Laura Maisenhelder
6/7/2019
CS544

README

To compile and run the programs:

SSH into tux.cs.drexel.edu.

Access the home directory for lmm326

(if you are not able to access this please let me know and I will provide you my password to log in as me on tux)

To compile: From the home directory, enter javac ConcurTCPServer.java and hit enter

Then enter javac TCPClient.java and hit enter

To run: enter java ConcurTCPServer and hit enter, this starts the server

Open a second tux window and in the same $/home/ directory enter java TCPClient to start the client

It will ask you to log in, please use the following:

Username: Bob

Password: 1234

game selection: T

From there you should be able to follow the prompts to play the game.

Testing & Analysis

The following tests were done to test the protocol's implementation:

In Authentication

Username should be Bob, entered Ana instead, this was recognized by the server as an incorrect username. Password should be 1234, entered 1423 instead, this was recognized by the server as an incorrect username.

Game selection can only be T right now, entered S, this was recognized by the server as an unsupported game.

Also tried input that would make the whole message being sent to the server too short or long (which could lead to an out of bounds exception), such as username Simpson or password 01 and this was recognized as incorrect input on the client side before the message was sent to the server.

In Game Initialization

When the user is prompted to begin a game, saying yes to playing a game has the correct response of starting the game on the client side. Saying no to this same prompt has the correct response of sending a message to alert the server, then displaying the Goodbye! message and disconnecting.

In Client's Turn

Played the game to client winning, this was recognized correctly and the server informed the client. Played the game to server winning, this was recognized correctly and the server informed the client. Played the game to stalemate this was recognized correctly and the server informed the client.

On client side, during game play, entered spaces to play that were not in the options of A1 – C3, such as B7 or asdf3523 and these inputs were identified by the client as erroneous and the player is prompted to try again.

Entered spaces to play that were already occupied and these inputs were identified by the client as erroneous and the player was prompted to try again.

Concurrency

I tested for concurrency by trying to have a third command line window open to have a second client connect to the server, this client did not connect, however I did follow the instructions in Chapter 11 of Forouzan for setting up a concurrent server, the methods are cited in my source code comments.

Summary

Overall, I find the protocol implementation is robust in that it doesn't give you the opportunity to do something out of sequence. I do think that the protocol is tough to crack because in my implementation of it, input is limited. You could enter a bad username, password or game option during authentication and the server would inform you that the input was incorrect and you need to try again. You could try to play a space that doesn't exist or is already occupied and the client would catch that before the board was even sent to the server, and the client would tell you to try again. If you do play a space the board is sent to the server where it is checked for a win before the server even plays (so if the client already won there's no need for the server to take a turn, it can just tell the client that it has won), and it is checked after the server plays as well to inform the client of a possible server win before the client plays takes another turn.

You are not able to move forward in the process until your input is accepted. The user interface doesn't really allow for things to even be done out of sequence either, your prompts are limited – there's no menu to click on to try to pick a game before you log in for example. This careful implementation that guides the user through only what the protocol wants them to be doing provides a robust experience.