**《计算机网络》实验三**

**分析SMTP协议，编写一个简单SMTP服务器实验报告**

**——计算机网络第三次实验作业简介**

**姓名：谢江**

**学号：1410594**

**专业：计算机科学与技术**

**年级：大三**

**完成时间：2016.11.29**

1. **实验内容**

本次实验分为两部分：

1. 利用Wireshark捕获数据包，分析一个电子邮件客户端（我选用Foxmail）与SMTP服务器的交互过程。
2. 编写一个简单的SMTP服务器。
3. 实现的服务器应能与outlook express等现有的客户端进行简单的交互；

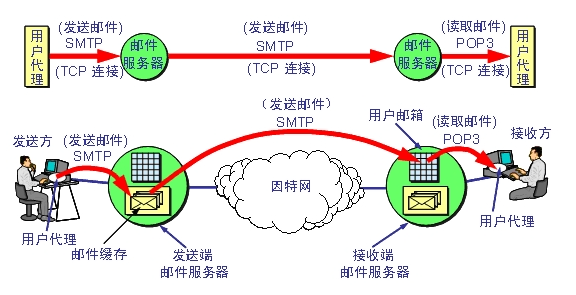
b）记录浏览器与服务的交互过程；

c）在屏幕上直接显示接收到的邮件内容；

d）将邮件附件中的图像（如bmp或jpg图像）显示在屏幕上；

1. **实验分析**

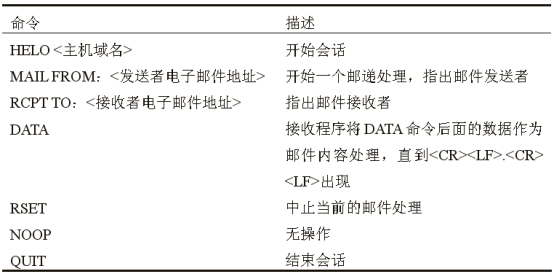
首先是第一部分，分析SMTP协议。如下：



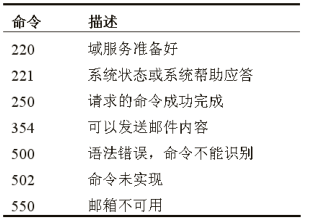
可以看出，SMTP是用来将客户机上的邮件传送到服务器上。这里的客户机是指某次连接中的发送方，服务器是指相应的接收方。

SMTP协议是基于TCP的协议，用于在网络上发送电子邮件。通过客户端与服务器的通信过程，每次向服务器发送不同的命令，得到服务器的回应可知是否成功，直到把邮件发送到服务器上。而后服务器由一条已经建立好的传输通道把邮件送到收件人所用的服务器上，收件人再根据POP协议从服务器上接收邮件，从而完成邮件的发送和接受过程。

SMTP 服务器在 TCP 的 25 端口守候, 其 规定发送程序和接收程序之间的命令和应答命令和响应都是可读的 ASCII 字符串。常见的 SMTP 命令有

如下：

其应答响应命令以 3 位数字开始，后面跟有该响应的具体描述，常见应答命令有如下：



POP3协议是SMTP服务器提供给用户代理读取邮件时的协议，其命令格式如下（本次测试主要分析SMTP协议）



1. **实验步骤**

**实验操作平台：360浏览器**

**Wireshark\_win32\_2.2.1.0**

**win10系统（64位）**

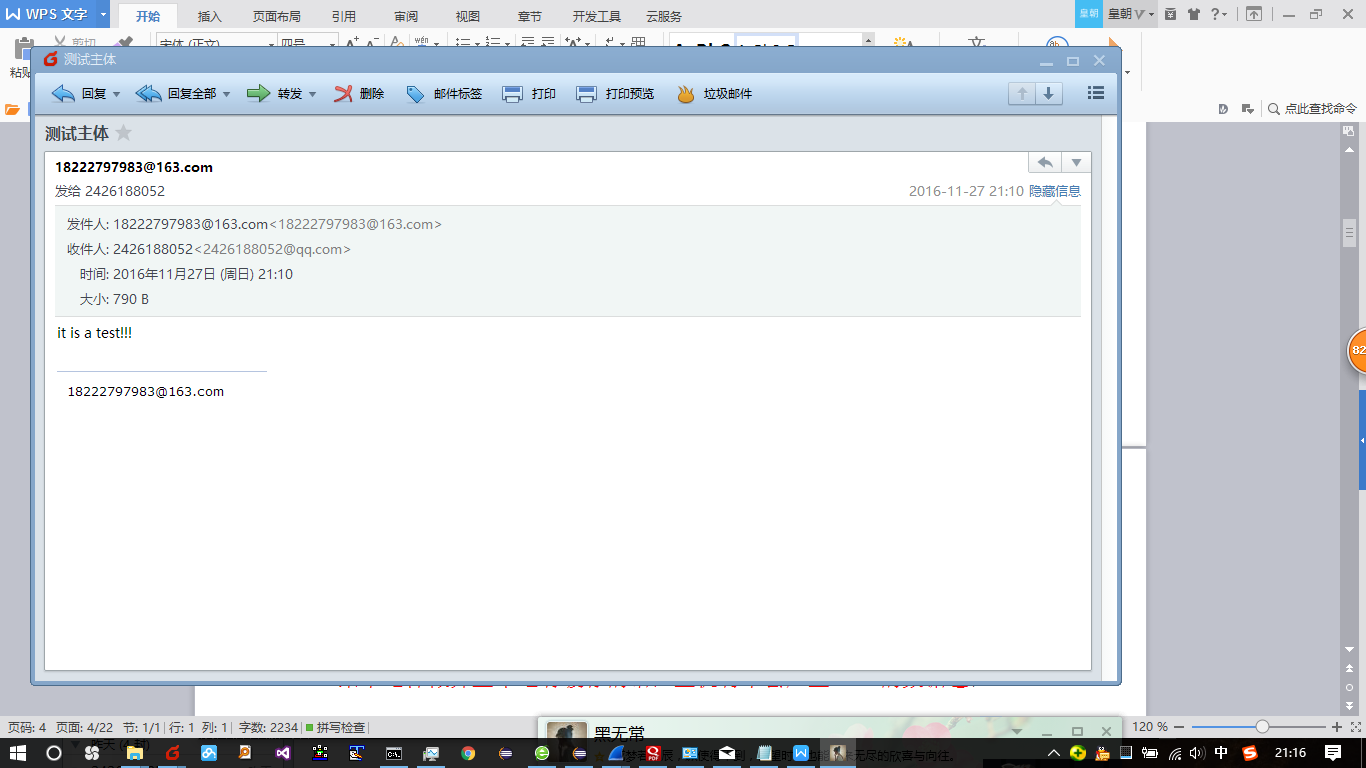
**VisualStudio2015（服务器编写）**

1. **利用Wireshark捕获数据包，分析一个电子邮件客户端（如Foxmail）与SMTP服务器的交互过程。**

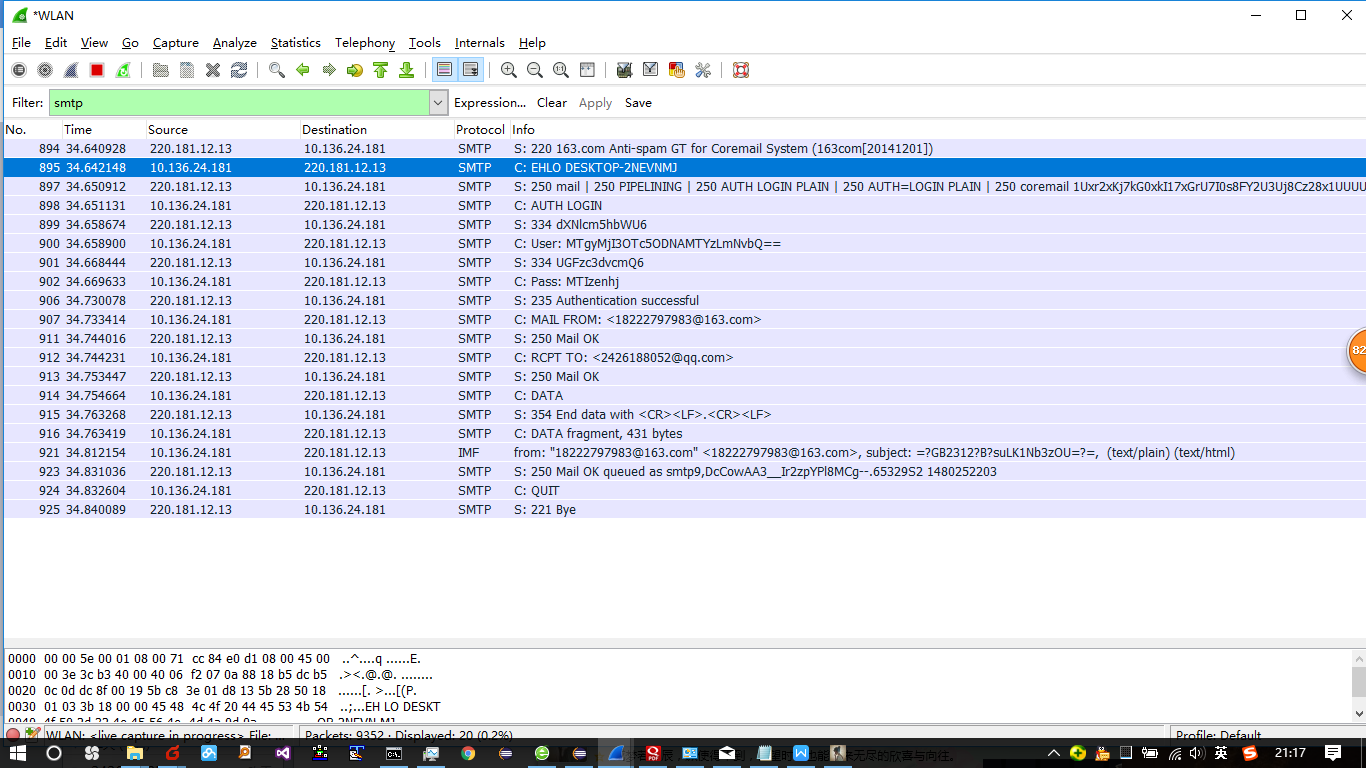
**注：（测试时我的主机分配的WLAN适配器IP地址是：10.136.24.181）**

1. 打开wireshark，选中ＷＬＡＮ端口，条件选择smtp，开始抓包。打开Foxmail，使用163设置的用户代理（18222797983@163.com）发送一封邮件，发送成功后如下





wireshark捕获数据包如下



b.分析捕获的数据包可知，Foxmail使用的IP地址是10.136.24.181,一个163SMTP邮件服务器的地址是220.181.12.13.

有如下交互==》》》

**SMTP 邮件服务器：220通知Foxmail与服务器准备好（894帧）**

**Foxmail提示开始会话（895）**

**SMTP邮件服务器:250表示允许（897）**

**Foxmail请求远程登录163邮箱，输入用户名和代理密码（898）**

**SMTP邮件服务器:334等待输入验证信息（用户名）（899）**

**[Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

**SMTP邮件服务器:334等待输入验证信息（密码）（901）**

**[Foxmail输入邮箱密码，此处采用base64编码（极不安全）（902）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

**SMTP邮件服务器:235远程登录成功，用户可以发邮件了（906）**

**[Foxmail 告诉服务器是谁发的（要发邮件）（907）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

**SMTP邮件服务器:250表示收到（911）**

**[Foxmail 告诉服务器发给谁（912）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

**SMTP邮件服务器:250表示收到（913）**

**[Foxmail 告诉服务器我接下来要发数据了（914）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

**SMTP邮件服务器:354表示可以发，最后以独立的“.”结尾就可以了（915）**

**[Foxmail 开始发送数据（由于数据比较小，所以只装在了一个包里，如果数据比较多，则会放在多个包进行发送）（916）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

**[Foxmail 发送数据完毕，告诉服务器邮件内容的完整的信息（收件方，发件方大小，格式等）（921）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

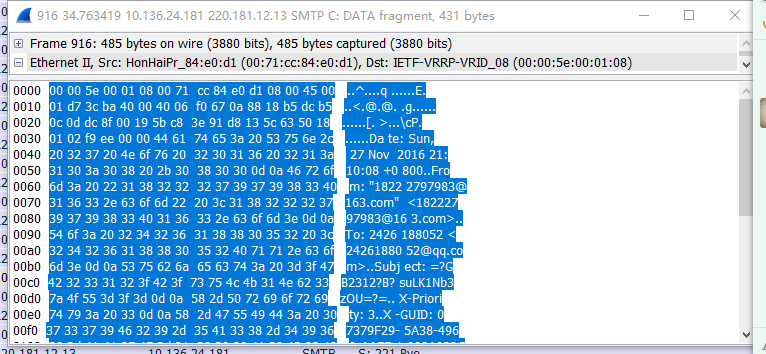
**SMTP邮件服务器:250表示完成请求（923）**

**[Foxmail 告诉服务器，我要走了（924）](mailto:Foxmail输入邮箱用户名（18222797983@163.com（此处采用base64编码））（900）)**

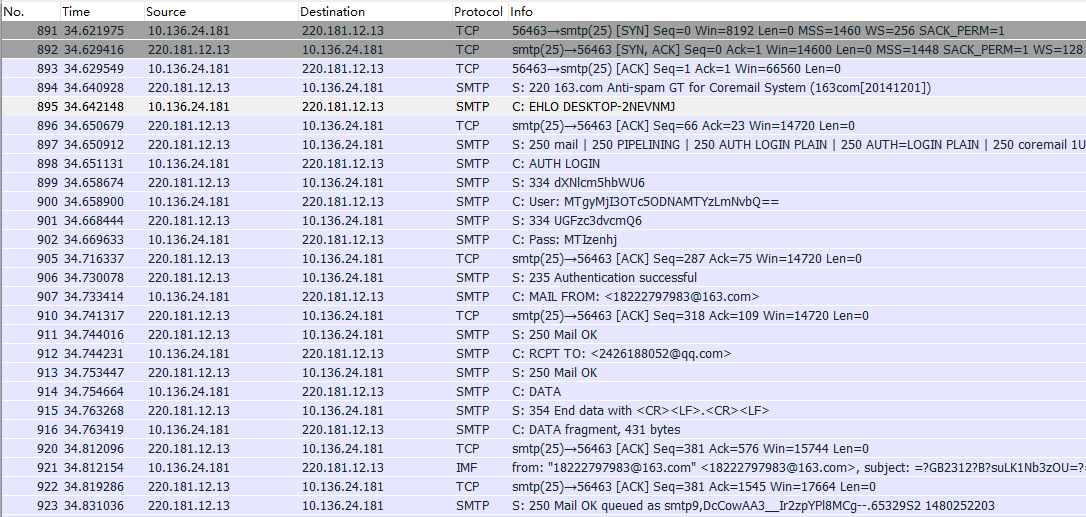
**SMTP邮件服务器:221走你！（925）**

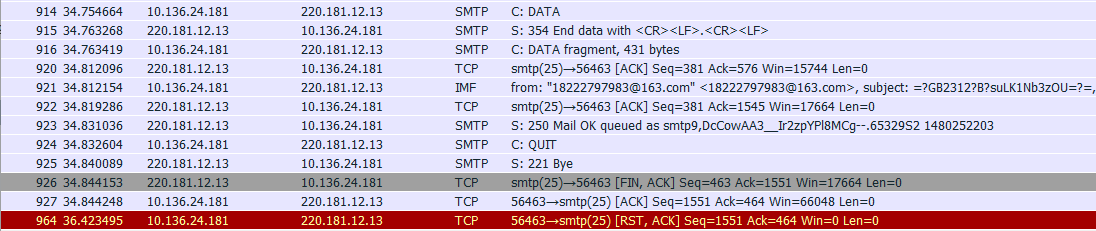
以上就是这一次Foxmail和SMTP邮件服务器的一次完整的交互。

c.分析地更深入可知，在发送数据的过程中，数据是一个字节一个字节发送过去的，按照base64编码的格式。因此上述分析只是交互的一个整体的概括，还有很多细节。如下：



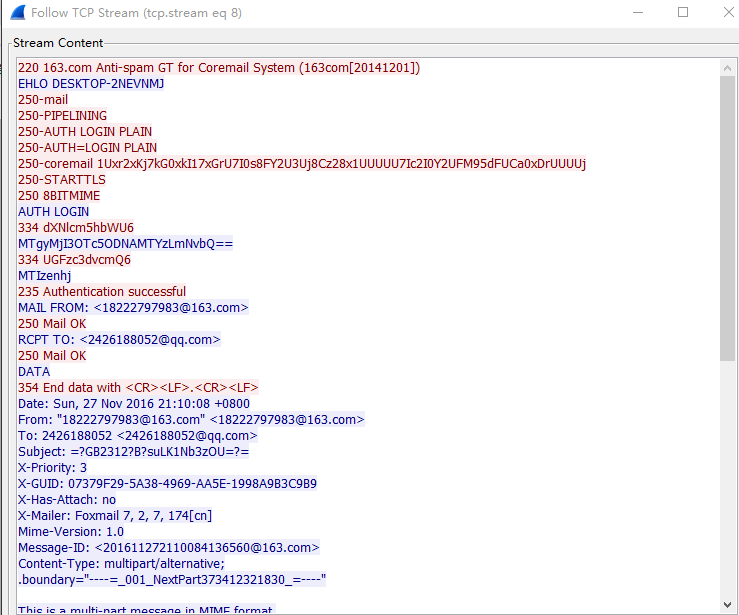
d.由于SMTP协议是建立TCP运输层协议之上的，因此必然会有TCP的三次握手和请求建立连接以及ACK应答报文之类的报文被wireshark捕获。选择交互过程中的数据包，利用wireshark的追踪TCP流可得如下数据包：

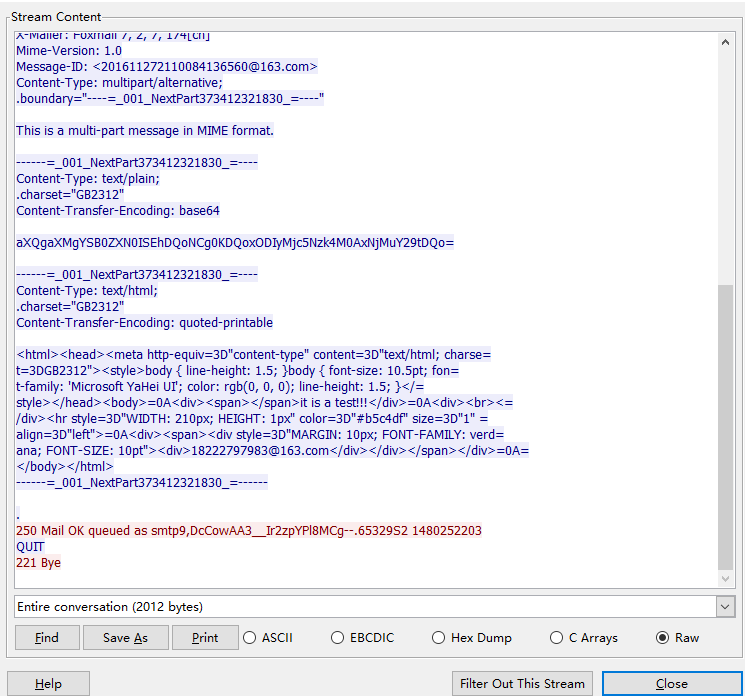




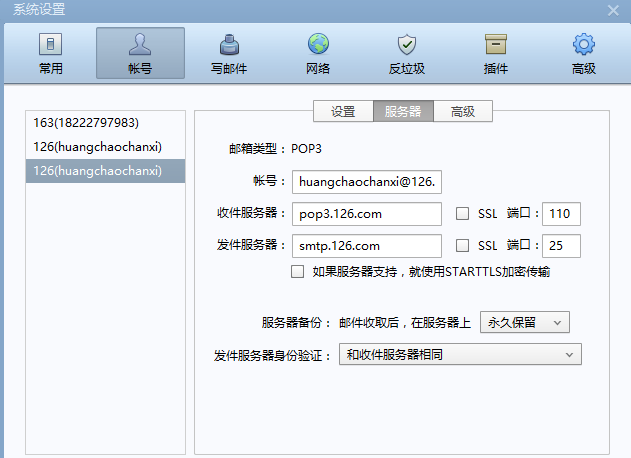
分析可知Foxmail首先与SMTP通过TCP三次握手建立连接之后才正式开始交互，而其中也夹杂着TCP为保证数据传输可靠性而产生的ACK等数据报流。这是一个TCP为SMTP协议提供可靠的服务保障。这是一个更为完整的Foxmail与SMTP邮件服务器交互的过程，也更加复杂。但是对于我们分析SMTP协议的逻辑性并没有太大的帮助。

e.获取Foxmail与SMTP整体的交换数据（红色为SMTP服务器发往Foxmail，蓝色相反），如下：

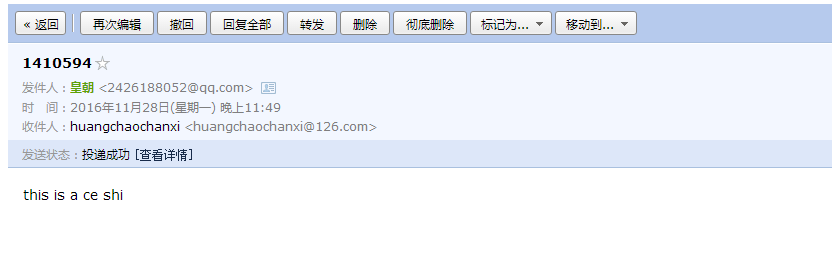




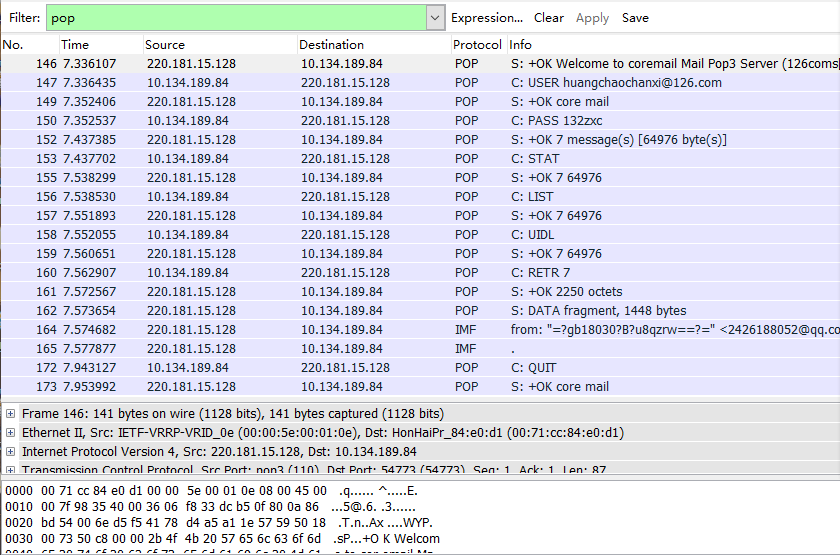
f.而在发送端，一般SMTP服务器采用pop3协议与用户代理交。由于我的163邮箱采用IMAP协议，因此这里我的Foxmail使用了126邮箱的用户代理（huangchaochanxi@126.com）收件服务器采用pop3协议，如下



g.使用网页版QQ邮箱发送邮件

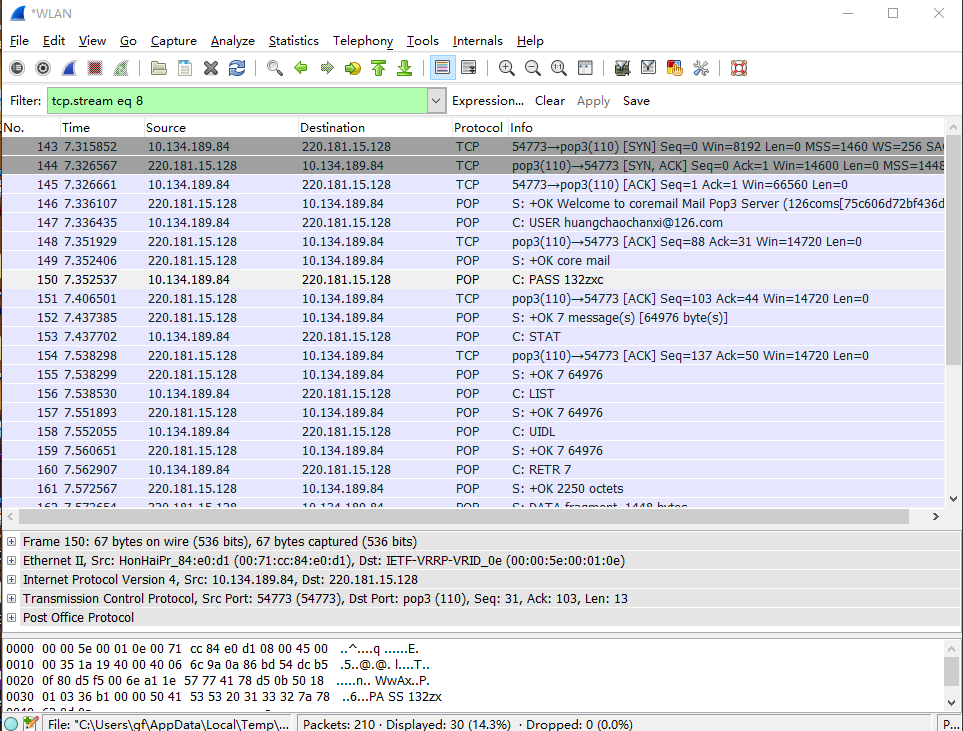


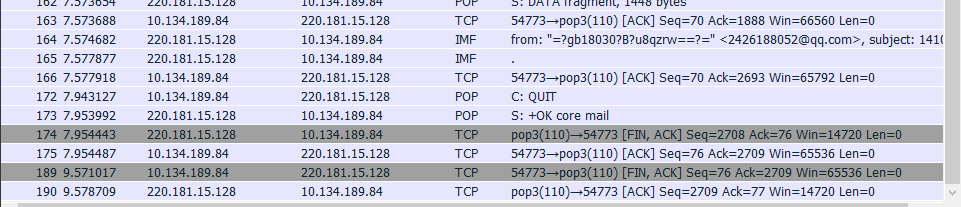
h.使用wireshark捕获数据包如下

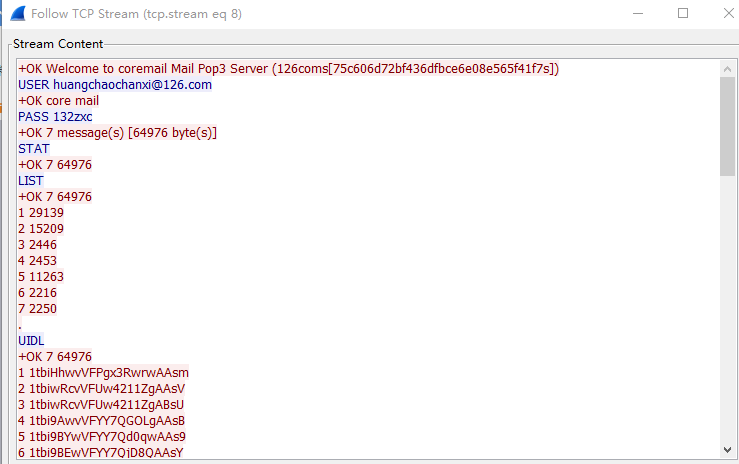


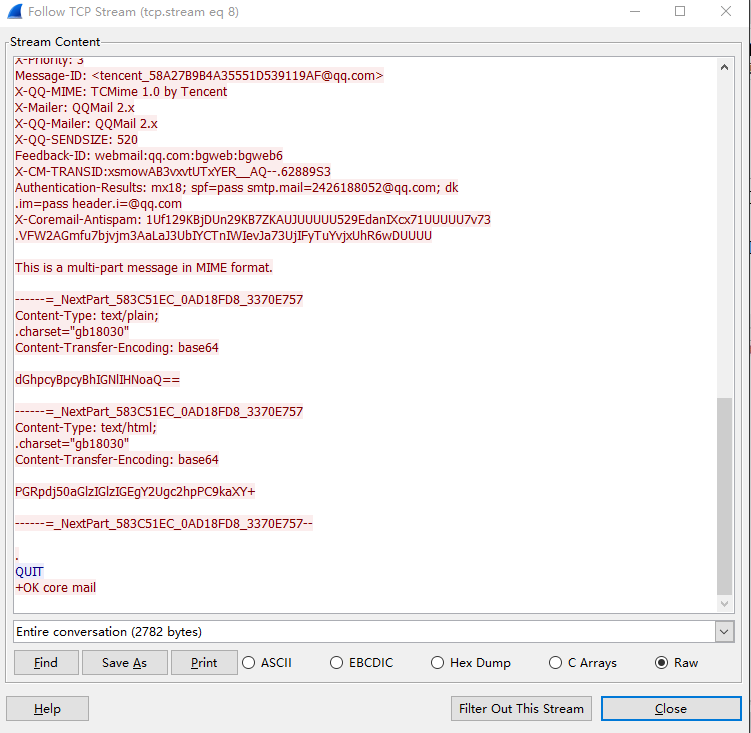
可知SMTP服务器与用户代理之间的交互遵循POP3协议，其分析和SMTP协议的分析一致，在此不再赘述。

i.而查看整体的TCP交互数据流，可知其仍旧遵循TCP连接交互过程（三次握手，四次挥手等），与SMTP协议一致，如下









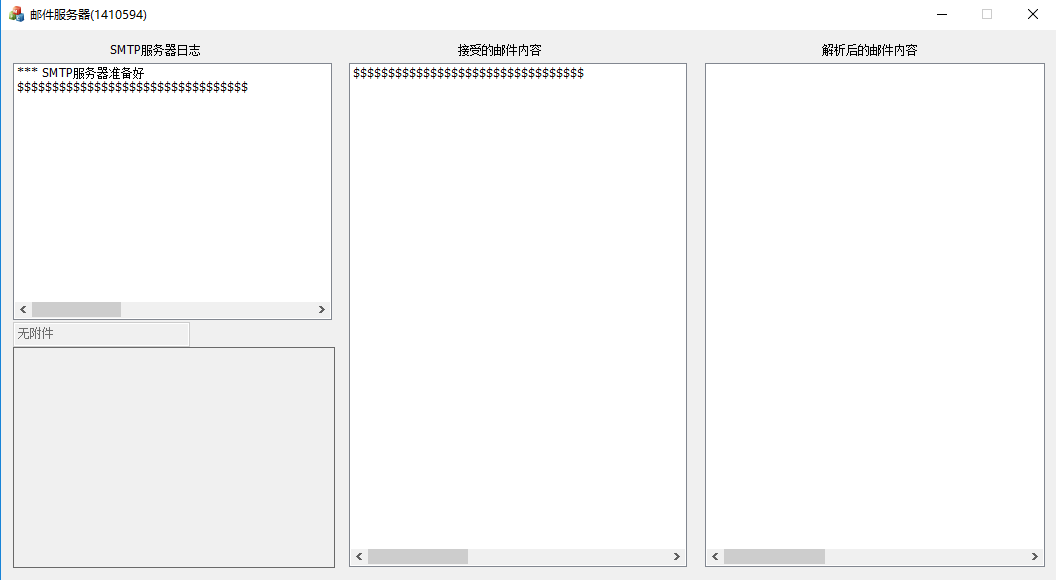
**至此，SMTP与一个电子邮件客户端（Foxmail）的交互分析完毕。**

**B.编写一个简单的SMTP服务器。**

1. 此次Web服务器编写采用C++语言

**—————————程序简介————————**

**a）服务器端界面如下：**



**【1】SMTP服务器日志：**此服务器与用户代理采用SMTP协议，底层采用TCP方式建立连接，故需要一个监听端口监听来自不同客户端的请求。

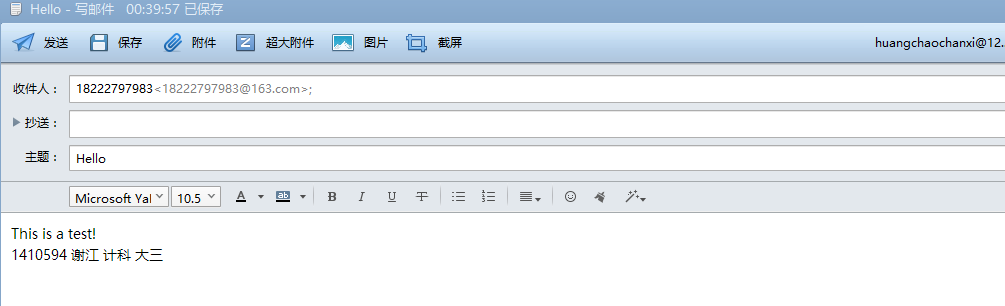
**【2】接收的邮件内容**：显示服务器收到的原始数据，包括协议头，附件数据等。

**【3】解析后的邮件内容：**显示邮件的正文（解码后）

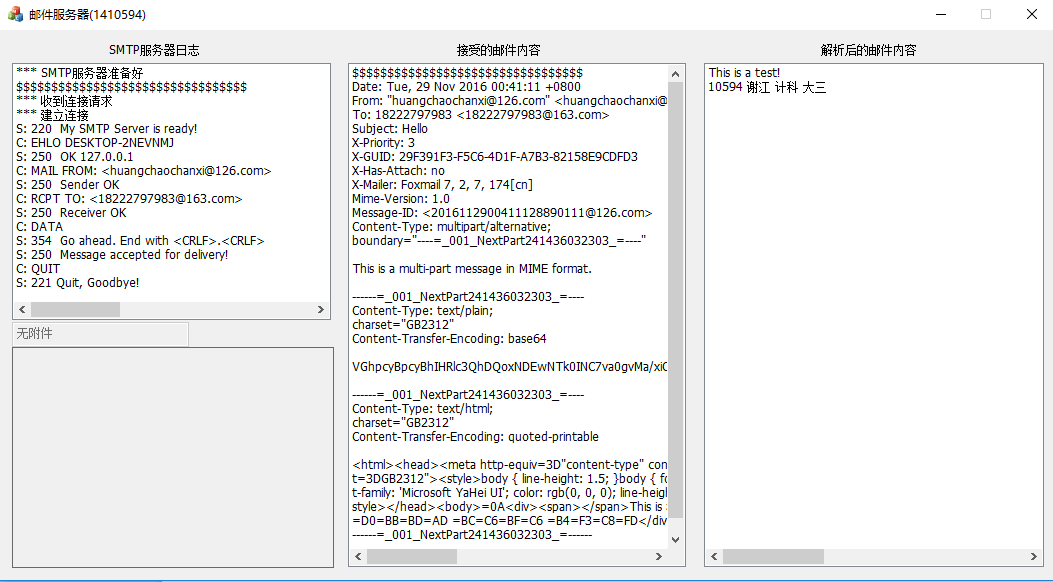
**【4】附件区：**如果有图片，会将图片显示出来

**b）观察无附件的邮件发送**

用Foxmail的126用户代理发送如下邮件



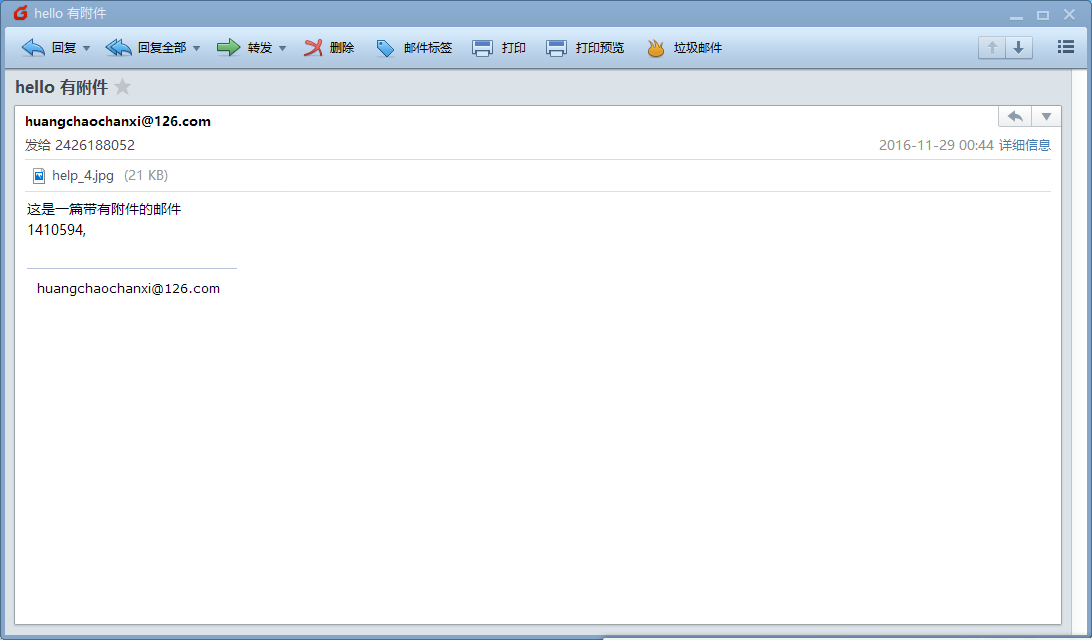
点击发送，服务器端显示如下



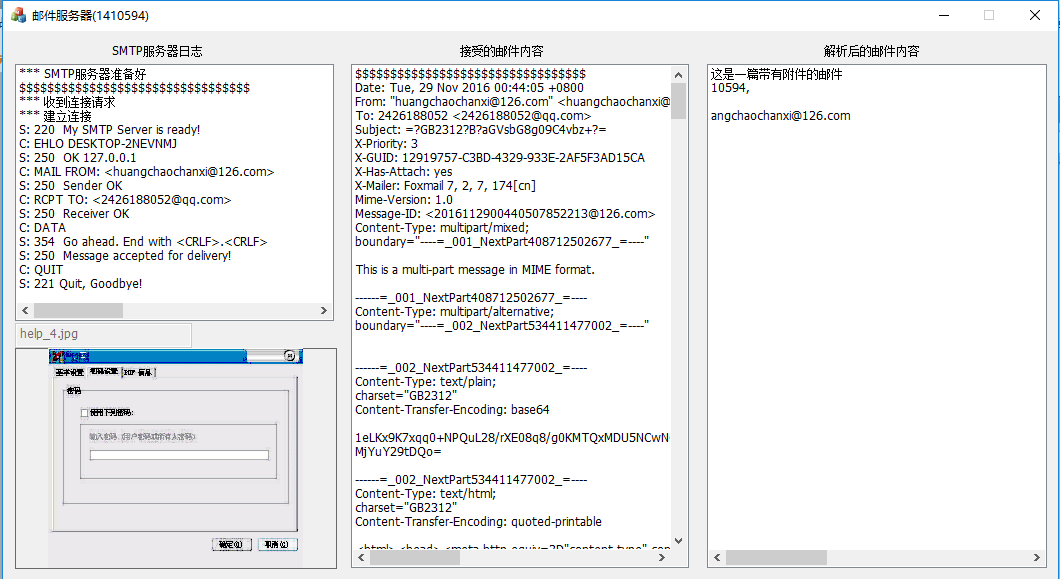
**c）观察带有图片附件的邮件发送**

**注：由于是MFC编写的，因此在显示图片的时候如果图片过大会有延迟。此处使用的图片只有21KB**

用Foxmail的126用户代理发送如下邮件



点击发送，服务器端显示如下



**d）实验原理**

此部分主要是邮件客户端与服务器建立通讯的过程，在此程序中主要是简单邮件传输协议 SMTP 的实现：

邮件内容的显示与处理按照’\r\n’分隔开每一行，当遇到’\r\n.\r\n’时结束邮件内容的显示。

将邮件内容中邮件正文显示出来时，当在邮件中遇到第一个Content-Transfer-Encoding: base64时作为邮件正文 base64 编码的开始处，接着在其后第一次遇到------时作为邮件正文 base64 编码的结束处。将其中间字符串截取出来进行 base64 解码，然后显示到该区域。其中 base64 编码规则如下：

 1.将 3 个字节转换成 4 个可打印字符

2.将每 3 个字节做为一个整体将其划分为 4 组，每组 6 位

3.将 6 位的值作为索引，映射为对应的可打印 ASCII 字符

 原始文件尾部处理

4.剩 1 个字节：后面补 4 个比特的“0”，再分成 2 个 6 位组，映射为 2 个ASCII 字符，而后再填充两个“=”

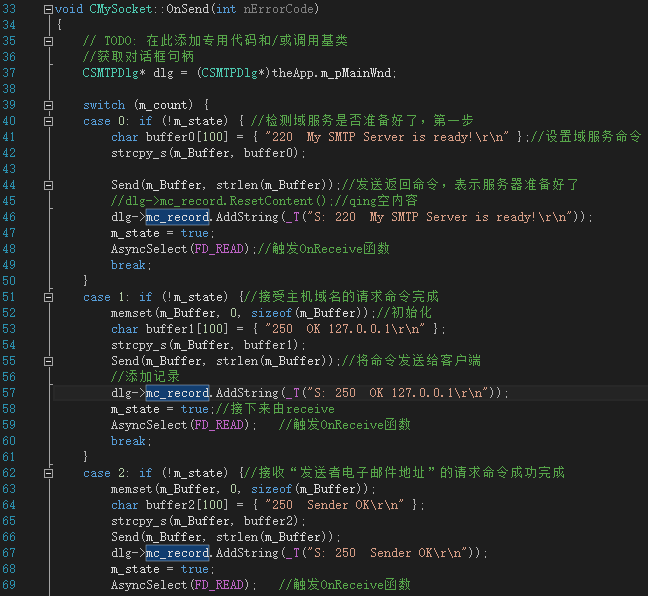
5.剩 2 个字节：后面补 2 个比特的“0”，再分成 3 个 6 位组，映射为 3 个ASCII 字符，而后再填充 1 个“=”

 6.添加回车换行：变换后，每 76 个字符后增加一回车换行

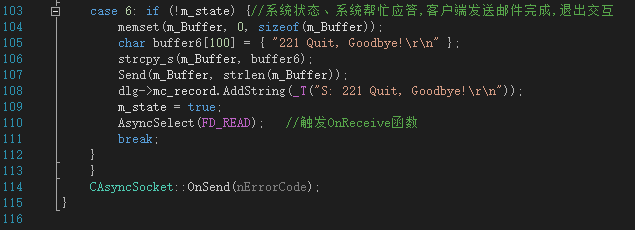
附件图片内容的获取是根据邮件中遇到第一个filename时作为邮件附件图片 base64 编码的开始处，接着在其后第一次遇到------时作为邮件附件图片 base64 编码的结束处。将其中间字符串截取出来进行 base64 解码，写入磁盘文件流，然后使用 CImage 类函数调用显示图片到该区域。

**程序截图如下：**

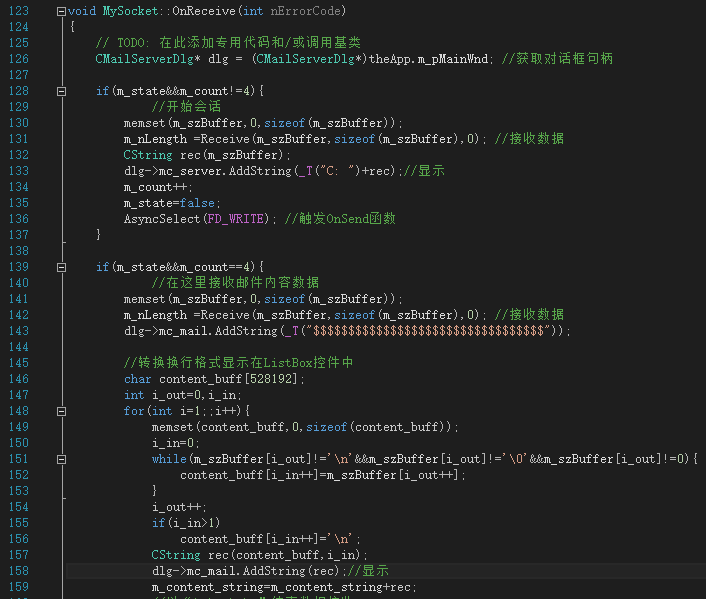
**OnSend（）//发送函数**

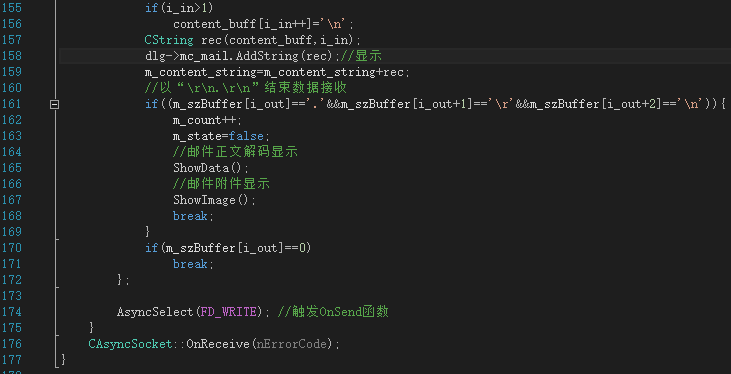




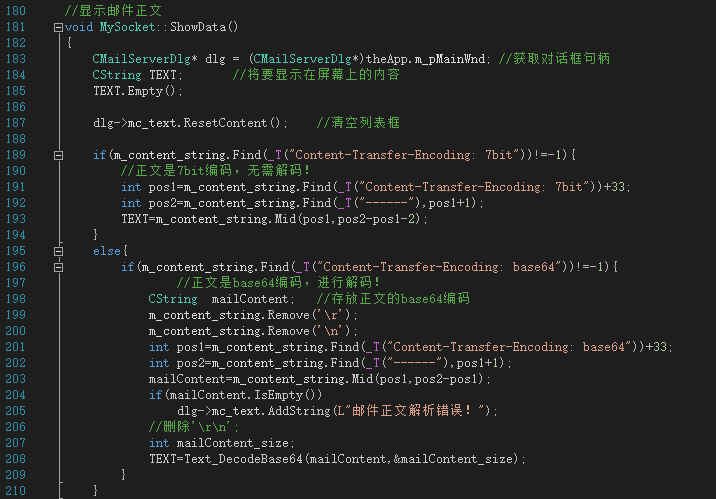


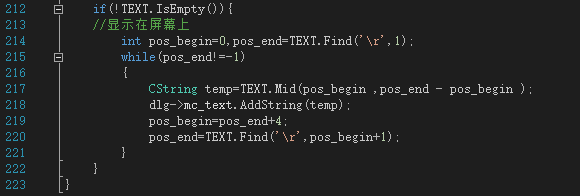
**OnReceive（）//接收函数**





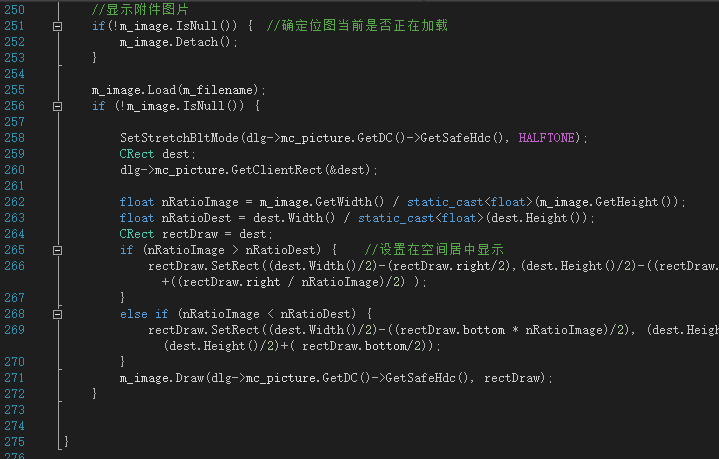
**ShowData()//显示邮件正文**





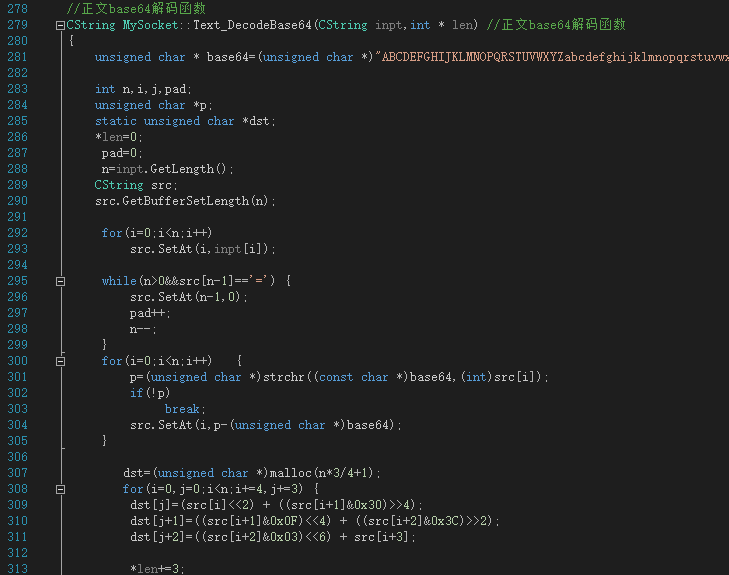
**ShowImage()//显示附件图像**

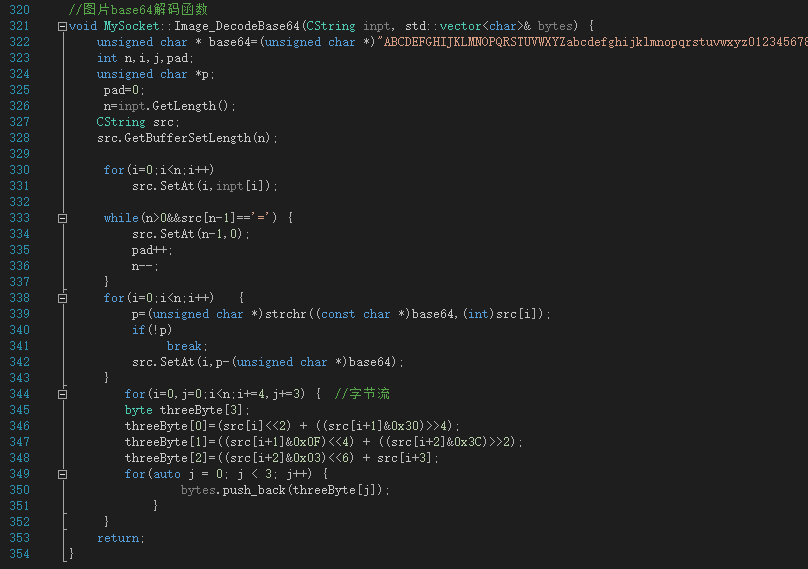




**Image\_DecodeBase64（）//附件图像的base64解码算法**

**Txt\_DecodeBase64//正文base64解码函数**





**C.实验总结**

有了前面两次的基础。此次SMTP服务器的编写比较顺利。这次实验让我清晰地认识到了SMTP服务器是怎么与一个邮件客户端交互的。也对自己平时发邮件的背后的机制有了更深刻的了解。提高了我对编程的兴趣。