基于蓝牙的手机文件传输软件①

陈雪林

(绵阳师范学院 数学与计算机科学系，绵阳 621000)

摘 要：针对具有蓝牙配置的手机，运用蓝牙协议栈和 J2ME 蓝牙通信 API 来实现手机文件数据的传输。在分析蓝牙通信流程的基础之上，研究 C/S 模式下 J2ME 蓝牙通信的实现、手机内文件的收发以及图片处理等问题，并对关键实现技术及代码进行详细阐述。

关键词：蓝牙；文件传输；手机

Mobile Phone File Transfer Software Based on Bluetooth

CHEN Xue-Lin

(Department of Mathematics and Computer Science, Mianyang Normal College, Mianyang 621000, China)

Abstract: A file transfer software based on bluetooth mobile phones is presented, which uses bluetooth protocol stack and J2ME bluetooth communication API to achieve data transmission between phones. Based on the analysis of bluetooth communication process, the paper discusses the realization of J2ME bluetooth communication, sending and receiving file in phone, processing image and so on. It also pictures the implementation of technical and detailed description of the code.

Keywords: bluetooth; file transfer; mobile phone

蓝牙(Bluetooth)是由东芝、爱立信、IBM、Intel 和诺基亚等公司等于 1998 年 5 月共同提出的近距离无线数据通信技术标准[1]。它能够在 10 米的半径范围内实现单点对多点的无线数据和声音传输，其数据传输带宽可达到 1Mbps。本文利用蓝牙技术开发一个用于手机文件数据传输的软件，具有即建即连、使用灵活、安全高效等特点，避免传统网络文件传输软件存在的问题。

1 蓝牙通信的关键技术

蓝牙无线电技术基于在工业、科学以及医学(ISM)上公用的 2.45GHz 开放频段，这一频段无需授权并全球通用。当蓝牙设备互相连接时，他们将组成一个微微网(piconet)，即以一个主设备和最大 7 个从设备的形式动态创建网络。其私有化和个性化特征表现得尤为突出。

1.1 蓝牙协议栈

蓝牙协议栈提供了一组的高层协议和 API 以完成发现服务和模拟串行 I/O，还有一个关于包分割和重组的低层协议以及多路技术协议和质量服务。蓝牙协议栈分为硬件和软件两部分，蓝牙硬件协议栈由设备硬件提供，蓝牙软件协议栈则由软件实现[2]。

蓝牙软件协议栈是程序开发中的关键部分，其层次从下至上依次是：宿主控制器接口(Host Controller Interface，HCI) 是蓝牙软件协议栈的最底层，直接和宿主控制器接口固件(Host Controller Interface Firmware)交互。逻辑链路控制和适配协议(Logical Link Control and Adaptation Protocol, L2CAP) 该层负责处理包分割重组，为上层协议提供了有保证的服务。服务发现协议(Service Discovery Protocol，SDP)包含用于发现服务是否有效等操作。RFCOMM 位于 L2CAP 之上，提供了模拟标准串口通信的能力。对象交换协议(Object Exchange Protocol，OBEX)用于实际程序中的对象数据交换。

① 收稿时间:2010-06-30;收到修改稿时间:2010-08-02

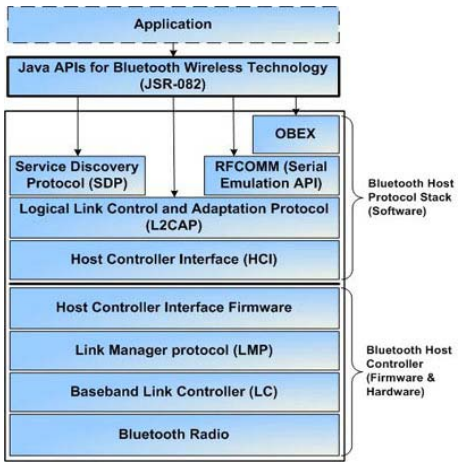

图 1　蓝牙协议栈

1.2 J2ME 对蓝牙的支持

早在 JSR82 规范中就定义了 javax.bluetooth 和 javax.obex 两个包，其中 javax.bluetooth 定义了与蓝牙通信相关的 API，而 javax.obex(Object Exchange Protocol)是建立在串口通信之上，实现以对象为单位的通信。在 javax.bluetooth 中，Java 蓝牙 API 可以被分解为三个部分：发现服务、设备管理和蓝牙通信，其主要类及接口有：本地蓝牙管理器 LocalDevice、远程蓝牙设备 RemoteDevice、搜索代理 DiscoveryAgent、搜索侦听 DiscoveryListener、描述蓝牙服务的特征属性 ServiceRecord 及蓝牙服务属性的类型 DataElement。

1.3 J2ME 平台下蓝牙通信流程

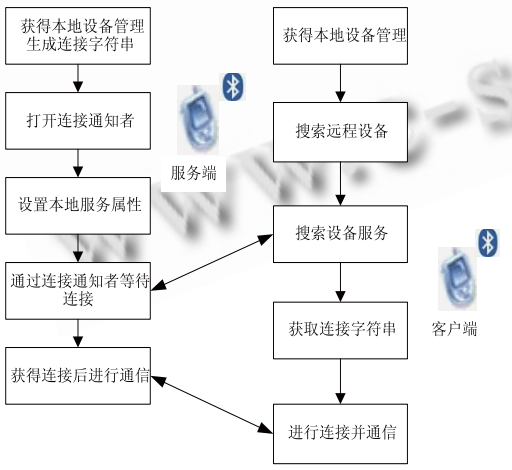

图 2　蓝牙通信流程图

蓝牙通信也是基于通用连接框架，与常见的 C/S 架构类似，只是客户端不知服务端的存在，需要通过无线搜索去发现。搜索到远程设备后，还需要进行服务搜索去发现对方提供了哪些服务[3]。

其中，蓝牙通信是基于通用连接框架，对不同客户端而言，需要通过搜索来获得与服务端的连接信息。蓝牙服务端使用连接通知者对象，用于等待远程设备的连接，类似于阻塞式 socket 服务端，它将一直等待直到接收到客户端的连接请求。对于蓝牙客户端的搜索服务分为设备搜索和服务搜索，后者需要基于指定的远程设备才能进行。客户端和服务器端在获得蓝牙协议连接后，通过连接创建输入/输出流来进行通信。

2　手机文件传输软件的实现

2.1 蓝牙服务端的实现

2.1.1 获得本地设备管理器

获得本地设备管理器会导致系统提示是否需要启动蓝牙服务，该步骤是蓝牙设备通信最基本的初始化。通过 LocalDevice 类的 getLocalDevice 方法即可获取本地设备管理器。

```
try {
    localDevice = LocalDevice.getLocalDevice();
} catch (BluetoothStateException init) {
    init.printStackTrace();
}
```

2.1.2 生成连接字符串

蓝牙通信协议的连接字符串有两种：一种用于串口通信；一种用于蓝牙链路通信(L2CAPConnection)。其中串口通信的连接字符串格式为：btspp://hostname:[CN|UUID];parameters。完整的蓝牙通信链接字符串的构造代码如下：

```
StringBuffer url = new StringBuffer("btspp://");
url.append("localhost").append(':');
url.append(BTConfigure.FILES_SERVER_UUID.toStrin
g() );  //服务 UUID
url.append(";name=FileServer"); //服务名称
url.append(";authorize=false");  //安全参数
```

2.1.3 通过连接字符串获得连接通知者(Notifier)

连接通知者类似阻塞式套接字的侦听过程。该对象只有在接收到远程设备请求时才会返回与该远程设备的连接，否则一直等待下去。

```
StreamConnectionNotifier notifier = null;
try { notifier = (StreamConnectionNotifier)
```

```
Connector.open(url.toString()); //获取流连接通知者
    … … }
```

2.1.4 设置本地设备的服务记录属性

服务器向外界“暴露”本设备可提供的服务信息，客户端才可能获取到这些服务，并向服务端提出请求。以下代码描述了服务端如何设置本地设备的服务记录。

```
    //获取服务记录（用于添加和编辑服务记录）
    record = localDevice.getRecord(notifier);
    //设置服务记录属性（文件名）
            DataElement fileName = new
DataElement(DataElement.STRING, FILE_NAME);

record.setAttributeValue(BTConfigure.FILES_NAMES_
ATTR_ID, fileName);
```

2.1.5 通过 Notifier 循环阻塞等待远程设备的连接

当有远程设备进行连接后，通过连接通知者获得连接对象。当接收到远程客户端设备的连接请求，首先获取到该客户端设备的地址信息，再启动服务线程进行连接的处理。这样对每一个客户端连接启用一个处理线程，可以避免多个客户端排队访问服务端的情形。

```
    while(true) {
    StreamConnection conn = null;
    try { //等待接受客户端的连接
        conn = notifier.acceptAndOpen();
    } catch (IOException e)
    { e.printStackTrace();     continue; }
     RemoteDevice remoteDevice = RemoteDevice.
getRemoteDevice(conn);        //获取远程设备
    //启动服务线程
    new BTServerThread(mainPanel, conn).start();
    }
```

2.1.6 服务端和客户端的通信

蓝牙服务端通信线程是整个服务端程序的核心。每接收到一个客户端的请求，都将创建一个独立的线程来进行处理。通过连接对象创建输入/输出流，实现服务端和客户端的通信。当某一客户端处理完毕，服务端将关闭与该客户端的连接。

```
    public void run()      {
      String fileName=readFileName(); //获取请求
      sendFile(fileName);     //发送答复
      uninit();              //关闭连接
    }
    //从客户端读取文件名
    private String readFileName() {
    String fileName = null;
    try   {   InputStream is = conn.openInputStream();
    int length = is.read();
    byte [] buffer = new byte[length];
    is.read(buffer, 0, length);
    fileName = new String(buffer);   is.close();
                } catch (IOException e) {
     e.printStackTrace(); }
    return (fileName);   }
    //发送文件数据
    private void sendFile(final String fileName) {
    InputStream is = null;   byte [] buffer = null;
    try {
       is = getClass().getResourceAsStream("/" +
fileName);   buffer = new byte[is.available() ];
    is.read(buffer); is.close();
    OutputStream os = conn.openOutputStream();
    os.write(buffer.length >> 8);
    os.write(buffer.length & 0xFF);
    os.write(buffer);      os.flush(); os.close();
    } catch (IOException e) { e.printStackTrace();      }
    }
    //释放连接流
    private void uninit() {
    try { conn.close(); } catch (IOException e)
    { e.printStackTrace(); }
    }
```

2.2 蓝牙客户端的实现

2.2.1 获得本地设备管理器

获得本地设备管理器，设置本地设备管理器的搜索模式，获取搜索代理实例，开始搜索远程设备。

```
    public  void  run() { //设备搜索线程核心
    try {
    LocalDevice localDevice = LocalDevice. getLocal
Device(); //获取本地设备实例
localDevice.setDiscoverable(DiscoveryAgent.GIAC);
    //得到搜索代理
    discoveryAgent = localDevice.getDiscoveryAgent();
```

```
startDiscover();  //开始探索
} catch (BluetoothStateException init) {
init.printStackTrace();  }  }
public  void  startDiscover() { //开始搜索
discoveryDevices.removeAllElements();
try {

discoveryAgent.startInquiry(DiscoveryAgent.GIAC
, this); } catch (BluetoothStateException e)
{ e.printStackTrace(); }  }
```

2.2.2 记录设备搜索结果

在搜索侦听的设备搜索事件中记录设备搜索结果。设备搜索事件是通过实现搜索侦听(DiscoveryListener)的接口来完成回调处理。

```
//搜索到设备
   public  void  deviceDiscovered(RemoteDevice
remoteDevice, DeviceClass deviceClass)  //添加远程设
备
{  discoveryDevices.addElement(remoteDevice);  }
//设备搜索结束
public void inquiryCompleted(int discType) {
switch(discType) {
case  //查询正常结束
DiscoveryListener.INQUIRY_COMPLETED: {
mainPanel.setDevices(discoveryDevices);
  break;  }
case  //查询被取消
DiscoveryListener.INQUIRY_TERMINATED:
break;
case  //查询错误
DiscoveryListener.INQUIRY_ERROR: {
mainPanel.showMsg("Inquiry error!!");
break;  }  }  }
```

2.2.3 记录服务搜索结果

对指定的远程设备搜索其服务，在搜索侦听的服务搜索事件中记录服务搜索结果。服务搜索事件的处理也是通过实现搜索侦听接口来完成回调的。

```
// 服务搜索线程执行代码
public void run() {  … …
try {
            LocalDevice  localDevice  =
LocalDevice.getLocalDevice(); //获取本地设备实例

localDevice.setDiscoverable(DiscoveryAgent.GIAC);
//得到搜索代理
discoveryAgent = localDevice.getDiscoveryAgent();
//开始搜索服务
  discoveryAgent.searchServices(attrSet,  uuidSet,
remoteDevice, this);
} catch (BluetoothStateException init){
   init.printStackTrace();
} }
// 搜索到服务
  public  void  servicesDiscovered(int  transID,
ServiceRecord[] servRecord) { //添加探索到的服务
for(int i = 0; i < servRecord.length; ++i)
 serviceRecords.addElement(servRecord[i]);
}
```

2.2.4 建立与远程设备的连接

通过服务记录来获取连接字符串，并建立与远程设备的连接。客户端通过搜索到的服务记录来获取可供连接的 URL，并与服务端进行连接。

```
// 客户端通信线程核心
public void run() {
//通过服务记录来获取建立连接的 URL
String url = serviceRecord.getConnectionURL(

ServiceRecord.NOAUTHENTICATE_NOENCRYPT,
false);  … …
StreamConnection conn = null;
try  {  // 通过 URL 建立连接
conn = (StreamConnection)Connector.open(url);
        }  catch  (IOException  open)
{ open.printStackTrace(); } }
```

2.2.5 实现服务端和客户端的通信

由连接对象获取输入/输出流，实现服务端和客户端的通信。请求处理完毕，关闭与该远程设备的连接。客户端发起与服务端进行通信并从服务器获取图片。

```
// 发送文件名称请求并下载文件
sendFileName(conn, fileName);
downloadFile(conn, fileName);
try { conn.close(); }
```

```
catch (IOException close) { close.printStackTrace(); }
// 发送文件名请求
private void sendFileName(StreamConnection conn, final String fileName) {
try { OutputStream out = conn.openOutputStream();
out.write(fileName.length() );
out.write(fileName.getBytes() );
out.flush(); out.close();
} catch (IOException write)
{ write.printStackTrace(); } }
// 下载文件
public void downloadFile(StreamConnection conn, final String fileName) {
byte [] buffer = null;
try { InputStream is = conn.openInputStream();
//头两个字节为数据长度
int length = (is.read() << 8); length |= is.read();
buffer = new byte[length]; length = 0;
while (length != buffer.length) {
int n = is.read(buffer, length, buffer.length - length);
if (n == -1) throw new IOException("Can't readdata");
length += n; }
is.close();
} catch (IOException read) { read.printStackTrace(); }
try { Image img = Image.createImage(buffer, 0, buffer.length);
mainPanel.appendImage(img);
} catch (Exception image) {
image.printStackTrace();
} }
```

3 手机文件传输软件的运行与测试

对软件打包后，利用手机安装管理程序将 MIDlet 应用程序从计算机下载到支持蓝牙技术的手机上，然后执行文件传输程序，在此可选择执行客户端或服务器端应用，如图 3 所示。用户选择 BT Server 开启服务器端程序，允许对蓝牙进行连接。在另外一手机上执行该程序，选择 BTClient，进入文件传输的客户端模式，菜单中执行 Search Devices 功能，进行本地蓝牙设备的搜索，结果如图 4 所示。再执行 Search Service 功能，即搜索服务器中提供的服务，选中要下载的文件，执行 Transport 进行文件传输，如图 5 所示。当下载的图像文件传输完毕，将其显示出来，如图 6 所示。

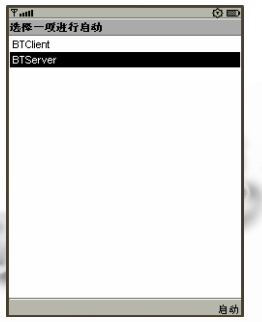

图 3 BTDemo 主界面

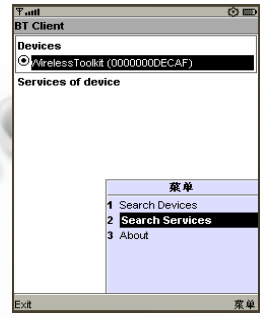

图 4 搜索蓝牙服务

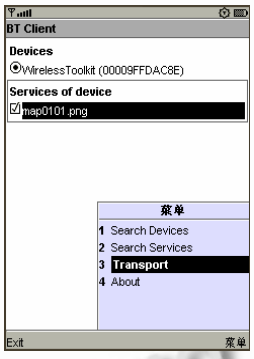

图 5 获取 Server 端资源

图 6 下载图像文件并显示

4 结束语

支持 JAVA 并具备蓝牙功能的手机，给软件业提供了新的机遇。本文开发一种以 J2ME 为平台的蓝牙文件传输软件，可以进行有效的文件传输。为建立微微网中文件服务做了一定的尝试。在一定程度上拓展了手机的功能，具有一定的应用价值。

参考文献

1 俞国红.BlueIM:基于蓝牙的手机即时通信软件.计算机工程,2009,35(17):258－261.
2 孙更新,宾晟,孙海伦.Java ME 手机应用开发大全.北京:科学出版社,2008.272－295.
3 汪永松.J2ME 手机高级编程.北京:机械工业出版社,2009.66－99.