Lab Project 5: **SMTP Mail Client**

- **Number of members per group: 1 or 2**
- **Use Python**

This lab has two sub projects:

- **Part 1:** We first create a simple SMTP mail client that communicates with a local SMTP server to send an email.
- **Part 2:** We then send email through Gmail. The email includes the result of ping program.

Part 1/2: Exchanging email with a local mail server using sockets

Your task is to develop a simple mail client that sends email to a local server. Your client will need to connect to a mail server, dialogue with the mail server using the SMTP protocol, and send an email message to the mail server.

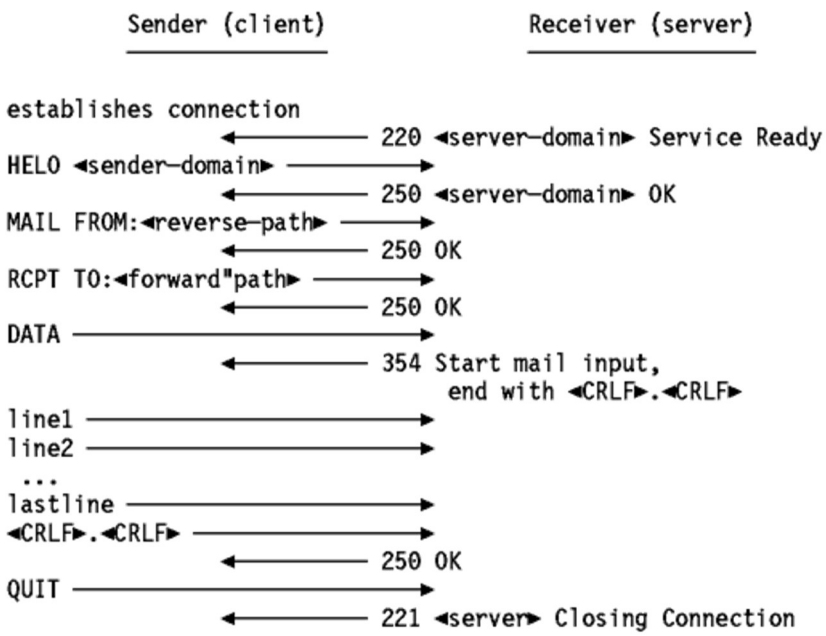
You can run a local SMTP server using the following command:

```
python -m smtpd -c DebuggingServer -n localhost:6000
```

Note: Python provides a module, called smtplib, which has built-in methods to send mail using SMTP protocol. However, we will not be using this module in this project, because it hides the details of SMTP and socket programming.

Here is how the SMTP protocol works:

1. The sender SMTP establishes a **TCP connection** with the destination SMTP and then **waits for the server to send a *220 Service ready* message or a *421 Service not available* message** when the destination is temporarily unable to proceed.
2. **HELO (HELO is an abbreviation for hello) is sent, to which the receiver will identify himself by sending back its domain name**. The sender SMTP can use this to verify if it contacted the right destination SMTP. The server responds with a multi-line ***250 OK*** message.
3. The sender now initiates the start of a mail transaction by sending a **MAIL** command to the receiver. This command contains the reverse-path which can be used to report errors. Note that a path can be more than just the *user_mailbox@host_domain_name* pair. If accepted, the receiver replies with a ***250 OK*.**
4. The second step of the actual mail exchange consists of providing the server SMTP with the **destinations for the message** (there can be more than one recipient). This is done by sending one or more **RCPT TO:*<forward-path>*** commands. Each of them will receive a reply ***250 OK*** if the destination is known to the server, or a ***550 No such user here*** if it isn't.
5. When all RCPT commands are sent, the sender issues a ***DATA*** command to notify the receiver that the message contents are following. The server replies with ***354 Start mail input, end with <CRLF>.<CRLF>***. Note the ending sequence that the sender should use to terminate the message data.
6. The client now sends the data line by line, **ending with the 5-character sequence <CRLF>.<CRLF>** line upon which the receiver acknowledges with a *250 OK* or an appropriate error message if anything went wrong.
7. The sender now ends the connection with a **QUIT** command, which will be answered with a ***221 Service closing transmission channel*** reply.



After sending the email, the expected output of your mail server should be:

---------- MESSAGE FOLLOWS ----------
SUBJECT: Greeting To you!
This is line 1
This is line 2.
------------ END MESSAGE ------------

Deliverables:

- **Demo your project to the TA in the lab**
- **Submit your code to Camino**

Please see the next page…

- **Part 2/2: Sending email using Python and Gmail**

We develop the following code:

- The program asks the user for her/his email address, password, and receiver's email address
- The program connects to gmail SMTP server
- The program runs the ping program and pings google.com twice
- The result of the ping program is sent to the receiver using smtp.gmail.com

Gmail: additional security setups:

- You need to activate: "2-Step Verification"
- You need to setup an "App password"
- Check the following Youtube, Tomi will get into it from 0:40 to 11:32
  https://www.youtube.com/watch?v=zxFXnLEmnb4&ab_channel=CodeWithTomi

Additional notes:

There are two ways to start a secure connection with your email server:

- Start an SMTP connection that is secured from the beginning using SMTP_SSL()
- Start an unsecured SMTP connection that can then be encrypted using .starttls()

We use the first approach in this project.
Gmail requires that you connect to port 465 if using SMTP_SSL()

You need to **allow login from less secure apps** in your gmail account. Otherwise gmail will complain about username and password.

Please make sure the subject of the email is "Server Ping Result!".

Deliverables:

- **Demo your project to the TA in the lab**
- **Submit your code to Camino**