CSE 160 Computer Networks

University of California, Merced

Project 4: Chat Client and Server

Introduction

Your assignment is to design and implement a chat client and server. Design an application similar to Project 3's test application to demonstrate basic use of a chat client and server.

Objectives and Goals

The server should be setup to listen on port 41 at node id 1. Once a client receives a "hello" command shown below, it will attempt to make a connection to the server. This command should be implemented in command.h in a similar fashion to the other commands.

Connecting to Server

Format: *hello **[username] [clientport]**\r\n*

Example: *hello acerpa 3\r\n*

Note at the end of the command there is '\r' and '\n'. '\r' is a carriage return character and '\n' signifies a new line character. Together these characters are used as a terminating character for a message. This format should be used in all commands since a command can take up multiple TCP packets. Once a connection has been established between client and the server, a tcp data pack should be sent with the message above excluding the port. This will tell the server the name of the client.

Broadcasting a Message

Format: *msg **[message]**\r\n*

Example: *msg Hello World!\r\n*

Once again create a command for the client to receive through injected packets. The above command should be written onto your socket client side. Once the server receives this message, it should then broadcast this message to all connected clients. The message that is sent to all the clients should include the initial client username that sent the message.

Unicasting a Message

Format: *whisper **[username] [message]**\r\n*

Example: *whisper acerpa Hi!\r\n*

This function is similar to the previous broadcast message. However, instead of the server broadcasting it to all the clients, it will only send the message to the client specified.

Print Users

Format: *listusr\r\n*

This message should be sent from the client to the server. The server should reply to the client that made the request with a list of users that are currently connected to the server.

Example Reply: listUsrRply acerpa, ayadav6, jshanmugasundaram, dwinlker2\r\n

Deliverables

What is expected on the due date is the following:

- Source code of the TinyOS implementation with working Chat Client and Server modules
- A single page document describing the design process and your design decisions.
- A document with short answers to the discussion questions.

A physical copy of the document and related questions is required on the due date. All documents and a copy of the source code should be submitted to the class web page. Please create a compressed tarball such as Student-Name-proj4.tar.gz. You must do this before class on the day that it is due.

Additionally, you will need to demonstrate that the code you submitted works in the next lab, and be able to describe how it works. Also discuss the design process in the code.

Discussion Questions

1. The chat client and server application as described above uses a single transport connection in each direction per client. A different design would use a transport connection per command and reply. Describe the pros and cons of these two designs.

2. Describe which features of your transport protocol are a good fit to the chat client an server application, and which are not. Are the features that are not a good fit simply unnecessary, or are they problematic, and why? If problematic, how can we best deal with them?
3. Read through the HTTP protocol specification covered in class. Describe which features of your transport protocol are a good fit to the web server application, and which are not. Are the features that are not a good fit simply unnecessary, or are they problematic, and why? If problematic, how can we best deal with them?
4. Describe one way in which you would like to improve your design.

Please be concise with your answers.

Grading Guidelines

Each part of the project is graded on a 5 point (0-4) scale, multiplied by the weight of the project. The weighted grades from all parts of the project are added together to produce the final grade.

The five point scale is based on how well you show your understanding of the problem, and in case of code how well your implementation works:

0 – nothing turned in
1 – show minimal understanding of the problem / most things don't work
2 – show some understanding of the problem / some things work
3 – show a pretty good understanding of the problem / most things work
4 – show excellent understand of the problem / everything works

The weights for project 4 are:

90% - Chat Client and Server implementation
5% - Write-up design decisions
5% - Discussion questions

Your submission will be graded on correctness, completeness, and your ability to explain its implementation.