Author: R. Bowie Smith
Class: CMSC 481 - Computer Networks
Assignment: Tic-Tac-Toe Project 2 (UDP)
Professor: Edward Zieglar
File: ttt_protocol_specs
Date: 2018-12-05
Description: This file documents the protocol used to exchange messages between the client and server in our UDP-based game of tic-tac-toe.

Overview

The server holds the state of the game. The state of a game of tic-tac-toe is completely described by the state of each of the 9 squares of a tic-tac-toe board. Thus the server holds a python list of length 9, where each element of the list may be: 'X', 'O', or 'u'.

Server moves are represented by 'O'. Client moves are represented by 'X'. Unplayed squares are represented by 'u'. Thus the state of the game starts as: ['u','u','u','u','u','u','u','u','u'], and progresses until the game state represents a winner (3 in a row) or a cat game (game board full with no winners).

The first move of the game is determined by arguments passed to tttc.py when invoked. If '-c' is passed as an argument, the client (user) makes the first move. If no '-c' argument is passed, then the server makes the first move. The server has an "AI" that randomly selects one of the remaining unplayed moves to play.

In order to isolate game states from one another and allow concurrent games, the server must keep a record of all game states that are currently in play, and delete game states after games are finished (to make room for more games and avoid future conflicts). According to the protocol, the client does not send identification information with each message, so the record of game states on the server must use the clients IP address and port to map game states to game instances.

Because the game protocol will be implemented over UDP, it is possible that messages will be lost in transit. To overcome this issue, the client is required to retransmit messages to the server after a set timeout value. The recommended timeout value is 1 second, and the client should abort the game after some maximum number of attempts.

Messages

Client

The client only sends single character messages to the server. There are 10 possible messages the client can send: 's', '0', '1', '2', '3', '4', '5', '6', '7', and '8'

`'s'` The client will send the character 's' to the server if the user did not include the '-c' argument when invoking the client program. This can only ever be sent as the *first message.* This indicates to the server that it should make the first move and reply with the state of the game.

`['0'-'8']` Other than 's', the client can send any of the 9 digits: '0', '1', '2', '3', '4', '5', '6', '7', '8'. These digits indicate what move the client would like to play next. Since the game state is stored on the server, the server is in charge of taking this move (which is an index on the tic-tac-toe board) and updating the game state appropriately. This is always sent in response to the server replying with the game state, except for the case when the client makes the first move, in which case the client will send one of these integers as the first message to the server. After updating the game state with the clients move, the server should should make it's move (if there are moves left to play) and respond with the updated game state.

Server

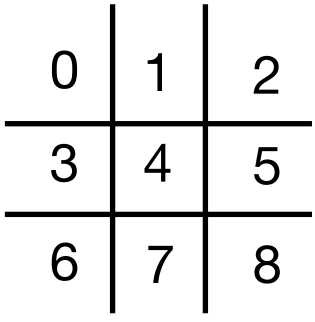
The server only sends 11 character messages back to the client. There are 4 different formats for the 11 character message the server sends back to the client.

`'p_CCCCCCCCC'`	Continue Play
`'w_CCCCCCCCC'`	Client Won
`'l_CCCCCCCCC'`	Client Lost
`'c_CCCCCCCCC'`	Cat Game

The first character in the response message indicates the state of the game to the client. `'p'` indicates that the client should continue play. That is, no winner has been determined and the game board is not full, so the server is expecting another request from the client. `'w'`, `'l'`, and `'c'` all indicate to the client that the game is over. `'w'` indicates the client won; `'l'` indicates the client lost; and `'c'` indicates a cat game (no winner).

In all 4 formats, the 9 character `'CCCCCCCCC'` string indicates the state of the game. Each of the `'C'` characters can be ones of three letters: `'X'` to represent the clients move, `'O'` to represent the servers move, and `'u'` to represent an unplayed square. The indices of the string (0 through 8) represent the 9 squares on the tic-tac-toe board, in order from left to right, top to bottom. An example of the message format is displayed below. The client responds to this with the index of the next move they would like to play if the server sends the `'p'` continue play format, or sends no response if the game is over. If the server sends the `'w'`, `'l'`, or `'c'` response, the game is over, and the server should clean up any memory used to manage the state of the game.

Tic-Tac-Toe Position Indices in gameState continue play message:
`'p_012345678'`



gameState character string for example game below:
`'p_XOuuXuuuO'`