Tic-Tac-Toe Client-Server Application over UDP

Protocol Specification

Author: Ressa Reneth Sareral
Email: saressa1@umbc.edu
Date Created: December 05, 2017

Addressed to: Dr. Edward Zieglar
Course Number: CMSC 481
Course Description: Networks

INTRODUCTION

The purpose of this document is to describe the messages that are sent between the client and the server so that others will be able to develop their own client or server that can interact with this the application.

UDP sockets were utilized to create the Tic-Tac-Toe client-server applications so that the client and the server can communicate with each other. The server is capable of accepting multiple simultaneous clients.

SERVER OPERATION

The server application executable file depends on the "tictactoe_support.py" file so both must be within the same directory to successfully operate the server application.

To operate the server application, the following command must be run in the directory that the "tictactoe_server.py" file is in:

```
python3 tictactoe_server.py
```

The server will then be listening for clients requesting to play tic-tac-toe. The request messages will be described in the Acceptable Message Formats section.

CLIENT OPERATION

The client application executable file depends on the "tictactoe_support.py" file so both must be within the same directory to successfully operate the client application.

To operate the client application, the following base command must be entered into the command line:

```
python3 tictactoe_client.py
```

Additional command line options include "-c" and "-s <serverIP >". Their functionality will be described below:

`-c`	Allows the client-side user to make the first move; otherwise, the server will make the first move.
`-s <serverIP>`	Specifies the IP address of the server that the messages should be sent to (required)

ACCEPTABLE MESSAGE FORMATS

The client and the server can only accept a finite set of messages. The types of messages are dictated by specific labels transmitted to each other. Since this is a Tic-Tac-Toe application, all messages with no labels are assumed to be in a specific Tic-Tac-Toe board format.

The client and the server do not accept the same types of messages, which are in the form of capitalized strings. A list of messages respective to whether the client is receiving or the server is receiving them is shown below:

If the client socket is the receiving end, the following messages dictate the following:

CLIENT	The client has won.
SERVER	The server has won.
DRAW	A draw has occurred.
CHEATER	An attempt to cheat was detected.

If the server socket is the receiving end, the following messages dictate the following:

CLIENT_FIRST	The client is requesting that the client-user makes the first move.
SERVER_FIRST	The client did not request to make the first move so the server should occupy the board first.
QUITTING	The client has terminated the program using a keyboard interrupt (ctrl-c) and has abandoned the game.

If none of the messages shown and described above are sent by either the client or the server, and a message is received, it is assumed that the received message is in Tic-Tac-Toe board format. Tic-Tac-Toe boards are two-dimensional lists where the first dimension represents rows and the second dimension represents columns.
There are only three rows and three columns in the board.

In python, an empty board appears in the following format:

```
[ [0,0,0] , [0,0,0] , [0,0,0] ]
```

The board is updated by providing pairs of integers between 1 and 3 on the client-user end.

Several users may interact with the Tic-Tac-Toe server at a single time, and quitting and initializing new games is permitted.

AFTERNOTE

The Tic-Tac-Toe client-server applications are in early stages of development. Further design additions including timeouts and server detection prior to connection would be possible to implement; however, additional development time would be required. The Python socket library is vast and allows for far more additional functionality.

Also, additional messages could be included to provide a far more robust Tic-Tac-Toe experience. For example, additional checking for if the Tic-Tac-Toe boards received are valid would be helpful, but as of now, the system does not check if the Tic-Tac-Toe boards are of a 3x3 cell list format. These are implementations for the future iterations of the Tic-Tac-Toe client-server application.

The current Tic-Tac-Toe client-server applications are overall true to a UDP design. No real checking is involved regarding whether the server is listening or not, and the server simply stores the received data in buffers that potentially may fill if too many client applications are playing the game.

If one would desire to design a client or server that would be able to interact with the current design, the python code would be overall simple. After initializing sockets, sending the messages listed in the Acceptable Message Formats to their respective client-server ends will be successful.