Network 3 Midterm Project

Head

Dr. Peyman Pahlevani

Teaching Assistant

Kiana Ghassabi

Written By

Solmaz Mohamadi

Sepide Bahrami

IASBS – Summer 2020

Institute for Advanced Studies
in Basic Sciences, IASBS

In this project, we have used Python programming language to implement the Networked version of Tic Tac Toe game. There are two .py codes available in our project, one for server and another for client. We have debugged and tested these codes by running them in command prompt.

Running the Codes

To start the game, you should go to the folder of codes, open three command prompt pages and type these commands respectively in each one of them:

- python server.py – python client.py – python client.py

By running the commands, the tkinter GUI will come up for server and both clients.

Initializing (both in server and client side)

The server and client sockets will be established the moment you run the command using this line of code in function _init_() for server.py and client.py: **socket.socket (socket.AF_INET, socket.SOCK_STREAM)** in which the AF_INET means that we are using IPv4 and the SOCK_STREAM indicates the use of TCP in our connections.

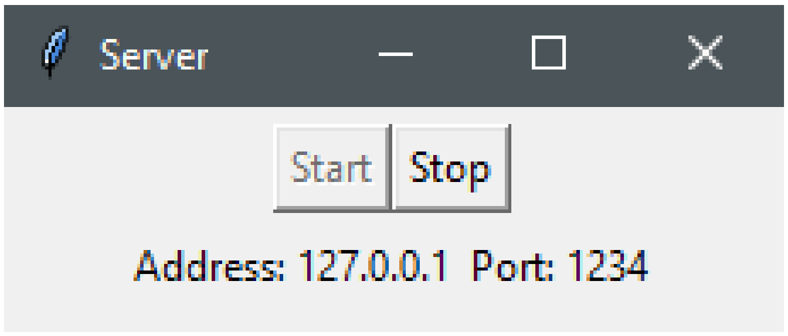
Message Format (both in server and client side)

To establish a same format of messaging in both sides, we add some specific headers into messages to be sent. Then on the other side, this header is extracted from the main message. The buffer size we have used in this project is 1024 bytes since we send short messages!

Server.py Functions

Start the Sever

Before anything, the game server should be started. You have to click on the start button, which has appeared on server GUI. By clicking the button, the function start_server() in server.py will be called and the server socket will be bind to host address: 127.0.0.1 and host port: 1234. The server can listen to maximum 4 users as we set the listen() parameter. Then a new thread called accept_clients() starts to accept new clients continuously until reaching the server's capacity.



Accept Clients

After this thread started in start_sever(), the server will respond to any new connection request from any client. When a new client connects, server gets the client information like its username and address and saves them into its clients' vector for any further use. Moreover, the server sends a welcome message to the new client. Then, another thread called show_online_users() will be started which is explained in the next part.

Show Online Users

After joining each client, the list of clients must be updated. However, when the number of online clients become more than 1, then a new list of currently online users will be sent to all of the clients from clients' vector.

Forwarding the Game Request

After receiving the format **'game_req # opponent_name # dimension'**, server sends the game request to the opponent name in the format below:
'offer_from # name_of_the_request_sender # dimension' and then it waits for the client's response to decide about starting the game or not.

Game Initialize and Ending

After the game request is accepted from the other client, server is informed and lets the game start. The game ends when one of the players score at least one point. The winner player stops the game process, sends the result to the server, and server announces the game results to both players. If none of them succeeds to win, game will stop when there is not any empty cell, and the result is draw.

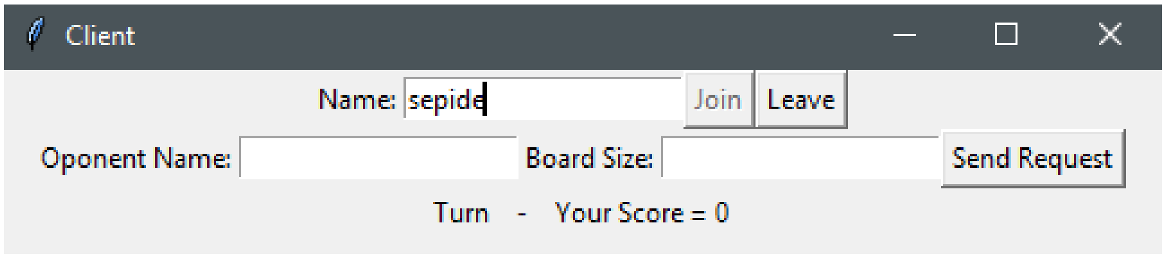
Stop the Sever

By clicking the stop button, the server checks for any client existence. If there is any, it sends a message to that client which indicates the server has stopped. Then it closes its GUI and socket. On the other side, when the clients gets this message it closes its GUI and socket too.

Clinet.py Functions

Join the Sever

After starting the server, you should type two client names in their specific fields and by clicking the join button, the client socket will connect to host address and port of the sever socket. Then a new thread which is called receive_message_from_server() starts to listen to the server and gets its messages continuously for rest of the connection.



Name: sepide

Turn - Your Score = 0

Sending Game Request

A client can send a game request after joining into the server. In addition, the number of online clients must be more than one. Therefore, after seeing the online users list provided by server, he/she chooses a name from the list and type it into the opponent name entry along with the desired board dimension. So on the server side the received message format would be like:
'game_req # opponent_name # dimension'

Accepting Game Request

The server forwards the game request to the desired opponent of a client. When the opponent gets this message, it answers to server in either 'accepted_offer' or 'declined_offer' message format to inform his/her decision about the incoming game request.

Choosing Squares

After both clients and server set everything, it is time to start the game. Meanwhile each client sends the desired coordinates to be chosen to server and the server forwards the coordinates to the other client, so the game movements are done in both sides' GUI.

Leave the Sever

Any client can leave the server anytime. To make this happen, the client sends a message with header **'exit'** to the server and closes its window and socket. On the server side, the list of clients will be updated after removing the current user.

Citation

As we were not familiar with socket programming, we reviewed and used lots of source codes and videos online. Here are some useful ones that helped us a lot:

1. https://www.geeksforgeeks.org/socket-programming-python/#:~:text=Socket%20programming%20is%20a%20way,reaches%20out%20to%20the%20server.
2. https://realpython.com/python-sockets/
3. https://levelup.gitconnected.com/program-a-networked-tic-tac-toe-game-in-python-30f8826e591d
4. http://programarcadegames.com/index.php?chapter=array_backed_grids
5. https://www.geeksforgeeks.org/python-implementation-automatic-tic-tac-toe-game-using-random-number/