
JMF per applicazioni multimediali in Java

Ing. Generoso Paolillo

Corso di ***Applicazioni Telematiche***

A.A. 2008-09

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II

Facoltà di Ingegneria

Contenuti della lezione

- Organizzazione API JMF
- Nozioni generali su applicazioni multimediali
- Principali componenti di JMF
- Trasmissione e ricezione di flussi RTP
- Esempi

Java Media Framework (JMF)

- Java Media Framework (JMF) è un package opzionale di 2 Platform, Standard Edition (J2SE)
- Permette di utilizzare contenuti multimediali all'interno di applicazioni e applet Java:
 - Cattura
 - Riproduzione
 - Trasmissione
 - Codifica e decodifica nei formati multimediali

Obiettivi di progetto di JMF

JMF è progettato per consentire:

- Accesso diretto ai dati multimediali
- Supporto per la cattura da sorgenti multimediali
- Invio e ricezione di streaming multimediali
- Integrazione di nuovi componenti (JMF plug-ins) con l'infrastruttura esistente:
 - Es: demultiplexers, codecs, effects processors, multiplexers, e renderers.

JMF RTP APIs

I principali package che compongono le API JMF sono:

- ***javax.media***: contiene le principali classi di JMF
- ***javax.media.rtp***: contiene le classi per accedere ai campi del protocollo RTP/RTCP
- ***javax.media.control***: permette di leggere e modificare parametri quali: bit rate, frame rate, lunghezza del buffer di ricezione, e altri parametri del processo di encoding.
- ***javax.media.format***: *per la descrizione dei formati supportati*

Lavorare con i *time-based media*

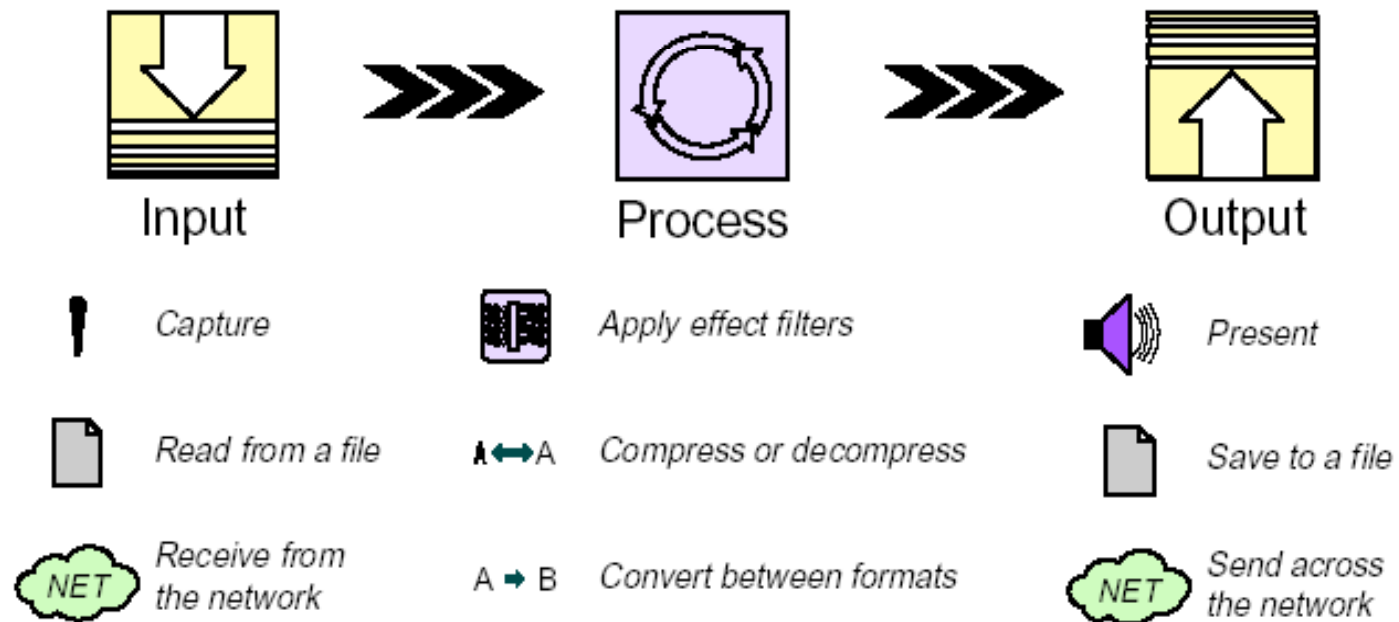
Ogni dato che cambia significativamente nel tempo può essere caratterizzato come time-based media:

es: audio clips, MIDI sequences, movie clips, e animazioni

Tali media possono essere ottenuti da una varietà di sorgenti:

es: file locali o remoti, camere, microfoni e trasmissioni broadcast

Modello di elaborazione di tali dati:



Streaming Media

Content type

Rappresenta il formato nel quale è memorizzato il dato multimediale

Media streams

Rappresenta il dato multimediale ottenuto da un file locale, acquisito dalla rete o catturato da una camera o da un microfono.

- Possono contenere più canali di dati detti tracks
 - detti in tal caso multiplexed o complex media streams
- *Demultiplexing*: processo di estrazione delle singole tracce dal complex media stream.

Identificare un *media stream*

Un media stream può essere identificato dalla sua posizione e dal protocollo usato per accedervi.

Alcuni esempi:

- **URL** (Uniform Resource Locator) rappresenta un puntatore ad una risorsa (multimediale in questo caso) sia essa locale (file:/D:/movie/test.mov) o remota (<http://WebSite/darkcity.mov>).

- **MediaLocator:** offre le stesse funzionalità della classe *URL* ed è specifico per contenuti multimediali

Modalità di consegna dei dati

I media stream possono essere classificati in base alla modalità con la quale i dati sono consegnati:

- Pull: il trasferimento dati è iniziato e controllato dal lato client
 - Es. il protocollo HTTP è di tipo pull.

- Push: Il server inizia il trasferimento dei dati e controlla il flusso
 - Es: il protocollo RTP (Real-time Transfert Protocol) è un protocollo push usato per lo streaming di dati multimediali.

Common Video Formats

La tabella seguente riporta alcune caratteristiche dei più comuni formati multimediali video:

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
Cinepak	AVI QuickTime	Medium	Low	High
MPEG-1	MPEG	High	High	High
H.261	AVI RTP	Low	Medium	Medium
H.263	QuickTime AVI RTP	Medium	Medium	Low
JPEG	QuickTime AVI RTP	High	High	High
Indeo	QuickTime AVI	Medium	Medium	Medium

Common Audio Formats

La tabella seguente riporta alcune caratteristiche dei più comuni formati multimediali audio:

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
PCM	AVI QuickTime WAV	High	Low	High
Mu-Law	AVI QuickTime WAV RTP	Low	Low	High
ADPCM (DVI, IMA4)	AVI QuickTime WAV RTP	Medium	Medium	Medium
MPEG-1	MPEG	High	High	High
MPEG Layer3	MPEG	High	High	Medium
GSM	WAV RTP	Low	Low	Low
G.723.1	WAV RTP	Medium	Medium	Low

Media presentation

- I media stream possono essere destinati verso:
 - dispositivi di output come altoparlanti o monitor
 - file o rete
- In generale, una destinazione di uscita per dati multimediali viene detta *data sink*
- Per media stream provenienti dalla rete, il tempo che precede l'inizio della presentazione viene detto *start latency*

Media presentation (cont.)

➤ **Qualità della presentazione**

➤ La qualità di una presentazione multimediale dipende da diversi fattori quali:

➤ Lo schema di compressione usato

➤ La capacità elaborativa del sistema che effettua la presentazione

➤ La banda disponibile (per media stream provenienti dalla rete)

➤ **Il sistema visivo umano:**

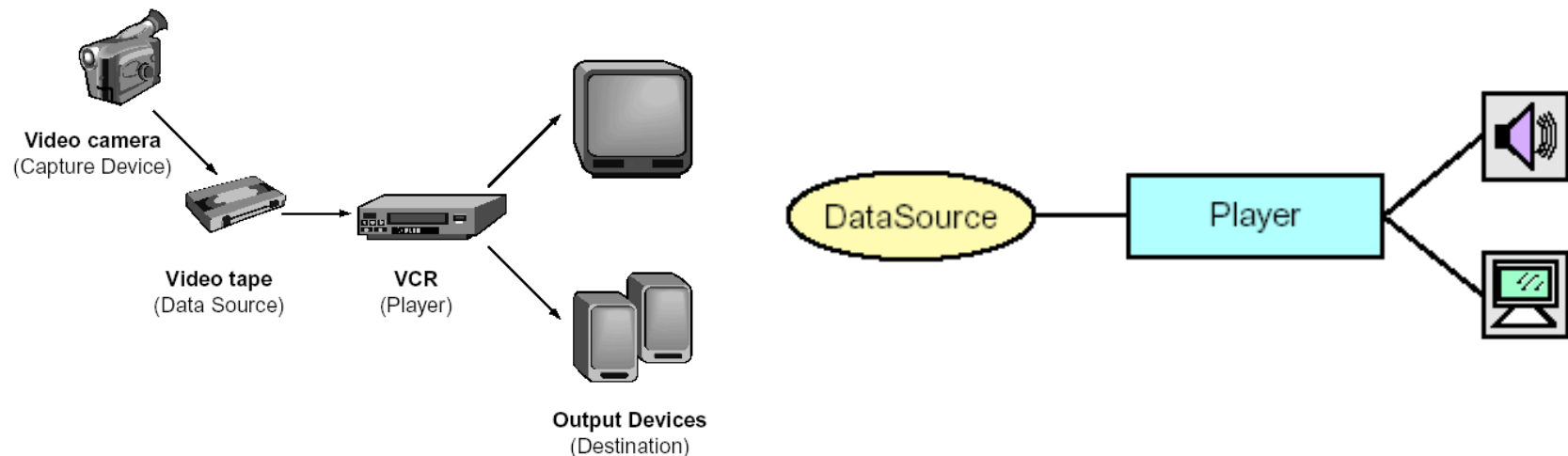
➤ Massimo frame rate distinguibile: 30 fps

Elaborazioni sui dati multimediali

- Alcune elaborazioni vengono effettuate sul media stream prima di essere riprodotto:
 - Se lo stream è multiplexed, le singole tracce sono estratte (operazione di demultiplexing)
 - Se le singole tracce sono compresse (encoded), vengono decodificate (decoding) in un formato non compresso (raw)
 - Se necessario, le tracce sono convertite in un formato differente (encoding)
 - Effetti di filtraggio possono essere applicati alle tracce decodificate

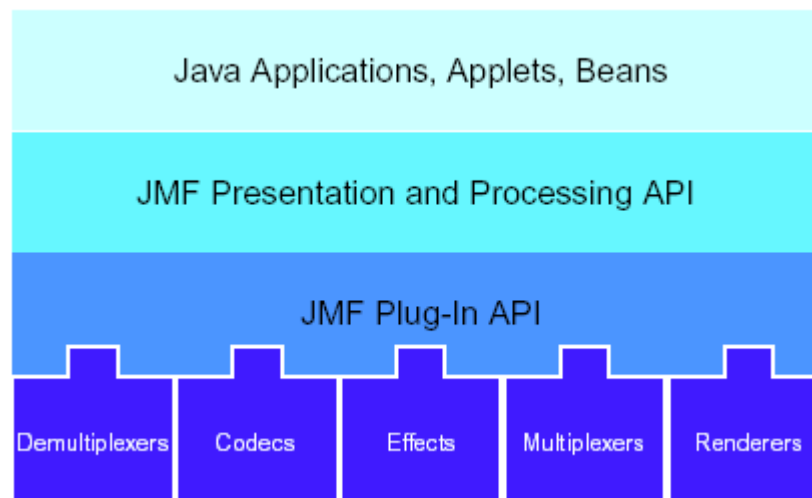
Modello di riferimento

- Dispositivi come un videoregistratore (VCR) offrono un modello familiare per registrare, elaborare, e presentare dei time-based media.
- Riprodurre un filmato con un VCR, equivale a spedire un media stream al VCR inserendo un video tape. Il VCR legge e interpreta i dati della cassetta generando gli opportuni segnali al televisore e agli altoparlanti.
- JMF usa lo stesso modello base. Un *data source* incapsula il media stream così come fa il video tape e un *player* offre le stesse funzionalità del VCR.



Modello di riferimento (cont.)

- Un *data source* incapsula sia la posizione del media che il protocollo per consegnare il media, costituisce infatti una astrazione di media protocol-handler.
- Data source e player sono parte integrante delle API di alto livello di JMF per la gestione della cattura, presentazione, ed elaborazione di time-based media. L'architettura di alto livello di JMF è riportata di seguito:



Manager

- Le API di JMF consistono principalmente di interfacce
- Le interfacce definiscono il comportamento e l'interazione tra gli oggetti usati per:
 - catturare, elaborare e presentare time-based media
- Attraverso oggetti intermediari chiamati *managers* è possibile integrare nuove implementazioni delle interfacce
- JMF utilizza quattro *managers*:
 - Managers, PackageManager, CaptureDeviceManager, PlugInManager

Manager (cont.)

- **Manager:** gestisce la costruzione di *Players*, *Processors*, *DataSources*, e *DataSinks*.
 - dalla prospettiva del client, non c'è alcuna differenza nell'utilizzo di un manager di default rispetto ad una realizzazione personalizzata (custom).
- **PackageManager:** gestisce un registro di package che contengono le classi JMF, come dei custom *Player*, *Processor*,...
- **CaptureDeviceManager:** gestisce un registro dei dispositivi di acquisizione disponibili
- **PluginManager:** gestisce un registro dei plug-in di componenti di elaborazione quali Multiplexers, Demultiplexers, Codecs, Effects e Renders.

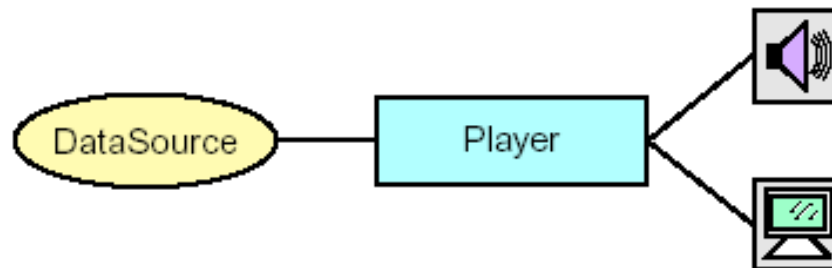
Manager (cont.)

Una generica applicazione basata su JMF utilizza il metodo *create()* del *Manager* per costruire i *Players*, *Processors*, e *DataSinks*.

Se si è interessati ad acquisire dei media da dispositivi di input, è necessario usare la classe *CaptureDeviceManager* per trovare i dispositivi disponibili e accedere alle loro informazioni.

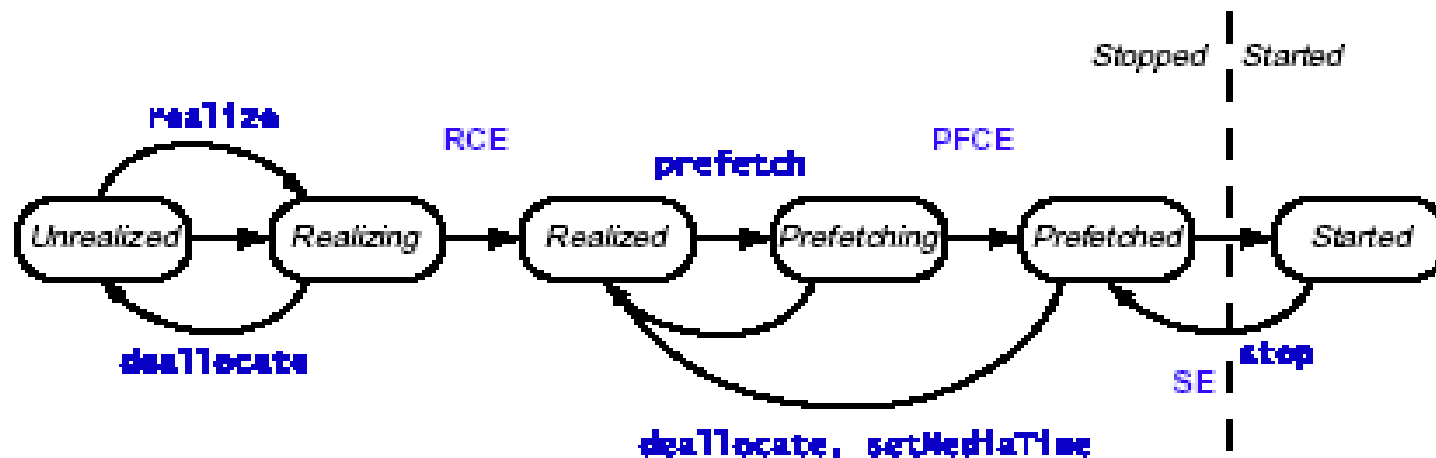
Presentazione di time-based media

- Per riprodurre un time-based media come audio e video con JMF bisogna creare un oggetto *Player*.
- Il player viene creato attraverso il metodo `createPlayer()` su *Manager* passando come parametro uno dei seguenti oggetti:
 - *URL* (Uniform Resource Locator)
 - *MediaLocator*
 - *DataSource*.



Stati di un Player

- Un player si può trovare in uno qualsiasi dei seguenti sei stati: *Unrealized*, *Realizing*, *Realized*, *Prefetching*, *Prefetched*, *Started*.
- ✓ Un Player istanziato ma senza un media associato si trova in uno stato *Unrealized*, quando ha acquisito le risorse si porta in uno stato *Realized*. Quando viene invocato il metodo *Prefetch()* il Player carica il media preparandosi per la riproduzione e transitando nello stato *Prefetched*. A questo punto il metodo *Start()* fa passare il Player nello stato *Started*.



Transition Events:

RCE	RealizeCompleteEvent
PFCE	PrefetchCompleteEvent
SE	StopEvent

Controllo degli stati di un Player

➤ L'interfaccia Player eredita dall'interfaccia Controller per cui, se la nostra classe vuole diventare listener degli eventi del Player deve implementare l'interfaccia ControllerListener e registrarsi come listener del Player attraverso il metodo:

addControllerListener(this)

eseguito sul Player.

➤ L'esempio [ProvaPlayer.java](#) mostra la creazione di un Player e la gestione degli eventi generati da questo nelle transizioni di stato.

Eseguire il programma passando come parametri di ingresso i seguenti MediaLocator :

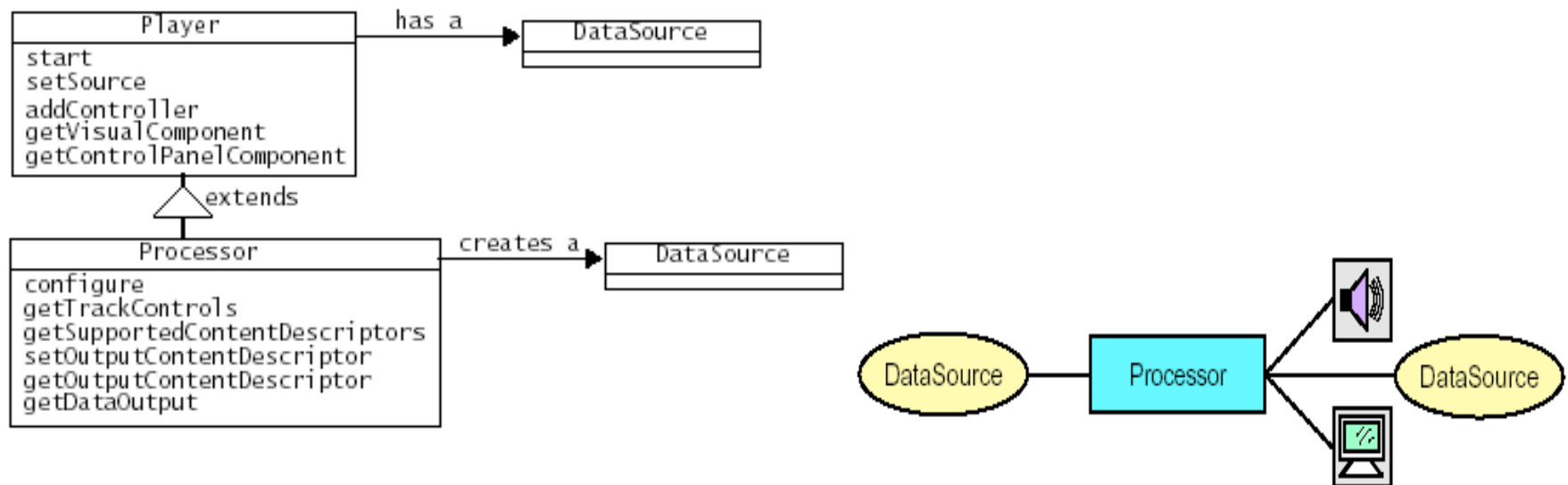
✓file:/D:/myDirectory/movie.mov

✓http://java.sun.com/products/java-media/jmf/2.1.1/samples/samples/media/darkcity.7.160x120.11khz.mov

Processor

Un *Processor* è un particolare *Player* che prende un *DataSource* in input, esegue qualche elaborazione definita dall'utente sui dati multimediali, e poi rilascia l'output.

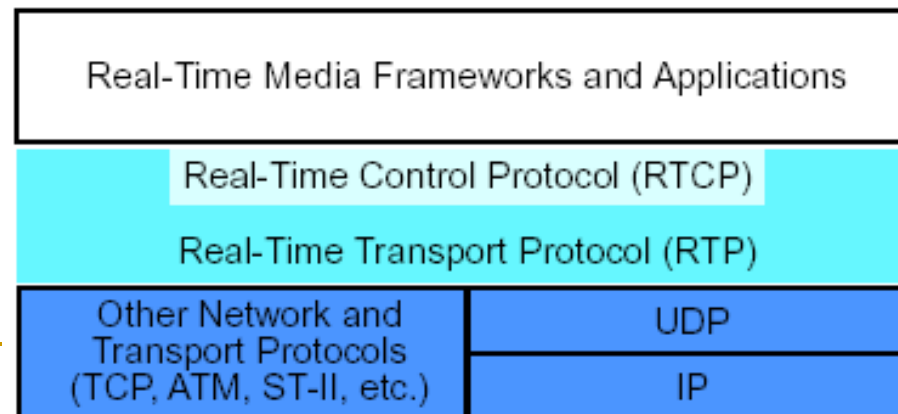
Differentemente da un *Player* può spedire i dati di output a un dispositivo di presentazione oppure ad un *DataSource*. Se i dati sono spediti ad un *DataSource*, questo può essere usato a sua volta come input per un altro *Player* o *Processor*.



Real-Time Media Stream

Real-Time Transport Protocol (RTP)

- RTP offre un servizio di consegna end-to-end per trasmissioni di dati real-time.
- RTP è indipendente dalla rete e dal protocollo di trasporto, sebbene sia spesso utilizzato su UDP.
- RTP può essere usato sia per servizi di rete di tipo unicast che multicast. Con servizi di rete unicast, diversi flussi dati sono spediti dalla sorgente ad ogni destinazione. Con servizi di rete multicast, i dati sono spediti da una sola sorgente ed è la rete che si occupa di consegnarla ai vari destinatari.



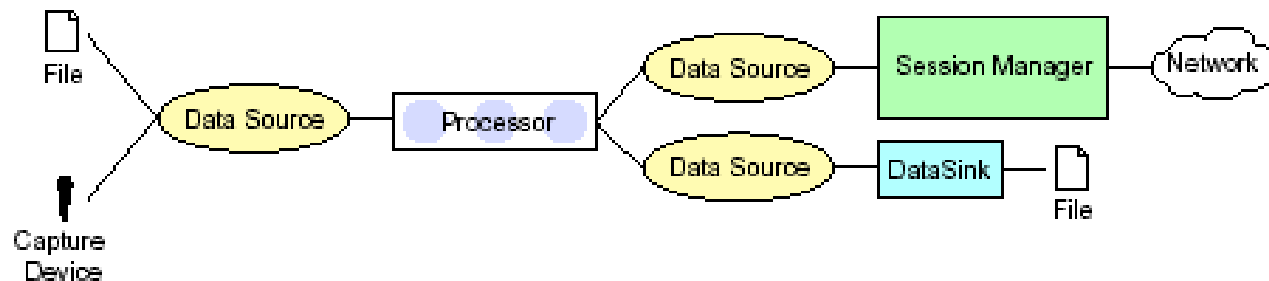
Real-Time Media Stream (cont.)

- RTP non offre alcun meccanismo per assicurare una consegna tempestiva dei dati
- Il protocollo di controllo, RTCP, permette però di monitorare la qualità della trasmissione dei dati attraverso dei pacchetti generati dai client che periodicamente riassumono la qualità della propria ricezione.

Trasmissione di stream RTP

Per trasmettere uno stream RTP si seguono i seguenti passi:

- Creare un *DataSource* o *MediaLocator* per identificare la sorgente dello stream
- Creare un *Processor* da collegare alla sorgente del punto precedente per produrre in uscita un *DataSource* con un particolare formato compresso RTP (attraverso la configurazione del *Processor*)
- A questo punto sono possibili due soluzioni:
 - ✓ Creare un *DataSink* che prenda in input il *DataSource* di uscita dal *Processor* per controllare la trasmissione
 - ✓ Creare un *RTPManager* per la trasmissione che permette la trasmissione di stream multipli nella sessione e permette inoltre di monitorare le statistiche associate alla sessione.



Trasmissione di stream RTP (cont.)

Un esempio di trasmissione di un flusso multimediale RTP con l'uso di un DataSink è offerto dalla classe [VideoTransmit.java](#).

Essa prende in input come primo parametro una URL del tipo:

- file:/C:/media/speech.mov
- http://mediacentral.com/speech.avi
- vfw://0

Come secondo parametro l'indirizzo IP della macchina destinazione e come terzo parametro il porto destinazione.

La classe java può quindi essere eseguita col comando:

```
java VideoTransmit file:/C:/media/speech.mov 224.224.224.224  
48000
```

Trasmissione di stream RTP (cont.)

In alternativa un esempio di utilizzo dell'RTPManager è proposto nella classe [AVTransmit2.java](#). Essa crea due sessioni sull'indirizzo IP dato come parametro differenziando il porto della seconda sessione di 2 unità.

In input come primo parametro accetta una URL del tipo:

- file:/C:/media/speech.mov
- http://mediacentral.com/speech.avi
- vfw://0

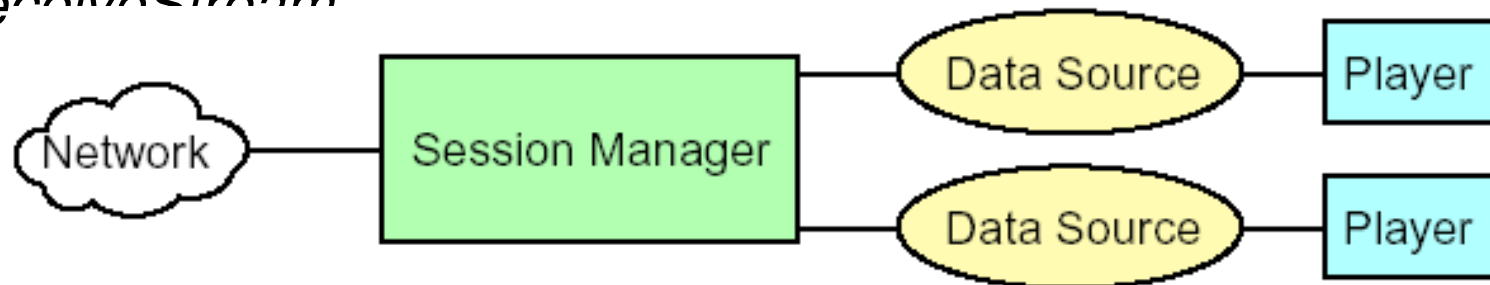
Come secondo parametro l'indirizzo IP della macchina destinazione e come terzo parametro il porto destinazione.

La classe java può quindi essere eseguita col comando:

```
java AVTransmit2 file:/C:/media/speech.mov 143.225.229.13  
48000
```

Ricezione e presentazione di stream RTP

- Per ricevere e riprodurre uno stream multimediale proveniente dalla rete bisogna costruire un *Player* per ogni stream trasmesso.
- Il *Player* può essere creato attraverso il metodo *createPlayer* invocato sul *Manager* passando come parametro o il *MediaLocator* oppure il *DataSource*. Se si usa il *MediaLocator* per costruire il *Player*, è possibile presentare soltanto il primo stream RTP della sessione.
- Se si vuole presentare più stream RTP nella sessione, si deve usare l'*RTPManager* direttamente e costruire un player per ogni *ReceiveStream*



Ricezione e presentazione di stream RTP

- Per ricevere e riprodurre ogni stream ricevuto in una sessione con l'*RTPManager* bisogna seguire i seguenti passi:
- Inizializzare la sessione RTP:
 - ✓ Creare un *SessionManager*. Per esempio, costruire un'istanza della *RTPManager* (una implementazione di *SessionManager*):
RTPManager mgrs= RTPManager.newInstance();
 - ✓ Registrarsi come listener:
mgrs.addSessionListener(this);
mgrs.addReceiveStreamListener(this);
 - ✓ Inizializzare la sessione:
mgrs.initialize(localAddress);
 - ✓ Far partire la sessione:
mgrs.addTarget(destAddress);
- Nel metodo *update* di *ReceiveStreamListener* gestire l'evento *NewReceiveStreamEvent* che indica la ricezione di un nuovo stream

Ricezione e presentazione di stream RTP

- Quando un `NewReceiveStreamEvent` arriva, si recupera lo stream arrivato col metodo `getReceiveStream`
- Si ottiene il `DataSource` dallo stream ricevuto richiamando `getDataSource`
- Si passa il `DataSource` al metodo `createPlayer` del `Manager` per costruire un nuovo `Player`. Deve essere disponibile il plugin per la decodifica e la spaccettizzazione dei dati formattati dello stream.

Ricezione e presentazione di stream RTP

Un esempio di utilizzo dell'RTPManager per la ricezione di stream è proposto nella classe [AVReceive2.java](#). Essa si pone in ascolto su due sessioni con indirizzi IP passati come parametri. Per ognuna delle sessioni si aprono tanti Player quanti sono gli stream trasmessi.

In input sono necessarie le due sessioni espresse come nel seguente esempio:

```
java AVReceive2 143.225.229.13/48000 143.225.229.13/48002
```

Riferimenti

- Sun Microsystem - JMF:

<http://java.sun.com/javase/technologies/desktop/media/jmf/index.jsp>

- Gli esempi trattati e molti altri possono essere trovati all'indirizzo:

<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/index.html>

- API specification:

<http://java.sun.com/javase/technologies/desktop/media/jmf/reference/api/index.html>

- Java Media Framework API Guide:

<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/index.html>