COMP S888 Multimedia Technology (Free Courseware)

香港公開大學
THE OPEN UNIVERSITY OF HONG KONG

Contents

Chapter 1 Streaming technology

1.1 About this module

Welcome to this free courseware module 'Streaming technology'!

This module is taken from the OUHK course *COMP S888 Multimedia Technology (http://www.ouhk.edu.hk/wcsprd/Satellite?pagename=OUHK/tcGenericPage2010&lang=eng&TYPE=CI&CODE=MT888)*, a 10-credit, postgraduate level course that is part of the Master of Science in Information Technology with Internet Applications and Master of Electronic Commerce programmes offered by the School of Science and Technology (http://www.ouhk.edu.hk/wcsprd/Satellite?pagename=OUHK/tcSubWeb&l=C_ST&lid=191133000200&lang=eng) of the OUHK. The course contains practical activities and illustrative programs in the Java language to illustrate important concepts such as compression and coding, audio and image processing, and digital entertainment, all of which play a central role in multimedia system development and analysis. Since the field of multimedia is rapidly changing, students will also be directed to read related articles and resources available on the Internet to keep up with the latest developments.

COMP S888 is mainly presented in printed format and comprises ten study units. Each unit contains study content, activities, self-tests, assigned readings, etc. for students' self-learning. This module (The materials for this module, taken from Unit 6 of the print-based course COMP S888, have been specially adapted to make them more suitable for studying online. In addition to this topic on 'Streaming technology', the original unit also includes the topics 'network features and performance', 'requirements of multimedia networking', 'Internet real-time and multimedia protocols', 'multicasting', and 'multimedia conferencing standards'.) retains most of these elements, so you can have a taste of what an OUHK course is like. Please note that no credits can be earned on completion of this module. If you would like to pursue it further, you are welcome to enrol in *COMP S888 Multimedia Technology (http://www.ouhk.edu.hk/wcsprd/Satellite?pagename=OUHK/tcGenericPage2010&lang=eng&TYPE=CI&CODE=MT888)*.

This module will take you about **six hours** to complete, including the time for completing the activities and self-tests (but not including the time for assigned readings).

Good luck, and enjoy your study!

1.2 Introduction

With the success of websites such as YouTube, streaming is no doubt one of the most important multimedia applications. This module explains two very different approaches to building a streaming application.

With a streaming server such as Microsoft Windows Media Server, you can put together a streaming application very quickly.

For applications with more specific requirements, however, you usually have to resort to programming. Java Media Framework API (JMF) is an application programming interface commonly used for developing multimedia applications in Java.

These two approaches are described in the rest of this module.

1.3 Streaming servers

The RTSP protocol is designed for providing streaming service. Streaming is a technique for transferring data in such a way that it is not necessary for user to wait for the entire file to arrive before it's played back. In other words, the client browser starts playing the streamed data while the data is still being sent from the server.

The RTSP protocol

RTSP is an application level protocol designed for the implementation of streaming service over the Internet. It provides VCR-type control over the playback of multimedia data. Similar to HTTP, RTSP is a client-server service involving a media server to provide the data. But unlike HTTP, which is stateless, RTSP has to maintain a session state for a stream such that the correct action can be taken when a command is issued. For example, when a movie is already playing, pressing the play button again should not invoke any action. Furthermore, HTTP requests are always initiated by clients, but in RTSP, both client and server can initiate requests. For example, the server may inform the clients to connect to another server.

Multimedia data requires a continuous data stream, and jitter is unavoidable on the Internet. A small buffer is therefore usually implemented on the client side so that data is played from the buffer while the buffer is being replenished. As long as the buffer is large enough, the effect of jitter can be smoothed out.

Incorporation of audio and video data within a Web document is now a common practice. However, ordinary Web servers are *not* designed to deliver streaming media. Therefore, a special class of server, called a 'streaming server' or 'media server', has been designed to do the job. The cooperation between a Web server and a streaming server is shown in Figure 1.1.

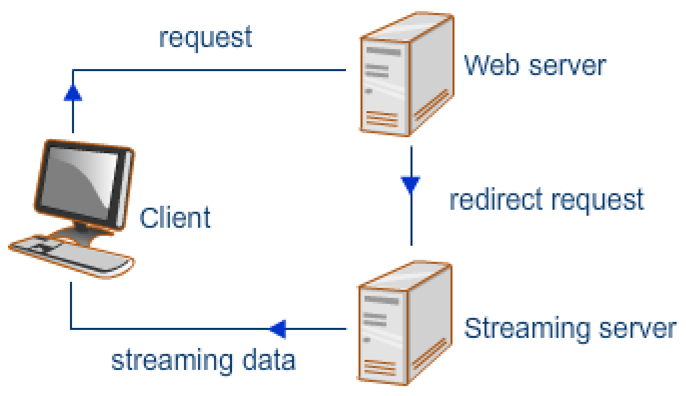


Fig. 1.1: Cooperation between a Web server and a streaming server

There are a number of streaming servers available either commercially or free of charge. In this module, we'll concentrate on introducing you to one of the most commonly-used streaming servers, i.e. Microsoft's Windows Media Server.

1.4 Microsoft's Windows Media Server

In the Windows environment, Microsoft Windows Media Server is the de-facto standard. Windows Media Server comes as part of Windows 2003 Server. Needless to say, its compatibility with Windows simply has no comparison.

Windows Media Server provides streaming support with two protocols:

- **Microsoft Media Server (MMS) protocol** -- This is a proprietary streaming protocol which can run on UDP, TCP and even HTTP. Although MMS over HTTP may appear to be a strange combination, it is almost guaranteed to work over any firewall.
- **Real-time Streaming Protocol (RTSP)** -- RTSP provides VCR-like commands such as play and pause, and random access to files on a server.

Windows Media Server's standard formats for streaming video and audio are WMV and WMA, respectively. These formats can be created using the Windows Media Encoder. Live inputs can also be captured using Windows Media Encoder. The design of WMV appears to be based on MPEG-4. To control the playback of media streams, an advanced stream redirector (ASX) file has to be used. ASX files are plain text files that manage streaming of WMV and WMA files.

As the name suggests, the main task of ASX files is to redirect requests for media to the streaming server, as explained in Figure 1.1. At a minimum, an ASX file needs to contain a URL that points to the server and file that the user wants to play. A simple ASX file is shown below:

```
<ASX version = "3.0">
<Entry>
    <Ref href = "mms://WindowsMediaServer/MyFile.wmv" />
</Entry>
</ASX>
```

ASX has many options for refining control of the playback. It can join multiple media types together to be played sequentially or in parallel. A detailed description of the format and use of ASX files can be found here (http://www.microsoft.com/windows/windowsmedia/howto/articles/introwmmeta.aspx) if you are interested.

It is also possible to embed a streaming media within a webpage, as shown in Figure 1.2.

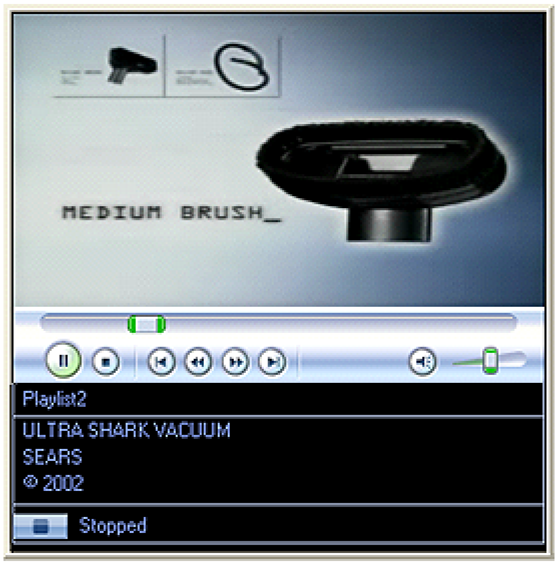


Fig. 1.2: Embedded streaming media in a webpage Windows Media Embed Sample

Source: http://www.audiovideoweb.com/windowsvideoembedsample.htm (visited in July 2012)

The following code embeds a streaming media into a webpage:

```
<OBJECT ID="MediaPlayer" WIDTH=320 HEIGHT=240
classid="CLSID:22D6F312-B0F6-11D0-94AB-0080C74C7E95"
codebase="http://activex.microsoft.com/activex/controls/mplayer/en/
nsmp2inf.cab#Version=6,4,7,1112"
standby="Loading Microsoft Windows Media Player components..."
type="application/x-oleobject">
     <PARAM NAME="FileName"
VALUE="">http://interface.audiovideoweb.com/lnk/avweb_windows_demos/
sears_1.wmv/play.asx">;
     <PARAM NAME="ShowControls" VALUE="1">
     <PARAM NAME="ShowDisplay" VALUE="1">
     <PARAM NAME="ShowStatusBar" VALUE="1">
     <PARAM NAME="AutoSize" VALUE="1">
     <Embed type="application/x-mplayer2"
pluginspage="http://www.microsoft.com/windows/windowsmedia/download/
AllDownloads.aspx/"
filename="http://interface.audiovideoweb.com/lnk/avweb_windows_demos/
sears_1.wmv/play.asx"
src="http://interface.audiovideoweb.com/lnk/avweb_windows_demos/
sears_1.wmv/play.asx"
     Name=MediaPlayer
     ShowControls=1
     ShowDisplay=1
     ShowStatusBar=1
     width=320
     height=240>
     </embed>
</OBJECT>
```

Click this link to watch the video:
http://www.opentextbooks.org.hk/system/files/resource/10/10528/10533/media/Microsoft%27s%20Windows%20Media%20Server.mp4

Click here. (http://www.audiovideoweb.com/index.php)

Basically, the above code embeds a Windows Media Player in a webpage and then passes a number of parameters to it using the PARAM element. These parameters control the player's look and functionality. Each of the parameters has a name and a value associated with it. Most of the parameters used above should be self-explanatory.

If you want to develop your own application with Windows Media Server, you can download the Windows Media Server SDK from Microsoft. With the SDK, you can programmatically configure the application, monitor both the server and clients connected to it, and access all logging statistics. The following is a list of functionalities provided by the SDK:

- authentication
- cache/proxy
- control protocol
- data writer
- data source
- event notification and Authorization
- logging
- media parser
- playlist parser.

As you would expect, the SDK works well with Microsoft Visual Studio. Further information about the SDK can be obtained here (http://en.wikipedia.org/wiki/Software_development_kit).

There are other popular streaming servers for different platforms. For example, the Darwin Streaming Server is an open source version of Apple's Quicktime Streaming Server. The Darwin Streaming Server is available for various platforms including Linux, Mac and Windows. Other commercial products are available from Real Networks (www.realnetworks.com) as well as Sun (www.sun.com).

1.4.1 Activity 1

Below you will find a link to an example of a webpage with embedded .wmv files. Study the source code of the webpage and modify it so that it can be automatically started and then repeats five times. To view the source code, you can right-click this page and select 'View source code'.

Sample webpage with embedded.wmv files (http://learn.ouhk.edu.hk/~mt888/activity/unit6/act6_4.htm)

1.4.2 Activity 2

Microsoft has launched a new version of its streaming server software. If you are interested in learning more about it, you can check out the features of it here (http://technet.microsoft.com/en-us/windowsserver/dd448620.aspx).

1.5 Java Media Framework API (JMF)

JMF is an application programming interface for incorporating audio, video and other time-based media into Java applications and applets. The current version of JMF is 2.1.1e. It's available for download here (http://www.oracle.com/technetwork/java/javase/download-142937.html).

JMF is very big and complicated. It includes a large number of classes to provide the following functionalities:

- play and stream different types of multimedia file (AVI, MPEG and WAV, etc.)
- capture audio and video data from microphones or cameras
- broadcast, unicast and multicast multimedia data over the Internet.

One of JMF's important design considerations is to support standard protocols. JMF provides these functionalities by working on top of RTP and RTCP.

The architecture of JMF is shown in Figure 1.3. One obvious feature of the architecture is that it contains some native components that talk directly to hardware. These components include the codecs, capturers and renderers. They are responsible for the capturing, coding/decoding and playback of media data. These tasks are intimately related to the hardware, so a native approach can help to boost performance and get around security issues, because Java code is not allowed to have direct control over the hardware of the host operating system. Apart from these exceptions, the core of JMF is implemented in pure Java.

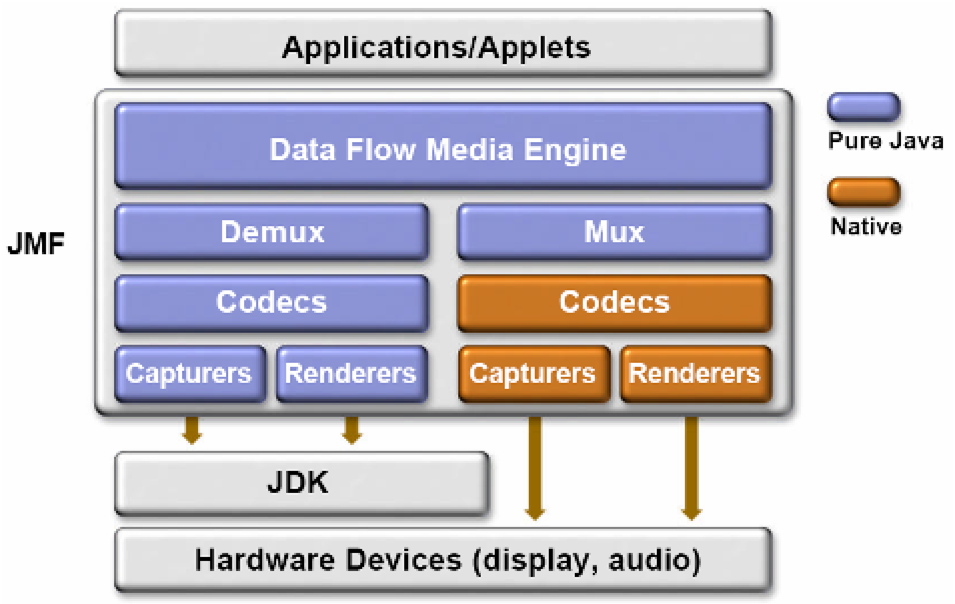


Fig. 1.3: JMF architecture

The hardware dependency of JMF creates an inconvenience for JMF applications, however. JMF must be installed on any client machine that runs these applications. This is one reason why JMF has not been very successful so far.

JMF components are designed after the model of media playback as shown in Figure 1.4. This model is familiar to us because it is very similar to the way common electronic devices are connected together - and how they have been for quite a long time. In this fundamental model, a video camera represents a capture device; the media data (a video tape, for example) is encapsulated as a data source; and the VCR, as a player, controls the playback of media data. Finally, during playback, the data is rendered on destination devices such as loudspeakers and a television.

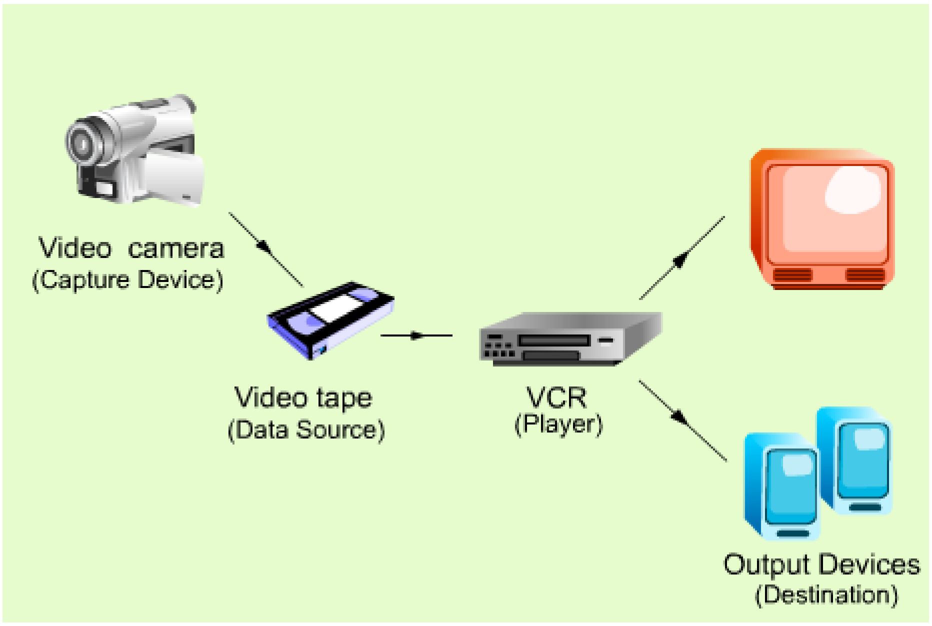


Fig. 1.4: JMF concept of recording, processing and presenting time-based media

Source: Sun Microsystems 1999, figure 2-1.

In Figure 1.4, the data source represents media such as video or audio objects. Java is an object-oriented programming language, so the DataSource class encapsulates everything required to retrieve the content, including the location of the file (which could be a local file or an Internet link), and determines what protocol and decoder should be used to read the content.

JMF DataSource can be classified according to how a data transfer is initiated:

- Pull data source -- The client initiates the transfer of data, and flow control is done on the client side. For example, HTTP is a protocol for this type of data.
- Push data source -- The transfer of data is initiated by the server, and flow control is also managed by the server. An example is multimedia broadcasting.

Several DataSources (audio, video and text) may be combined to form a single DataSource (i.e. a movie) for easy control. Capture devices are the hardware you use to capture the data. They include microphones, video cameras, etc. You should note that a video camera delivers both video and audio data. Capture devices can also be classified as either push or pull sources. For example, a still camera is a pull source because the user controls when to capture an image. A microphone is a push source because the microphone provides a continuous stream of audio data.

The Player is the component that actually brings the data to a speaker and screen. It is not concerned with how the original data in the data source should be interpreted.

This is the job of the DataSource class. Its main function is to control the playback of data, very much like the VCR in Figure 1.4. The data from capture devices can also be fed to a Player for playback. Just as an ordinary VCR can have different states (stopped, playing, paused, etc.), a Player also has a number of states. There are six states defined for a Player; the interactions between these states are shown in Figure 1.5.

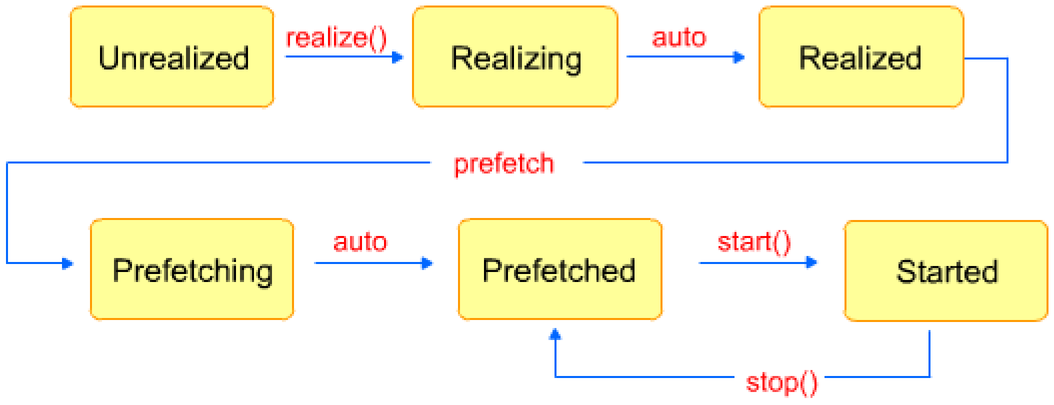


Fig. 1.5: Six states of the Player class

The meaning of each state is briefly explained here:

- Unrealized -- The Player has just been created, and knows nothing about its media content.
- Realizing -- The Player is determining the resources requirements of its media content.
- Realized -- The Player has already worked out the resources requirements of its media content.
- Prefetching -- The Player is preparing itself for data playback.
- Prefetched -- The Player is now ready to start playing.
- Started -- The Player is playing the media content.

Below is a simple example of the Player class in practice:

```
import java.net.*;
import javax.media.*;
import java.awt.*;

public class JMFAudio extends Applet{
     Player player;

     public void init() {
     try {
         URL url = new URL(getCodeBase(), "a.mp3");
         // create a player using the url
         player = Manager.createPlayer(url);
```

```
        }
        catch (Exception e) {
            System.out.println(e);
            }
        }
        public void start() {
        player.start();
        }
        public void destroy() {
        player.stop();
        player.close();
        }
  }
```

Download a.mp3 (http://olefree.ouhk.edu.hk/course1126/comps888.nsf/cm_lookup/00000506/%24File/a.mp3)

Click this link to watch the video: http://www.opentextbooks.org.hk/system/files/resource/10/10528/10536/media/JMF%20Audio.mp4

As you can see, it is not very complicated. To use the JMF classes, the package javax.media* has to be imported. The Manager class is a factory class that handles the construction of other objects in JMF. In the init() method of the applet, a Player is instantiated, using Manager, with the given URL of the audio file. In the applet's start() method, the player is started. By calling the start() method, the player is implicitly asked to go through all the states listed above until the Started state is reached. When the browser unloads the applet, the destroy() method is invoked, in which the player is stopped and closed. It is important to close the player to ensure that the audio hardware is released.

1.5.1 Streaming with JMF

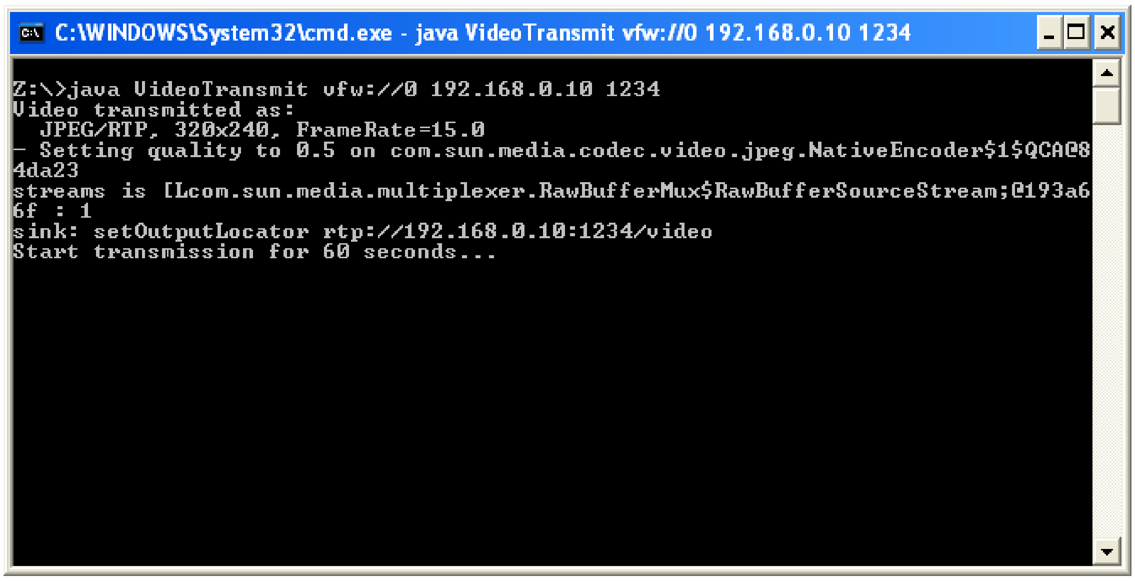
JMF supports streaming media over the RTP and RTSP protocols. To get a more concrete idea of how streaming is supported, get the following files:

- VideoTransmit.java (http://learn.ouhk.edu.hk/~mt888/activity/unit6/VideoTransmit.java)
- VideoReceive.java (http://learn.ouhk.edu.hk/~mt888/activity/unit6/VideoReceive.java)

VideoTransmit is an application that serves as an RTP video server. After compilation, it should be executed as:

```
java VideoTransmit vfw://0 192.168.0.10 1234
```

which states that it will capture from the capture device vfw://0, which represents the camera, and will send the RTP stream to 192.168.0.10 using port 1234.


C:\WINDOWS\System32\cmd.exe - java VideoTransmit vfw://0 192.168.0.10 1234

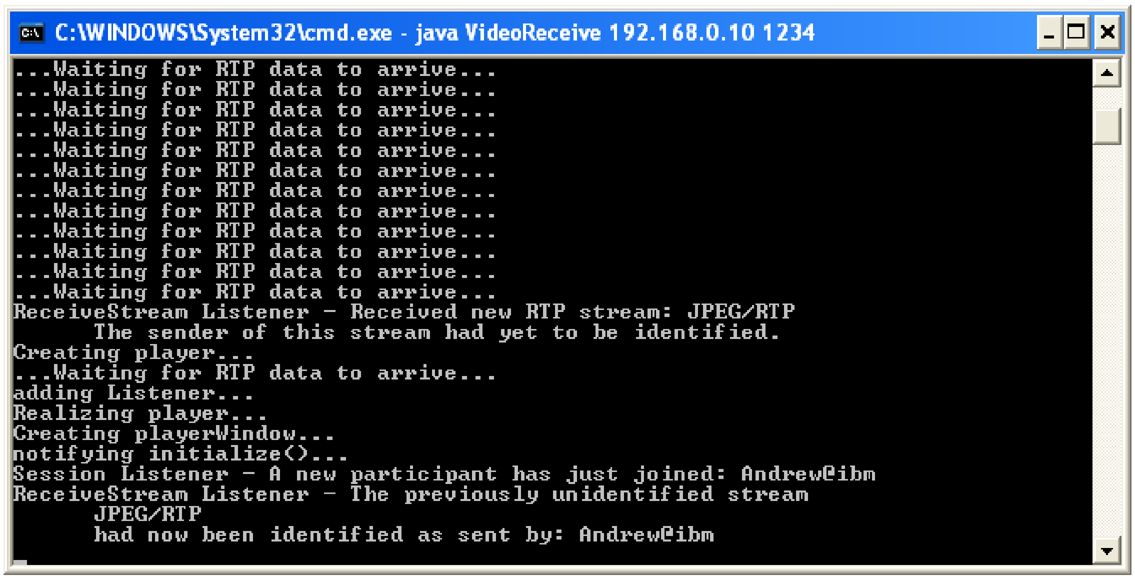
```
Z:\>java VideoTransmit vfw://0 192.168.0.10 1234
Video transmitted as:
  JPEG/RTP, 320x240, FrameRate=15.0
- Setting quality to 0.5 on com.sun.media.codec.video.jpeg.NativeEncoder$1$QCA@8
4da23
streams is [Lcom.sun.media.multiplexer.RawBufferMux$RawBufferSourceStream;@193a6
6f : 1
sink: setOutputLocator rtp://192.168.0.10:1234/video
Start transmission for 60 seconds...
```

Fig. 1.6: Result of starting the VideoTransmit server

The client should be executed as:

```
java VideoReceive 192.168.0.10 1234
```

which states the source of the RTP stream and port number (I am using only one computer).



```
C:\WINDOWS\System32\cmd.exe - java VideoReceive 192.168.0.10 1234
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
...Waiting for RTP data to arrive...
ReceiveStream Listener - Received new RTP stream: JPEG/RTP
      The sender of this stream had yet to be identified.
Creating player...
...Waiting for RTP data to arrive...
adding Listener...
Realizing player...
Creating playerWindow...
notifying initialize()...
Session Listener - A new participant has just joined: Andrew@ibm
ReceiveStream Listener - The previously unidentified stream
      JPEG/RTP
      had now been identified as sent by: Andrew@ibm
```

Fig. 1.7: Detecting an RTP stream by VideoReceive

If an RTP stream is detected by the client, a window will be opened with the necessary controls and visual components needed to view the video.

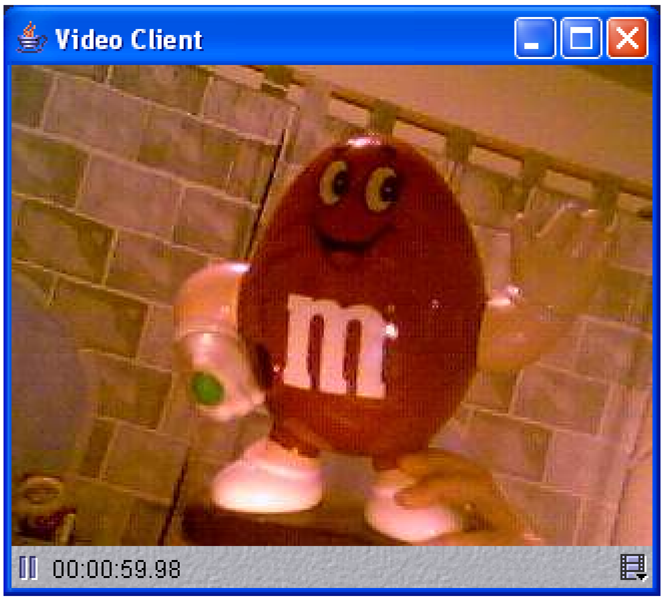


Fig. 1.8: JMF client receiving video stream

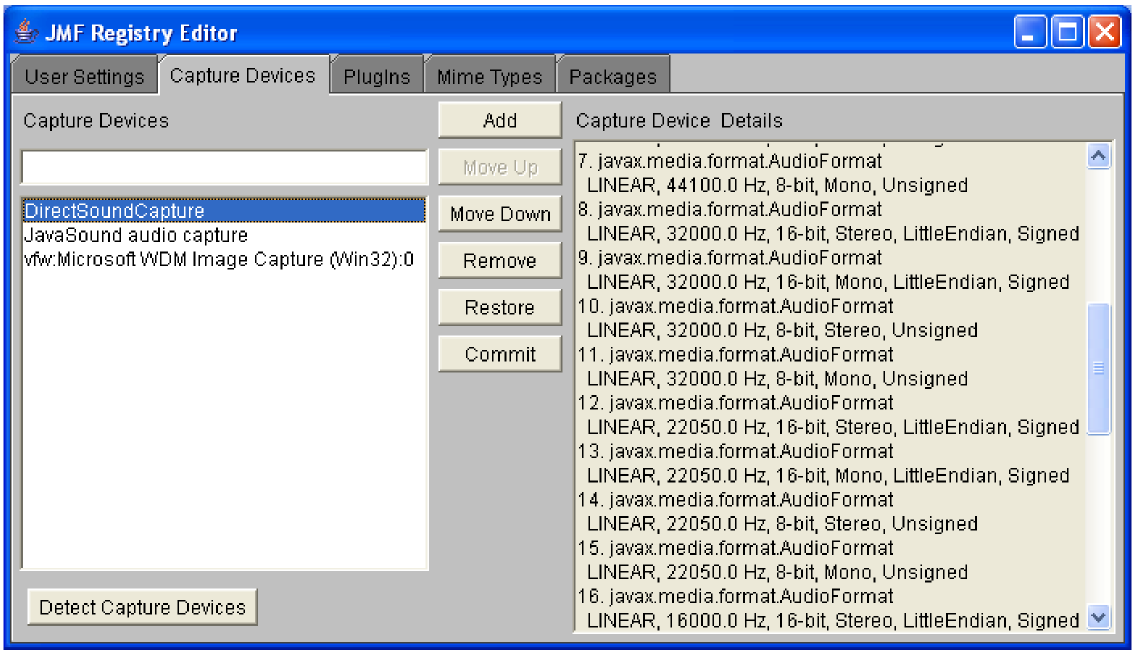
Click this link to watch the video:
http://www.opentextbooks.org.hk/system/files/resource/10/10528/10537/media/Video%20Transmit%20Program_0.mp4

I do not intend to go into the details of VideoTransmit.java and VideoReceive.java because this is not a programming course. Feel free, however, to take a look at the source code to appreciate how a minimal RTP server and client can be constructed with JMF.

To sum up, JMF is an attempt by Sun to embrace the media streaming market using the Java technology. It was designed to be compatible with standard protocols such as RTP. Both Java and RTP streaming are intimately related to the Internet, and appeared to be a perfect match. Yet JMF didn't see tremendous success. JMF's major problem is that it is complicated. Even an experienced Java programmer would have a hard time learning all the JMF classes in VideoTransmit.java and VideoReceive.java. A second deadly blow to JMF is the requirement that it be installed separately on client machines. When compared with other media players such as Windows Media Player and Flash Player, installing JMF is too much of a nuisance. JMF is therefore more suitable for building in-house solutions and for use in situations in which compatibility with standard protocols is essential.

1.5.1.1 Activity 3

Connect a USB camera (any cheap camera will do) to your computer. Run the JMF Registry application (installed together with JMF) as shown below.



```
DirectSoundCapture
JavaSound audio capture
vfw:Microsoft WDM Image Capture (Win32):0
```

```
7. javax.media.format.AudioFormat
 LINEAR, 44100.0 Hz, 8-bit, Mono, Unsigned
8. javax.media.format.AudioFormat
 LINEAR, 32000.0 Hz, 16-bit, Stereo, LittleEndian, Signed
9. javax.media.format.AudioFormat
 LINEAR, 32000.0 Hz, 16-bit, Mono, LittleEndian, Signed
10. javax.media.format.AudioFormat
 LINEAR, 32000.0 Hz, 8-bit, Stereo, Unsigned
11. javax.media.format.AudioFormat
 LINEAR, 32000.0 Hz, 8-bit, Mono, Unsigned
12. javax.media.format.AudioFormat
 LINEAR, 22050.0 Hz, 16-bit, Stereo, LittleEndian, Signed
13. javax.media.format.AudioFormat
 LINEAR, 22050.0 Hz, 16-bit, Mono, LittleEndian, Signed
14. javax.media.format.AudioFormat
 LINEAR, 22050.0 Hz, 8-bit, Stereo, Unsigned
15. javax.media.format.AudioFormat
 LINEAR, 22050.0 Hz, 8-bit, Mono, Unsigned
16. javax.media.format.AudioFormat
 LINEAR, 16000.0 Hz, 16-bit, Stereo, LittleEndian, Signed
```

JMF Registry shows the detected capture devices together with the supported format for each device. If no capture device is shown, use the 'Detect Capture Devices' button to try to detect the devices.

After confirming that JMF is able to capture the video device vfw://0, start the VideoReceive application, and then the VideoTransmit application. Make sure you are using an available port on your computer (you learned earlier in this unit how to check which ports are being used), and remember to substitute your IP address.

1.5.1.2 Activity 4

If you are interested in further study of JMF and its applications for streaming media, you can check out the following links.

This article (http://www.interactivetvweb.org/tutorials/ocap/jmf) provides a good overview of applying JMF to real-world situations.

This article (http://www2002.org/CDROM/alternate/XS3/) includes a section (6.2) on how JMF has been used here in Hong Kong to facilitate the development of an intelligent video content management system.

1.6 References

Below are the resources referred to or cited by the developer(s) of the original unit:

Ferrari, D (1990) 'Client requirements for real-time communication services', *IEEE Communications Magazine*, 28(11): 65–72.

Fluckiger, F (1995) *Understanding Networked Multimedia: Applications and Technology*, Prentice Hall.

Kozierok, C M (2005) *The TCP/IPGuide*, http://www.tcpipguide.com/free/index.ht

Liu, C L (1998) 'Multimedia over IP: RSVP, RTP, RTCP, RTSP', http://www.cs.wustl.edu/~jain/cis788-97/ftp/ip_multimedia.pdf.

Sun Microsystems (1999) *Java^{TM} Media Framework API Guide*, http://java.sun.com/products/java-media/jmf/2.1.1/guide/.

1.7 Conclusion

This module has introduced you to two quite different ways of handling streaming content. You have been given the chance to compare their advantages and disadvantages, and even to begin to try them out for yourself. You should now be better-prepared for dealing with streaming content issues in your own media and web development.

If you would like to learn more on this subject, you are welcome to enrol in *COMP S888 Multimedia Technology (http://www.ouhk.edu.hk/wcsprd/Satellite?pagename=OUHK/tcGenericPage2010&lang=eng&TYPE=CI&CODE=MT888)* offered by the School of Science and Technology (http://www.ouhk.edu.hk/wcsprd/Satellite?pagename=OUHK/tcSubWeb&l=C_ST&lid=191133000200&lang=eng) of the OUHK.