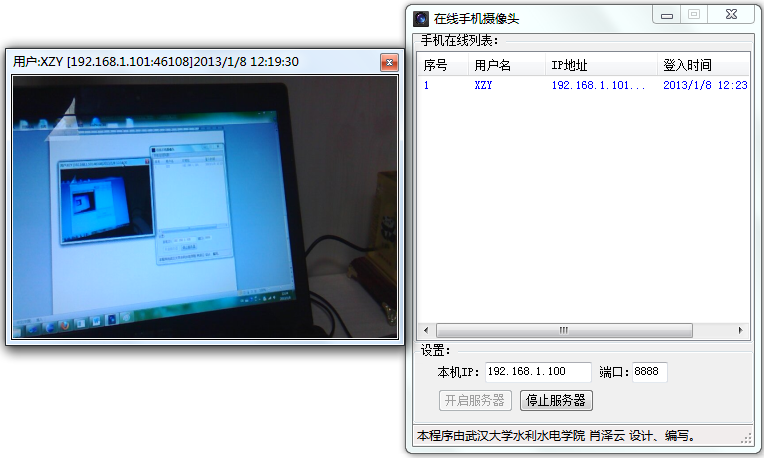
基于Socket的Android手机视频实时传输

肖泽云

首先，简单介绍一下原理。主要是在手机客户端（Android）通过实现Camera.PreviewCallback接口，在其onPreviewFrame重载函数里面获取摄像头当前图像数据，然后通过Socket将图像数据和相关的用户名、命令等数据传输到服务器程序中。服务器端（PC端）采用C#编写，通过监听相应的端口，在获取数据后进行相应的命令解析和图像数据还原，然后将图像数据传递至PictureBox控件中用于显示，这样就实现了手机摄像头的视频数据实时传输到服务器上。如果需要将这些视频进行转发，通过服务器再将这些数据复制转发即可。效果如下：



对于Android客户端上主要有几个地方需要注意，第一个就是Socket通信。Socket通信可以通过Socket类来实现，直接结合PrintWriter来写入命令，如下定义的一个专门用于发送命令的线程类，当要连接到服务器和与服务器断开时，都需要发送命令通知服务器，此外在进行其他文字传输时也可以采用该方法，具体代码如下：

/\*\*发送命令线程\*/

**class** MySendCommondThread **extends** Thread{

**private** String commond;

**public** MySendCommondThread(String commond){

**this**.commond=commond;

}

**public** **void** run(){

//实例化Socket

**try** {

Socket socket=**new** Socket(serverUrl,serverPort);

PrintWriter out = **new** PrintWriter(socket.getOutputStream());

out.println(commond);

out.flush();

} **catch** (UnknownHostException e) {

} **catch** (IOException e) {

}

}

}

如果是采用Socket发送文件，则可以通过OutputStream将ByteArrayInputStream数据流读入，而文件数据流则转换为ByteArrayOutputStream。如果需要在前面添加文字，同样也需要转换为byte，然后写入OutputStream。同样也可以通过定义一个线程类发送文件，如下：

/\*\*发送文件线程\*/

**class** MySendFileThread **extends** Thread{

**private** String username;

**private** String ipname;

**private** **int** port;

**private** **byte** byteBuffer[] = **new** **byte**[1024];

**private** OutputStream outsocket;

**private** ByteArrayOutputStream myoutputstream;

**public** MySendFileThread(ByteArrayOutputStream myoutputstream,String username,String ipname,**int** port){

**this**.myoutputstream = myoutputstream;

**this**.username=username;

**this**.ipname = ipname;

**this**.port=port;

**try** {

myoutputstream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** run() {

**try**{

//将图像数据通过Socket发送出去

Socket tempSocket = **new** Socket(ipname, port);

outsocket = tempSocket.getOutputStream();

//写入头部数据信息

String msg=java.net.URLEncoder.*encode*("PHONEVIDEO|"+username+"|","utf-8");

**byte**[] buffer= msg.getBytes();

outsocket.write(buffer);

ByteArrayInputStream inputstream = **new** ByteArrayInputStream(myoutputstream.toByteArray());

**int** amount;

**while** ((amount = inputstream.read(byteBuffer)) != -1) {

outsocket.write(byteBuffer, 0, amount);

}

myoutputstream.flush();

myoutputstream.close();

tempSocket.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

而获取摄像头当前图像的关键在于onPreviewFrame()重载函数里面，该函数里面有两个参数，第一个参数为byte[]，为摄像头当前图像数据，通过YuvImage可以将该数据转换为图片文件，同时还可用对该图片进行压缩和裁剪，将图片进行压缩转换后转换为 ByteArrayOutputStream数据，即前面发送文件线程类中所需的文件数据，然后采用线程发送文件，如下代码：

@Override

**public** **void** onPreviewFrame(**byte**[] data, Camera camera) {

// **TODO** Auto-generated method stub

//如果没有指令传输视频，就先不传

**if**(!startSendVideo)

**return**;

**if**(tempPreRate<VideoPreRate){

tempPreRate++;

**return**;

}

tempPreRate=0;

**try** {

**if**(data!=**null**)

{

YuvImage image = **new** YuvImage(data,VideoFormatIndex, VideoWidth, VideoHeight,**null**);

**if**(image!=**null**)

{

ByteArrayOutputStream outstream = **new** ByteArrayOutputStream();

//在此设置图片的尺寸和质量

image.compressToJpeg(**new** Rect(0, 0, (**int**)(VideoWidthRatio\*VideoWidth),

(**int**)(VideoHeightRatio\*VideoHeight)), VideoQuality, outstream);

outstream.flush();

//启用线程将图像数据发送出去

Thread th = **new** MySendFileThread(outstream,pUsername,serverUrl,serverPort);

th.start();

}

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

值得注意的是，在调试中YuvImage可能找不到，在模拟机上无法执行该过程，但是编译后在真机中可以通过。此外，以上传输文字字符都是采用UTF编码，在服务器端接收时进行解析时需要采用对应的编码进行解析，否则可能会出现错误解析。

Android客户端中关键的部分主要就这些，新建一个Android项目（项目名称为SocketCamera），在main布局中添加一个SurfaceView和两个按钮，如下图所示：



然后在SocketCameraActivity.java中添加代码，具体如下：

**package** com.xzy;

**import** java.io.ByteArrayInputStream;

**import** java.io.ByteArrayOutputStream;

**import** java.io.IOException;

**import** java.io.OutputStream;

**import** java.io.PrintWriter;

**import** java.net.Socket;

**import** java.net.UnknownHostException;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.content.DialogInterface;

**import** android.content.Intent;

**import** android.content.SharedPreferences;

**import** android.graphics.Rect;

**import** android.graphics.YuvImage;

**import** android.hardware.Camera;

**import** android.hardware.Camera.Size;

**import** android.os.Bundle;

**import** android.preference.PreferenceManager;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.SurfaceHolder;

**import** android.view.SurfaceView;

**import** android.view.View;

**import** android.view.WindowManager;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

**public** **class** SocketCameraActivity **extends** Activity **implements** SurfaceHolder.Callback,

Camera.PreviewCallback{

**private** SurfaceView mSurfaceview = **null**; // SurfaceView对象：(视图组件)视频显示

**private** SurfaceHolder mSurfaceHolder = **null**; // SurfaceHolder对象：(抽象接口)SurfaceView支持类

**private** Camera mCamera = **null**; // Camera对象，相机预览

/\*\*服务器地址\*/

**private** String pUsername="XZY";

/\*\*服务器地址\*/

**private** String serverUrl="192.168.1.100";

/\*\*服务器端口\*/

**private** **int** serverPort=8888;

/\*\*视频刷新间隔\*/

**private** **int** VideoPreRate=1;

/\*\*当前视频序号\*/

**private** **int** tempPreRate=0;

/\*\*视频质量\*/

**private** **int** VideoQuality=85;

/\*\*发送视频宽度比例\*/

**private** **float** VideoWidthRatio=1;

/\*\*发送视频高度比例\*/

**private** **float** VideoHeightRatio=1;

/\*\*发送视频宽度\*/

**private** **int** VideoWidth=320;

/\*\*发送视频高度\*/

**private** **int** VideoHeight=240;

/\*\*视频格式索引\*/

**private** **int** VideoFormatIndex=0;

/\*\*是否发送视频\*/

**private** **boolean** startSendVideo=**false**;

/\*\*是否连接主机\*/

**private** **boolean** connectedServer=**false**;

**private** Button myBtn01, myBtn02;

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

//禁止屏幕休眠 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*,

WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*);

mSurfaceview = (SurfaceView) findViewById(R.id.*camera\_preview*);

myBtn01=(Button)findViewById(R.id.*button1*);

myBtn02=(Button)findViewById(R.id.*button2*);

//开始连接主机按钮

myBtn01.setOnClickListener(**new** OnClickListener(){

**public** **void** onClick(View v) {

//Common.SetGPSConnected(LoginActivity.this, false);

**if**(connectedServer){//停止连接主机，同时断开传输

startSendVideo=**false**;

connectedServer=**false**;

myBtn02.setEnabled(**false**);

myBtn01.setText("开始连接");

myBtn02.setText("开始传输");

//断开连接

Thread th = **new** MySendCommondThread("PHONEDISCONNECT|"+pUsername+"|");

th.start();

}

**else**//连接主机

{

//启用线程发送命令PHONECONNECT

Thread th = **new** MySendCommondThread("PHONECONNECT|"+pUsername+"|");

th.start();

connectedServer=**true**;

myBtn02.setEnabled(**true**);

myBtn01.setText("停止连接");

}

}});

myBtn02.setEnabled(**false**);

myBtn02.setOnClickListener(**new** OnClickListener(){

**public** **void** onClick(View v) {

**if**(startSendVideo)//停止传输视频

{

startSendVideo=**false**;

myBtn02.setText("开始传输");

}

**else**{ // 开始传输视频

startSendVideo=**true**;

myBtn02.setText("停止传输");

}

}});

}

@Override

**public** **void** onStart()//重新启动的时候

{

mSurfaceHolder = mSurfaceview.getHolder(); // 绑定SurfaceView，取得SurfaceHolder对象

mSurfaceHolder.addCallback(**this**); // SurfaceHolder加入回调接口

mSurfaceHolder.setType(SurfaceHolder.*SURFACE\_TYPE\_PUSH\_BUFFERS*);// 设置显示器类型，setType必须设置

//读取配置文件

SharedPreferences preParas = PreferenceManager.*getDefaultSharedPreferences*(SocketCameraActivity.**this**);

pUsername=preParas.getString("Username", "XZY");

serverUrl=preParas.getString("ServerUrl", "192.168.0.100");

String tempStr=preParas.getString("ServerPort", "8888");

serverPort=Integer.*parseInt*(tempStr);

tempStr=preParas.getString("VideoPreRate", "1");

VideoPreRate=Integer.*parseInt*(tempStr);

tempStr=preParas.getString("VideoQuality", "85");

VideoQuality=Integer.*parseInt*(tempStr);

tempStr=preParas.getString("VideoWidthRatio", "100");

VideoWidthRatio=Integer.*parseInt*(tempStr);

tempStr=preParas.getString("VideoHeightRatio", "100");

VideoHeightRatio=Integer.*parseInt*(tempStr);

VideoWidthRatio=VideoWidthRatio/100f;

VideoHeightRatio=VideoHeightRatio/100f;

**super**.onStart();

}

@Override

**protected** **void** onResume() {

**super**.onResume();

InitCamera();

}

/\*\*初始化摄像头\*/

**private** **void** InitCamera(){

**try**{

mCamera = Camera.*open*();

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Override

**protected** **void** onPause() {

**super**.onPause();

**try**{

**if** (mCamera != **null**) {

mCamera.setPreviewCallback(**null**); // ！！这个必须在前，不然退出出错

mCamera.stopPreview();

mCamera.release();

mCamera = **null**;

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Override

**public** **void** surfaceChanged(SurfaceHolder arg0, **int** arg1, **int** arg2, **int** arg3) {

// **TODO** Auto-generated method stub

**if** (mCamera == **null**) {

**return**;

}

mCamera.stopPreview();

mCamera.setPreviewCallback(**this**);

mCamera.setDisplayOrientation(90); //设置横行录制

//获取摄像头参数

Camera.Parameters parameters = mCamera.getParameters();

Size size = parameters.getPreviewSize();

VideoWidth=size.width;

VideoHeight=size.height;

VideoFormatIndex=parameters.getPreviewFormat();

mCamera.startPreview();

}

@Override

**public** **void** surfaceCreated(SurfaceHolder holder) {

// **TODO** Auto-generated method stub

**try** {

**if** (mCamera != **null**) {

mCamera.setPreviewDisplay(mSurfaceHolder);

mCamera.startPreview();

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

@Override

**public** **void** surfaceDestroyed(SurfaceHolder holder) {

// **TODO** Auto-generated method stub

**if** (**null** != mCamera) {

mCamera.setPreviewCallback(**null**); // ！！这个必须在前，不然退出出错

mCamera.stopPreview();

mCamera.release();

mCamera = **null**;

}

}

@Override

**public** **void** onPreviewFrame(**byte**[] data, Camera camera) {

// **TODO** Auto-generated method stub

//如果没有指令传输视频，就先不传

**if**(!startSendVideo)

**return**;

**if**(tempPreRate<VideoPreRate){

tempPreRate++;

**return**;

}

tempPreRate=0;

**try** {

**if**(data!=**null**)

{

YuvImage image = **new** YuvImage(data,VideoFormatIndex, VideoWidth, VideoHeight,**null**);

**if**(image!=**null**)

{

ByteArrayOutputStream outstream = **new** ByteArrayOutputStream();

//在此设置图片的尺寸和质量

image.compressToJpeg(**new** Rect(0, 0, (**int**)(VideoWidthRatio\*VideoWidth),

(**int**)(VideoHeightRatio\*VideoHeight)), VideoQuality, outstream);

outstream.flush();

//启用线程将图像数据发送出去

Thread th = **new** MySendFileThread(outstream,pUsername,serverUrl,serverPort);

th.start();

}

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

/\*\*创建菜单\*/

**public** **boolean** onCreateOptionsMenu(Menu menu)

{

menu.add(0,0,0,"系统设置");

menu.add(0,1,1,"关于程序");

menu.add(0,2,2,"退出程序");

**return** **super**.onCreateOptionsMenu(menu);

}

/\*\*菜单选中时发生的相应事件\*/

**public** **boolean** onOptionsItemSelected(MenuItem item)

{

**super**.onOptionsItemSelected(item);//获取菜单

**switch**(item.getItemId())//菜单序号

{

**case** 0:

//系统设置

{

Intent intent=**new** Intent(**this**,SettingActivity.**class**);

startActivity(intent);

}

**break**;

**case** 1://关于程序

{

**new** AlertDialog.Builder(**this**)

.setTitle("关于本程序")

.setMessage("本程序由武汉大学水利水电学院肖泽云设计、编写。\nEmail：xwebsite@163.com")

.setPositiveButton

(

"我知道了",

**new** DialogInterface.OnClickListener()

{

@Override

**public** **void** onClick(DialogInterface dialog, **int** which)

{

}

}

)

.show();

}

**break**;

**case** 2://退出程序

{

//杀掉线程强制退出

android.os.Process.*killProcess*(android.os.Process.*myPid*());

}

**break**;

}

**return** **true**;

}

/\*\*发送命令线程\*/

**class** MySendCommondThread **extends** Thread{

**private** String commond;

**public** MySendCommondThread(String commond){

**this**.commond=commond;

}

**public** **void** run(){

//实例化Socket

**try** {

Socket socket=**new** Socket(serverUrl,serverPort);

PrintWriter out = **new** PrintWriter(socket.getOutputStream());

out.println(commond);

out.flush();

} **catch** (UnknownHostException e) {

} **catch** (IOException e) {

}

}

}

/\*\*发送文件线程\*/

**class** MySendFileThread **extends** Thread{

**private** String username;

**private** String ipname;

**private** **int** port;

**private** **byte** byteBuffer[] = **new** **byte**[1024];

**private** OutputStream outsocket;

**private** ByteArrayOutputStream myoutputstream;

**public** MySendFileThread(ByteArrayOutputStream myoutputstream,String username,String ipname,**int** port){

**this**.myoutputstream = myoutputstream;

**this**.username=username;

**this**.ipname = ipname;

**this**.port=port;

**try** {

myoutputstream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** run() {

**try**{

//将图像数据通过Socket发送出去

Socket tempSocket = **new** Socket(ipname, port);

outsocket = tempSocket.getOutputStream();

//写入头部数据信息

String msg=java.net.URLEncoder.*encode*("PHONEVIDEO|"+username+"|","utf-8");

**byte**[] buffer= msg.getBytes();

outsocket.write(buffer);

ByteArrayInputStream inputstream = **new** ByteArrayInputStream(myoutputstream.toByteArray());

**int** amount;

**while** ((amount = inputstream.read(byteBuffer)) != -1) {

outsocket.write(byteBuffer, 0, amount);

}

myoutputstream.flush();

myoutputstream.close();

tempSocket.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

此外还有一些参数，在res/xml新建一个setting.xml文件，添加服务器地址、端口、用户名等参数设置，如下：

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<PreferenceScreen

xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<PreferenceCategory android:title=*"服务器设置"*>

<EditTextPreference

android:key=*"Username"*

android:title=*"用户名"*

android:summary=*"用于连接服务器的用户名"*

android:defaultValue=*"XZY"*/>

<EditTextPreference

android:key=*"ServerUrl"*

android:title=*"视频服务器地址"*

android:summary=*"保存服务器地址"*

android:defaultValue=*"192.168.1.100"*/>

<EditTextPreference

android:key=*"ServerPort"*

android:title=*"服务器端口"*

android:summary=*"连接服务器的端口地址"*

android:defaultValue=*"8888"*/>

</PreferenceCategory>

<PreferenceCategory android:title=*"视频设置"*>

<EditTextPreference

android:key=*"VideoPreRate"*

android:title=*"视频刷新间隔"*

android:summary=*"设置视频刷新的间隔值，应大于等于0，值越大视频传输间隔越长"*

android:defaultValue=*"1"*/>

<EditTextPreference

android:key=*"VideoQuality"*

android:title=*"图像质量"*

android:summary=*"设置图像压缩的质量，值为0~100，值越高越清晰，但同时数据也更大"*

android:defaultValue=*"85"*/>

<EditTextPreference

android:key=*"VideoWidthRatio"*

android:title=*"图像宽度缩放比例"*

android:summary=*"设置图像的宽度缩放比例，值为0~100，值越高图像分辨率越高"*

android:defaultValue=*"100"*/>

<EditTextPreference

android:key=*"VideoHeightRatio"*

android:title=*"图像高度缩放比例"*

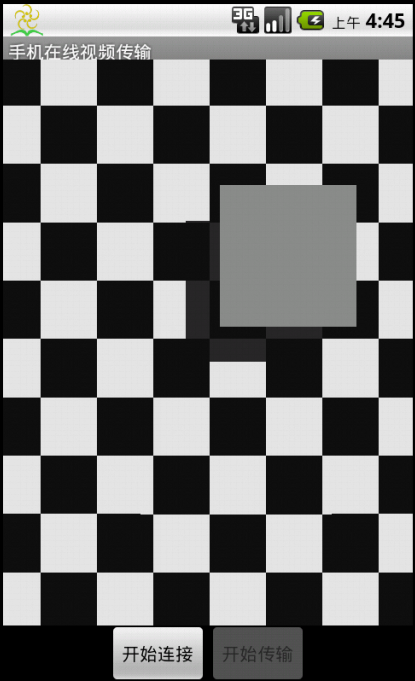
android:summary=*"设置图像的高度缩放比例，值为0~100，值越高图像分辨率越高"*

android:defaultValue=*"100"*/>

</PreferenceCategory>

</PreferenceScreen>

编译程序，在模拟机上效果如下：



接下来就是服务器端接收手机传输的视频数据，这与一般CS架构中服务器程序类似，主要是监听端口，然后解析数据。现新建一个C#应用程序项目（项目名称为“手机摄像头”），首先定义一些全局变量，主要包括服务器地址、端口以及相关监听对象等，如下：

/// <summary>

/// 服务器状态，如果为false表示服务器暂停,true表示服务器开启

/// </summary>

public bool ServerStatus = false;

/// <summary>

/// 服务器地址

/// </summary>

public string ServerAddress;

/// <summary>

/// 服务器端口

/// </summary>

public int ServerPort;

/// <summary>

/// 开启服务的线程

/// </summary>

private Thread processor;

/// <summary>

/// 用于TCP监听

/// </summary>

private TcpListener tcpListener;

/// <summary>

/// 与客户端连接的套接字接口

/// </summary>

private Socket clientSocket;

/// <summary>

/// 用于处理客户事件的线程

/// </summary>

private Thread clientThread;

/// <summary>

/// 手机客户端所有客户端的套接字接口

/// </summary>

private Hashtable PhoneClientSockets = new Hashtable();

/// <summary>

/// 手机用户类数组

/// </summary>

public ArrayList PhoneUsersArray = new ArrayList();

/// <summary>

/// 手机用户名数组

/// </summary>

public ArrayList PhoneUserNamesArray = new ArrayList();

/// <summary>

/// 图像数据流

/// </summary>

private ArrayList StreamArray;

然后定义处理客户端传递数据的函数ProcessClient()，主要对接收数据进行命令解析。如果是手机连接的命令（"PHONECONNECT"），就在记录该套接字对象，同时在列表中添加该对象；如果是断开连接的命令（"PHONEDISCONNECT"）,就移除该对象；如果是手机视频命令（"PHONEVIDEO"），就分解其包含的图像数据，如果存在该用户对应的视频窗口，就传递该图像数据到这个视频窗口中。具体代码如下：

#region 处理客户端传递数据及处理事情

/// <summary>

/// 处理客户端传递数据及处理事情

/// </summary>

private void ProcessClient()

{

Socket client = clientSocket;

bool keepalive = true;

while (keepalive)

{

Thread.Sleep(50);

Byte[] buffer = null;

bool tag = false;

try

{

buffer = new Byte[1024];//client.Available

int count = client.Receive(buffer, SocketFlags.None);//接收客户端套接字数据

if (count > 0)//接收到数据

tag = true;

}

catch (Exception e)

{

keepalive = false;

if (client.Connected)

client.Disconnect(true);

client.Close();

}

if (!tag)

{

if (client.Connected)

client.Disconnect(true);

client.Close();

keepalive = false;

}

string clientCommand = "";

try

{

clientCommand = System.Text.Encoding.UTF8.GetString(buffer);//转换接收的数据,数据来源于客户端发送的消息

if (clientCommand.Contains("%7C"))//从Android客户端传递部分数据

clientCommand = clientCommand.Replace("%7C", "|");//替换UTF中字符%7C为|

}

catch

{

}

//分析客户端传递的命令来判断各种操作

string[] messages = clientCommand.Split('|');

if (messages != null && messages.Length > 0)

{

string tempStr = messages[0];//第一个字符串为命令

if (tempStr == "PHONECONNECT")//手机连接服务器

{

try

{

string tempClientName = messages[1].Trim();

PhoneClientSockets.Remove(messages[1]);//删除之前与该用户的连接

PhoneClientSockets.Add(messages[1], client);//建立与该客户端的Socket连接

UserClass tempUser = new UserClass();

tempUser.UserName = tempClientName;

tempUser.LoginTime = DateTime.Now;

Socket tempSocket = (Socket)PhoneClientSockets[tempClientName];

tempUser.IPAddress = tempSocket.RemoteEndPoint.ToString();

int tempIndex = PhoneUserNamesArray.IndexOf(tempClientName);

if (tempIndex >= 0)

{

PhoneUserNamesArray[tempIndex] = tempClientName;

PhoneUsersArray[tempIndex] = tempUser;

MemoryStream stream2 = (MemoryStream)StreamArray[tempIndex];

if (stream2 != null)

{

stream2.Close();

stream2.Dispose();

}

}

else//新增加

{

PhoneUserNamesArray.Add(tempClientName);

PhoneUsersArray.Add(tempUser);

StreamArray.Add(null);

}

RefreshPhoneUsers();

}

catch (Exception except)

{

}

}

else if (tempStr == "PHONEDISCONNECT")//某个客户端退出了

{

try

{

string tempClientName = messages[1];

RemovePhoneUser(tempClientName);

int tempPhoneIndex = PhoneUserNamesArray.IndexOf(tempClientName);

if (tempPhoneIndex >= 0)

{

PhoneUserNamesArray.RemoveAt(tempPhoneIndex);

MemoryStream memStream = (MemoryStream)StreamArray[tempPhoneIndex];

if (memStream != null)

{

memStream.Close();

memStream.Dispose();

}

StreamArray.RemoveAt(tempPhoneIndex);

}

Socket tempSocket = (Socket)PhoneClientSockets[tempClientName];//第1个为客户端的ID,找到该套接字

if (tempSocket != null)

{

tempSocket.Close();

PhoneClientSockets.Remove(tempClientName);

}

keepalive = false;

}

catch (Exception except)

{

}

RefreshPhoneUsers();

}

else if (tempStr == "PHONEVIDEO")//接收手机数据流

{

try

{

string tempClientName = messages[1];

string tempForeStr = messages[0] + "%7C" + messages[1] + "%7C";

int startCount = System.Text.Encoding.UTF8.GetByteCount(tempForeStr);

try

{

MemoryStream stream = new MemoryStream();

if (stream.CanWrite)

{

stream.Write(buffer, startCount, buffer.Length - startCount);

int len = -1;

while ((len = client.Receive(buffer)) > 0)

{

stream.Write(buffer, 0, len);

}

}

stream.Flush();

int tempPhoneIndex = PhoneUserNamesArray.IndexOf(tempClientName);

if (tempPhoneIndex >= 0)

{

MemoryStream stream2 = (MemoryStream)StreamArray[tempPhoneIndex];

if (stream2 != null)

{

stream2.Close();

stream2.Dispose();

}

StreamArray[tempPhoneIndex] = stream;

PhoneVideoForm form = GetPhoneVideoForm(tempClientName);

if (form != null)

form.DataStream = stream;

}

}

catch

{

}

}

catch (Exception except)

{

}

}

}

else//客户端发送的命令或字符串为空,结束连接

{

try

{

client.Close();

keepalive = false;

}

catch

{

keepalive = false;

}

}

}

}

#endregion

关于开启服务监听、刷新用户列表、获取手机视频窗体、删除用户、寻找用户序号等代码在此就不详细介绍，具体参见源代码。