Project 3 – A Networked Boggle Protocol Design

Due date of protocol design – October 15th (Monday, in class)

Students may work on this project alone, in a group of two, or in a group of three. Students who work alone will later implement a game client and server in the same programming language. Students in groups of two must implement a game client and server in different programming languages. Students in groups of three must implement one server and two clients, where the two clients are in different programming languages.

Deliverable in class – A document listing a) all possible messages sent to the server as well as all possible messages sent to the client, and b) the formatting of those messages so that the server/client is able to identify connection requests, player usernames, board configurations, word guesses, results, etc. This deliverable is worth 10 points.

Intro

Boggle © is a challenging word-finding game that can be played in as little as 3 minutes and can be played by two or more players. You can find the official rules of the game at this website: http://www.hasbro.com/common/instruct/boggle.pdf . (Ignore the Boggle challenge cube.)

Our goal is to turn Boggle into a multi-player networked game consisting of a single server program which will interact with two or more clients. New players can start up their clients and join a game whenever it is in the Stage 1 "joining mode", but they cannot join a game already in progress.

This project assignment will be broken up into two phases. Phase 1 is to design the client/server protocol on paper. Phase 2 is to actually implement the client and server programs. This specification discusses Phase 1.

Design the Client/Server Protocol

We already know how to create a simple client/server program that sends messages through TCP sockets from Project 1. This project expands on this concept but requires you to design specialized strings sent to/from the client and server to allow everyone to move correctly through the various stages of gameplay. This document will be due in class on the due date and must include all names of the people in the project. To assist you in creating your protocol, the general communication mechanics of the gameplay are shown below.

In all examples below, if the client or server receives a message not corresponding to the current game play stage, the message must be discarded.

Game Play

Server: Stage 1 – Allow multiple clients to connect

The server should be started as shown below, where <port> is the port number it will be listening on.

```
./server <port>
```

The server should then display the following message:

```
"Waiting for clients to connect."
```

If a client connects to the server, the client program should somehow convey its wishes to join a boggle game, as well as the human player name running the client. As each client connects, the server should send some sort of a welcome message back to the client as well as print information on the server's screen showing who has connected like below:

```
"Trinity has joined the room."
```

If more than 4 clients attempt to connect, the server should send the client a message indicating the room is full.

After 1 minute is up, the server should check to make sure at least two clients have connected. If not, the server should print a message on its screen indicating that not enough clients are available, send a message to the connected client (if any) that there aren't enough players, and the server should terminate. Otherwise, the server should send a message to each client listing each player name and that the game is about to begin. The server should then transition to Stage 2.

If a client connects while the server is not in stage 1, the server should send a message to the client indicating they need to try again later.

Client: Stage 1 – Connect to the server

The client should be started as shown below, where <server> is the IP address or hostname of the server, <port> is the port number of the server, and <username> is the username of the player (up to 10 characters long).

```
./client <server> <port> <username>
```

The client should send a message to the server containing the player's username and its wishes to join a boggle game. On successful connection, the server should send some sort of welcome message back to the client (see below).

```
"Welcome to the game, Trinity.  Waiting for others to join."
```

After some time is up, the server will send a message either terminating the game(not enough players have joined the room) or else will send a message listing all the connected players with a message that the game is about to begin. After displaying this message, the client should move to Stage 2.

```
"Trinity and Neo have joined the game.  The game is about to begin."
```

Stage 2 – Send out Boggle board information and accept words

The server will create a random Boggle board of 4x4 letters arranged in a cube and send this information out to each client. (Remember, clients cannot necessarily see what is printed on the server screen, so the board information must be sent to each client.) The clients must accept the board information and display it on each client screen in a text square (4 lines of 4 letters). A sample client screen may look like this:

```
E T S M
A S A P
W Z D E
U W H V

Start entering words!
```

Next, the server will allow the clients to send unlimited packets back to the server containing the client's username and word guesses for up to three minutes. After 3 minutes, the server and clients will move on to Stage 3.

Stage 3 – Tabulate the results

In Stage 3, the server will accept no more word guesses from the clients. At this point, the server should send a message to each client indicating that it is tabulating the results. Each client should display that message on their screens.

The server then must eliminate duplicate words from each player's list and score the remaining words according to Boggle rules. The final results, usernames, and words should be sent to each client and displayed on the clients' screens. A sample client screen is below.

```
Neo unique words: paste, mad, sad
Trinity unique words: pasta, pastas, mast, saw, was
Neo points: 4
Trinity points: 7
Trinity wins!
```

The server could be improved by having subroutines to verify the submitted words are in the Boggle board and to check the submitted words against an English dictionary. However, that functionality is not strictly necessary, since each player will see the other players' submitted words and can call a game null and void if one of the players is a cheater.

Finally, after the results have been sent to the clients, the clients should send a goodbye message to the server and terminate. After the last client has said goodbye, the server should also terminate.