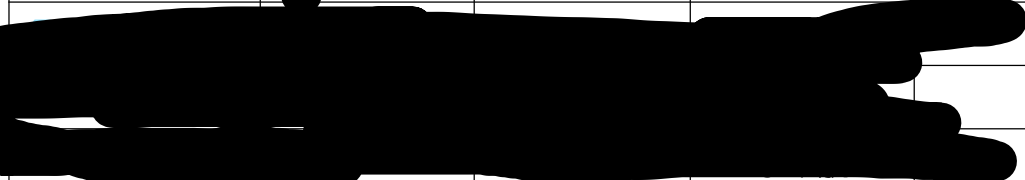


北京邮电大学实验报告

实验名称	计算机网络实验二	学院	网络空间安全	指导教师	程莉
班级	班内序号	学号	学生姓名	成绩	
					
实验内容	<p>1) 使用 Socket API 编写一个 HTTP 代理服务器程序, 使用 HTTP 接收来自邮件客户端的邮件并转发给实际的接收邮件服务器的用户邮箱, 并对来自其他服务 HTTP 请求进行转发到正确的服务器, 并返回给对应的客户端。</p> <p>2) 完成 HTTP 服务器, 接收邮件客户端的 TCP 连接请求, 按照 SMTP 协议接收并响应 SMTP 命令和邮件数据, 保存邮件;</p> <p>3) 完成 SMTP 客户端, 建立到实际邮件服务器的 TCP 连接, 发送 SMTP 命令, 将保存的邮件发送给实际邮件服务器;</p> <p>4) 实现提供发件人和收件人 Email 地址格式检查、提供差错报告、支持一封邮件多个接收者、在命令行窗口显示通信的主要过程的基本功能。</p>				
学生实验报告 (附页)					
实验成绩评定	<p>评语:</p> <p>成绩:</p> <p style="text-align: right;">指导教师签名:</p> <p style="text-align: right;">年 月 日</p>				

1. 成员分工

黄达威：实现 HTTP 代理服务器的相关功能并完成相关部分的实验报告；

陈煜博：实现抓取的邮件信息通过 smtp 协议发送到真正的 smtp 服务器的过程与这部分实验报告的撰写；

李丰航：实现实验报告主要的撰写，以及协助完成部分代码的实现。

2. 实验内容和实验环境描述

2.1 实验内容

(1) 使用 Socket API 编写一个 HTTP 邮件代理服务器程序，使用 HTTP 接收来自邮件客户端的邮件并转发给实际的接收邮件服务器的用户邮箱，并对来自其他服务 HTTP 请求进行转发到正确的服务器，且返回给对应的客户端。

(2) 完成 HTTP 服务器，接收邮件客户端的 TCP 连接请求，按照 SMTP 协议接收并响应 SMTP 命令和邮件数据，保存邮件；

(3) 完成 SMTP 客户端，建立到实际邮件服务器的 TCP 连接，发送 SMTP 命令，将保存的邮件发送给实际邮件服务器；

(4) 实现提供发件人和收件人 Email 地址格式检查、提供差错报告、支持一封邮件多个接收者、在命令行窗口显示通信的主要过程的基本功能。

2.2 实验环境

Windows11 操作系统，PyCharm 以及 QQ-Webmail, 以及 Wireshark 抓包

3. 软件设计

3.1 数据结构

使用 python 中的类来存储数据，以及一些全局变量。有 share_value 作为

全局变量、Http_proxy、Client 这两个类。

(1) share_value 类里的数据成员有 m 这个字典, 作为保存邮件的关键信息, 其中有 http_request_head(保存 http 包的请求头), http_repond_head (http 响应消息头, http_ip(http 请求者的 ip 地址), http_port(http 请求的进程端口), User (发件人邮箱), Pass (发件人邮箱授权码), MAIL FROM (发件人邮箱), RCPT TO (收件人邮箱列表), subject (邮件主题), data (邮件内容), flag(互斥变量, 作为邮件信息的写入以及发送的标志, 告知服务器以及客户端, 进行对应操作), mutex (作为互斥锁, 在写入时不能读取邮件信息, 在读取邮件信息时, 不能写入, 灵感来自于 os 的进程间通信的机制)

(2) Http_proxy 类中, 需要参数, host, port, listenbufsize, delay 作为类的启动, 以及

(3) Http_request_packet 类中有 req_line (请求头), method (请求方法), req_uri (请求网页), version, req_data (请求数据), host (请求的服务器地址, 端口), headers (请求头域)

(4) Client 类里的数据成员:

(5) 主函数 main 中有变量 host (监听 ip 地址), port (监听端口), listen (监听 socket 数量) bufsize (数据传输缓冲区大小), delay (数据转发延迟)。还有 proxy_thread 作为主线程启动, 以及 Client 以及 Http_proxy 类。

3.2 模块结构

3.2.1 Share_value

作为共享变量, 供 HTTP 代理服务器进行邮件信息的写入, 以及 smtp 客户端的读取邮件信息, 以及存在 flag 作为 HTTP 代理服务器写入邮件和 smtp 发送邮件的标志。以及 mute 作为互斥锁, 对邮件信息的保护, 只能单工操作。要么写, 要么读。

3.2.2 HttpRequestPacket()

(1) http_parse:对 http 的请求行、请求头、请求网页、请求方法, 请求数据进行分隔, 提取以及处理。

3.2.3 Http_proxy

Http_proxyn 类负责对 http 请求的处理, 以及对邮件发送请求的提取, 特殊处理, 还有对其他 http 请求的转发和回复对应的客户端。

- (1) `__init__ (self, host='127.0.0.1', port=8800, listen=5, bufsize=32, delay=1)`: 初始化函数, 实现对 host (监听 ip 地址), port (监听端口), listen (监听 socket 数量) bufsize (数据传输缓冲区大小), delay (数据转发延迟) 的赋值。
- (2) `self.socket_proxy` 建立 socket, 建立 `self.socket_proxy`, 将 `SO_REUSEADDR` 标记为 True, 实现当 socket 关闭后, 立刻回收该 socket 的端口。
- (3) `self.socket_proxy.bind()` 绑定 ip 和端口号。
- (4) `self.socket_proxy.listen()` 与 `self.socket_proxy.delay()` 设置监听 http 请求个数以及数据转发延迟。
- (5) `__del__(self)`: 关闭 socket。
- (6) `client_socket_accept(self)`: 获取已经与代理端建立连接的客户端套接字, 如无则阻塞, 直到可以获取一个建立连接套接字 并返回 `socket_client` 代理端与客户端之间建立的套接字。
- (7) `handle_client_request(self, socket_client)`: 连接客户端 socket, 并进入代理函数。
- (8) `http_proxy(self, socket_client)`:
接收来自客户端请求数据, 并对数据进行判断是否是邮件发送的请求, 若是邮件发送的请求则进行拦截处理, 转到 client。若不是则进行代理转发到想对应的服务端, 以及返回给客户端对应的响应。
(`socket_client.recv()` 对其中的数据判定有无发送邮件请求, `find('POST http://mail.qq.com/cgi-bin/compose_send')` != -1, 找到则说明有请求, 无则没有对应邮件发送请求)

- (9) `text_deal(temp, str(mail_send_ip), str(send_mail_port))`: 对应请求的处理, 提取出发件人, 收件人, 主题, 内容, 以及对共享变量 `m` 实现对应的赋值。

```
def text_deal(data, mail_send_ip, mail_send_port):
    print("接收到了webmail的发送邮件请求, 正在处理.....")
    pattern_head = r"(.*?)Cookie:" #针对http请求头的提取
    match0 = re.search(pattern_head, data)
    share_value.m["http_request_head"] = match0.group(1).replace("b'", "")
    send_mail_pat = 'sendmailname=([^\&]+)' #发件人的字符串匹配/正则
    accpet_name_pat = r'[a-z0-9\.\~+]+@[a-z0-9\.\~+]+\.[a-z]+' #收件人的字符串匹配/正则
    match = re.findall(send_mail_pat, data)
    match1 = re.findall(accpet_name_pat, data)
    #mail_sendname = match[0] 假如使用授权码登录的话, 就要开启
    mail_acceptname = match1[0:-1] #提取收件人列表
    pattern_subject = r"&subject=([^\&]*)" #主题的字符串匹配/正则
    subject = re.search(pattern_subject, data).group(1)
    subject = codecs.escape_decode(subject.encode('latin1'))[0].decode('utf-8') #输入是中文时, 实现对应的转码, 转成中文
    pattern_content = r"content_html=([^\&]+)" #内容的字符串匹配/正则
    content = re.search(pattern_content, data).group(1).replace('<div>', '').replace('</div>', '') #对<div>处理
    content = codecs.escape_decode(content.encode('latin1'))[0].decode('utf-8').replace("%26nbsp;", " ").replace("<br>", "") #空格处理, 有时会有<br>
    share_value.mutex.acquire() #开启锁 互斥锁, 保证共享变量写入, 或者读取是单一的
    share_value.m["http_ip"] = mail_send_ip #http邮件发送请求的ip地址
    share_value.m["http_port"] = mail_send_port #http邮件发送请求的端口
    share_value.m["RCPT TO"] = mail_acceptname #邮件的收件人列表
    share_value.m["subject"] = subject #邮件主题
    share_value.m["data"] = content #邮件内容
    #print(share_value.m)
    share_value.mutex.release() #关闭锁
```

- (10) 设置几个正则表达式, 匹配对收件人, 发件人, 主题, 内容, 并将传入请求 `http` 的 `ip` 与端口给共享变量赋值, 先关锁, 进行赋值操作, 赋值完再开锁。保证写入过程中不被读取。实现了对邮件信息的处理。
(这里要注意对一些, 内容的单独处理如空格, 中文, 网页标签)(由于 `qq` 邮件的发送比较复杂, 并且错误的邮件格式, `qq` 自己检查, 故直接用邮箱格式正则匹配, 匹配出, 就可以做到邮箱格式检查)
- (11) `save_info()`: 用来保存邮件文本信息

```
def save_info():
    f = open('record_mail.txt', 'a+', encoding="utf-8")
    f.write("邮箱发送HTTP请求: " + "ip地址来自:" + share_value.m["http_ip"] + " 端口:" + share_value.m["http_port"] + '\n')
    # 遍历字典的元素, 将每项元素的key和value分拆组成字符串, 注意添加分隔符和换行符
    f.write("发件人: " + share_value.m["MAIL FROM"] + '\n')
    f.write("收件人:")
    for accept_name in share_value.m["RCPT TO"]:
        f.write(" " + accept_name)
    f.write('\n' + "主题: " + share_value.m["subject"])
    f.write('\n' + "内容: " + share_value.m["data"] + '\n')
    share_value.m["flag"] = '1'
    #print("保存数据完成, flag=", share_value.m["flag"])
    # 注意关闭文件
    f.close()
```

- (12) 注意对中文写入的操作, 以及写完邮件信息后, 将 `share_value_m["flag"]` 修改为 `1`, 通知 `client` 可以进行发送了。
- (13) `__connect(self, host, port)`: 解析 `DNS` 得到套接字地址并与之建立连接, 并获得目的服务器主机的 `socket`。用于对正常 `http` 请求的处理。

- (14) `__nonblocking(self, socket_client, socket_server)`: 使用 `select` 实现异步处理数据,

```
#接收到邮件发送的http请求 进行代理, 转发, 保存
if temp.find('POST http://mail.qq.com/cgi-bin/compose_send') != -1:
    mail_send_ip, mail_send_port = socket_client.getpeername()
    #print("我是持久连接")
    # print("是邮件post")
    text_deal(temp, mail_send_ip, mail_send_port)
    #print("处理完持久连接数据")
    save_info()
else:
```

- (15) 注意这里时一些持久连接的处理, 有时候邮件发送是在持久连接处进行处理, 我们也要处理这个邮件发送请求。

3.2.4 Client

```
def send(self, s, r_fn):
    current_time = datetime.datetime.now()
    f = open(fn, "a", encoding="utf-8")
    self.msg = bytes(r_fn.encode()) # 对发送内容进行编码
    print(self.msg.decode().replace("\r\n", "")) # 显示解码及去掉\r\n后的发送内容
    f.write "[" + str(current_time) + "]" + self.msg.decode()
    s.send(self.msg) # 发送
    s.settimeout(10)
    self.info = s.recv(1024).decode() # 将接收到的内容解码
    print(self.info) # 显示接收到的内容
    current_time = datetime.datetime.now()
    f.write "[" + str(current_time) + "]" + self.info
    f.close()
```

`send()` 函数: 将请求传送出去并接受返回的消息显示出来并记录到日志中。之后还有两个与之功能相似的 `send` 函数, `send1` 函数实现了在输出账号密码时候将账号密码经 `base64` 编码后再输出的过程, `send2` 函数实现了只发送但不接受返回值的作用, 这样是为了在写信件主要内容时不会陷入因等不到返回而超时。

`check()` 函数, 为了判断返回的消息是否代表着通信正常

```
def check(self):
    if self.info[0:3] not in self.code: # 出现错误返回0, 通信正常返回1
        return 0
    else:
        return 1
```

```

def report(self):
    mail_host = "smtp.qq.com"
    mail_user = "3300469404@qq.com" # 发送者账号
    mail_pwd = "" # 发送者密码
    text = "An error occurred while sending the message " + self.msg.decode()
    # reg = re.compile("(?<=>[^>]*")
    MAILFrom = share_value.m["MAIL FROM"]
    message = MIMEText(text, 'plain', 'utf-8')
    message['From'] = Header("3300469404@qq.com")
    message['To'] = Header(MAILFrom)
    subject = 'Error report'
    message['Subject'] = Header(subject, 'utf-8')
    # 发送差错报告
    try:
        smtp_obj = smtplib.SMTP()
        smtp_obj.connect(mail_host, 25) # 连接邮箱服务器
        smtp_obj.login(mail_user, mail_pwd)
        smtp_obj.sendmail(mail_user, MAILFrom, message.as_string()) # 发送报告
        print("Error report sent to", MAILFrom)
        self.info = "xxx"
        self.msg = b""
    except smtplib.SMTPException as e:
        print("Failed to send the error report:")
        print(str(e))
        self.info = "xxx"
        self.msg = b""

```

report() 函数：若 check() 函数检测到不正常则调用，用我的 qq 邮箱（3300469404@qq.com）向发送方发送一个差错报告，写有响应值和具体内容。

slender() 函数，在确认共享变量中的 flag 为 1 后，即可开始传输信息，调用之前的函数进行与 smtp.qq.com 服务器的交互，运用 http 抓包后传入共享变量 share_value.m 中的值来完成请求，每一个请求语句之后都调用 check 和 report 函数来检验是否有差错，如果有则停止运行，在发送完毕之后将 flag 置 0。

其中实现了差错报告发送时通过 ssl 协议更安全的发送到发件人处：

```

# smtp_obj = smtplib.SMTP()
# smtp_obj.connect(mail_host, 25) # 连接邮箱服务器
smtp_obj = smtplib.SMTP_SSL()#SSL加密：端口号是465，通信过程加密，邮件数据安全。
smtp_obj.connect(mail_host, 465)

```

3.2.5 main

```

def main():#主函数
    host = "10.122.224.254"#监听ip,本机: 10.122.224.254
    port = "8800"#端口号
    listen = 5#监听数
    bufsize = 32#数据包获取的大小
    delay = 1#
    http_proxy = Http_Proxy(host, int(port), listen, bufsize, delay)
    client = Client()

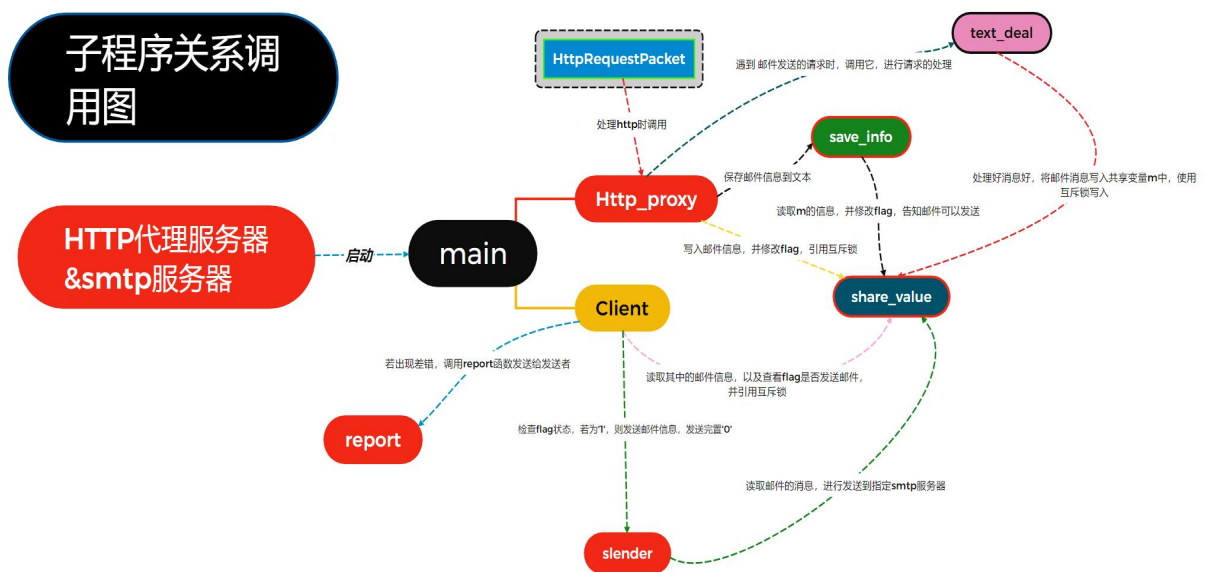
    # 使用线程启动代理服务器
    # 为了避免主线程退出,需要设置daemon=True
    proxy_thread = threading.Thread(target=http_proxy.start, daemon=True)
    proxy_thread.start()
    client.start()

    # 主线程阻塞
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("Interrupted by user")

```

设置一些初始化的量，以及类。设置主线程 http_proxy，开启线程。（这个也可以设置为双线程，但是我们想实现可以一直发邮件，想发多少就发多少。）在 client 在设置一个循环，一直处于即将发送邮件，（用共享变量 flag 实现），每当保存完一次邮件信息，就开始发送，发送完后，修改 flag，等待下一次邮件的信息传进来。其实是用 os 中的互斥锁，共享变量一些方式实现的通信过程。但是其实我们设想的是服务器一直处于运行的状态，几乎不会关闭。故没有对代理服务器的关闭进行设置。（可以通过设置类似于 flag 一样的 flag1，每一次发送完邮件，向输入中获取一个数 0/1，1 表示继续发送，或者停止，或者不再拦截邮件发送的 http 请求，当想关闭服务时，通过这个 flag1 关闭 client 的服务 break，从而跳出循环，使用 proxy_thread.join() 关闭主线程）

子程序关系调用图



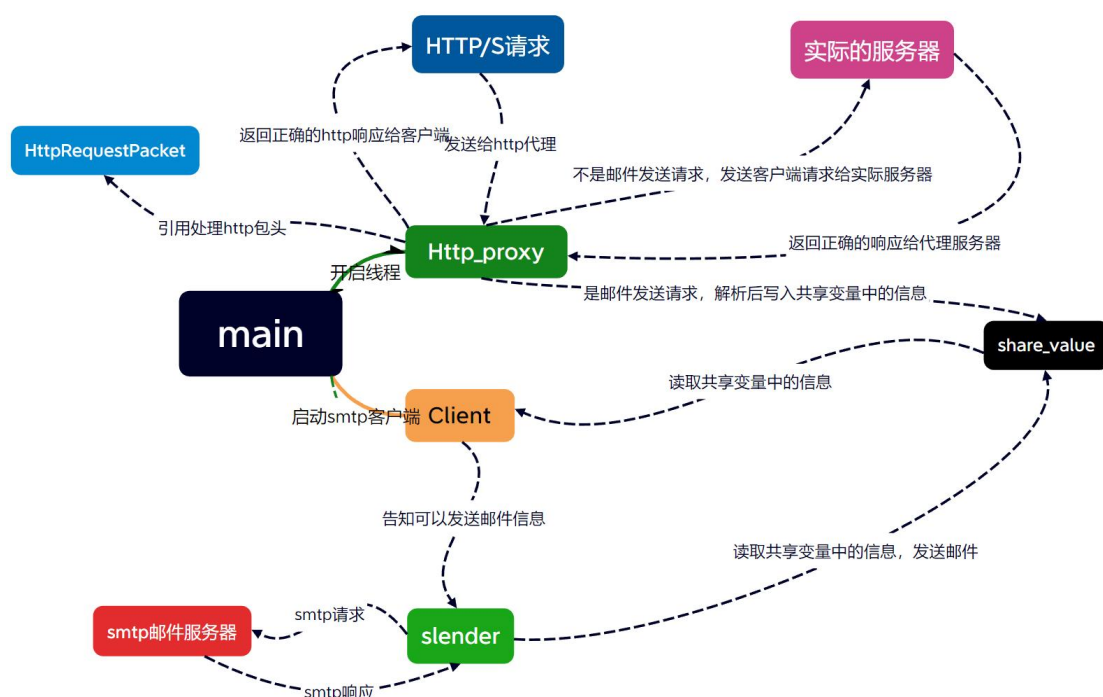
3.3 算法流程

首先运行 HTTP 代理服务器的。运行 main 函数，给 host、port 指定值分别为 127.0.0.1（或者本机 ip）、8800。创建 HTTP 服务器类 http_proxy 和 smtp 客户端类。使用 threading 创建函数式线程 proxy_thread 绑定 http_proxy，设置为主线程，为了避免主线程退出，需要设置 daemon=True。启动 proxy_thread。Start()。然后运行 client 类使它一直处于待发送邮件的状态。

HTTP 服务器创建 socket，绑定端口，ip，数据延迟，数据包大小。HTTP 接收来自 http/https 的请求，并获取客户端的 http 请求的 socket，ip 与端口。然后从客户端的 socket 获得 http 请求包。并对请求包进行判断，假如是正常的 http/s 请求则，解析它，并进行一定的处理，发送它到它对应需要的服务器中，并根据实际的服务器响应，将响应返回给 http 请求的客户。要是 http 请求是邮件发送的请求，则进行拦截，处理邮件消息输出“接收到了 webmail 的发送邮件请求，正在处理.....”，提取收件人，主题和内容并保存到共享变量中，然后保存邮件文本信息。修改 flag，告诉 clientsmpt 客户端，可以发送了，

Client 客户端 smtp 根据共享变量中的邮件信息，完成 smtp 客户端的功能进行发送，并生成日志 log。最后发送成功后输出“Sender released”，等待下一次的邮箱发送请求。一次一直循环。

算法流程图：



3.4 主要功能模块的实现要点

3.4.1 Http_proxy

(1) 作为 http 代理服务器，创建 socket，接收客户端的 socket，并冲 client_socket 中收到 http 请求的数据。并处理接收到的 http/https 请求,若是正常请求，则进行正常转发到实际服务器，并返回。

(2) 保存邮件数据：在 save_info() 子程序中实现了将邮件数据保存在本地的功能，将接收到的邮件数据后保存在目标文件中。

(3) 处理邮件发送 http 请求：text_deal(data, mail_send_ip, mail_send_port) 实现处理 http 请求，并分离发件人，收件人，主题，内容。（由于 qqwebmail 的发送邮件比较复杂，我们在这里只做了直接提取收件人的功能）

(4) 支持一封邮件多个接收者：我们的共享变量的收件人信息为列表的形式，可以存储多个收件人。

3.4.2 share_value

作为共享变量，供 HTTP 代理服务器进行邮件信息的写入，以及 smtp 客户端的读取邮件信息，以及存在 flag 作为 HTTP 代理服务器写入邮件和 smtp 发送邮

件的标志(提醒http服务器可以写入邮件信息,告知smtp客户端可以进行发送)。以及 mute 作为互斥锁,实现只能单工操作。要么写,要么读。实现了对邮件共享变量的保护,以及邮件信息的正确发送,并且可以有效地进行拥塞控制,流量控制,等等。

3.4.3 Client

(1) 作为 SMTP 客户端,与实际邮件服务器建立连接。对 server 得到的数据进行指令编辑并依次 send 编辑好的语句,分析 recv 到的返回值及其状态码,判断邮件发送的状态。核心功能为:消息发送及接收响应功能(send),邮件发送功能(start)。

(2) 提供差错报告:在邮件发送(slender)过程中,借助状态检测功能(check),判断通信过程中是否出现异常,若出现异常则将实际邮件服务器的差错报告状态码及其短语解释转发给邮件发送者。核心功能为差错报告功能(report)、通信状态检测功能(check)。

(3) 提供邮件收件人 Email 地址格式检查功能:在 isValid(email)中,检查邮箱地址格式是否正确。

(4) 提供可以一直发送 smtp 邮件的服务:我们为了实现可以一直处理邮件发送请求,在 client 中设置了一个循环来一直等待发送。

(5) 实现了差错报告发送时通过 ssl 协议更安全的发送到发件人处

3.4.4 main

(1) 采用主线程,作为 HTTP 代理服务器的线程(一直监听):在主函数 main 中使用 threading 创建线程实例,把(host, port, listen.BuFSIZE, delay)作为传入函数的参数,在__init__()函数中使用 socket 创建套接字,绑定(host, port),对 http 代理服务器的整个通信过程进行监听。

(2) 启动 Client 类,实现一直等待发送 smtp 邮件,等待邮件信息地写入以及通知。(我们为了实现可以一直处理邮件发送请求,若只发送一次,修改 client 中的循环即可)

4. 实验结果演示及分析

4.1 功能测试及结果

功能 1: 作为 Web 服务器，接收浏览器的 TCP 连接请求，接收 HTTP 命令和邮件数据，将邮件保存在文件中；

以下为发至代理服务器的邮件，将其中的关键信息保存进行字符串匹配并保存到了本地的文件中：



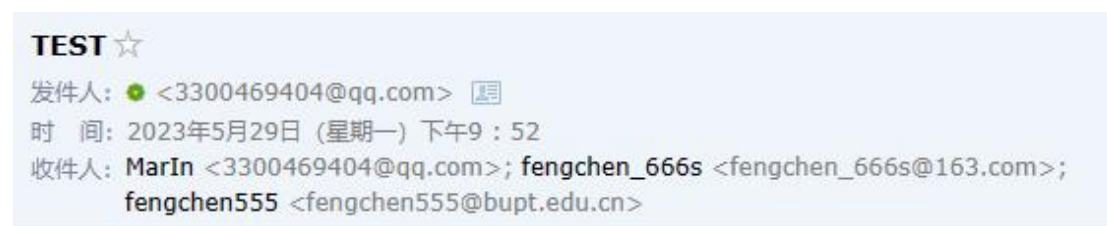
以下为保存在 txt 文件中的信息

```
邮箱发送HTTP请求: ip地址来自:127.0.0.1 端口:8596
发件人: 3300469404@qq.com
收件人: 3300469404@qq.com fengchen_666s@163.com fengchen555@bupt.edu.cn
主题: TEST
内容: 计算机网络实验二，陈煜博，黄达威
邮箱发送HTTP请求: ip地址来自:127.0.0.1 端口:8663
发件人: 3300469404@qq.com
收件人: 3300469404@qq.com fengchen_666s@163.com fengchen555@bupt.edu.cn
主题: TEST
内容: 计算机网络实验二，陈煜博，黄达威
```

功能 2: 作为 SMTP 客户端，建立到实际邮件服务器的 TCP 连接，发送 SMTP 命令，将保存的邮件发送给实际邮件服务器；

这里进行的是本地测试，所以客户端和服务端都是同一台电脑，在发送给真正的

服务器之后我们可以在收件人 qq 邮箱收到刚刚发送失败的邮件：



计算机网络实验二，陈煜博，黄达威



功能 3：提供发件人和收件人 Email 地址格式检查功能，例如下列邮件地址是错误的：chengli, [chengli@](mailto:chengli@bupt.edu.cn), bupt.edu.cn,

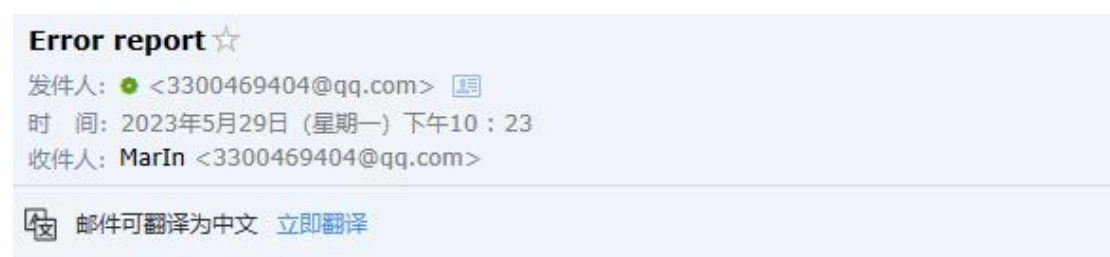
不存在不符合规划的邮箱，开始交互
计算机网络实验二，陈煜博，黄达威
开始进行一次smtp邮件发送

功能 4：提供邮件差错报告，要求处理两种差错：收件人不存在、邮件超过规定长度，将差错报告转发给发件人

这里我们把规定长度限制为了 1000，若超过 1000 则发送差错报告：



An error occurred while sending the message RCPT TO: <3300469404m>
, the server return 501 Bad address syntax. <http://service.mail.qq.com/cgi-bin/help?subtype=1&&id=20022&&no=1000730>



An error occurred while sending the message , the length data is out of range

功能 5: 支持一封邮件多个接收者, 要求接收者属于不同的域 (如
2 bupt.edu.cn、163.com、qq.com,...) ;



功能 6: 在屏幕上显示: 当前时间、浏览器端的 IP 地址及端口号、代理的 IP 地址及端口号、实际邮件服务器的域名/IP 地址及端口号、发件人邮件地址、收件人邮件地址、邮件长度 (字节数)


```

接收到了webmail的发送邮件请求，正在处理.....
{'http_request_head': 'POST http://mail.qq.com/cgi-bin/compose_send?sid=yFRNLgSuqhNn5yNH HTTP/1.1\r\nHost:
不存在不符合规划的邮箱，开始交互
计算机网络实验二，陈煜博，黄达威
开始进行一次smtp邮件发送
发送方的ip地址与端口号为：127.0.0.1:11664
代理的ip地址与端口号为：127.0.0.1:8800
实际邮件地址的域名与端口号为：smtp.qq.com:25
发件人邮件地址：3300469404@qq.com
收件人邮件地址为：3300469404@qq.com, fengchen_666s@163.com, fengchen555@bupt.edu.cn
邮件长度为：48
开始与邮件服务器交互：

```

功能 7：将与浏览器及邮件服务器的通信过程（HTTP 请求/响应消息头、SMTP 命令/响应、时间戳）记录在日志文件中，日志文件格式为纯文本文件，名字为 Log-时间戳.txt

[202305292227]POST http://mail.qq.com/cgi-bin/compose_send?sid=yFRNLgSuqhNn5yNH HTTP/1.1\r\nHost: mail.qq.com\r\nProxy-Connection: keep-alive\r\nContent-Length: 404497\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.1774.50\r\nContent-Type: application/x-www-form-urlencoded\r\nAccept: */*\r\nOrigin: http://mail.qq.com\r\nReferer: http://mail.qq.com/zh_CN/htmledition/ajax_proxy.html?mail.qq.com&v=140521\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9\r\n	[2023-05-29 22:27:58.616095]cHRzZ2F0dWN3bXNrY2lpYw==
[202305292227]POST http://mail.qq.com/cgi-bin/compose_send?sid=yFRNLgSuqhNn5yNH HTTP/1.1\r\nHost: mail.qq.com\r\nProxy-Connection: keep-alive\r\nContent-Length: 1556\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.1774.50\r\nContent-Type: application/x-www-form-urlencoded\r\nAccept: */*\r\nOrigin: http://mail.qq.com\r\nReferer: http://mail.qq.com/zh_CN/htmledition/ajax_proxy.html?mail.qq.com&v=140521\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9\r\n	[2023-05-29 22:27:58.840102]235 Authentication successful
[2023-05-29 22:27:58.470100]EHLO 127.0.0.1	[2023-05-29 22:27:58.840102]MAIL FROM: <3300469404@qq.com>
[2023-05-29 22:27:58.519096]250-newxmesmtplgicsvrszc2-0.qq.com	[2023-05-29 22:27:58.922103]250 OK
250-PIPELINING	[2023-05-29 22:27:58.922103]RCPT TO: <3300469404@qq.com>
250-SIZE 73400320	[2023-05-29 22:27:59.032102]250 OK
250-STARTTLS	[2023-05-29 22:27:59.032102]RCPT TO: <fengchen_666s@163.com>
250-AUTH LOGIN PLAIN XOAUTH XOAUTH2	[2023-05-29 22:27:59.125102]250 OK
250-AUTH=LOGIN	[2023-05-29 22:27:59.125102]RCPT TO: <fengchen555@bupt.edu.cn>
250-MAILCOMPRESS	[2023-05-29 22:27:59.288102]250 OK
250 8BITMIME	[2023-05-29 22:27:59.289102]DATA
[2023-05-29 22:27:58.519096]AUTH LOGIN	[2023-05-29 22:27:59.343103]354 End data with <CR> <LF>. <CR> <LF>.
[2023-05-29 22:27:58.567096]334 VXNlcm5hbWU6	[2023-05-29 22:27:59.343103]SUBJECT: TEST
[2023-05-29 22:27:58.567096]MzMwMDQOTQwNEBxcS5jb20=	[2023-05-29 22:27:59.344103]FROM: 3300469404@qq.com <3300469404@qq.com>
[2023-05-29 22:27:58.616095]334 UGFzc3dvcmQ6	[2023-05-29 22:27:59.344103]TO: 3300469404 <3300469404@qq.com>,fengchen_666s <fengchen_666s@163.com>,fengchen555 <fengchen555@bupt.edu.cn>
	[2023-05-29 22:27:59.345103]计算机网络实验二，陈煜博，黄达威
	[2023-05-29 22:27:59.345103]
	.
	[2023-05-29 22:27:59.736680]250 OK: queued as.
	[2023-05-29 22:27:59.736680]QUIT
	[2023-05-29 22:27:59.788684]221 Bye.

4.2 Wireshark 捕获信息

浏览器与代理程序之间的 HTTP 消息流:

12	0.298787	111.30.170.247	10.122.210.120	TCP	60	[TCP Retransmission] 80 → 12827 [FIN, ACK] Seq=1 Ack=2 Win=133 Len=0			
13	0.298829	10.122.210.120	111.30.170.247	TCP	54	[TCP ZeroWindow] 12827 → 80 [ACK] Seq=2 Ack=2 Win=0 Len=0			
14	0.738866	10.122.210.120	183.47.114.18	TCP	66	12851 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM			
15	0.782004	183.47.114.18	10.122.210.120	TCP	66	80 → 12851 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1386 SACK_PERM WS=128			
16	0.782112	10.122.210.120	183.47.114.18	TCP	54	12851 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0			
17	0.782169	10.122.210.120	183.47.114.18	TCP	1440	12851 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=1386 [TCP segment of a reassembled PDU]			
18	0.782169	10.122.210.120	183.47.114.18	TCP	1440	12851 → 80 [ACK] Seq=1387 Ack=1 Win=131584 Len=1386 [TCP segment of a reassembled PDU]			
19	0.782169	10.122.210.120	183.47.114.18	HTTP	623	POST /cgi-bin/compose_send?sid=yFRNLgSugHn5yNH HTTP/1.1 (application/x-www-form-urlencoded)			
20	0.869651	10.122.210.120	183.47.114.18	TCP	1440	[TCP Retransmission] 12851 → 80 [PSH, ACK] Seq=1956 Ack=1 Win=131584 Len=1386			
21	0.939502	183.47.114.18	10.122.210.120	TCP	60	80 → 12851 [ACK] Seq=1 Ack=1387 Win=64128 Len=0			
22	0.939502	183.47.114.18	10.122.210.120	TCP	60	80 → 12851 [ACK] Seq=1 Ack=2773 Win=64128 Len=0			
23	0.939502	183.47.114.18	10.122.210.120	TCP	60	80 → 12851 [ACK] Seq=1 Ack=3342 Win=64128 Len=0			
24	0.939502	183.47.114.18	10.122.210.120	HTTP/1.1	363	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)			
25	0.980382	183.47.114.18	10.122.210.120	TCP	66	[TCP Dump ACK 23811 80 → 12851 [ACK] Seq=310 Ack=3342 Win=64128 Len=0 SLF=1956 SRF=3342]			

> Frame 19: 623 bytes on wire (4984 bits), 623 bytes captured (4984 bits) on interface \Device\NPF{...}

> Ethernet II, Src: IntelCor_03:36:b7 (e0:d4:64:03:36:b7), Dst: ArubaHe_6c:24:00 (10:4f::)

> Internet Protocol Version 4, Src: 10.122.210.120, Dst: 183.47.114.18

> Transmission Control Protocol, Src Port: 12851, Dst Port: 80, Seq: 2773, Ack: 1, Len: 51

> [3 Reassembled TCP Segments (3341 bytes): #17(1386), #18(1386), #19(569)]

> Hypertext Transfer Protocol

> HTML Form URL Encoded: application/x-www-form-urlencoded

> Form item: "ebef99c05d3fec52cd4a28364d3598c" = "1de0cc09355329a73428cd8b3dc65d0"

> Form item: "sid" = "yFRNLgSugHn5yNH"

> Form item: "from_s" = "cnew"

> Form item: "to" = ""

> Form item: "subject" = "TEST"

> Form item: "content_html" = "计算机网络实验二，陈煜博，黄达威"

> Form item: "sendmailname" = "3300469404@qq.com"

> Form item: "savesendbox" = "1"

> Form item: "actiontype" = "send"

> Form item: "sendname" = "MarIn"

> Form item: "acctid" = "0"

> Form item: "separatedcopy" = "false"

> Form item: "s" = "comm"

> Form item: "hitaddbook" = "0"

> Form item: "selfdefinestation" = "-1"

> Form item: "domaincheck" = "0"

> Form item: "coita" = "1685367A70A62"

0000 10 4f 58 6c 24 00 e0 d4 64 03 36 b7 08 00 45 00

0010 02 61 52 4a 40 00 80 06 00 00 0a 7a d2 78 b7 2f

0020 72 12 32 33 00 50 44 30 a8 11 ef 8d 85 e1 50 18

0030 02 02 08 88 00 00 74 74 63 6e 74 3d 30 26 6c 6f

0040 67 61 74 74 63 6e 74 3d 30 26 6c 6f 67 61 74 74

0050 63 6e 74 3d 30 26 6c 6f 67 61 74 74 73 69 7a 65

0060 3d 30 26 6c 6f 67 61 74 74 73 69 7a 65 3d 30 26

0070 6c 6f 67 61 74 74 73 69 7a 65 3d 30 26 6c 6f 67

0080 61 74 74 73 69 7a 65 3d 30 26 6c 6f 67 61 74 74

0090 73 69 7a 65 3d 30 26 6c 6f 67 61 74 74 73 69 7a

00a0 65 3d 30 26 6c 6f 67 61 74 74 73 69 7a 65 3d 30

00b0 26 6c 6f 67 61 74 74 73 69 7a 65 3d 30 26 6c 6f

00c0 67 61 74 74 73 69 7a 65 3d 30 26 6c 6f 67 61 74

00d0 74 73 69 7a 65 3d 30 26 6c 6f 67 61 74 74 73 69

00e0 7a 65 3d 30 26 74 69 6d 65 7a 6f 6e 65 3d 32 38

00f0 38 30 30 26 74 69 6d 65 7a 6f 6e 65 3d 32 38 38

0100 30 30 26 74 69 6d 65 7a 6f 6e 65 3d 32 38 38 30

0110 30 26 74 69 6d 65 7a 6f 6e 65 3d 32 38 38 30 30

0120 26 74 69 6d 65 7a 6f 6e 65 3d 32 38 38 30 30 26

0130 74 69 6d 65 7a 6f 6e 65 3d 32 38 38 30 30 26 74

0140 69 6d 65 7a 6f 6e 65 3d 32 38 38 30 30 26 74 69

0150 6d 65 7a 6f 6e 65 3d 32 38 38 30 30 26 74 69 6d

0160 65 7a 6f 6e 65 3d 32 38 38 30 30 26 74 69 6d 65

0170 7a 6f 6e 65 3d 32 38 38 30 30 26 74 69 6d 65 7a

0180 6f 6e 65 3d 32 38 38 30 30 26 74 69 6d 65 7a 6f

0190 6e 65 5f 64 73 74 3d 30 26 74 69 6d 65 7a 6f 6e

POST /cgi-bin/compose_send?sid=yFRNLgSuqhNn5yNH HTTP/1.1
Host: mail.qq.com
Proxy-Connection: keep-alive
Content-Length: 874
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.1774.50
Content-Type: application/x-www-form-urlencoded
Accept: /*/*
Origin:
Referer: /zh_CN/html/edition/ajax_proxy.html?mail.qq.com&v=140521
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: iip=0; pgv_pvid=5383036250; RK=qxG1CmqgXI; ptcz=ab04d35e5537853e036e09678de197dc6e417b2943a0318c0a6f33dbfe3cc7d5; webp=1; tvfe_boss_uuid=ea49895b1058e18e; o_cookie=3300469404; pac_uid=1_3300469404; qm_device_id=0wR68HbXfWQ0weJWk+gy7dkuKF5w8pNC6PW4UGyfvH+uvm6potZN+UvIygyXRns5; edition=mail.qq.com; qm_login_type=qq; qm_username=3300469404; uin=o3300469404; qm_domain=https://mail.qq.com; ssl_edition=sail.qq.com; username=-994497892&3300469404; _qpsvr_localtk=0.0464670598011272; qm_authimgs_id=0; qm_verifyimagesession=h013234f6646db4b779a5ae8d53876eaf18a10c369858a476c0f3812f47f3702163610c7ce3f97713fa; CcSHOW=000000; sid=-994497892&1de0cc09355329a73428cd8b3d1c65d0; qm_muti_sid=13102664325343900&yFRNLgSuqhNn5yNH; xm_uin=13102664325343900; xm_sid=zZxqSIwqN0cuuVRrAMRBWAAA; xm_muti_sid=13102664325343900&zZxqSIwqN0cuuVRrAMRBWAAA; xm_skey=13102664325343900&0ba784641ef4035937dd12a7da6382b0; xm_ws=13102664325343900&d138c4e03a34fa9dadb82dbe2ce32bf8; skey=@MZVYf6h99; p_uin=o3300469404; pt4_token=0iInv34lpwH9oMOuEQE9D5jsmFxAUhuJ4ayQ2JoI3sw; new_mail_num=-994497892&834

ebef99c05d53fec52cd4a28364d3598c=1de0cc09355329a73428cd8b3d1c65d0&sid=yFRNLgSuqhNn5yNH&from_s=cnew&to=3300469404@qq.com&subject=TEST&content__html=<div>.....</div>&sendmailname=3300469404@qq.com&savesendbox=1&actiontype=send&acctid=0&separatedcopy=false&s=comm&hitaddrbook=0&selfdefinestation=-1&domaincheck=0&cgitm=1685373038268&cgitm=1685373038268&clitm=1685373039939&clitm=1685373039939&comtm=1685373070497&comtm=1685373114380&comtm=1685373148900&logattcnt=0&logattcnt=0&logattcnt=0&logattsize=0&logattsize=0&logattsize=0&timezon e=28800&timezone=28800&timezone=28800&timezone_dst=0&timezone_dst=0&timezone_dst=0&cginame=compose_send&ef=js&t=c ompose_send.json&resp_charset=UTF8HTTP/1.1 200 OK
Date: Mon, 29 May 2023 15:12:28 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 97
Connection: keep-alive
Server: nginx
Cache-control: no-cache
Referrer-Policy: origin

{title : "cgi exception", appname : "compose_send", errcode: "-4", errmsg: "GET....."}

分組 946. 1 客戶端 分組, 1 服務器 分組, 1 turn(s). 点击查看.

整个对话 (2737 bytes) Show data as ASCII

查找: 查找下一个(N)

滤掉此流 打印 另存为... 返回 Close Help

代理程序与实际邮件服务器之间的 SMTP 消息流：

No.	Time	Source	Destination	Protocol	Length	Info
497	3.316973	109.244.198.105	10.122.210.120	SMTP	119 S:	220 newmesmtplgicvrszc5-0.qq.com XMail Esmtp QQ Mail Server.
498	3.317531	10.122.210.120	109.244.198.105	SMTP	70 C:	EHL0 127.0.0.1
500	3.356463	109.244.198.105	10.122.210.120	SMTP	225 S:	250-newmesmtplgicvrszc5-0.qq.com PIPELINING SIZE 73408320 STARTTLS AUTH LOGIN PL
501	3.357030	10.122.210.120	109.244.198.105	SMTP	66 C:	AUTH LOGIN
502	3.398465	109.244.198.105	10.122.210.120	SMTP	72 S:	334 VXNlcm5hbWU6
503	3.399199	10.122.210.120	109.244.198.105	SMTP	80 C:	User: MzMwMDQ0TQwNEBxc55jb20=
506	3.443267	109.244.198.105	10.122.210.120	SMTP	72 S:	334 UGFzc3dvcmQ6
507	3.443834	10.122.210.120	109.244.198.105	SMTP	80 C:	Pass: cHRzZ2F0dWlnb3bXlnY2lpYw==
523	3.643396	109.244.198.105	10.122.210.120	SMTP	85 S:	235 Authentication successful
524	3.644000	10.122.210.120	109.244.198.105	SMTP	86 C:	MAIL FROM: <3300469404@qq.com>
611	3.723967	109.244.198.105	10.122.210.120	SMTP	62 S:	250 OK
615	3.725076	10.122.210.120	109.244.198.105	SMTP	84 C:	RCPT TO: <3300469404@qq.com>
633	3.829850	109.244.198.105	10.122.210.120	SMTP	62 S:	250 OK
634	3.830408	10.122.210.120	109.244.198.105	SMTP	88 C:	RCPT TO: <fengchen_666s@163.com>
637	3.908783	109.244.198.105	10.122.210.120	SMTP	62 S:	250 OK
638	3.909415	10.122.210.120	109.244.198.105	SMTP	90 C:	RCPT TO: <fengchen555@bupt.edu.cn>
645	4.039975	109.244.198.105	10.122.210.120	SMTP	62 S:	250 OK
646	4.040615	10.122.210.120	109.244.198.105	SMTP	60 C:	DATA
650	4.082133	109.244.198.105	10.122.210.120	SMTP	92 S:	354 End data with <CR><LF>.<CR><LF>.
651	4.082702	10.122.210.120	109.244.198.105	SMTP	69 C:	DATA fragment, 15 bytes
656	4.162993	10.122.210.120	109.244.198.105	SMTP/L	266	subject: TEST, from: 3300469404@qq.com <3300469404@qq.com>, , SUBJECT: TEST , FROM: 330046940-
662	4.469383	109.244.198.105	10.122.210.120	SMTP	74 S:	250 OK: queued as.
663	4.470006	10.122.210.120	109.244.198.105	SMTP	60 C:	QUIT
670	4.512246	109.244.198.105	10.122.210.120	SMTP	64 S:	221 Bye.

> Frame 634: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface \Device
> Ethernet II, Src: IntelCor.03:36:b7 (e0:d4:64:03:36:b7), Dst: ArubaHe.6c:24:00 (10:4f:58:
> Internet Protocol Version 4, Src: 10.122.210.120, Dst: 109.244.198.105
> Transmission Control Protocol, Src Port: 13499, Dst Port: 25, Seq: 143, Ack: 320, Len: 34
> Simple Mail Transfer Protocol

0000 10 4f 58 6c 24 00 e0 d4 64 03 36 b7 08 00 45 00 .OX1\$... d 6...E:
0010 00 4a 76 85 00 00 80 06 00 00 0a 7a d2 78 6d f4 .Jv @... ..z xm:
0020 c6 69 34 bb 00 19 fe e0 63 a9 94 fe 19 32 50 18 .14..... C...zP:
0030 02 01 11 8d 00 00 52 43 50 54 20 54 4f 3a 20 3cRC PT TO: <
0040 60 03 0e 07 63 68 05 6a 5f 36 36 36 73 40 31 36 fengchen_666s@16
0050 b3 2e 63 6f 6d 3e 0d 0a 3,com>=

```
220 newxmesmtplgicsvrszb6-0.qq.com XMail Esmtp QQ Mail Server.
EHLO 127.0.0.1
250-newxmesmtplgicsvrszb6-0.qq.com
250-PIPELINING
250-SIZE 73400320
250-STARTTLS
250-AUTH LOGIN PLAIN XOAUTH XOAUTH2
250-AUTH=LOGIN
250-MAILCOMPRESS
250 8BITMIME
AUTH LOGIN
334 VXNlcm5hbWU6
MzMwMDQ2OTQwNEBxcS5jb20=
334 UGFzc3dvcmQ6
cHRzZ2F0dWN3bXNrY2lpYW==
235 Authentication successful
MAIL FROM: <3300469404@qq.com>
250 OK
RCPT TO: <3300469404@qq.com>
250 OK
DATA
354 End data with <CR><LF>.<CR><LF>.
SUBJECT: TEST
FROM: 3300469404@qq.com <3300469404@qq.com>
TO: 3300469404 <3300469404@qq.com>
<span style=%22font-family: %26quot;lucida Grande%26quot;; Verdana, %26quot;Microsoft YaHei%26quot;;
%22>.....</span>
.
250 OK: queued as.
QUIT
221 Bye.
```

分框 2231. 10 客户端 分框, 10 服务器 分框, 18 turn(s). 点击选择.

整个对话 (803 bytes) Show data as ASCII 流 38

查找: 查找下一个(N)

滤掉此流 打印 另存为... 返回 Close Help

4.3 与 SMTP 协议的比较以及程序的优点和不足

与 SMTP 协议相同。

优点：能将客户端发送的邮件信息保存在 SMTP 邮件代理服务器本地；采取多线程技术。

不足：不能连续发送多封邮件记录邮件队列。

5. 实验总结和心得体会

5.1 实际上机调试时间

一周。

5.2 编程工具方面遇到的问题

暂无。

5.3 编程语言方面遇到的问题

在处理 http/https 协议上遇到了比较大的困难，如何提取信息，如何处理其他的 http 请求。有关 Socket、Threading 方面的函数的使用熟练度需要提高。

5.4 协议方面遇到的问题

对于 http/https 的协议还不够熟悉，真实的 http/https 跟书本上的知识还是有些许区别，需要多抓包熟悉，了解。

5.5 HTTP 和 SMTP 协议的不足及改进思路

HTTP 协议的不足：

明文传输：HTTP 协议在传输过程中使用明文，容易被攻击者窃听和篡改。缺乏加密机制，导致数据的机密性和完整性难以保证。改进思路：引入加密机制，例如使用 TLS/SSL 来确保通信的安全性。

无状态性：HTTP 协议是无状态的，服务器不会保留客户端的状态信息，每个请求都是独立的，无法追踪会话状态。改进思路：引入会话管理机制，例如使用 Cookie 或者 Token 来跟踪用户状态，提供更好的用户体验。

性能问题：HTTP 协议在每次请求时需要建立和断开连接，导致频繁的连接开销，尤其在处理大量并发请求时性能较差。改进思路：引入持久连接，例如 HTTP 1.1 中的 keep-alive 机制，允许多个请求和响应复用同一个 TCP 连接，减少连接的建立和断开次数。

SMTP 协议的不足

SMTP 协议功能较为基础，缺少安全性，SMTP 协议信息本质上均是明文传输（经 base64 编码后和明文传输并无二致），容易受到第三方的攻击，被第三方截取邮件内容，或是篡改收件人，发件人的地址以及邮件内容，十分容易受到中间人攻击。

反垃圾邮件机制不完善：对于垃圾邮件的防范机制相对薄弱，容易受到垃圾邮件发送者的滥用。

邮件传输延迟：SMTP 协议使用一种存储转发的方式进行邮件传输，需要多次中转才能到达目标服务器，可能引起传输延迟

改进思路：使用的反垃圾邮件机制，例如使用 DNSBL（域名黑名单）来过滤和拦截垃圾邮件。使用直接传输机制，例如使用 SMTP 直接发送邮件，减少中转次数，提高传输效率。引入扩展改进等已有 SSL，X2.5 等增强版本存在。

5.6 总结

通过本次使用 Socket API 编写一个 HTTP 代理服务器程序的实验，我们初步了解了 socket 及其用法，进一步学习了 python 语言，并对 python 语言中线程，变量，socket 更加熟悉。对 HTTP 协议也有了更深的认识，对 smtp 命令和返回状态码的理解更加透彻；对邮件发送的整个通信过程有了更深的了解，对 weimail、HTTP 代理服务器、SMTP 邮件服务器在通信过程中所起的作用有了更清晰的理解，真正切实地感受到了网络中地各种协议，其实网络中地协议还有很多，还有很多协议值得我们深入学习，在这其中收获了不少！

5.7 设计和安排的不足

对于实验的一些要求和任务，我感觉有一些不太明确，导致我们开始时比较迷茫，在想是直接做个前后端的实现邮件发送，还是写个代理再发送。具体的任务要求不够详细，两个实验指导 pdf 部分内容不一致，可能一个有这个要求，一个又有其他要求，看多了有点晕。就是主要在任务和要求上不是很明确需要干什么，要是能具体写清楚就更好了。

其他的来说还是很不错的，是一个有点困难但是有趣的过程，接触到了很多的协议内容，收获了很多。是个十分不错的实验！

源代码附录

main.py

```
from Http_proxy import Http_Proxy
from Client import Client
import threading
import time

def main():#主函数
    host = "localhost"#监听 ip,本机: 10.122.224.254
    port = "8800"#端口号
    listen = 5#监听数
    bufsize = 32#数据包获取的大小
    delay = 1#
    http_proxy = Http_Proxy(host, int(port), listen, bufsize, delay)
    client = Client()

    # 使用线程启动代理服务器
    # 为了避免主线程退出, 需要设置 daemon=True
    proxy_thread = threading.Thread(target=http_proxy.start,
    daemon=True)
    proxy_thread.start()
    try:
        client.start()
    except:
        print("SMTP 停止服务")
        # proxy_thread.join(timeout=10)
        # print("HTTP 代理服务结束")

    #proxy_thread.join()

    # 主线程阻塞
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("Interrupted by user")
```

```

if __name__ == "__main__":

    main()
    print('Service terminated')

```

Http_proxy.py

```

from __future__ import print_function
import socket
import select
import time
import re
import codecs
import share_value
import _thread as thread
from Http_request_packet import HttpRequestPacket
#对传入的http post 请求进行处理, 提取出对应的发件人, 收件人, 主题, 内容
def text_deal(data,mail_send_ip,mail_send_port):
    print("接收到了 webmail 的发送邮件请求, 正在处理.....")
    #print(data)
    pattern_head = r"(.*?)Cookie:"#针对 http 请求头的提取
    match0 = re.search(pattern_head , data)
    share_value.m["http_request_head"] =
match0.group(1).replace("\b","")
    send_mail_pat = 'sendmailname=([^&]+)'#发件人的字符串匹配/正则
    accpet_name_pat = r'[a-z0-9\.\-+_]@[a-z0-9\.\-+_]\\. [a-z]+'#收件人的字符串匹配/正则
    match = re.findall(send_mail_pat, data)
    match1 = re.findall(accpet_name_pat, data)
    #mail_sendname = match[0] 假如使用授权码登录的话, 就要开启
    mail_acceptname = match1[0:-1]#提取收件人列表
    pattern_subject = r"&subject=([^&]*)"#主题的字符串匹配/正则
    subject = re.search(pattern_subject, data).group(1)
    subject =
codecs.escape_decode(subject.encode('latin1'))[0].decode('utf-8')
#输入是中文时, 实现对应的转码, 转成中文
    pattern_content = r"content__html=([^&]+)"#内容的字符串匹配/正则
    content= re.search(pattern_content,
data).group(1).replace('<div>',' ').replace('</div>',' ')#对<div>处理
    content =
codecs.escape_decode(content.encode('latin1'))[0].decode('utf-8').
replace("%26nbsp;"," ").replace("<br>","")#空格处理, 有时会有<br>

```



```

    share_value.mutex.acquire() #开启锁 互斥锁, 保证共享变量写入, 或者读取
    是单一的
    share_value.m["http_ip"] = mail_send_ip #http 邮件发送请求的 ip 地址
    share_value.m["http_port"] = mail_send_port #http 邮件发送请求的端口
    share_value.m["RCPT TO"] = mail_acceptname #邮件的收件人列表
    share_value.m["subject"] = subject #邮件主题
    share_value.m["data"] = content #邮件内容
    #print (share_value.m)
    share_value.mutex.release() #关闭锁
#保存邮件文本
def save_info():
    f = open('record_mail.txt', 'a+', encoding="utf-8")
    share_value.mutex.acquire() # 开启锁 互斥锁, 保证共享变量写入, 或者读
    取是单一的
    f.write("邮箱发送HTTP请求: " + share_value.m["http_ip"]
    + " 端口:" + share_value.m["http_port"] + '\n')
    # 遍历字典的元素, 将每项元素的 key 和 value 分拆组成字符串, 注意添加分隔符和
    换行符
    f.write("发件人: " + share_value.m["MAIL FROM"] + '\n')
    f.write("收件人:")
    for accept_name in share_value.m["RCPT TO"]:
        f.write(" " + accept_name )
    f.write('\n' + "主题: " + share_value.m["subject"])
    f.write('\n' + "内容: " + share_value.m["data"] + '\n')
    share_value.m["flag"] = '1'
    share_value.mutex.release() # 关闭锁
    #print("保存数据完成, flag=", share_value.m["flag"])
    # 注意关闭文件
    f.close()

#核心 http 代理程序
class Http_Proxy(object):

    # 实现简单的 HTTP 服务器代理
    # 对于正常的 http 请求:
    # 客户端(client) <=> 代理端(proxy) <=> 服务端(server)
    # 对于发送的 http 请求:
    # 进行适当处理
    # 客户端(client) <=> 代理端(proxy) -> 通知 client (客户端) <=> 真正的服务
    端(server)
    # 代理端将 flag 设为 1, 告诉 client 写入了一封待发送的邮件, 要进行发送

```

```

def __init__(self, host='127.0.0.1', port=8800, listen=5,
bufsize=32, delay=1):#一些默认参数
    # 初始化代理套接字, 用于与客户端、服务端通信
    # 参数: host 监听地址, 默认 127.0.0.1 代表本机 ip 地址
    # 参数: port 监听端口, 默认 8800
    # 参数: listen 监听客户端数量, 默认 5
    # 参数: bufsize 数据传输缓冲区大小, 单位 kb, 默认 32kb
    # 参数: delay 数据转发延迟, 单位 ms, 默认 1ms
    #建立 socket 并设置当 socket 关闭后, 立刻回收该 socket 的端口
    self.socket_proxy = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    self.socket_proxy.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR,1)
    # 将 SO_REUSEADDR 标记为 True, 当 socket 关闭后, 立刻回收该 socket 的
    端口
    #绑定监听的 ip 与端口号
    self.socket_proxy.bind((host, port))
    #设置监听客户端个数
    self.socket_proxy.listen(listen)
    self.socket_recv_bufsize = bufsize * 1024
    self.delay = delay / 1000.0
    #socket 启动 并开始监听
    print('info: bind={}:{}'.format(host, port))
    print('info: listen={}'.format(listen))
    print('info: bufsize={}kb, delay={}ms'.format(bufsize,
delay))
    #输出开启监听信息
def __del__(self):#socket 关闭
    self.socket_proxy.close()

def __connect(self, host, port):
    #
    # 解析 DNS 得到套接字地址并与之建立连接
    #
    # 参数: host 主机
    # 参数: port 端口
    # 返回: 与目标主机建立连接的套接字
    #
    # 解析 DNS 获取对应协议簇、socket 类型、目标地址
    # getaddrinfo -> [(family, sockettype, proto, canonname,
target_addr),]

```



```

        (family, sockettype, _, _, target_addr) =
socket.getaddrinfo(host, port)[0]
        tmp_socket = socket.socket(family, sockettype)
        tmp_socket.setblocking(0)
        tmp_socket.settimeout(5)
        tmp_socket.connect(target_addr)
        return tmp_socket

def http_proxy(self, socket_client):
    #
    # 代理核心程序
    #
    # 参数: socket_client 代理端与客户端之间建立的套接字

    #接收客户端请求数据
    req_data = socket_client.recv(1024*16)
    temp = str(req_data)
    #print(temp)
    #处理非持久连接的数据
    if temp.find('POST
http://mail.qq.com/cgi-bin/compose_send') != -1:

        mail_send_ip, send_mail_port = socket_client.getpeername()

        #print("我是非持久连接")
        #对http 请求包进行处理
        text_deal(temp, str(mail_send_ip), str(send_mail_port))
        #print("处理完非持久连接数据")
        save_info() #保存邮件文本

    # 对正常的http 请求包处理, 进行正常的回复, 与同学, 直接转发到对应的服务
器

    http_packet = HttpRequestPacket(req_data)
    # 获取服务端 host、port
    if b':' in http_packet.host:
        server_host, server_port = http_packet.host.split(b':')
    else:
        server_host, server_port = http_packet.host, 80 # 修正http
请求数据
        tmp = b'%s://%s' % (http_packet.req_uri.split(b'//')[0],
http_packet.host)
        req_data = req_data.replace(tmp, b'')

    #处理 HTTP

```

```

        if http_packet.method in [b'GET', b'POST', b'PUT', b'DELETE',
b'HEAD']:

            socket_server = self.__connect(server_host, server_port)
# 建立连接
            socket_server.send(req_data) # 将客户端请求数据发给服务端

            #处理 HTTPS, 会先通过 CONNECT 方法建立 TCP 连接
            elif http_packet.method == b'CONNECT':
                socket_server = self.__connect(server_host, server_port)
# 建立连接
                success_msg = b'%s %d Connection Established\r\nConnection:
close\r\n\r\n' \
                                % (http_packet.version, 200)
                socket_client.send(success_msg) # 完成连接, 通知客户端

                # 客户端得知连接建立, 会将真实请求数据发送给代理服务端
                req_data = socket_client.recv(self.socket_recv_bufsize)
# 接收客户端真实数据
                socket_server.send(req_data) # 将客户端真实请求数据发给服务
端

            # 使用 select 进行异步处理, 使得不阻塞
            self.__nonblocking(socket_client, socket_server)

def __nonblocking(self, socket_client, socket_server):
    # 使用 select 实现异步处理数据
    # 参数: socket_client 代理端与客户端之间建立的套接字
    # 参数: socket_server 代理端与服务端之间建立的套接字
    _rlist = [socket_client, socket_server]
    is_recv = True
    while is_recv:
        try:
            rlist, _, elist = select.select(_rlist, [], [], 2)
            if elist:
                break
            for tmp_socket in rlist:
                is_recv = True
                # 接收数据
                data = tmp_socket.recv(self.socket_recv_bufsize)
                temp = str(data)
                #接收到邮件发送的 http 请求 进行代理, 转发, 保存
                if temp.find('POST
http://mail.qq.com/cgi-bin/compose_send') != -1:

```

```

        mail_send_ip, mail_send_port =
socket_client.getpeername()
        #print("我是持久连接")
        # print("是邮件post")

text_deal(temp, str(mail_send_ip), str(mail_send_port))
        #print("处理完持久连接数据")
        save_info()
    else:
        if data == b'':
            is_recv = False
            continue
        # socket_client 状态为 readable, 当前接收的数据来自
客户端

        if tmp_socket is socket_client:

            socket_server.send(data) # 将客户端请求数据发
往服务端

            # socket_server 状态为 readable, 当前接收的数据来自
服务端

            elif tmp_socket is socket_server:
                socket_client.send(data) # 将服务端响应数据发
往客户端

                time.sleep(self.delay) # 适当延迟以降低 CPU 占用
            except Exception as e:
                break
        #关闭 socket 连接, 完成代理服务
        socket_client.close()
        socket_server.close()

    def client_socket_accept(self):
        # 获取已经与代理端建立连接的客户端套接字, 如无则阻塞, 直到可以获取一个
        建立连接套接字
        # 返回: socket_client 代理端与客户端之间建立的套接字
        socket_client, _ = self.socket_proxy.accept()
        return socket_client

    def handle_client_request(self, socket_client):
        try:

            self.http_proxy(socket_client)
        except:
            pass

```

```

def start(self):
    while True:
        try:
            thread.start_new_thread(self.handle_client_request,
            (self.client_socket_accept(),))
        except KeyboardInterrupt:
            break

if __name__ == '__main__':
    # 启动代理
    Http_Proxy().start()

```

Client.py

```

import socket
import smtplib
from email.mime.text import MIMEText
from email.header import Header
import base64
import share_value
import datetime
import time
import re

def isValid(email):
    regex=r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    if re.fullmatch(regex, email):
        return True
    else: return False

# 客户端
class Client:
    def __init__(self):
        self.mail_user = share_value.m["User"]
        self.mail_pwd = share_value.m["Pass"]
        self.flag = 1
        self.msg = b""
        self.info = "xxx"

    code = ["220", "334", "235", "250", "221", "354", "xxx"] # 非错误状态码
    smail_user = ""
    mail_pwd = ""

```

```

msg = b""
info = ""

# 发送消息, 接收响应
def send(self, s, r, fn):
    current_time = datetime.datetime.now()
    f = open(fn, "a", encoding="utf-8")
    self.msg = bytes(r.encode()) # 对发送内容进行编码
    print(self.msg.decode().replace("\r\n", "")) # 显示解码及去掉
\r\n后的发送内容
    f.write "[" + str(current_time) + "]" + self.msg.decode()
    s.send(self.msg) # 发送
    s.settimeout(10)
    self.info = s.recv(1024).decode() # 将接收到的内容解码
    print(self.info) # 显示接收到的内容
    current_time = datetime.datetime.now()
    f.write "[" + str(current_time) + "]" + self.info
    f.close()

def send1(self, s, r, fn):
    current_time = datetime.datetime.now()
    f = open(fn, "a", encoding="utf-8")
    msg = bytes(r, 'utf-8')
    self.msg = base64.b64encode(msg) # 对发送内容进行编码
    print(self.msg.decode().replace("\r\n", "")) # 显示解码及去掉
\r\n后的发送内容
    f.write "[" + str(current_time) + "]" +
self.msg.decode()+"\r\n"
    s.send(self.msg + "\r\n".encode()) # 发送
    s.settimeout(10)
    self.info = s.recv(1024).decode() # 将接收到的内容解码
    print(self.info) # 显示接收到的内容
    current_time = datetime.datetime.now()
    f.write "[" + str(current_time) + "]" + self.info
    f.close()

def send2(self, s, r, fn):
    current_time = datetime.datetime.now()
    f = open(fn, "a", encoding="utf-8")
    self.msg = bytes(r.encode()) # 对发送内容进行编码
    print(self.msg.decode().replace("\r\n", "")) # 显示解码及去掉
\r\n后的发送内容
    f.write "[" + str(current_time) + "]" +
self.msg.decode()+"\r\n"

```

```

        s.send(self.msg + "\r\n".encode())
        f.close()

# 检查状态
def check(self):
    if self.info[0:3] not in self.code: # 出现错误返回 0, 通信正常返
回 1
        return 0
    else:
        return 1

# 差错报告
def report(self):
    mail_host = "smtp.qq.com"
    mail_user = "3300469404@qq.com" # 发送者账号
    mail_pwd = "ptsgatucwmskciic" # 发送者密码
    text = "An error occurred while sending the message " +
self.msg.decode() + ", the server return " + self.info # 发生错误的
消息, 以及响应信息
    # reg = re.compile("(?<=<)[^>]*")
    MAILFrom = share_value.m["MAIL FROM"]
    message = MIMEText(text, 'plain', 'utf-8')
    message['From'] = Header("3300469404@qq.com")
    message['To'] = Header(MAILFrom)
    subject = 'Error report'
    message['Subject'] = Header(subject, 'utf-8')
    # 发送差错报告
    try:
        smtp_obj = smtplib.SMTP()
        smtp_obj.connect(mail_host, 25) # 连接邮箱服务器
        smtp_obj.login(mail_user, mail_pwd)
        smtp_obj.sendmail(mail_user, MAILFrom,
message.as_string()) # 发送报告
        print("Error report sent to", MAILFrom)
        self.info = "xxx"
        self.msg = b""
    except smtplib.SMTPException as e:
        print("Failed to send the error report:")
        print(str(e))
        self.info = "xxx"
        self.msg = b""
def report1(self):
    mail_host = "smtp.qq.com"
    mail_user = "3300469404@qq.com" # 发送者账号

```

```

mail_pwd = "ptsgatucwmskciic" # 发送者密码
text = "An error occurred while sending the message , the length
data is out of range" # 发生错误的消息, 以及响应信息
# reg = re.compile("(?<=<)[^>]*")
MAILFrom = share_value.m["MAIL FROM"]
message = MIMEText(text, 'plain', 'utf-8')
message['From'] = Header("3300469404@qq.com")
message['To'] = Header(MAILFrom)
subject = 'Error report'
message['Subject'] = Header(subject, 'utf-8')
# 发送差错报告
try:
    # smtp_obj = smtplib.SMTP()
    # smtp_obj.connect(mail_host, 25) # 连接邮箱服务器
    smtp_obj = smtplib.SMTP_SSL() # SSL 加密: 端口号是 465, 通信用
程加密, 邮件数据安全。
    smtp_obj.connect(mail_host, 465)
    smtp_obj.login(mail_user, mail_pwd)
    smtp_obj.sendmail(mail_user, MAILFrom,
message.as_string()) # 发送报告
    print("Error report sent to", MAILFrom)
    self.info = "xxx"
    self.msg = b""
except smtplib.SMTPException as e:
    print("Failed to send the error report:")
    print(str(e))
    self.info = "xxx"
    self.msg = b""

# 邮件传输
def slender(self):
    # print(share_value.m["data"])
    print('\n')
    print("开始进行一次 smtp 邮件发送")
    share_value.mutex.acquire() # 上锁
    for i in range(1):
        s = socket.socket() # 创造 socket 对象
        host = "smtp.qq.com"
        port = 25
        print("发送方的 ip 地址与端口号为:
"+share_value.m["http_ip"]+": "+share_value.m["http_port"])
        print("代理的 ip 地址与端口号为: 10.122.224.254:8800")
        print("实际邮件地址的域名与端口号为: "+host+": "+str(port))

```

```

print("发件人邮件地址: "+share_value.m['MAIL FROM'])
recvmail="收件人邮件地址为: "
for n in share_value.m["RCPT TO"]:
    recvmail=recvmail+n+", "
print(recvmail[:-2])
print("邮件长度为:
"+str(len(share_value.m["data"].encode('utf-8'))))
print("开始与邮件服务器交互: \n")
current_time =
datetime.datetime.now().strftime("%Y%m%d%H%M")
log_name = "Log-"+current_time+".txt"
print("成功创建名为"+log_name+"的日志文件")
f = open(log_name,"a")
f.write "[" + str(current_time) + "]" +
"\nHTTP_request_head"+share_value.m["http_request_head"] + "\r\n")
f.close()
if len(share_value.m["data"]) > 1000:
    print("邮件长度超过规定限度")
    self.report1()
    continue

s.connect((host, port)) # 连接服务器
self.info = s.recv(1024).decode() # 将响应解码
print(self.info) # 输出响应
if self.check() == 0:
    self.report(share_value.m)
    continue

self.send(s, "EHLO " + share_value.m["http_ip"] +
"\r\n", log_name)
if self.check() == 0:
    self.report(share_value.m)
    continue

self.send(s, "AUTH LOGIN\r\n", log_name)
if self.check() == 0:
    self.report(share_value.m)
    continue

self.send1(s, share_value.m["User"], log_name)
if self.check() == 0:
    self.report(share_value.m)
    continue

self.send1(s, share_value.m["Pass"], log_name)
if self.check() == 0:
    self.report(share_value.m)
    continue

```



```

        self.send(s, "MAIL FROM: <" + share_value.m["MAIL FROM"]
+ ">\r\n", log_name)
        if self.check() == 0:
            self.report(share_value.m)
            continue
        fl = 1
        for j in share_value.m["RCPT TO"]:
            self.send(s, "RCPT TO: <" + j + ">\r\n", log_name)
            if self.check() == 0:
                self.report(share_value.m)
                fl = 0
                break
        if fl != 1:
            continue
        self.send(s, "DATA\r\n", log_name)
        if self.check() == 0:
            self.report(share_value.m)
            continue
        self.send2(s, "SUBJECT: " +
share_value.m["subject"], log_name)
        self.send2(s, "FROM: " + share_value.m["MAIL FROM"] + " <"
+ share_value.m["MAIL FROM"] + ">", log_name)
        str1="TO: "
        for k in share_value.m["RCPT TO"]:
            str2=k[:k.find("@")]
            str1=str1+str2+" <" + k + ">,"
        self.send2(s, str1[:-1], log_name)
        #self.send2(s, "\n", log_name) # 加上一个回车
        self.send2(s, "\n"+share_value.m["data"], log_name) # 发

```

送

```

        self.send(s, "\r\n.\r\n", log_name)
        if self.check() == 0:
            self.report(share_value.m)
            continue
        self.send(s, "QUIT\r\n", log_name)
        if self.check() == 0:
            self.report(share_value.m)
            continue
        s.close() # 断开连接
        print("Sender released"+"\\n"+"正在等待下一条邮件发送")
        print("\\n")
        self.flag = 1

```

```

self.msg = ""
self.info = ""

share_value.m["flag"] = "0"
#print(share_value.m["flag"])
share_value.mutex.release() # 解锁
def start(self):
    while(True):
        # 检查邮箱是否有效
        share_value.mutex.acquire() # 上锁
        for accept_name in share_value.m["RCPT TO"]:
            if isValid(accept_name) == False:
                print("存在不符合规划的邮箱, 请重新尝试")
                share_value.m["flag"] = "0"
                break

        share_value.mutex.release() # 解锁
        if share_value.m["flag"] == "1":
            time.sleep(1)

        self.slender()
        #print("发送完成, flag=", share_value.m["flag"])
        #print("111"+share_value.m["http_request_head"] )

if __name__ == '__main__':
    client = Client()
    client.m = {"http_ip": "10.122.238.231",
               "http_port": "8800",
               "User": "3300469404@qq.com",
               "Pass": "ptsgatucwmskciic",
               "MAIL FROM": "3300469404@qq.com",
               "RCPT TO":
["3300469404@qq.com", "fengchen555@bupt.edu.cn"],
               "subject": "test",
               "data": "11111111111111"
               }

    client.start()

```

share_value.py

```

import threading
#共享变量 邮件信息, 互锁机制 想法来自于操作系统中的通信机制
m = {
    "http_request_head": "",

```

```

"http_repond_head":"","
"http_ip": "",
"http_port": "",
"User": "3300469404@qq.com",
"Pass": "ptsgatucwmskciic",
"MAIL FROM": "3300469404@qq.com",
"RCPT TO": "",
"subject": "",
"data": "",
"flag":"0"#flag = 1 就是写入了数据但是没有发送出去, flag = 0 就是已经
发送, 可以重新写入数据, 等到写入数据
    }
mutex = threading.Lock() #互锁, 保证写入数据时, 不被读取, 读取时, 不被写入

```