* berhubungan dengan erat dengan *latency*. Jaringan yang memiliki *latency* yang stabil tidak memiliki variasi delay atau *jitter*. *Jitter* dan delay adalah rintangan utama untuk aplikasi multimedia.

Aplikasi berbasis web yang dapat berkomunikasi secara *real time* membutuhkan *plug-in* (seperti flash player) agar dapat digunakan. WebRTC [3] merupakan teknologi komunikasi *real time* antar browser yang menerapkan peer-to-peer arsitektur. WebRTC mulai banyak digunakan dan meninggalkan aplikasi yang memerlukan *plug-in* pada browser. Dengan estimasi dari kinerja komunikasi data yang ada, pengiriman paket multimedia dari aplikasi WebRTC dapat diatur seperti resolusi dan fps (*frame per second*) dari video. Semakin kecil resolusi dari video, semakin kecil paket yang dikirim, dan semakin cepat pula pengiriman

paketnya. Kondisi *traffic* jaringan yang ada seperti *jitter*, *payload* (kecepatan pengiriman paket), dan *packet-loss* dijadikan acuan untuk mengatur pengiriman multimedia tersebut. Semakin baik kondisi *traffic* jaringan, maka semakin baik pula pengiriman yang dilakukan. Sebaliknya jika kondisi *traffic* jaringan jelek, maka semakin lambat pula pengiriman yang dilakukan.

Dalam tugas akhir ini, komunikasi secara *real time* dengan multimedia diimplementasikan dengan pembuatan aplikasi menggunakan WebRTC berdasarkan pada kondisi *traffic* jaringan. Dengan mengembangkan mekanisme adaptif *live streaming*, aplikasi ini dapat mengatur konstrain dari video dan audio agar mengantisipasi adanya delay yang terlalu lama antar pengiriman antar paketnya serta tidak mengurangi kualitas layanan dari multimedia streaming.

1.2.Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana membuat sambungan peer to peer menggunakan WebRTC pada web browser?
2. Bagaimana mendapatkan dan mengatur sumber media seperti video dan audio untuk dibagikan kepada peer lain?
3. Bagaimana estimasi *jitter, packet-loss, dan bandwidth* pada kondisi kualitas koneksi jaringan dinamis?
4. Bagaimana mengatur konstrain video berdasarkan kondisi *traffic* jaringan?

1.3.Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi ini berbasis web dengan bahasa pemrograman PHP dan menggunakan framework CodeIgniter.

2. Uji coba dilakukan pada web browser yang mendukung sistem layanan multimedia WebRTC yaitu Google Chrome 23 (atau versi yang lebih baru).
3. Uji coba dilakukan pada jaringan lokal *Ethernet* dan Wi-Fi.
4. Uji coba dilakukan pada hubungan dua node.

1.4.Tujuan dan Manfaat

Tujuan pembuatan tugas akhir ini yaitu untuk mengembangkan *live streaming* multimedia yang adaptif terhadap kondisi jaringan dinamis berbasis webRTC sehingga dapat mengantisipasi adanya delay yang terlalu lama dalam pengiriman antar paketnya.

Manfaat dari hasil pembuatan tugas akhir ini yaitu untuk mengantisipasi kendala *live streaming* pada kondisi jaringan dinamis. Serta tidak mengurangi layanan kualitas (QoS) dari streaming multimedia.

1.5.Metodologi

a. Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan untuk pembuatan tugas akhir. Pendahuluan ini terdiri atas latar belakang diajukannya tugas akhir, permasalahan yang diangkat, batasan masalah, tujuan dan manfaat dibuatnya tugas akhir ini. Selain itu juga dijelaskan tentang tinjauan pustaka sebagai referensi pendukung untuk mengerjakan tugas akhir tersebut yaitu WebRTC, Media Stream, dan Statistik Jaringan. Pada Sub Bab Metodologi menjelaskan tentang mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Jadwal kegiatan yang dilakukan juga dilampirkan dalam proposal tugas akhir ini agar tepat waktu.

b. Studi Literatur

Pada studi literature ini, yang akan dipelajari untuk pembuatan tugas akhir ini adalah WebRTC yang merupakan teknologi komunikasi *real time* dengan arsitektur peer-to-peer. WebRTC mampu berkomunikasi dengan multimedia secara *real time* antar pengguna web browser tanpa *plug-in.* Multimedia yang ada seperti audio dan video merupakan Media Stream yang ada di browser. Dan untuk mengantisipasi adanya delay yang terlalu tinggi antara paket satu dengan yang lain, maka diperlukan pengukuran kinerja dari jaringan untuk mengatur konstrain video dan audio.

c. Analisis dan Desain Perangkat Lunak

Fitur dari aplikasi ini yaitu:

1. Koneksi peer-to-peer antar browser.
2. Video dan Audio Stream.
3. Pengaturan konstrain video dan audio berdasarkan kondisi jaringan.

d. Implementasi Perangkat Lunak

Aplikasi ini akan dibangun dengan bahasa pemrograman PHP dan menggunakan *framework* CodeIgniter. Aplikasi ini akan dibangun dengan menggunakan *Integrated Development Environment* (IDE) Sublime Text 2.0.2.

e. Pengujian dan Evaluasi

Pengujian dari aplikasi ini akan dilakukan dengan menggunakan 1 server dan 4 virtual host dari 2 komputer, yang masing-masing komputer mempunyai 2 virtual host. Virtual host yang dijadikan pengujian diberikan kondisi jaringan yang dinamis, misal dengan menambahkan packet loss dan bandwidth limit.

f. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Metodologi
    f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.6.Sistematika Penulisan

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

1. Bab I. Pendahuluan

Bab ini berisi latar belakang, permasalahan, tujuan, batasan permasalahan, metodologi, dan sistematika penulisan.

2. Bab II. Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar teori penunjang yang digunakan untuk mendukung penyelesaian Tugas Akhir.

3. **Bab III. Perancangan Perangkat Lunak**

Bab ini berisi tentang perancangan sistem, diagram alir, dan perancangan antarmuka yang akan dibuat.

4. **Bab IV. Implementasi Perangkat Lunak**

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *pseudocode* dan *screenshot* aplikasi.

5. **Bab V. Evaluasi dan Uji Coba**

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian fungsionalitas dan pengujian performa dalam beberapa skenario.

6. **Bab VI. Penutup**

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini menjelaskan mengenai teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Bab ini bertujuan untuk memberikan gambaran secara umum mengenai teori serta alat bantu yang digunakan pada implementasi perangkat lunak pada tugas akhir ini.

2.1.WebRTC

WebRTC *(*Web *Real-Time Communication)* [3] adalah sebuah standar baru yang memungkinkan browser berkomunikasi secara *real time* dengan menggunakan arsitektur peer-to -peer. Hal ini berkonsen pada audio / video (dan data) komunikasi peer-to - peer antara HTML5 browser. Ini merupakan evolusi di dunia aplikasi web, karena untuk pertama kalinya, pengembang web mampu untuk membangun *real time* aplikasi multimedia tanpa plug-in**.** WebRTC membutuhkan STUN *(Session Traversal Utilities for NAT)* server agar tiap pengguna dari aplikasi dapat saling terhubung. TURN *(The Traversal Using Relay around NAT)* server merupakan fitur dari STUN server yang menempatkan TURN server soalah-olah berada di tengah-tengah kedua pengguna yang mengurus tentang pengiriman multimedia seperti video dan audio.

API *(Application Programming Interface)* [4] adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh *programmer* saat membangun sistem perangkat lunak untuk sistem operasi tertentu. API memungkinkan *programmer* menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. API dapat menjelaskan cara sebuah tugas *(task)* dilakukan. Dalam pemrograman prosedural seperti bahasa C, aksi biasanya dilakukan dengan media pemanggilan fungsi. Karena itu, API biasanya menyertakan penjelasan dari fungsi yang disediakan. WebRTC API berupa javascript dan memiliki sejumlah fungsi

yang dapat digunakan untuk berinteraksi dengan aplikasi web browser.

2.2.Node.js

Node.js [5] adalah *open source*, *cross-platform runtime environment* pada sisi server dan aplikasi jaringan. Node.js merupakan aplikasi yang ditulis dalam JavaScript, dan dapat dijalankan pada OS X, Microsoft Windows, Linux, FreeBSD, NonStop dan IBM i. Node.js menggunakan mesin V8 JavaScript Google untuk mengeksekusi kode, dan sebagian besar dari modul dasar yang ditulis dalam JavaScript. Node.js berisi perpustakaan built-in untuk memungkinkan aplikasi untuk bertindak sebagai server Web tanpa software seperti Apache HTTP Server atau IIS.

Node.js memberikan arsitektur event-driven dan non-blocking I/O API yang dapat mengoptimalkan throughput aplikasi dan skalabilitas. Teknologi ini biasanya digunakan untuk aplikasi web real-time seperti WebRTC.

2.2.1. Peer

Peer adalah modul dari Node.js yang dibangun untuk membantu koneksi peer.js klien pada aplikasi website peer-to-peer. Peer digunakan untuk *session metadata* dan *candidate signaling.* Ada dua koneksi yang bisa dibuat oleh modul peer tersebut yaitu koneksi data, dan koneksi media seperti audio dan video [6].

Sebelum koneksi dilakukan, modul peer harus dijalankan terlebih dahulu pada server. Sebuah koneksi membutuhkan identitas sebagai pengenal seperti IP pada komputer. Ketika peer ingin berhubungan, peer membutuhkan identitas peer yang dituju. Peer akan menerima *event* `connection` untuk koneksi data atau `call` untuk koneksi media ketika peer menerima permintaan untuk saling berhubungan. Data dari masing-masing peer dikirim melalui websocket.

2.2.2. Media Stream Recorder

Media Stream Recorder adalah modul dari Node.js yang dibangun untuk merekam audio atau video media stream. Media stream recorder bertujuan untuk menyediakan mekanisme yang sangat sederhana dimana pengembang dapat merekam media stream dari perangkat media pengguna dan langsung menggunakannya dalam aplikasi web, daripada harus melakukan operasi encoding pengguna pada data PCM mentah, dll [7].

Media Recorder bekerja dengan `getUserMedia()` untuk menangkap data media. Dalam sebuah aplikasi sederhana, fungsi `MediaRecorder.start()` dipanggil untuk mulai merekam. Kemudian *event* `dataavailable` digunakan ketika data tersedia dan peramban siap untuk mengambil. Data yang direkam berupa Blob, data attribut dari `dataavailable`.

2.3. Media Stream

Media Stream [3] adalah representasi dari aliran data audio dan / atau video. Ini berfungsi untuk mengelola tindakan pada media stream, seperti menampilkan konten, merekam, atau mengirimnya ke remote peer. Sebuah media stream bisa berasal dari webcam, mikrofon, dll. Untuk membuat dan menggunakan media stream lokal, aplikasi web harus meminta akses dari pengguna melalui fungsi `getUserMedia()`. Aplikasi harus menentukan jenis media - audio atau video – yang akan dipakai. Setelah aplikasi selesai, pengambilan media bisa diakhiri dengan memanggil fungsi `stop()`. Video pada WebRTC umunya menggunakan codec VP8 dan H.264. Konstrain meliputi rasio kamera, audio dan video *frame rate*, resolusi video, *bandwidth* yang digunakan, dll.

2.4. MySQL

MySQL [8] adalah sistem manajemen *database* SQL yang bersifat Open Source dan paling populer saat ini. Sistem *database* MySQL mendukung beberapa fitur seperti multithreaded, multi-user, dan SQL *database* managemen sistem (DBMS). *Database* ini dibuat untuk keperluan sistem *database* yang cepat, handal dan mudah digunakan.

Pada MySQL terdapat *logical model* yang terdiri dari *databases, tables, views, rows,* dan *columns* untuk membantu penggunaan data yang terstruktur di lingkungan programer. MySQL juga mendukung sistem klien server, sehingga data dapat terpusat di server dan dapat diakses oleh banyak klien.

Dalam tugas akhir ini MySQL digunakan untuk manajemen data user sebagai pengguna dari aplikasi WebRTC seperti user, room, dll.

2.5. PHP

PHP [9] merupakan bahasa pemrograman *script server-side* yang didesain untuk pengembangan web. Selain itu, PHP juga bisa digunakan sebagai bahasa pemrograman umum. PHP disebut bahasa pemrograman *server-side* karena PHP diproses pada komputer server. Hal ini berbeda dibandingkan dengan bahasa pemrograman *client-side* seperti JavaScript yang diproses pada web browser (client).

Kode PHP dapat dicampur dengan kode HTML, atau dapat digunakan dalam kombinasi dengan berbagai mesin template dan kerangka web. Kode PHP biasanya diproses oleh interpreter PHP, yang biasanya diimplementasikan sebagai modul asli web server atau *Common Gateway Interface* (CGI) *executable*. Setelah kode PHP ditafsirkan dan dilaksanakan, server web mengirimkan output yang dihasilkan ke klien, biasanya dalam bentuk bagian dari halaman web yang dihasilkan; misalnya, kode PHP dapat menghasilkan halaman Web kode HTML, gambar, atau data

lainnya. PHP juga telah berkembang untuk menyertakan *Command Line Interface* (CLI) dan dapat digunakan dalam aplikasi grafis.

2.6. JavaScript

JavaScript [10] merupakan bahasa pemrograman yang umum digunakan dalam pemrograman web. Bahasa pemrograman ini didukung pada berbagai peramban, salah satunya Google Chrome. Karena sudah umum digunakan, banyak peramban modern berbasis *desktop* maupun *mobile* saat ini memanamkan dukungan *JavaScript*.

JavaScript merupakan bahasa pemrograman yang berjalan pada sisi klien (*client-side scripting*). Hal ini menyebabkan setiap eksekusi perintah dilakukan oleh peramban dimana pengguna mengakses situs. Penggunaan *JavaScript* sendiri berdampingan dengan HTML dan CSS, dimana *JavaScript* dapat digunakan untuk memanipulasi konten dan desain dari situs.

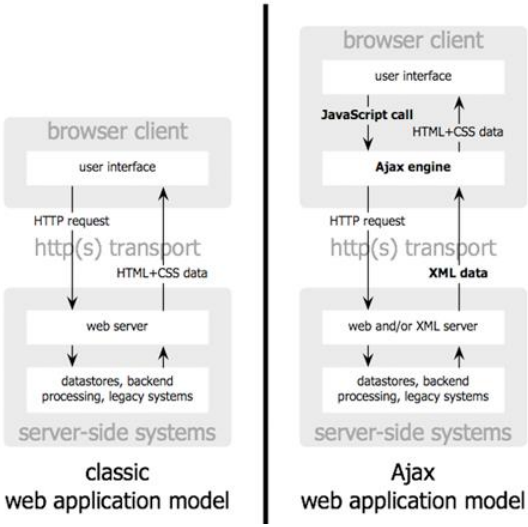
2.7. *Asynchronous JavaScript and XML* (AJAX)

AJAX [11] merupakan salah satu teknik pengembangan *website* yang memungkinkan halaman situs dapat menerima konten secara dinamis melalui server. Pertukaran data antara pengguna dengan server dilakukan secara asinkron, sehingga tidak mengganggu tampilan situs. Implementasi AJAX sendiri memanfaatkan gabungan beberapa komponen, yaitu:

- Situs dengan halaman XHTML dan CSS.
- Tampilan dinamis dan interaktif dengan memanfaatkan *Document Object Model* (DOM).
- Pertukaran dan manipulasi data menggunakan XML dan XSLT.
- Menerima data secara asinkron menggunakan XMLHttpRequest
- JavaScript untuk menggabungkan seluruh komponen tersebut.

Gambar 2.4 menunjukkan bagaimana perbandingan antara aplikasi *web* klasik dengan aplikasi *web* dengan menggunakan AJAX. Aplikasi *web* tanpa AJAX melakukan permintaan ke *web*

server kemudian memberikan respon dalam bentuk halaman HTML, sehingga pengguna akan mengalami perpindahan halaman. Berbeda jika situs mengimplementasi AJAX, permintaan ke *web server* dilakukan melalui perantara JavaScript dengan memanggil objek XMLHttpRequest. *Web server* akan mengirimkan respon berupa data XML yang kemudian diolah untuk ditampilkan ke dalam bentuk HTML dan CSS.



Gambar 2.1. Perbandingan Antara Aplikasi *Web* Klasik dengan Aplikasi *Web* Berbasis AJAX [13]

Meskipun objek yang digunakan adalah XMLHttpRequest, namun pada perkembangannya respon yang diolah tidak selalu berupa XML. Saat ini, jenis respon yang populer digunakan adalah dalam format JSON (JavaScript *Object Notation*) yang umumnya digunakan bersamaan dengan jQuery. Selain dalam bentuk teks yang telah terformat, respon juga dapat berupa *plain text* yang dibuat dalam format sesuai keinginan *programmer*.

2.8. Kondisi Jaringan Dinamis

Kondisi jaringan dinamis adalah keadaan suatu koneksi jaringan yang berubah-ubah sesuai dengan kondisi *traffic* jaringan. Keadaan yang berubah-ubah tersebut yang menyebabkan pengiriman paket pada jaringan memerlukan waktu yang berbeda-beda. Kondisi tersebut juga dapat mempengaruhi komunikasi multimedia pada jaringan. Untuk mengantisipasi adanya delay yang terlalu lama antar paket yang dikirimkan, diperlukan adanya mekanisme untuk mengatur pengiriman paket multimedia. Mekanisme tersebut bisa diterapkan dengan mengukur kinerja jaringan untuk mendapatkan *jitter*, *packet-loss*, dll. Nilai dari pengukuran tersebut kemudian dijadikan acuan untuk mengatur konstrain video dan audio yang akan dikirim.

2.8.1. Pengukuran Kinerja Jaringan

Sebuah komunikasi *real time* [3] juga membutuhkan mekanisme untuk mendapatkan pengukuran dari kinerjanya. pengukuran itu sederhana seperti mengetahui berapa byte data yang dikirim, atau berapa kecepatan pengiriman data tiap detiknya. Dalam pengukuran ini terdapat beberapa nilai yang dijadikan acuan untuk mengatur konstrain video dan audio yang disesuaikan dengan kondisi *traffic* jaringan seperti *jitter*[3] dan *througput* (kecepatan pengiriman data Kbps). Secara umum *jitter* dan *packet-loss* bisa dihitung.
Estimasi *jitter* pada RTP [5] akan dijelaskan sebagai berikut.

$$D(i,j) = (Rj - Ri) - (Sj - Si) \quad (1)$$

[3] Variasi delay atau selisih dari delay yang pertama dengan delay selanjutnya

$$J(i) = J(i-1) + \frac{|D(i-1,i)| - J(i-1)}{16} \quad (2)^4$$

Keterangan:
D = Delay
J = Jitter
R = Waktu dari paket diterima (dalam unit RTP *timestamp*)
S = Waktu paket dikirim (dalam unit RTP *timestamp*)

Estimasi persentasi *packet-loss* pada RTP [5] akan dijelaskan sebagai berikut dimana PL adalah *packet-loss*.

$$PL(i) = \sum PL - PL(i-1) \times 100\% \quad (3)$$

Pengaturan konstrain terbaik sesuai kondisi jaringan dijelaskan pada Tabel 1.1.

Tabel 1.1 Bandwidth resolusi video VP8 [3]

No.	Video Resolusi	Codec	Min Bandwidth	Max Bandwidth
1.	720p	VP8	1	2
2.	480p	VP8	0,5	1
3.	360p	VP8	0,1	0,5

2.9. *Network Emulation*

Network Emulation [12] adalah teknik memanipulasi jaringan kompiter sehingga dapat mengemulasikan kondisi jaringan yang diinginkan untuk menilai kinerja, memprediksi

[4] Parameter 1/16 memberikan *noise reduction ratio* dengan tetap menjaga tingkat konvergensi

dampak perubahan atau mengoptimalkan jaringan yang ada. Sistem operasi Linux memiliki fungsi *network emulation* yang disebut dengan netem. Netem yang ada pada Linux saat ini dapat mengemulasikan variabel delay, *packet loss*, dll.

Netem pada Linux dikontrol dengan perintah *tc* yang merupakan bagian dari paket *iproute2*. Berikut adalah contoh-contoh perintah *network emulation* menggunakan perintah *tc*.

2.9.1. Emulasi *Jitter*

Karena *jitter* adalah variasi delay, maka berikut merupakan perintah dari *network emulation*. Berikut merupakan contoh yang paling sederhana untuk memberikan variasi *delay* sebesar 100ms ± 25ms untuk seluruh paket yang dikirim menggunakan *interface* eth0 dari *local Ethernet*.

tc qdisk add dev eth0 root netem delay 100ms 25ms

Berikut adalah penjelasan tiap perintah tersebut diambil dari *Linux manual page*:

1. *tc* adalah perintah yang digunakan untuk memanipulasi *traffic* di Linux.
2. *qdisc* (singkatan dari "*queueing discipline*") merupakan tempat antrian paket sebelum paket tersebut dikirimkan ke *interface* yang dituju.
3. *add* adalah lanjutan dari perintah *qdisc* yang bertujuan untuk menambahkan *qdisc* ke dalam *interface* yang dituju (eth0).
4. *root* menandakan bahwa *qdisc* yang digunakan berjenis *classless*.
5. *netem* adalah perintah untuk memanipulasi *traffic* jaringan.
6. *delay* adalah lanjutan perintah netem yang menciptakan *delay* jaringan sebesar nilai yang ditentukan.
7. 100ms merupakan nilai dari *delay* yang telah ditentukan.
8. 25ms merupakan nilai pengurangan dan penambahan dari *delay* yang telah ditentukan.

2.9.2. Emulasi *Packet-Loss*

Berikut adalah contoh perintah untuk mengemulsikan *packet loss*.

tc qdisc add dev eth0 root netem loss 0.1%

Perintah tersebut akan menciptakan *packet-loss* sebesar 0.1% dari total keseluruhan paket yang dikirimkan.

2.9.3. Emulasi *Bandwidth*

Berikut adalah contoh perintah untuk mengemulsikan *bandwidth*.

tc qdisc add dev eth0 root handle tbf rate 500kbit buffer 1600 limit 3000

Perintah tersebut akan menciptakan pembatasan pada bandwidth dengan nilai 500kbit, buffer 1600 paket, dan maksimal 3000 paket.

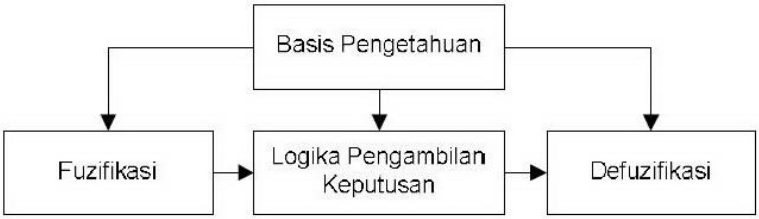
2.10. Logika Fuzzy

Logika Fuzzy [13] adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Saat logika klasik menyatakan bahwa segala hal dapat diekspresikan dalam istilah biner (0 atau 1, hitam atau putih, ya atau tidak), logika fuzzy menggantikan kebenaran boolean dengan tingkat kebenaran.

Logika Fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan", dan "sangat". Logika ini berhubungan dengan set fuzzy dan teori kemungkinan.

Logika fuzzy dan logika probabilitas secara matematis sama. Keduanya mempunyai nilai kebenaran yang berkisar antara 0 dan 1, namun secara konsep berbeda. Logika fuzzy berbicara mengenai "derajat kebenaran", sedangkan logika probabilitas mengenai "probabilitas, kecenderungan". Karena kedua hal itu berbeda,

logika fuzzy dan logika probabilitas mempunyai contoh penerapan dalam dunia nyata yang berbeda.



Gambar 2.2. Struktur Dasar Logika Fuzzy

Pada Gambar 2.2 merupakan struktur dasar logika fuzzy. Kendali logika Fuzzy dilakukan dalam tiga tahap, yaitu fuzzifikasi, evaluasi aturan dan defuzzifikasi. Komponen Fuzifikasi berfungsi untuk memetakan masukan data tegas ke dalam himpunan Fuzzy menjadi nilai Fuzzy dari beberapa variabel linguistik masukan.
Ada beberapa hal yang perlu diketahui dalam memahami sistem fuzzy, diantaranya:

1. Variabel fuzzy : Variabel fuzzy merupakan variabel yang hendak dibahas dalam suatu sistem fuzzy. Contoh: umur, temperatur, permintaan, dsb.
2. Himpunan fuzzy : Himpunan fuzzy merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel fuzzy. Contoh: Variabel error, terbagi menjadi 7 himpunan fuzzy, yaitu: NB (Negative Big), NM (Negative Medium), NS (Negative Small), Z (Zero), PS (Positive Small), PM (Positive Medium), PB(Positive Big).
3. Semesta pembicaraan : Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel fuzzy. Semesta pembicaraan merupakan himpunan bilangan riil yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya

nilai semesta pembicaraan ini tidak dibatasi batas atasnya. Contoh:

- Semesta pembicaraan untuk variabel error: [-10 +10]
- Semesta pembicaraan untuk variabel temperatur: [0 40].

4. Domain himpunan fuzzy : Domain himpunan fuzzy adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan fuzzy. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan riil yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif. Contoh:
   - Negative Big = [-∞ -6]
   - Negative Medium = [-10 -3]
   - Negative Smal l = [-6 0]
   - Zero = [-3 3]
   - Positive Small = [0 6]
   - Positive Medium = [3 10]
   - Positive Big = [6 ∞]

2.10.1. Fuzzyfikasi

Fuzzifikasi adalah proses yang dilakukan untuk mengubah variabel nyata menjadi variabel fuzzy, ini ditujukan agar masukan kontroler fuzzy bisa dipetakan menuju jenis yang sesuai dengan himpunan fuzzy. Pemetaan dilakukan dengan bantuan model dari fungsi keanggotaan agar dapat diketahui besar masukan tersebut (derajat keanggotaan). Terdapat beberapa jenis penggambaran fungsi keanggotaan, antara lain Gaussian, Segitiga, Trapesium, Bahu, dll.

2.10.2. Aturan Fuzzy

Rules (aturan-aturan), merupakan pengetahuan prosedural yang menghubungkan informasi yang diberikan dengan tindakan (action). Struktur rule, secara logika menghubungkan satu atau

lebih antecedent (atau premises) yang berada pada bagian IF, dengan satu atau lebih consequents (atau conclusions/kesimpulan) pada bagian THEN. Misalnya: IF Warna bola itu merah THEN Saya suka bola itu. Sebuah rule dapat memiliki multiple premise yang tergabung dengan menggunakan operasi logika (AND, OR). Bagian Konklusi dapat berupa kalimat tunggal atau gabungan dengan menggunakan operasi logika (AND) dan dapat pula memiliki kalimat ELSE.

2.10.3. Defuzzyfikasi

Input dari proses defuzzifikasi adalah suatu himpunan fuzzy yang diperoleh dari komposisi aturan fuzzy, sedangkan output yang dihasilkan merupakan suatu bilangan tegas pada domain himpunan fuzzy tersebut. Sehingga jika diberikan suatu himpunan fuzzy dalam range tertentu, maka harus dapat diambil suatu nilai crisp tertentu sebagai output.

Dalam Tugas Akhir ini logika fuzzy digunakan untuk menentukan konstrain media seperti resolusi video yang dikirim. Konstrain dari logika fuzzy didapatkan dari estimasi kondisi jaringan yang ada.

2.11. *Peak Signal to Noise Ratio* (PSNR)

Peak Signal to Noise Ratio [14] adalah ekspresi untuk rasio antara nilai maksimum yang mungkin dari sinyal tersebut dan nilai dari distorsi yang mempengaruhi kualitas representasinya. PSNR biasanya diukur dengan decibel. PSNR biasanya digunakan untuk mengukur kualitas antara data asli dan data yang telah dikompresi. Ketika membandingkan codec kompresi, PSNR adalah sebuah pendekatan persepsi manusia untuk kualitas rekonstruksi. Meskipun PSNR yang lebih tinggi secara umum menunjukkan bahwa rekonstruksi yang berkualitas tinggi, dalam beberapa kasus

tidak mungkin. Ini hanya berlaku bila membandingkan kedua codec yang sama atau konten yang sama.

Peak Signal Noise Ratio merupakan tolak ukur yang berguna untuk mengevaluasi kinerja objektif metrik kualitas video baru. Karena perhitungan PSNR sangat tergantung pada estimasi yang tepat dari keselarasan spasial, keselarasan temporal, keuntungan, dan tingkat offset antara urutan video diproses dan urutan video asli, metode pengukuran untuk PSNR idealnya mencakup metode untuk melakukan prosedur kalibrasi ini. Metode perhitungan PSNR dalam Rekomendasi ini memiliki keuntungan dari otomatis menentukan kemungkinan nilai PSNR tertinggi untuk video yang diberikan urutan selama rentang pergeseran spasial dan temporal . Hanya satu pergeseran temporal diperbolehkan untuk semua frame di urutan video seluruh diproses. Caranya yaitu dengan delay video yang konstan antara pengirim dan penerima.

Untuk menentukan PSNR, terlebih dahulu harus ditentukan nilai MSE (Mean Square Error). MSE adalah nilai error kuadrat rata-rata antara citra asli dengan citra kompresi. Dalam suatu pengembangan dan pelaksanaan rekonstruksi gambar diperlukan perbandingan antara gambar hasil rekonstruksi dengan gambar asli. Ukuran umum yang digunakan untuk tujuan ini adalah *Peak Signal to Noise Ratio* (PSNR). Nilai PSNR yang lebih tinggi menyiratkan kemiripan yang lebih erat antara hasil rekonstruksi dan gambar asli. PSNR didefinisikan sebagai berikut:

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right) \quad (4)$$

$$PSNR = 20 \log_{10} \left(\frac{MAX}{\sqrt{MSE}} \right) \quad (5)$$

$$PSNR = 20 \log_{10} MAX - 10 \log_{10} MSE \quad (6)$$

Dimana MSE dinyatakan sebagai mean square error yang didefinisikan sebagai berikut.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)] \quad (7)$$

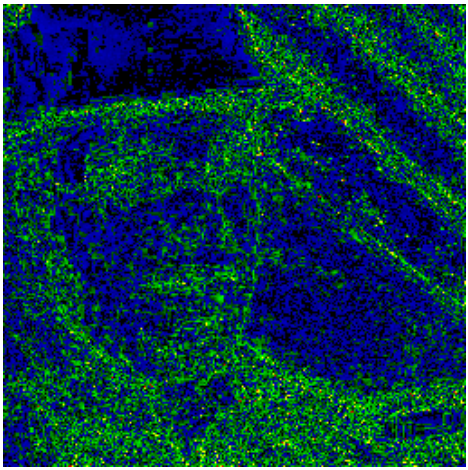
Dimana MAX adalah nilai maksimum dari piksel sebuah gambar.

Ketika piksel menggunakan 8 bit per sampel, maka nilai maksimumnya adalah 255. Secara umum, ketika sampel direpresentasikan menggunakan PCM linier dengan B bit per sampel MAX adalah 2^B-1. Untuk gambar berwarna dengan tiga nilai RGB per piksel, definisi PSNR sama kecuali MSE adalah jumlah atas semua perbedaan nilai kuadrat dibagi dengan ukuran gambar dan tiga.

Berikut merupakan contoh dari perhitungan PSNR dari kedua gambar yaitu Gambar 2.3 dan Gambar 2.4. Perbedaan dari kedua gambar ditunjukkan pada Gambar 2.5. Hasil perhitungan PSNR dari kedua gambar adalah 26,0635. Yang mempengaruhi nilai tersebut adalah noise yang terdapat pada gambar hasil kompresi. Semakin kecil hasil dari perhitungan PSNR, maka kualitas dari gambar akan semakin buruk.

Gambar 2.3. Gambar Asli

Gambar 2.4. Gambar Hasil Kompresi



Gambar 2.5. Gambar Hasil Perhitungan PSNR

BAB III
PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai dasar perancangan perangkat lunak yang akan dibuat dalam tugas akhir ini. Secara khusus akan dibahas mengenai deskripsi umum aplikasi, perancangan proses, alur, serta gambaran implementasi perangkat lunak.

3.1. Deskripsi Umum Sistem

Pada tugas akhir ini dibangun sistem komunikasi *real time* multimedia peer-to-peer berbasis WebRTC (Web *Real Time Communication*) yang dapat mengatur konstrain media (video dan audio) berdasarkan kondisi jaringan. Sistem akan mencatat data dari masing-masing peer yang sedang berkomunikasi mulai dari *jitter, packet-loss* dan *bandwidth*. Data yang dicatat dapat berubah secara dinamis sesuai dengan kondisi jaringan pada saat ini. Data yang ada digunakan untuk menentukan konstrain media seperti resolusi dari video sehingga sistem akan menentukan konstrain yang tepat untuk masing-masing peer untuk mengantisipasi adanya delay yang terlalu lama antar pengiriman antar paketnya.

Perangkat lunak yang dibangun berbasis web dengan *framework* CodeIgniter. Perangkat lunak dibangun pada sisi server dan memanfaatkan Node.js dengan modul Peer. Modul tersebut berfungsi untuk inisialisasi jaringan peer kepada klien serta untuk menghubungkan peer satu dengan yang lain. Inisialisasi yang dilakukan oleh klien kepada server membutuhkan adanya identitas. Identitas tersebut nantinya digunakan untuk saling berhubungan dengan peer lain. Identitas dan informasi dari setiap klien akan disimpan pada *database* ketika melakukan registrasi.

Perangkat lunak yang dibangun pada sisi klien terdiri dari dua bagian yaitu peer server dan peer klien. Perangkat lunak yang dibagun memanfaatkan API dari WebRTC yang berbasis javascript. Inisialisasi koneksi peer dari klien ke server dilakukan dengan mengirimkan identitas dan meminta ijin kepada pengguna

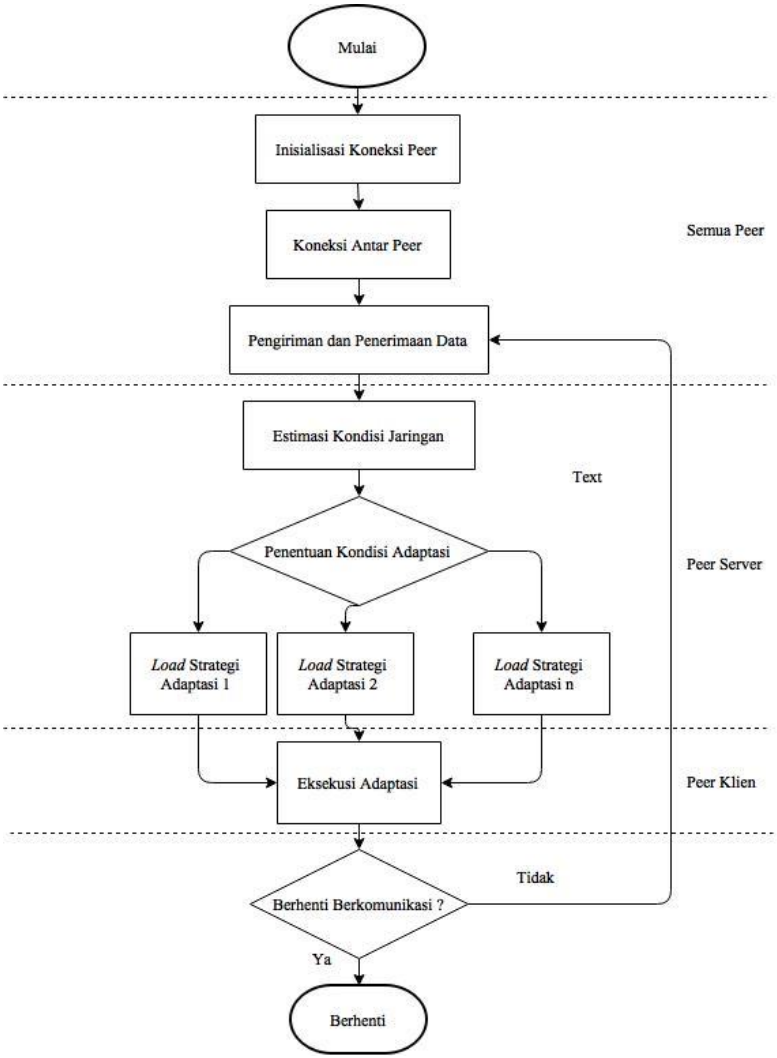
untuk memperbolehkan perangkat lunak untuk mengakses perangkat multimedia serta menampilkan hasil *capture* dari perangkat multimedia pada tampilan perangkat lunak. Ketika antar peer saling berhubungan, peer akan mengirimkan data multimedia seperti video dan audio. Pada saat pengiriman data tersebut dilakukan estimasi kondisi jaringan dari peer klien. Peer server mencatat hasil yang didapat dari pengecekan kondisi jaringan tersebut. Hasil yang dicatat berupa *jitter, packet-loss* dan *bandwidth*. Kemudian hasil tersebut dihitung menggunakan logika fuzzy. Logika fuzzy berfungsi untuk menentukan konstrain terbaik dari pengiriman data yang dilakukan oleh peer klien kepada peer server.

Untuk tahapan penentuan konstrain, logika fuzzy memerlukan beberapa member dan klasifikasi dari masing-masing member serta hasil-hasil yang ingin didapat dari logika fuzzy. Hal ini dimaksudkan agar setiap kondisi jaringan mendapatkan hasil yang sesuai. Hasil yang didapatkan berupa resolusi video (*width* dan *height*). Kemudian hasil tersebutakan dikirim kepada peer klien.

Pada sisi peer klien, hasil konstrain yang telah dikirimkan oleh peer server dieksekusi. Pada saat kondisi jaringan berubah, peer klien akan menunjukkan hasil berupa perubahan konstrain video yang dikirim oleh klien dari perhitungan logika fuzzy pada peer server. Perubahan yang terjadi pada peer klien juga berpengaruh pada penerimaan paket dari peer klien. Hasil yang didapat dari perhitungan logika fuzzy tersebut digunakan untuk mengantisipasi adanya delay yang terlalu lama dari pengiriman antara paket satu dengan yang lainnya dan tidak mengurangi QoS.

3.2. Arsitektur Umum Sistem

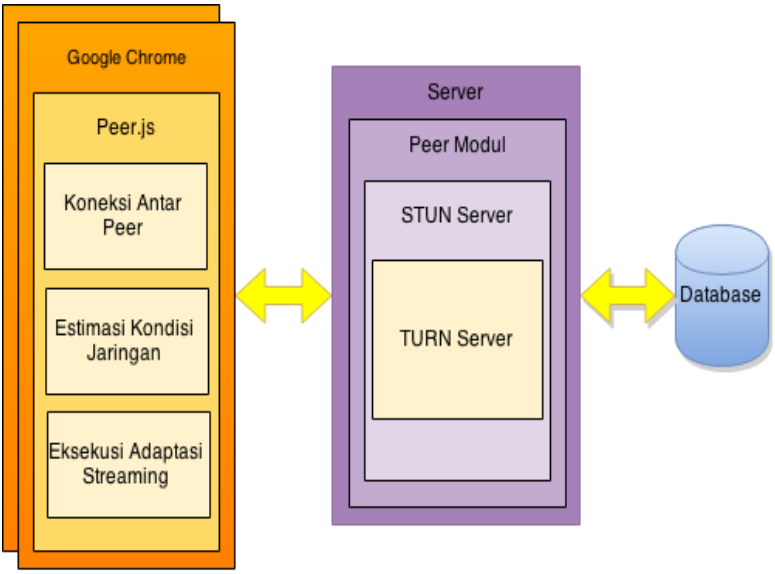
Agar dapat menjalankan fungsinya, maka alur kerja dari kesatuan aplikasi ini dirancang seperti pada Gambar 3.1.



Gambar 3.1. Gambaran Umum Alur Kerja Aplikasi

Berdasarkan Gambar 3.1, alur kerja aplikasi ini dijabarkan sebagai berikut:

1. Pengguna melakukan inisialisasi koneksi peer dengan mengirimkan identitas yang telah didaftarkan sebelumnya kepada server. Server akan mengecek identitas dari klien pada *database*.
2. Ketika peer satu ingin berhubungan dengan peer lain maka, peer akan mengirimkan identitas peer lain yang ingin dihubungkan ke server. Server akan menghubungkan kedua peer tersebut melalui TURN Server yang merupakan fitur dari STUN Server yang seolah-olah pengguna satu dapat berhubugan dengan yang lain dalam satu jaringan.
3. Masing-masing peer akan saling mengirim dan menerima data ketika peer saling berhubungan. Data yang dikirim seperti video dan audio hasil *capture* dari perangkat multimedia.
4. Ketika peer klien mengirim data, peer server melakukan estimasi kondisi jaringan dari peer klien. Estimasi dilakukan dengan menggunakan API dari WebRTC.
5. Peer server akan melakukan penentuan kondisi adaptasi. Penentuan kondisi adaptasi dilakukan menggunakan logika fuzzy dengan member yaitu *jitter*, *packet-loss* dan *bandwidth*. Hasil yang didapat dari penentuan adaptasi ini berupa konstrain video seperti resolusi.
6. Hasil dari penentuan strategi adaptasi digunakan untuk mengubah konstrain dari video peer klien yang akan dikirim selanjutnya.
7. Peer klien mengirim data yang sudah ditentukan berdasarkan estimasi kondisi jaringan. Peer klien menerima data yang telah ditentukan dari peer server.



Gambar 3.2. Arsitektur Sistem Aplikasi

Pada Gambar 3.2 merupakan gambaran arsitektur dari pembangunan aplikasi. Aplikasi terdiri dari dua modul utama. Modul pertama adalah peramban yang berfungsi sebagai antarmuka pengguna. Pada sisi pengguna terdapat peer.js klien untuk membantu koneksi peer ke server serta untuk menghubungkan peer satu dengan yang lain. Estimasi kondisi jaringan juga terdapat pada sisi pengguna karena peer merupakan hubungan antar pengguna. Setelah estimasi kondisi jaringan dilakukan, maka eksekusi adaptasi dilakukan.

Modul selanjutnya adalah server yang terdiri atas Node.js yang berfungsi untuk menjalankan modul Peer untuk membantu koneksi peer.js klien pada aplikasi website peer-to-peer. Di dalam modul Peer terdapat fitur STUN server dan TURN server. TURN server merupakan fitur dari STUN server yang berfungsi menghubungkan peer satu dengan yang lain seolah-olah TURN server berada di tengah-tengah kedua peer. *Database* yang

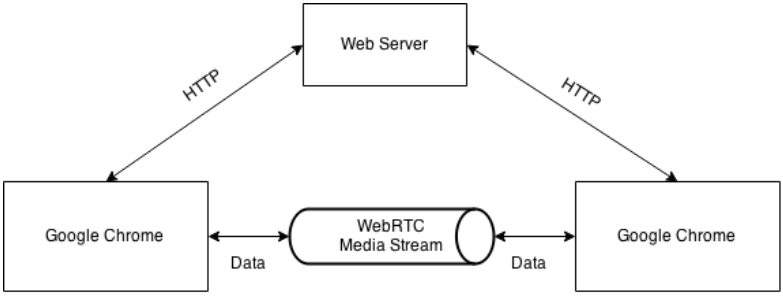
digunakan untuk menyimpan identitas dari setiap pengguna. Identitas tersebut digunakan untuk tanda pengenal pengguna. Setiap pengguna memiliki identitas yang berbeda dan tidak sama agar server tidak bingung pada saat menghubungkan peer.

3.3. Perancangan Diagram Alir Data Aplikasi

Pada bagian ini akan dibahas mengenai gambaran aliran data dan fungsionalitas sistem secara umum. Hal ini direpresentasikan berupa diagram konteks dan diagram alir data level 0.

3.3.1. Perancangan Diagram Konteks Aplikasi

Diagram konteks merupakan diagram alir yang menggambarkan sistem secara umum. Semua aktor eksternal serta aliran data masuk dan keluar sistem digambarkan dalam satu diagram, dimana keseluruhan sistem digambarkan dalam satu proses. Konteks diagram aplikasi ini ditunjukkan pada Gambar 3.3.



Gambar 3.3. Diagram Konteks Aplikasi

Seperti yang ditunjukkan pada Gambar 3.3, sistem ini akan menerima HTTP *request* dari peramban untuk membuka aplikasi. Web server mengirim konten yang diminta oleh peramban. Peramban akan meminta inisialisasi koneksi peer kepada server. Setelah inisialisasi dilakukan, koneksi peer telah dibuat. Ketika

peer melakukan komunikasi dengan peer lain, data dikirim melalui WebRTC media stream.

3.3.2. Diagram Alir Data Level 0 Aplikasi

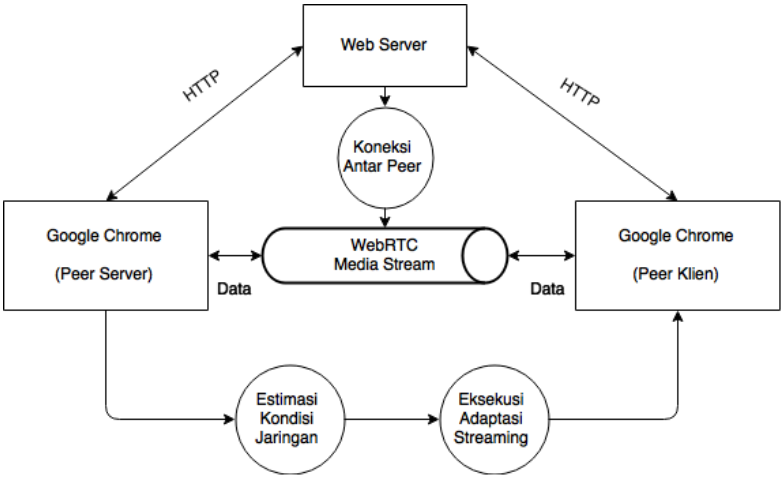
Diagram alir data level 0 ini merupakan dekomposisi dari proses utama pada diagram konteks. Diagram ini menggambarkan fungsionalitas yang terjadi pada proses di sistem ini. Diagram alir data level 0 ditunjukkan pada Gambar 3.4.

Diagram pada Gambar 3.4 menunjukkan proses-proses yang terjadi pada sistem. Proses awal yang terjadi yaitu peramban meminta HTTP *request* kepada web server untuk membuka aplikasi. Web server mengirimkan konten yang diminta oleh peramban. Kemudian peramban meminta inisialisasi koneksi peer kepada server. Server melakukan inisialisasi koneksi kepada pengguna.

Koneksi antar peer bisa dilakukan apabila peer mengetahui identitas dari peer lain. Koneksi yang terjadi antar peer yaitu masing-masing peer dapat mengirim dan menerima data melalui WebRTC Media Stream. Di sinilah jaringan peer-to-peer terjadi.

Setelah masing-masing peer dapat mengirim dan menerima data, dilakukan estimasi kondisi jaringan. Estimasi kondisi jaringan dilakukan di salah satu peer yang disebut dengan peer server. Peer server bertugas untuk melakukan estimasi kondisi jaringan yang ada pada peer klien. Estimasi dilakukan setiap kali peer klien mengirim data ke peer server.

Estimasi yang telah dilakukan menghasilkan keluaran berupa konstrain dari data yang akan dikirim selanjutnya. Konstrain tersebut diatur di dalam peer klien. Jadi pengiriman data selanjutnya dari peer klien ke peer server sesuai dengan kondisi jaringan yang ada.



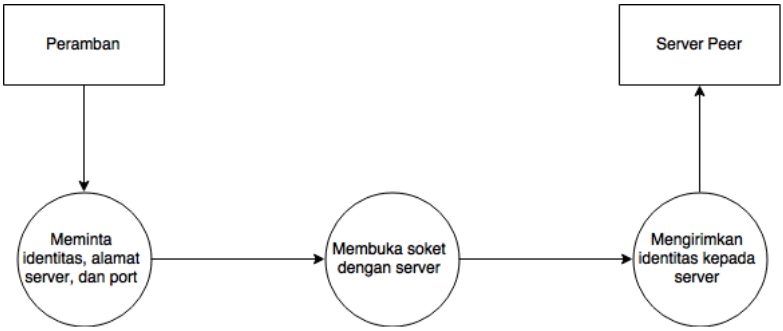
Gambar 3.4. Diagram Alir Data Level 0

3.4. Diagram Alir Aplikasi

Pada bagian ini akan dijelaskan secara lebih mendetail setiap proses yang terjadi pada sistem. Proses ini akan digambarkan menggunakan diagram alir. Hal ini dimaksudkan untuk mempermudah memahami alur kejar aplikasi pada tugas akhir ini. Diagram yang dibahas pada bagian ini adalah proses yang berjalan pada aplikasi diluar yang dikerjakan oleh pustaka pemrograman dan pustaka.

3.4.1. Diagram Alir Proses Inisialisasi Koneksi Peer

Pada proses ini dilakukan inisialisasi koneksi peer yang bertujuan untuk menghubungkan antara peramban dengan server peer. Pada proses inisialisasi tersebut dibutuhkan identitas. Peer satu dengan yang lain tidak boleh memiliki identitas yang sama karena identitas tersebut digunakan untuk keluar masuknya data pada koneksi peer.



Gambar 3.5. Diagram Alir Koneksi Antar Peer

Pada Gambar 3.5 terlihat bahwa peramban meminta identitas, alamat server modul peer, dan port yang digunakan untuk inisialisasi koneksi peer. Kemudian peramban membuat soket dengan server. Soket dimulai dengan mengirimkan identitas yang dimiliki oleh peer. Server akan menyimpan koneksi tersebut yang berfungsi agar dapat berhubungan dengan peer melalui identitas tersebut.

3.4.2. Diagram Alir Proses *Capture* Media Stream

Pada proses ini dilakukan *capture* media stream yang bertujuan untuk mendapatkan gambar atau video dan audio dari perangkat media yang terhubung dengan komputer pengguna. Proses ini memerlukan ijin dari pengguna untuk mengakses perangkat media melalui peramban. Proses *capture* media stream dapat dilihat pada Gambar 3.6.

Dari Gambar 3.6 terlihat bahwa peramban menginisialisasi konstrain dari media seperti resolusi, *frame rate*, dll. Konstrain tersebut berfungsi sebagai aturan pengiriman data kepada peer lain. Kemudian peramban mengirimkan permintaan untuk mengakses perangkat media pada komputer pengguna. Peramban akan menampilkan hasil *capture* dari media stream yang didapat pada halaman web.

Gambar 3.6. Diagram Alir *Capture* Media Stream

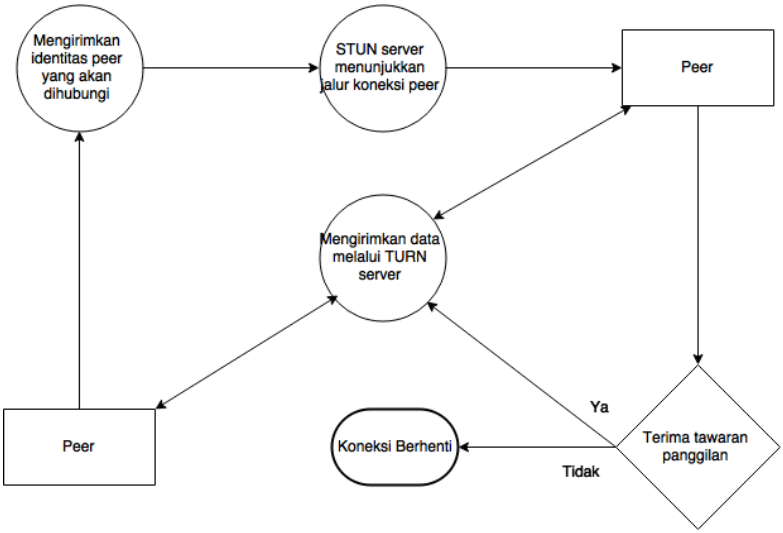
3.4.3. Diagram Alir Proses Koneksi Antar Peer

Pada proses ini dilakukan koneksi antar peer yang digunakan agar antar peer dapat saling terhubung. Pada proses ini dibutuhkan suatu identitas agar peer satu dengan yang lainnya dapat terhubung. Proses koneksi antar peer dapat dilihat pada Gambar 3.7.

Dari Gambar 3.7 terlihat bahwa peer memanggil peer lain dengan mengirimkan identitas dari peer yang akan dihubungi. STUN server berfungsi sebagai penghubung keduanya. STUN server menyimpan koneksi dari masing-masing peer. Peer yang dihubungi akan menerima tawaran panggilan. Jika diterima, maka keduanya akan terhubung. Pada saat berhubungan, data dikirimkan melalui TURN server. Data yang dikirim berupa video stream dari peer lain yang akan ditampilkan dalam halaman web.

3.4.4. Diagram Alir Proses Estimasi Kondisi Jaringan

Pada proses ini dilakukan estimasi terhadap kondisi jaringan saat ini. Kondisi jaringan tersebut berupa *jitter packet-loss*, dan *bandwidth*. Pada saat estimasi kondisi jaringan, sistem akan mencatat kondisi jaringan dari peer klien secara *real time*. Diagram alir proses ini dapat dilihat pada Gambar 3.8.



Gambar 3.7. Diagram Alir Koneksi Antar Peer

Gambar 3.8. Diagram Alir Estimasi Kondisi Jaringan

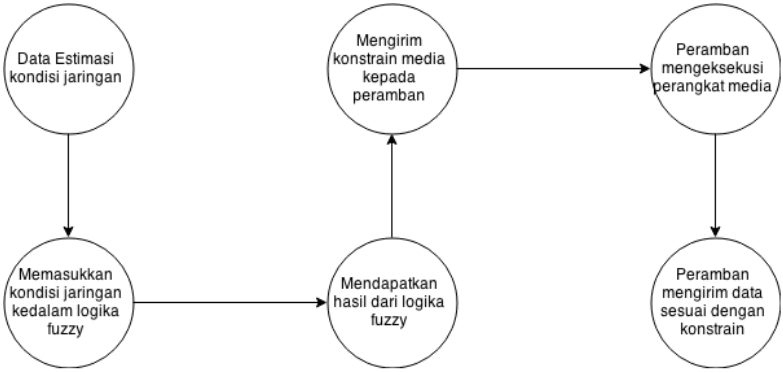
Dari Gambar 3.8 terlihat proses yang dilakukan dalam estimasi kondisi jaringan. Peer klien melakukan pengiriman data berupa video dan audio. Setelah itu dilakukan estimasi kondisi

jaringan untuk peer klien. Peer server menerima data dan juga estimasi kondisi jaringan dari peer klien secara simultan.

3.4.5. Diagram Alir Proses Eksekusi Adaptasi Streaming

Pada proses ini dilakukan proses eksekusi adaptasi streaming. Adaptasi streaming tersebut berupa konstrain dari media yang akan dikirimkan oleh peer server kepada peer host. Konstrain tersebut berupa resolusi (lebar dan tinggi). Diagram alir proses ini dapat dilihat pada Gambar 3.9.

Dari Gambar 3.9 terlihat proses yang dilakukan dalam eksekusi adaptasi streaming. Awalnya, data kondisi jaringan didapat dari estimasi kondisi jaringan peer host. Estimasi kondisi jaringan tersebut dijadikan sebagai argumen dari logika fuzzy. Setelah logika fuzzy mendapatkan hasil, hasil tersebut dikirim kepada peramban. Hasil tersebut dijadikan sebagai konstrain media stream. Setelah konstrain media sudah dieksekusi oleh peramban, maka data yang dikirim kepada peer host sesuai dengan konstrain yang telah ada.

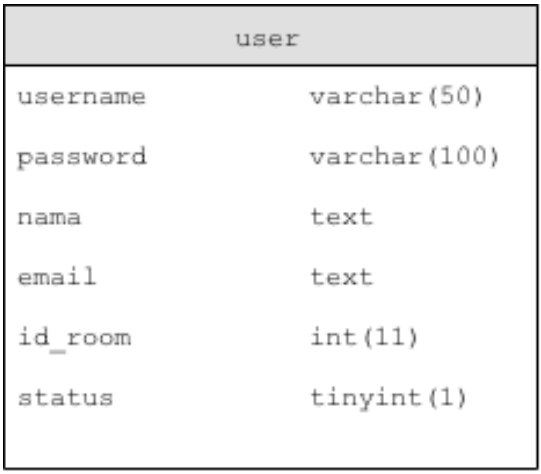


Gambar 3.9. Diagram Alir Eksekusi Adaptasi Streaming

3.5. Perancangan *Database*

Perancangan *database* diperlukan untuk menyimpan informasi pengguna dan *room* aplikasi. Pada Tugas Akhir ini dibuat dua table yaitu tabel `user` untuk menyimpan identitas dan informasi dari pengguna dan tabel `room` untuk menyimpan informasi dari *room*.

3.5.1 Tabel *User*



Gambar 3.10 Tabel user

Pada Gambar 3.10 adalah tabel `user` yang digunakan untuk menyimpan identitas dan informasi dari setiap pengguna aplikasi. Informasi yang disimpan adalah `username` untuk menyimpan nama pengguna untuk masuk ke dalam aplikasi serta sebagai identitas saat inisialisasi koneksi peer, `password` untuk menyimpan kata sandi pada saat masuk ke dalam aplikasi, `nama` untuk menyimpan nama asli pengguna, `email` untuk menyimpan alamat email pengguna, `id_room` untuk

menyimpan pengguna sedang bergabung dengan *room* atau tidak, `status` untuk menyimpan status dari pengguna.

3.5.2 Tabel *Room*

room	
id_room	int(11) auto increment
nama	varchar(100)

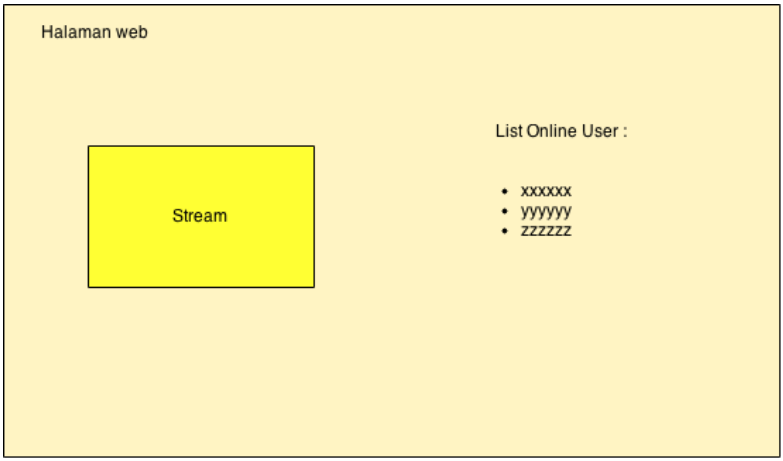
Gambar 3.11. Tabel room

Pada Gambar 3.11 adalah tabel `room` yang digunakan untuk menyimpan informasi dari *room*. Informasi yang disimpan adalah `id_room` untuk menyimpan id dari *room* yang nantinya akan menjadi *reference* dari tabel *user*, `nama` untuk menyimpan nama dari *room*.

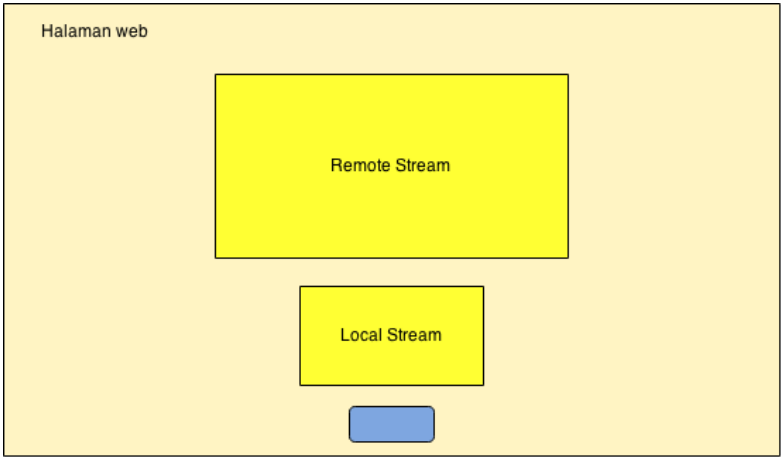
3.6. Rancangan Antarmuka Sistem

Pada Tugas Akhir ini, antarmuka sistem berupa tampilan web. Untuk itu diperlukan perancangan antarmuka yang untuk web ini. Rancangan desain web dapat dilihat pada Gambar 3.12 dan Gambar 3.13.

Pada Gambar 3.12 menunjukkan halaman web menampilkan stream dari perangkat media pengguna tersebut. Resolusi yang diambil dari perangkat media default 640x380. Resolusi tersebut belum mendapatkan konstrain karena belum mendapatkan data estimasi kondisi jaringan dari peer host. Disamping itu terdapat tampilan pengguna yang sedang aktif. Tampilan ini berfungsi untuk mengetahui siapa saja pengguna yang sedang aktif dan pengguna mana yang akan dihubungi. Jadi tidak perlu mengetikkan identitas untuk menghubungi pengguna lain.



Gambar 3.12. Desain Antarmuka Tampilan Web



Gambar 3.13. Desain Antarmuka Tampilan *on Call*

Pada Gambar 3.13 menunjukkan halaman web menampilkan stream dari kedua pengguna. Di sisi atas merupakan tampilan dari *remote* stream, sedangkan sisi bawah merupakan tampilan dari

stream lokal. Pada tampilan ini, stream lokal akan merubah resolusi dari perangkat media sesuai dengan kondisi jaringan yang ada pada peer host. Tetapi pada halaman web peer server, stream lokal akan beresolusi tetap. Akan terlihat pada halaman web peer klien di *remote* stream

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

Setelah melewati proses analisis dan perancangan perangkat lunak, maka dilakukan implementasi sistem. Bab ini akan membahas implementasi dari perancangan sistem perangkat lunak yang telah dibahas pada bab sebelumnya.

4.1. Lingkungan Pembangunan Perangkat Lunak

Pembangunan perangkat lunak pada tugas akhir ini dibangun pada lingkungan yang akan dijabarkan pada bagian selanjutnya.

4.1.1. Lingkungan Perangkat Lunak

Pembangunan aplikasi tugas akhir ini menggunakan bantuan perangkat lunak sebagai berikut:

- Sistem Operasi Ubuntu Server versi 14.04.1 LTS 64 bit sebagai server.
- Sublime Text versi 2.0.2 sebagai IDE untuk pembangunan aplikasi web.
- Node.js sebagai penyedia layanan web server.
- Peer.js sebagai modul pendukung jalannya koneksi peer-to-peer pada web.
- Media Stream Recorder sebagai modul untuk merekam media streaming.
- Fuzzy-class.js sebagai kelas dari logika fuzzy untuk web klien.
- MySQL versi 5.5 sebagai basis data aplikasi.
- CodeIgniter versi 2.1.4 sebagai framework pembuatan web berbasis php.
- MSU Video Quality Measurement Tool versi 4.3 BETA 64-bit

4.1.2. Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk pembangunan aplikasi tugas akhir ini adalah menggunakan *Virtual Machine* (VM) sebagai server dari sistem. VM mempunyai spesifikasi 1 core Common KVM processor 3.0 Ghz dan 512MB RAM.

4.2. Mekanisme Penentuan Resolusi Video

Pada proses penentuan resolusi video streaming bertujuan untuk tetap menjaga kualitas video pada saat kondisi jaringan dinamis. Metode yang digunakan untuk menentukan resolusi video yaitu logika fuzzy.

4.2.1. Penentuan Variabel Fuzzy

Berdasarkan analisis terhadap data yang telah dikumpulkan, ada tiga variabel fuzzy yang digunakan pada mekanisme adaptif streaming antara lain *jitter*, *packet-loss*, dan *available bandwidth*. Pemilihan variabel tersebut berdasarkan pada kualitas video yang semakin menurun ketika variabel tersebut juga naik.

4.2.2. Fuzzifikasi dan Fungsi Keanggotaan Fuzzy

Penentuan nilai faktor pada subbab ini digunakan untuk menentukan nilai faktor yang ideal untuk media streaming. Nilai yang telah didapat digunakan untuk menjadi variabel dari logika fuzzy. Dari subbab yang telah dijelaskan sebelumnya, didapatkan faktor yang dapat mempengaruhi kualitas video yaitu *jitter*, *packet-loss*, dan *available bandwidth*.

Setelah menentukan variabel fuzzy, selanjutnya adalah menentukan semesta pembicaraan dan domain dari masing-masing fungsi keanggotaan yang ada. Semesta dari fungsi keanggotaan *jitter* adalah 0 sampai dengan 200. Semesta dari fungsi keanggotaan *packet-loss* adalah 0 sampai dengan 100. Sedangkan

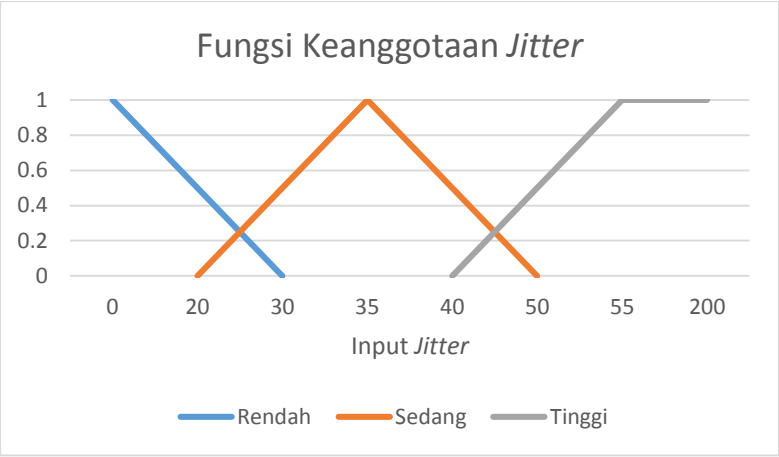
semesta untuk *available bandwidth* yaitu 0 sampai dengan 10. Kemudian untuk nilai resolusi sebagai keluaran dari logika fuzzy memiliki semesta dari 180 sampai dengan 720.

Dalam fungsi keanggotaan fuzzy, digunakan kurva segitiga. Kurva segitiga memiliki 3 titik ditunjukkan oleh *a*,*b*, dan *c*. Titik *a* menunjukkan nilai domain terkecil yang mempunyai derajat keanggotaan nol. Titik *b* menunjukkan nilai domain yang mempunyai derajat keanggotaan satu. Titik *d* menunjukkan nilai domain terbesar yang mempunyai derajat keanggotaan nol. Tabel 4.1 sampai dengan Tabel 4.3 menunjukkan fungsi keanggotaan yang digunakan pada logika fuzzy. Tabel 4.4 merupakan fungsi keanggotaan keluaran dari logika fuzzy berupa nilai resolusi video.

Tabel 4.1. Fungsi Keanggotaan *Jitter*

Titik *a*	**Titik *b***	**Titik *c***	**Variabel Linguistik**
40	55	200	Tinggi
20	35	50	Sedang
0	0	30	Rendah

Dari Tabel 4.1 dilihat bahwa kategori *jitter* dibagi menjadi 3 kategori yaitu tinggi, sedang, dan rendah. Kategori *jitter* tinggi yaitu titik *a* dengan nilai domain 40, titik *b* dengan nilai domain 55, dan titik *c* dengan nilai domain 200. Untuk kategori sedang, yaitu titik *a* dengan nilai domain 20, titik *b* dengan nilai domain 35, dan titik *c* dengan nilai domain 50. Sedangkan untuk kategori rendah yaitu titik *a* dan titik *b* dengan nilai domain 0 dan untuk titik *c* dengan nilai domain 30.

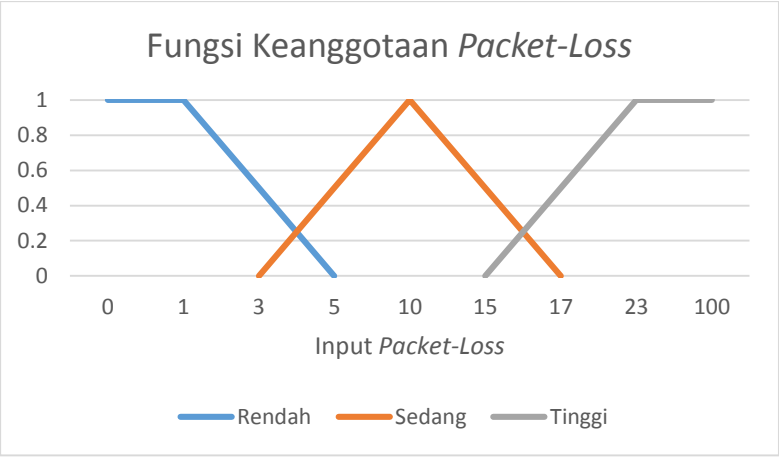


Gambar 4.1. Kurva Fungsi Keanggotaan *Jitter*

Dari Tabel 4.2 dilihat bahwa kategori *packet-loss* dibagi menjadi 3 kategori yaitu tinggi, sedang, dan rendah. Kategori *packet-loss* tinggi yaitu titik *a* dengan nilai domain 15, titik *b* dengan nilai domain 23, dan titik *c* dengan nilai domain 100. Untuk kategori sedang, yaitu titik *a* dengan nilai domain 3, titik *b* dengan nilai domain 10, dan titik *c* dengan nilai domain 17. Sedangkan untuk kategori rendah yaitu titik *a* dengan nilai domain 0, titik *b* dengan nilai domain 1, dan titik *c* dengan nilai domain 5.

Tabel 4.2. Fungsi Keanggotaan *Packet-loss*

Titik *a*	**Titik *b***	**Titik *c***	**Variabel Linguistik**
15	23	100	Tinggi
3	10	17	Sedang
0	1	5	Rendah



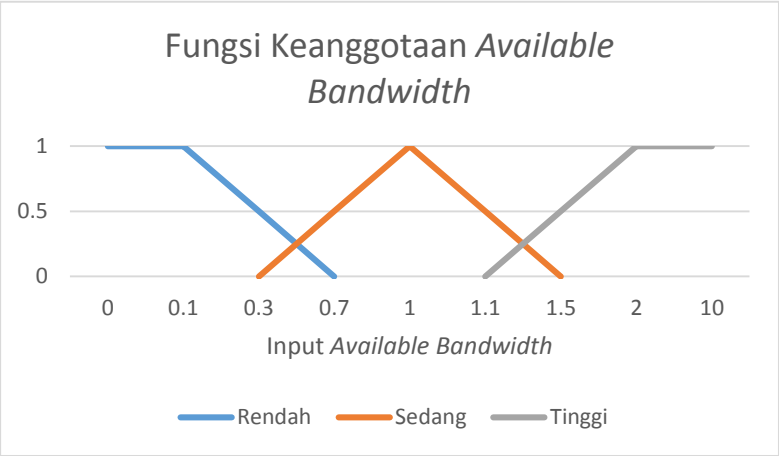
Gambar 4.2. Kurva Fungsi Keanggotaan *Packet-Loss*

Tabel 4.3. Fungsi Keanggotaan *Available Bandwidth*

Titik *a*	**Titik *b***	**Titik *c***	**Variabel Linguistik**
1.1	2	10	Tinggi
0.3	1	1.5	Sedang
0	0.1	0.7	Rendah

Dari Tabel 4.3 dilihat bahwa kategori *available bandwidth* dibagi menjadi 3 kategori yaitu tinggi, sedang, dan rendah. Kategori *available bandwidth* tinggi yaitu titik *a* dengan nilai domain 1.1, titik *b* dengan nilai domain 2, dan titik *c* dengan nilai domain 10. Untuk kategori sedang, yaitu titik *a* dengan nilai domain 0.3, titik *b* dengan nilai domain 1, dan titik *c* dengan nilai domain 1.5. Sedangkan untuk kategori rendah yaitu titik *a* dengan

nilai domain 0, titik *b* dengan nilai domain 0.1, dan titik *c* dengan nilai domain 0.7.



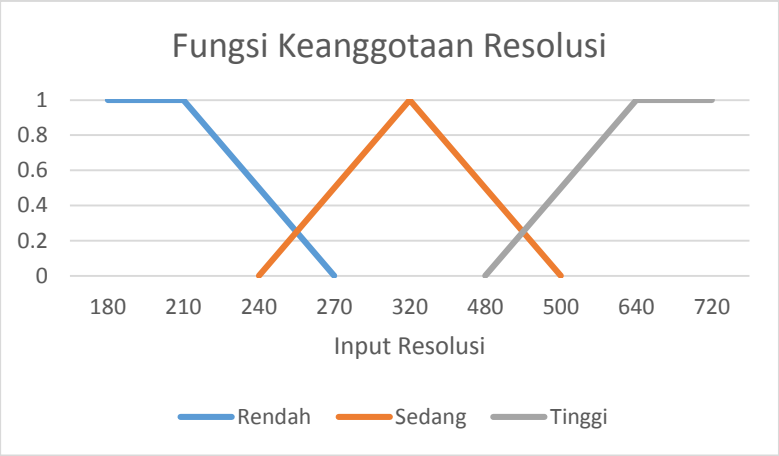
Gambar 4.3. Kurva Fungsi Keanggotaan *Available Bandwidth*

Tabel 4.4. Fungsi Keanggotaan Resolusi

Titik *a*	**Titik *b***	**Titik *c***	**Variabel Linguistik**
180	210	270	Rendah
240	320	500	Sedang
480	640	720	Tinggi

Dari Tabel 4.4 dilihat bahwa kategori resolusi dibagi menjadi 3 kategori yaitu rendah, sedang, dan tinggi. Kategori resolusi rendah yaitu titik *a* dengan nilai domain 180, titik *b* dengan nilai domain 210, dan titik *c* dengan nilai domain 270. Untuk kategori sedang, yaitu titik *a* dengan nilai domain 240, titik *b*

dengan nilai domain 320, dan titik *c* dengan nilai domain 500. Sedangkan untuk kategori tinggi yaitu titik *a* dengan nilai domain 480, titik *b* dengan nilai domain 640, dan titik *c* dengan nilai domain 720.



Gambar 4.4. Kurva Fungsi Keanggotaan Resolusi

4.2.3. Aturan Kontrol Fuzzy

Dari ketiga variabel keanggotaan fuzzy yang dipilih, dibuat aturan sesuai dengan data yang telah didapatkan. Dari ketiga variabel keanggotaan fuzzy yang dipilih dibuat menjadi beberapa kombinasi aturan untuk pemilihan resolusi. Awal mula pembuatan aturan fuzzy yaitu dengan membentuk matrik tiga dimensi dari ketiga variabel yaitu *jitter*, *packet-loss*, dan *available bandwidth*. Matrik tiga dimensi yang telah dibuat kemudian diisi dengan hasil resolusi yang didapat sesuai dengan kondisi dari ketiga variabel. Resolusi yang dipilih adalah nilai yang terbaik pada kondisi jaringan tertentu berdasarkan analisa data. Matrik tersebut dapat dilihat di halaman Lampiran. Dalam pembuatan aturan ini, terdapat beberapa acuan yang akan dijelaskan sebagai berikut.

1. Bandwidth merupakan lebar pita yang memungkinkan berapa banyak data yang dapat dikirim dalam jaringan. Apabila bandwidth semakin kecil, maka data yang dikirim akan semakin sedikit. Begitu pula sebaliknya, apabila bandwidth semakin besar, maka data yang dikirim akan semakin banyak.
2. Packet-Loss merupakan kegagalan dalam pengiriman data kepada tujuan. Karena video streaming yang dipakai dalam Tugas akhir ini menggunakan protokol UDP, data yang sudah lalu tidak dikirimkan kembali karena UDP bukan *content based*. Packet-Loss menyebabkan video streaming yang diterima tidak lengkap karena adanya data yang tidak terkirim pada jaringan.
3. Jitter merupakan variasi dari delay. Jika perbedaan delay yang jauh antara data yang dikirimkan sebelumnya dan data yang dikirimkan selanjutnya akan menyebabkan terjadinya *lagging* pada video.
4. Resolusi adalah besarnya ukuran dari video. Resolusi yang semakin besar, maka diperlukan pengiriman data yang besar juga. Begitu pula sebaliknya, apabila resolusi semakin kecil, maka data yang dikirimkan juga kecil.

Dari keempat acuan tersebut dibuat suatu indeks prioritas yang mempengaruhi kualitas dari video streaming. Tabel 4.5. merupakan indeks prioritas yang dapat mempengaruhi kualitas dari video streaming.

Tabel 4.5. Indeks Prioritas Variabel

Variabel	Indeks Prioritas
Available Bandwidth	1
Packet-Loss	2
Jitter	3

Setelah itu aturan yang sudah ada dieliminasi. Eliminasi aturan dilakukan apabila aturan-aturan tersebut memiliki hasil yang sama pada kondisi jaringan tertentu dengan menggabungkan menjadi satu aturan. Tabel 4.6 merupakan contoh eliminasi aturan. Bagaimanapun kondisi *packet-loss* dan *jitter*, apabila *available bandwidth*nya rendah, maka resolusi yang dipilih adalah rendah. Tabel 4.7 adalah aturan akhir yang didapat dari hasil proses eliminasi beberapa aturan.

Tabel 4.6. Aturan yang Digabungkan

Available Bandwidth	**Packet-Loss**	**Jitter**	**Resolusi**
Rendah	Rendah	Rendah	Rendah
Rendah	Rendah	Sedang	Rendah
Rendah	Rendah	Tinggi	Rendah
Rendah	Sedang	Rendah	Rendah
Rendah	Sedang	Sedang	Rendah
Rendah	Sedang	Tinggi	Rendah
Rendah	Tinggi	Rendah	Rendah
Rendah	Tinggi	Sedang	Rendah
Rendah	Tinggi	Tinggi	Rendah
Hasil yang didapat			
Rendah	**-**	**-**	**Rendah**

Tabel 4.7. Aturan Kontrol Fuzzy

Available Bandwidth	Packet-Loss	Jitter	Resolusi
Rendah	-	-	Rendah
Sedang	Tinggi	-	Rendah
Sedang	Sedang	Tinggi	Rendah
Sedang	Sedang	Sedang	Sedang
Sedang	Sedang	Rendah	Sedang
Sedang	Rendah	-	Sedang
Tinggi	Rendah	-	Tinggi
Tinggi	Sedang	-	Sedang
Tinggi	Tinggi	-	Rendah

4.2.4. Defuzzyfikasi

Keluaran yang dihasilkan dari proses inferensi adalah suatu bilangan pada domain himpunan fuzzy tersebut. Sehingga jika diberikan suatu himpunan fuzzy dalam rentang tertentu, maka harus diambil suatu nilai pasti sebagai keluaran. Pada Tugas akhir ini resolusi adalah keluaran dari proses inferensi logika fuzzy. Pada proses defuzzyfikasi, digunakan metode centroid atau pengambilan titik pusat daerah fuzzy. Jadi, nilai keluaran resolusi yang diambil pada proses defuzzyfikasi yaitu 210, 320, dan 640.

4.3. Implementasi Perangkat Lunak

Implementasi perangkat lunak adalah implementasi dalam pembuatan perangkat lunak. Setiap bagian dari implementasi akan dijelaskan selanjutnya.

4.3.1. Implementasi Inisialisasi Koneksi Peer

Gambar 4.5 menunjukkan *pseudocode* implementasi dari inisialisasi koneksi peer. Pada saat klien pertama kali membuka halaman, pengguna dihadapkan pada halaman login. Pengguna memasukkan username dan password agar bisa masuk pada halaman selanjutnya. Fungsi dari username adalah sebagai identitas inisialisasi koneksi peer. Dalam proses inisialisasi membutuhkan modul Peer yang terdapat dalam Node.js serta port yang digunakan.

Prosedur 1. Inisialisasi Koneksi Peer	
Input : -	
1.	Load peer.js klien
2.	SEND id, host modul peer, port
3.	IF id sudah digunakan
4.	Tidak bisa inisialisasi
5.	ELSE
6.	Koneksi peer terjadi
Output : -	

Gambar 4.5. *Pseudocode* Inisialisasi Koneksi Peer

Penjelasan dari *pseudocode* inisialisasi koneksi peer adalah sebagai berikut.

1. Melakukan load peer.js klien yang ada pada aplikasi.
2. Mengirimkan id, host modul peer, dan juga port yang digunakan untuk inisialisasi koneksi dengan modul peer yang ada pada server.

3. Jika id yang dikirim kepada modul peer sudah digunakan, halaman web akan menampilkan pop-up bahwa id sudah digunakan
4. Jika id belum digunakan, maka koneksi peer akan terjadi.

4.3.2. Implementasi Mendapatkan Media Stream

Gambar 4.6 menunjukkan *pseudocode* implementasi untuk mendapatkan media stream pengguna. Proses ini dilakukan setelah inisialisasi koneksi peer. Media stream yang didapat kemudian diletakkan pada halaman web. Tujuannya untuk menampilkan media stream yang ditangkap oleh kamera.

Penjelasan *pseudocode* untuk mendapatkan media stream adalah sebagai berikut.

1. Memanggil fungsi `getUserMadia()` yang telah disediakan oleh peer.js klien.
2. Men-set konstrain media seperti resolusi dan *frame rate*.
3. Menampilkan stream pada halaman web.

Prosedur 2. Mendapatkan media stream	
Input : -	
1.	`getUserMedia()`
2.	`  SET konstrain`
3.	`  IF sukses`
4.	`  Tampilkan dalam web`
5.	`  ELSE`
6.	`  Tampilkan pesan error`
Output : -	

Gambar 4.6. *Pseudocode* Mendapatkan Media Stream

4.3.3. Implementasi Panggilan ke Peer Lain

Gambar 4.7 menunjukkan *pseudocode* implementasi peer melakukan panggilan ke peer lain. Proses ini dilakukan untuk menghubungankan antara satu peer dengan peer lain. Saat peer

melakukan panggilan ke peer lain, peer memerlukan identitas peer lain. Fungsi identitas yaitu agar STUN server dapat menemukan peer yang dimaksud dan agar tidak saling terturkar antara peer satu dengan yang lain.

Prosedur 3. Panggilan ke peer lain	
Input : -	
1.	`Call (id, stream)`
2.	`  IF masih melakukan panggilan`
3.	`  Panggilan ditutup`
4.	`  Tampilkan stream dalam web`
Output : -	

Gambar 4.7. *Pseudocode* Panggilan ke Peer Lain

Penjelasan *pseudocode* untuk melakukan panggilan ke peer lain adalah sebagai berikut.

1. Memanggil fungsi `call()` yang mempunyai parameter id serta stream. Id adalah identitas dari peer yang akan dituju. Stream adalah data yang akan dikirimkan kepada peer lain saat berhubungan.
2. Jika peer masih melakukan panggilan, maka panggilan yang sebelumnya akan ditutup.
3. Tampilkan stream dari peer lain dalam halaman web.

4.3.4. Implementasi Estimasi Kondisi Jaringan

Gambar 4.8 menunjukkan *pseudocode* implementasi untuk mendapatkan estimasi kondisi jaringan. Proses ini dilakukan pada saat kedua peer saling mengirim data. Tujuan estimasi kondisi jaringan adalah untuk mengetahui kondisi jaringan yang ada pada kedua peer. Kondisi jaringan yang didapat berupa *jitter*, *packet-loss* dan *available bandwidth*. Kondisi jaringan yang telah didapat nantinya akan digunakan sebagai parameter untuk proses selanjutnya.

Prosedur 4. Estimasi kondisi jaringan	
Input : peer	
1.	results <= hasil perhitungan
2.	PL(0) = 0
3.	FOR 0 to length results
4.	Jitter = googJitterReceived
5.	PL(i) = packetLost – PL(i-1)
6.	x 100%
7.	Bandwidth =
8.	googAvailableReceiveBandwidth
9.	CALLBACK 5s
Output : result	

Gambar 4.8. *Pseudocode* Estimasi Kondisi Jaringan

Penjelasan *pseudocode* untuk mendapatkan estimasi kondisi jaringan adalah sebagai berikut.

1. Estimasi kondisi jaringan memerlukan input berupa peer.
2. Result yang nantinya dipakai berupa array yang mempunyai member audio, video, dan results.
3. Hasil dari results di ulang sampai panjang dari results tersebut.
4. Hasil Jitter dan RTT didapat dari codec VP8.
5. Hasil bandwidth didapat dari tipe VideoBWE.
6. Fungsi ini dipanggil terus menerus dengan jeda 10 sekon.

4.3.5. Implementasi Mendapatkan Konstrain Adaptasi

Gambar 4.9 menunjukkan *pseudocode* implementasi untuk mendapatkan konstrain adaptasi. Input dari implementasi ini didapatkan dari proses sebelumnya yaitu estimasi kondisi jaringan. Konstrain didapatkan dari logika fuzzy dengan member dan hasil yang sudah ditentukan.

Prosedur 4. Estimasi kondisi jaringan	
Input : jitter, RTT, availableBandwidth	
1.	Load class fuzzy
2.	Inisialisasi fuzzy
3.	ADD member jitter low
4.	ADD member jitter medium
5.	ADD member jitter high
6.	ADD member packetLost low
7.	ADD member packetLost medium
8.	ADD member packetLost high
9.	ADD member bandwidth low
10.	ADD member bandwidth medium
11.	ADD member bandwidth high
12.	ADD result resolusi low
13.	ADD result resolusi medium
14.	ADD result resolusi high
15.	ADD rules IF bandwidth low THEN
16.	resolusi low
17.	ADD rules IF bandwidth medium AND
18.	packetLost high
19.	THEN resolusi low
20.	ADD rules IF bandwidth medium AND
21.	packetLost medium AND jitter
22.	high THEN resolusi low
23.	ADD rules IF bandwidth medium AND
24.	packetLost medium AND jitter
25.	medium THEN resolusi medium
26.	ADD rules IF bandwidth medium AND
27.	packetLost medium AND jitter
28.	low THEN resolusi medium
29.	ADD rules IF bandwidth medium AND
30.	packetLost low
31.	resolusi medium
32.	ADD rules IF bandwidth high AND
33.	packetLost low THEN
34.	resolusi high

```
35.      ADD rules IF bandwidth high AND
36.          packetLost medium THEN
37.          resolusi medium
38.      ADD rules IF bandwidth high AND
39.          packetLost high THEN
40.          resolusi low
41.      Hasil = CalculateFuzzy
Output = Hasil
```

Gambar 4.9. *Pseudocode* Mendapatkan Konstrain Adaptasi

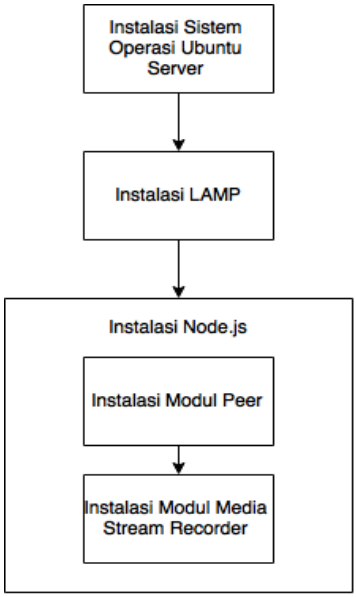
Penjelasan *pseudocode* untuk mendapatkan konstrain adaptasi adalah sebagai berikut.

1. Melakukan load kelas logika fuzzy.
2. Menambahkan member *bandwidth*, *packet-lost*, dan *jitter* dengan kondisi *low*, *medium*, dan *high*.
3. Menambahkan hasil berupa resolusi dengan kondisi low, medium, dan high.
4. Menambahkan rules jika kondisi *bandwidth* low maka resolusi yang diambil adalah low.
5. Menambahkan rules jika kondisi *bandwidth* medium dan *packet-lost* high maka resolusi yang diambil adalah low.
6. Menambahkan rules jika kondisi *bandwidth* medium dan *packet-lost* medium dan *jitter* high maka resolusi yang diambil adalah low.
7. Menambahkan rules jika kondisi *bandwidth* medium dan *packet-lost* medium dan *jitter* medium maka resolusi yang diambil adalah medium.
8. Menambahkan rules jika kondisi *bandwidth* medium dan *packet-lost* medium dan *jitter* low maka resolusi yang diambil adalah medium.
9. Menambahkan rules jika kondisi *bandwidth* medium dan *packet-lost* low maka resolusi yang diambil adalah medium.
10. Menambahkan rules jika kondisi *bandwidth* high dan *packet-lost* low maka resolusi yang diambil adalah high.

11. Menambahkan rules jika kondisi *bandwidth* high dan *packet-lost* medium maka resolusi yang diambil adalah medium.
12. Menambahkan rules jika kondisi *bandwidth* high dan *packet-lost* high maka resolusi yang diambil adalah low.
13. Menghitung hasil kalkulasi logika fuzzy.

4.4. Implementasi Server

Pada subbab ini akan dibahas mengenai implementasi server. Implementasi dilakukan dengan pemasangan beberapa perangkat lunak untuk mendukung proses implementasi perangkat lunak.



Gambar 4.10. Tahap Instalasi Server

Pada Gambar 4.10 ditunjukkan tahap-tahap utama untuk implementasi server. Pada setiap tahap terdapat konfigurasi agar perangkat lunak dapat mendukung jalannya peer server.

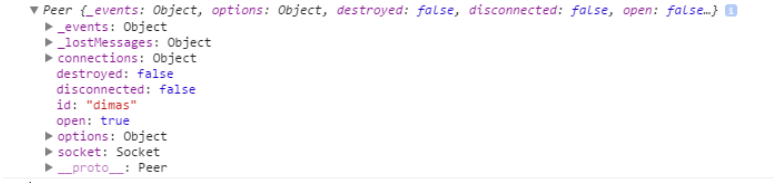
Pada server, sistem operasi menggunakan Ubuntu Server 14.04.1 LTS 64bit. Kemudian dilakukan instalasi LAMP (Linux Apache MySQL PHP) yang berfungsi untuk web server dan *database* server. Setelah itu dilakukan instalasi Node.js yang berfungsi sebagai tempat instalasi modul peer dan modul media stream recorder.

4.5. Implementasi Antarmuka Pengguna

Pada subbab ini menampilkan secara umum antarmuka pengguna. Tampilan antarmuka pengguna aplikasi ini berupa halaman web. Interaksi antara pengguna dan perangkat lunak dilakukan pada halaman web tersebut. Antarmuka pengguna yang diimplementasikan yaitu inisialisasi koneksi peer, *capture* media stream, koneksi antar peer, estimasi kondisi jaringan, dan eksekusi konstrain adaptasi.

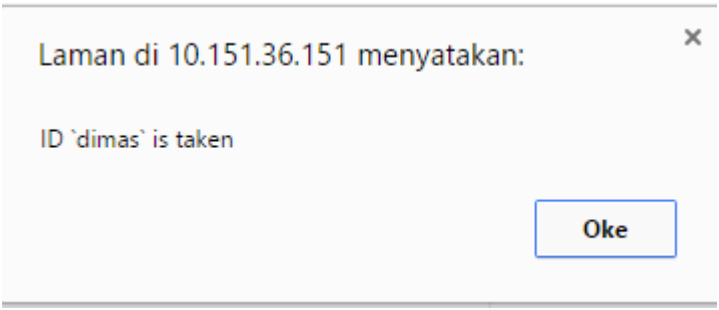
4.5.1. Antarmuka Inisialisasi Koneksi Peer

Pada Gambar 4.11 adalah tampilan inisialisasi koneksi peer yang berhasil. Koneksi peer ditunjukkan dalam bentuk konsol dari peramban. Pada tampilan tersebut konsol akan menampilkan peer yang dibuat oleh peramban jika inisialisasi yang dilakukan berhasil. Di dalam objek Peer terdapat beberapa nilai dan salah satunya adalah identitas.



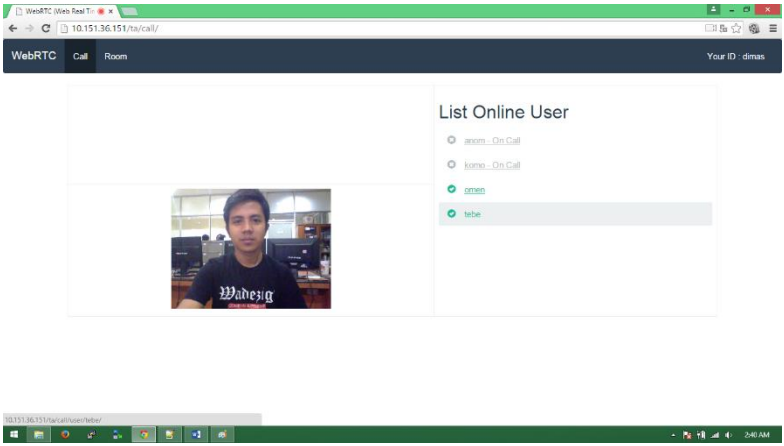
Gambar 4.11. Antarmuka Inisialisasi Koneksi Peer Berhasil

Pada Gambar 4.12 adalah tampilan inisilasi koneksi peer yang gagal. Kegagalan ini ditemukan saat ditemukan identitas dari peer lain yang sama. Antarmuka berupa peringatan pada halaman web.



Gambar 4.12. Antarmuka Inisialisasi Koneksi Peer Gagal

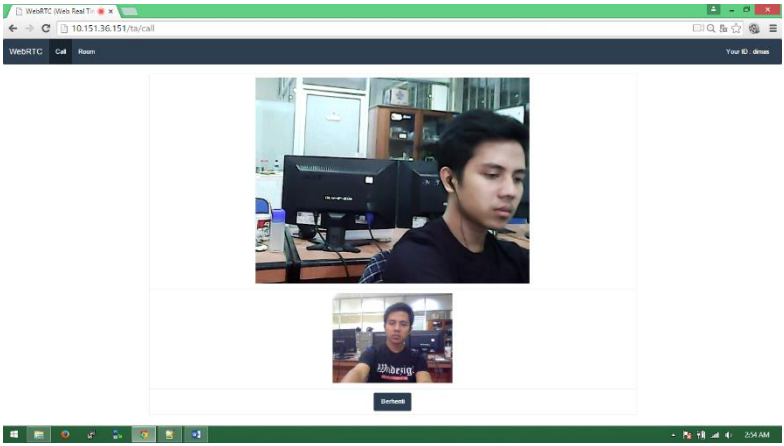
4.5.2. Antarmuka *Capture* Media Stream



Gambar 4.13. Antarmuka *Capture* Media Stream

Pada Gambar 4.13 adalah antarmuka *capture* media stream yang sebelumnya memerlukan ijin untuk mengakses perangkat media pada komputer pengguna. Perangkat media yang telah memiliki ijin untuk mengakses perangkat media akan menampilkan hasil *capture* media pada halaman web. Di samping hasil *capture* dari perangkat media terdapat list pengguna yang sedang aktif. Pengguna yang sedang aktif juga dibagi menjadi dua tipe yaitu pengguna yang tidak sedang berhubungan dengan pengguna lain dan pengguna yang sedang dalam panggilan atau sedang berhubungan dengan pengguna lain.

4.5.3. Antarmuka Koneksi Antar Peer

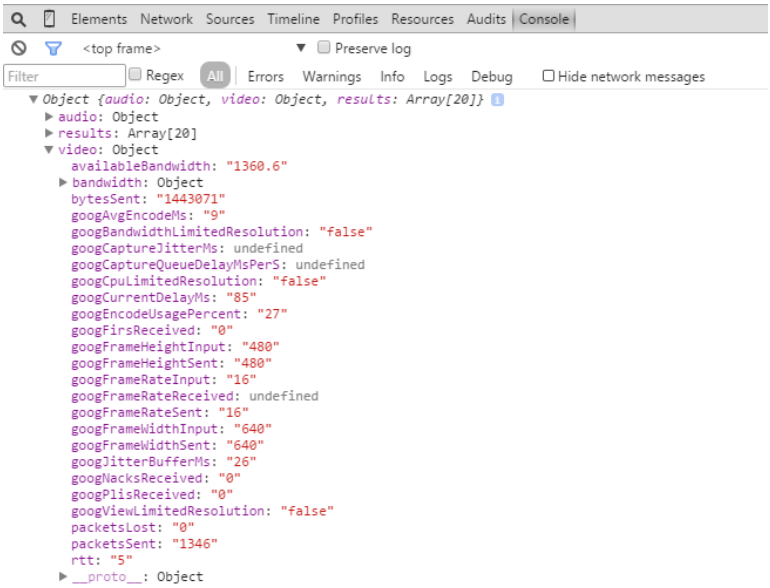


Gambar 4.14. Antarmuka Koneksi Antar Peer

Pada Gambar 4.14 adalah antarmuka Koneksi Antar Peer. Sebelum melakukan koneksi antar peer, identitas peer yang akan dihubungi dikirimkan kepada server. Ketika peer berhubungan, kedua peer akan saling mengirim dan menerima data berupa video dan audio. Media stream dari kedua peer akan ditampilkan pada halaman web.

4.5.4. Antarmuka Estimasi Kondisi Jaringan

Pada Gambar 4.15 adalah antarmuka Estimasi Kondisi Jaringan. Ketika masing-masing peer mengirimkan data, dilakukan estimasi kondisi jaringan setiap 10 detik. Estimasi kondisi jaringan akan ditampilkan pada konsol peramban. Ada banyak nilai yang ditampilkan dalam konsol seperti *jitter*, *packet-loss*, dll.



Gambar 4.15. Antarmuka Estimasi Kondisi Jaringan

4.5.5. Antarmuka Eksekusi Strategi Adaptasi

Antarmuka eksekusi strategi jaringan terlihat apabila peer sedang melakukan hubungan dengan peer lain dan kondisi jaringan berubah. Strategi jaringan yang dieksekusi sesuai dengan kondisi jaringan yang ada. Pada Gambar 4.16 telah ditampilkan antarmuka koneksi antar peer yang terjadi. Eksekusi strategi adaptasi akan berpengaruh pada media stream peer klien. Dimana kondisi

jaringan yang ada dilakukan estimasi oleh peer server dan dihitung menggunakan logika fuzzy, kemudian hasil dari perhitungan tersebut dikirim kepada peer klien. Hasil tersebut dijadikan sebagai konstrain media stream dari peer klien. Konstrain tersebut dapat dilihat pada video stream klien yang berubah sesuai dengan strategi adaptasi yang ditentukan.

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas mengenai hasil ujicoba fungsionalitas dan performa dari aplikasi tugas akhir. Tahapan ujicoba ini akan dilakukan dalam beberapa skenario yang akan dibahas pada bagian selanjutnya dari bab ini.

5.1. Lingkungan Uji Coba

Dalam melakukan uji coba aplikasi tugas akhir ini, dilakukan menggunakan 2 komputer yang masing-masing spesifikasi nya sebagai berikut.

Spesifikasi Perangkat Komputer 1:

- Intel Core i7-4700MQ (2.4 Ghz)
- Windows 8.1 64-bit sebagai Sistem Operasi
- 8GB RAM
- Google Chrome versi 43

Spesifikasi Perangkat Komputer 2:

- Intel Core i3-530 (2.9 Ghz)
- Linux Mint Debian 15
- 2GB RAM
- Google Chrome versi 43

5.2. Skenario Uji Coba

Proses uji coba mengenai aplikasi ini akan dibagi ke dalam beberapa skenario. Uji coba bertujuan dimaksudkan untuk menguji fungsionalitas serta performa dari aplikasi tugas akhir ini. Pengujian fungsionalitas bertujuan untuk memastikan aplikasi berjalan sesuai fungsi yang diharapkan. Uji coba dilakukan dengan

menguji kualitas layanan dari sistem adaptif streaming multimedia peer-to-peer pada WebRTC.

5.2.1. Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian yang dilakukan terhadap jalannya fungsi-fungsi utama pada sistem yang telah dibuat. Pengujian dilakukan ke seluruh fungsi sistem. Uji coba fungsionalitas ini meliputi semua alur program yang sudah dijelaskan pada bab sebelumnya diantaranya sebagai berikut.

1. Inisialisasi Koneksi Peer (UC-01)
   Inisialisasi konek peer merupakan salah satu fungsi yang dilakukan untuk menghubungkan peer dengan server peer.
2. Capture Media Stream (UC-02)
   Setelah inisialisasi koneksi peer, kemudian masing-masing peer akan melakukan capture perangkat media stream yang ada pada komputer pengguna.
3. Koneksi Antar Peer (UC-03)
   Setelah mendapatkan hasil capture dari media stream, uji coba selanjutnya adalah koneksi antar peer. Peer yang telah terhubung dengan server akan dapat berhubungan dengan peer lain.
4. Estimasi Kondisi Jaringan (UC-04)
   Estimasi kondisi jaringan dilakukan ketika kedua peer sedang berhubungan. Estimasi dilakukan pada salah satu peer yang nantinya disebut dengan peer server.
5. Eksekusi Adaptasi Streaming (UC-05)
   Setelah didapat hasil estimasi kondisi jaringan, kemudia peer klien akan mengeksekusi konstrain adaptasi streming yang dikirim oleh peer server.

5.2.1.1. Uji Coba Insialisasi Koneksi Peer

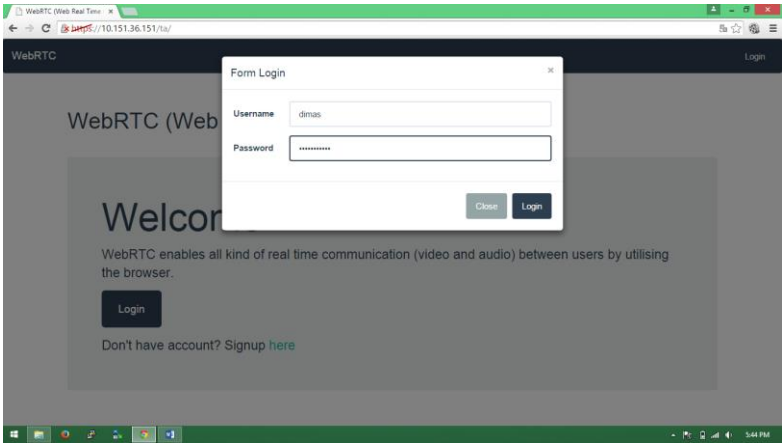
Uji coba dilakukan dengan menjalankan perangkat lunak pada server yaitu modul Peer yang digunakan sebagai server peer dari pengguna sistem. Kemudian uji coba dilanjutkan dengan

permintaan identitas oleh peramban kepada pengguna sehingga server peer dapat mengenalinya serta meminta alamat dari server Peer dan port yang digunakan untuk menghubungi server.

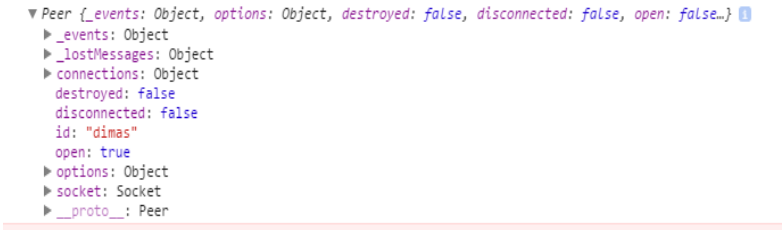
Tabel 5.1. Prosedur Uji Coba Inisialisasi Koneksi Peer

ID	**UC-01**
Nama	**Uji Coba Inisialisasi Koneksi Peer**
Tujuan Uji Coba	Menguji fitur sistem untuk melakukan inisialisasi koneksi peer dari pengguna kepada server.
Kondisi Awal	Server Peer belum dijalankan
Skenario	**Menjalankan modul Peer yang ada pada server dan melakukan login pada halaman web untuk mengambil identitas dari pengguna.**
Masukan	**Identitas, Alamat server Peer, Port**
Keluaran	**-**
Hasil Uji Coba	**Berhasil**

Sesuai yang dijelaskan pada Tabel 5.1, uji coba dilakukan dengan menjalankan modul Peer yang ada pada server. Setelah modul dijalankan, pengguna melakukan login untuk masuk kedalam sistem dan juga sebagai identitas dari peer yang akan dibuat. Gambar 5.1 menunjukkan saat user login menggunakan akun yang sudah didaftarkan. Gambar 5.2 menunjukkan koneksi peer yang telah terjadi. Terdapat id yang merupakan nilai dari identitas peer.



Gambar 5.1. Prosedur Uji Coba Inisialisasi Koneksi



Gambar 5.2. Uji Coba Peer yang Sudah Terhubung dengan Server

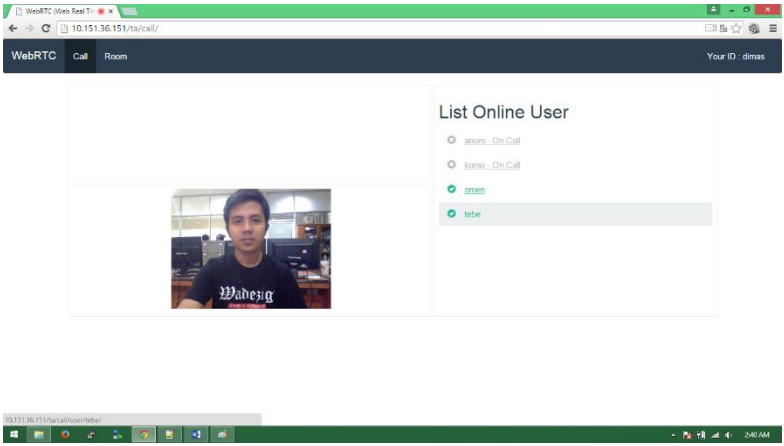
5.2.1.2. Uji Coba *Capture* Media Stream

Uji coba ini dilakukan untuk mendapatkan *capture* media stream. Uji coba dilakukan dengan meminta ijin kepada pengguna agar dapat mengakses perangkat media. Pada saat meminta akses perangkat media, sistem akan mengatur konstrain dari perangkat media untuk ditampilkan dalam halaman web. Keluaran yang didapat adalah hasil capture dari perangkat media stream komputer.

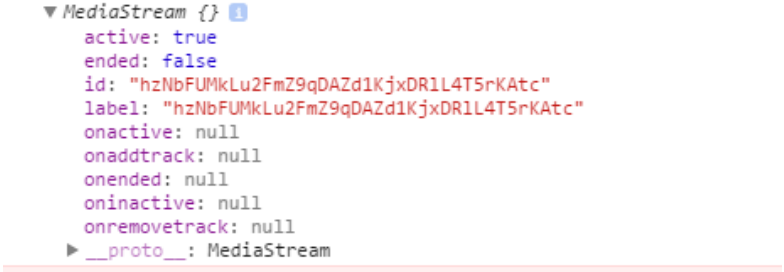
Tabel 5.2. Prosedur Uji Coba *Capture* Media Stream

ID	**UC-02**
Nama	**Uji Coba *Capture* Media Stream**
Tujuan Uji Coba	Menguji fitur sistem *capture* media stream untuk mendapatkan gambar perangkat media.
Kondisi Awal	Inisialisasi Koneksi sudah dilakukan.
Skenario	**Membuka halaman web dengan fitur media stream dan mengiijinkan peramban untuk mengakses perangkat media stream.**
Masukan	**Perangkat Media Stream**
Keluaran	Tampilan capture media stream pada halaman web.
Hasil Uji Coba	**Berhasil**

Gambar 5.3 dan Gambar 5.4 merupakan hasil uji coba *capture* dari media stream. Seperti yang telah dijelaskan pada Tabel 5.2, tujuan dilakukan uji coba ini adalah untuk menguji fitur sistem *capture* media stream untuk mendapatkan gambar dari perangkat media komputer pengguna. Uji coba dilakukan dengan membuka halaman web dengan fiture media stream, kemudian peramban akan meminta ijin untuk mengakses perangkat media stream. Keluaran dari uji coba ini adalah tampilan *capture* media stream pada halaman web.



Gambar 5.3. Uji Coba *Capture* Media Stream



Gambar 5.4. Objek Media Stream

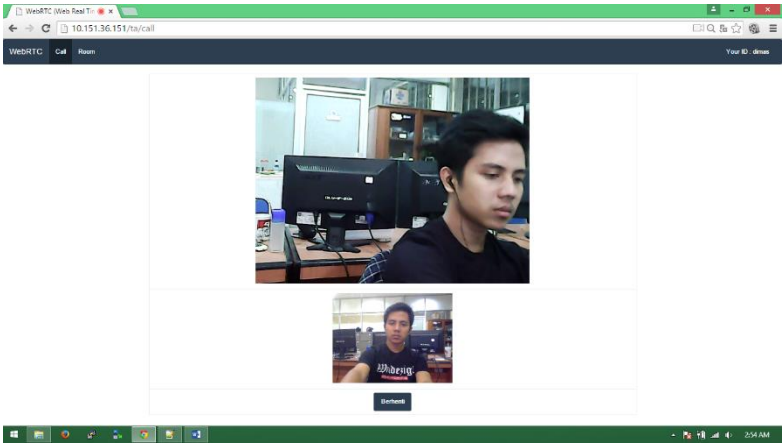
5.2.1.3. Uji Coba Koneksi Antar Peer

Uji coba ini dilakukan untuk menunjukkan koneksi yang terjadi antar peer. Uji coba dilakukan dengan memberikan identitas dari masing-masing peer, dan salah satu peer akan menghubungi peer lain menggunakan id yang telah diberikan. Sebelumnya media sudah dapat mengakses perangkat media stream. Ketika kedua peer sudah terhubung, masing-masing peer akan saling menerima data seperti video streaming dan akan ditampilkan dalam halaman web.

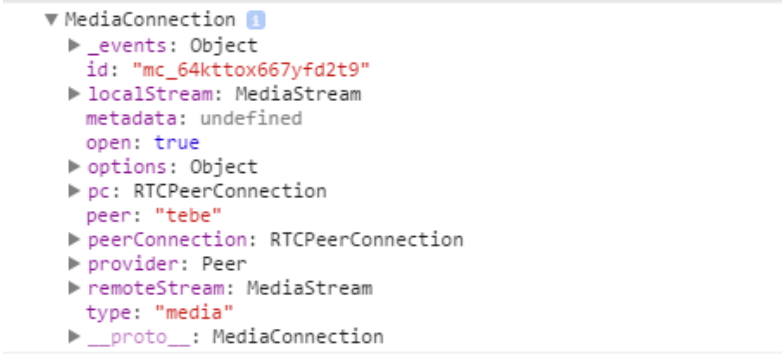
Tabel 5.3. Uji Coba Koneksi Antar Peer

ID	**UC-03**
Nama	**Uji Koneksi Antar Peer**
Tujuan Uji Coba	Menguji fitur sistem melakukan koneksi antar peer agar kedua peer dapat saling berhubungan
Kondisi Awal	Inisialisasi Koneksi sudah dilakukan dan streaming media sudah ada
Skenario	**Menghubungi peer lain yang ada di dalam *list online user***
Masukan	**-**
Keluaran	Tampilan media stream lokal dan *remote* stream pada halaman web.
Hasil Uji Coba	**Berhasil**

Gambar 5.5 dan Gambar 5.6 merupakan hasil uji coba dari koneksi antar peer. Dalam Tabel 5.3 dijelaskan tentang uji coba yang dilakukan untuk koneksi antar peer. Tujuan dari dilakukannya uji coba ini adalah untuk menguji fitur sistem melakukan koneksi antar peer agar kedua peer dapat saling berhubungan. Kondisi awal adalah inisialisasi sudah dilakukan oleh masing-masing peer dan media stream sudah ditampilkan dalam halaman web. Sekenario dilakukan dengan menghubungi peer lain yang ada di dalam *list online user*. Hasil yang didapatkan adalah tampilan media stream lokal dan *remote* stream pada halaman web.



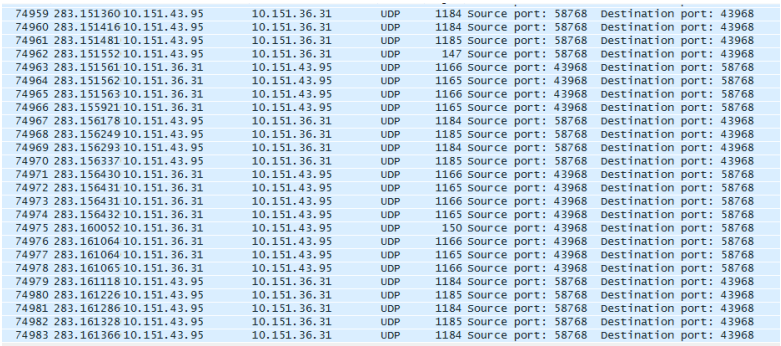
Gambar 5.5. Uji Coba Koneksi Antar Peer



Gambar 5.6. Objek MediaConnection dalam Koneksi Antar Peer

Pada percobaan ini komputer 1 menggunakan IP 10.151.36.31, sedangkan komputer 2 menggunakan IP 10.151.43.95. Untuk melihat *traffic* jaringan pada komputer digunakan aplikasi wireshark. Tujuan dari pengecekan *traffic* pada

jaringan adalah untuk menunjukkan pengiriman data yang dilakukan pada saat koneksi peer-to-peer terjadi. Gambar 5.7 menunjukkan bahwa pengiriman data pada koneksi peer-to-peer hanya melalui dua node yang telah terhubung dan tidak melaui server. IP 10.151.43.95 menggunakan port 58768 dan IP 10.151.36.31 menggunakan port 43968 serta menggunakan protokol UDP untuk saling mengirimkan data.



74959	283.151360	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968
74960	283.151416	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968
74961	283.151481	10.151.43.95	10.151.36.31	UDP	1185	Source port: 58768	Destination port: 43968
74962	283.151552	10.151.43.95	10.151.36.31	UDP	147	Source port: 58768	Destination port: 43968
74963	283.151561	10.151.36.31	10.151.43.95	UDP	1166	Source port: 43968	Destination port: 58768
74964	283.151562	10.151.36.31	10.151.43.95	UDP	1165	Source port: 43968	Destination port: 58768
74965	283.151563	10.151.36.31	10.151.43.95	UDP	1166	Source port: 43968	Destination port: 58768
74966	283.155921	10.151.36.31	10.151.43.95	UDP	1165	Source port: 43968	Destination port: 58768
74967	283.156178	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968
74968	283.156249	10.151.43.95	10.151.36.31	UDP	1185	Source port: 58768	Destination port: 43968
74969	283.156293	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968
74970	283.156337	10.151.43.95	10.151.36.31	UDP	1185	Source port: 58768	Destination port: 43968
74971	283.156430	10.151.36.31	10.151.43.95	UDP	1166	Source port: 43968	Destination port: 58768
74972	283.156431	10.151.36.31	10.151.43.95	UDP	1165	Source port: 43968	Destination port: 58768
74973	283.156431	10.151.36.31	10.151.43.95	UDP	1166	Source port: 43968	Destination port: 58768
74974	283.156432	10.151.36.31	10.151.43.95	UDP	1165	Source port: 43968	Destination port: 58768
74975	283.160052	10.151.36.31	10.151.43.95	UDP	150	Source port: 43968	Destination port: 58768
74976	283.161064	10.151.36.31	10.151.43.95	UDP	1166	Source port: 43968	Destination port: 58768
74977	283.161064	10.151.36.31	10.151.43.95	UDP	1165	Source port: 43968	Destination port: 58768
74978	283.161065	10.151.36.31	10.151.43.95	UDP	1166	Source port: 43968	Destination port: 58768
74979	283.161118	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968
74980	283.161226	10.151.43.95	10.151.36.31	UDP	1185	Source port: 58768	Destination port: 43968
74981	283.161286	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968
74982	283.161328	10.151.43.95	10.151.36.31	UDP	1185	Source port: 58768	Destination port: 43968
74983	283.161366	10.151.43.95	10.151.36.31	UDP	1184	Source port: 58768	Destination port: 43968

Gambar 5.7. *Traffic* Jaringan Pada Saat Koneksi Peer-to-Peer Terjadi

5.2.1.4. Uji Coba Estimasi Kondisi Jaringan

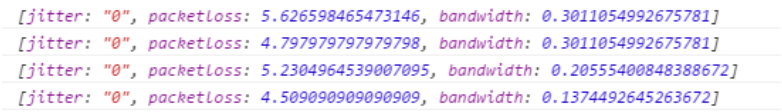
Uji coba ini dilakukan untuk menghitung estimasi kondisi jaringan yang ada pada suatu waktu. Uji coba dilakukan dengan menghubungan antar peer terlebih dahulu. Estimasi kondisi jaringan dilakukan pada peer server. Pada saat kedua peer saling berhubungan, peer akan saling mengirimkan data. Pada saat itulah estimasi kondisi jaringan dilakukan.

Gambar 5.8 merupakan hasil uji coba dari estimasi kondisi jaringan. Terdapat 3 nilai yang didapat yaitu *jitter*, *packet-loss*, dan *available bandwidth*. Dari Tabel 5.4 dijelaskan bahwa tujuan dari estimasi kondisi jaringan adalah untuk menentukan kondisi jaringan yang ada pada suatu waktu. Uji coba dilakukan dengan melakukan hubungan antara dua peer dan paket data dikirim.

Kondisi jaringan yang ada pada saat pengiriman paket antara peer klien dan peer server.

Tabel 5.4. Uji Coba Estimasi Kondisi Jaringan

ID	**UC-04**
Nama	**Uji Estimasi Kondisi Jaringan**
Tujuan Uji Coba	Menentukan kondisi jaringan yang ada pada suatu waktu.
Kondisi Awal	Hubungan antara dua peer sudah terjadi.
Skenario	**Salah satu peer menghubungi peer lain. Ketika pengiriman data, estimasi kondisi jaringan dilakukan.**
Masukan	**RTCPeerConnection**
Keluaran	*Jitter*, *packet-loss*, dan *available bandwidth*
Hasil Uji Coba	**Berhasil**



Gambar 5.8. Estimasi Kondisi Jaringan

5.2.1.5. Uji Coba Eksekusi Strategi Adaptasi

Uji coba ini dilakukan untuk eksekusi strategi adaptasi oleh peer sesuai dengan kondisi jaringan yang ada. Uji coba dilakukan dengan mendapatkan estimasi kondisi jaringan terlebih dahulu.

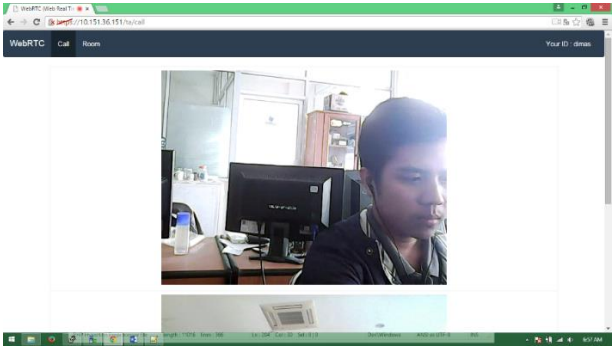
Hasil estimasi kondisi jaringan dijadikan sebagai masukan untuk perhitungan logika fuzzy. Hasil tersebut merupakan konstrain dari media streaming klien. Eksekusi adaptasi dilakukan pada peer klien.

Tabel 5.5. Uji Coba Eksekusi Strategi Adaptasi

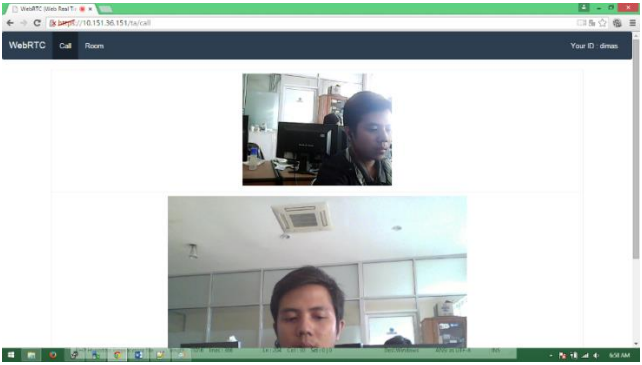
ID	**UC-05**
Nama	**Uji Eksekusi Strategi Adaptasi**
Tujuan Uji Coba	Menentukan konstrain yang sesuai dengan kondisi jaringan untuk mengirim data.
Kondisi Awal	Hubungan antara dua peer sudah terjadi.
Skenario	**Ketika sedang melakukan koneksi antar peer, jaringan yang ada pada peer klien dinaikkan jitter atau packet-loss nya.**
Masukan	**Data hasil perhitungan logika fuzzy**
Keluaran	Perubahan resolusi sesuai dengan kondisi jaringan yang telah ditentukan
Hasil Uji Coba	**Berhasil**

Gambar 5.9 menunjukkan awal mula koneksi peer terjadi dengan resolusi standar yaitu 640x480. Ketika kondisi jaringan mulai tidak stabil, makan konstrain media stream juga berpengaruh. Terlihat pada Gambar 5.10 perubahan resolusi yang terjadi. Resolusi yang digunakan adalah hasil perhitungan logika fuzzy yang dikirimkan kepada peer klien dari peer server karena peer server yang bertugas untuk menghitung estimasi kondisi jaringan dan logika fuzzy. Pada Gambar 5.11 dan Gambar 5.12

merupakan hasil yang didapat dari perhitungan logika fuzzy yang ada pada peer server dan peer klien. Hasil tersebut yaitu resolusi (*width* dan *height*) yang ditentukan berdasarkan member dari logika fuzzy yang telah ditulis pada bab sebelumnya. Dari Tabel 5.5 dijelaskan bahwa tujuan dari eksekusi strategi adaptasi adalah untuk memberikan konstrain yang tepat kepada peer untuk mengirimkan data kepada peer lain yang berfungsi agar kualitas layanan dari *live* streaming multimedia peer-to-peer.



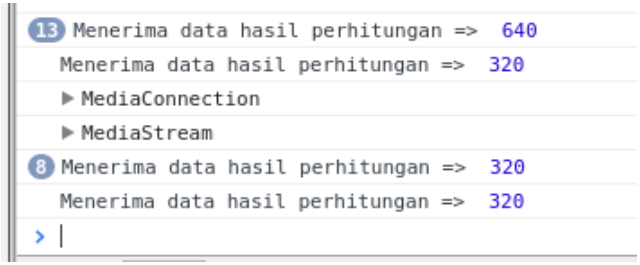
Gambar 5.9. Kondisi Awal Koneksi Peer Terjadi



Gambar 5.10. Perubahan yang Terjadi Terhadap Video Stream Klien

```
Kirim hasil kepada peer klien =>  640
Hasil perhitungan funzzy =  620
Kirim hasil kepada peer klien =>  640
Hasil perhitungan funzzy =  609
Kirim hasil kepada peer klien =>  640
Hasil perhitungan funzzy =  584
Kirim hasil kepada peer klien =>  640
Hasil perhitungan funzzy =  574
Kirim hasil kepada peer klien =>  640
Hasil perhitungan funzzy =  328
Kirim hasil kepada peer klien =>  320
Hasil perhitungan funzzy =  328
Kirim hasil kepada peer klien =>  320
```

Gambar 5.11. Hasil Perhitungan Fuzzy pada Peer Server



Gambar 5.12. Hasil Perhitungan yang Telah Diterima Peer Klien

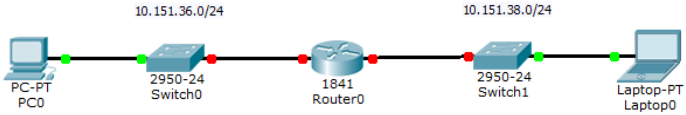
5.2.2. Uji Coba Performa

Pada bagian ini dilakukan uji coba performa untuk mengetahui perilaku dari sistem ketika dijalankan pada keadaan sebenarnya. Uji coba ini dilakukan menggunakan 2 skenario. Skenario pertama adalah untuk mengetahui kualitas layanan yang diberikan sistem kepada pengguna. Kualitas tersebut yaitu dengan menghitung setiap frame menggunakan PSNR untuk mengetahui kualitas video yang dikirimkan dan frame rate yang ditampilkan video. Sedangkan skenario kedua adalah untuk mengetahui

perubahan kondisi jaringan seperti *bandwidth*, *packet-loss*, dan *jitter*.

5.2.2.1.Uji Coba Evaluasi Kualitas Video

Skenario uji coba evaluasi kualitas video adalah dengan memberikan *network emulation* pada peer klien ketika kedua peer saling mengirimkan data. *Network emulation* yang diberikan kepada peer klien ditunjukkan pada Tabel 5.6, Tabel 5.7, dan Tabel 5.8. Uji coba dilakukan dengan merekam video lokal stream dari peer klien dan video remote stream dari peer server. Gambar 5.13 merupakan topologi jaringan uji coba. Uji coba dilakukan menggunakan dua komputer yang terhubung dengan jaringan lokal *Ethernet* dan subnet yang berbeda. Komputer 1 menggunakan jaringan lokal dengan subnet 10.151.36.0/24 dan Komputer 2 menggunakan Jaringan lokal dengan subnet 10.151.38.0/24.



Gambar 5.13. Topologi Jaringan Uji Coba Evaluasi Kualitas Video

Tabel 5.6. *Network Emulation* Percobaan 1

No.	Waktu	*Network Emulation*
1.	20	Pemberian *jitter* 25ms
2.	40	Pemberian *packet-loss* 5%
3.	60	Perubahan *jitter* 20ms

Tabel 5.7. *Network Emulation* Percobaan 2

No.	Waktu	*Network Emulation*
1.	20	Pemberian *packet-loss* 5%
2.	40	Pembatasan *bandwidth* 500 kbps
3.	60	Pemberian *jitter* 20ms

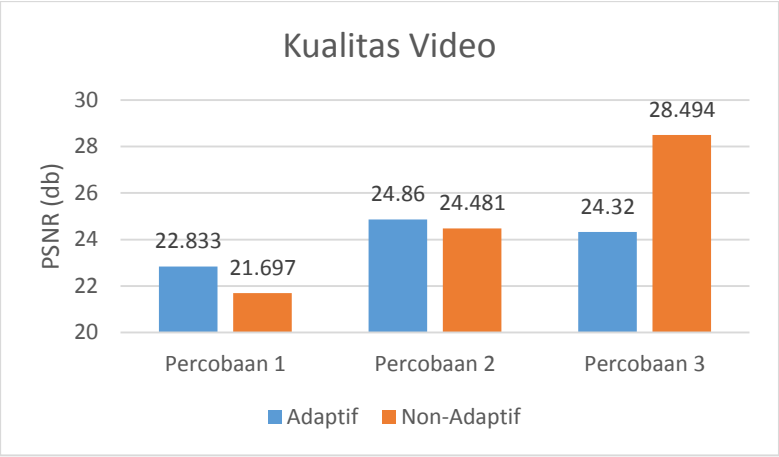
Tabel 5.8. *Network Emulation* Percobaan 3

No.	Waktu	*Network Emulation*
1.	20	Pembatasan *bandwidth* 500 kbps
2.	40	Pemberian *jitter* 20ms
3.	60	Pemberian *packet-loss* 10%

5.2.2.1.1. Evaluasi Kualitas Video dengan PSNR

Pada Tugas Akhir ini kualitas video dihitung menggunakan metode PSNR yang telah disediakan Video Quality Measurement Tool (VQMT). Hasil dari perhitungan PSNR terhadap video stream tanpa menggunakan konstrain dan perhitungan PSNR dan video stream dengan menggunakan konstrain dapat dilihat pada Gambar 5.14.

Dari Gambar 5.14 didapatkan nilai PSNR dari percobaan 1, 2, dan 3 dari rekaman video stream tanpa menggunakan adaptif streaming dan rekaman video stream menggunakan adaptif streaming.

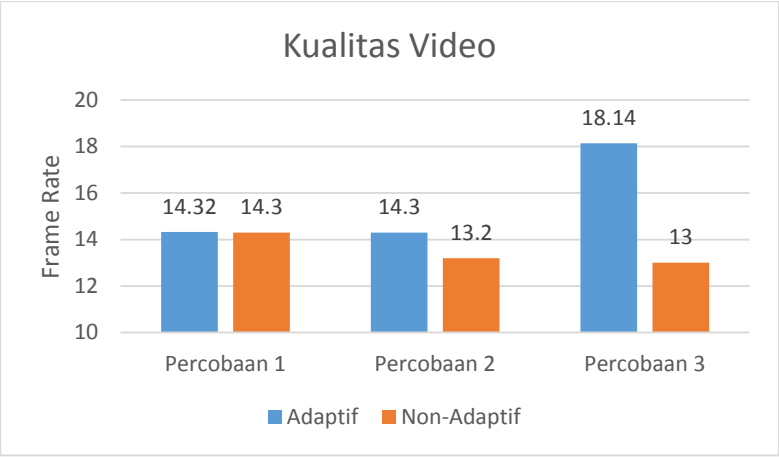


Gambar 5.14. Perbandingan Kualitas Video Terhadap Nilai PSNR

5.2.2.1.2. Evaluasi Kualitas Video dengan Frame Rate

Uji coba untuk menentukan kualitas video selain menggunakan PSNR, penentuan kualitas video dengan frame rate dari video tersebut. Karena video yang memiliki frame rate kecil, maka akan mengurangi *lagging* sehingga mengurangi kualitas dari video tersebut. Uji coba dilakukan hal yang sama dengan PSNR. Hasil dari perhitungan video stream tanpa menggunakan konstrain dan perhitungan frame rate dan video stream dengan menggunakan konstrain dapat dilihat pada Gambar 5.15.

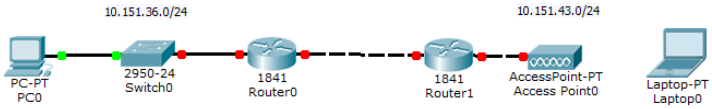
Dari Gambar 5.15 didapatkan nilai frame rate dari percobaan 1, 2, dan 3 dari rekaman video stream tanpa menggunakan adaptif streaming dan rekaman video stream menggunakan adaptif streaming.



Gambar 5.15. Perbandingan Kualitas Video Terhadap Nilai Frame Rate

5.2.2.2.Uji Coba Evaluasi Kondisi Jaringan

Skenario uji coba evaluasi kondisi jaringan adalah dengan simulasi pergerakan pengguna. Pengguna yang bergerak adalah peer klien dimana peer klien terhubung dengan *Access Point*. Pergerakan yang terjadi akan mempengaruhi perubahan kondisi jaringan terhadap peer klien pada jarak tertentu.

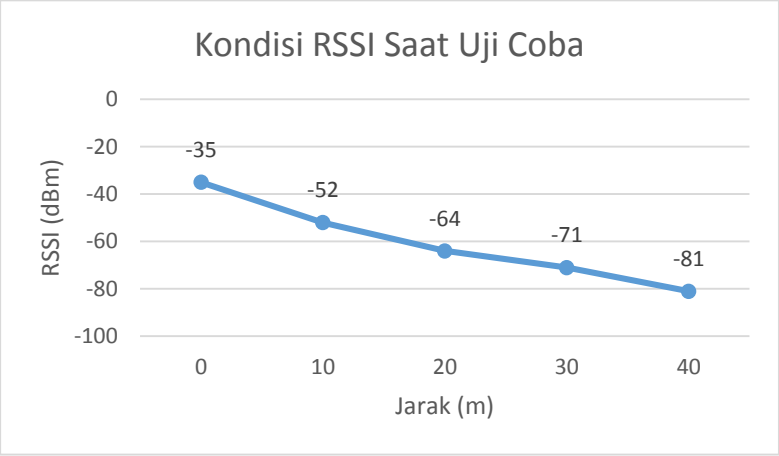


Gambar 5.16. Topologi Jaringan Uji Coba Evaluasi Kondisi Jaringan

Uji coba dilakukan menggunakan dua komputer. Komputer 1 terhubung dengan jaringan lokal *Ethernet* dengan subnet 10.151.36.0/24 sedangkan komputer 2 terhubung dengan jaringan lokal *Wireless* dengan subnet 10.151.43.0/24. Gambar 5.16 menunjukkan topologi jaringan yang digunakan untuk uji coba evaluasi kondisi jaringan.

5.2.2.2.1. Pengaruh Jarak Terhadap *Received Signal Strength Indicator* (RSSI)

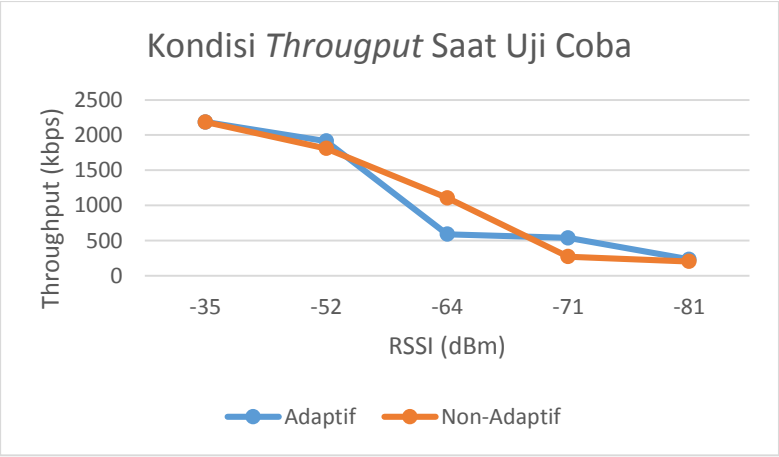
RSSI merupakan parameter yang menunjukkan daya terima dari seluruh sinyal pada band *frequency channel* pilot yang diukur. Parameter ini diukur pada arah downlink dengan acuan pengukuran pada konektor antena pada penerima. Pada uji coba ini didapatkan kondisi dari RSSI berdasarkan jarak yang dilakukan pada saat uji coba dilakukan. Kondisi RSSI pada saat uji coba ditunjukkan pada Gambar 5.17.



Gambar 5.17. Pengaruh Jarak Terhadap RSSI

5.2.2.2.2. Evaluasi Kondisi *Throughput*

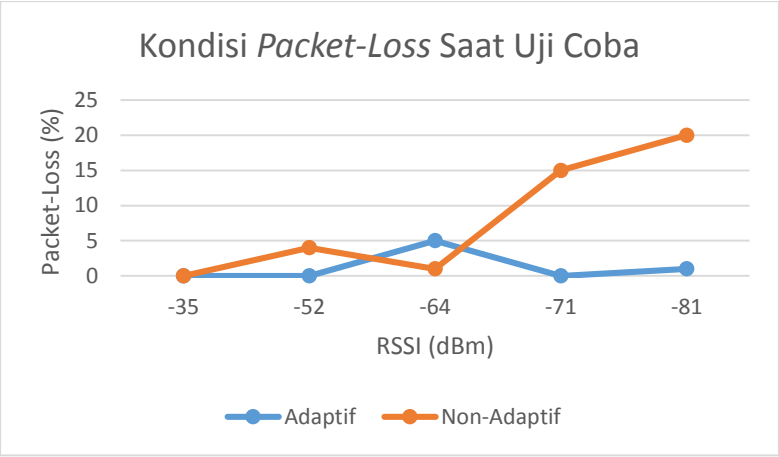
Dari hasil uji coba pada Gambar 5.18 tampak bahwa semakin menurun seiring bertambahnya jarak peer klien dari *Access Point*. Saat pengguna berada dekat dengan *access point*, *throughput* dapat mencapai 2185 kbps. RRSI pada jarak terjauh yaitu 40 m adalah -81 dBm meter dengan *throughput* 233 kbps.



Gambar 5.18. Perbandingan Kondisi *Throughput* Berdasarkan RSSI

5.2.2.2.3. Evaluasi Kondisi *Packet-Loss*

Dari hasil uji coba pada Gambar 5.19 tampak bahwa *packet-loss* dari setiap jarak yang ditentukan. Uji coba tanpa menggunakan konstrain mengalami kenaikan pada jarak 20 sampai 40 meter yaitu dengan RSSI -71 dBm dan -81 dBm. Aplikasi masih memiliki nilai *packet-loss* yang sangat kecil jika dilihat pada Tabel 5.9 tentang kategori *packet-loss* menurut tiphon.



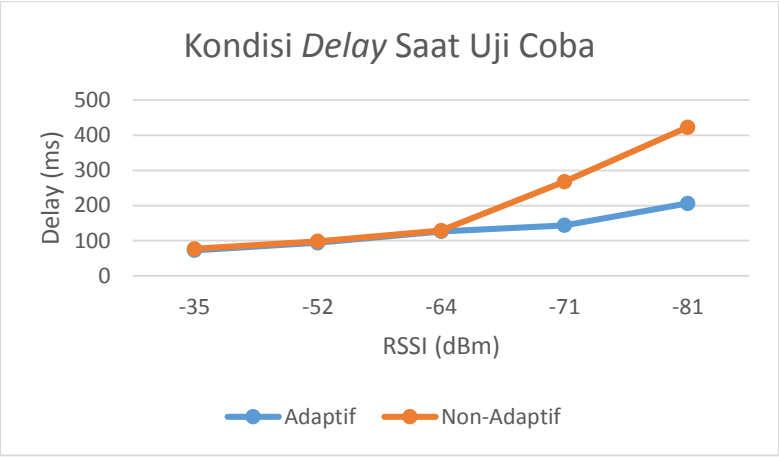
Gambar 5.19. Perbandingan Kondisi *Packet-Loss* Berdasarkan RSSI

Tabel 5.9. Kategori *Packet-Loss* Menurut Tiphon [15]

Kategori *Packet-Loss*	*Packet-Loss* (%)
Sangat Bagus	0
Bagus	3
Kurang	15
Sangat Kurang	25

5.2.2.2.4. Evaluasi Kondisi *Delay*

Dari hasil uji coba pada Gambar 5.20 tampak bahwa semakin meningkat seiring bertambahnya jarak peer klien dari *Access Point*. Uji coba menggunakan konstrain memiliki *delay* lebih rendah dibandingkan dengan tanpa menggunakan konstrain. Aplikasi masih memiliki nilai *delay* yang bagus jika dilihat pada Tabel 5.10 tentang kategori *delay* menurut ITU-T G. 114.



Gambar 5.20. Perbandingan Kondisi *Delay* Berdasarkan RSSI

Tabel 5.10. Kategori *Delay* Menurut ITU-T G. 114 [16]

Kategori *Delay*	***Delay* (ms)**
Sangat Bagus	< 150
Bagus	150 – 300
Kurang	300 – 450
Sangat Kurang	> 450

BAB VI
PENUTUP

Pada bab ini akan dibahas mengenai mengenai kesimpulan yang dapat diambil dari tujuan rancang bangun aplikasi serta hasil coba yang telah dilakukan pada Tugas Akhir ini. Selain itu juga terdapat beberapa saran untuk pengembangan aplikasi lebih lanjut.

6.1.Kesimpulan

Berdasarkan hasil pengamatan, perancangan, implementasi dan uji coba aplikasi, maka dapat diambil beberapa kesimpulan dari hasil pembuatan tugas akhir ini, yaitu:

1. Sistem aplikasi adaptif live streaming multimedia peer-to-peer dapat membuat hubungan peer dari pengguna kepada server.
2. Sistem aplikasi adaptif live streaming multimedia peer-to-peer dapat mengakses perangkat media komputer pengguna dari peramban.
3. Sistem aplikasi adaptif live streaming multimedia peer-to-peer dapat menentukan estimasi kondisi jaringan dari hubungan peer yang sedang terjadi.
4. Sistem aplikasi adaptif live streaming multimedia peer-to-peer dapat menentukan kondisi adaptif streaming yang tepat pada saat peer saling berhubungan.
5. Kualitas video dengan adaptif streaming menggunakan PSNR didapatkan rata-rata 24 db
6. Kualitas video dengan adaptif streaming menggunakan frame rate didapatkan rata-rata 17 fps
7. Mekanisme adaptif streaming yang diajukan berhasil menunjukkan hasil yang lebih baik daripada native streaming pada WebRTC dilihat berdasarkan uji coba dan evaluasi.

6.2.Saran

Berdasarkan hasil pengamatan, perancangan, implementasi, serta hasil uji coba aplikasi ini, maka diperlukan beberapa saran untuk pengembangan aplikasi lebih lanjut, yaitu:

1. Penambahan jumlah data training untuk meningkatkan keakuratan hasil uji coba.
2. Penginkatan kualitas video PSNR dan frame rate dengan menggunakan logika fuzzy.
3. Mendapatkan cara untuk mengganti konstrain video pada WebRTC dengan tepat.

DAFTAR PUSTAKA

[1] Novella, J. M., & Castano, F. G. (n.d.). QoS Requirements for Multimedia Service.

[2] Salerno, P. (2013, November 23). *WebRTC 101*. Retrieved from Low Profile: http://petersalerno.com/webrtc-101/

[3] Loreto, S., & Romano, S. P. (2014). *Real-Time Communication with WebRTC.* California: O'Reilly Media Inc.

[4] 3scale Networks S.L. (2011). What is an API? Your Guide to the Internet Business (R)evolution. *3scale*, 4.

[5] Cantelon, M., Harter, M., Holowaychuk, T., & Rajlich, N. (2014). *Node.js In Action.* New York: Manning Publications Co.

[6] Peerjs. (2015, May 27). *Peerjs-Server*. Retrieved from Peerjs: http://peerjs.com

[7] W3C. (2014, August 1). *MediaStream Recording.* Retrieved from W3C: http://www.w3.org/TR/mediastream-recording/

[8] MySQL AB. (2002). *MySQL Reference Manual.* MySQL AB.

[9] Doyle, M. (2019). *Beginning PHP 5.3.* Indianapolis: Wiley Publishing.

[10] Certified Internet Webmaster. (2002). *JavaScript Fundamentals: Academic Student Guide.* Prosoft Training.

[11] Batra, S. (2003). *AJAX - Asynchronous Java Script and XML*. Salzburg: Information Technology and System Management.

[12] Linux Foundation. (2009, November 19). *netem*. Retrieved from Linux Foundation: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[13] Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, 338-353.

[14] Wolf, S. (2009). Reference Algorithm for Computing Peak Signal to Noise Ratio (PSNR) of a Video Sequence with a Constant Delay.

[15] ETSI. (2000). Telecommunications and Internet Protocol Harmonization Over Network (TIPHON). *End to End Quality of Service in TIPHON Systems*.

[16] ITU-T. (2003). Series G: Transmission system and media, digital system and networks. *One Way Transmission*.

LAMPIRAN

Pada bagian ini adalah lampiran sebagai dokumen pelengkap dari buku Tugas Akhir. Pada bagian ini akan diberikan potongan kode sumber dari fungsi-fungsi dan matrik tiga dimensi aturan kontrol fuzzy yang digunakan untuk membangun aplikasi.

1. Fungsi Inisialisasi Koneksi Peer, Capture Media Stream, dan Hubungan dengan Peer Lain

```
navigator.getUserMedia = navigator.getUserMedia
||        navigator.webkitGetUserMedia        ||
navigator.mozGetUserMedia;

var peer = new Peer('<?php echo $username; ?>',
{host: '10.151.36.206', port: 9000});

function calling(){
<?php
if($call_to_id!=''){
      echo          "var          call          =
peer.call('".$call_to_id."',
window.localStream);";
      echo          "var          conn          =
peer.connect('".$call_to_id."');";
      }
?>
step3(call, conn);
}

call.on('stream', function(stream){
            console.log('masuk sini');
            $('#their-video').prop('src',
URL.createObjectURL(stream));
            window.remoteStream = stream;
 });
```

2. Fungsi Estimasi Kondisi Jaringan

```
function getStats(peer, conn) {
  if(peer)_getStats(peer.pc, function (results)
{

    var result = {
      audio: {},
      video: {},
      results: results
    };
    var stat = []
    for (var i = 0; i < results.length; ++i) {
      var res = results[i];
      if(res.googCodecName == 'opus'){
                  stat['jitter'] =
res.googJitterReceived;
      }
        if(res.type == 'VideoBwe'){
                  stat['bandwidth'] =
res.googAvailableReceiveBandwidth/1048576;
        }
        if(res.packetsReceived){
                  stat['packetloss'] =
(res.packetsLost / res.packetsReceived)*100;
        }
    }
      console.log(stat);
}

function _getStats(peer, cb) {
  if (!!navigator.mozGetUserMedia) {
    peer.getStats(
      function (res) {
        var items = [];
        res.forEach(function (result) {
          items.push(result);
        });
        cb(items);
```

```
      },
    cb
    );
  } else {
    peer.getStats(function (res) {
      var items = [];
      res.result().forEach(function (result) {
        var item = {};
        result.names().forEach(function (name) {
          item[name] = result.stat(name);
        });
        item.id = result.id;
        item.type = result.type;
        item.timestamp = result.timestamp;
        items.push(item);
      });
    cb(items);
    });
  }
}

function merge(mergein, mergeto) {
        if (!mergein) mergein = {};
        if (!mergeto) return mergein;

        for (var item in mergeto) {
            mergein[item] = mergeto[item];
        }
        return mergein;
}
```

3. Fungsi Logika Fuzzy

```
function fuzzy(result) {
            var F = new FuzzyLogic();
            F.clearMembers();

      F.setInputNames(['bandwidth','jitter','pac
ketloss']);
```

```
            F.addMember('bandwidth','rendah',  0,
0.1, 0.7, LINFINITY);
            F.addMember('bandwidth','sedang',
0.3, 1, 1.5, TRIANGLE);
            F.addMember('bandwidth','tinggi',
1.1, 2, 10, RINFINITY);

            F.addMember('jitter','rendah',  0, 0,
30 ,LINFINITY);
            F.addMember('jitter','sedang',    20,
35, 50 ,TRIANGLE);
            F.addMember('jitter','tinggi',    40,
55, 200 ,RINFINITY);

            F.addMember('packetloss','rendah',
0, 1, 5 ,LINFINITY);
            F.addMember('packetloss','sedang',
3, 10, 17 ,TRIANGLE);
            F.addMember('packetloss','tinggi',
15, 23, 100,RINFINITY);

            F.setOutputNames(['res']);

            F.addMember('res','low_res',     180,
210, 270 ,LINFINITY);
            F.addMember('res','mid_res'   ,  240,
320, 500 ,TRIANGLE);
            F.addMember('res','hgh_res'   ,  480,
640, 720,RINFINITY);

            F.clearRules();
            F.addRule('IF  bandwidth.rendah  THEN
res.low_res');

            F.addRule('IF  bandwidth.sedang  AND
packetloss.tinggi THEN res.low_res');
            F.addRule('IF  bandwidth.sedang  AND
packetloss.sedang    AND    jitter.tinggi    THEN
res.low_res');
```

```
            F.addRule('IF  bandwidth.sedang  AND
packetloss.sedang    AND    jitter.sedang    THEN
res.mid_res');
            F.addRule('IF  bandwidth.sedang  AND
packetloss.sedang    AND    jitter.rendah    THEN
res.mid_res');
            F.addRule('IF  bandwidth.sedang  AND
packetloss.rendah THEN res.mid_res');

            F.addRule('IF  bandwidth.tinggi  AND
packetloss.rendah THEN res.hgh_res');
            F.addRule('IF  bandwidth.tinggi  AND
packetloss.sedang THEN res.mid_res');
            F.addRule('IF  bandwidth.tinggi  AND
packetloss.tinggi THEN res.low_res');

            var bw = result['bandwidth'];
            var j = result['jitter'];
            var pl = result['packetloss'];
            F.setRealInput('bandwidth'  , bw );
            F.setRealInput('packetloss' , pl );
            F.setRealInput('jitter' , j   );
            var fuzzy_arr = F.calcFuzzy();

            var              width              =
(fuzzy_arr['res']).toFixed(0);
            //var             height             =
((fuzzy_arr['res']).toFixed(0)/4)*3;

            return width;
```

4. Fungsi Eksekusi Strategi Adaptasi

```
conn.on('data',function(data){
console.log('Menerima data hasil perhitungan =>
', data);
if(data!=window.streamWidth){
      window.existingCall.close();
      window.streamWidth = data;
```

```
        step1(data, (data/4)*3);
        conn.close();
        setTimeout(function () {
                calling();
                }, 1000);
        }
        console.log(call);

});
```

5. Matrik Tiga Dimensi Aturan Kontrol Fuzzy

Available Bandwidth	Packet-Loss	Jitter	Resolusi
Rendah	Rendah	Rendah	Rendah
Rendah	Rendah	Sedang	Rendah
Rendah	Rendah	Tinggi	Rendah
Rendah	Sedang	Rendah	Rendah
Rendah	Sedang	Sedang	Rendah
Rendah	Sedang	Tinggi	Rendah
Rendah	Tinggi	Rendah	Rendah
Rendah	Tinggi	Sedang	Rendah
Rendah	Tinggi	Tinggi	Rendah
Sedang	Rendah	Rendah	Sedang

Available Bandwidth	Packet-Loss	Jitter	Resolusi
Sedang	Rendah	Sedang	Sedang
Sedang	Rendah	Tinggi	Sedang
Sedang	Sedang	Rendah	Sedang
Sedang	Sedang	Sedang	Sedang
Sedang	Sedang	Tinggi	Tinggi
Sedang	Tinggi	Rendah	Rendah
Sedang	Tinggi	Sedang	Rendah
Sedang	Tinggi	Tinggi	Rendah
Tinggi	Rendah	Rendah	Tinggi
Tinggi	Rendah	Sedang	Tinggi
Tinggi	Rendah	Tinggi	Tinggi
Tinggi	Sedang	Rendah	Sedang
Tinggi	Sedang	Sedang	Sedang
Tinggi	Sedang	Tinggi	Sedang
Tinggi	Tinggi	Rendah	Rendah
Tinggi	Tinggi	Sedang	Rendah

Available Bandwidth	Packet-Loss	Jitter	Resolusi
Tinggi	Tinggi	Tinggi	Rendah

NB : Setiap warna digabungkan menjadi satu aturan

: IF Available Bandwidth Rendah THEN Resolusi Rendah

: IF Available Bandwidth Sedang AND Packet-Loss Rendah THEN Resolusi Sedang

: IF Available Bandwidth Sedang AND Packet-Loss Tinggi THEN Resolusi Rendah

: IF Available Bandwidth Tinggi AND Packet-Loss Rendah THEN Resolusi Tinggi

: IF Available Bandwidth Tinggi AND Packet-Loss Sedang THEN Resolusi Sedang

: IF Available Bandwidth Tinggi AND Packet-Loss Tinggi THEN Resolusi Rendah

BIODATA PENULIS

Penulis dilahirkan di Banyuwangi, 24 Februari 1993 merupakan anak pertama dari 3 bersaudara. Penulis menempuh pendidikan formal di SD Negeri 2 Ketapang, SMP Negeri 1 Banyuwangi, dan SMA Negeri 1 Glagah Banyuwangi. Pada tahun 2011, penulis melanjutkan S1 di Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Selama menempuh pendidikan S1 di Teknik Informatika ITS, penulis menekuni bidang minat komputasi berbasis jaringan atau lebih dikenal dengan *Net Centric Computing*.

Penulis aktif sebagai anggota organisasi mahasiswa di jurusan yaitu Himpunan Mahasiswa Teknik Computer (HMTC) sebagai Kepala Depatmemen Kewirausahaan dan Minat Bakat (KMB). Penulis juga terlibat pada kegiatan Schematics 2013 sebagai Staff REEVA. Selama masa perkuliahan penulis pernah berkesempatan menjadi asisten praktikum pada mata Sistem Digital, Sistem Operasi, dan Jaringan Komputer. Penulis dapat dihubungi melalui email dimasranggaf@gmail.com.