基于 J2ME 平台的手机远程监控软件的分析与实现

Analyse and Realization of Remote Watch and Control with J2ME on Mobile Phone

郭春雷 李 祥 (贵阳 贵州大学计算机软件与理论研究所 550025)

摘要:本文讨论了在 Windows 下的 J2ME (Java 2 Mobile Edition)开发环境—J2ME Wireless Toolkit 下的手机(或 PDA 等手持移动设备)远程监控问题,设计并实现了手机对远程计算机的监视、控制。

关键词: J2ME J2SE 远程监视 控制

1 引言

远程控制在众多的领域里有着非常广泛的应用,如远程监控、远程办公、对计算机及网络的远程管理与维护、远程培训与教学等。随着手持移动设备(例如手机)的普及,研究开发基于手持设备的远程监控软件是十分有意义的课题。本文在 J2SE 和 J2ME 的技术下讨论了手机的远程监控技术,实现了从手持设备(例如手机)对远程计算机的监视与控制,例如实时拷贝监视远程计算机的屏幕,发送信息,远程关闭系统等。

2 J2ME 与 J2ME Wireless Toolkit

2.1 J2ME 的发展

J2ME(Java 2 Micro Edition)是 Java 2 的一个组成部分,它与 J2SE、J2EE 并称。根据 Sun 的定义:J2ME 是一种高度优化的 Java 运行环境,主要针对消费类电子设备的,例如蜂窝电话和可视电话、数字机顶盒、汽车导航系统等等。J2ME 技术在 1999 年的 JavaOne Developer Conference 大会上正式推出,它将 Java 语言的与平台无关的特性移植到小型电子设备上,允许移动无线设备之间共享应用程序。

2.2 J2ME 体系结构

J2ME 使用配置和简表定制 Java 运行时环境(JRE)。J2ME 对各种不同设备分类,将其中的一些共性提取出来形成适合于某个范畴中设备可用的规范称为配置(Configuration),包括虚拟机和核心的类库,他们都是一些通用的特性,能在所有平台上通用。J2ME 将某一个行业或领域内设备的特性提取出来,形成简表(Profile),指的是某个行业或某个领域内特定的特性总结,他们不是通用的东西,是针对某一类设备所制定的规范和 API。

2.3 J2ME Wireless Toolkit

J2ME Wireless Toolkit 是用于创建 MIDP(Mobile Information Device Profile)应用程序的工具集。该工具集包含 3 个主要组件:

(1) KToolbar 使创建 MIDP 应用程序时涉及的多项任务能够自动执行。

(2) 仿真器是一部模拟移动电话。用于测试 MIDP 应用程序。

(3) 实用程序集提供了其他有用的功能,包括文本消息传送控制台和加密实用程序。

3 系统设计说明与总体结构

本系统分为服务器端和客户端两部分。客户端运行于小型移动设备(如手机、PDA 等),由 J2ME 实现。服务器端运行于远程计算机,由 J2SE 实现。服务器端运行后处于等待连接状态,由客户端发送服务请求(比如屏幕抓图等),服务器端根据请求完成工作后将结果返回客户端,客户端再将结果显示于手机屏幕。系统的总体结构如图 1 所示。

4 服务器端关键技术的设计与实现

4.1 利用流套接字实现服务器端的通信

基于 java 的特性,用流套接字实现服务器并不困难。大致分为几步:首先创建一个 ServerSocket 对象,

然后调用 ServerSocket 的方法 accept 实现监听。如果有客户端来访问，accept 会返回一个 socket 对象，利用

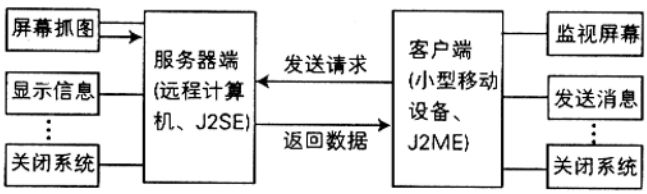

图 1

这个对象就可以很轻松的完成服务器和客户端的数据交换。需要说明的是，为了在监听阶段使程序的其他部分正常运行、减少资源的占用，本文示例中将这部分代码放入线程中运行，另外考虑到手机访问 internet 的带宽和稳定性等实际因素，监听部分处于不断循环等待之中。也即是说，只有客户端有请求时才建立连接，一旦服务完成，立即断开，这样也能有效的节省网络费用。当得到 socket 对象后，建立 BufferedReader 和 PrintWriter 对象实现与客户短信息的接收和发送。完成通信后，调用 ServerSocket 和 socket 对象的 close 关闭套接字，结束通信。其核心代码如下：

```
ServerSocket serverSocket = new ServerSocket(PORT);
Socket clientSocket = serverSocket.accept();
BufferedReader in; PrintWriter out;
in = newBufferedReader(new InputStreamReader(clientSocket.getInputStream()));
out = new PrintWriter(clientSocket.getOutputStream(),true); //接收、发送数据
serverSocket.close(); //关闭 socket,结束通信。
clientSocket.close();
```

4.2 屏幕抓图模块的实现

(1) 抓取屏幕图像。"屏幕的截取"是比较接近操作系统底层的操作。在 Java 中提供了一个 Robot 类，该类用于产生与本地操作系统有关的底层输入，测试应用程序运行或自动控制应用程序运行。Robot 类提供了一个方法：createScreenCapture()，可以直接将全屏幕或某个屏幕区域的像素拷贝到一个 BufferedImage 对象中，我们需要将该对象经过处理后发送至客户端显示以实现屏幕监视。核心代码如下：

```
Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
BufferedImage screenshot = (new Robot()).createScreenCapture(new Rectangle(0,0,(int)d.getWidth(),(int)d.getHeight()));
```

(2) 图像的缩放和旋转处理。当屏幕抓图工作完成后我们得到一个 BufferedImage 对象，在这个对象中存放着同当前屏幕大小、内容相同的图像数据。由于条件的限制(如网络通信的带宽和稳定性限制，或者小型移动设备的处理器的处理能力和屏幕大小的限制)，我们需要对图像进行缩放处理后再发送客户端，以适应手机的屏幕大小和加快通信速度。这样既可以减少网络通信的开销，又可以使手机接收到数据后不需任何处理直接进行屏幕显示即可，减少手持设备的运算。

为此，我们会用到 Java2D API 中的几个包，他们是：javax.imageio.image.BufferedImage、java.awt.image.AffineTransformOp、java.awt.geom.AffineTransform 和 java.awt.image.BufferedImageOp。核心代码如下：

```
int dstwidth = 300; //定义目标图像的宽度,由客户端发送过来的屏幕尺寸数据决定。
int w = screenshot.getWidth(); int h = screenshot.getHeight(); //获取源图像的高和宽
int dsth = (dstw*h)/w; //求得目标图像相应的高度
double sx = (double) dstwidth /w; double sy = (double)dsth/h; //算出转化比率
AffineTransform transform = new AffineTransform();
transform.setToScale(sx,sy);
AffineTransformOp ato = new AffineTransformOp(transform,AffineTransformOp.TYPE_BILINEAR);
BufferedImage dst = new BufferedImage(nw,nh,screenshot.getType());
ato.filter(screenshot,dst);
```

在上面程序中，定义 AffineTransformOp 对象时使用了两个参数，第一个为仿射变化类型(AffineTransform)，第二个参数设置为 TYPE_BILINEAR，即采用双线性插值(Bilinear interpolation)算法(由周围象素点的值，依照双线性函数计算出指定位置点的象素值)计

算。也可以设置为 TYPE_NEAREST_NEIGHBOR，即采用最近邻插值算法，此种算法输出象素的灰度等于离它所映射位置最近的输入象素的灰度值。两种算法相比较，本文示例中采用的是双线性插值算法，虽然速度稍慢，但是得到的图像质量要好的多。最后使用 AffineTranformOp 类的 filter 方法将源图像数据转换并保存在新的 BufferedImage 对象中。

为了适应手机的屏幕大小并且尽可能的不浪费屏幕空间，需要将图像逆时针旋转 90 度。实现方法同缩放基本类似，不再缀述，核心代码如下：

```
AffineTransform afRight = AffineTransform.getRotateInstance(Math.toRadians(90));
afRight.translate(0, bi.getHeight() * -1);
```

（3）如何向客户端发送图像数据。在 JDK 1.4 之后的版本中新加入了 ImageIO 类，我们可以使用它的 write 方法实现将 BufferedImage 对象的数据按照指定的图像格式写入到 OutputStream 对象中。然后利用套接字发送至手机客户端。核心代码如下：

```
OutputStream sos = clientSocket.getOutputStream();
//定义 OutputStream 对象
ImageIO.write(dst,imageFormat,sos); //写入
PrintWriter output = new PrintWriter(clientSocket.getOutputStream(),true); //发送
```

5 客户端关键技术的设计与实现

5.1 J2ME 中使用 socket 技术和线程技术实现手机客户端与服务器的通信

J2ME 的网络连接方式可以按照通信协议分为三种：低级别的 IP 连接（包括套接字、数据报、串口和文件 I/O 通讯），安全连接（为了和基于 WebService 进行安全通讯而提供的额外接口，这些接口由 IP 网络上的 HTTPS 和 SSL/TLS 协议提供）和 HTTP 连接（用于移动设备和 Web 服务器互连）。

本文示例中采用套接字进行通信。Socket 编程时，首先要使用 Connector.open() 方法，打开 Socket，并获得 StreamConnection 对象。Connector.open() 方法需要一个参数，即包含通信协议、网络地址和通信参数的字符串，如：

```
StreamConnection conn = (StreamConnection) Connector.open("socket://www.myserver.com:80");
```

然后需要建立跟该 Socket 相连的输入及输出流：

```
InputStream in = conn.openInputStream();
PrintStream out = new PrintStream(conn.openOutputStream());
```

考虑到硬件和网络通信的局限，小型移动设备在进行网络访问时最好将相应代码放入线程执行，以免因网络阻塞导致程序完全停滞。J2ME 中线程的使用与 J2SE 中区别不大，这里不再赘述。

5.2 网络连接的超时控制

手机通讯时，由于信号时强时弱，时常会出现连接超时，造成无谓的等待。本文中采用 J2ME 中的 Timer 类进行超时控制。当开始进行连接时，同时启动时钟器进行计时，连接成功则将其关闭，如果不能在规定的时限内完成连接，则时钟器开始执行任务，即将本次连接取消并通知用户。

5.3 手机中接收屏幕图像数据并显示的实现

执行屏幕监视功能时，在手机客户端使用 InputStream 类的 read() 方法来接收服务器端发至的数据。实际上它读取输入流后返回的都是整型数据，为此我们定义一个 ByteArrayOutputStream 对象 baos，将 read 返回的整型按字节依次存入 baos 中。传输结束后通过 baos.toByteArray() 得到 Image 的字节类型数组 imageData，这样我们就可以很容易的构建出图片来进行显示。

6 结束语

本文中所述内容笔者都已在所写软件中实现，并且在 J2ME Wireless Toolkit 的模拟器上反复测试，其运行结果正常，能够实现预期的目的。

参考文献

1 James Keogh 著，潘颖、王磊译，《J2ME 开发大全》，清华大学出版社，2004。
2 http://www.j2medev.com/Article/ShowArticle.asp?ArticleID=74，J2ME 中通过 Http 协议传输图片。
3 http://gceclub.sun.com.cn/yuanchuang/week-6/robot.html，用 Java Robot 实现服务器屏幕远程监视。