华东师范大学数据科学与工程学院实验报告

课程名称：计算机网络与编程　　**年级：**2021 级　　**上机实践成绩：**

指导教师：张召　　**姓名：**彭一珅　　**学号：**10215501412

上机实践名称：HTTP、SMTP、POP3 协议分析　　**上机实践日期：**2023.4.7

上机实践编号：6　　**组号：**　　**上机实践时间：**9:50

一、 实验目的

熟悉 HTTP 协议的工作原理

了解 HTTP 协议在实际网络中的运行过程

熟悉 SMTP 和 POP3 协议的工作原理

了解 SMTP 和 POP3 协议在实际网络中的运行过程

二、 实验任务

通过 Wireshark 分析 HTTP 协议

通过 Wireshark 分析 SMTP 和 POP3 协议

三、 使用环境

Windows11

Wireshark

四、 实验过程

3.1 预备知识

3.2 HTTP 数据包抓取及分析

task1: 利用 Wireshark 抓取一条 HTTP 请求网络包，分析 HTTP 请求网络包的组成（要求根据报文结构正确标识每个部分)，请将实验结果附在实验报告中。

使用过滤器 http，访问网站华文慕课 http://www.chinesemooc.org/，用 wireshark 抓包结果如下：

http

No.	Time	Source	Destination	Protocol	Length	Info
1331	1.447536	58.205.209.183	172.30.187.81	HTTP	617	HTTP/1.1 200 OK (PNG)
1334	1.447767	172.30.187.81	58.205.209.183	HTTP	469	GET /static/index/img/home-section-tit4.png HTTP/1.1
1341	1.452142	58.205.209.183	172.30.187.81	HTTP	427	HTTP/1.1 200 OK (PNG)
1373	1.452489	58.205.209.183	172.30.187.81	HTTP	454	HTTP/1.1 200 OK (PNG)
1388	1.452516	172.30.187.81	58.205.209.183	HTTP	469	GET /static/index/img/home-section-tit5.png HTTP/1.1
1391	1.452489	58.205.221.200	172.30.187.81	HTTP	217	HTTP/1.1 200 OK (PNG)
1416	1.452774	172.30.187.81	58.205.209.183	HTTP	466	GET /static/common/img/header-logo.png HTTP/1.1
1419	1.452842	172.30.187.81	58.205.209.183	HTTP	475	GET /static/widget/footer/img/footer-sns-wx.png HTTP/1.1
1431	1.453074	172.30.187.81	58.205.209.183	HTTP	475	GET /static/widget/footer/img/footer-sns-wb.png HTTP/1.1
1585	1.484621	182.92.233.49	172.30.187.81	HTTP	793	HTTP/1.1 200 OK (JPEG JFIF image)
1668	1.495674	58.205.209.183	172.30.187.81	HTTP	389	[TCP Previous segment not captured] Continuation
1671	1.495674	58.205.209.183	172.30.187.81	HTTP	833	[TCP Previous segment not captured] Continuation
1681	1.495674	58.205.221.200	172.30.187.81	HTTP	374	HTTP/1.1 200 OK (PNG)
1691	1.496277	172.30.187.81	58.205.209.183	HTTP	458	GET /static/dep/slick/ajax-loader.gif HTTP/1.1
1692	1.496549	172.30.187.81	58.205.209.183	HTTP	599	GET /static/widget/teacher_list/img/teachers-staff-icon1.png HTTP/1.1
1695	1.502349	58.205.209.183	172.30.187.81	HTTP	522	[TCP Previous segment not captured] Continuation
1696	1.502349	58.205.209.183	172.30.187.81	HTTP	660	HTTP/1.1 200 OK (PNG)
1698	1.502349	58.205.209.183	172.30.187.81	HTTP	363	[TCP Previous segment not captured] Continuation
1721	1.502731	172.30.187.81	58.205.209.183	HTTP	599	GET /static/widget/teacher_list/img/teachers-staff-icon3.png HTTP/1.1
1726	1.502977	172.30.187.81	58.205.209.183	HTTP	429	GET /images/new_edition/huawen.jpg HTTP/1.1
1727	1.503060	172.30.187.81	58.205.209.183	HTTP	426	GET /images/email/find_pass.png HTTP/1.1
1920	1.537463	58.205.209.183	172.30.187.81	HTTP	202	[TCP Previous segment not captured] Continuation
1922	1.537463	58.205.209.183	172.30.187.81	HTTP	987	HTTP/1.1 200 OK (PNG)
1925	1.537463	58.205.209.183	172.30.187.81	HTTP	1005	HTTP/1.1 200 OK (JPEG JFIF image)
2416	1.716795	172.30.187.81	182.92.233.49	HTTP	524	GET /favicon.ico HTTP/1.1
2417	1.747275	58.205.209.183	172.30.187.81	HTTP	1371	HTTP/1.1 200 OK (JPEG JFIF image)
2441	1.781871	182.92.233.49	172.30.187.81	HTTP	820	HTTP/1.1 200 OK (image/x-icon)

点击请求图标 ico 的包，观察报文结构如图所示：

其中，请求方法为 GET，URL 为/favicon.ico，也就是对应的 api 中图标保存的路径。协议版本为 HTTP/1.1。

接下来包含了几个头部字段，以键值对的方式保存，每行后跟着回车符和换行符，这些字段分别是：

Host（设置服务器域名和 TCP 端口号，以网址的形式呈现，与端口号用:分隔，此处网址为 **www.chinesemooc.org**），

User-Agent（用户代理的字符串值，此处表示我使用的 Mozilla Firefox 浏览器），

Accept（表示接受的内容类型，此处是图片形式 **image/avif,image/webp,*/***），Accept-Language（表示接受的语言，zh-CN 代表中文），

Accept-Encoding（接受的编码形式，此处的 **gzip, deflate** 表示浏览器支持 gzip 压缩方式），

Connection（它决定了客户端和服务器进行了一次会话后，服务器是否立即关闭网络连接，可以取值为 close 和 keep-alive，如果控制不再转发给代理的首部字段，则设为 Upgrade），

Referer（包含一个 URL，用户从该 URL 代表的页面出发访问当前请求的页面，一般是网页的首页），

Cookie（用于告知服务器当客户端想获得 HTTP 状态管理支持，包含从服务器接收到的 Cookie，可有多个）。

回车符和换行符后，是响应正文。这个请求的响应正文包括一个 Full request URI，打开发现是网页 logo 的图标：

Http request 2/2 表示这个包是第二个包，这次发送请求总共有 2 个包；Prev request in frame 表示了上一个请求的帧为 296，Response in frame 表示回复这条报文的帧为 2441.

task2: 利用 Wireshark 找到上述请求网络包相对应的 HTTP 响应网络包，然后对比分析两个网络包的组成，请在实验报告中说明两者之间的区别。

对应的响应网络包为：

http 的 Response 报文如图所示：

```
> Frame 2441: 820 bytes on wire (6560 bits), 820 bytes captured (6560 bits) on interface \Device\NPF_{77D02574-1C20-4EC1-9BD5-DC7869F5B097}, id 0
> Ethernet II, Src: NewH3CTe_7b:38:02 (54:c6:ff:7b:38:02), Dst: IntelCor_2d:c7:9c (44:e5:17:2d:c7:9c)
> Internet Protocol Version 4, Src: 182.92.233.49, Dst: 172.30.187.81
> Transmission Control Protocol, Src Port: 80, Dst Port: 60841, Seq: 427791, Ack: 848, Len: 766
> [6 Reassembled TCP Segments (7666 bytes): #2418(1380), #2419(1380), #2420(1380), #2421(1380), #2427(1380), #2441(766)]
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Server: nginx\r\n
    Date: Fri, 07 Apr 2023 02:28:17 GMT\r\n
    Content-Type: image/x-icon\r\n
  > Content-Length: 7358\r\n
    Last-Modified: Sun, 28 Mar 2021 12:40:40 GMT\r\n
    Connection: keep-alive\r\n
    ETag: "60607948-1cbe"\r\n
    Expires: Sun, 07 May 2023 02:28:17 GMT\r\n
    Cache-Control: max-age=2592000\r\n
    Accept-Ranges: bytes\r\n
    \r\n
    [HTTP response 2/2]
    [Time since request: 0.065076000 seconds]
    [Prev request in frame: 296]
    [Request in frame: 2416]
    [Request URI: http://www.chinesemooc.org/favicon.ico]
    File Data: 7358 bytes
v Media Type
    Media type: image/x-icon (7358 bytes)
```

其中第一行是协议版本 HTTP/1.1，与请求报文协议相同。状态码为 200，通过状态码描述 OK 告诉我们请求成功。

响应报文的头部信息也有所不同，

Server 代表服务器，

Date 代表当天日期，

Content-Type（表示发送给客户端的数据类型，此处为图片 icon），

Content-Length（表示响应主体的字节数），

Last-Modified（表示网页制作者上次修改该图标的时间），

Connection 与请求时相同，表示连接方式，

Etag（与 last-modified 作用相同，当文件被修改时这个值也会改变，用于判断这个文件是否被修改过），

Expires（告诉客户端在这个时间前，可以直接访问缓存副本），

Cache-Control（max-age=2592000 是通知浏览器 2592000 秒之内不要再从网页里获取，自己从缓冲区中刷新），

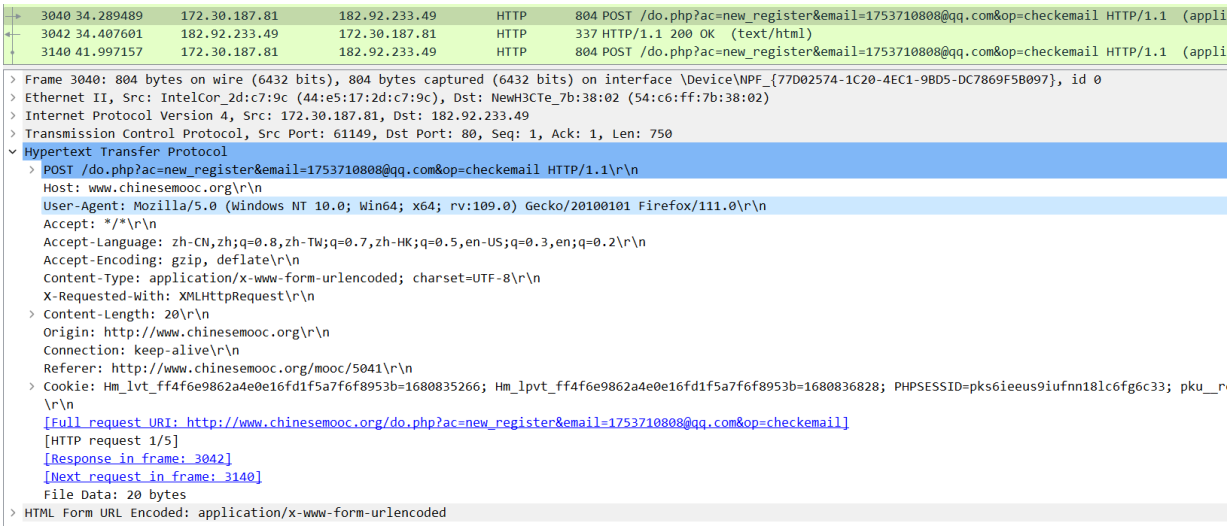
Accept-Ranges（允许客户端以 bytes 的方式获得数据）。

正文中 time since request 表示过了多长时间对请求进行的回复。也标注了上一个响应的帧位置和所回应的请求帧位置。

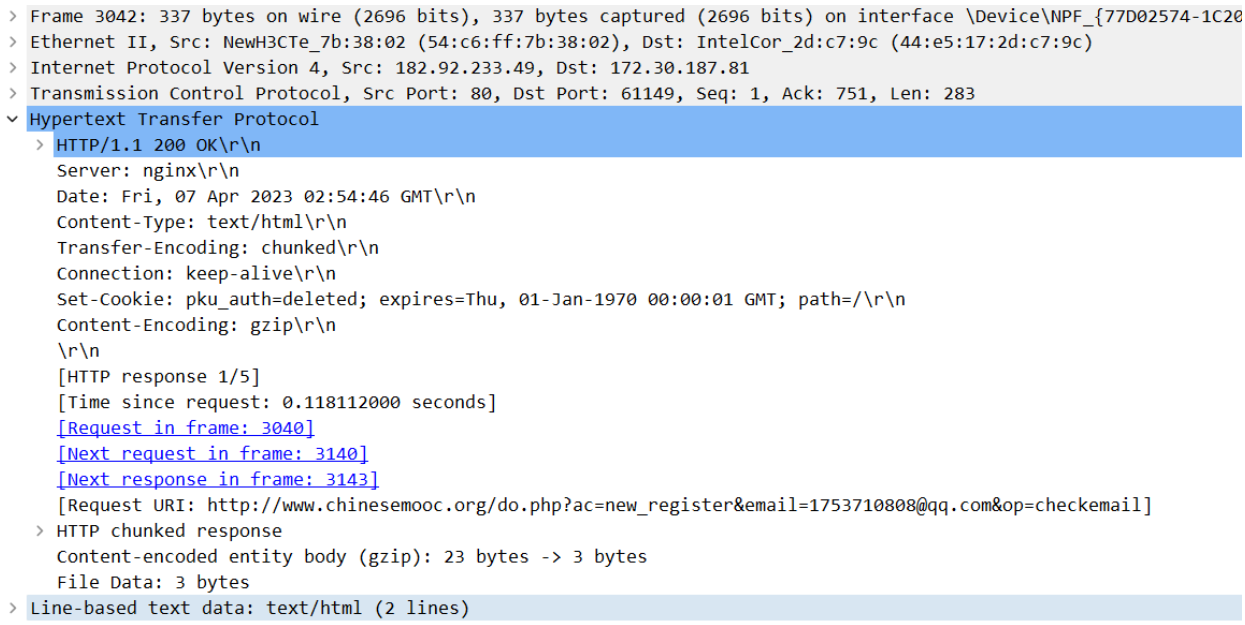
请求和响应的第一行不同，而头信息的种类也有所不同，但是种类相同时，内容也是相匹配的。

task3: 学习了解 GET 和 POST 方法，请在实验报告中分析对比 GET 和 POST 方法的请求报文，以及 GET 和 POST 方法的和响应报文之间的区别。

点击注册，输入账号密码，发现浏览器发送如下 POST 请求：



可以看到 POST 读取了我的邮箱内容，但是没有像 get 请求一样放入 url 里。



这是 POST 请求的响应包，可以看到它具有 body 包体。

所以，get 和 post 的区别有这样几条：

1. GET 请求的数据会附在 URL 之后，以?分割 URL 和传输数据，参数之间以&相连，POST 把提交的数据则放置在是 HTTP 包的包体中。因此，POST 请求必须有请求体，而且传输数据没有大小限制。

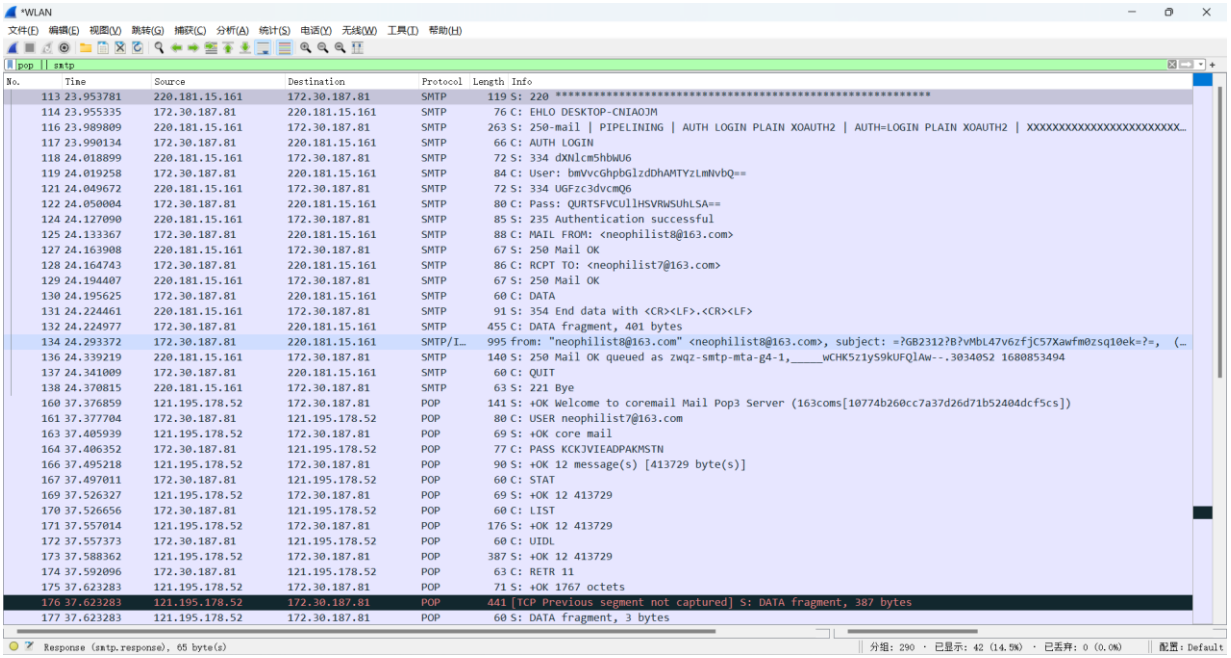
2. POST 的数据不进行缓存，而 GET 将数据缓存在浏览器中，一段时间内刷新页面，不会重复发送请求。

3. POST 请求一般出现在网页进行表单提交的时候，将用户输入的信息传给服务器，或执行 ajax 请求等，而获取网页数据一般使用 GET。

3.3 SMTP 和 POP3 数据包抓取及分析

task4: 利用 Wireshark 抓取 SMTP 和 POP3 网络包，分析 SMTP 和 POP3 数据包组成（要求根据报文结构正确标识每个部分），请将实验结果附在实验报告中。

在 Foxmail 中发送邮件，抓包结果如图所示：



其中，SMTP 的报文中，由 Server 发送的包是 response 及其内容，由 Client 发送的内容一般是 Command Line，或者与服务器通信的相关数据。

POP 的报文中，服务器发送的信息为 Response indicator 和 Response description

```
Post Office Protocol
  +OK core mail\r\n
      Response indicator: +OK
      Response description: core mail
```

用户发送的信息是 Request command 命令和所携带的参数

```
Post Office Protocol
  RETR 11\r\n
      Request command: RETR
      Request parameter: 11
```

task5: 利用 Wireshark 抓取 SMTP 网络包，分析一个在 SMTP 客户（C）和 SMTP 服务器（S）之间交换报文文本的例子（参考书本 p77-78)，请将实验结果附在实验报告中。

借用 task4 所发送的邮件为例，SMTP 报文内容如下：

```
119 S: 220 ************************************************************
 76 C: EHLO DESKTOP-CNIAOJM
263 S: 250-mail | PIPELINING | AUTH LOGIN PLAIN XOAUTH2 | AUTH=LOGIN PLAIN XOAUTH2 | XXXXXXXXXXXXXXXXXXXXXXXX…
 66 C: AUTH LOGIN
 72 S: 334 dXNlcm5hbWU6
 84 C: User: bmVvcGhpbGlzdDhAMTYzLmNvbQ==
 72 S: 334 UGFzc3dvcmQ6
 80 C: Pass: QURTSFVCUllHSVRWSUhLSA==
 85 S: 235 Authentication successful
 88 C: MAIL FROM: <neophilist8@163.com>
 67 S: 250 Mail OK
 86 C: RCPT TO: <neophilist7@163.com>
 67 S: 250 Mail OK
 60 C: DATA
 91 S: 354 End data with <CR><LF>.<CR><LF>
455 C: DATA fragment, 401 bytes
995 from: "neophilist8@163.com" <neophilist8@163.com>, subject: =?GB2312?B?vMbL47v6zfjC57Xawfm0zsq10ek=?=,  (…
140 S: 250 Mail OK queued as zwqz-smtp-mta-g4-1,_____wCHK5z1yS9kUFQlAw--.30340S2 1680853494
 60 C: QUIT
 63 S: 221 Bye
```

首先，状态字 220 表示连接服务器成功，这是在 TCP 三次握手后建立连接后，服务器发来的回应。

```
> Frame 113: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface \Device\NPF_{77D02574-1C20-4EC1-9BD5-DC7869F5B09
> Ethernet II, Src: NewH3CTe_7b:38:02 (54:c6:ff:7b:38:02), Dst: IntelCor_2d:c7:9c (44:e5:17:2d:c7:9c)
> Internet Protocol Version 4, Src: 220.181.15.161, Dst: 172.30.187.81
> Transmission Control Protocol, Src Port: 25, Dst Port: 65306, Seq: 1, Ack: 1, Len: 65
v Simple Mail Transfer Protocol
  v Response: 220 ************************************************************\r\n
      Response code: <domain> Service ready (220)
      Response parameter: ************************************************************
```

服务端返回 220 代码后，客户端继续发送请求，首先是 EHLO 命令，参数为主机名，用于验证身份。服务器发送 250OK 代码后，向客户端要求验证方式。

```
> Internet Protocol Version 4, Src: 220.181.15.161, Dst: 172.30.187.81
> Transmission Control Protocol, Src Port: 25, Dst Port: 65306, Seq: 66, Ack: 23, Len: 209
v Simple Mail Transfer Protocol
  v Response: 250-mail\r\n
      Response code: Requested mail action okay, completed (250)
      Response parameter: mail
      Response parameter: PIPELINING
      Response parameter: AUTH LOGIN PLAIN XOAUTH2
      Response parameter: AUTH=LOGIN PLAIN XOAUTH2
      Response parameter: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXA
      Response parameter: STARTTLS
      Response parameter: XB
      Response parameter: 8BITMIME
```

客户端使用 AUTH 命令进行身份验证，携带参数为 LOGIN。服务器 334 代码表示接受用户登录。

```
> Frame 118: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface \Device\NPF_{77D02574-1C20-4EC1-9BD
> Ethernet II, Src: NewH3CTe_7b:38:02 (54:c6:ff:7b:38:02), Dst: IntelCor_2d:c7:9c (44:e5:17:2d:c7:9c)
> Internet Protocol Version 4, Src: 220.181.15.161, Dst: 172.30.187.81
> Transmission Control Protocol, Src Port: 25, Dst Port: 65306, Seq: 275, Ack: 35, Len: 18
v Simple Mail Transfer Protocol
  v Response: 334 dXNlcm5hbWU6\r\n
      Response code: AUTH input (334)
      Response parameter: dXNlcm5hbWU6
```

输入用户名。

```
> Frame 119: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface \Device\NPF_{77D02574-1
> Ethernet II, Src: IntelCor_2d:c7:9c (44:e5:17:2d:c7:9c), Dst: NewH3CTe_7b:38:02 (54:c6:ff:7b:38:02)
> Internet Protocol Version 4, Src: 172.30.187.81, Dst: 220.181.15.161
> Transmission Control Protocol, Src Port: 65306, Dst Port: 25, Seq: 35, Ack: 293, Len: 30
v Simple Mail Transfer Protocol
    Username: bmVvcGhpbGlzdDhAMTYzLmNvbQ==
```

然后以相同的步骤输入密码。服务器发送 235 表示登陆成功：

```
> Frame 124: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface \Device\NPF_{77D02574-1C20-4EC1-9
> Ethernet II, Src: NewH3CTe_7b:38:02 (54:c6:ff:7b:38:02), Dst: IntelCor_2d:c7:9c (44:e5:17:2d:c7:9c)
> Internet Protocol Version 4, Src: 220.181.15.161, Dst: 172.30.187.81
> Transmission Control Protocol, Src Port: 25, Dst Port: 65306, Seq: 311, Ack: 91, Len: 31
v Simple Mail Transfer Protocol
  v Response: 235 Authentication successful\r\n
      Response code: Authentication successful (235)
      Response parameter: Authentication successful
```

这一部分向服务器确认报文的发送者与接收者，用户必须存在。

```
SMTP        88 C: MAIL FROM: <neophilist8@163.com>
SMTP        67 S: 250 Mail OK
SMTP        86 C: RCPT TO: <neophilist7@163.com>
SMTP        67 S: 250 Mail OK
```

用户用 DATA 命令请求输入邮件正文，服务器回应可以 input 了

```
v Simple Mail Transfer Protocol
  v Response: 354 End data with <CR><LF>.<CR><LF>\r\n
      Response code: Start mail input; end with <CRLF>.<CRLF> (354)
      Response parameter: End data with <CR><LF>.<CR><LF>
```

接下来可以看到邮件正文：

```
v Internet Message Format
    Date: Fri, 7 Apr 2023 15:44:54 +0800
  > From: "neophilist8@163.com" <neophilist8@163.com>, 1 item
  > To: neophilist7 <neophilist7@163.com>, 1 item
    Subject: =?GB2312?B?vMbL47v6zfjC57Xawfm0zsq10ek=?=
  > Unknown-Extension: X-Priority: 3 (Contact Wireshark developers if you want this supported.)
  > Unknown-Extension: X-Has-Attach: no (Contact Wireshark developers if you want this supported.)
    X-Mailer: Foxmail 7.2.25.213[cn]
    MIME-Version: 1.0
    Message-ID: <2023040715445379472 36@163.com>
  > Content-Type: multipart/alternative;\r\n\tboundary="----=_001_NextPart021437651655_=----"
  > MIME Multipart Media Encapsulation, Type: multipart/alternative, Boundary: "----=_001_NextPart021437651655_=----"
```

对邮件正文追踪流，对内容进行解密：

```
This is a multi-part message in MIME format.

------=_001_NextPart021437651655_=----
Content-Type: text/plain;
        charset="GB2312"
Content-Transfer-Encoding: base64

dGVzdA0KDQoNCg0KbmVvcGhpbGlzdDhAMTYzLmNvbQ0K

------=_001_NextPart021437651655_=----
Content-Type: text/html;
        charset="GB2312"
Content-Transfer-Encoding: quoted-printable
```

可以看到邮件所发送的信息。

收到正文信息后，服务器返回 250 消息，然后用户执行 QUIT 指令，断开连接。

```
60 C: QUIT
63 S: 221 Bye
```

五、总结

HTTP 协议(超文本传输协议 HyperText Transfer Protocol)是基于 TCP 协议的应用层传输协议。

SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议，用于控制信件的中转方式。SMTP 协议属于 TCP/IP 协议簇，它帮助每台计算机在发送或中转信件时找到下一个目的地。

POP3（Post Office Protocol version3）协议协助用户代理（即客户端）从邮件服务器上获取邮件。POP3 允许用户从服务器上把邮件存储到本地主机（即自己的计算机）上，同时删除保存在邮件服务器上的邮件。

其中，SMTP 和 HTTP 有一些重要的区别。HTTP 的 TCP 连接是由想接受文件的机器发起的，而 SMTP 的 TCP 连接是由要发送该文件的机器发起的。第二，SMTP 要求每个报文采用 7 比特 ASCII 码格式。第三，HTTP 把每个对象封装到它自己的 HTTP 响应报文中，而 SMTP 则把所有报文对象放在一个报文之中。

对于 smtp 和 pop 包的获取，需要将 ssl 连接断开，然后使用代理客户端发送邮件。