

TIC TAC TOE Game Using RPC-XML

Server – Client Architecture Implementation

SWE 545 Distributed Systems Programming

Term Project

**Name :** İlker YILMAZ

**Student ID :** 201571915

**Introduction**

XML-RPC is a remote procedure call (RPC) protocol which uses XML to encode its calls and HTTP as a transport mechanism. XML-RPC works by sending an HTTP request to a server implementing the protocol. The client in that case is typically software wanting to call a single method of a remote system. Multiple input parameters can be passed to the remote method, one return value is returned. The parameter types allow nesting of parameters into maps and lists, thus larger structures can be transported. Therefore, XML-RPC can be used to transport objects or structures both as input and as output parameters.

The goal of this project, is to implement a “tic tac toe” game using Python’s SimpleXMLRPCServer (server-side) and xmlrpclib (client-side) modules with any “tic tac toe engine from internet and thoroughly comprehend the ideas that we’ve learnt through this course.

In this respect I’ve used a simple tic tac toe engine from github <https://github.com/cklone/tictactoe-python> and I’ve modified it to fit in my purposes. The complete project files can be found in my github folder at <https://github.com/iy536/SWE545termproject>.

### Internals and API

server = **xmlrpclib.ServerProxy**(**"http://localhost:8000"**)

Establish a connection to game server

clientId server.**create\_game()**

Creates a new game and returns client unique identifier clientId

server.**start\_game(**clientId**)**

Starts created game with associated clientId, when **print** **start\_game(**clientId**)** is called, the welcome banner of the game is displayed on client screen

server.**print\_board(**clientId**)**

Displays the current state of the tic tac toe board.

e.g

|O |X |X| |1 |2 |3 |

| |O |X| |4 |5 |6 |

|O |X |O| |7 |8 |9 |

server.**make\_move(**clientId,reply**)**

Sends the user input which is given by **reply** argument to tic tac toe server. Server receives given input from client with this command.

server.**move\_valid(**clientId**,**reply**)**

Checks the user input is valid or not. If the tic tac toe board size is 3, the valid input is between 1-9. Also, If the cell already is filled, it returns false.

server.**print\_result(**clientId**)**

After the game is finished, the command must be called to display the result of the game.

* If the user wins, displays "Winner" banner
* If the computer wins, displays "Loser" banner
* If the game ties, displays "Game Tied"

server.**end\_game(**clientId**)**

Ends the game. When this command is called, server kills the thread which is attached to this client

### Server – Client Sequence Diagram

### The following diagram shows message flow between client and server.

### 

### Figure 1 – Sequence Diagram

### Server is able to serve multiple clients simultaneously. To achieve this, when each game() object is being created, a unique clientId will be assigned to this object and a thread is being created for each clientId. Every clientId is mapped to its own thread, so server can serve multiple clients using this unique clientId. Whenever a function called by client, server matches its clientId with the corresponding thread and complete function calls.

### 

### Figure 2 – Mapped Threads

**Conclusion**

### Project serves its purpose well. We applied the concepts that we learnt in the class to a real world example. We gathered uselful ınformation about RPC-XML and server and client architecture. This valuable knowledge will be basis for the future projects and will lead us to solve problems ahead.