PROGRAMMING ASSIGNMENT PART 1
Course Number: CSE 3101
Course Title: Computer Networking
Submission Deadline: 19th January 2018, 11:59 p.m
Maximum points: 40

GOAL: Design of a Simple SMTP E-mail Client and Server Over TCP

(A) E-mail Client:

- The following is a detailed description of the specification of your SMTP e-mail client:
  Your e-mail client will take the following 3 parameters on its command line:
  - The address to which the mail is being sent
  - The subject of the message as a string
  - The name of the file containing the body of the e-mail message
- The address is in the form user@host:port. The address (user@host:port) allows you to explicitly specify the location of the SMTP sever (the host name and the port number) of the user whom you wish to e-mail.
- This form of the address will be used for connecting to the SMTP server that you will develop as part of this assignment. Note that when you run your own SMTP server, the operating system will choose a port number for it. Your SMTP client can connect to your SMTP server using this user@host:port address form where the host is the host machine on which your SMTP server is running and the port is the port number assigned by the OS for your server.
- The subject of the message can be any text. Your client should ensure that multi-word strings could be used as the subject.
- The file name parameter simply contains the name of a text file that will be sent as the mail content of the e-mail message. The file can be created using any standard editor, and your client may assume that it is 7-bit ASCII text.
- For example, suppose you want to send an e-mail to bob@wagner.cse.univdhaka.edu with the subject "This is a test" and the e-mail message stored in the text file mail.txt, you will invoke your mail client using the following command:
  mail_client bob@wagner.cse.univdhaka.edu:2222 "This is a test" mail.txt
- Note that, your client should be able to detect simple errors such as incorrect number of arguments, incorrect file, etc and respond appropriately prompting the user with the current usage format.
- Upon starting, your client will need to establish a TCP connection to the specified mail server. If the address is of the form user@host:port , you will have to establish a connection with the SMTP server on the host machine host

at port number port.

- Upon establishing a connection, your client will receive a welcome message from the server. If the status code received from the server does not begin with a 2, you can assume an error occurred. In this case, your client should print out the status code and status phrase received from the server, close its connection, and promptly exit.
- Your client will now identify itself and its host using the standard HELO request. To do this you will need to find out the name of the host on which your client is executing.
- In return to its HELO request, the client will receive a reply from the server. If the code received from the server does not begin with a 2, you can assume an error occurred. In this case, your client should print out the status code and status phrase received from the server, close its connection, and promptly exit.
- Upon receiving a response to the HELO request, your client will send a MAIL FROM request, indicating the address of the sender of the message. This address should contain the user name of the person executing your mail client and also the name of the host machine on which the client is executing. There are several ways to retrieve the user name and host name in a LINUX environment.
- In return to its MAIL FROM request, the client will receive a reply from the server. If the code received from the server does not begin with a 2, you can assume an error occurred. In this case, your client should print out the status code and status phrase received from the server, close its connection, and promptly exit.
- Once the response to the MAIL FROM request is received, you must now use an RCPT TO command to indicate the recipient of your message. This is simply going to be the e-mail address given as the first parameter to your client. (If the address was in the form user@host:port, be sure to strip off the :port part first!)
- In return to its RCPT TO request, the client will receive a reply from the server. If the code received from the server does not begin with a 2, you can assume an error occurred. In this case, your client should print out the status code and status phrase received from the server, close its connection, and promptly exit.
- Once the response to the RCPT TO command is received, you can now send the DATA command to indicate you want to send the message. In return to its DATA request, the client will receive a reply from the server. If the code received from the server does not begin with a 2 or a 3, you can assume an error occurred. In this case, your client should print out the status code and status phrase received from the server, close its connection, and promptly exit.
- Once you receive the response to the DATA request, you can send the message. This will consist of sending both some standard headers and the message body as

well.

- You will need to include the basic headers, To:, From:, Subject:, and Date:. The e-mail addresses to use in the To: and From: headers were previously sent in the RCPT TO and MAIL FROM requests, and can be reused here. To get the subject for the Subject: line, use the subject given as the second parameter to your mail client. After the last header line, be sure to include a blank line to separate the headers from the message body.
- The message body is simply read in from the file named as the third parameter to your mail client. As it is read in from the file, it can now safely be copied down the socket to the mail server.
- Once you have sent the message, do not forget to indicate to the server that you are done sending by sending a . on a line by itself. You can safely assume that no one will be so malicious as to include a line with a . all by itself in the file sent above.
- At this point, your client is about ready to wrap up. It now issues a QUIT request, awaits its reply, and closes its connection to the server. Your client can now exit.
- Once again, if the client receives a code that does not begin with 2 in response to the message submission or its QUIT request, your client should print the status code and status phrase, and exit.

(B) E-mail Server:

The following is a detailed description of the specifications of your SMTP e-mail server:

- Your server is relatively simple, and just appends each incoming e-mail message to a file named after the user the mail was intended for. If e-mail was intended for user bob, there will be a file called bob in the server's directory containing all of the mail for this user.
- Each user that your server will accept mail for must have a file created for the user before the server accepts mail for that user. Consequently, you will need to create empty files for each user that your server will store mail for. If such a file does not exist for a user, and the server is sent e-mail for that user, your server will reject the message. Note that your server should send an appropriate status code and message to the mail client who sent the message indicating this.
- Unlike your e-mail client, your e-mail server will not take any command line parameters. It is just started on its own. As part of its initialization, it will have the operating system choose a port number for it, and will print it out to the screen This way, you will know the port number used by your server, so you can include it in the address used by your e-mail client. After doing this, the server waits for a connection request from your e-mail client. Assuming that your mail server has been assigned port number 2222 and is running on the host wagner.cse.univdhaka.edu, you will be able to send an e-mail to bob using your

e-mail client as follows:

mail_client bob@wagner.cse.univdhaka.edu:2222 “Hello bob” mail.txt

- When your server receives a connection, it sends a status code and welcome message to the client. Be sure to use a successful status code, as discussed in class and in the course text, or else the client will abort the connection. The choice of text for the welcome message is up to you.
- The server will now receive a HELO request. It should verify that the request was actually HELO, but does not need to check what comes after HELO in the client request. An appropriate status code and phrase should be sent back to the client.
- The server will now receive a MAIL FROM request. It should verify that the request was actually MAIL FROM, but does not need to check what comes after FROM in the client request. An appropriate status code and phrase should again be sent back to the client.
- The server will now receive an RCPT TO request. It should verify that the request was actually RCPT TO, and should also pick the user name out of the address sent along. (You can assume that a correctly structured e-mail address was sent, since your client was the program that sent it.)
- Before generating a response, the server should attempt to open the given user's mail file. If this fails because the file does not exist, the incoming message should be rejected by sending back an appropriate status code and message. If the mail box exists and the mail can be accepted, an appropriate response should be sent back.
- The server will now receive a DATA request. It should verify that the request was actually DATA, but does not need to check what comes after DATA in the client request. An appropriate status code and phrase should again be sent back to the client.
- The client now sends the server the message, terminated by a . on a line all by itself. (You can safely assume there will only be one such line, and it will come at the end of the message.) The message will need to be read from the incoming socket and appended to the user's mailbox, including all headers. A single blank line should be used to separate individual messages in the mailbox. The . terminating the message should not be appended to the mailbox. When done, an appropriate status code and message should be sent to the client.
- The client now sends a QUIT command to the server. Once again, the server should send an appropriate status code and phrase back to the client. As mentioned earlier, only one message per connection will be sent. The server can now close its connection.
- With the connection closed, the server should return to accept new connections

from additional clients. The server need not process multiple client requests at the same time, but the server should be able to process several client requests consecutively. To terminate the server, the user should press Ctrl-C.

- Note that, anytime during the dialogue between client and server, whenever the server receives a message that it does not expect it should respond with the appropriate status code and close the connection. However, the server should return to handle new connections.

(C) Final Notes and Hints

Here are some final notes and hints to help you in your assignment:

- Be sure to skim through RFC 821, which describes the SMTP protocol in detail.
- The man pages are your friends.
- At many points in the client and server, you will need to do things to parse text and break it into its various pieces. There are several simple C & C++ functions to help you do this. You will also come across the complementary task of composing strings out of other data. There are again several string library functions for these tasks.
- Do not forget that in most Internet protocols like SMTP, protocol messages that need a line terminator want a carriage return and a line feed. For example, the . terminating a message being sent should be surrounded by both a carriage return and a line feed.
- While you are encouraged to adopt good programming practices, your coding style is NOT subject to marking.
- Language and Platform: You are free to use C or C++ to implement this assignment (note that you can choose only one of these languages for the entire assignment, not a combination of them). Your assignment will be tested on Linux Platform. Make sure you develop your code under Linux.
- Submission: All server code files and header files must begin with mail_server, and all client files must begin with mail_client. Using a Makefile is required. It will make your life, as well as your marker's, easier in the long run. All of these files must be submitted with your assignment.
- .How to Submit: TBA
- Late Submission Penalty: Late penalty will be applied as follows:
  - 1 day after deadline: 10% reduction
  - 2 days after deadline: 20% reduction
  - 3 days after deadline: 30% reduction
  - 4 days after deadline: 40% reduction
  - 5 or more days late: NOT accepted
- Plagiarism: You are to write all of the code for this assignment yourself. You are not allowed to use C functions such as system() or anything similar to execute

other programs for you. All source codes are subject to strict checks for plagiarism. These checks may include comparison with available code from Internet sites and assignments from previous semesters. In addition, each submission will be checked against all other submissions of the current semester. Please note that we take this matter quite seriously. LIC will decide on appropriate penalty for detected cases of plagiarism. The most likely penalty would be to reduce the assignment mark to ZERO.

- IMPORTANT NOTE: Now that you know how, always use good ethical judgement and do not get into the habit of sending forged e-mails; after all, a careful examination of the full e-mail header will reveal your forgery.