# INFO1112 Week 10 Tutorial

## Networking and Application Layer

### 1: Talking to an SMTP server

Your machine will play the role of a client communicating with the “Simple Mail Transfer Protocol” (SMTP) server whose hostname is mail.usyd.edu.au.

The language used in this communication is defined by the SMTP protocol specified in the RFCs. SMTP is associated with a specific port, number 25, so that the server expects to hear the client using a specific language that sends a request to this port.

To initiate a connection on port 25 with mail.usyd.edu.au, use the terminal emulation protocol for TCP/IP through the telnet command as follow:

```
$ telnet mail.usyd.edu.au 25
```

The server should answer something along these lines if the connection is successful.

```
Trying 10.86.166.36...
Connected to mail.usyd.edu.au.
Escape character is '^]'.
220 staff.cs.usyd.edu.au. V1.4 ready at Mon, 02 Jul 2012
10:05:29 +1000
```

Try typing `HELO` *`hostname`* with any hostname and observe the response from the server. Leave the connection open and continue with the next activity.

## 2: Using the SMTP protocol

Now that the connection is opened, the next step is to ask the SMTP server to send an email. Try to send an email from your email account at the School of IT to yourself, use a different recipient address if you have one.

`MAIL FROM`: is used to indicate the sender of the message. Use angle brackets to encapsulate an email address, and terminate by pressing the "Enter" key. Note that `any` can be replaced by any valid name that you prefer. The example is as follows:

```
MAIL FROM:<any@it.usyd.edu.au>
```

If the server accepts the address by returning code 250, then tell it the recipient address. You should replace `your_unikey` with literally your unikey. Your unikey is of form abcd1234.
How can you specify more than one recipients?

```
RCPT TO:<your_unikey@uni.sydney.edu.au>
```

If the server accepts the address by returning code 250, then you can start giving the data to be sent by typing:

```
DATA
```

Make sure that the server answers you how to indicate that you are done writing your data: “end with a ’.’ on a line”. The last thing to do (next activity below) is to write a structured message with the appropriate fields so that the email client of your recipient, once connecting to its receiving mail server (IMAP), will nicely represent this information on the screen of the computer.

## 3: Formatting a message

The email information itself is represented in a format specified in another RFC. First, tell the server the format of the content (use the MIME format version 1.0):

```
MIME-Version: 1.0
```

Specify the correct UTC hour with +1000 indicating that (standard) Sydney time is 10 hours 00 minutes later than UTC. However, Sydney always switches to the Australian Eastern Daylight Time (AEDT) time from the first Sunday of October, so it is actually UTC +1100 now. To generate the date string, try the command `date --rfc-email` in your terminal.

```
Date: Mon, 10 Oct 2022 12:34:47 +1100
```

Specify the sender information, the recipient information, the subject and body of the message, and that this is a text message terminated by a dot in one single line. Please replace the highlighted fields with the information you typed in previous commands. Note, an intentional empty line is presented after `Content-Type`, this is needed to mark the end of the header message and start the real email body.

```
MIME-Version: 1.0
Date: Mon, 10 Oct 2022 12:34:47 +1100
From: Any Name <any@it.usyd.edu.au>
To: Your Name <your_unikey@uni.sydney.edu.au>
Subject: Testing SMTP
Content-Type: text/plain;

The body of my message...
...spans multiple lines.
.
```

Check whether you successfully received the message in your university Outlook inbox. Resend the above email with only the following fields in the header changed:

```
From: My Name <my@name.com>
To: You <your_unikey@uni.sydney.edu.au>
Subject: Emailing through telnet
```

What do you see?

Furthermore, in the Outlook web interface, click on the 3-dot icon -> View -> view message details to investigate what data was received. You should search for “Received: from” and is there anything looks familiar shows up?

Finally, terminates the connection with:

```
QUIT
```

## 4: More SMTP

After going through the successful process of emailing yourself, try to make intentional random attempts, predict what may happen and observe what the server actually replies. For example, switch the order of commands `MAIL`, `RCPT`, and `DATA`; execute MAIL more than one time; give more than one parameter to DATA, etc.
There also exists other SMTP commands. Try RSET and NOOP, what do they do?

## 5: Simple client

After going through the process of emailing yourself, create a simple python program using sockets that will allow the user to enter their email, the recipient, subject and the contents of the email. Identify points of the application layer which are constant or can be derived by your python program.
It is suggested that you start from reviewing and adapting the program developed in Homework “Python Echo Server”, however this time you want a client.

## 6: GPG and Email

Using your solution from last week, incorporate python-gnupg within your program to encrypt your email and send it to your classmate. Make sure you have generated a public and private key and have received a key. You can retrieve recipients from a public key server such as pgp.mit.edu.au.