计算机网络

**课程设计报告**

**(15级计算机、网络工程、信息安全、物联网专业)**

**姓名： 李东**

**学号： 1505040117**

**班级： 物联网一班**

**指导老师： 余庆春**

**湖南科技大学计算机科学与工程学院**

**2017年12月**

1. 前言

这一次计算机网络作业我做了4个题目，分别是聊天程序、ping程序、邮件客户端、Web服务器。使用的软件是VS2017企业版，使用的语言是C#。

二、课程设计目的

1．加深对计算机网络通信系统的工作原理的理解

通过编写计算机程序实现、模拟网络的某些功能，将书本上抽象的概念与具体实现技术结合起来，理解并掌握计算机网络的基本工作原理及工作过程。

2．实现应用进程跨越网络的通信

了解系统调用和应用编程接口基本知识，理解应用程序和操作系统之间传递控制权的机制，掌握套接字的创建和运用，通过socket系统调用实现跨网通信。

3．提高网络编程和应用的能力

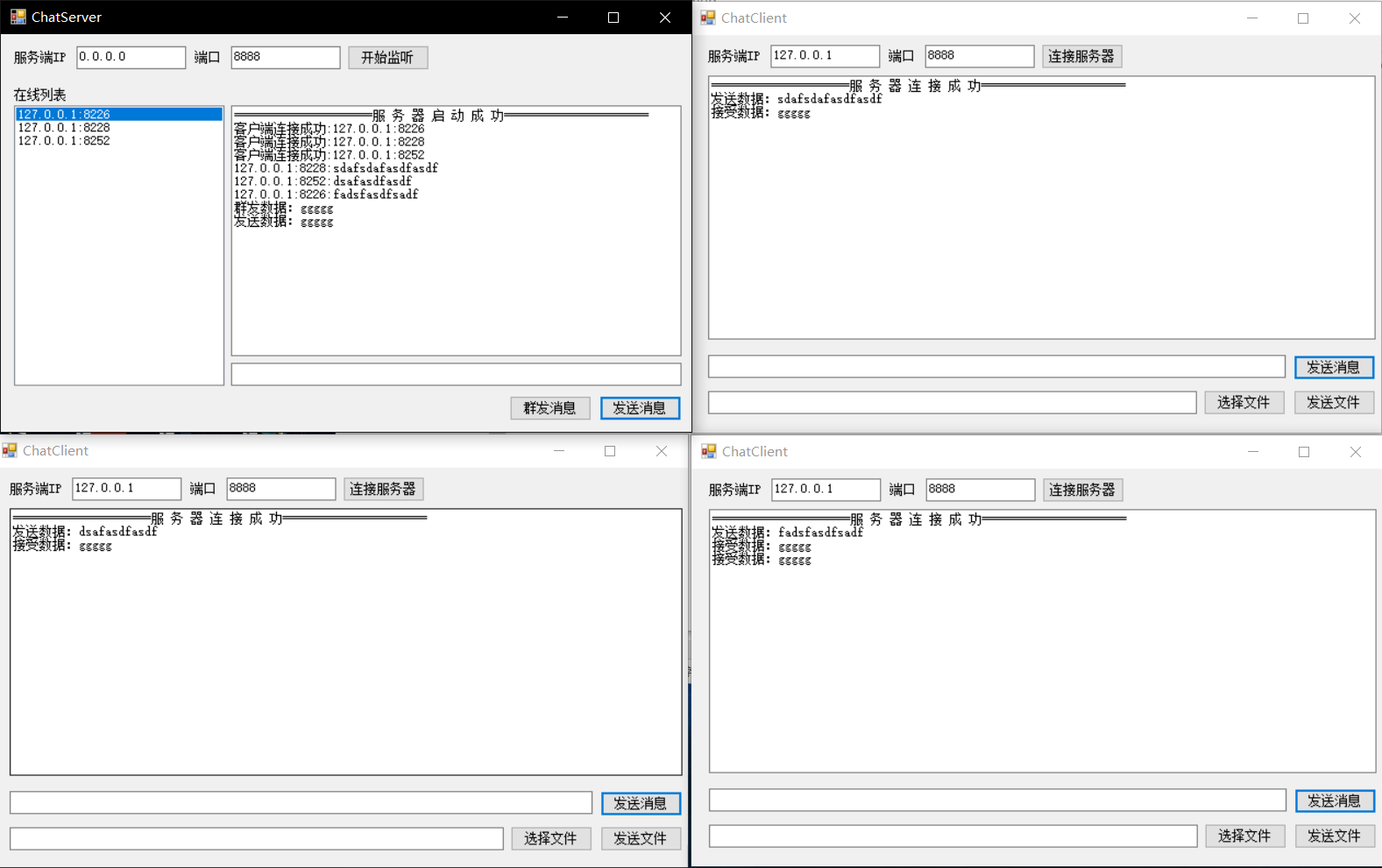
提高实际编程能力和灵活运用所学知识解决问题的能力。培养调查研究、查阅技术文献、资料、手册以及编写技术文档的能力，理论应用于实践的能力。

## 三、课程设计内容

#### I、聊天程序

1、功能介绍

服务器首先在主线程里创建一个监听线程，监听线程一直运行，一旦有一个socket连接进来，马上创建一个通信线程专门用来通信。实现的功能有服务器端接受所有客户端发来的消息，客户端能够发送文件，发送消息，服务器端能够群发消息给所有客户端，也能够私发消息给客户端，缺陷是客户端之间不能私聊，而且发送给服务器的信息其他客户端不能接收。



程序界面（左上角为服务器，其他为客户端）

2、核心代码

①服务器开启监听事件代码

**private** **void** btnBeginListen\_Click(object sender, EventArgs e)

{

*// 创建 服务器 负责监听的套接字 参数(使用IP4寻址协议，使用流式连接，使用TCP传输协议)*

socketServer = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

*//获取IP地址*

IPAddress ip = IPAddress.Parse(**this**.txtIP.Text.Trim());

*//创建 包含IP和Port的网络节点对象*

IPEndPoint endPoint = **new** IPEndPoint(ip, int.Parse(**this**.txtPort.Text.Trim()));

*//将负责监听 的套接字 绑定到 唯一的IP和端口上*

socketServer.Bind(endPoint);

*//设置监听队列 一次可以处理的最大数量*

socketServer.Listen(10);

*//创建线程 负责监听*

threadWatch = **new** Thread(WatchConnection);

*//设置为后台线程*

threadWatch.IsBackground = **true**;

*//开启线程*

threadWatch.Start();

ShowMsg("=====================服 务 器 启 动 成 功======================");

}

②监听线程

*//监听方法*

**void** WatchConnection()

{

*//持续不断的监听*

**while** (**true**)

{

*//开始监听 客户端 连接请求 【注意】Accept方法会阻断当前的线程--未接受到请求 程序卡在那里*

Socket sokConnection = socketServer.Accept();*//返回一个 负责和该客户端通信的 套接字*

*//将返回的新的套接字 存储到 字典序列中*

socketDict.Add(sokConnection.RemoteEndPoint.ToString(), sokConnection);

*//向在线列表中 添加一个 客户端的ip端口字符串 作为客户端的唯一标识*

ItemChanged(sokConnection.RemoteEndPoint.ToString(), **true**);

*//打印输出*

ShowMsg("客户端连接成功:" + sokConnection.RemoteEndPoint.ToString());

*//为该通信Socket 创建一个线程 用来监听接收数据*

*//在可以调用 OLE 之前，必须将当前线程设置为单线程单元(STA)模式。请确保您的 Main 函数带有*

*//////STAThreadAttribute 标记。 只有将调试器附加到该进程才会引发此异常。*

threadRec = **new** Thread(**new** ParameterizedThreadStart(RecMsg));

threadRec.SetApartmentState(ApartmentState.STA);

threadRec.IsBackground = **true**;

threadRec.Start(sokConnection);

*//通信线程集合*

dictThread.Add(sokConnection.RemoteEndPoint.ToString(), threadRec);

}

}

3、通信线程

**void** RecMsg(object socket)

{

Socket m\_socket = (Socket)socket;

*//持续监听接收数据*

**while** (**true**)

{

**try**

{

*//实例化一个字符数组*

byte[] data = **new** byte[1024 \* 1024];

*//接受消息数据*

*//远程主机强迫关闭了一个现有的连接。*

int receiveBytesLength = m\_socket.Receive(data);*//客户端关闭了出错*

**if** (data[0] == 0)

{

SaveFileDialog sfd = **new** SaveFileDialog();

**if** (sfd.ShowDialog() == DialogResult.OK)

{

**using** (FileStream fs = **new** FileStream(sfd.FileName, FileMode.Create))

{

fs.Write(data, 1, receiveBytesLength - 1);

fs.Flush();

ShowMsg("文件保存成功，路径为：" + sfd.FileName);

}

}

}

**else**

{

*//转换成字符串*

string recMsg = Encoding.UTF8.GetString(data, 0, receiveBytesLength);

*//打印接收到的数据*

ShowMsg(((Socket)socket).RemoteEndPoint.ToString() + ":" + recMsg);

}

}

*//自动义异常处理方法，把已保存的socket数据移除*

**catch**(System.Exception err)

{

ShowMsg(err.Message);

ItemChanged(m\_socket.RemoteEndPoint.ToString(), **false**);

socketDict.Remove(m\_socket.RemoteEndPoint.ToString());

dictThread.Remove(m\_socket.RemoteEndPoint.ToString());

m\_socket.Close();

**break**;

}

}

}

④客户端初始化

**private** **void** btnConnect\_Click(object sender, EventArgs e)

{

*//新建一个Socket 负责 监听服务器的通信*

socketClient = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

timerClient.Tick -= TimerClient\_Tick;

timerClient.Tick += TimerClient\_Tick;

timerClient.Interval = 1000;

timerClient.Enabled = **true**;

}

**private** **void** TimerClient\_Tick(object sender, EventArgs e)

{*//连接远程主机*

**try**

{

*//获取IP*

IPAddress ip = IPAddress.Parse(txtIP.Text.Trim());

*//新建一个网络节点*

IPEndPoint endPoint = **new** IPEndPoint(ip, int.Parse(txtPort.Text.Trim()));

socketClient.Connect(endPoint);

*//打印输出*

ShowMsg("=====================服 务 器 连 接 成 功======================");

*//创建线程 监听服务器 发来的消息*

threadReceive = **new** Thread(RecMsg);

*//设置为后台线程*

threadReceive.IsBackground = **true**;

*//开启线程*

threadReceive.Start();

**this**.timerClient.Enabled = **false**;

}

**catch**

{}

}

⑤、客户端通信线程

**void** RecMsg()

{

**while** (**true**)

{

**try**

{

*//初始化一个 1M的 缓存区(字节数组)*

byte[] data = **new** byte[1024 \* 1024];

*//将接受到的数据 存放到data数组中 返回接受到的数据的实际长度*

int receiveBytesLength = socketClient.Receive(data);*//服务端关闭了出错*

*//将字符串转换成字节数组*

string strMsg = Encoding.UTF8.GetString(data, 0, receiveBytesLength);

*//打印输出*

ShowMsg("接受数据：" + strMsg);

}

**catch**

{

socketClient.Close();

**break**;

}

}

}

3、设计步骤

第一天编写了客户端和服务端的界面，然后对客户端服务端进行初始化，然后写了监听线程，第二天编写了接收线程，listbox选择功能，界面回调功能，绑定对应事件，优化代码逻辑。

4、调试过程

这个程序调试的主要问题是接收线程，接受消息之后在主线程UI显示，还有就是服务器端选择客户端私发消息特别麻烦，你要保证接受没有丢失，而且线程不会突然关闭。选择客户端发送的难点在于我对于listbox的运用很少(基本没有)，接收消息在主线程UI显示是我很少编写多线程程序，不知道子线程修改主线程UI需要回调。

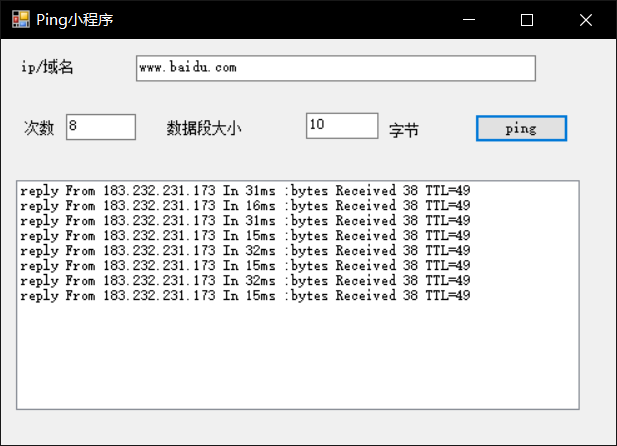
5、心得体会

编写这个程序我发现我很少重视用户体验，作为一个软件开发者，首先你的软件一定要容易上手，对不同的操作返回不同的消息，让用户对当前程序运行状态有个充分的了解。 没有把所有功能都想好就提前下手写了，导致后面客户端发送的消息除了服务器端其他客户端都看不到。

#### II、ping程序

1、功能介绍

定义了一个icmppackage类，封装好，然后在点击ping按钮时进行调用，然后发送icmp包给host，然后从返回来的数据报文中找到TTL数值显示。



界面如上所示

1. 、核心代码
2. IcmpPackage类

public class IcmpPacket

{

private Byte \_type;

// 类型

private Byte \_subCode;

// 代码

private UInt16 \_checkSum;

// 校验和

private UInt16 \_identifier;

// 识别符

private UInt16 \_sequenceNumber;

// 序列号

private Byte[] \_data;

//选项数据

public IcmpPacket(Byte type, Byte subCode, UInt16 checkSum, UInt16 identifier, UInt16 sequenceNumber, int dataSize)

{

\_type = type;

\_subCode = subCode;

\_checkSum = checkSum;

\_identifier = identifier;

\_sequenceNumber = sequenceNumber;

\_data = new Byte[dataSize];

//在数据中，写入指定的数据大小

for (int i = 0; i < dataSize; i++)

{

//由于选项数据在此命令中并不重要，所以你可以改换任何你喜欢的字符

\_data[i] = (byte)'#';

}

}

public UInt16 CheckSum

{

get

{

return \_checkSum;

}

set

{

\_checkSum = value;

}

}

//初始化ICMP报文

public int CountByte(Byte[] buffer)

{

Byte[] b\_type = new Byte[1] { \_type };

Byte[] b\_code = new Byte[1] { \_subCode };

Byte[] b\_cksum = BitConverter.GetBytes(\_checkSum);

Byte[] b\_id = BitConverter.GetBytes(\_identifier);

Byte[] b\_seq = BitConverter.GetBytes(\_sequenceNumber);

int i = 0;

Array.Copy(b\_type, 0, buffer, i, b\_type.Length);

i += b\_type.Length;

Array.Copy(b\_code, 0, buffer, i, b\_code.Length);

i += b\_code.Length;

Array.Copy(b\_cksum, 0, buffer, i, b\_cksum.Length);

i += b\_cksum.Length;

Array.Copy(b\_id, 0, buffer, i, b\_id.Length);

i += b\_id.Length;

Array.Copy(b\_seq, 0, buffer, i, b\_seq.Length);

i += b\_seq.Length;

Array.Copy(\_data, 0, buffer, i, \_data.Length);

i += \_data.Length;

return i;

}

//将整个ICMP报文信息和数据转化为Byte数据包

public static UInt16 SumOfCheck(UInt16[] buffer)

{

int cksum = 0;

for (int i = 0; i < buffer.Length; i++)

cksum += (int)buffer[i];

cksum = (cksum >> 16) + (cksum & 0xffff);

cksum += (cksum >> 16);

return (UInt16)(~cksum);

}

}

2、ping事件

**private** **void** ping\_Click(object sender, EventArgs e)

{

Listbox1.Items.Clear();

String Hostclient = Textbox1.Text;

int K;

int j = Convert.ToInt16(textBox2.Text);

**for**(K = 0 ; K < j; K++ )

　 {

Socket Socket = **new** Socket(AddressFamily.InterNetwork, SocketType.Raw, ProtocolType.Icmp);

IPHostEntry Hostinfo;

**try**{

*//解析主机ip入口*

Hostinfo = Dns.GetHostByName(Hostclient);}

**catch**(Exception)

{ *//解析主机名错误。*

Listbox1.Items.Add("没有发现此主机！");

**return**;}

*// 取服务器端主机的30号端口*

EndPoint Hostpoint = (EndPoint)**new** IPEndPoint(Hostinfo.AddressList[0], 30);

IPHostEntry Clientinfo;

Clientinfo = Dns.GetHostByName(Hostclient);

*// 取客户机端主机的30端口*

EndPoint Clientpoint = (EndPoint)**new** IPEndPoint(Clientinfo.AddressList[0], 30);

*//设置icmp报文*

int Datasize = Convert.ToInt16(textBox3.Text); *// Icmp数据包大小 ;*int Packetsize = Datasize + 8;*//总报文长度*

**const** int Icmp\_echo = 8;

IcmpPacket Packet = **new** IcmpPacket(Icmp\_echo, 0, 0, 45, 0, Datasize);

Byte[] Buffer = **new** Byte[Packetsize];

int Index = Packet.CountByte(Buffer);

*//报文出错*

**if**(Index != Packetsize)

{Listbox1.Items.Add("报文出现问题!");**return**;}

int Cksum\_buffer\_length = (int)Math.Ceiling(((Double)Index) / 2);

UInt16[] Cksum\_buffer = **new** UInt16[Cksum\_buffer\_length];

int Icmp\_header\_buffer\_index = 0;

**for**(int I = 0 ; I < Cksum\_buffer\_length; I++ )

　　 {

*//将两个byte转化为一个uint16*

Cksum\_buffer[I] = BitConverter.ToUInt16(Buffer, Icmp\_header\_buffer\_index);

Icmp\_header\_buffer\_index += 2;

}

*//将校验和保存至报文里*

Packet.CheckSum = IcmpPacket.SumOfCheck(Cksum\_buffer);

*// 保存校验和后，再次将报文转化为数据包*

Byte[] Senddata = **new** Byte[Packetsize];

Index = Packet.CountByte(Senddata);

*//报文出错*

**if**(Index != Packetsize)

{Listbox1.Items.Add("报文出现问题!");

**return**;}

int Nbytes = 0;

*//系统计时开始*

int Starttime = Environment.TickCount;

*//发送数据包*

**if**((Nbytes = Socket.SendTo(Senddata, Packetsize, SocketFlags.None, Hostpoint)) == -1)

{

Listbox1.Items.Add("无法传送报文！");

}

Byte[] Receivedata = **new** Byte[256]; *//接收数据*

Nbytes = 0;

int Timeout = 0;

int Timeconsume = 0;

**while**(**true**)

{

Nbytes = Socket.ReceiveFrom(Receivedata, 256, SocketFlags.None, **ref** Clientpoint);

**if**(Nbytes == -1)

{

Listbox1.Items.Add("主机没有响应！");

**break**;}

**else** if(Nbytes > 0 )

　　　 {

Timeconsume = System.Environment.TickCount - Starttime;

*//得到发送报文到接收报文之间花费的时间*

Listbox1.Items.Add("reply From " + Hostinfo.AddressList[0].ToString() + " In "

+ Timeconsume + "ms :bytes Received " + Nbytes+" TTL="+Receivedata[8]);

**break**;

}

Timeconsume = Environment.TickCount - Starttime;

**if**(Timeout > 1000)

{

Listbox1.Items.Add("time Out");

**break**;

}}

*//关闭套接字*

Socket.Close();

}}

2、设计步骤

首先编写了IcmpPackage类，然后设计界面，关联Ping按钮事件，将结果显示在listbox里，TTL首先没有写出来，老师要求后从数据报中取出。

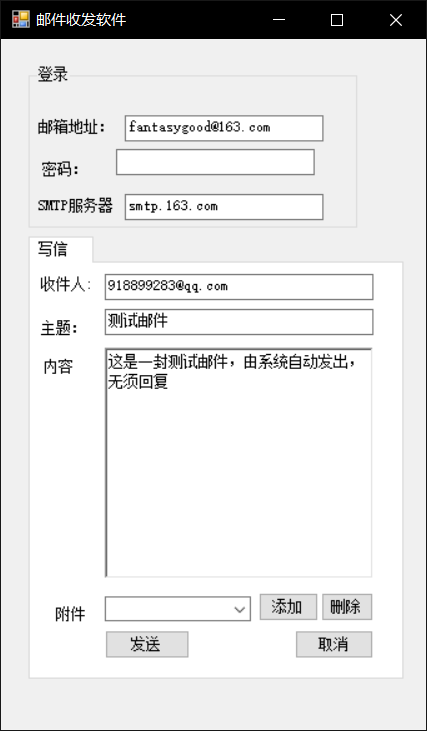
3、心得体会

知道了icmp包的构成，从发送返回的数据报中得晓了IP报文。

#### III、邮件客户端

1、功能介绍

使用了163的smtp服务器，然后登陆163邮箱账号，发送给收件人，实现了群发，添加附件的功能，缺陷是没有用POP3协议完成邮件的拉取，将邮件从网易云邮件中拉取到本地，并且实现附件下载的功能。



邮箱客户端界面

1. 核心代码

①添加附件

**private** **void** btnAddFile\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = **new** OpenFileDialog();

openFileDialog.CheckFileExists = **true**;

*// 只接受有效的文件名*

openFileDialog.ValidateNames = **true**;

*// 允许一次选择多个文件作为附件*

openFileDialog.Multiselect = **true**;

openFileDialog.Filter = "所有文件(\*.\*)|\*.\*";

**if** (openFileDialog.ShowDialog() != DialogResult.OK)

{

**return**;

}

**if** (openFileDialog.FileNames.Length > 0)

{

*// 因为这里允许选择多个文件，所以这里用AddRange而没有用Add方法*

cmbAttachment.Items.AddRange(openFileDialog.FileNames);

}

}

1. 、发送功能

**private** **void** btnSend\_Click(object sender, EventArgs e)

{*// 界面控件控制*

smtpClient = **new** SmtpClient();

smtpClient.Host = tbxSmtpServer.Text;

smtpClient.UseDefaultCredentials = **true**;

smtpClient.Port = 25;

smtpClient.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;

string mail = tbxUserMail.Text; string pass = txbPassword.Text;

smtpClient.Credentials = **new** System.Net.NetworkCredential(mail, pass);

**this**.Cursor = Cursors.WaitCursor;

*// 实例化一个发送的邮件*

*// 相当于与现实生活中先写信，程序中把信（邮件）抽象为邮件类了*

MailMessage mailMessage = **new** MailMessage();

*// 指明邮件发送的地址，主题，内容等信息*

*// 发信人的地址为登录收发器的地址，这个收发器相当于我们平时Web版的邮箱或者是OutLook中配置的邮箱*

mailMessage.From = **new** MailAddress(mail);

string[] toarray = txbSendTo.Text.Split(';');

int k = toarray.Length;

**for**(int j=0;j<k;j++) mailMessage.To.Add(toarray[j]);

mailMessage.Subject = txbSubject.Text;

mailMessage.SubjectEncoding = System.Text.Encoding.UTF8;

mailMessage.Body = richtbxBody.Text;

mailMessage.BodyEncoding = System.Text.Encoding.UTF8;

*// 设置邮件正文不是Html格式的内容*

mailMessage.IsBodyHtml = **false**;

*// 设置邮件的优先级为普通优先级*

mailMessage.Priority = MailPriority.High;

*// 封装发送的附件*

System.Net.Mail.Attachment attachment = **null**;

**if** (cmbAttachment.Items.Count > 0)

{

**for** (int i = 0; i < cmbAttachment.Items.Count; i++)

{

string fileNamePath = cmbAttachment.Items[i].ToString();

string extName = Path.GetExtension(fileNamePath).ToLower();

**if** (extName == ".rar" || extName == ".zip")

{

attachment = **new** System.Net.Mail.Attachment(fileNamePath, MediaTypeNames.Application.Zip);

}

**else**

{

attachment = **new** System.Net.Mail.Attachment(fileNamePath,MediaTypeNames.Application.Octet);

}

*// 表示MIMEContent-Disposition标头信息* ContentDisposition cd = attachment.ContentDisposition;

cd.CreationDate = File.GetCreationTime(fileNamePath);

cd.ModificationDate = File.GetLastWriteTime(fileNamePath);

cd.ReadDate = File.GetLastAccessTime(fileNamePath);

*// 把附件对象加入到邮件附件集合中*

mailMessage.Attachments.Add(attachment);

}

}

*// 发送写好的邮件*

**try**

{

*// SmtpClient类用于将邮件发送到SMTP服务器*

*// 该类封装了SMTP协议的实现，*

*// 通过该类可以简化发送邮件的过程，只需要调用该类的Send方法就可以发送邮件到SMTP服务器了。*

smtpClient.Send(mailMessage);

MessageBox.Show("邮件发送成功！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

**catch**(SmtpException smtpError)

{

MessageBox.Show("邮件发送失败：[" + smtpError.StatusCode + "]；["

+ smtpError.Message+"];\r\n["+smtpError.StackTrace+"]."

,"错误",MessageBoxButtons.RetryCancel,MessageBoxIcon.Error);

}

**finally**

{

mailMessage.Dispose();

**this**.Cursor = Cursors.Default;

}

}

3、编写步骤

编写界面，注册163邮箱并且打开smtp功能，获取授权码(第三方客户端登陆密码)，编写发送功能，然后增加附件功能和群发功能，群发功能通过加’;’来实现判别几个收件人。

4、调试步骤

一开始登陆的时候没有打开smtp服务，一直提示连接错误，然后打开smtp但是密码不是用的是授权码，用的是登录密码，然后一直授权错误，最后发现163发过来短信显示登陆密码已经改为授权码。

5、心得体会

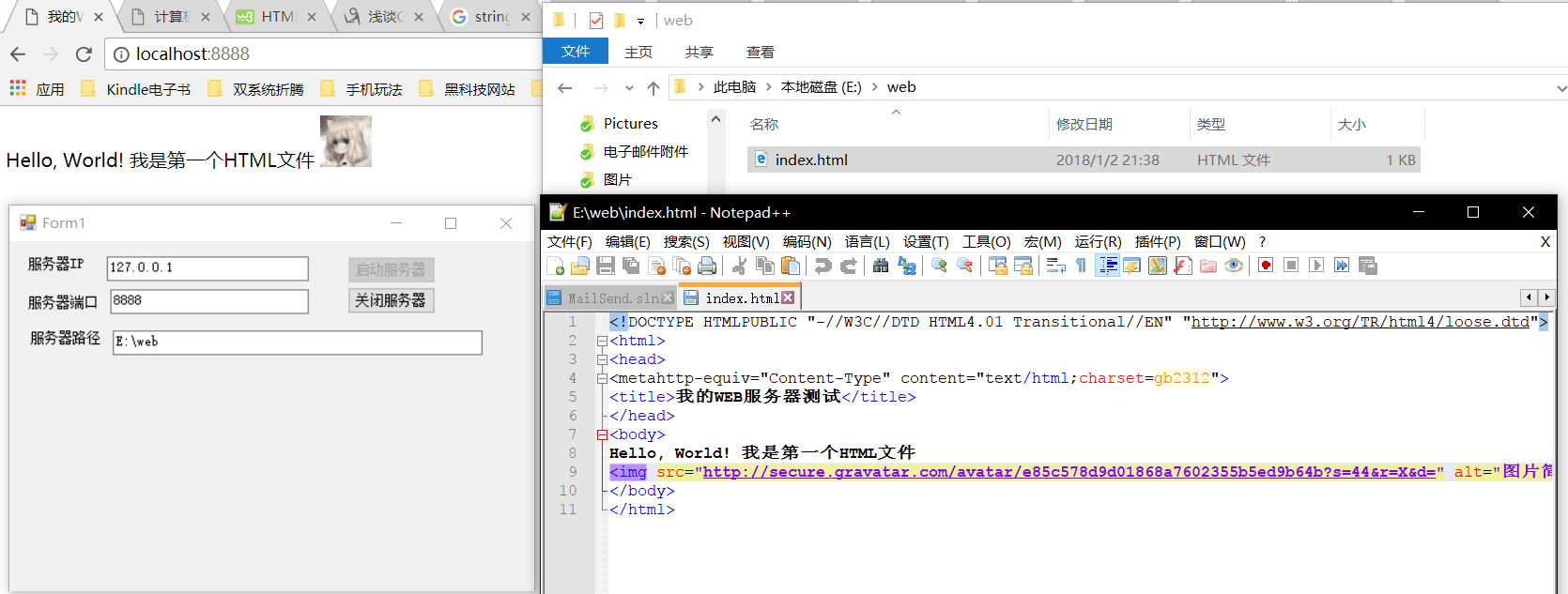
邮件客户端如果用别人的smtp服务确实还是比较好写的，而且一般不会被收信人拒收，如果用smtp协议直接发送，基本都进垃圾箱了。

#### Ⅳ、Web服务器

1、功能实现

我实现了设置网站目录，然后设置好服务器IP地址端口之后，启动服务器，然后我们就能够通过浏览器用GET方法查看网站目录下的文件，对于支持的文件类型，通过

一个Dictionary保存所有应该响应的格式，如果有需要增加的就写到Dictionary里就好了。POST功能通过我自己编写的一个HTTP POST客户端发送，服务器端辨别GET和POST命令，如果是POST命令，创建一个POST的URL请求文件，然后将客户端发来的信息用UTF8格式储存在文件里，然后我们就可以同浏览器查看了。比如我POST的URL为<http://localhost:8888/post.html。POST>成功之后我们就可以在浏览器里输入<http://localhost:8888/post.html查看POST>后储存在网站目录下面的新文件。



1. 核心代码

①、Dictionary estension包括响应文件类型

**private** Dictionary<string, string> extensions = **new** Dictionary<string, string>()

{

*//{ "extension", "content type" }*

{ "htm", "text/html" },

{ "html", "text/html" },

{ "xml", "text/xml" },

{ "txt", "text/plain" },

{ "css", "text/css" },

{"aspx","text/aspx" },

{"php","text/php" },

{ "png", "image/png" },

{ "gif", "image/gif" },

{ "jpg", "image/jpg" },

{ "jpeg", "image/jpeg" },

{ "zip", "application/zip"}

};

1. 、start方法

**public** bool start(IPAddress ipAddress, int port, int maxNOfCon, string contentPath)

{

**if** (running) **return** **false**; *// 如果已经在运行就退出程序*

**try**

{

*// A tcp/ip socket (ipv4)一个ipvv4的tcp/ip socket*

serverSocket = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream,

ProtocolType.Tcp);

serverSocket.Bind(**new** IPEndPoint(ipAddress, port));

serverSocket.Listen(maxNOfCon);

serverSocket.ReceiveTimeout = timeout;

serverSocket.SendTimeout = timeout;

running = **true**;

**this**.contentPath = contentPath;

}

**catch** { **return** **false**; }

*//监听线程，在此线程中会创建新线程处理连接请求*

Thread requestListenerT = **new** Thread(() =>

{

**while** (running)

{

Socket clientSocket;

**try**

{

clientSocket = serverSocket.Accept();

*//创建一个新线程处理请求并且继续监听*

Thread requestHandler = **new** Thread(() =>

{

clientSocket.ReceiveTimeout = timeout;

clientSocket.SendTimeout = timeout;

**try** { handleTheRequest(clientSocket); }

**catch**

{

**try** { clientSocket.Close(); } **catch** { }

}

});

requestHandler.Start();

}

**catch** { }

}

});

requestListenerT.Start();

**return** **true**;

}

1. 、处理请求的方法

**private** **void** handleTheRequest(Socket clientSocket)

{

byte[] buffer = **new** byte[10240]; *// 设置10kb的缓冲区，以防万一*

int receivedBCount = clientSocket.Receive(buffer); *// 接受请求*

string strReceived = charEncoder.GetString(buffer, 0, receivedBCount);

string strtemp = strReceived;

*// 解析请求功能*

*//*

int a = strReceived.LastIndexOf("Host")-2;

int b = strReceived.IndexOf("Content-Type")+14;

string type = strReceived.Substring(b, a - b);

string httpMethod = strReceived.Substring(0, strReceived.IndexOf(" "));

int start = strReceived.IndexOf(httpMethod) + httpMethod.Length + 1;

int length = strReceived.LastIndexOf("HTTP") - start - 1;

string requestedUrl = strReceived.Substring(start, length);

string requestedFile;

**if** (httpMethod.Equals("GET") || httpMethod.Equals("POST"))

requestedFile = requestedUrl.Split('?')[0];

**else** *// 你可以实现其他操作*

{

notImplemented(clientSocket);

**return**;

}

requestedFile = requestedFile.Replace("/", @"\").Replace("\\..", "");

start = requestedFile.LastIndexOf('.') + 1;

*//判断请求文件是否支持*

**if** (start > 0)

{

length = requestedFile.Length - start;

string extension = requestedFile.Substring(start, length);

**if** (extensions.ContainsKey(extension))

{ *// 检查是否支持这种文件类型*

**if** (httpMethod.Equals("POST"))

{

string filepath = contentPath + requestedFile;

string filename = requestedFile.Substring(1, requestedFile.Length-1);

StreamWriter sw = **new** StreamWriter(filepath,**false**);

sw.WriteLine(strtemp);

sw.Close();

}

**if** (File.Exists(contentPath + requestedFile)) *//如果确实支持*

*// 发送对应内容类型的响应报文*

sendOkResponse(clientSocket,

File.ReadAllBytes(contentPath + requestedFile), extensions[extension]);

**else**

notFound(clientSocket); *//不支持这种格式*

}

}

**else**

{

*// 如果没有指定请求文件，发送index.htm或者index.html*

**if** (requestedFile.Substring(length - 1, 1) != @"\")

requestedFile += @"\";

**if** (File.Exists(contentPath + requestedFile + "index.htm"))

sendOkResponse(clientSocket,

File.ReadAllBytes(contentPath + requestedFile + "\\index.htm"), "text/html");

**else** if (File.Exists(contentPath + requestedFile + "index.html"))

sendOkResponse(clientSocket,

File.ReadAllBytes(contentPath + requestedFile + "\\index.html"), "text/html");

**else**

notFound(clientSocket);

}

}

1. 、不同的响应处理（部分）

**private** **void** notFound(Socket clientSocket)

{

sendResponse(clientSocket, "<html><head><metahttp-equiv =\"Content-Type\" content=\"text/html;charset=utf-8\"></head><body><h2>Simple Web Server</h2><div>404-NotFound</div></body></html>", "404 Not Found", "text/html");

}

**private** **void** sendOkResponse(Socket clientSocket, byte[] bContent, string contentType)

{

sendResponse(clientSocket, bContent, "200 OK", contentType);

}

*// 发送字符串响应报文*

**private** **void** sendResponse(Socket clientSocket, string strContent, string responseCode,

string contentType)

{

byte[] bContent = charEncoder.GetBytes(strContent);

sendResponse(clientSocket, bContent, responseCode, contentType);

}

*// 发送bytes响应报文*

**private** **void** sendResponse(Socket clientSocket, byte[] bContent, string responseCode,

string contentType)

{

**try**

{

byte[] bHeader = charEncoder.GetBytes(

"HTTP/1.1 " + responseCode + "\r\n"

+ "Server: Simple Web Server\r\n"

+ "Content-Length: " + bContent.Length.ToString() + "\r\n"

+ "Connection: close\r\n"

+ "Content-Type: " + contentType + "\r\n\r\n");

clientSocket.Send(bHeader);

clientSocket.Send(bContent);

clientSocket.Close();

}

**catch** { }

}

1. 、POST客户端核心代码

**public** **static** string PostHttp(string url, string body, string contentType)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(url);

httpWebRequest.ContentType = contentType;

httpWebRequest.Method = "POST";

httpWebRequest.Timeout = 2000000;

byte[] btBodys = Encoding.UTF8.GetBytes(body);

long a = btBodys.Length;

httpWebRequest.ContentLength = a;

**using** (var stream = httpWebRequest.GetRequestStream())

{

stream.Write(btBodys, 0, btBodys.Length);

}

HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();

StreamReader streamReader = **new** StreamReader(httpWebResponse.GetResponseStream());

string responseContent = streamReader.ReadToEnd();

httpWebResponse.Close();

streamReader.Close();

httpWebRequest.Abort();

httpWebResponse.Close();

**return** responseContent;

}

3、编写步骤

花了一天半将WebServer类写了出来，然后编写界面，调用WebServer类。能在浏览器上实现GET方法之后用半天事件编写了POST客户端。

4、调试步骤

这个软件基本上是我调试事件最久的了，因为编码问题，和对URL处理的问题，要对POST传进来的数据进行操作，而且写到文件的时候要用UTF8编码才不会在用浏览器的时候出现乱码。

5、心得体会

做一个基本的htppsetver还是比较麻烦的，一个httpserver应该要实现6个基本操作，但是写2个都让我耗时较多，有所得的就是注意了文字编码，如果你的文字编码没有设置正确，在其他上面显示中文一般都会乱码。

### 四、最后总结

由于时间问题并没有完成所有的项目，只完成了4道题，但是对于计算机网络的很多协议已经有了初步的了解，socket的使用，数据报的组成，邮件协议，还有做web服务器让我学习了HTTP协议，由于时间没有做网络嗅探器和基于UDP协议的多播程序，还有代理程序，Telnet客户端，寒假我应该会找个时间写写试试。在这次课程设计中还得到了一个教训，写程序不要不设置很大的目标，先写出主要功能，然后在编写次要的功能能够让你对软件的开发得心应手。

全部源代码：<https://github.com/fantasylidong/http_proxy/blob/master/c%23_project.rar>