高清晰快速图像信息传输的研究与设计

李 勇[1], 赵宏伟[2], 郭剑毅[1]
(1. 昆明理工大学 信自学院,云南 昆明 650051; 2. 云南大学 信电学院,云南 昆明 650091)

摘要: 在图像信息处理系统中,图像信息传输速度慢、图像清晰度低一直是此类系统存在的问题.文中从图像信息传输所采用的协议和压缩技术进行了分析,提出了一种通过改进协议和压缩技术实现图像信息高清晰快速传输的可行方案.并在此基础上给出了用Java技术实现的设计模型.分析表明,文中提出的方案是一种有效、科学的途径.

关键词: 图像信息; 网络传输; Java; 对象序列化
中图分类号: TN919.81 **文献标识码**: A **文章编号**: 1007-855X(2004)05-0060-04

Research and Design of High Definition and Speed Image Information Transforms

LI Yong[1], ZHAO Hong-wei[2], GUO Jian-yi[1]
(1. Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650051, China;
2. Institute of Electronics and Information Science, Yunnan University, Kunming 650091, China)

Abstract: To efficiently resolve the low speed and the low definition of image transforms in network, the protocol and compression technology used in image information transforms at present are analyzed firstly. And then, a feasible solution using new protocol and compression technology is discussed. Finally, according to the solution, an effective method of image information transforms in network with Java serializable is given. The analysticals results show that the scheme presented is an effective approach.

Key words: image information; network transform; java; serializable

0 引 言

随着多媒体技术以及网络技术的发展,通过网络实现图像信息传输的技术已经很成熟,例如视频点播和视频会议,但高清晰度图像以及图像部分属性的实时传输还有待研究,这一问题在X射线扫描图像传输方面尤其突出.虽然不少公司和科研单位对这一问题进行了研究,也取得了一定成果.但通过对目前使用的各种应用图像的网络管理系统的多次实地考察,例如医疗、安全检查等分层管理系统,我们发现图像传输均存在着图像传输速度慢,图像清晰度低的问题,这在很大程度上影响了图像的视觉效果.

本文正是以此为出发点提出了基于TCP/IP协议的网络下通过UDP(User data protocol:用户数据报协议),使用JPEG2000(Joint photographic experts group:图像专家联合小组)无损编码实现图像及其部分属性信息高清晰度、实时传输的可行方案.

1 图像信息传输技术分析

目前图像信息的传输均是基于TCP/IP(Transmission control protocol/Internet protocol:传输控制协议与互连协议)的,图像的编码格式大多采用JPEG标准.那么,要从根本上解决图像传输速度慢、图像清晰度低的问题势必从这两方面入手.

收稿日期: 2004-8-15.
第一作者简介: 李 勇(1971.2~),男,硕士,讲师. 主要研究方向: 电子商务、数据库应用技术. E-mail:lyon-cocoon@hotmail.com.

1.1 利用 UDP 实现图像实时性传输的可行性分析

TCP/IP 在图像传输中广泛使用,这是由于在网络的各种传输协议中,TCP/IP 是目前最为成熟可靠的协议,已经成为远程通信网的标准.其中,TCP 是专门设计用于在不可靠的互联上提供可靠的、端到端的字节流通信的协议,并且能动态满足互联网的要求,足以健壮能面对多种出错.而 IP 层并不能保证将数据报正确地传送到目的端,因此 TCP(实体)需要判断是否超时并根据需要重发数据报[1].TCP 的这种机制虽然能够有效地解决数据报的丢失和可靠传输问题,但在实时图像传输应用中,却成为制约数据报传输速度的客观因素.这是因为从表面上看数据报的丢失或位置错乱是静态的,这是用户可以忍受的,反而是 TCP 要求重传或等待不可预知的数据报所引起的图像数据暂停为用户所不可接受.而 UDP 恰好能弥补 TCP 的这些不足.

UDP 和 TCP 是 TCP/IP 传输层上有两个并列的协议.两者的共同点在于均提供了进程间通信的能力,主要区别在于:首先,UDP 是 TCP 协议的一个替换协议,是不可靠的、无连接的、面向消息的.它没有为 IP 增加可靠性、流量控制或者差错恢复.而 TCP 是面向连接的、可靠的协议,它比 UDP 提供了更多的功能,特别是差错恢复、流量以及可靠性等功能;其次,UDP 省去了 TCP 中建立连接和释放连接过程,取消了重发检验机制.因此 UDP 能够达到较高的速率,通常比 TCP 要快 3 倍.这就为从协议方面解决图像传输的实时性问题提供了依据.

但由于 UDP 不具备诸如接收保证和避免重复等有序投递功能,因此 UDP 中可靠性检测必须由用户想办法解决.参照 TFTP(Trivial File Transfer Protocol:简单文件传输协议)在应用层实现解决 UDP 数据报的丢失和失序问题的方法,并综合上述分析可以得出结论:在运行 TCP/IP 的网络环境下采用 UDP 实现图像信息的实时传输是完全可行的.

1.2 利用 JPEG2000 编码标准实现图像高清晰度传输的可行性分析

目前通过网络传输的图像均采用 JPEG 格式,这是由于 JPEG 常能产生 20:1 甚至更高的压缩比,从理论上来讲它所采用的离散余弦变换是无损的,实际上,使用这种变换需引入一些舍入误差而导致部分信息损失,从而影响图像的清晰度.而作为 JPEG 升级版本的 JPEG2000 具有许多原先的标准所不可比拟的优点:①JPEG2000 压缩比较 JPEG 高约 10%至 30%;②采用小波变换模式,同时支持有损和无损压缩(Lose and loseless compression)两种压缩模式,而小波变换优于离散余弦变换[2];③JPEG2000 能实现渐进传输(Progressive Transmission),即"渐进"性.这是不同于 JPEG 的一个十分重要的特征[3].

由于 JPEG2000 的这些优点,采用 JPEG2000 压缩格式来改进目前图像传输中利用 JPEG 压缩的图像格式,显然不仅从实时性上,而且从清晰度上都能获得较大的改善.因此,采用 JPEG2000 的图像压缩格式提高图像的清晰度是可行的.另外,JPEG2000 另一个重要的特点就是 ROI(Region of Interest:感兴趣区域).这在图像传输中非常重要.若图像中有一小块区域对判读人员是有用的,则对这些区域采用低压缩比;而感兴趣区域之外采用高压缩比,这样的压缩策略为图像传输带来了很大的灵活性.

2 利用对象序列化接口实现有序、快速、高清晰地传输图像信息

通过上述各种技术参数的可行性分析可得到这样的结论:采用 JPEG2000 编码格式和 UDP 协议完全可以实现快速、高清晰传输图像信息,但是需要在应用层解决数据报传输的无序问题.在 Java 技术中,Serializable (对象序列化)是对象永远活着的概念[4].Serializable 是一个接口,实现了该接口的类在网络传输或文件读写时可利用对象序列化来恢复原有的类,它面向那些实现了 Serializable 接口的对象,可将它们转换成一系列字节,并可在以后完全恢复回原来的样子,这意味着序列化机制能自动补偿操作系统间的差异.那么利用这种机制实现图像的在网络上的无损传输显然也就成为可行的.

由于图像信息一般是通过客户端通过网络向服务器端传输,因此除了需要设计 Serializable 接口外还需要设计客户端和服务器端程序.整个设计的基本思路是:首先在客户端获取由摄像头等获取的图像,按 JPEG2000 的格式压缩形成图像对象,然后将图像序列化为对象输出流(对象序列化接口的写操作);由于不能在网络上直接传输对象流,因此需要将对象输出流串接到字节输出流中,然后打包成 UDP 数据报,最

后通过网络传输到服务器端;服务器端接收图像的操作同客户端刚好相反,当服务端接收到对象输入流后,通过对象序列化接口的读操作将该流无损地还原成图像对象,再调用图像显示方法显示该图像即可完成整个过程.

2.1 客户端的构成

客户端主要负责获取图像,并将图像、图像分辨率、给定区域像素值等信息按JPEG2000格式压缩后,再序列化该对象,通过对象输出流和字节输出流(和服务器端的对象输入流和字节输入流一一对应)的串接后(即将对象输出流加载到字节输出流中实现传输),打包成UDP数据报(一般每个包应小于8k字节).客户端主要由下述几个类构成:

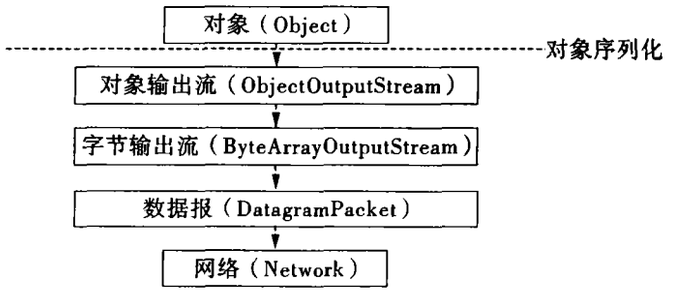

图1 客户端图像传输过程示意图
Fig.1 Process of inage disposod in client

- Client类(客户端主类):继承了JFrame类,用于设计界面,包含main()方法;
- SendThread类:继承了Thread类,是客户端最关键的类,该类用于图像源数据的获取并用数据报传输;
- SaveImage类:用于存储图像,可根据需要把图像保存成JPEG和JPEG2000两种格式.

实现过程参见2.4.

2.2 服务器端的构成

服务器端主要负责接收图像信息,此过程串接了几个流,包括字节流、对象流,并调用线程显示.服务器端主要由下述几个类构成.

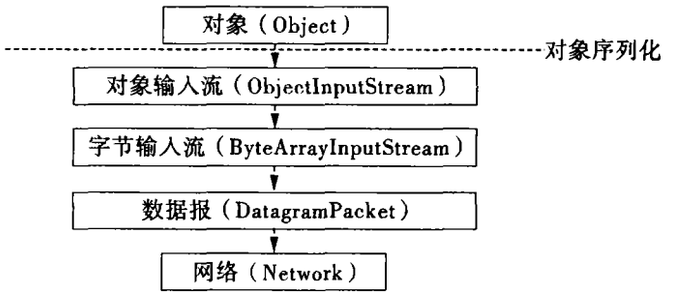

图2 服务器端图像接收过程示意图
Fig.2 Proress of inage clisposed in server

- Server类(服务器端主类):继承了JFrame类,用于设计界面,显示图像等;
- ReceiveThread类:继承了Thread类,是服务器端最关键的类,主要用于接收图像数据.采用线程的目的是为了解决多点到单点的数据传输问题,因为图像网络传播不是简单的点对点结构;
- DrawImage类:继承了Jpanel类,用来显示图像,实现双缓冲,减小图像的抖动.

实现过程参见2.4.

2.3 序列化包的实现

在上述分析基础上,我们开发了一个自定义图像序列化包(SerializableImage.jar).该包包括服务器端和客户端的实现以及一个关键类.由于服务器端和客户端都使用到了该包,因此需要分别在两端引入,即import SerializableImage.*;.其中关键类主要实现了两个方法,一是将压缩后的对象序列化为对象输出流(写方法,由客户端调用);二是将对象输入流还原为对象(读方法,由服务器端调用).下述是写方法的部分构成.

```
private void writeObject(ObjectOutputStream stream) throws java.io.IOException
{
//此方法写 Image 对象到对象输出流
    stream.defaultWriteObject();//stream 为对象输出流,向输出流 stream 写 non - static 和 non - transient 数据
    PixelGrabber grabber = new PixelGrabber(image,0,0, - 1, - 1,true);
```

```
//我们用的是 PixelGrabber 类三个构造函数中的其中一个,image 为 awt 库中 Image 对象,参数 0,0, -1,
-1表示整个 image 默认宽和高的矩形区域,true 表示 forceRGB,即象素总是按 RGB 颜色模式转换
    try{
    grabber.grabPixels();//捕获图像象素,该方法触发 InterruptedException 异常
    }catch(InterruptedException e){}
    Object pix = grabber.getPixels();//象素缓冲器
    Dimension dim = new Dimension(image.getWidth(this),image.getHeight(this));//图像宽和高
    image = null;
    stream.writeObject(dim);//写图像宽和高到输出流
    stream.writeObject(pix);//写图像象素到输出流
...
}
```

与之相对应的方法为 readObject(ObjectInputStream stream)方法,用来读对象输入流来形成图像.

2.4 数据流的串接设计

客户端和服务器端的实现中均涉及到了流的串接.例如,在客户端获取的对象序列化后形成对象流,而网络上只能传输字节流,因此需要实现对象流和字节流之间的串接,即将对象流包装到字节流中进行传输(服务器端相反).通过使用 Java 新增的 ObjectInputStream 和 ObjectOutputStream 两个类所包含的两个 API(ObjectOutputStream 类中的 writeObject(object 0)方法和 ObjectInputStream 类中的 Object readObject()方法),就可以随时获取运行对象的快照,而不管它的对象有多复杂.因为这种快照是通过 ObjectOutputStream 类(OutputStream 类的子类)提供的,所以很容易将它包装在其他输出流中,从而实现所需的任何功能(如 FileOutputStream),实现不同流之间的串接.然后再通过紧密耦合的套接字或 DatagramSocket 类将字节流打包成 UDP 数据报,实现字节流对象的输入/输出,达到利用数据报传输对象的目的.

在流的传接中,首先通过实现对象序列化接口使对象(例如 myObject)可序列化;其次创建字节输出流对象,例如 packetStream;第三用 packetStream 构造一个对象输出流对象,例如 outputStream;第四通过调用 outputStream 的 writeObject()方法将对象 myObject 写入 packetStream 中;第五使用 packetStream 的 toByteArray()方法从 packetStream 中检索字节数组缓冲区;第六使用由第五步检索到的数组缓冲区构造 DatagramPacket,例如 datagramPacket;最后通过调用 DatagramSocket 的 send()方法发送 datagramPacket.

要接收对象时,以逆序完成以上所列各步,用对象输入流代替对象输出流,同时用字节输入流代替字节输出流既可实现.

3 结束语

本文针对目前图像信息传输高速度、高清晰的需求,设计出了利用 UDP 协议和 JPEG2000 的压缩格式,采用对象序列化方法在网络上实时无损多点传输高清晰度图像信息的方法.由于上述设计中采用的是软件压缩,相对于硬件压缩速度有所下降,但在速度允许的条件下大大降低了系统的成本,因此能取得较好的效果.

参考文献:

[1] AndrewS.Tanenbaum 著.计算机网络(第3版),熊桂喜等译[M].北京:清华大学出版社,1998.402~419.

[2] 赵松年,熊小云.子波变换与子波分析[M].北京:电子工业出版社,1997,40~55.

[3] William N.Distributed multimedia applications A review[J].Computer Communitcations,1994,17(2):19~132.

[4] Ell iotte Rusty Harold 著.JAVATM 网络编程(第2版),刘东华等译[M].北京:中国电力出版社,2001.145~150.