CMSC 417 Computer Networks **Fall 2012**

Programming Assignment 1

Assigned: Sep 10th *Due: Sep 17th, 11:59:59 PM.*

1 Description

In this assignment you will develop the timestamp server for the client you developed in the previous assignment. This time, instead of using UDP you have to use TCP for both the server and the client.

2 Protocol

This section describes the communication protocol between the server and the client. The protocol is similar to the first assignment. However, the size of the nonce is always 40 bytes long this time and not 32.

1. Initialization (Client → Server): The initialization message has exactly two fields. Your username and a message of your choice. The total message size will be arbitrarily long. The fields in the message are separated by '`#`'. The sequence '`\n\n\r`' denotes the end of the message. An example of an initialization message might be: `lex#networks\n\n\r`.

2. Timestamp (Server → Client): The timestamp message has exactly three fields. The first is a random nonce that is produced from the server. The length of the nonce is always **40** bytes. The second field is the SHA1 digest of the nonce, username, the message you send to the server and the timestamp the server sends you. The length of the digest is always 40 bytes. The third is the string representation of the timestamp. The server sends the timestamp as in Unix time format (elapsed seconds from January, 1, 1970). The string representation of the timestamp is 10 bytes long. The fields in the message are separated by '`#`'. Therefore, remember to count the number of the `#` when you compute the size of the message server sends you. An example of a timestamp message is:

```
 60feaf...b695baa3#99800b8c...eade3224ab345#1346177187
|---nonce(40B)----|-------digest(40B)------|-timestamp(10B)-|
```

- You should modify the client of the previous assignment to use TCP.
- You will submit only your server.
- You **must** implement your server using `select(2)`. You can find more details about how network servers are implemented in your book TCP/IP sockets in C. The select man page as well as pages 130-133 of the TCP/IP sockets in C book will help you build a select based server.
- In order to produce the digest, you should use the function `p_ascii_hash()` we provide.

- To produce a nonce, use the `p_ascii_hash()` and pass a random number as an argument.
- The `p_ascii_hash()` takes a `void *` and its size as arguments.
- The `test.c` file includes uses of the `p_ascii_hash()` function.
- To compile your server code, use the Makefile we provide. In the Makefile, we assume that the server is in the `tserver.c` file.
- Your code must be -Wall clean on gcc. Do not ask the TA for help on (or post to the forum) code that is not -Wall clean.
- The TA will post information on Piazza about how to submit your assignment.