CMSC 417 Computer Networks **Fall 2021**

Programming Assignment 3

Assigned: October 7 *Due: October 22, 11:59:59 PM.* *Weight: 1.5x*

1 Introduction

In this project, we will be providing a chat client and will host a chat server. Clients allow users to communicate with one another by connecting to the server and interacting with it in accordance to an application protocol. Through this protocol, the server allows clients to engage in group chats in chat rooms and send private messages to one another.

Your task will be to reverse engineer the chat server and its protocol and use this information to write a compatible [illegible] 20 point penalty and instead implement a c[illegible]

2 Client (Pro[illegible]

The client is available [illegible] rectory. The provided client takes single opt[illegible]

1. **-u** = Run with [illegible] ience.

While running, th[illegible] mmands are preceded by a backslash. Not e[illegible] supports the following commands:

1. **connect** <**IP**[illegible] to a new chat server, specified by the [illegible]
2. **disconnect** = [illegible]
3. **join** <**Room**[illegible] ting it if it does not already exist. The Password is optional, with the default being the empty string. Users may only join rooms for which they know the password. Both Room and Password must be less than 256 characters in length.
4. **leave** = If in a room, this exits the room. Otherwise, it leaves the server.
5. **list users** = List all users. If in a room, it lists all users in that room. Otherwise, it lists all users connected to the server.
6. **list rooms** = List all rooms that currently exist on the server.
7. **msg** <**User**> <**Message**> = Send a private message to the specified user. User must be less than 256 characters in length, and the Message must be less than 65536 characters in length.

8. **\nick <Name>** = Set your nickname to the specified name. Name must be less than 256 characters in length.

9. **\quit** = Exits out of the client.

All other input is interpreted as a message being sent to the room the user is currently in.

3 Server (Hosted)

We will be running a server; see Piazza for details.

4 Replica Implementations

4.1 Client

Your replica client m eed to support the -u option. The outpu e provided client for all sequences of com t implementation by comparing the output

4.2 Server

Your replica server m

1. **-p <Number>** as a base-10 integer. Must be specifie

5 Grading

Your project grade w ned when connecting to your server. We wi put and output mode (i.e., without the **-u** c al input against your server and the referen

If you elect to im 20 point penalty), we will `diff` the output of your client when fed identical input against the reference client in standard input and output mode.

We strongly encourage you to write tests against the reference implementation to compare against your own implementation.

6 Additional Requirements

1. Your code must be submitted as a series of commits that are pushed to the origin/master branch of your Git repository. We consider your latest commit prior to the due date/time to represent your submission.

2. The directory for your project must be called 'assignment3' and be located at the root of your Git repository.

3. You must provide a Makefile that is included along with the code that you commit. We will run 'make' inside the 'assignment3' directory, which must produce either a 'rserver' or 'rclient' executable also located in the 'assignment3' directory.

4. You must submit code that compiles in the provided VM, otherwise your assignment will not be graded.

5. Your code must be -Wall clean on gcc/g++ in the provided VM, otherwise your assignment will not be graded. Do not ask the TA for help on (or post to the forum) code that is not -Wall clean, unless getting rid of the warning is the actual problem.

6. You are not allowed to work in teams or to copy code from any source.