HW #4: Simplified SMTP on TCP, Due Date*: 11:59pm 02/27/2020, Cutoff Date: 11:59pm 02/29/2020**
Submission: (1) Upload all source code (.java) files to both Blackboard and the Virtual Servers (cs3700a and cs3700b), AND (2) set up and test java programs (.class files) on cs3700a and cs3700b (See Part II for details)
****Submission will NOT be accepted after the cutoff deadline**

An option of peer programming: You may choose to work on this assignment individually or in a team of two students. If you choose to work in a ***team of two students***, you **must** 1) **add** both team members' first and last names as the **comments** on Blackboard when submitting the .java files (both team members are required to submit the same .java files on Blackboard), and 2) put a ***team.txt*** file including both team members' names under your HW4/ on cs3700a (both team members are required to complete Task II under both home directories on cs3700a). For grading, I will randomly pick the submission in one of the two *home directories* of the team members on **cs3700a** and **cs3700b**, and then give both team members the same grade. If the team information is *missing* on Blackboard or cs3700a, two or more submissions with similar source codes with just variable name/comment changes will be considered to be a violation of the Integrity described in the course policies and both or all will be graded as 0.

Grading: Your programs will be graded via testing and points are associated with how much task it can complete. A program that cannot be compiled or crashes while running will receive up to 5% of the total points. A submission of .java files that are similar to any online Java program with only variable name changes will receive 0% of the total points.

Programming in Java is highly recommended, since all requirements in this assignment are guaranteed to be supported in Java. If you choose to use any other language such as Python or C/C++, it is YOUR responsibility, before the cutoff deadline, to (2) set up the compiling and running environment on cs3700a and cs3700b, (2) make sure that I can run/test your programs in your home directory on cs3700a and cs3700b, and (3) provide a README file under HW4/ on cs3700a to include the commands that I need to use.

Part I (85%): Write a ***SMTP client program*** and a ***SMTP server program*** to implement the following simplified SMTP protocol based on TCP service. Please make sure your program supports multiple clients. You must use the port assigned to you and you may hard code it in both client and server programs.

- "**Telnet <your server's ip address> <port>**" can be used to test your SMTP server program first (especially those "503" responses), then your SMTP client can be tested using your SMTP server program.
- **SMTP Client Program** (NOT a TELNET client!!!):
  1. Display a message to ask the user to input the Host Name (or ip-address) of your SMTP server.
  2. Buildup the TCP connection to your SMTP server with the Host Name input by User at the given port. Catch the exception, terminate the program, and display error messages on the standard output if any. Wait for, read, and display the "220" response from the SMTP server.
  3. Display prompt messages on the standard output to ask the user to input sender's email address, receiver's email address, subject, and email contents, respectively. Since there may be multiple lines in email contents, the prompt message displayed by your Client program MUST prompt the ending signature pattern like "." on a line by itself.
  4. Use the user inputs collected above in Step 3 for the following 3-phase data transfer procedure (see step 3 on server side). In each of the following steps a. through e., display the **RTT (round-trip-time)** of each conversation in millisecond (e.g., RTT = 212.08 ms).
     a. Send the "HELO <sender's mail server domain name>" command (e.g., HELO xyz.com) to the SMTP server program, wait for server's response and display it on the standard output.
     b. Send the "MAIL FROM: <sender's email address>" command to SMTP server, wait for SMTP server's response and display it on the standard output.
     c. Send the "RCPT TO: <receiver's email address>" command to the SMTP server program, wait for SMTP server's response and display it on the standard output.
     d. Send the "DATA" command to the SMTP server program, wait for SMTP server's response and display it on the standard output.
     e. Send the Mail message to the SMTP server. The format of this Mail message MUST follow the format detailed on the **slide titled "Mail message format"**. Wait for SMTP server's response and display it on the standard output.
  5. Display a prompt message to ask the User whether to continue. If yes, repeat steps 3 through 5. Otherwise, send a "QUIT" command to the SMTP server, display SMTP Server's response, close TCP connection, and terminate the Client program.
- **SMTP Server Program**: (server's ip or client's ip can be its dns name below.)
  1. Listen to the given port and wait for a connection request from a SMTP Client.
  2. Create a new thread for every incoming TCP connection request from a SMTP client. Send the "220" response including server's ip address or dns name to the SMTP client.
  3. Implement the following 3-phase data transfer procedure (see step 4 on client side):
     a. Wait for, read, and display the "HELO …" command from the SMTP client. If the incoming command is NOT "HELO …", sends "503 5.5.2 Send hello first" response to the SMTP client and repeat step 3.a.
     b. Send the "250 <server's ip> Hello <client's ip>" response to the SMTP client.

c. Wait for, read, and display the "MAIL FROM: …" command from the SMTP client. If the incoming command is NOT "MAIL FROM: …", sends "503 5.5.2 Need mail command" response to the SMTP client and repeat step 3.c.
d. Send the "250 2.1.0 Sender OK" response to the SMTP client.
e. Wait for, read, and display the "RCPT TO: …" command from the SMTP client. If the incoming command is NOT "RCPT TO: …", send "503 5.5.2 Need rcpt command" response to the SMTP client and repeat step 3.e.
f. Send the "250 2.1.5 Recipient OK" response to the SMTP client.
g. Wait for, read, and display the "DATA" command from the SMTP client. If the incoming command is NOT "DATA", send "503 5.5.2 Need data command" response to the SMTP client and repeat step 3.g.
h. Send the "354 Start mail input; end with <CRLF>.<CRLF>" response to the SMTP client.
i. Wait for, read, and display the Mail message from the SMTP client line by line. (hint: "." is the ending signature.)
j. Send the "250 Message received and to be delivered" response to the SMTP client.

4. Repeat Step 3 until the "QUIT" command is read. Upon receiving "QUIT", send the "221 <server's ip> closing connection" response to the SMTP client and go to Step 5.
5. Close all i/o streams and the TCP socket for THIS Client, and terminate the thread for THIS client.

Part II (15%): Test your programs on the Virtual Servers in the cloud and your laptop/home computer.

 **Warning**: to complete this part, especially when you work at home, you must first (1) **connect to GlobalProtect** using your NetID account (please read "**how to connect to GlobalProtect …**" at https://msudenver.edu/vpn/); then (2) **connect to the virtual servers cs3700a and cs3700b** using ***sftp*** and ***ssh*** command on MAC/Linux or ***PUTTY*** and ***PSFTP*** on Windows.

ITS only supports GlobalProtect on MAC and Windows machines. If your home computer has a different OS, it is your responsibility to figure out how to connect to cs3700a and cs3700b for programming assignments and submit your work by the cutoff deadline. Such issues cannot be used as an excuse to request any extension.

1. MAKE a directory "**HW4**" under your home directory on **cs3700a**.msdenver.edu and **cs3700b**.msudenver.edu, a subdirectory "**server**" under "**HW4**" on **cs3700a.msudenver.edu**, and a subdirectory "**client**" under "**HW4**" on **cs3700b.msudenver.edu**.
2. UPLOAD and COMPILE the ***server*** program under "**HW4/server**" and the ***client*** program under "**HW4/client**" on the VMs.
3. TEST ***the server program*** running on **cs3700a**.msudenver.edu together with ***a client program*** running on your laptop or lab computer and ***another client program***, simultaneously, running on **cs3700b**.msudenver.edu to test all the possible cases.
4. SAVE a file named ***testResultsClient.txt*** under "**HW4/client**" on **cs3700b**.msudenver.edu, which captures the outputs of your ***client*** program when you test it. You can use the following command to redirect the standard output (stdout) and the standard error (stderr) to a **file** on UNIX, Linux, or Mac, and view the contents of the file

```
java prog_name_args | tee testResultsClient.txt //copy stdout to the .txt file
      //if you want, you may also use "script" command instead of the "tee" command
      //to write both stdin and stdout into testResultsClient.txt while testing your
      //client program on cs3700a.  For how to use "script", see
      // https://www.geeksforgeeks.org/script-command-in-linux-with-examples/
      //or Google "script command in Linux" if the above link is broken.
cat file-name     //display the file's contents.
```