DNP project report

authors:

Ivan Isakov, Andrey Torgashinov, Darya Tolmeneva SD-02
link to the repo: https://github.com/TheVex/Multiplayer-Tic-Tac-Toe/tree/main

Introduction

Our task was to implement **project 1 of variant A**, which consisted in creating a multiplayer game server with Python sockets. We have implemented a multiplayer game server using **TCP sockets**. The game is a turn-based tic-tac-toe game between two players. The server manages the states of the players and ensures consistent synchronization.

This project is relevant to us because we love to play tic-tac-toe among ourselves, and so we were able to try to implement our own game and practice the skills we learned in the Distributed network systems (DNP) course. Namely, the creation of **TCP sockets** and **multithreading**

Methods

1. **System architecture**
   The server is implemented in Python using the `socket` module for TCP connections and `threading` to handle multiple clients.

   Client: The client is the game on the player's computer, showing the board and letting them click to make moves. It talks to the server over the internet, sending messages about the player's actions and receiving updates about the game. It uses menus to let players find and join games.

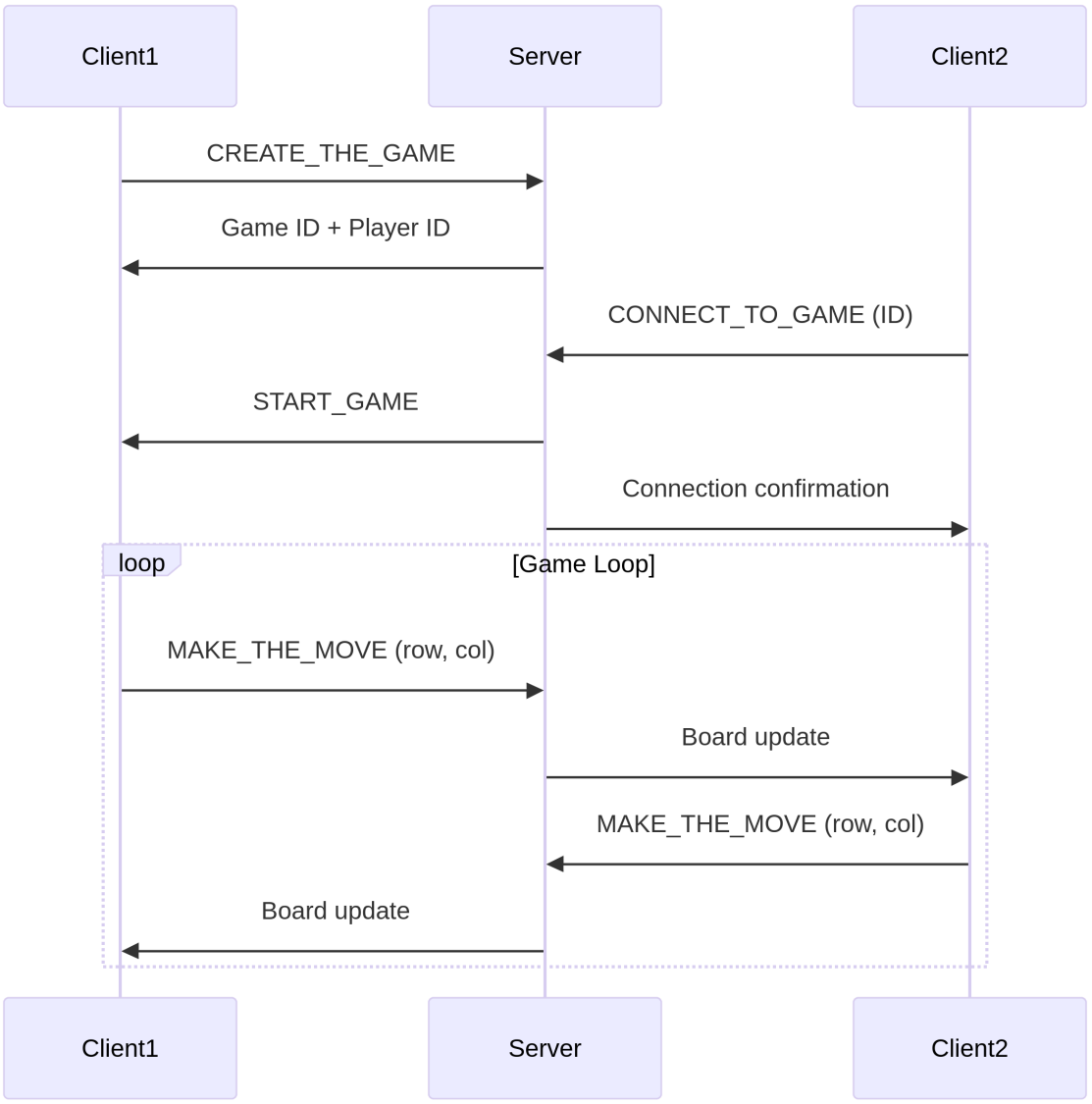
   Server: The server is the central computer that runs the game logic. It listens for players connecting, creates new games, keeps track of the game state (like who played where), and decides if a move is valid. It also figures out when
   the game ends and tells the clients the result. It can handle many games at once.

   Expected flow of requests and responses

   - The player opens the game on their computer.
   - They might see a list of available games from the server or create a new one.
   - When a player makes a move, their computer sends a message to the server.
   - The server checks if the move is okay and updates the game.
   - The server then tells both players about the updated board.

- This continues until someone wins or there's a draw, and the server sends a message about the game's end.

Interaction Workflow:



2. **Communication Protocol**
   Message Format: JSON with `type` (request/response) and `data` fields.
   Key Requests:
   - `CREATE_THE_GAME` : Create a new room.
   - `CONNECT_TO_GAME` : Join an existing game.
   - `MAKE_THE_MOVE` : Submit a move (row, column).
   - `RECONNECT_CLIENT` : Reconnect after dropout.
3. **Error Handling**
   - **Timeout**: Waits for reconnection before ending the game.
   - **Notifications**: Sends `PLAYER_DISCONNECTED` to the opponent if a player drops.
4. **File-by-File Code Analysis**

   - **protocols/enums.py**: This file sets up the basic language for the game, defining things like what marks players can use (X or O), how someone can win, and the

types of messages the client and server send to each other.

- **log_meth/log.py**: This file contains simple tools for the server to keep a record of what's happening, like when players connect or if there are any errors.
- **server.py**: This file holds the main brain of the game, the server. It manages the connections of players, keeps track of different games, decides whose turn it is, and checks if anyone has won. It uses Python tools to handle network connections and can manage multiple games at the same time.
- **client.py**: This file is what the player sees and interacts with. It uses a Python game library to show the Tic-Tac-Toe board and let players make moves. It also includes menus for joining or creating games.

Multiplayer-Tic-Tac-Toe-main/
├── **log_meth/**
│ └── log.py
├── **protocols/**
│ └── enums.py
├── diagram.png
├── client.py
├── README.md
├── Report.md
├── server.py
└── requirements.txt

Results

You can find screenshots and demo videos here:
https://disk.yandex.ru/d/MH1hmEJZ4H5tQg

Discussion

Achieving a result

The goals of this task might be considered as successfully achieved. The server should be able to handle multiple games at once and support up to 16 players concurrently. Players will see a menu to start, then a list of games with possibility to create their own game, and finally the Tic-Tac-Toe board where they can play. The game will draw corresponding symbols when a player makes a move and will announce the winner or a draw when the game ends. If a player disconnects, the server will wait some time and grant autowin to the other connected player. The client and server will exchange messages in a structured way using a JSON format.

Challenges

We faced such difficulties in the project:

- tracking the correct serialization of transmitted data between the client and the server throughout the code.
- rendering the board on the screen using pygame tools (for example, the screen did not always display what was happening on the board in time, or the marking of marks on the board was different from the real one due to the different perception of the coordinate systems of the matrix (X x Y) and the pygame canvas (Y x X))

Potential improvements and optimizations

The game should also have clearer error messages for players. The server can be improved to support more than 16 players at the same time. Adding more security to the server would help prevent cheating. The game lobby could be improved with more features for players. The ability to reconnect to the session should be added.

References

To implement the project, we used lectures on DNP, as well as the pygame documentation and pygame-menu documentation