Warren Quattrocchi
CSC 138
April 5, 2019

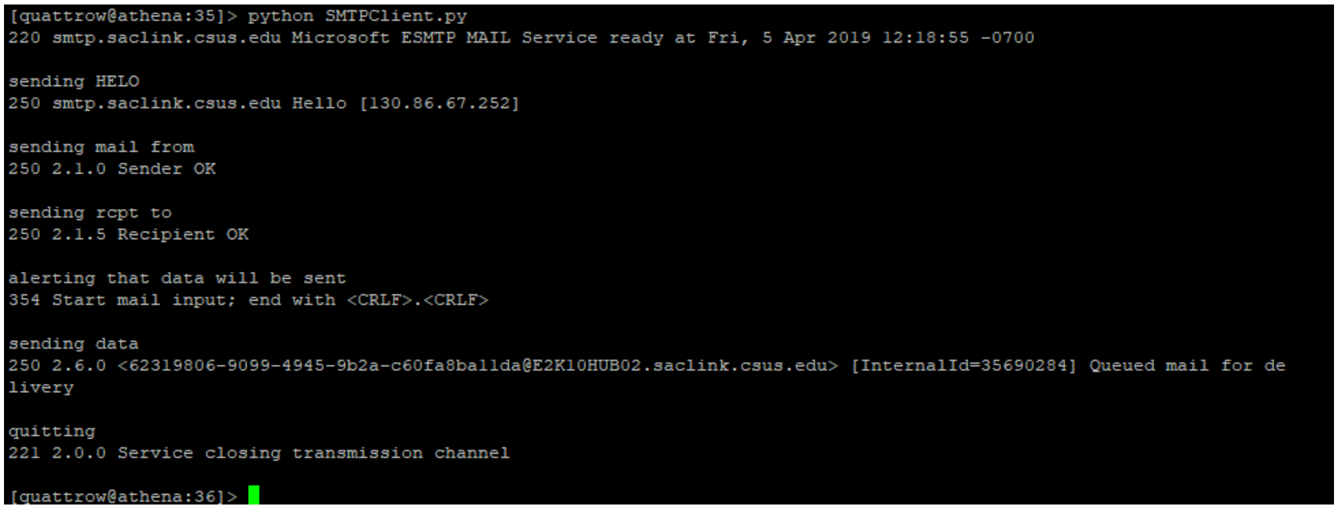
Socket Programming Assignment 2 – Mail Client

Source Code

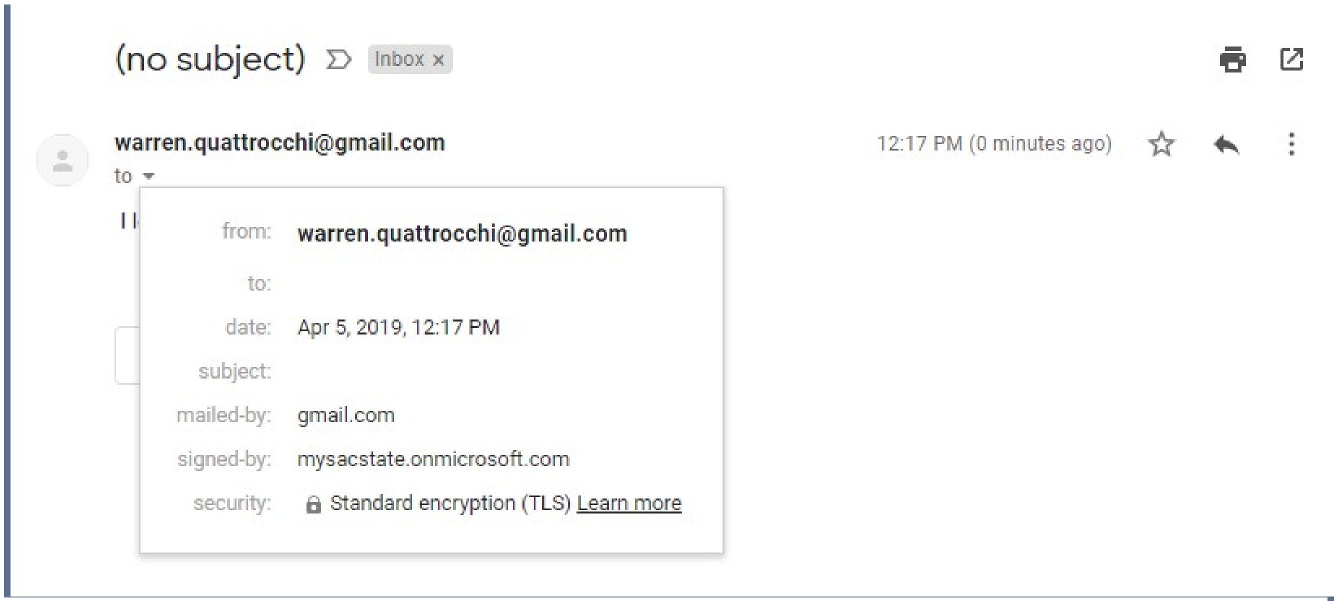
```
#Warren Quattrocchi
#CSC 138
#Socket programming assignment 2 - Mail Client

from socket import *

#message to be sent, must end in \r\n.\r\n
msg = "\r\n I love computer networks!"
endmsg = "\r\n.\r\n"
mailserver = 'smtp.csus.edu'

#crete the socket and connect to mailserver over port 25
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((mailserver,25))
recv = clientSocket.recv(1024).decode()
print recv
if recv[:3] != '220':
   print('220 reply not recieved from server.')

#send HELO to mailserver
print 'sending HELO'
heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())
recv1 = clientSocket.recv(1024).decode()
print recv1
if recv1[:3] != '250':
   print('250 reply not recieved from server.')

#mailFrom address
print 'sending mail from'
mailFromCommand = "MAIL FROM: <warren.quattrocchi@gmail.com>\r\n"
clientSocket.send(mailFromCommand.encode())
recv2 = clientSocket.recv(1024).decode()
print recv2
if recv2[:3] != '250':
   print('250 reply not recieved from server.')

#rcptTo email
print 'sending rcpt to'
rcptToCommand = "RCPT TO: <warren.quattrocchi@gmail.com>\r\n"
clientSocket.send(rcptToCommand.encode())
recv3 = clientSocket.recv(1024).decode()
print recv3
if recv3[:3] != '250':
   print('250 reply not recieved from server.')
```

```
#send data command
print 'alerting that data will be sent'
dataCommand = "DATA\r\n";
clientSocket.send(dataCommand.encode())
recv4 = clientSocket.recv(1024).decode()
print recv4
if recv4[:3] != '354':
   print('354 reply not recieved from server.')

#send data
print 'sending data'
clientSocket.send(msg.encode())
clientSocket.send(endmsg.encode())
recv5 = clientSocket.recv(1024).decode()
print recv5
if recv1[:3] != '250':
   print('250 reply not recieved from server.')

#close socket
print 'quitting'
quitCommand = "QUIT\r\n"
clientSocket.send(quitCommand.encode())
recv6 = clientSocket.recv(1024)
print recv6
if recv6[:3] != '221':
   print('221 reply not recieved from server')

clientSocket.close()
```

Output



```
[quattrow@athena:35]> python SMTPClient.py
220 smtp.saclink.csus.edu Microsoft ESMTP MAIL Service ready at Fri, 5 Apr 2019 12:18:55 -0700

sending HELO
250 smtp.saclink.csus.edu Hello [130.86.67.252]

sending mail from
250 2.1.0 Sender OK

sending rcpt to
250 2.1.5 Recipient OK

alerting that data will be sent
354 Start mail input; end with <CRLF>.<CRLF>

sending data
250 2.6.0 <62319806-9099-4945-9b2a-c60fa8ba11da@E2K10HUB02.saclink.csus.edu> [InternalId=35690284] Queued mail for de
livery

quitting
221 2.0.0 Service closing transmission channel

[quattrow@athena:36]>
```

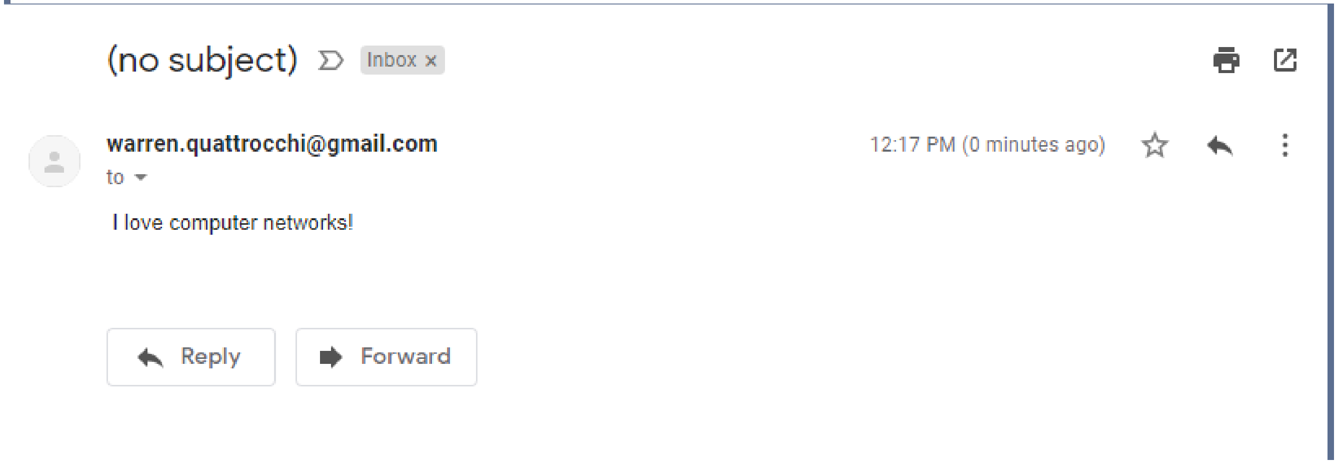


(no subject) Inbox ×

warren.quattrocchi@gmail.com 12:17 PM (0 minutes ago)
to

I l

from:	warren.quattrocchi@gmail.com
to:	
date:	Apr 5, 2019, 12:17 PM
subject:	
mailed-by:	gmail.com
signed-by:	mysacstate.onmicrosoft.com
security:	Standard encryption (TLS) Learn more



(no subject) Inbox ×

warren.quattrocchi@gmail.com 12:17 PM (0 minutes ago)
to

I love computer networks!

Analysis

While working on this programming assignment I learned a lot about the exchanges that happen between SMTP to establish a connection and send data. It is a very structured back and forth communication that must wait for a response from the mail server or time out after sending. The most difficulty I had came from figuring out where to go after the initial HELO request, which was made easier by looking into the SMTP format, which showed the order in which the client communicates with the server. Printing the incoming server responses made it simple to check for errors as the success code is returned by the server.