VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND TECHNOLOGY

COMPUTER NETWORKING

EXTENDED COURSE (CO309B)

Rock, Paper, Scissors

Lecturer: Thầy Hoàng Lê Hải Thanh
Students: Đinh Xuân Quyết - 2212854
Đỗ Hoàng Quân - 2212779
Lê Ngọc Vinh - 2213964

Class: TN01

HO CHI MINH CITY, JANUARY 2025

Tasks Assignment

No.	Full Name	Student ID	Assigned Task
1	Lê Ngọc Vinh	2213964	Code + Report
2	Đinh Xuân Quyết	2212854	Code + Report
3	Đỗ Hoàng Quân	2212779	Code + Report

Contents

1 Introduction

Rock-Paper-Scissors is a simple yet engaging game enjoyed by people worldwide. The objective of this project is to create an online version of the game, leveraging a network-based client-server architecture to enable real-time interaction between players. This implementation highlights the principles of network programming, socket communication, and game design in a distributed environment. The goal of the group is to apply the knowledge gained from the Computer Networking course and create a fun, accessible game and the Rock-Paper-Scissors game is well-suited for achieving this goal.

1.1 The game Rock-Paper-Scissors

Game Rules:

- Players: The game is played by 2 players
- Gameplay: The rules follow the traditional Rock-Paper-Scissors game:
  - Rock beats Scissors
  - Scissors beat Paper
  - Paper beats Rock
- Scoring:
  - A score counter keeps track of each player's wins
  - The first player to score 5 points wins the round, and the scores are reset for a new round
- Winning Condition: The game continues until one player consistently achieves the target score in multiple rounds, or the players decide to stop

1.2 Network Architecture

The game application uses a client-server architecture to implement the Rock-Paper-Scissors game. The client-server architecture is a model in which tasks or services are divided between two main components: the server and the client. The server is a centralized system that provides resources, services, or data to multiple clients. It is responsible for managing requests, processing data, and ensuring consistent communication

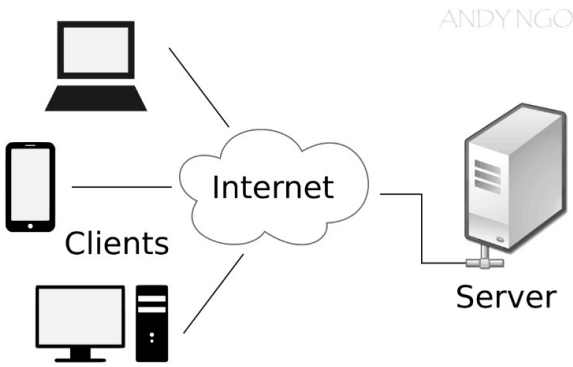

Figure 1: Client-server Architecture

In this architecture, the server acts as the central hub, managing client connections, pairing players for matches, and processing the game logic. It is responsible for facilitating communication between players. Each player interacts with the game through a client application, which sends the player's choice (Rock, Paper, or Scissors) to the server and receives the results

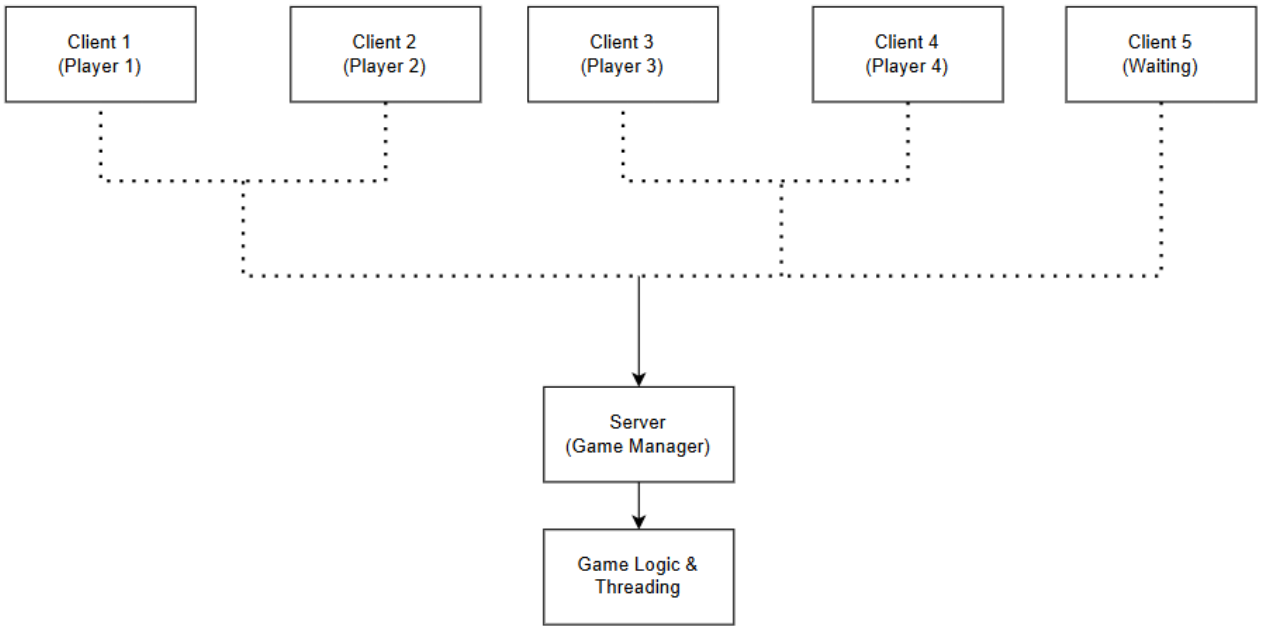

Figure 2: Network Architecture Diagram

How the Game Works:

- The Rock-Paper-Scissors game uses a client-server architecture, so it requires a server to run continuously throughout the duration of the game matches
- Pairing Mechanism:

  - The server increments *id* with each new connection, and uses this to calculate the *game id* for pairing players
  - Odd-numbered players (1st, 3rd, 5th, etc.) start a new game, while even-numbered players (2nd, 4th, 6th, etc.) are paired with the previous player
  - The server creates a new game instance for each pair
  - Each player is handled in a separate thread to allow for concurrent gameplay

- After the server creates a game room, players can join the room and connect directly with each other to exchange game data.

2 Main Functions of Server and Client

- **Server** mainly organizes and controls the games, matchmaking phase and connections between Server and Clients.
- **Client** can create connection through socket to Server, then being automatically paired with an another client, or wait until an other client is available. After being paired, the game starts between 2 clients.

2.1 Server

The server creates and controls the games between clients. Here are the server's main functions:

- Setup connection: When a client tries to connect to the server through server's IP, using a socket, the server accepts.
- Create a game: If there is currently even number of clients playing, and a new client wants to play, the server creates a new game and lets the client wait.
- Matchmaking: The server also has the role of a matchmaker. If there is currently odd number of clients playing, and a new client comes in, the new player is paired with the waiting player, and 2 players starts playing.
- Control the game: The server is responsible for control the flow, including receiving and sending data between 2 clients, calculate the score and result, as well as updating the game's status in real-time.
- Remove a game: When a client in the game exits (disconnects), the server lets the other client out of the game and stay in the status as when the client just connected to the server (the screen with "Click to Play" line). If this player exits as well, the server removes the game.

2.2 Client

- Find a game: The client connects to the server through server's IP address, and a port, then click "Click to Play" to start finding a game.
- Play the game: While in the game, the client is able to interact with all features of the game, playing in real-time with other clients.
- Exit the game: The client can click a button to disconnect from the game.

3 Flow Control

3.1 Flow Control of Server

1. First, the server is started.
2. When a client connect to the server through the server's IP address and a port (for example 5555), the server accepts the connection.
3. If there is currently even number of clients playing, and the new client click "Click to Play" line, the server creates a new game, and lets the new client wait.

   On the other hand, if there is currently odd number of clients playing, and the new client click "Click to Play" line, the new player is paired with the waiting player, and 2 players starts playing.
4. When a client in the game exits (disconnects), the server lets the other client out of the game and stay in the status as when the client just connected to the server (the screen with "Click to Play" line). If this player exits as well, the server removes the game.

3.2 Flow Control of Client

- First, the client connects to the server through server's IP address, and a port (for example 5555).
- When the connection is accepted by the server, the client reaches a screen with "Click to Play" line. The client can click this to get into the matchmaking phase, or click "Exit" box below to disconnect from the game.
- After clicking "Click to Play", the client waits until there's an another available player, then this client can start playing.
- The client can click "X" button at the top-right corner to disconnect from the game. If the oher player exit, this client will be brought back to the screen with "Click to Play" line, then back in step 2.

4 Implementation

We use the Python programming language in order to implement the program. We primarily use the 2 libraries: socket to support networking and pygame to implement the game and the UI.

4.1 Game client

It is implemented in the module client.py. This module creates the graphical user interface (GUI) using the pygame library. It displays the game screen, including the game buttons for the player's move selection and updates based on the game's current state. The client interacts with the server to send and receive game data, such as the player's chosen move and the overall game state.

4.2 Server

The server, implemented in the module game.py, is responsible for managing player connections and running multiple game sessions. It listens for incoming connections, assigns players to games, and handles communication between clients. The server ensures that the games are synchronized, maintaining the game state and determining the outcome of each round. The server can manage multiple games simultaneously, each identified by a unique gameId.

4.3 Game Logic

The module game.py encapsulates the game's core mechanics. It tracks the state of each game, including the players' moves, readiness, and the results of each round. It determines the winner based on the players' moves and handles resetting the game state after each round.

4.4 Networking

The nodule network.py is responsible for managing communication between the client and the server. It establishes the connection to the server and provides methods for sending and receiving data. This module serializes and deserializes the game state to synchronize the client and server during gameplay.

5 Conclusions

5.1 Results

We successfully implemented a simple game which is capable of allowing devices within the same LAN network to join and play together.

Here are the screenshot images of our game:

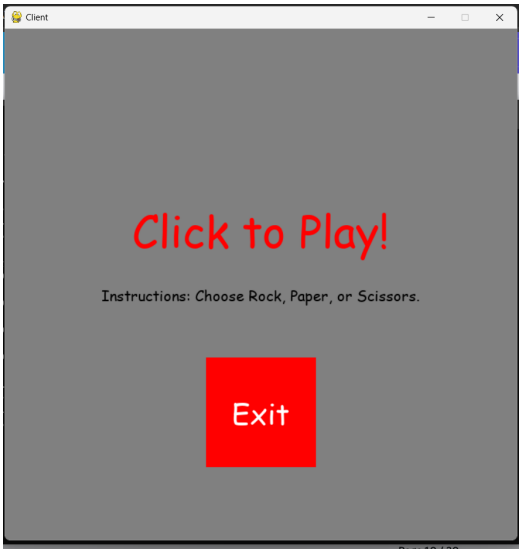

Figure 3: "Click to Play"

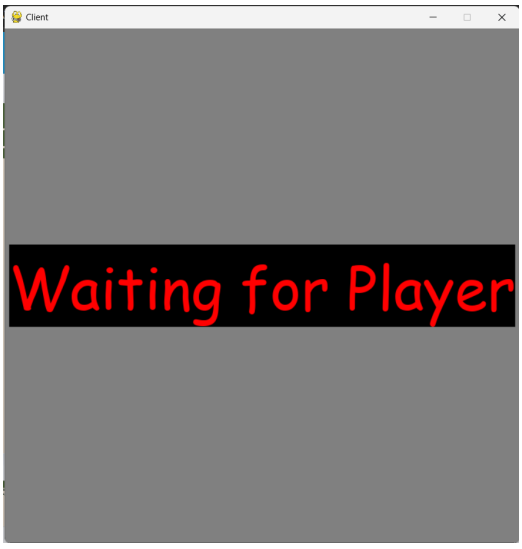

Figure 4: Waiting Screen

Figure 5: Game Screen

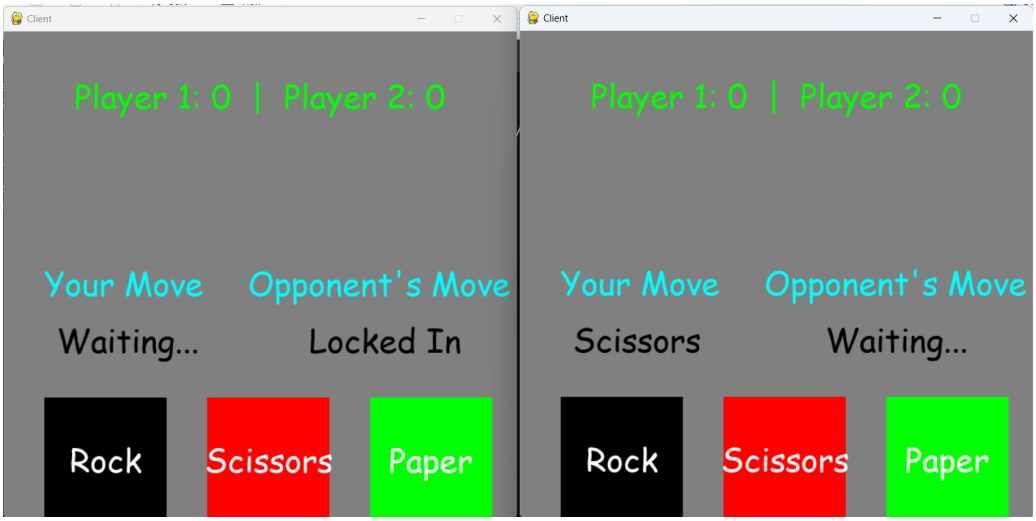

Figure 6: 2 players playing the game

Image before a player score their 5th win:

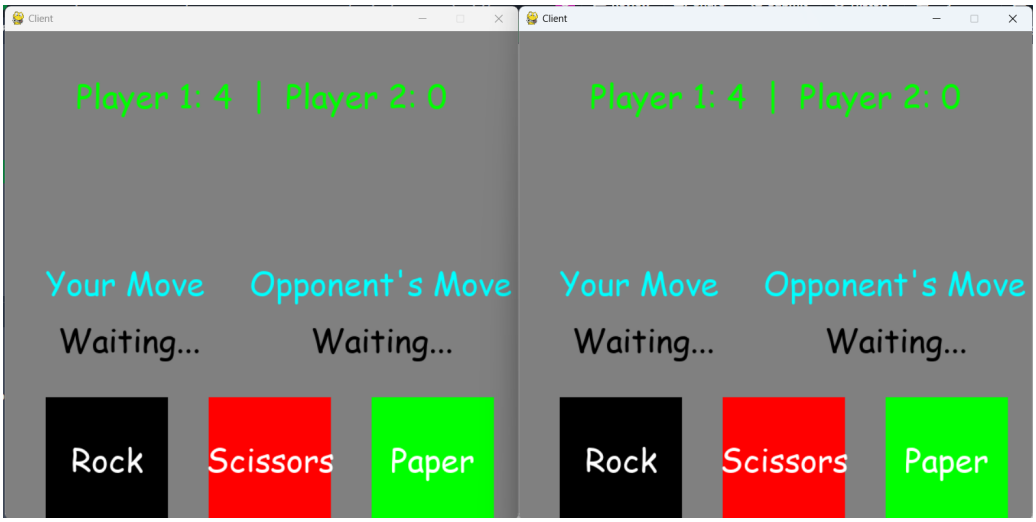

Figure 7: Before player 1 scores their 5th win

After their 5th win, the system will inform that they won the series:

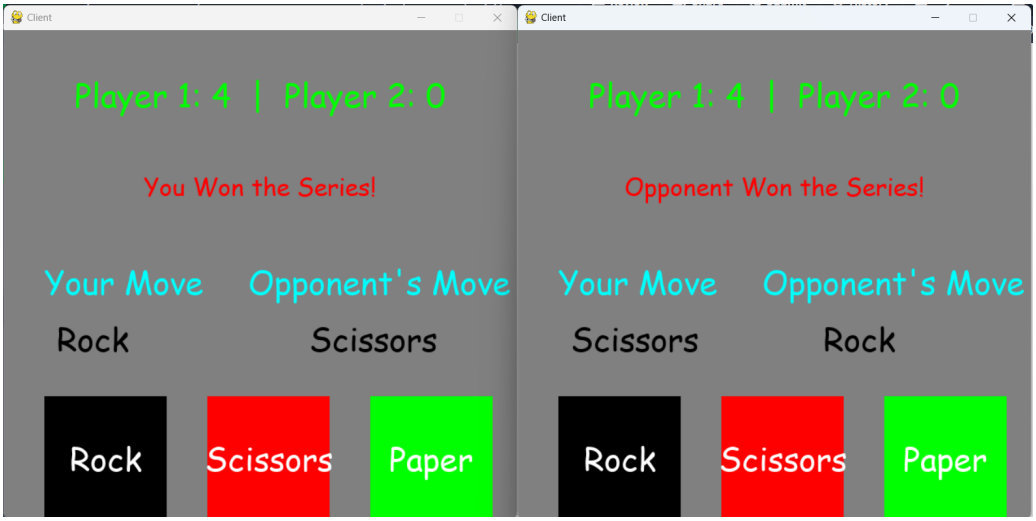

Figure 8: After player 1 scores their 5th win

And after that, both players' scores will be reset to zero.

For more information you can find our project, here is our group project's GitHub link: https://github.com/Antwerp2004/Computer-Networking-Extended-Course-CO309B-HCMUT.git

5.2 Challenges

Rock-Paper-Scissors is actually a really simple game. The project itself currently still has many flaws, such as the UI isn't good, and the players are not allowed to choose the players

they want to play with, because matchmaking is done automatically by the server. Moreover, the game doesn't have much features.

5.3 Conclusions and Future works

In the future, we intend to make a more complex game, as well as adding more features to the game. Moreover, the client should be able to interact with each other rather than communicating via a server as of present.