Available online at www.sciencedirect.com

SciVerse ScienceDirect

Physics Procedia 25 (2012) 1850 – 1856

Physics Procedia

2012 International Conference on Solid State Devices and Materials Science

Design and Implementation of Telemedicine based on Java Media Framework

Fengguang Xiong, Zhiyan Jia

School of Electronics and Computer Science and Technology
North University of China
Taiyuan Shanxi 030051, China

Abstract

According to analyze the importance and problem of telemedicine in this paper, a telemedicine system based on JMF is proposed to design and implement capturing, compression, storage, transmission, reception and play of a medical audio and video. The telemedicine system can solve existing problems that medical information is not shared, platform-dependent is high, software is incompatibilities and so on. Experimental data prove that the system has low hardware cost, and is easy to transmission and storage, and is portable and powerful.

Key words-Java Media Framework; Telemedicine; Capturing; Transmission; Reception; Storage

1. Introduction

Telemedicine is a new integrated subject that provides medical information and services by using remote communication technology and computer multimedia technology. With the development of information society, it's playing more and more important role in medical diagnostic and therapeutic. The concept of telemedicine system is originally proposed by United States in 1988, which typically includes: remote consultation, information services and distance learning. It's based on computer communication and multimedia technology, and primarily realizes the medical data (including data, text, pictures, and audio-visual information) and remote transmission , storage, query, display and share of the video and audio information[1]. Remote medical consultations establishes a new relation between medical expert and patient, which achieves a remote " face to face" consultation between the expert and patient, between expert and medical personnel. Telemedicine involve in not only medical and clinical issues, but also communications networks, databases and other technology. At present, although there have been some

1875-3892 © 2012 Published by Elsevier B.V. Selection and/or peer-review under responsibility of Garry Lee
Open access under CC BY-NC-ND license. doi:10.1016/j.phpro.2012.03.321

telemedicine systems, all are based on broadband network, and limited heavily on the different platform. Otherwise, there is a terrible problem that data can't be shared and exchanged between various telemedicine. Those problems have hindered seriously the process of telemedicine construction.

2. Java Media Framework

Java Media Framework(JMF)[2] is an optional application programming interface in Java SE, which provides a unified framework for capture, playback, transmission and encoding conversion of audio and video media content. JMF encapsulates the bottom operations, and doesn't depend on the hardware platform and operating system, and has the future of device-independent and platform independence, and has the powerful compatibility of hardware and software, and is able to solve compatibility issues that exist in telemedicine.

Main classes in JMF are shown as follows.

2.1 Capture Device

Capture device is a hardware which is able to capture audio or video, such as microphones, cameras and so on. The information of capture device is stored in CaptureDeviceInfo object. Each device is corresponding to a CaptrueDeviceInfo object. Captured data can be sent to player or processor object for processing.

2.2 DataSource

Datasource contains the media data streams. In JMF, datasource object is the data source, which can be data from the collection device, can also be a multimedia file, and can also be data stream downloaded from the Internet. For datasource object, once you've confirmed its location and type, location of a multimedia information and software information to play this multimedia can both be got. object on the location that contains and software to play the multimedia information. When you create a datasource object, it can be sent to player object, while player don't care how the data source is obtained, and format is. In addition, multiple datasources can be merged into one datasource.

2.3 Player

Player object takes the audio, video data stream as input, and then the audio signal is sent to the speaker, and the video data sent to the same screen, such as player reads data from digital video.

2.4 Processor

Processor interface inherits the player interface. So, it not only has the function of player object, but also can process or interpret media data, and then process outputs media data in a fit format, and the output goal can be local output device, local files or network. In addition, saving streaming media, format conversion, and transmitting stream media can only use processor.

2.5 Format

Format object saves the format information of media. Class format has subclass : AudioFormat and VideoFormat. VideoFormat describes the type of video data such as H.363 and so on, and has six subclass: H361Format, H363Format, IndexedColorFormat, JPEGFormat, RGBFormat and YUVFormat

class [3]. AudioFormat describes the properties of AudioFormats such as sampling frequency, data bits per sample and so on..

2.6 DataSink

DataSink is used to read the media data of datasource and output it to a specific destination. A specific datasink object can output data to a file, or transmit data through the network, or broadcast by RTP. Same as player, datasink need a datasource object as a parameter, which can be constructed by manager.

2.7 Manager

Manager controls how to construct player, processor, datasource object and integrates seamless new application and JMF. JMF provides four Managers: Manager, PackageManager, CaptureDeviceManager and PlugInManager.

3. Design and Implemention of Telemedicine

The function of telemedicine system mainly includes: capturing, compression, storage, transmission, reception and play of audio and video.

3.1 Capturing of audio and video data

Real-time audio and video captured from the patient (or doctors) is sent to the doctor(or the patient) to playback, which mainly involves in capturing, compression, storage, transmission, reception and playback. JMF can be used to achievement easily these five stages.

Stage 1: according to the selected audio and video formats to locate audio and video equipment and take the first device of the list. Codes are shown as follows.

```
Vector audioDevList =CaptureDeviceManager.getDeviceList
(new AudioFormat("linear",44100,16,2));
Vector videoDevlist = CaptureDeviceManager.getDeviceList
(new VideoFormat(VideoFormat.JPEG));
CaptureDeviceInfo audioDI =(CaptureDeviceInfo) audioDev-
List.firstElement();
CaptureDeviceInfo videoDI =(CaptureDeviceInfo) videoDev-
list.firstElement();
```

Stage 2: Obtaining MediaLocator object from the audio and video object and creating audio and video data streams. Codes are shown as follows.

```
DataSource[] ds = new DataSource[2];
ds [0] = Manager.createDataSource(audioDI.getLocator());
ds [1] = Manager.createDataSource(videoDI.getLocator());
```

Stage 3: To be able to play, save and transmit audio and video data at the same time, audio and video data must be encapsulates as a cloneable data source. Codes are shown as follows.

```
ds[0]=Manager.createCloneableDataSource(ds[0]);
ds[1]=Manager.createCloneableDataSource(ds[1]);
DataSource[] clonedDS=new DataSource[2];
clonedDS [0]= (SourceCloneable)ds[0].createClone();
clonedDS [1]= (SourceCloneable)ds[1].createClone();
```

Stage 4: Creating player object using the cloned datasource. Codes are shown as follows.

```
Player audioPlayer=Manager.createPlayer(camDS[0]);
Player videoPlayer=Manager.createPlayer(camDS[1]);
audioPlayer.realize();
audioPlayer.start();
videoPlayer.realize();
videoPlayer.getControlPanelComponent();
videoPlayer.getVisualComponent();
videoPlayer.start();
```

Data flow diagram of capturing audio and video data is shown as figure 1.

3.2 Compression and storage of audio and video data

There are four steps shown as follows.

Step 1: Cloning datasource of audio and video. Codes are shown as follows.

```
DataSource[] clonedDS=new DataSource[2];
clonedDS [0]= (SourceCloneable)ds[0].createClone();
clonedDS [1]= (SourceCloneable)ds[1].createClone();
```

Step 2: Merging audio and video data stream. Codes are shown as follows.

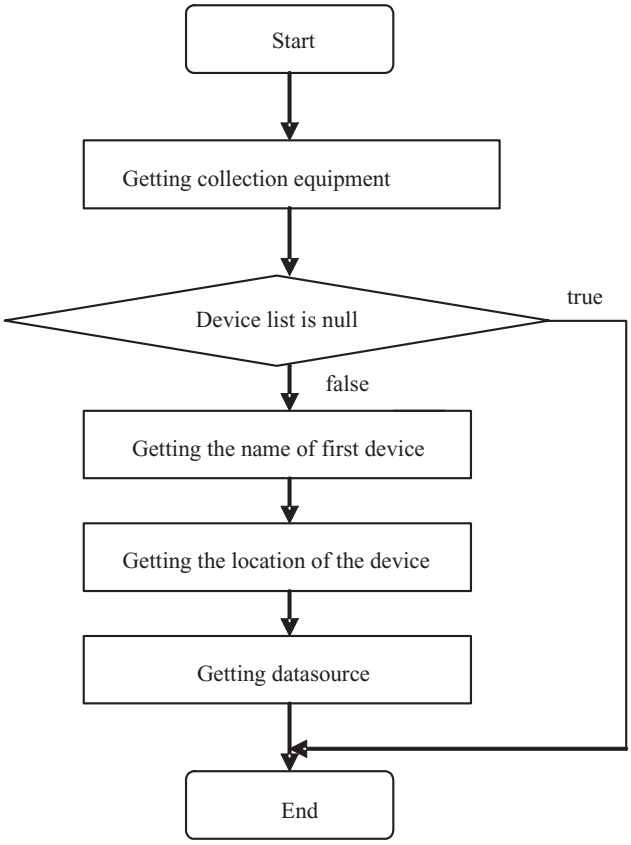

Figure 1 Capturing of audio and video data

```
DataSource mgDS =Manager.createMergingDataSource
```

(clonedDS);

Step 3: creating Processor object by cloned datasource object. Codes are shown as follows.

Processor processor = Manager.createProcessor(clonedDS);

Step 4: Invoking methods of getTrackControls and setFormat on each track. Codes are shown as follows.

```
VideoFormat videoFmt = new VideoFormat(VideoFormat
.JPEG);
AudioFormat audioFmt = new AudioFormat(AudioFormat
.LINEAR);
(processor.getTrackControls())[0].setFormat(audioFmt);
(processor.getTrackControls())[1].setFormat(videoFmt);
```

Step 5: Creating a MediaLocator object used to specify a location to save the file.

```
MediaLocator dest = new MediaLocator ("file:///c:/f1.mov");
```

Step 6: Creating datasink object with datasource outputted by processor object, and writing data to the specified format file. Codes are shown as follows.

```
processor.setContentDescriptor (new FileTypeDescriptor(
FileTypeDescriptor.QUICKTIME));
StreamWriterControl control = processor.getControl
("javax.media.control.FrameRateControl")
control.setStreamSizeLimit(1000000000);
DataSink datasink = Manager.createDataSink(processor
.getDataOutput(), dest);
processor.start();
datasink.open();
datasink.start();
```

3.3 Transmission of audio and video

Transmitting data in the network can use TCP and UDP. TCP is connection-oriented and more reliable, so it is applied to the application which doesn't allow data loss, such as file transfer. The UDP doesn't need to establish connection, so it is more applied to the application which requests faster processing speed, higher reliable data transfer, such as data broadcasting. The transmission of audio and video requires highest real-time, so UDP is more suitable to transmit data. But considering of UDP is not reliable transmission and easy to loss packet, RTP and RTCP based on UDP is most suitable to transmit data. RTP is used to transmit data and RTCP is used to control RTP. The processing steps are shown as follows.

Step 1: Creating a RTPManager object with a static newInstance method of RTPManager.

Step 2: Using the initialize (RTPConnector) method of RTPManager to set the local session's address and port, and destination address and port.

Step 3: Using RTPManagecreateSendStream(DataSource dataSource, int streamIndex) method to create the output stream, and write the encoded signal obtained into the output stream in order to send.

Step 4: Invoking start method of output stream to send data stream.

Data flow diagram of transmitting audio and video is shown as figure 2.

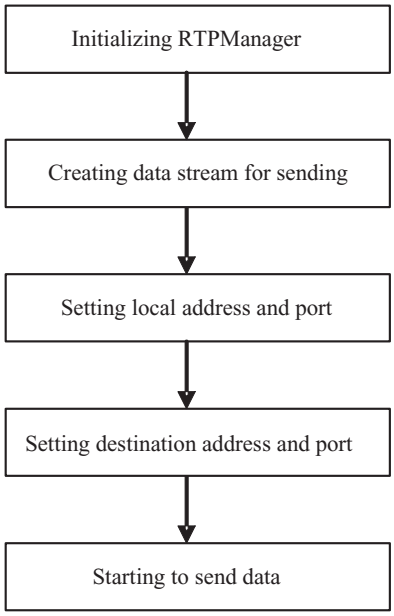

Figure 2 Transmission of audio and video

3.4 Reception and play of audio and video

Reception and play of audio and video uses RTP to receive and play signal, and uses RTCP to manage session. The process steps are shown as follows.

Step 1: Setting RTP manager

- Creating and initializing RTP manager.
- Using addReceiveStreamListener(ReceiveStreamListener) method to add RTP event listener for RTPManager.
- Setting local address and port, and destination address and port.

Step 2: RTP event handling.

- Listener monitors NewReceiveStreamEvent event that the RTPManager receives
- Obtaining input stream by getReceiveStream method.
- Obtaining data source received by getDataSource method of this input stream.
- Creating player by received datasource.
- Adding listener to the player by addControllerListener(ControllerListener) method, and changing the state of listener.
- Instancing a player and changing its state from unrealized to realized.

Step 3: Player event handling.

- Listener monitors RealizeCompleteEvent event receive.
- Getting a player object by getSourceController method of this event.
- Getting a visual component by this player, and judging whether this component is null.
- If component is not null, showing this component and playing audio and video. If component is null, playing direct audio and video。Video signal has a visual component to show video, and audio signal has not visual component.

Data flow diagram of receiving and playing audio and video is shown as figure 3.

4. Conclusion

Telemedicine is based on JMF framework which has feature of cross-platform. Capturing, compression, storage, transmission, reception and play of audio and video is implemented with JFM. Experimental data prove that the system is low cost hardware, easy to transmission and storage, of security and stability, of portability and scalability. Further function is to implement remote education to perfect the system.

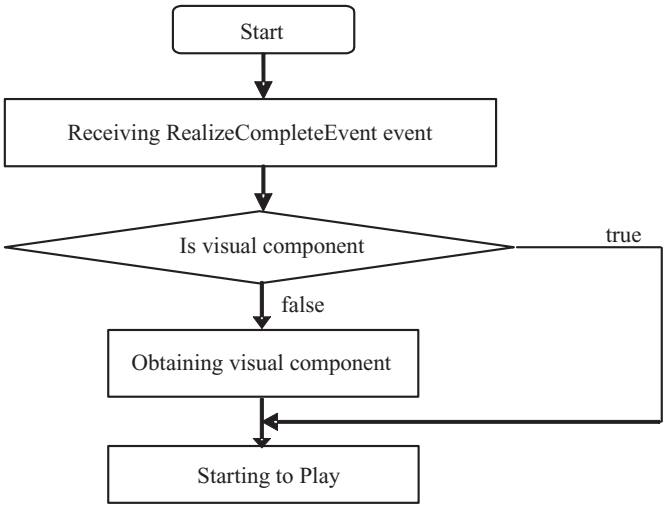

Figure 3. Reception and play of audio and video

References

[1] Peredia D A.Telemedicine Technology and Clinical Applications .JAMA, 1995, 273(6):483.

[2] JMF 3.0 API Guide．http://Java.sun.com/products/Java-media/jmf/3.1.1/specdownload.html.

[3] Christopoulos C, Skodras A, Ebrahimi T. The JPEG2000 Still Image Coding System: An Overview. IEEE Transactions on ConsumerElectronics, 2000, 46(4): 1103-1127.

[4] Helen K.Li. Telemedicine and Ophthalmology. Survey of Ophthalmology, Volume 44, Issue 1, 8 July 1999, Pages 61-72

[5] Takashi Takahashi. The present and future of telemedicine in Japan. International Journal of Medical Informatics, Volume 61, Issues 2-3, May 2001, Pages 131-137.

[6] Chung-Chih Lin, Heng-Shuen Chen, Ching-Yu Chen, Sheng-Mou Hou. Implementation and evaluation of a multifunctional telemedicine system in NTUH. International Journal of Medical Informatics, Volume 61, Issues 2-3, May 2001, Pages 175-187.

[7] Brett C. Meyer, Rema Raman, Marcus R. Chacon, Matt Jensen, Janet D. Werner. Reliability of Site-Independent Telemedicine when Assessed by Telemedicine-Naive Stroke Practitioners. Journal of Stroke and Cerebrovascular Diseases, Volume 17, Issue 4, July-August 2008, Pages 181-186.