Implementasi Teknologi Java *Media Framework* (JMF) Untuk Komunikasi Suara

Ferdynand Kesi Tandawuja, Wiranto Herry Utomo, Theophillus Erman Wellem

Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana
Jl. Diponegoro 52-60, Salatiga 50711, Indonesia
E-mail: ferdy_taz@yahoo.com, whu2whu@yahoo.do.id,
erman_wellem@yahoo.com

Abstract

Demanding to get information by simple and quickly is just one requirement that wouldn't avoid again, with the technology information that look it up at now could be support the human for receive and give information as quickly and effective. Even today the technology begin move to sending information that might could be represent an communications encoded by multimedia built in real time. JMF it is interface multimedia application that developed by SUN MICROSYSTEM and could be apply for all operation system into using do not accordance that too much. How it this, will be implemented any voice application or voice by technology JMF implemented till every computer that found in the office room or building through networking (LAN) as Peer to Peer could be communicated to one and each other.

Keywords: Java Media Framework, Real Time Protocol, Real Time Communication.

1. Pendahuluan

Perkembangan teknologi saat ini begitu cepat, hal ini terbukti bahwa hampir semua pekerjaan dapat dilakukan dengan menggunakan bantuan teknologi. Salah satu contoh, dalam satu ruang kerja atau gedung yang masing-masing komputer sudah terhubung dalam sebuah jaringan, jika ingin berkomunikasi tidak perlu lagi mengeluarkan suara yang keras dan menggunakan alat *intercom* cukup menggunakan komputer untuk berkomunikasi menggunakan informasi suara secara *real time* menggunakan teknologi *JMF* [1].

JMF merupakan *interface* aplikasi multimedia yang dikembangkan oleh *Sun Microsystem* dan dapat bekerja pada semua sistem operasi sehingga dalam penerapannya tidak memerlukan penyesuaian yang begitu banyak untuk dapat dioperasikan.

JMF mempunyai arsitektur yang menggabungkan protokol dan pemrograman *interface* untuk merekam, mentransmisi, dan *playback* media dengan pemrosesan *time-base media* yang digunakan untuk mengubah data yang diterima dengan berdasarkan waktu, termasuk didalamnya *audio* dan *video*, *MIDI*, dan animasi menggunakan *RTP* [2].

2. Tinjauan Pustaka dan Landasan Teori

2.1. Hasil Penelitian Sebelumnya

Implementasi teknologi JMF yang dikembangkan oleh *SUN Microsystem* telah mengalami beberapa proses modifikasi yang dilakukan oleh pihak-pihak yang membahas penelitian tentang *JMF* itu sendiri antara lain:

1. Analisis kolaborasi antara JMF, Java 3D dan Java untuk prototype pembuatan MPEG-4 [3]
2. Aplikasi *webcam* dengan *JMF*, yang digunakan untuk menangkap gambar dan menyimpannya dalam format *file jpeg* dan menghitung waktu tunda ketika mengirim gambar dari *webcam* [4]
3. Penelitian tentang komunikasi audio streaming yang digunakan untuk *bluetooth device* [5]
4. Pembuatan aplikasi *Java TV* yang dijalankan melalui internet [6]

Berdasarkan beberapa temuan dan penelitian diatas muncul suatu ide untuk membuat suatu rancangan aplikasi berupa *audio* atau suara secara *real time* menggunakan teknologi *JMF* yang akan digunakan untuk berkomunikasi antara komputer dalam jaringan *peer to peer*.

2.2. Landasan Teori

2.2.1. Java Media Framework

JMF adalah bagian dari *Java Sound Engine* yang merupakan *interface* aplikasi multimedia yang dapat bekerja baik pada *Microsoft windows* maupun *linux* sehingga dalam penerapannya tidak memerlukan penyesuaian yang begitu banyak untuk dapat dioperasikan pada paltform-platform tersebut. *Sun Microsystem* sebagai perusahaan pengembang bahasa pemrograman *Java* berinisiatif untuk membawa pemrosesan *time-base media* kedalam bahasa pemrograman *Java*. *Time-base media* adalah proses yang dilakukan untuk mengubah data yang diterima dengan berdasarkan waktu, termasuk didalamnya seperti *audio,video, video klip,* file *MIDI*, dan animasi.

Beberapa fungsi dari *JMF* adalah :

1. Dapat digunakan untuk berbagai file multimedia pada Java Applet atau aplikasi. Format yang mendukung antara lain *AU, AVI, MIDI, MPEG, WAV*, dan beberapa file audio yang didukung oleh Java.
2. Memutar media *streaming* dari internet
3. *Capture* audio dan video dengan mikropon dan kamera video kemudian menyimpan data tersebut kedalam format yang mendukungnya.
4. Mengolah media *time-based* dan mengubah format *content-type.*
5. Mengirimkan audio dan video secara *realtime* ke dalam jaringan internet atau intranet.
6. Dapat digunakan untuk pemrograman penyiaran radio atau televisi secara langsung.

2.2.2. Real Time Transport Protocol (RTP)

RTP adalah protokol yang *header format* dan kontrolnya didesain untuk mendukung aplikasi-aplikasi transmisi data *real-time* seperti *audio, video*, dan juga simulasi data melalui layanan jaringan.

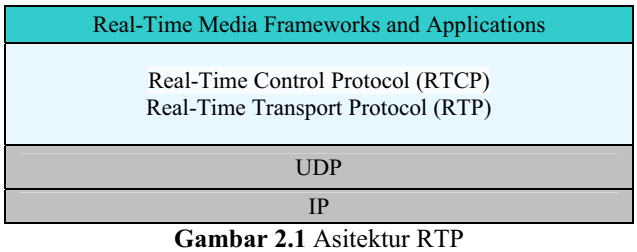
Pada *TCP* pemrograman yang berorientasi pada koneksitas (*connection-oriented programming*), dimana *client* dan *server* menjaga koneksitas selama komunikasi berlangsung hingga data diterima dan komunikasi diakhiri. Kelebihan dari tipe ini adalah jaminan bahwa semua data, dalam bentuk paket data yang dikirim oleh *server* akan diterima di *client*, sedangkan pada *UDP* tidak berorientasi pada koneksitas (*connectionless*) dimana setiap paket data dikirim secara terpisah tanpa ada hubungan antara *client* dan *server* setelah paket data dilepas oleh *server*. Kelebihannya adalah kecepatan transfer data data *server* ke *client* yang lebih tinggi daripada TCP.

JMF dapat mentransmisikan dan playback *RTP* stream dengan *API* yang terdapat pada javax.media.rtp, javax.media.event, dan javax.media.rtp.rtcp. Pada *RTP receiver*, dapat dilakukan *playback* atau menerima media data yang dikirimkan oleh *RTP transmitter.*

2.2.3. Real Time Transport Control Protocol (RTCP)

RTCP merupakan bagian dari *RTP*, berfungsi sebagai pendamping *RTP* untuk mengontrol komunikasi secara berkala pada paket transmisi yang menggunakan mekanisme distribusi paket data agar bisa digunakan oleh *receivers* untuk sinkronisasi audio dan video. *RTCP* akan mengirimkan paket secara berkala kepada *participants* yang teridentifikasi mengenai data yang dikirim dan yang diterima. Beberapa tipe paket *RTCP* tersebut adalah sebagai berikut:

1. *Sender Report*, *Sender Report* adalah informasi yang akan diperoleh jika *participant* telah mengirim data paket RTP setelah laporan yang terakhir, bersamaan saat itu *Receiver Report* juga di informasikan.
2. *Receiver Report*, sama seperti halnya *Sender Report*, hanya saja dilakukan pada sisi penerima.
3. *Source Description, Source description* akan menunjukkan identitas *pariticipants*. Pada SDES, *Canonical Name* merupakan item yang harus dikirimkan pada setiap paket *RTCP* karena mengandung informasi identitas serta lokasi sumber pengirim.
4. *Stop* (*end*), Paket *Stop* merupakan paket yang akan memberikan informasi *participant* yang tidak aktif atau yang akan meninggalkan session. Ketika *participants* meninggalkan session maka paket *stop* akan terkirim.



Gambar 2.1 Asitektur RTP

2.2.4. Arsitektur JMF

Dalam pemrograman JMF untuk membuat *interface* dan menghubungkan antar *class* terdapat beberapa langkah yang harus diperhatikan yaitu [10] [11]:

1. Capture Device
   Capture device merupakan hardware yang akan digunakan untuk pengambilan media data,seperti mikropon, camera atau video kamera, hasil dari pengambilan media data kemudian diberikan pada Player untuk dikonversi agar bisa ditampilkan atau disimpan untuk dipakai kembali sebagai data.
   Capture device dapat digolongkan sebagai push atau pull source, dengan pull source user bisa mengatur kapan akan mengambil gambar dengan menggunakan kamera dan mikropon digunakan untuk mengambil data audio secara stream.

2. DataSource
   DataSource berfungsi untuk mengatur transfer dari media. Pada JMF Data Source diidentifikasi dengan MediaLocator. Data source bisa membuat media stream seperti CD musik, pada JMF objek datasource dapat merepresentasikan media audio, media video, ataupun kombinasi dari kedua media tersebut. Data source bisa berupa file dan streaming yang terdapat pada jaringan, kelebihan class ini adalah jika sudah ditentukan lokasi atau protokol tujuan, data source langsung mengenali lokasi media, protokol dan software untuk mengirim media stream dan data source bisa memberikan informasi kepada player tanpa harus mencari asal dari data source yang akan digunakan. Media data bisa diperoleh dari file lokal, file jaringan ataupun broadcast live internet, data source diklasifikasikan menurut inisialisasi data yang akan ditransfer yaitu:
   1. Pull data source : Client melakukan inisialisasi terhadap data yang akan ditransfer dan mengotrol data *Flow* dari sourcenya, sebagai contoh *HTTP* dan *File Server*.
   2. Push data source : Server melakukan inisilisasi terhadap data yang akan ditransfer dan mengontrol data Flow dari sourcenya, sebagai contoh Broadcast media dan Video on Demand
3. Player
   Player memperoleh input *stream* data audio dan video kemudian mengirimnya ke speaker atau layar. Player merupakan *interface* yang akan mempersiapkan suatu DataSource untuk dipresentasikan.
4. Processor
   Processor merupakan jenis dari *Player*. Dalam JMF API, *Processor* adalah *interface* dengan *extends Player*. Seperti halnya *Player*, *Processor* juga mendukung untuk kontrol menampilkan media data. Disamping enam langkah *Player* seperti yang dibahas sebelumnya, *Processor* memasukkan dua langkah sebelum proses ke Realizing tetapi setelah Unrealized.

1. Configuring : *Processor* masuk ke tahapan konfigurasi dari Unrealized ketika metode configure dipanggil. *Processor* berada di Configuring setelah terhubung ke d*ataSource*.
2. Configured : setelah Configuring, *Processor* pindah ke Configured ketika *Processor* telah terhubung ke *DataSource*, dan data telah memiliki format yang telah ditentukan .

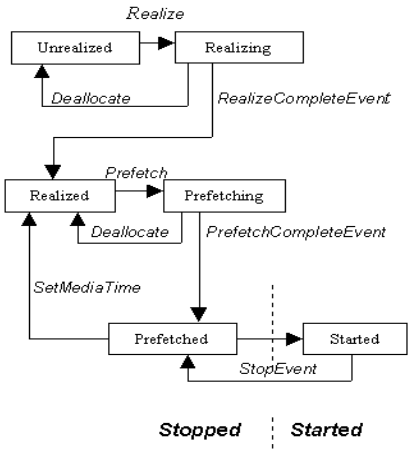
- DataSink
  DataSink adalah *interface* dasar untuk objek yang membaca isi media yang dikirimkan oleh suatu *Data Source* dan mengirimkan media tersebut ke beberapa tujuan.

- Format
  Format merupakan *class* yang akan menempatkan suatu objek ke suatu format media yang tepat.

- Manager
  Manager adalah *interface* yang berfungsi sebagai penghubung objek dan mengintegrasikan implementasi *interface* yang digunakan dengan kelas-kelas yang ada. Misalnya dengan *Manager* dapat dibuat *Player* dari *DataSource*. Dalam JMF terdapat empat manager yaitu:
  - Manager : digunakan untuk membuat Players, Processors, Data Sources, dan Data Sinks.
  - Package Manager : digunakan untuk memelihara paket registry yang berisi class-class JMF seperti Players, Processors, Data Sources, dan Data Sinks
  - Capture Device Manager : digunakan untuk memelihara registry yang disediakan oleh Capture Device
  - PlugIn Manager : digunakan untuk memelihara *registry* yang disediakan JMF untuk memproses komponen *plug-in*.

2.2.5. JMF Player State

Ketika metode *Player* dijalankan menuju ke *Realize()*, *Player* pindah dari state *Unrealized* menuju ke state *Realizing* kemudian *Realizing Player* memproses kebutuhan yang diperlukan agar *Player* bisa dijalankan secara otomatis ke *state Realized*. Sesudah melewati *state Realized Player* menuju ke *Prefetch()*, *Player* diproses melewati *state Prefetching*, setelah selesai melalui *state Prefetching, Player* secara otomatis bisa diproses dan melewati *state Prefetched*. Setelah itu *Player state Prefetched* menuju ke *Start()* kemudian siap untuk dijalankan melewati *state Started*. Setelah media *Player* dijalankan melalui *state Started*, media data atau file yang sudah tersedia dikembalikan ke *state Prefetched* dan bisa digunakan lagi ketika media *Player* akan dijalankan [12].

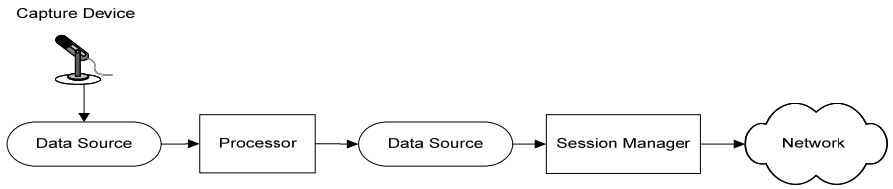


Gambar 2.2.5 JMF Player State

2.2.6. Blok Diagram RTP/JMF

1. RTP Transmitter

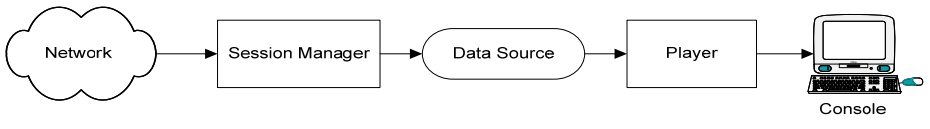
Gambar 2.2.6.1 menjabarkan tentang bagaimana *Data Source* memproses file yang diperoleh dari *Capture Device* (mikropon) kemudian dimasukkan ke *Processor* untuk di proses menjadi *Data Source*, setelah itu dikirim ke *Session Manager* dan ditransmisikan ke jaringan [13].



Gambar 2.2.6.1 Blok Diagram RTP Transmitter

2. RTP Receiver

Setelah *Transmitter* mengirimkan file *streaming* secara *real time* dalam jaringan, file diterima dan dimasukkan pada *Session Manager* dan diteruskan ke *Data Source* diproses kembali lalu dikirim ke *Player* untuk diolah untuk menghasilkan *state* yang akan ditampilkan ke *Console* secara berkala berdasarkan pengiriman file *streaming* secara *real time* yang diperoleh dari Transmitter melewati jaringan [13].



Gambar 2.2.6.2 Blok Diagram RTP Receiver

3. METODE PENELITIAN

4.1. Diagram Use Case

Use Case adalah teknik untuk merekam persyaratan fungsional sebuah sistem, *Use Case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan.

> *User masuk ke aplikasi, aplikasi kemudian berjalan dan user pada bagian Transmitter melakukan pengiriman pesan informasi suara. User pada bagian receiver menerima kiriman informasi suara, kemudian melakukan pengiriman informasi suara pada user bagian tansmtter, selanjutnya aplikasi akan mengirimkan informasi suara dari user receiver dan diterima kembali oleh user transmitter.*

4.2. Diagram Activity

Diagram activity merupakan teknik untuk menggambarkan *logika prosedural*, *proses bisnis*, dan *jalur kerja*, dalam beberapa hal diagram activity memainkan peran mirip sebuah diagram alir, tetapi perbedaan yang paling mendasar antara diagram activity dan diagram alir adalah diagram ini mendukung behavior paralel.

Diagram activity memungkinkan siapapun yang melakukan proses untuk memilih urutan dalam melakukannya, dengan kata lain diagram hanya menyebutkan aturan-aturan rangkaian dasar yang harus diikuti. Hal ini penting untuk pemodelan secara bisnis karena proses-proses sering muncul secara paralel, dan berguna juga bagi pemakain algoritma yang bersamaan, dimana urutan-urutan independen dapat melakukan hal-hal secara paralel. Node pada sebuah diagram activity disebut sebagai action bukan activity, mengapa disebut demikian, karena activity mengacu pada serangkaian action sehingga diagram ini menampilkan sebuah activity yang tersusun dari action. Diagram activity juga mendeskripsikan aksi-aksi dan hasilnya berupa operasi-operasi dan aktifitas-aktifitas di *use case* [15] [16].

4.3. Diagram Class

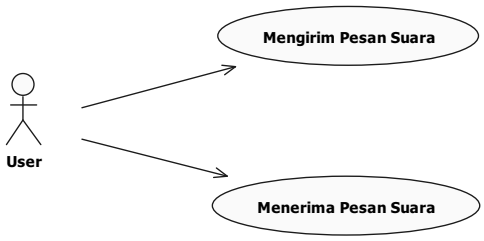
Diagram class mendeskripsikan objek-objek yang terlibat dalam sistem dan hubungan-hubungan diantara mereka kemudian menguraikan objek-objek yang saling berelasi secara statis yang ada didalam sistem dan berbagi relasi statis yang ada diantaranya. Diagram *class* juga menunjukan properti, operasi sebuah *class* dan batasan-batasan yang terdapat dalam hubungan objek. Diagram class tidak hanya digunakan secara luas tetapi juga memiliki banyak konsep pemodelan.

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Diagram *Class* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [16].

4. Hasil Implementasi dan Pembahasan

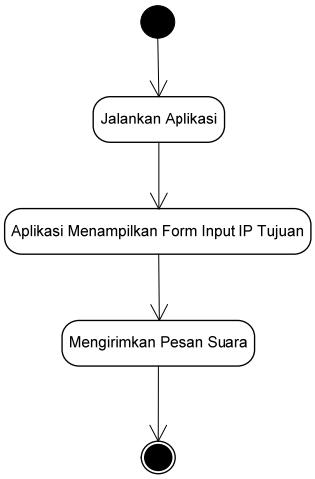
4.1. Hasil Perancangan

4.4.1. Diagram Use Case

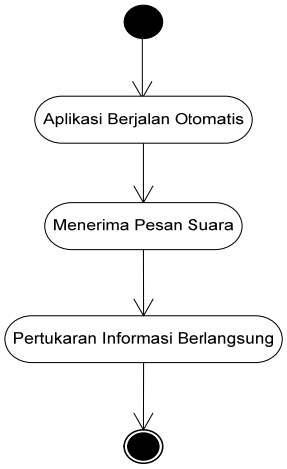


Gambar 4.1.1 Diagram Use Case

4.4.2. Diagram Activity

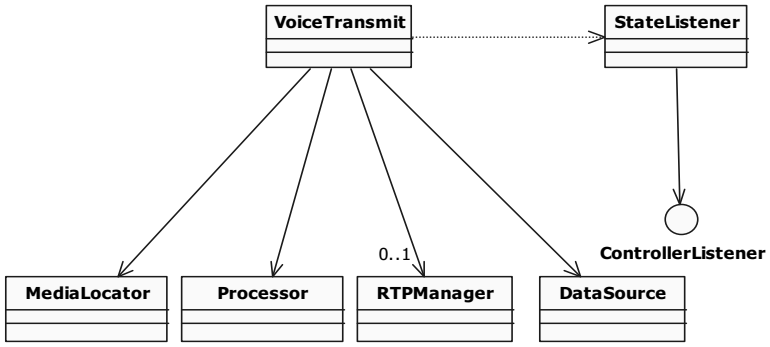


Gambar 4.2.1a Diagram Activity Transmitter

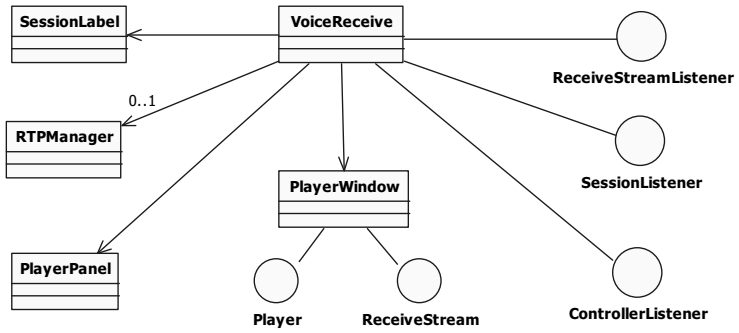


Gambar 4.2.1b Diagram Activity Receiver

4.4.3. Diagram Class



Gambar 4.1.3a Diagram Class VoiceTransmit



Gambar 4.1.3b Diagram Class VoiceReceive

4.2. Hasil Implementasi

4.4.1. Spesifikasi Sistem

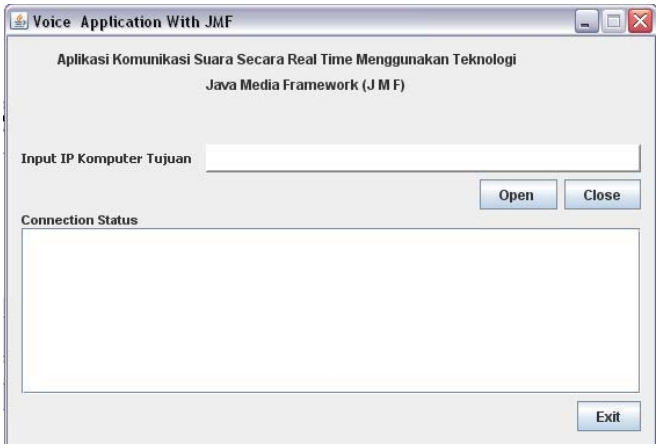
1. Perangkat Keras/Hardware

Satu unit komputer *desktop* dengan processor AMD AthlonXP 2400+, Memory DDR 512Mb, Harddisk dengan freespace sekitar 1Gb, VGA 128Mb, Monitor, CDRom 52X, LAN Card Onboard, Satu unit Headset, dan kabel LAN jenis UTP panjang sekitar 5 meter, Satu unit Laptop IBM ThinkPad T43 dengan processor Intel Centrino Mobile 1,86GHz, Memory DDR 512Mb, Harddisk dengan freespace sekitar 1Gb, VGA Ati Radeon Mobile X300, DVDRom/CDRW, LAN, dan speaker sudah disediakn di Laptop.

2. Perangkat Lunak/Software

Sistim Operasi Microsoft Windows XP Professional Service Pack 2, Java 2 SDK 1.6, Netbeans 5.5, JMStudio 2.1.1.

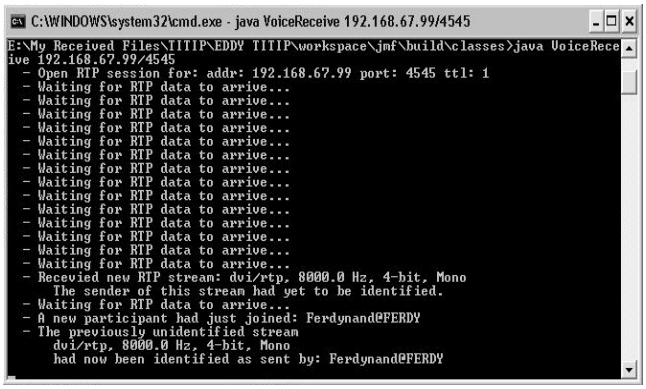
4.4.2. Antar Muka/User Interface



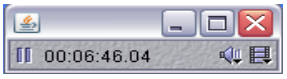
Gambar 4.2.2a Aplikasi Bagian Transmitter

Gambar 4.2.2b Koneksi terbentuk

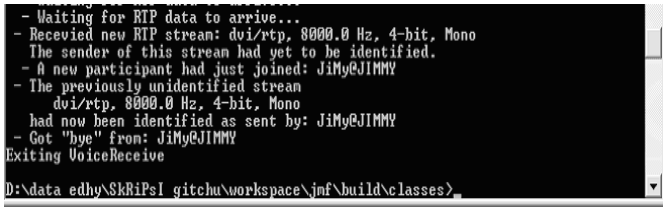
Gambar 4.2.2c Koneksi berakhir



Gambar 4.2.2d Command Line Komputer Receiver



Gambar 4.2.2e Library Player JMStudio



Gambar 4.2.2f Koneksi Berakhir

4.3. Pembahasan

Dalam implementasi ini aplikasi dapat dijalankan tanpa ada *error* dengan asumsi kedua komputer sudah terhubung dalam satu jaringan *peer to peer* dan pengamatan *IP* tidak mengalami kesalahan, karena aplikasi mengasumsikan bahwa *IP* yang dimasukkan adalah *IP* dari komputer yang akan digunakan sebagai *receiver* dan *port* yang digunakan untuk dilewati oleh file-file *streaming* sementara tidak digunakan atau tidak ada aplikasi lain yang menggunakan *port* tersebut. Sehingga ketika user menekan tombol ***Open*** aplikasi langsung mengidentifikasi dan membuka *track* yang digunakan sebagai *transmitter* melalui *DVI/RTP* dengan format audio dengan frekwensi 8000.0 Hertz, 4 Bit Mono dan membuat *RTP* Session pada *IP* yang dituju dan *port* yang akan digunakan. Setelah proses diatas sudah dilakukan maka aplikasi telah siap untuk menerima dan menjalankan data/file yang diperoleh dari *Capture Device* dan akan di transmisikan secara *real time*.

Aplikasi pada bagian *Receiver* dijalankan pada command line. User mengetik nama class java yang akan dipanggil, kemudian menentukan *IP* dan *port* yang akan digunakan untuk berkomunikasi (*IP* dan *port* yang dimasukkan adalah *IP* komputer *transmitter* dan *port* yang sudah ditentukan). Setelah itu aplikasi mulai menginisialisasi *IP* dan *port* yang sudah dimasukkan kemudian aplikasi menunggu data yang dikirim dari bagian transmitter. Setelah data yang dikirim

telah melewati *IP* dan *port* yang sudah disediakan aplikasi mulai memproses data/file yang ada kemudian ditampilkan dan library player JMStudio menampilkan waktu berapa lama kedua komputer terhubung untuk melakukan komunikasi secara *real time*.

Pengujian dilakukan dengan cara kedua komputer sudah terhubung dengan baik dengan pengaturan IP dan workgroup dalam jaringan secara peer to peer, kemudian aplikasi dijalankan, *user* mengisi Input IP Komputer Tujuan dan menekan tombol *Open*, pada connection status aplikasi menyediakan setelah itu aplikasi berjalan dan kedua user bisa saling berkomunikasi.

Dari hasil perancangan aplikasi dan analisa sistem diperoleh beberapa hasil implementasi yang sesuai dengan tujuan yang disampaikan pada bab-bab sebelumnya, antara lain:

1. Aplikasi dapat berjalan dengan baik tanpa ada pesan kesalaan atau error dari masing-masing user (bagian *transmitter* dan bagian *receiver*).
2. File suara atau voice dapat ditransmisikan oleh user bagian transmitter dan diterima oleh user bagian receiver secara real time tanpa ada cacat komunikasi (suara putus-putus dan data/file semua dapat diterima oleh user *receiver*)

5. Kesimpulan dan Saran

Dari hasil perancangan dan implementasi ditarik beberapa kesimpulan antara lain:

1. pembuatan aplikasi komunikasi suara secara *real time* menggunakan teknologi *JMF* mempermudah *user* memperoleh dan bertukar informasi dengan *user* yang lain tanpa harus bertemu secara langsung, tetapi user cukup hanya terhubung dalam satu jaringan (dalam hal ini *peer to peer*),
2. Dari Use Case diagram diketahui bahwa user dapat melakukan pengiriman dan menerima pesan suara dari aplikasi.
3. Dari Activity diagram, user bagian *transmitter* bisa menjalankan aplikasi kemudian memasukkan almat IP komputer yang dituju kemudian melakukan pengiriman pesan suara. Dan user bagian *receiver* menjalankan aplikasi kemudian aplikasi menampilkan pesan suara dari bagian transmitter dan pertukaran informasi berlangsung,
4. Aplikasi komunikasi suara secara real time menggunakan teknologi JMF dibuat berdasarkan dua *Class* diagram yaitu: *Class VoiceTransmit* dan *VoiceReceive*

Saran untuk pengembangan aplikasi lebih lanjut adalah:

1. Aplikasi menentukan langkah-langkah apa saja yang harus dijalankan oleh user dan menampilkan pesan kesalahan atau error jika ada ketentuan yang belum dipenuhi oleh user(akan ada pesan kesalahan),
2. Aplikasi nantinya bisa menampilkan file audio dan video secara real time sehingga user bisa langsung bertatap muka,
3. Dimungkinkannya terjadi komunikasi lebih dari dua user agar informasi yang diperoleh bisa lebih luas dan cepat (*conference*)

6. Daftar Pustaka

[Ano08] Anonim
URL:http://www.en.wikipedia.org/wiki/java_media_framework.html

[Sig06] Sigit Priyanggoro, 2006, "M*embuat Player dengan Java Media framework 2.1*", Ilmu Komputer, http//:ilmukomputer.com

[Ral08] Ralf Ackerman, "*Suitability of Java for Virtual Collaboration*", Dept. Information Technology, Darmstadt University, Germany

[Agu05] Agung B.P, dkk, Juni 2005 "*Aplikasi Web Cam dengan Java Media Framework*", *Transmisi*, Vol. 9, No. 1, http//:ilmukomputer.com

[Bel03] P. Bellavista, A. Corradi, C. Stefanelli, Nov.-Dec. 2003, pp. 16-24, "*The ubiQoS Middleware for Audio Streaming to Bluetooth Devices*", *IEEE Internet Computing*, Vol. 7, No. 6, http//:deis.unibo.it

[Ste04] Steven Morris, 2004, "*The Java Media Framework*", *Tutorials*, http://www.interactivetvweb.org/tutorial/index.shtml

[Ano08] Anonim, URL:http://www.ibm.com/developerworks/ javamediaframework basics.html

[Col03] Collin Perkins, Juni 2003, *"RTP: audio and Video for the internet",* Addison Wesley, USA

[Sch02] Schulzrinne, Casner, juni 2002, "*RTP: A Transport Protocol for Real Time Application*", RFC 1889.

[Sun99] Sun Microsystems, Inc. 19 November 1999, "*Java Media Framework API Guide*", Sun Microsystems, California, http://sun.java.com/jmf-guide/pdf.

[Bud01] Budi Kurniawan, April 2001, "*Programming Multimedia With JMF*", http://www.javaworld.com/javaworld/jw-04-2001.html.

[Chn00] Chengyuan Phen, 2000, "*Development of Java User Interface for Digital Television*", Helsinky Univesity, Finland, http://www.hut.fi

[Sun00] Sun Microsystems, Inc. 2000 "*JMF 2.1.1 Specification*", http:// java.sun.com/product/java-media/jmf/2.1.1/doc.html.

[Mar04] Martin fowler, 2004, "*UML Distilled*", Andi Offset, Yogyakarta

[Gra03] Grady Booch, Jmaes Rumbaugh, Ivar Jacobson, October 2003, "*The Unified Modeling Language User Guide*", Addison Wesley,USA

[Dha00] Dharwiyanti, 2000, "*Pengantar Unfied Modeling Language*", Ilmukomputer, http//ilmukomputer.com.