APLIKASI WEBCAM DENGAN JAVA MEDIA FRAMEWORK

Agung Budi Prasetijo, Aghus Sofwan, Hery Oktafiandi
Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro
Jl. Prof. Sudharto – Tembalang, Semarang
Email : agungbp@ft.undip.ac.id

Abstrak - *Perkembangan Internet akhir-akhir ini ini telah membuat Internet menjadi begitu besar perannya baik sebagai sarana memperoleh informasi dengan cepat dan selalu diperbaharui. Setelah pertukaran informasi dilakukan dengan pengiriman dan penerimaan electronic mail maka pada saai ini mulai bergerak ke arah pengiriman informasi yang memungkinkan dapat terselenggaranya sebuah komunikasi berbasis multimedia.*

Java Media Framework merupakan interface aplikasi multimedia. Salah satu bentuk multimedia yang dapat dioperasikan dengan JMF adalah webcam. Dengan menggunakan RTP (Real Time Protocol), hasil pemrograman Java Media Framework dapat diterapkan dalam jaringan dengan transfer media data yang diperoleh dari webcam.

Program aplikasi webcam dengan Java Media Framework yang dibuat pada Makalah ini dilengkapi dengan RTCP(Real Time Control Protocol) sehingga dapat diperoleh waktu tunda untuk satu pengiriman SR (Sender Report) yang dalam pengujiannya dilakukan dengan menggunakan dua tipe webcam berbeda. Dari dua webcam tersebut diperoleh kesimpulan bahwa webcam dengan tingkat resolusi tinggi akan diperoleh waktu tunda yang lebih besar dibandingkan dengan webcam dengan resolusi yang rendah.

Kata kunci: Java Media Framework, Real Time Protocol, Real Time Control Protocol

I. PENDAHULUAN

A. Latar Belakang

Perkembangan Internet akhir-akhir ini ini telah membuat Internet menjadi begitu besar peranannya baik sebagai sarana memperoleh informasi dengan cepat dan selalu diperbaharui. Setelah pertukaran informasi dilakukan dengan pengiriman dan penerimaan elektronik *mail* maka pada saat ini mulai bergerak ke arah pengiriman informasi yang memungkinkan dapat terselenggaranya sebuah komunikasi berbasis multimedia.

Beberapa sistem operasi yang ada seperti Microsoft dan UNIX telah menyediakan perangkat lunak sebagai pendukungnya. Untuk versi Windows 98 SE, Windows ME, Windows XP, dan Windows 2000 biasanya telah tersedia perangkat lunak NetMeeting 3.01. Sedangkan bagi pengguna Linux dapat menggunakan GnomeMeeting.

Java Media Framework merupakan *interface* aplikasi multimedia yang dapat bekerja baik pada Microsoft windows maupun UNIX sehingga dalam penerapannya tidak memerlukan penyesuaian yang begitu banyak untuk dapat dioperasikan pada *platform-platform* tersebut. Salah satu bentuk multimedia yang dapat dioperasikan dengan JMF adalah *webcam.*

B. Tujuan

Tujuan dari Penelitian ini adalah membuat program aplikasi untuk menangkap gambar dan menyimpannya dalam format file jpeg, dan juga membuat program RTP (*Real Time Protocol*) untuk dihitung waktu tunda dengan RTCP pada jaringan lokal dengan menggunakan Java Media Framework.

C. Batasan Masalah

1. *Capture device* yang digunakan adalah *webcam* produksi logitech dengan tiga versi yang berbeda dengan sistem operasi Windows.
2. Proses menangkap gambar atau objek dan menyimpannya dalam file *ekstensi* jpeg.
3. Proses pengiriman media data dari RTP *server* dan diterima oleh RTP *client* untuk ditampilkan hasil dari *capture* objek *webcam* pada *client*. Format video disesuaikan dengan sistem operasi windows yang digunakan dengan format JPEG. Format dan algoritma encoder video sendiri tidak dibahas.
4. Hubungan antar *participant* hanya terjadi antara *server* dengan satu tujuan *client*. *Client* akan menerima media data berupa video *streaming* dari *server*.

II. Java Media Framework dan Protokol Data Streaming

2.1.1 Java Media Framework (JMF)

JMF API merupakan arsitektur yang menggabungkan protokol dan pemrograman *interface* untuk merekam, mentransmisi, dan *playback* media. Pada JMF versi 2.1.1, Sun's sebagai perusahaan pengembang bahasa pemrograman java berinisiatif untuk membawa pemrosesan time-base media kedalam bahasa pemrograman Java. Time-base media adalah mengubah data yang diterima dengan berdasarkan waktu, termasuk didalamnya seperti audio dan video klip, MIDI, dan animasi[1,2,3,4].

Beberapa dari fungsi JMF, yaitu :

a. Dapat digunakan untuk berbagai file multimedia pada Java Applet atau aplikasi. Format yang mendukung antara lain AU, AVI, MIDI, MPEG, QUICKTIME dan WAV.

b. *Play* media *streaming* dari internet
c. *Capture* audio dan video dengan mikropon dan kamera video kemudian menyimpan data tersebut kedalam format yang mendukungnya.
d. Mengirimkan audio dan video secara *realtime* ke dalam jaringan internet atau intranet.
e. Dapat digunakan untuk pemrograman penyiaran radio atau televisi secara langsung.

2.1.2 Dasar Pemrograman JMF

Dikembangkan atas kerjasama Sun Microsystem dan IBM Haifa pada tahun 1998 dengan keunggulan dapat digunakan sebagai Media *capture* dan pemrograman untuk media *playback*. JMF 2.1.1 API merupakan rilis terbaru dari Sun's.
Interface dan *Class* untuk koneksi *high-level* API dalam pemrograman JMF adalah sebagai berikut:

a. DataSource

`DataSource` berfungsi untuk mengatur transfer dari media. Pada JMF DataSource diidentifikasi dengan `MediaLocator`.

b. Capture Device

Capture device merupakan perangkat keras yang digunakan untuk memperoleh data, seperti mikropon, kamera biasa, atau video kamera. *Capture* media data akan menjadi input untuk *Player* agar dapat ditampilkan.

c. 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.

Terdapat enam tahapan pada JMF *Player*:

a. `Unrealized` : pada tahapan ini *Player* belum mengenali media yang akan digunakan.
b. `Realizing` : pada tahapan ini *Player* akan menentukan materi atau media yang akan dipakai.
c. `Realized` : pada tahapan ini *Player* mengetahui materi atau media yang digunakan dan memiliki informasi tentang tipe media yang akan ditampilkan.
d. `Prefetching` : *Player* akan mempersiapkan untuk menampilkan media. Selama tahapan ini, *Player* akan *preload* media data untuk memperoleh *resource* yang digunakan dan apa saja yang dibutuhkan untuk mulai memainkan media data.
e. `Prefetched` : *Player* telah selesai `prefetching` media data, dan siap untuk start *Player*.
f. `Started` : Langkah ini merupakan hasil ketika memanggil `start()`. *Player* bisa untuk menampilkan media data sekarang.

d. 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`.

a. `Configuring` : *Processor* masuk ke tahapan `configuring` dari `Unrealized` ketika metode `configure ()` dipanggil. *Processor* berada di `Configuring` setelah terhubung ke *DataSource*.
b. `Configured` : setelah `Configuring`, *Processor* pindah ke `Configured` ketika *Processor* telah terhubung ke *DataSource*, dan data telah memiliki format yang telah ditentukan .

e. DataSink

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

f. Format

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

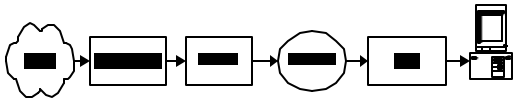
g. Manager

Manager adalah *interface* yang berfungsi sebagai penghubung objek, mengintegrasikan implementasi *interface* yang digunakan dengan kelas-kelas yang ada. Misalnya dengan *Manager* dapat dibuat *Player* dari *DataSource*.

2.2.1 RTP (*Real Time Protocol*)

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/IP terdapat dua protokol transport, yaitu: *Transmission Control Protocol* (TCP) dan *User Datagram Protocol* (UDP). 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 kecepatan transfer data data *server* ke *client* yang lebih tinggi hingga mendukung unutk aplikasi data *real-time*[1,6,7].

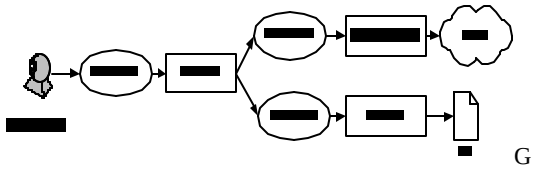
JMF dapat mentransmisikan dan *playback* RTP *stream* dengan API yang terdapat pada `javax.media.rtp`, `javax.media.rtp.event`, dan `javax.media.rtp.rtcp`. Pada RTP *receiver*/*client*, dapat dilakukan *playback* atau menerima media data yang dikirimkan oleh RTP *transmitter*/*server* (lihat Gambar 1)



Gambar 1 RTP Receiver

Pada Gambar 1 terlihat tahapan diagram alur proses RTP pada bagian *receiver* yang menerima media data dari jaringan dan kemudian mempresentasikannya dengan *player*.

Proses transmisi media data melalui tahapan-tahapan yang memiliki fungsi sendiri. RTP pengirim data memiliki tahapan yang dapat dilihat pada Gambar 2. Pada RTP transmisi, input yang diperoleh *data source* berupa *capture device webcam*. Kemudian *Processor* akan memprosesnya menjadi *data source* yang digunakan untuk RTP atau DataSink.



Gambar 2 RTP Transmisi

2.2.2 RTCP (*Real-Time Control Protocol*)

RTCP paket akan mengirimkan secara berkala kepada participants yang teridentifikasi mengenai data yang dikirim dan yang diterima. Beberapa tipe paket RTCP tersebut adalah sebagai berikut:

a. 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.

b. Receiver Report

Sama seperti halnya Sender Report, hanya saja dilakukan pada sisi penerima.

c. Source Description

Source description akan menunjukkan identitas pariticipants. Pada SDES, *Cononical Name* merupakan item yang harus dikirimkan pada setiap paket RTCP karena mengandung informasi identitas serta lokasi sumbernya, biasanya dengan format user@full hostname.

d. Goodbye (Bye)

Paket Bye merupakan paket yang akan memberikan informasi participant yang tidak aktif atau yang akan meninggalkan session. Ketika participants meninggalkan session maka paket bye akan terkirim.

III. PERANCANGAN

Perancangan program ada tiga tahapan, yaitu:

a. Program Meyimpan capture objek dalam file jpeg

Tahapan-tahapan yang dilakukan dalam pembuatan program untuk menyimpan hasil *capture* objek dalam file dengan format jpeg dimulai dengan mengetahui source yang digunakan dengan CaptureDeviceManager, CaptureDeviceInfo, memulai proses media data dengan Player, kemudian *capture* objek, dan menyimpan dalam bentuk file dengan format jpeg. Gambar 3 menunjukkan *flowchart* program menyimpan objek yang ditangkap dalam format jpeg.

b. RTP Receiver

Tahapan-tahapan yang dilakukan dalam pembuatan *file* RTPReceiver adalah dimulai dengan mengidentifikasikan *port* yang dapat diakses oleh *client*, membuka koneksi dengan *server*, update event dari objek *source*, *start receiver* data dengan memulai RTP session, menerima stream dengan getRecvStream, mengenali datasource, membuat player dengan PlayerWindow, menampilkan media data yang diterima, menutup RTP session setelah *server* menutup stream dan memutuskan koneksi dengan *client*. Gambar 4 menunjukkan *flowchart* program RTP Receiver.

c. RTP Transmiter

Tahapan-tahapan yang dilakukan dalam pembuatan RTP *transmiter* adalah dimulai dengan mengidentifikasikan *port* yang dapat diakses oleh *client* RTPManager, mengidentifikasi *source*, mengidentifikasi medialocator, proses pengolahan data media oleh processor, mengatur tracks yang akan dipakai pada processor, mengirim data media ke *client*, menutup *stream-stream* dan menutup koneksi dengan *client*. Gambar 5 menunjukkan *flowchart* RTP Transmitter.

IV. HASIL DAN PEMBAHASAN

4.1 Menyimpan capture objek dalam file JPEG

Pada program *capture* objek dalam file jpeg, dalam perancangannya perlu diketahui dulu format video untuk Windows yang akan digunakan. Format ini perlu diketahui sebagai inisialisasi dari *data source* yang pada percobaan ini digunakan *capture device* berupa *webcam*. Dari percobaan yang dilakukan digunakan dua buah *webcam* produksi Logitech yang masing-masing dicoba pada Windows 98ND dan Windows XP. Inisialisasi *webcam* dapat dilihat pada Tabel 1.

Tabel 1 Video Format Windows

OS / Device	WIN 98 SE		WIN XP	
	Video Device Init		Video Device Init	
	Logitech USB	Mic WDM	Logitech USB	Mic WDM
Logitech QC Messenger	-	<0>	-	<0>
Logitech QC Pro 4000	-	<0>	-	Win32<0>
Logitech QC Pro 3000	<1>	<0>	<1>	Win32<0>

Pada Tabel 1 terlihat bahwa masing-masing *webcam* memiliki inisialisasi yang berbeda satu sama lain. Dalam percobaan yang dilakukan hanya menggunakan *webcam* QC Pro 4000 dan QC Messenger saja. Pada saat inisialiasasi *webcam*,

JMF hanya bisa melakukan inisialisasi satu *webcam* saja pada satu komputer.

4.1.1 Capture dan Menyimpan File JPEG

Untuk *capture* objek yang tampil pada layar monitor dari Player Windows digunakan FrameGrabbingControl. FrameGrabbingControl ini pada JMF diperoleh dari *interface* pada `javax.media.control.FrameGrabbingControl` yang kemudian akan mengetahui format video yang pada percobaan ini digunakan format video JPEG yang diperoleh dari *interface* `import com.sun.image.codec.jpeg.*`. Setelah objek yang sudah di *capture* maka dengan `Imgpanel.setImage` akan dicetak ulang untuk dikonversi ke format JPEG dengan dengan *interface* yang sama dengan format video sebelumnya.

Objek kemudian akan diproses menjadi gambar pola grapik 2D dengan ukuran frame 320x240 dengan file test.jpeg. Penyimpanan objek pada kesempatan pertama hanya dengan nama test.jpeg, apabila dilakukan *capture* objek pada kesempatan berikutnya maka objek terakhir yang akan tersimpan dengan file test.jpeg sedangkan yang sebelumnya akan terhapus.

Dari Gambar 6 hasil yang diperoleh dengan menggunakan webcam QC Pro 4000 dan webcam QC Messenger.

Gambar 6 File Test.JPEG

Dari hasil program *capture* objek dan disimpan pada file jpeg diperoleh hasil ukuran file dari *webcam* QC Messenger lebih kecil dari *capture* dari QC Pro 4000. Hal ini dimungkinkan karena kualitas gambar yang diperoleh dari QC Pro 4000 lebih baik dari QC Messenger.

4.2 Pengujian Program RTP

Program RTP dibagi menjadi dua bagian, yaitu: RTP client dan RTP server. Masing-masing program RTP tersebut dibuat dengan tampilan GUI untuk mempermudah dalam penggunaannya.

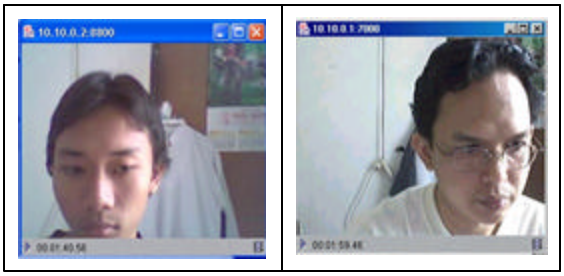
4.2.1 RTP *Client*

Program RTP *Client* terdapat pada `package multimedia.receiver`. RTP *client* akan menerima media data yang dikirimkan oleh RTP *server* dan menampilkan data yang diterima dengan player Windows. Program RTP Client terdiri atas empat file *.java, yaitu:

a. Rx.java
b. AVReceiver.java
c. Config.java
d. Target.java
e. RTCPViewer.java

Pada saat koneksi terjadi antara *client* dan *server* akan diterima informasi berupa format video yang dikirim dan *participant* yang mengirimkan media data.

Dengan adamya tampilan pada *receiver* tempat informasi IP *address* dan *port* yang harus diisi. Setelah diketahui IP address lokal dan IP address *server*, pada GUI JMF/RTP *Receiver* diisi dengan IP address masing-masing, dengan disertai port yang akan dipakai sebagai jalur komunikasi.



(a) (b)

Gambar 7 Tampilan Video *Client*, (a) QC Pro 4000, (b) QC Messenger

4.2.2 RTP Server

RTP *server* juga dilengkapi dengan GUI, yang akan mempermudah dalam penggunaannya. Program RTP *server* terdapat lima file *.java, yaitu:

a. Tx.java.
b. AVTransmitter.java.
c. Config.java.
d. Target.java.
e. RTCPViewer.java.

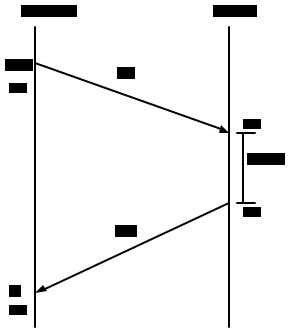
Pada tahap awal akan diisi IP *address* dan *port* yang akan digunakan pada *form* JMF/RTP *Transmitter*. Setelah IP *address* dan data *port* telah diisi pada tahap berikutnya mengisi data pada *Source* sesuai dengan Media *Locator* yang digunakan. Media yang digunakan sesuai dengan hasil inisialisasi webcam yang dipakai. Media Locator diisi dengan vfw://0.

4.2.3 RTCP

Selama proses pengiriman dan penerimaan media data, pada RTP *Server* dan RTP *Client* dilengkapi dengan RTCP (*Real Time Control Protocol*) yang akan memberikan informasi SR dan RR.

Dari tampilan pada RTCP Monitor terdapat tiga informasi yang dapat diterima pada *server* dan *client* selama proses pengiriman dan penerimaan data dilakukan. Masing-masing adalah: waktu pada saat data dikirim dan diterima, CNAME, dan SSRC.

Waktu yang tertera pada tampilan menggunakan format jam:menit:detik.



Gambar 8 Diagram Waktu SR dan RR

Dari Gambar 4.6 dapat dihitung *Round Trip Delay* (RTD) dengan rumus:

$$RTD = A - LSR - DLSR$$

dimana:

$$DLSR = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$

Hari dan waktu dihitung menggunakan format *timestamps* dari NTP (*Network Time Protocol*) dengan satuan detik dimulai dari 0h UTC (*Universal Time Coordinated*) pada 1 Januari 1900[5]. Gambar 4.6 menunjukkan diagram waktu SR dan RR.

Pada Gambar 8 menunjukkan diagram waktu SR dan RR. Pengujian dilakukan pada tanggal 14 Februari 2004 dengan waktu sesuai dengan tampilan pada *server* dan *client* Tabel 2 menunjukkan hasil konversi waktu dari UTC ke NTP.

Perhitungan SR(8) secara lengkap adalah sebagai berikut:

Tabel 2 Delay SR(8)

[14 Februari 2004 22:15:08]	**[14 Februari 2004 22:15:21]**
SR(8)	NTP = $C3D91C79_{(16)}$ A = $1C79_{(16)}$ (7289 detik)
NTP sec = $C3D91C6C_{(16)}$ NTP frac = 00000000 (3,385,785,708 detik)	dlsr = (6.5 detik) lsr = 1C6C (7276 detik)

Maka diperoleh waktu tunda untuk SR(8) sebesar 6,5 detik.

Dari hasil perhitungan waktu tunda pada saat pengujian tanggal 14 Februari 2004 dengan format NTP terlihat pada SR ke-8 waktu tunda menjadi semakin besar yaitu: 6,5 detik Pada Tabel 3 hasil percobaan 1 menunjukkan hasil perhitungan waktu tunda untuk pengujian program RTP *server* dengan menggunakan *webcam* QC Pro 4000.

Dari tabel hasil pengujian 1 terlihat bahwa pada saat SR(12) waktu tunda mencapai 9.5 detik yang dalam hal ini merupakan waktu tunda tertinggi dalam percobaan ini. Waktu tunda yang semakin besar semakin menurunkan kualitas gambar yang diterima pada *client*.

Pada percobaan 2 menggunakan webcam QC Messenger dengan *server* Win 98SE dan *client* menggunakan Win XP. Percobaan 2 dilakukan pada tanggal 17 Februari 2004. Tabel 4 merupakan hasil dari percobaan 2.

Dari hasil percobaan 2 yang menggunakan webcam QC Messenger diperoleh waktu tunda yang relatif lebih kecil dibandingkan hasil percobaan 1. Hasil percobaan juga menunjukkan waktu tunda diperoleh lebih stabil sehingga kualitas gambar yang ditampilkan pada *client* lebih baik.

V. PENUTUP

5.1 Kesimpulan

1. *Capture device* pada sistem operasi Windows memiliki inisialisasi yang berbeda-beda tergantung dengan tipe webcam yang digunakan
2. Inisialisasi *webcam* pada program JMF hanya bisa dilakukan pada satu tipe *webcam* untuk satu komputer saja.
3. Pada pengujian program RTP, pemakaian *webcam* QC Messenger dengan resolusi rendah akan diperoleh waktu tunda yang lebih kecil dan relatif stabil dibandingkan dengan webcam QC Pro 4000.

5.2 Saran

1. Mengkombinasikan pengiriman data audio dan video yang dapat memungkinkan terselenggaranya konferensi video.
2. Mengembangkan format video yang lain untuk digunakan seperti H261 dan H263.
3. Diharapkan adanya percobaan dan penelitian lebih lanjut untuk penggunaan Java Media Framework dengan sistem operasi yang lain khususnya dalam inisialisasi *capture device* yang digunakan.

DAFTAR PUSTAKA

1. Aitenbichler, Erwin., "*JMF, Java Media Framework*", Abt. Telekooperation TU Darmstadt, May 2002.
2. Haneef, Anwar M., "*JMF-Multimedia Networking for The Rest of Us"*, URL: http://www. -unix.ecs.umass.edu/~ahaneef, 2002.
3. Kurniawan, Budi., "*Program Multimedia With JMF, Part 1"*, URL: http://www.javaworld.com/jw-04-2001/jw-0406-jmf1.html, 6 April 2001.
4. Kurniawan, Budi., "*Program Multimedia With JMF, Part 2"*, URL:

http://www.javaworld.com/jw-06-2001/jw-0504-jmf1.html, 4 May 2001.
5. Schulzrinne., Casner., "*RTP: A Transport Protocol for Real Time Applications, Revisi 1.1* ", RFC 1889, 12 Juni 2002.
6. Zhou, S., P King, "*A Simple Platform Independent Video on Demand Application*", Electrical Engineering & Telecomunication The University of New South Wales, Autralia, 2002.
7. …,"*Java Media Framework API Guide*", URL: http://www. java.sun.com/ products /java-media/jmf/2.1/guide/JMFTOC.html, 19 November 1999.