Distributed Tic-Tac-Toe Assignment Description

You will design and build a distributed two-player tic-tac-toe game where the players are located on two different hosts and use explicit message passing to communicate with a server (tic-tac-toe server).

The basic play of the game will be as follows:

A server is uniquely identified by a host:port combination. The server could be run on any machine with valid IP address and you can use `ipconfig` command to check IP address. The server will be waiting for connections from players at two different port numbers.

Assume that your server program name is TTTserver and client is TTTclient. To test your application, you can put these two players and server on the same local machine, i.e. both server and clients will be run on localhost using IP address 127.0.0.1 and we will use port number 7001 and 7002 for each client. First, we will start up TTTserver using command line `java TTTserver` . Then player 1 will connect to the TTTserver using command line `java TTTclient 7001`. And player 2 will connect to the TTTserver using command line `java TTTclient 7002`.

From one of the games, e.g., Player 1, enter the port of the TTSserver machine, and request that a game begin with Player 1 being X's. The player playing X's will make the first move. Play will alternate until one player wins, or a tie is reached and the board is full (a tie cannot be declared until all 9 squares are filled). To start over, the user should be required to clear the game, and then one of the players can start over by making a new request to start a new game.

User interface:

This is not a user interface class. You do not need to build a fancy graphical user interface. A text-based only interface is a '**mandatory**'. The only requirement is that your interface (i.e. text-based instructions / commands) must be clear. Remember, i have to test all of your codes line by lines, and if i get confused, that may not be good for your grade. Don't forget to properly comments in your coding.

Submission (via elearning.utm.my):

You need to submit following items:

1. all source code files (*.java)
2. user manual (how to use your program)

Due date:

15 June 2021 (Tuesday)

- Any late submissions deserve to have the point deducted.

The following part defines the communication protocol for the distributed tic-tac-toe project.

Overview:

Once a connection is established between the client and the server, all communication is initiated by the server. Client behavior is defined by the following state diagram:

$$\begin{tikzcd}
\text{sta} \arrow[dr, "\text{recv 0}"'] & & 2 \arrow[d, shift left, "\text{send 2R}"] & & 5 \arrow[d, shift left, "\text{send 4}"] & \\
& 0 \arrow[r, "\text{send 0}"] & 1 \arrow[u, shift left, "\text{recv 1W}"] \arrow[r, "\text{recv 1R}"] & 3 \arrow[r, "\text{send 2R}"] & 4 \arrow[u, shift left, "\text{recv 3}"] \arrow[r, "\text{recv 3'}"] & \text{sto}
\end{tikzcd}$$

The transitions in this diagram indicated messages sent or received by the client. The numeric codes match the message type codes defined below. In the cases of message types 1 and 2, the message body is shown here. The message body is omitted for message types 3 and 4. The final message, indicated here as 3', is a type 3 message in which the game status code is not 'P'.

Client states are as follows:
0) Client must respond to accept connection.
1) Client is waiting to play.
2) Client must determine if it will continue waiting.
3) Client must determine if it will commit to playing the game.
4) Client is waiting to be told if it should make a move or if the game is over.
5) Client must determine its next move.

The client can choose to quit prematurely by sending the appropriate message while in state 2, 3 or 5. This action will take the client immediately to the stop state.

Message Types:

code	direction	meaning	comment
0	server to client client to server	connection accepted	sent once when connection is opened
1	server to client	status update	sent periodically while in server waiting state
2	client to server	status response	sent in response to message 1
3	server to client	game state	sent periodically while in game active state
4	client to server	game move	sent in response to message 3

Detailed Message Descriptions:

All messages are character strings. The first character is the message type code. The remaining characters are the message body, which must be interpreted according to the message type.

message 0: connection accepted

description: This message is sent from the server to the client when the server accepts a connection from the client.

body of message: Empty.

response: Client should respond with the same message.

message 1: status update

description: This message is sent periodically from the server to the client while before the game begins. Zero or more 'W' messages may be sent. Exactly one 'R' message will be sent, which indicates that the game is beginning and that subsequent messages will be game state messages (message 3).

body of message: One character status code:

'W'	= Waiting for second player.
'R'	= Ready to play.

response: The client must respond with message 2.

message 2: status response

description: The message is sent from the client to the server in response to message 1.

body of message: One character status code.

'R'	= I will continue waiting or I am ready to play.
'Q'	= I quit.

message 3: game state

description: This message is sent periodically from server to client while the game is in progress.

body of message: The current board and status information. 11 characters in all.

characters 0-8 : The current game consisting of 'X', 'O' or space stored in row-major order.
character 9 : Game status:

'X'	= X has won.
'O'	= O has won.
'D'	= Game is a draw.
'P'	= Game in progress.

character 10: Error code.

'0'	= No error.
'1'	= Last move was invalid.
'2'	= Game forfeited by other player.

response: If game status is 'P', the client must respond with message 4, otherwise the game is over, and no response is expected.

message 4: game move

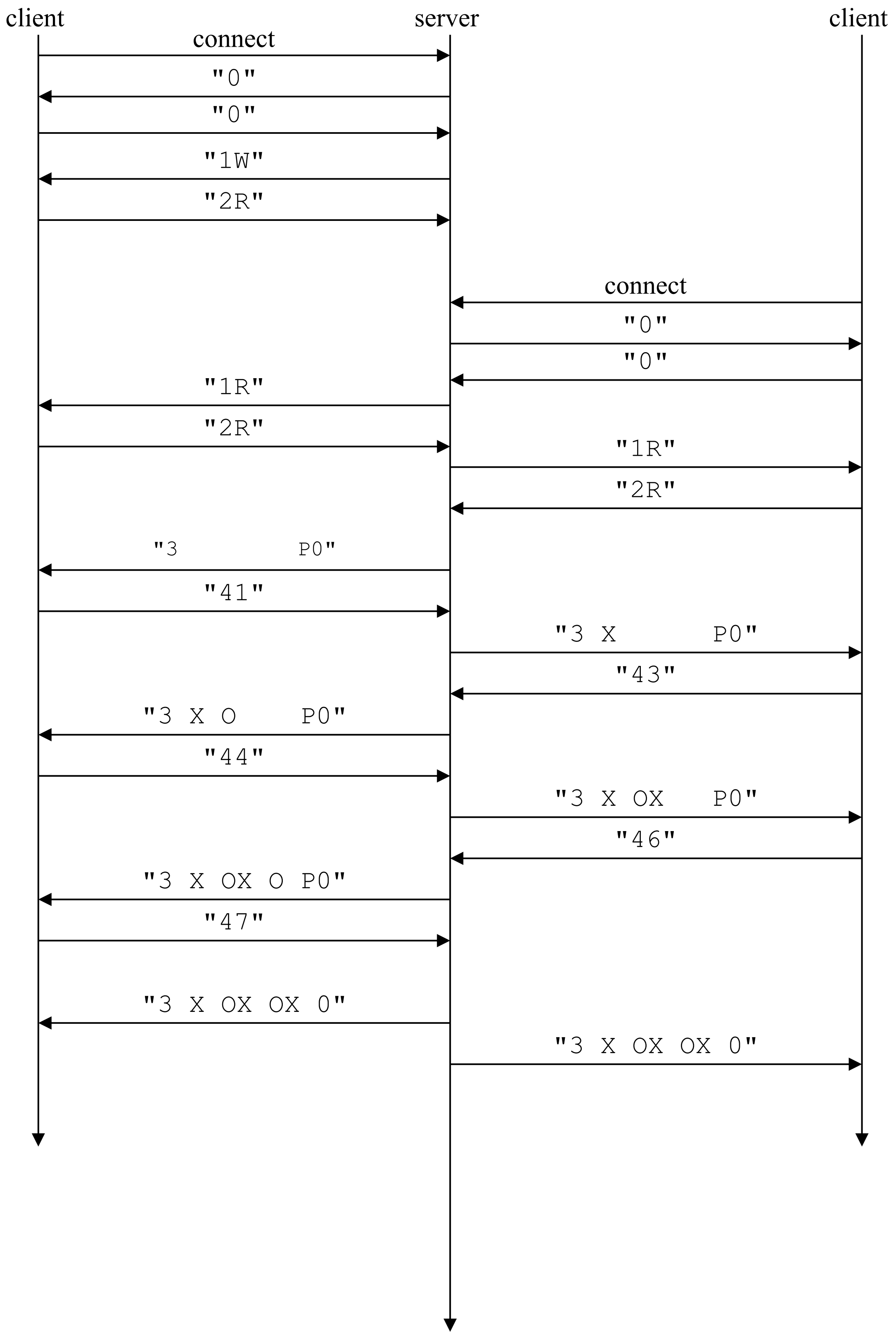
description: Set from the client to the server in response to message 3.

body of message: One character response.

'q' = I quit and forfeit the game.
'0' – '8' = A move, indicating a board position as defined by the following diagram:

'0'	'1'	'2'
'3'	'4'	'5'
'6'	'7'	'8'

Sample Game:


client
server
client
connect
"0"
"0"
"1W"
"2R"
connect
"0"
"0"
"1R"
"2R"
"1R"
"2R"
"3   P0"
"41"
"3 X   P0"
"43"
"3 X O   P0"
"44"
"3 X OX   P0"
"46"
"3 X OX O P0"
"47"
"3 X OX OX 0"
"3 X OX OX 0"