Networking Lab Assignment 13

SMTP implementation

Albin Antony

20 April 2019

1 Simple Mail Transfer Protocol

1.1 Aim

To implement a subset of Simple Mail transfer Protocol using TCP.

1.2 Theory

SMTP is a simple ASCII protocol. This is not a weakness but a feature. Using ASCII text makes protocols easy to develop, test, and debug. They can be tested by sending commands manually, and records of the messages are easy to read. Most application-level Internet protocols now work this way (e.g., HTTP). After establishing the TCP connection to port 25, the sending machine, operating as the client, waits for the receiving machine, operating as the server, to talk first. The server starts by sending a line of text giving its identity and telling whether it is prepared to receive mail. If it is not, the client releases the connection and tries again later. If the server is willing to accept email, the client announces whom the email is coming from and whom it is going to. If such a recipient exists at the destination, the server gives the client the go-ahead to send the message. Then the client sends the message and the server acknowledges it. No checksums are needed because TCP provides a reliable byte stream. If there is more email, that is now sent. When all the email has been exchanged in both directions, the connection is released.

1.2.1 Protocol Overview

An SMTP Session consits of commands originated by an SMTP client and corresponding responses from the SMTP server so that the session is opened and session parameters are exchanged. A session may include zero or more SMTP transactions. A typical SMTP transaction consists of three command/reply sequences.

- 1. HELO command, to establish connection with server.
- 2. MAILFROM command, to establish the return address, also called return-path, reverse-path, bounce address, mfrom, or envelope sender.
- 3. RCPTTO command, to establish a recipient of the message. This command can be issued multiple times, one for each recipient. These addresses are also part of the envelope.
- 4. DATA to signal the beginning of the message text; the content of the message, as opposed to its envelope. It consists of a message header and a message body separated by an empty line. DATA is actually a group of commands, and the server replies twice: once to the DATA command itself, to acknowledge that it is ready to receive the text, and the second time after the end-of-data sequence, to either accept or reject the entire message.

1.3 Algorithm

Algorithm 1 Algorithm for SMTP server

```
START
Create TCP SOCKET
Bind SOCKET to a PORT
Start listing at the binded PORT for connection from CLIENT
WHILE TRUE:
    ACCEPT connection from CLIENT
    RECEIVE commands from CLIENT
    IF COMMAND == HELO
        RESPONSE:250
    ELSE IF COMMAND == MAILFROM
        RECEIVE mailid from CLIENT
        validate mailid
        IF valid
            RESPONSE:250 SENDER OK
        ELSE
            RESPONSE:421 SERVICE UNAVILABLE
   ELSE IF COMMAND == RCPTTO
        RECEIVE mailid from CLIENT
        validate mailid
        IF valid
            RESPONSE:250 RECIPIENT OK
        ELSE
            RESPONSE:421 RECIPIENT UNAVILABLE
    ELSE IF COMMAND == DATA
        IF HELO==250 & MAILFROM==250 &RCPTTO==250
            RESPONSE:354 Go Ahead, Enter data ending with <
               CRLF>.<CRLF>
            BREAK
        ELSE
            RESPONSE:421 SERVICE UNAVILABLE
READ message from CLIENT
IF COMMAND==QUIT
    STOP
```

Algorithm 2 Algorithm for SMTP client

```
START
CREATE TCP SOCKET
Connect to server using IP and PORT
WHILE INPUT != 'QUIT'
    READ commands from INPUT
    SEND commands to server
    RECEIVE response from server
STOP
```

1.4 Program

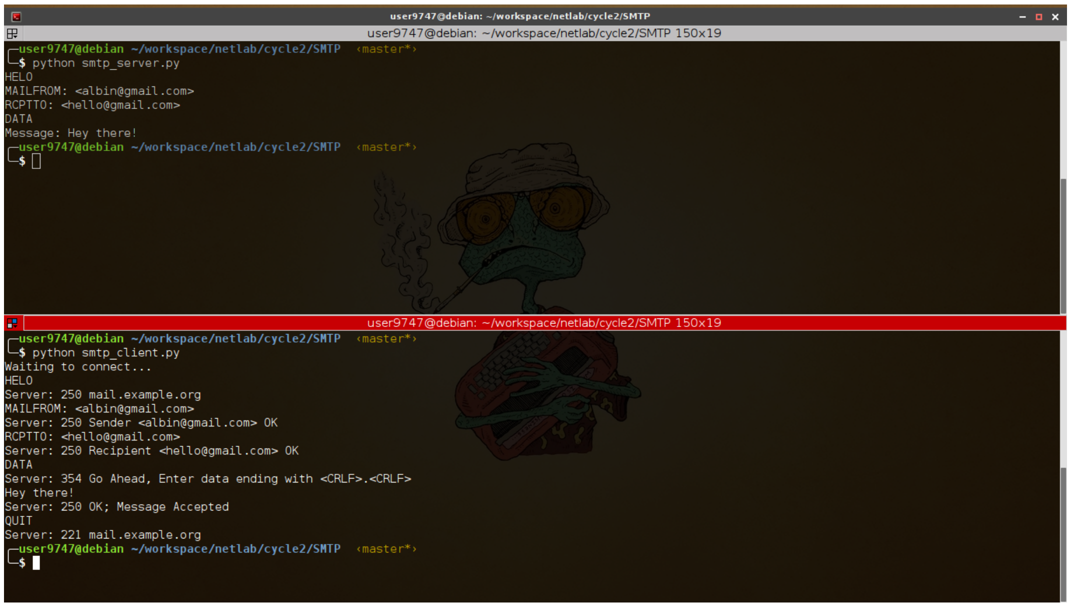
1.4.1 Server

```
import socket
import datetime
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind(("127.0.0.1",8080))
x=["albin@gmail.com","hello@gmail.com","jhondoe@gmail.com"]
s.listen(5)
conn,addr=s.accept()
while 1:
        mail=conn.recv(1024)
        tok=mail.split()
        print(mail)
        if tok[0]=='HELO':
                conn.send("250 mail.example.org")
        elif tok[0]=='MAILFROM:':
                flag1=0
                for i in x: #checking if mail is in the list
                        if tok[1]=="<"+i+">":
                                conn.send("250 Sender "+tok
                                    [1]+" OK")
                                flag1=1
                                break
                if flag1==0:
                        conn.send("421 Service Unavailable")
        elif tok[0]=='RCPTTO:':
                flag2=0
                for i in x: #checking if mail is in the list
                        if tok[1]=="<"+i+">":
                                conn.send("250 Recipient "+
                                    tok[1]+" OK")
                                flag2=1
                break
        if flag2==0:
            conn.send("421 Service Unavailable")
    elif tok[0]=='DATA':
        flag3=0
        if flag1==1 and flag2==1:
            flag3=1
            conn.send("354 Go Ahead, Enter data ending with <CRLF>.<CRLF>")
            break
        else:
            conn.send("421 Service Unavailable")




if flag3==1:
    buff=conn.recv(2048)
    print("Message: "+buff)
    conn.send("250 OK; Message Accepted")
mail=conn.recv(1024)
tok=mail.split()
if tok[0]=='QUIT':
    conn.send("221 mail.example.org")
```

1.4.2 Client

```
import socket
import sys
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("127.0.0.1",8080))
print("Waiting to connect...")
while 1:
    comm=raw_input()
    s.send(comm)
    print("Server: "+s.recv(1024))
    if comm=='QUIT':
        s.close()
        break
```

1.5 Output



user9747@debian: ~/workspace/netlab/cycle2/SMTP

user9747@debian: ~/workspace/netlab/cycle2/SMTP 150x19

```
┌user9747@debian ~/workspace/netlab/cycle2/SMTP  ‹master*›
└$ python smtp_server.py
HELO
MAILFROM: <albin@gmail.com>
RCPTTO: <hello@gmail.com>
DATA
Message: Hey there!
┌user9747@debian ~/workspace/netlab/cycle2/SMTP  ‹master*›
└$
```

user9747@debian: ~/workspace/netlab/cycle2/SMTP 150x19

```
┌user9747@debian ~/workspace/netlab/cycle2/SMTP  ‹master*›
└$ python smtp_client.py
Waiting to connect...
HELO
Server: 250 mail.example.org
MAILFROM: <albin@gmail.com>
Server: 250 Sender <albin@gmail.com> OK
RCPTTO: <hello@gmail.com>
Server: 250 Recipient <hello@gmail.com> OK
DATA
Server: 354 Go Ahead, Enter data ending with <CRLF>.<CRLF>
Hey there!
Server: 250 OK; Message Accepted
QUIT
Server: 221 mail.example.org
┌user9747@debian ~/workspace/netlab/cycle2/SMTP  ‹master*›
└$
```

- Client sends a HELO command server replies with 250
- Client sends from address with MAILFROM command server verifies and sends 250 if everything is ok.
- Client sends to address like above using RCPTO command.
- Client sends DATA command server when its ready to recieve message with a 354 response.
- Client sends message to server.Server saves it and sends 250 response.

1.6 Result

Implemented SMTP using TCP in Python2.7 and executed on Debian 9.4 Kernel 4.9 and outputs were verified.