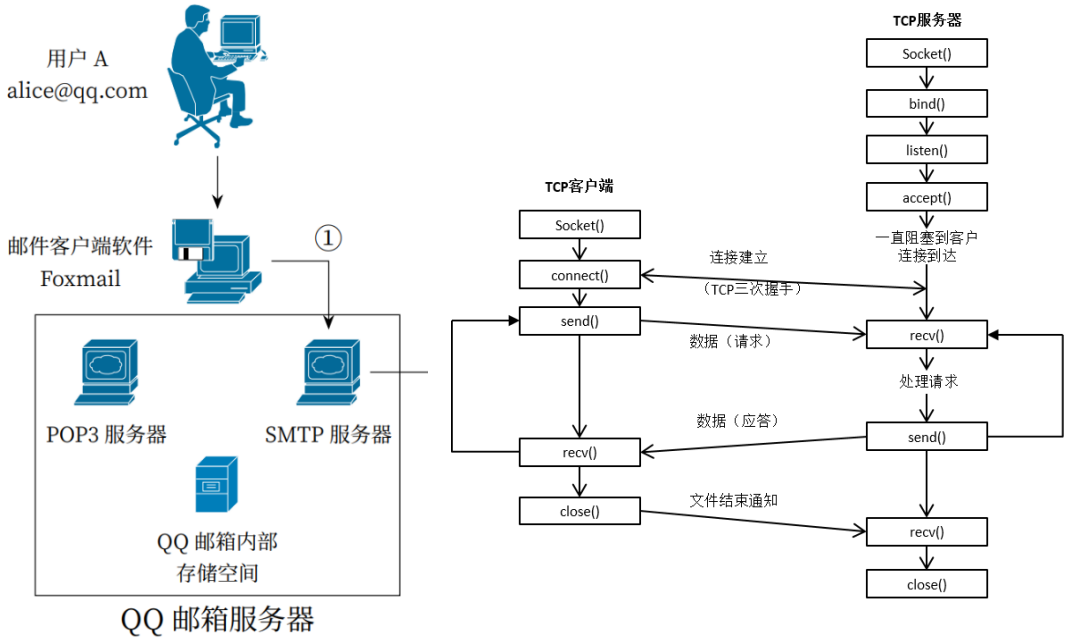
一、 实验详细设计

（注意不要完全照搬实验指导书上的内容，请根据你自己的设计方案来填写图文并茂地描述实验实现的所有功能和详细的设计方案及实验过程中的特色部分。）

1. 邮件发送客户端详细设计



客户端与 SMTP 服务器通过使用 TCP 协议的流式 socket（SOCK_STREAM）来进行通信。可由指导书上的理论讲解，以及 telnet 连接的例子充分理解通信过程。

此外，发信的过程中使用了 MIME 来拓展电子邮件的内容，以支持非 ASCII 码消息。

实验框架给出的收发消息的 socket 函数过于简陋，故而进行了包装：

```
/*
 * @brief 自定义发信方法，封装了 socket 的 send，并打印己方发送的信息。
 * 多了一个参数 error_msg 是附上错误信息，用于定位
 *
 * @param sockfd socket 函数返回的套接字描述符
 * @param buf 发送数据的缓冲区指针
 * @param len 缓冲区的大小
 * @param flags 一般取 0
 * @error_msg 错误信息，用于定位
 */
int cus_send(int sockfd, void *buf, int len, int flags, char *error_msg)
{
    int ret = -1;
    if ((ret = send(sockfd, buf, len, flags)) == -1)
    {
        perror(error_msg);
        exit(EXIT_FAILURE);
```

```
    }

    printf("%s\r\n", (char *)buf);

    return ret;
}

/*
 * @brief 自定义收信方法，封装了 socket 的 recv，并打印接收的信息。
 * 多了一个参数 error_msg 是附上错误信息，用于定位
 *
 * @param sockfd socket 函数返回的套接字描述符
 * @param buf 发送数据的缓冲区指针
 * @param len 缓冲区的大小
 * @param flags 一般取 0
 * @error_msg 错误信息，用于定位
 */
int cus_recv(int sockfd, void *buf, int len, int flags, char *error_msg)
{
    int r_size = -1;
    if ((r_size = recv(sockfd, buf, len, 0)) == -1)
    {
        perror(error_msg);
        exit(EXIT_FAILURE);
    }
    char *tmp_buf = (char *)buf;
    tmp_buf[r_size] = '\0'; // Do not forget the null terminator
    printf("%s", tmp_buf);
    return r_size;
}
```

关于邮件正文以及附件的字符串的处理，也包装了以下函数：

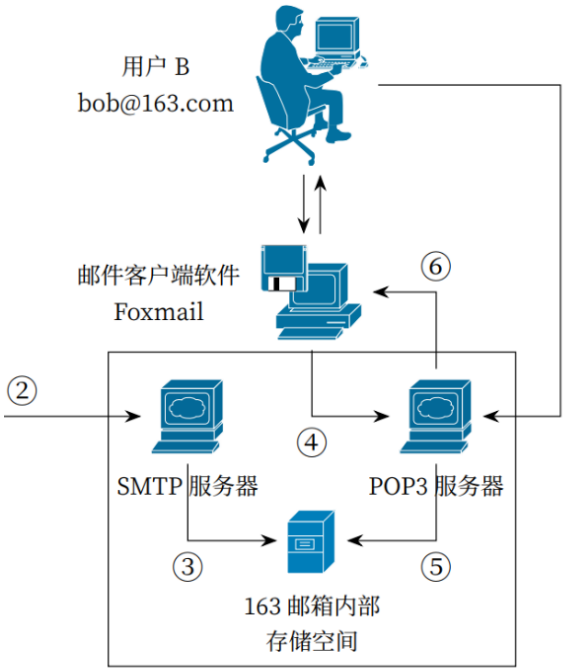
```
/*
 * @brief 把文本文件转换为字符串
 *
 * @param path 文本文件地址
 */
char *file2str(const char *path)
{
    FILE *fp = fopen(path, "r");
    fseek(fp, 0, SEEK_END);
    int fplen = ftell(fp);
    fseek(fp, 0, SEEK_SET);
    char *content = (char *)malloc(fplen);
    fread(content, 1, fplen, fp);
```

```
    fclose(fp);
    return content;
}

/*
 * @brief 从已有文件转换为 base64 文件
 *  转换的文件名都为 attach_base64
 *
 * @param path 文本文件地址
 */
int gen_base64_file(const char *path)
{
    FILE *fp = fopen(path, "r");
    if (fp == NULL)
        return -1;

    FILE *fp_base64 = fopen("attach_base64", "w");
    encode_file(fp, fp_base64);
    fclose(fp);
    fclose(fp_base64);
    return 0;
}
```

2. 邮件接收客户端详细设计



163 邮箱服务器

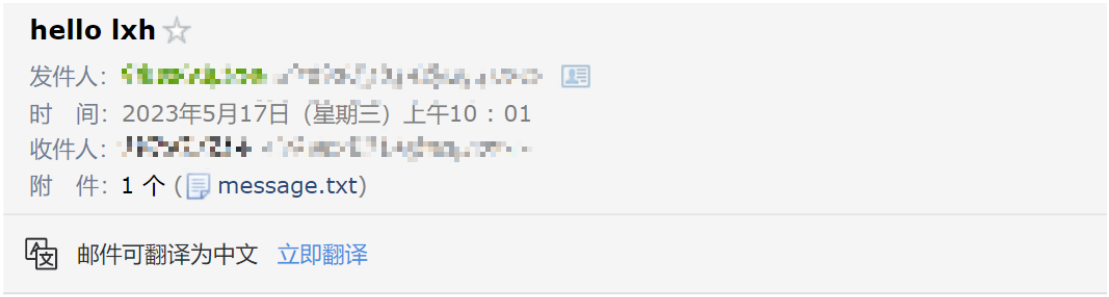
客户端与 POP3 服务器通过使用 TCP 协议的流式 socket（SOCK_STREAM）来进行通信。可由指导书上的理论讲解，以及 telnet 连接的例子充分理解通信过程。

复用了 send.c 中的 cus_send 和 cus_recv 函数。相比于 send.c 简单了很多。

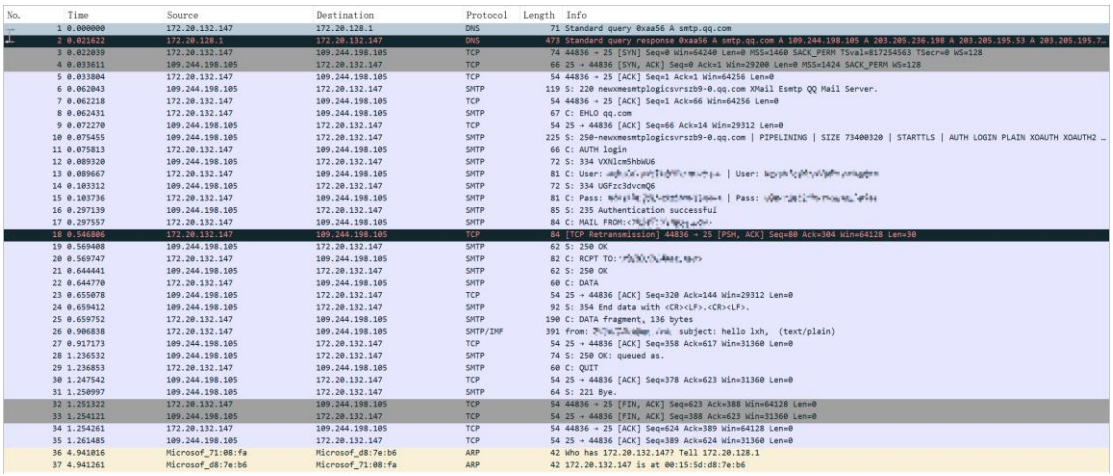
二、 实验结果截图及分析

（对你自己实验的测试结果进行评价）

1. 邮件发送客户端实验结果及分析



成功接收。



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.132.147	172.20.128.1	DNS	71	Standard query 0xaa56 A smtp.qq.com
2	0.021622	172.20.128.1	172.20.132.147	DNS	473	Standard query response 0xaa56 A smtp.qq.com A 109.244.198.105 A 203.205.236.198 A 203.205.195.53 A 203.205.195.7…
3	0.022039	172.20.132.147	109.244.198.105	TCP	74	44836 → 25 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=817254563 TSecr=0 WS=128
4	0.033611	109.244.198.105	172.20.132.147	TCP	66	25 → 44836 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1424 SACK_PERM WS=128
5	0.033804	172.20.132.147	109.244.198.105	TCP	54	44836 → 25 [ACK] Seq=1 Ack=1 Win=64256 Len=0
6	0.062043	109.244.198.105	172.20.132.147	SMTP	119	S: 220 newxmesmtplogicsvrszb9-0.qq.com XMail Esmtp QQ Mail Server.
7	0.062218	172.20.132.147	109.244.198.105	TCP	54	44836 → 25 [ACK] Seq=1 Ack=66 Win=64256 Len=0
8	0.062431	172.20.132.147	109.244.198.105	SMTP	67	C: EHLO qq.com
9	0.072270	109.244.198.105	172.20.132.147	TCP	54	25 → 44836 [ACK] Seq=66 Ack=14 Win=29312 Len=0
10	0.075455	109.244.198.105	172.20.132.147	SMTP	225	S: 250-newxmesmtplogicsvrszb9-0.qq.com | PIPELINING | SIZE 73400320 | STARTTLS | AUTH LOGIN PLAIN XOAUTH XOAUTH2 …
11	0.075813	172.20.132.147	109.244.198.105	SMTP	66	C: AUTH login
12	0.089320	109.244.198.105	172.20.132.147	SMTP	72	S: 334 VXNlcm5hbWU6
13	0.089667	172.20.132.147	109.244.198.105	SMTP	81	C: User: [redacted] | User: [redacted]
14	0.103312	109.244.198.105	172.20.132.147	SMTP	72	S: 334 UGFzc3dvcmQ6
15	0.103736	172.20.132.147	109.244.198.105	SMTP	81	C: Pass: [redacted] | Pass: [redacted]
16	0.297139	109.244.198.105	172.20.132.147	SMTP	85	S: 235 Authentication successful
17	0.297557	172.20.132.147	109.244.198.105	SMTP	84	C: MAIL FROM:<[redacted]>
18	0.546806	172.20.132.147	109.244.198.105	TCP	54	[TCP Retransmission] 44836 → 25 [PSH, ACK] Seq=80 Ack=304 Win=64128 Len=30
19	0.569408	109.244.198.105	172.20.132.147	SMTP	62	S: 250 OK
20	0.569747	172.20.132.147	109.244.198.105	SMTP	82	C: RCPT TO:<[redacted]>
21	0.644441	109.244.198.105	172.20.132.147	SMTP	62	S: 250 OK
22	0.644770	172.20.132.147	109.244.198.105	SMTP	60	C: DATA
23	0.655078	109.244.198.105	172.20.132.147	TCP	54	25 → 44836 [ACK] Seq=320 Ack=144 Win=29312 Len=0
24	0.659412	109.244.198.105	172.20.132.147	SMTP	92	S: 354 End data with <CR><LF>.<CR><LF>.
25	0.659752	172.20.132.147	109.244.198.105	SMTP	190	C: DATA fragment, 136 bytes
26	0.906838	172.20.132.147	109.244.198.105	SMTP/IMF	391	from: [redacted] subject: hello lxh, (text/plain)
27	0.917173	109.244.198.105	172.20.132.147	TCP	54	25 → 44836 [ACK] Seq=358 Ack=617 Win=31360 Len=0
28	1.236532	109.244.198.105	172.20.132.147	SMTP	74	S: 250 OK: queued as.
29	1.236853	172.20.132.147	109.244.198.105	SMTP	60	C: QUIT
30	1.247542	109.244.198.105	172.20.132.147	TCP	54	25 → 44836 [ACK] Seq=378 Ack=623 Win=31360 Len=0
31	1.250997	109.244.198.105	172.20.132.147	SMTP	64	S: 221 Bye.
32	1.251322	172.20.132.147	109.244.198.105	TCP	54	44836 → 25 [FIN, ACK] Seq=623 Ack=388 Win=64128 Len=0
33	1.254121	109.244.198.105	172.20.132.147	TCP	54	25 → 44836 [FIN, ACK] Seq=388 Ack=623 Win=31360 Len=0
34	1.254261	172.20.132.147	109.244.198.105	TCP	54	44836 → 25 [ACK] Seq=624 Ack=389 Win=64128 Len=0
35	1.261485	109.244.198.105	172.20.132.147	TCP	54	25 → 44836 [ACK] Seq=389 Ack=624 Win=31360 Len=0
36	4.941016	Microsof_71:08:fa	Microsof_d8:7e:b6	ARP	42	Who has 172.20.132.147? Tell 172.20.128.1
37	4.941261	Microsof_d8:7e:b6	Microsof_71:08:fa	ARP	42	172.20.132.147 is at 00:15:5d:d8:7e:b6

2. 邮件接收客户端实验结果及分析

成功接收。
总邮件个数及大小：

```
+OK
STAT

+OK 34 921144
```

每封邮件的编号及大小：

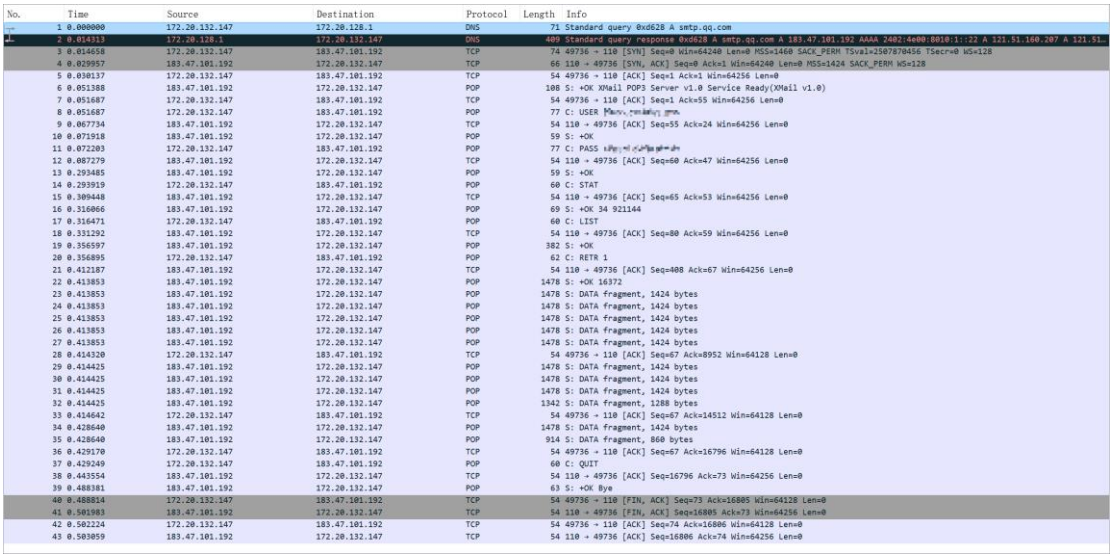
```
+OK
1 16372
2 42054
3 17489
4 21023
5 33183
6 137062
7 64004
8 32804
9 20640
10 43877
11 39353
12 30979
13 31377
14 31268
15 30598
16 10516
17 36591
18 146353
19 31202
20 31223
21 44281
22 8540
23 1369
24 1564
25 1508
26 1548
27 1564
28 1782
29 1822
30 1790
31 1818
32 1853
33 1853
34 1884
.
```

第一封邮件的内容：（节选）

```
RETR 1

+OK 16372
Received: from rn2-txn-msbadger06101.apple.com (rn2-txn-msbadger06101.apple.com [
17.111.110.96])
        by newxmmxsza11-4.qq.com (NewMX) with SMTP id 26A3AC23
        for <792972314@qq.com>; Tue, 18 Apr 2023 11:09:42 +0800
X-QQ-mid: xmmxza11-4t1681787382t9l5f7q3g
X-QQ-XMAILINFO: NWJExA51GmJJSA+NRi4O0MezgvUEf+7RQD/MfzQYQeufQ/m01x2l81eFlGFGaF
        JQ3lsngcHkNS8m7UisiNpn00BwPfRhXKTpE8ArXI2q+/FfeinzijcfeCZaZorLGG9LGGhtYJ
qZPY
        XBUI5xbFJ6EIDnbL2PtICOwfBmPg0G6nptryTlOWQRRM+Z8wNjncQ4WrQBQIDBxCTjoknfEp
UE+p
        5Eit9wbX6Fml1Wsum8vV9n4EAXaBs+IN/sfPqa/9rnewqSKny9hydFsWNryYRiANpalz3AgN
/WTG
        tLIZY3F6T6FlXvbL4ynWH0MJIpR02zDe5T1F6KQ1AkB2BdGUReP6utRVrmY1gUEl6NeBqQg3
XWAN
        0UKJDYo2NxQgA3C8m0YYEnJnKWCuBQXeBFnM5+7mzg3LqM+N24CEp4sWq1WUg6vfOmzDXws1
RLPa
        rxszcAnywaBLq11vWaLB67TXgD+oxCeM1tnJIsQHyGOoROhnh6jD5KOE6tee/UGbzbPNsgT2
5X8g
        zxC74V4rIRjy+bm7wWXAEnlKA5HdjWwxi1TzSuZ2vXz5W/WTw2aUn2K0fzcYu9DwTJF6UBqj
VlMA
        ez+tZgq2O5A+x5Lx+87FIwd2wQp6JvCCI+RRFGa5I1NPJ4t5feAPLawtGjV5d2xtq+mFX8ix
Os0p
        m9Ys9AsBF72+s7vQqRMya99lQA99Dj7PTB55Uq8N33yyOcLbUo5l0fCz6nGG6RAx8gS4LQaJ
h1HX
        93m9N2ViRXQ5GwYDtosQc7QRn6utOEhPmnsORzIjitFRcfkKhgLf1bYDuDFgxZL4lg/gd9i4
n11v
        5ClNhp0mx8m7RQVY3maz56bdNDJTQ+h6yEUY7ZSILimH6PGhWMKzUM9jbu2nDRNPvSk+pPYs
fgCs
        WdsbkSipqjcnmmoI4ODwpMqFOtiTo+wkdcIzPrKu5l6tnPmqHffp5lKTdNte4NrIfTZvLnUC
2HGq
        PBXrmevUIJnnSHyR3HDrr+QumxllTEoC+pTodXBzERfBWjFkp2ykdKgdZsSGSAxCwPZNu0F+
S15t
        v+GQFOpVL9Wgi1NVHMRhjlLQ+RblTkcQdR1Vl/XLUcCXJNXOWbpg6yjmAR/yjQhM9A6+pEZ1
aD/g
        0RVQcbtiFytIuVnCpeSGNI86ineVmuMAqw/dxHCW7iSiZfW+6ZjM02u4qaethZ3HY2MHzC84
LoHO
        GctAtZYvF0reUszpR2Prcqj56P8pQf5MS4Kirkh1PsRzgqkQKazxoafzzFKonkdMLOvuiyeq
JSin
        SElhlBRSTl0E16cVnBVVUuTP4/4bmqRuDX2+vs+ryxGfGBUlJIHVPG8Aw=
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=itunes.com;
        s=itunes0517; t=1681787380;
```

Pcap 文件：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.132.147	172.20.128.1	DNS	71	Standard query 0xd628 A smtp.qq.com
2	0.014313	172.20.128.1	172.20.132.147	DNS	409	Standard query response 0xd628 A smtp.qq.com A 183.47.101.192 AAAA 2402:4e00:8010:1::22 A 121.51.160.207 A 121.51…
3	0.014658	172.20.132.147	183.47.101.192	TCP	74	49736 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2507870456 TSecr=0 WS=128
4	0.029957	183.47.101.192	172.20.132.147	TCP	66	110 → 49736 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1424 SACK_PERM WS=128
5	0.030137	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [ACK] Seq=1 Ack=1 Win=64256 Len=0
6	0.051388	183.47.101.192	172.20.132.147	POP	108	S: +OK XMail POP3 Server v1.0 Service Ready(XMail v1.0)
7	0.051687	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [ACK] Seq=1 Ack=55 Win=64256 Len=0
8	0.051687	172.20.132.147	183.47.101.192	POP	77	C: USER [redacted]
9	0.067734	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=55 Ack=24 Win=64256 Len=0
10	0.071918	183.47.101.192	172.20.132.147	POP	59	S: +OK
11	0.072203	172.20.132.147	183.47.101.192	POP	77	C: PASS [redacted]
12	0.087279	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=60 Ack=47 Win=64256 Len=0
13	0.293485	183.47.101.192	172.20.132.147	POP	59	S: +OK
14	0.293919	172.20.132.147	183.47.101.192	POP	60	C: STAT
15	0.309448	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=65 Ack=53 Win=64256 Len=0
16	0.316066	183.47.101.192	172.20.132.147	POP	69	S: +OK 34 921144
17	0.316471	172.20.132.147	183.47.101.192	POP	60	C: LIST
18	0.331292	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=80 Ack=59 Win=64256 Len=0
19	0.356597	183.47.101.192	172.20.132.147	POP	382	S: +OK
20	0.356895	172.20.132.147	183.47.101.192	POP	62	C: RETR 1
21	0.412187	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=408 Ack=67 Win=64256 Len=0
22	0.413853	183.47.101.192	172.20.132.147	POP	1478	S: +OK 16372
23	0.413853	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
24	0.413853	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
25	0.413853	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
26	0.413853	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
27	0.413853	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
28	0.414320	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [ACK] Seq=67 Ack=8952 Win=64128 Len=0
29	0.414425	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
30	0.414425	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
31	0.414425	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
32	0.414425	183.47.101.192	172.20.132.147	POP	1342	S: DATA fragment, 1288 bytes
33	0.414642	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [ACK] Seq=67 Ack=14512 Win=64128 Len=0
34	0.428640	183.47.101.192	172.20.132.147	POP	1478	S: DATA fragment, 1424 bytes
35	0.428640	183.47.101.192	172.20.132.147	POP	914	S: DATA fragment, 860 bytes
36	0.429170	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [ACK] Seq=67 Ack=16796 Win=64128 Len=0
37	0.429249	172.20.132.147	183.47.101.192	POP	60	C: QUIT
38	0.443554	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=16796 Ack=73 Win=64256 Len=0
39	0.488301	183.47.101.192	172.20.132.147	POP	63	S: +OK Bye
40	0.488814	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [FIN, ACK] Seq=73 Ack=16805 Win=64128 Len=0
41	0.501983	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [FIN, ACK] Seq=16805 Ack=73 Win=64256 Len=0
42	0.502224	172.20.132.147	183.47.101.192	TCP	54	49736 → 110 [ACK] Seq=74 Ack=16806 Win=64128 Len=0
43	0.503059	183.47.101.192	172.20.132.147	TCP	54	110 → 49736 [ACK] Seq=16806 Ack=74 Win=64256 Len=0

三、 实验中遇到的问题及解决方法

（包括设计过程中的错误及测试过程中遇到的问题）

1. Emm，一开始觉得 socket 编程会很难，写了发现难度一般。
2. 这种 const char * 不能当作 char * 传入函数，所以要先强转成 void *指针。

```
const char *AUTH = "AUTH login\r\n";
    cus_send(s_fd, (void *)AUTH, strlen(AUTH), 0, "send AUTH");
    cus_recv(s_fd, (void *)buf, MAX_SIZE, 0, "recv AUTH");
```

3. 再就是上面也提到过的，第一遍写完才觉得按照框架给的例子写的太罗嗦了，就把共性的部分抽出来当作函数了。

四、 实验收获和建议

（关于本学期计算机网络实验的三种类型：配置验证实验、协议栈系列实验、Socket 编程实验，请给出您对于这三种类型实验的收获与体会，给出评论以及改进的建议。）

配置验证实验：感觉比较一般。听说前几届不是只在思科这个软件上做的，还有真家伙。

协议栈系列实验：帮助熟悉了协议栈的知识。个人觉得挺不错，只不过有些细节不太行，典型的就是 UDP 伪首部的部分。希望能继续改进实验框架。

Socket 编程实验：指导书很详细和简要的写出了实验需要的知识，这点不错。