第32卷 第21期 计 算 机 工 程 2006年11月
Vol.32 No.21 Computer Engineering November 2006
·工程应用技术与实现· 文章编号：1000—3428(2006)21—0238—03 文献标识码：A 中图分类号：TP391

基于JMF的音视频实时交互及存储的具体实现

张书梅[1]，HU Huosheng [2]
(1. 石家庄学院计算机系，石家庄 050035；2. Department of Computer Science, University of Essex Wivenhoe Park, Colchester CO4 3SQ, UK)

摘 要：针对目前多媒体远程教学及实时监控系统大部分是信息单向传输，缺乏用户间的多媒体信息实时交互功能及多媒体数据存储海量的弊端，文章提供了一种用JMF的数据克隆、数据轨道控制、流媒体的存储及传输等技术来实现网络音、视频数据双向传输及由用户选择和控制将音视频数据以MOV文件格式存储到本地硬盘的方案，并介绍了实现方法。系统软、硬件成本低，且数据量小，易于网络传输和存储。

关键词：JMF；远程教学；远程监控；多媒体

Implementation of Audio-visual Bilateral Transmission and Saving with JMF

ZHANG Shumei[1], HU Huosheng [2]
(1. Department of Computer Science, Shijiazhuang College, Shijiazhuang 050035;
2. Department of Computer Science, University of Essex Wivenhoe Park, Colchester CO4 3SQ, UK)

【Abstract】The disadvantages of traditional multimedia remote teaching and real-time monitoring are that the information transmitted is unidirectional, and that students receive information passively from Internet or Intranet, which may seriously affect their learning initiative and efficiency. In addition, because multimedia data needs a large space to store, this limits the speed of data transmission on the Internet, and influences the implementation of software development in audio and video bilateral transmission. This article provides a solution to the above disadvantages by adding a chat room featuring audio and visual contact to the traditional multimedia remote teaching websites. The system implements the function of the audio and video bilateral transmission with JMF and RTP on the Internet. In addition, users can save the video and audio to their hard disk in the movie file form if they want to and they can use it at their disposal. This system has the advantages of low cost for software and hardware, and has less data storage requirement than traditional DV does. It can transmit and store data easily.

【Key words】Java media framework(JMF); Remote teaching; Remote monitor; Multimedia

Java Media Framework (JMF) API是Sun免费提供的基于Java的多媒体框架，利用这个框架能够编写出功能强大的多媒体程序，却不用关心底层复杂的实现细节，所以用JMF实现网络上的多媒体实时交互行为更方便快捷[1]。作者最近用JMF开发一个用于远程监控机器人行为的多媒体软件时，实现了一个可应用到很多领域的功能，即：可让网络两端的用户进行音视频实时交互，并根据需要由用户来控制将一段音视频保存到一个MPG或MOV文件中。该功能也可用于远程教学或远程监控系统。查看了国内外很多著名远程多媒体教学网站，发现在现有的远程教学系统中，确实提供了大量的多媒体教学课件，并利用Internet的非同步教学和单向式预录视频教学技术实现了网上的虚拟大学。但不足之处是学生只能被动地收听学习，有问题不能及时与老师进行“面对面”交流，只通过无声无影的E-mail或BBS等文本或单向音视频来解答问题，教学效果肯定要打折扣。本文提供一种方案，在现有远程教学网站中增加一个可实现双向音视频实时交互的答疑室，且答疑过程中师生均可根据需要随时将实时的音视频数据录制到自己本地硬盘中保存，以便重复观看。师生或同学间通过远距离“面对面”交流，可激发学生的学习热情，提高学习效率及效果。另外，本方案中因是通过Microphone和网络摄像头(webcam)捕获音视频数据，以MOV格式存储到本地硬盘，所以具有网络传输速度快及占用存储空间小的优点。

1 相关的概念[1]

1.1 时基媒体或流媒体

时基媒体(Time-based Media)或流媒体(Streaming Media)是用于描述任何流动数据的一个术语，它要求必须实时地接收并处理数据。声音素材、MIDI序列、电影素材和动画都属于时基媒体。一些媒体数据可从多种资源获得，比如，本地或网络文件、数码相机、麦克风和现场广播等。时基媒体是稳定的信息流，在将它输出到用户端或保存到文件中之前需对它进行寻址、处理和输出。处理操作包括转换数据格式、压缩或解压缩数据、或与来自其它资源的数据流进行合并。

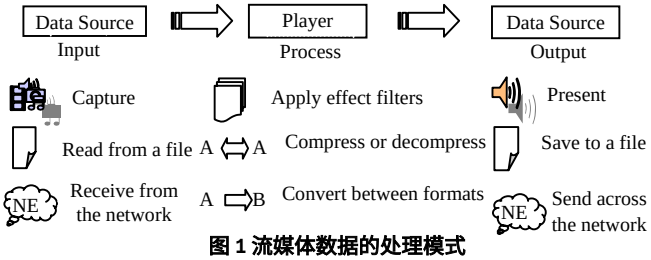

图1 流媒体数据的处理模式

作者简介：张书梅(1962－)，女，副教授、硕士，主研方向：多媒体技术；HU Huosheng，教授、博导

收稿日期：2006-06-07 **E-mail：**smz6688@tom.com

影响视频或声音质量的几个因素包括带宽、系统的处理效率及数据传输时采用的压缩格式。图 1 表示了流媒体数据的处理模式。

1.2 数据源

数据源(DataSource)就像录像带一样，其中包含了媒体流。JMF 数据源根据数据传输怎样被启动分为两类，即 Pull Data-Source 和 Push Data-Source。Pull Data-Source 可以是文件或 Web 页。为这种类型的数据制定的协议包括超文本传输协议(HTTP)和文件(FILE)。使用 Pull Data-Source 由客户端启动并控制数据的传输。

Push Data-Source 是由服务器启动并控制数据传输。Push Data-Source 包括广播媒体 (broadcast media)，在线组播媒体（Multicast media）和视频点播 (VoD) 。对 broadcast media 要用到实时传输协议 (Real-time Transport Protocol，RTP)。Push Data-Source 可以是麦克风或网络摄像头(Webcam)。

1.3 播放器

播放器(Player)是一个对时基媒体进行 Render 和 Control 的媒体处理者。播放器处理媒体数据的输入流并及时输出它。DataSource 被用于传递输入媒体流给 Player，至于 Player 输出媒体流的目的地要依赖媒体被输出的类型，如：声音被输出到扬声器，视频被输出到计算机屏幕。播放器可有 6 种状态。在整个事件发展过程中，播放器有 2 个主要状态：Stopped and Started。而 Stopped 状态又能被分解成 5 个待命状态：Unrealized, Realizing, Realized, Prefetching, and Prefetched。在正常情况下，Player 需要经历上述的每个状态直到它达到开始状态。

1.4 处理器

处理器(Processor)只是播放器的一个特殊类型，除了继承 Player 的所有功能外，还可以控制对于输入的媒体流进行何种处理。Processor 能发送输出数据到一个输出设备或到一个 DataSource。如果数据被发送到一个 DataSource，则这 DataSource 可以被用于另外的 Player 或 Processor 的输入，或作为 DataSink 的输入。Processor 还可解析媒体流，执行特殊功效的编码或解码，并可将多个输入轨道的数据融合到一起。如可将分开的视频和音频轨道中的数据流合并成单一的 MPEG-1 数据流。可以指定输出流的数据类型。

除了在播放器中提到了 6 种状态外，Processor 对象还有另外的两种新的状态，这两种状态是在 Unrealized 状态之后，但是在 Realizing 状态之前。

(1) Configuring：当调用 configure（）方法后，Processor 对象进入该状态。在该状态下，Processor 对象连接到数据源并获取输入数据的格式信息。

(2)Configured：当完成数据源连接，获得输入数据格式的信息后，Processor 对象就处于此状态。

当一个 Processor 在 Configured 状态时，可对某个单一的轨道调用 getTrackControls()方法获得对该轨道对象的控制。

1.5 捕捉设备

捕捉设备(Capture devices)是指可以捕获到视频或音频数据的硬件。如麦克风可以捕获音频数据、网络摄像头可捕获视频数据，因此它们两个都是数据源(Push DataSource)。捕获到的数据可以被送入到 Player 或 Processor 对象进行处理。

1.6 媒体数据存储和传输

DataSink 经常被用于从 DataSource 读取媒体数据并传送媒体数据到一些输出目的地。JMF 提供一默认的 DataSink，可以被用于写数据到文件。其它类型的 DataSink 类能写数据到网络或另外的目的地。注意，Plaryers、Processors 和 DataSinks 都属于 MediaHandlers，它们都是从 DataSource 读取数据。如果想保存捕获的媒体数据到文件，必须用 Processor 替代 Player，然后用 DataSink 从 Processor 对象输出的 DataSource 中读取数据并输出数据到文件。

2 代码示例

JMF 中包含了许多用于处理多媒体的 API。它是一个相当复杂的系统，上面只是介绍了一些主要概念。下面将通过远程机器人监控为例说明如何利用 JMF 进行编程。该软件包括几个功能：网络两端能从网络摄像头及麦克风接收对方的视频和音频，能保存照片到文件、能由用户控制录制一段视频及声音保存到一电影文件，从而实现远程监控机器人行动并根据需要录制现场数据。其主界面如图 2 所示。当然，此功能也可用于远程教学或远程会议或很多领域的远程监控，只需根据需要稍作改进即可。

图 2 软件主界面

因篇幅限制，本文只举例说明如何用 JMF 实现从网络摄像头、麦克风获取数据到屏幕和扬声器及保存一段录像到电影文件。关于怎样利用 JMF RTP API (Real-time Transport Protocol) 在 Internet 或 Intranet 上进行网络传输和接收这些数据将在下文介绍。

2.1 捕获视频及声音

在使用JMF之前，首先需正确设置开发及运行环境。关于如何设置运行Java及JMF环境的文章已有很多，这里就不详述了。为了使用视频和音频设备捕获媒体数据，需要以下几个步骤[1]：

(1)通过调用方法 CaptureDeviceManage()定位想用的捕获设备。

(2)对定位的设备获得相应的 CaptureDeviceInfo 对象。

(3)通过 CaptureDeviceInfo 对象获得 MediaLocator 并用之创建 DataSource。

(4)用 DataSource 创建 Player 或 Processor。

(5)Start 这个 Player 或 Processor，开始捕获媒体数据。

相应的代码示例如下[1,2]。注意，若单独考虑捕获音视频数据到屏幕，直接用SUN提供的示例代码即可。但为使录制电影文件与屏幕显示不冲突，这里用到了数据克隆。

```
//定位捕获设备
MediaLocator[] locatorVA = new MediaLocator[2];
locatorVA[0] = new MediaLocator("javasound://0");
locatorVA[1] = new MediaLocator("vfw://0");

//获得 MediaLocator 并用之创建 DataSource
DataSource[] dsc = new DataSource[2];
```

```
dsc[0] = Manager.createDataSource(locatorVA[0]);
dsc[1] = Manager.createDataSource(locatorVA[1]);
//将视频与音频合并
DataSource mainSource=Manager.createMergingDataSource(dsc);
//为了能让几个 Processor 访问同一数据源，需将之转换成可克
//隆的数据源
mainSource=Manager.createCloneableDataSource(mainSource);
//用克隆的数据源创建 Processor
DataSource camSource = (SourceCloneable)mainSource.
createClone();
Processor processor = Manager.createProcessor(camSource);
// 用类 camStateHelper 可帮助控制 Processor 的媒体事件
camStateHelper playhelper = new camStateHelper(processor);
//Configure, Realize 然后开始我们的 Processor
playhelper.configure(10000);
// Processor 的内容描述符必须设置成空，为防止 Processor 的数
//据源来自输出的原始数据
processor.setContentDescriptor(null);
playhelper.realize(10000);
processor.start();
//用 Processor 的可视组件将电影显示到屏幕上
processor.getVisualComponent().setBackground(Color.white);
centerPanel.add(processor.getVisualComponent(),
BorderLayout.CENTER);
```

2.2 保存一段录像到电影文件[3,4]

可用 DataSink 从 Processor 对象的输出数据源中读取数据并将之输出到文件中。具体步骤如下：

(1)创建一个 MediaLocator 对象用于指定想保存的文件位置。

```
URL movieUrl = file.toURL();
MediaLocator dest = new MediaLocator(movieUrl);
```

(2)再克隆一份数据源并用之又创建一个 Processor。

```
DataSource recordCamSource = dataSource.cloneCamSource();
Processor recordProcessor = Manager.createProcessor
 (recordCamSource);
camStateHelper playhelper = new camStateHelper
(recordProcessor);
```

(3)对该 Processor 调用 configure

```
Playhelper.configure(10000);
```

(4)对每个轨道调用方法 getTrackControls 和 setFormat。音视频的轨道序号应与捕获代码中的音视频顺序一致。

```
VideoFormat vfmt = new VideoFormat(VideoFormat.JPEG);
AudioFormat afmt = new AudioFormat(AudioFormat.LINEAR);
(recordProcessor.getTrackControls())[0].setFormat(afmt);
(recordProcessor.getTrackControls())[0].setEnabled(true);
(recordProcessor.getTrackControls())[1].setFormat(vfmt);
(recordProcessor.getTrackControls())[1].setEnabled(true);
```

(5)用 Processor 的输出数据源创建 DataSink，并将数据写到指定格式的文件中。

```
recordProcessor.setContentDescriptor(new
FileTypeDescriptor(FileTypeDescriptor.QUICKTIME));
Control control = recordProcessor.getControl ("javax.media.
control.FrameRateControl");
playhelper.realize(10000);
DataSink dataSink = Manager.createDataSink(recordProcessor.
getDataOutput(), dest);
recordProcessor.start();
dataSink.open(); //打开文件
dataSink.start();
```

(6)调用 stop()和 close()方法，结束数据的捕获。

```
recordProcessor.stop();
recordProcessor.close();
dataSink.stop();
dataSink.close();
```

请注意，虽然已经停止了用于录制电影文件的 Processor，但显示在屏幕上的视频并未受到任何影响，仍然显示。原因是停止的只是新克隆的数据源。

3 结束语

本文通过对相关概念的理解及实例代码说明，详细介绍了利用 JMF 开发具有远程音视频监控及录制功能的多媒体软件的具体实现方法。远程音视频监视及录像功能可应用到我们生活的各个领域，如：网上直播或点播进行远程教学，其与现有的远程教学系统相比很容易实现用户间的实时音视频交互功能；校园数字监控(机房等重要房间安全监控、考场监控等)，小区和家庭远程安全监控，银行数字视频监控，幼儿园远程监控等。用 JMF 开发的多媒体软件进行远程音视频监控与现有的该类监控系统相比有以下几个优点：

(1)成本低。只需购买价廉的网络摄像头若干个，利用现有的 Intranet 或 Internet 即可。不需另外布线、安装设备等工程投资。另外，用 JMF 可以同时从一台电脑上连接的多个网络摄像头中捕获数据，即可同时采集多路视频。

(2)捕获的音视频数据占用空间小，易于网上实时传输或保存到电影文件中。

(3)JMF 实际上是一组 Java 的类库，它包括了各种媒体应用程序接口，用很少的代码就可编写出功能强大的多媒体程序，且由于 Java 的多线程 (multithreading) 机制，可使软件轻松地实现网络上的实时交互行为。

当然，用Java编程的缺点是程序启动时初始化过程长一点，但运行时的速度还是可以的。有数据显示Java程序执行比C++程序执行代码慢 10s~20s，这是Java需要改进的地方。但在现在处理器速度不断提高，Java 新的实时编译器“Just-in-time”和快速虚拟机的推出，二者之间的速度差距已经越来越小。

参考文献

1 Java Media Framework API Guide[EB/OL]. http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html, 1999.

2 Ratner G. Introduction to JMF[EB/OL]. http://java.about.com/library/weekly/uc_jmfmovie1.htm, 2006-02.

3 The Java Developers Almanac 1.4[EB/OL]. http://javaalmanac.com/egs/index.html, 2005-12.

4 JMF 2.1.1 Solutions[EB/OL]. http://java.sun.com/products/java-media/jmf/2.1.1/documentation.html, 2005-12.

5 Eckel B. 侯 捷译. Java 编程思想[M]. 北京: 机械工业出版社, 2001.