Distributed Systems, Spring 2013
Assignment 2: Tic-Tac-Toe Suplemental info

1 Tic-Tac-Toe

The goal of this part of the assignment is to put some of your knowledge together in the construction of a tic-tac-toe client and server.

Tic-tac-toe is a simple two player game played on a board consisting of 9 cells organized in 3×3 grid. One player plays `X`'s and the other `O`'s. Each cell of the grid can be in one of 3 states:

free: the cell does not contain a mark of either player and is either player in their turn can mark it. Initially all cells are initialized as free.

X: the cell has been marked by the `X` player and its state cannot be changed for the remainder of the game.

O: the cell has been marked by the `O` player and its state cannot be changed for the remainder of the game.

The game is played in turns that alternates between the players. The `X` player takes the first turn and then the `O` player takes the next turn. A turn consists of the player picking one of the free cells and marking it with their mark. The game ends immediately when one of three conditions is meet:

X win: Three cells in a line (row, column, or diagonal) contain `X`'s.

O win: Three cells in a line (row, column, or diagonal) contain `O`'s.

Draw: Neither of the above conditions is meet and there are no free cells.

For additional information about the game see `http://en.wikipedia.org/wiki/Tic-tac-toe`.

1.1 The Game

Your task is to construct a client and server that implements tic-tac-toe.

1.1.1 Client

Each player uses a client to connect to the server. Upon connection the server will let the client know if it's player is the X or O player. The client display the board to the user as an ascii grid with each cell assigned a number. A free cell is displayed with its number a marked cell is displayed with the appropriate mark of

the player that marked it (X or O). A prompt is also displayed to the user that is the user's marker followed by the greater than sign and a space. Prior to connecting to the server a '?' is displayed as the user's marker. The following displays the initial view that the X player would seen upon starting the client and connecting to a server along with some initial interactions:

```
?> connect 192.168.1.1:24567
1|2|3
-----
4|5|6
-----
7|8|9
X> 5
1|2|3
-----
4|X|6
-----
7|8|9
X>
O|2|3
-----
4|X|6
-----
7|8|9
X>
```

A user identifies a cell to mark by entering a cell number. The client communicates with the server to validate the action and update the game board if the action is valid (it is the user's turn and the cell is free). The client should display the current state of the game board and any additional information from the server:

- "Connected to <ip:port>: You are X's"
- "Connected to <ip:port>: You are O's"
- "Not able to connect to <ip:port>"
- "Not your turn yet!"
- "Not a valid move!"
- "Game Over: You win"
- "Game Over: You loose"
- "Game Over: Draw"
- "Game Over: Other Side Quit"
- "Game Over: You Quit"

Note a client must also display an updated board when the other player changes the board with a valid move.

The client should support the following commands:

1. **connect <ip:port>** connect to a server. Once connected this command should do nothing. On failure it should display the appropriate message above.
2. **disconnect** disconnect from the server. If not connected this command should do nothing. This command is actually a request to the server to quit and the actual connection should be terminated by the server after it sends a reply back to the clients.
3. **¡enter¿** if the user presses enter/return on its own then if client is connected the current version of the board should be displayed. This should do nothing if the client is not connected.
4. **[0-9]** Mark the appropriate cell.
5. **where** Display the <ip:port> of server that you are connected to. If no connect has been made then display "not connected".
6. **quit** quit client. If a game is in progress this command implies a disconnect first.

1.1.2 Server

The server only needs to support one game at a time. The first client to connect is assigned 'X's and the second O's. The server should correctly arbitrate the game board and send the state of the board to the clients as a reply when it changes. It should also validate client actions and update the clients when the game is over. This of course means that the server must be able to identify winning and draw states.

1.2 Recommendations

I suggest that you meet immediately to settle on your game protocol and jointly write your messaging code that encodes your protocol. After this you should be able to split up the work.

2 Evaluation

Bonus grades will be give if you handle errors such as arbitrary termination of the clients or the server.

I will test your client and server to see that they work and conform to the requested function. Don't forget to document any meetings in which you discuss what you learnt from your readings.