核心类：

UDPMessageListener，该类实现了Runnable接口

广播地址：同一网络中的所有设备都会收到数据包。

BROADCASTIP = "255.255.255.255";

UDP接收和发送的Socket、发送包、接收包

DatagramSocket *UDPSocket*;  
private static DatagramPacket *sendDatagramPacket*;  
private DatagramPacket receiveDatagramPacket;

在run()方法中，写一个while(true){}循环

While(isThreadRunning){

try {  
 UDPSocket.receive(receiveDatagramPacket);  
}  
catch (IOException e) {  
 isThreadRunning = false;  
 receiveDatagramPacket = null;  
 if (*UDPSocket* != null) {  
 *UDPSocket*.close();  
 *UDPSocket* = null;  
 }  
 receiveUDPThread = null;  
 *logger*.e("UDP数据包接收失败！线程停止");  
 e.printStackTrace();  
 break;  
}  
  
if (receiveDatagramPacket.getLength() == 0) {  
 *logger*.e("无法接收UDP数据或者接收到的UDP数据为空");  
 continue;  
}  
  
String UDPListenResStr = "";  
try {  
 UDPListenResStr = new String(*receiveBuffer*, 0, receiveDatagramPacket.getLength(),  
 "gbk");  
}  
catch (UnsupportedEncodingException e) {  
 *logger*.e("系统不支持GBK编码");  
}

}

当开启这个线程的时候，接收数据包。

发送数据包：

public static void sendUDPdata(final IPMSGProtocol ipmsgProtocol, final String targetIP) {  
 *executor*.execute(new Runnable() {  
  
 @Override  
 public void run() {  
 try {  
 InetAddress targetAddr = InetAddress.*getByName*(targetIP); // 目的地址  
 *sendBuffer* = ipmsgProtocol.getProtocolJSON().getBytes("gbk");  
 *sendDatagramPacket* = new DatagramPacket(*sendBuffer*, *sendBuffer*.length,  
 targetAddr, IPMSGConst.*PORT*);  
 *UDPSocket*.send(*sendDatagramPacket*);  
 *logger*.i("数据发送成功");  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 *logger*.e("发送UDP数据包失败");  
 }  
  
 }  
 });  
  
}

TCPService

该类实现了Runnable接口，有一个SaveFileToDisk内部类，SaveFileToDisk继承Thread

在run()方法中，执行下面这个方法

private void scan\_recv() {  
 try {  
 Socket socket = serviceSocket.accept(); // 接收UDP数据报  
 // socket.setSoTimeout(5000); // 设置掉线时间  
 *logger*.d("客户端连接成功");  
  
 SaveFileToDisk fileToDisk = new SaveFileToDisk(socket, filePath);  
 fileToDisk.start();  
  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 *logger*.d("客户端连接失败");  
 SCAN\_FLAG = false;  
 }  
}

SaveFileToDisk是TCPService的内部类，继承了Thread。

run()方法：

public void run() {  
 super.run();  
 *logger*.d("SaveFileToDisk线程开启");  
 if (SCAN\_RECIEVE)  
 recieveFile();  
}

recieveFile()方法如下：

public void recieveFile() {  
 int readSize = 0;  
 FileOutputStream fileOutputStream = null;  
 BufferedOutputStream bufferOutput = null;  
 String strFiledata;  
 String[] strData = null;  
 String fileSavePath;  
  
 try {  
 strFiledata = dataInput.readUTF().toString();  
 strData = strFiledata.split("!");  
 long length = Long.*parseLong*(strData[1]);// 文件大小  
  
 *logger*.d("传输文件类型:" + strData[3]);  
 fileSavePath = savePath + File.*separator* + strData[2] + File.*separator* + strData[0];  
 fileOutputStream = new FileOutputStream(new File(fileSavePath));// 创建文件流  
 *logger*.d("文件存储路径:" + fileSavePath);  
 FileState fileState = new FileState(length, 0, fileSavePath, getType(strData[3]));  
 BaseApplication.*recieveFileStates*.put(fileSavePath, fileState);  
 FileState fs = BaseApplication.*recieveFileStates*.get(fileSavePath);  
 bufferOutput = new BufferedOutputStream(fileOutputStream);// 创建带缓冲区的文件流  
 long currentLength = 0;  
 int count = 0;  
 while (-1 != (readSize = dataInput.read(mBuffer))) {  
 bufferOutput.write(mBuffer, 0, readSize);  
 currentLength += readSize;  
 count++;  
 if (count % 10 == 0) {  
 //long Length = currentLength - lastLength;  
 fs.currentSize = currentLength;  
 fs.percent = (int) ((float) currentLength / (float) length \* 100);  
  
 switch (fs.type) {  
 case *IMAGE*:  
 break;  
  
 case *VOICE*:  
 break;  
  
 case *FILE*:  
 android.os.Message msg = *mHandler*.obtainMessage();  
 msg.obj = fs;  
 msg.sendToTarget();  
 break;  
  
 default:  
 break;  
 }  
 }  
 }  
  
 // 将byte数组的数据写进指定路径  
 bufferOutput.flush();  
  
 input.close();  
 dataInput.close();  
 bufferOutput.close();  
 fileOutputStream.close();  
  
 switch (fs.type) {  
 case *IMAGE*:  
 break;  
  
 case *VOICE*:  
 break;  
  
 case *FILE*:  
 android.os.Message msg = *mHandler*.obtainMessage();  
 fs.percent = 100;  
 msg.obj = fs;  
 msg.sendToTarget();  
 break;  
  
 default:  
 break;  
 }  
  
 BaseApplication.*recieveFileStates*.remove(fs.fileName);  
 }  
 catch (IOException e) {  
 // *TODO Auto-generated catch block  
 logger*.d("写入文件失败");  
 e.printStackTrace();  
 }  
}

TcpClient类，实现Runnable接口，拥有一个继承Thread的SendFileThread

Run()方法：

处理信息：

public void processMessage(int commandNo, IPMSGProtocol ipmsgRes, String senderIMEI,  
 String senderIp) {  
 TcpService tcpService;  
 switch (commandNo) {  
  
 // 收到上线数据包，添加用户，并回送IPMSG\_ANSENTRY应答。  
 case IPMSGConst.*IPMSG\_BR\_ENTRY*: {  
 *logger*.i("收到上线通知");  
 addUser(ipmsgRes);  
 *sendUDPdata*(IPMSGConst.*IPMSG\_ANSENTRY*, receiveDatagramPacket.getAddress(),  
 mLocalUser);  
 *logger*.i("成功发送上线应答");  
 }  
 break;  
  
 // 收到上线应答，更新在线用户列表  
 case IPMSGConst.*IPMSG\_ANSENTRY*: {  
 *logger*.i("收到上线应答");  
 addUser(ipmsgRes);  
 }  
 break;  
  
 // 收到下线广播  
 case IPMSGConst.*IPMSG\_BR\_EXIT*: {  
 removeOnlineUser(senderIMEI, 1);  
 *logger*.i("成功删除imei为" + senderIMEI + "的用户");  
 }  
 break;  
  
 case IPMSGConst.*IPMSG\_REQUEST\_IMAGE\_DATA*:  
 *logger*.i("收到IMAGE发送请求");  
  
 tcpService = TcpService.*getInstance*(*mContext*);  
 tcpService.setSavePath(BaseApplication.*IMAG\_PATH*);  
 tcpService.startReceive();  
 *sendUDPdata*(IPMSGConst.*IPMSG\_CONFIRM\_IMAGE\_DATA*, senderIp);  
 break;  
  
 case IPMSGConst.*IPMSG\_REQUEST\_VOICE\_DATA*:  
 *logger*.i("收到VOICE发送请求");  
  
 tcpService = TcpService.*getInstance*(*mContext*);  
 tcpService.setSavePath(BaseApplication.*VOICE\_PATH*);  
 tcpService.startReceive();  
 *sendUDPdata*(IPMSGConst.*IPMSG\_CONFIRM\_VOICE\_DATA*, senderIp);  
 break;  
  
 case IPMSGConst.*IPMSG\_SENDMSG*: {  
 *logger*.i("收到MSG消息");  
 Message msg = (Message) ipmsgRes.getAddObject();  
  
 switch (msg.getContentType()) {  
 case *TEXT*:  
 *sendUDPdata*(IPMSGConst.*IPMSG\_RECVMSG*, senderIp, ipmsgRes.getPacketNo());  
 break;  
  
 case *IMAGE*:  
 *logger*.i("收到图片信息");  
 msg.setMsgContent(BaseApplication.*IMAG\_PATH* + File.*separator* + msg.getSenderIMEI() + File.*separator* + msg.getMsgContent());  
 String THUMBNAIL\_PATH = BaseApplication.*THUMBNAIL\_PATH* + File.*separator* + msg.getSenderIMEI();  
  
 *logger*.d("缩略图文件夹路径:" + THUMBNAIL\_PATH);  
 *logger*.d("图片文件路径:" + msg.getMsgContent());  
  
 ImageUtils.*createThumbnail*(*mContext*, msg.getMsgContent(), THUMBNAIL\_PATH  
 + File.*separator*);  
 break;  
  
 case *VOICE*:  
 *logger*.i("收到录音信息");  
 msg.setMsgContent(BaseApplication.*VOICE\_PATH* + File.*separator* + msg.getSenderIMEI() + File.*separator* + msg.getMsgContent());  
 *logger*.d("文件路径:" + msg.getMsgContent());  
 break;  
  
 case *FILE*:  
 *logger*.i("收到文件 发送请求");  
 tcpService = TcpService.*getInstance*(*mContext*);  
 tcpService.setSavePath(BaseApplication.*FILE\_PATH*);  
 tcpService.startReceive();  
 *sendUDPdata*(IPMSGConst.*IPMSG\_CONFIRM\_FILE\_DATA*, senderIp);  
 msg.setMsgContent(BaseApplication.*FILE\_PATH* + File.*separator* + msg.getSenderIMEI() + File.*separator* + msg.getMsgContent());  
 *logger*.d("文件路径:" + msg.getMsgContent());  
 break;  
 }  
  
 // 加入数据库  
 mDBOperate.addChattingInfo(senderIMEI, SessionUtils.*getIMEI*(), msg.getSendTime(),  
 msg.getMsgContent(), msg.getContentType());  
  
 // 加入未读消息列表  
 android.os.Message pMessage = new android.os.Message();  
 pMessage.what = commandNo;  
 pMessage.obj = msg;  
  
 ChatActivity v = ActivitiesManager.*getChatActivity*();  
 if (v == null) {  
 addUnReadPeople(getOnlineUser(senderIMEI)); // 添加到未读用户列表  
 for (int i = 0; i < mListenerList.size(); i++) {  
 android.os.Message pMsg = new android.os.Message();  
 pMsg.what = IPMSGConst.*IPMSG\_RECVMSG*;  
 mListenerList.get(i).processMessage(pMsg);  
 }  
 }  
 else {  
 v.processMessage(pMessage);  
 }  
  
 addLastMsgCache(senderIMEI, msg); // 添加到消息缓存  
 BaseApplication.*playNotification*();  
  
 }  
 break;  
  
 default:  
 *logger*.i("收到命令：" + commandNo);  
  
 android.os.Message pMessage = new android.os.Message();  
 pMessage.what = commandNo;  
  
 ChatActivity v = ActivitiesManager.*getChatActivity*();  
 if (v != null) {  
 v.processMessage(pMessage);  
 }  
  
 break;  
  
 } // End of switch  
}