UNIVERSIDAD FRANCISCO MARROQUÍN

Facultad de Ciencias Económicas

Introducción a Redes

Reporte de Análisis: Proyecto Final Juego Tic - Tac - Toe

Ruiz, Keneth 2021010

Fernandez, Juan Luis 20200112

Sebastian Rivera 20210389

Introduccion

Este reporte explica el funcionamiento de dos programas Python que implementan un servidor y un cliente para jugar Tic-Tac-Toe en red. A continuación, se detalla cómo funcionan ambos códigos y cómo ejecutarlos.

1. Servidor de Tic-Tac-Toe

Descripción General

El servidor gestiona la lógica del juego Tic-Tac-Toe y la comunicación entre dos clientes. Utiliza sockets para la comunicación en red y threading para manejar múltiples conexiones de clientes simultáneamente.

Código y Funcionamiento:

Python

```python
import socket
import threading
import json
import time

class TicTacToeServer:
    def __init__(self, host='localhost', port=5556):
        self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.server.bind((host, port))
        self.server.listen(2)
        print("Servidor de Tic-Tac-Toe iniciado, esperando conexiones...")

        self.clients = {}
        self.board = [' ' for _ in range(9)]
        self.current_player = None
        self.players = ['X', 'O']
        self.winner = None

    def handle_client(self, client, player):
        while True:
            try:
```

```
                message = client.recv(1024).decode()
                if not message:
                    break

                data = json.loads(message)
                print(f"Received data from player {player}: {data}")

                if data['action'] == 'move':
                    self.make_move(player, data['position'])

                self.broadcast_game_state()

                if self.winner:
                    self.broadcast_game_state(game_over=True)
                    time.sleep(3)  # Espera 3 segundos antes de reiniciar el
juego
                    self.reset_game()
                    self.broadcast_game_state()

            except Exception as e:
                print(f"Error: {e}")
                self.clients.pop(player, None)
                client.close()
                break

    def make_move(self, player, position):
        if self.board[position] == ' ' and self.current_player == player:
            self.board[position] = self.players[player]
            self.check_winner()
            self.current_player = 1 - self.current_player

    def check_winner(self):
        winning_combinations = [
            [0, 1, 2], [3, 4, 5], [6, 7, 8],
```

```
            [0, 3, 6], [1, 4, 7], [2, 5, 8],
            [0, 4, 8], [2, 4, 6]
        ]
        for combo in winning_combinations:
            if self.board[combo[0]] == self.board[combo[1]] ==
self.board[combo[2]] != ' ':
                self.winner = self.board[combo[0]]

    def broadcast_game_state(self, game_over=False):
        for player, client in self.clients.items():
            game_state = {
                'board': self.board,
                'current_player': self.current_player,
                'winner': self.winner,
                'your_symbol': self.players[player],
                'game_over': game_over
            }
            client.sendall((json.dumps(game_state) + "\n").encode())

    def reset_game(self):
        self.board = [' ' for _ in range(9)]
        self.current_player = 0
        self.winner = None

    def start(self):
        threading.Thread(target=self.accept_clients).start()

    def accept_clients(self):
        while len(self.clients) < 2:
            client, address = self.server.accept()
            player = len(self.clients)
            self.clients[player] = client
            threading.Thread(target=self.handle_client, args=(client,
player)).start()
```

```
            print(f'Player {player} conectado desde {address}')

        self.current_player = 0
        self.broadcast_game_state()

if __name__ == "__main__":
    server = TicTacToeServer()
    server.start()
```

Explicación del Código

1. Inicialización: El servidor se inicializa y se pone a la escucha de conexiones en el puerto 5556.
2. Manejo de Clientes: Utiliza hilos (threading) para manejar las conexiones de dos clientes simultáneamente.
3. Lógica del Juego: Gestiona el tablero, verifica movimientos válidos y determina el ganador.
4. Comunicación: Utiliza JSON para enviar y recibir el estado del juego entre el servidor y los clientes.
5. Reinicio del Juego: Tras un juego terminado, reinicia el tablero y permite comenzar un nuevo juego.

Librerías Necesarias

- socket
- threading
- json
- Time

Cómo Ejecutar el Programa

- Asegúrate de tener Python instalado en tu sistema.
- Guarda el código en un archivo llamado tic_tac_toe_server.py.
- Ejecuta el archivo desde la línea de comandos con: python tic_tac_toe_server.py.

2. Cliente de Tic-Tac-Toe

Descripción General
El cliente se conecta al servidor de Tic-Tac-Toe y permite a un jugador interactuar con el juego a través de una interfaz gráfica construida con Tkinter.

Python

```python
import socket
import json
import threading
import tkinter as tk
from tkinter import messagebox

class TicTacToeClient:
    def __init__(self, host='localhost', port=5556):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.client.connect((host, port))
        self.root = tk.Tk()
        self.root.title("Tic-Tac-Toe Cliente")
        self.create_interface()
        threading.Thread(target=self.receive_data).start()
        self.root.mainloop()

    def create_interface(self):
        self.buttons = []
        for i in range(9):
            button = tk.Button(self.root, text=' ', font=('Helvetica', 20),
height=3, width=6, command=lambda i=i: self.send_move(i))
            button.grid(row=i//3, column=i%3)
            self.buttons.append(button)
        self.info_label = tk.Label(self.root, text="Esperando jugador...",
font=("Helvetica", 14))
        self.info_label.grid(row=3, columnspan=3)

    def update_interface(self, game_state):
        for i in range(9):
            self.buttons[i].config(text=game_state['board'][i])
```

```
        if game_state['winner']:
            winner = game_state['winner']
            messagebox.showinfo("Juego Terminado", f"¡{winner} ha ganado!")
            self.disable_buttons()
        else:
            current = 'X' if game_state['current_player'] == 0 else 'O'
            self.info_label.config(text=f"Turno de {current} | Tu símbolo:
{game_state['your_symbol']}")

        if game_state['game_over']:
            self.enable_buttons()

    def send_move(self, position):
        data = json.dumps({'action': 'move', 'position': position})
        self.client.sendall(data.encode())

    def receive_data(self):
        buffer = ""
        while True:
            try:
                message = self.client.recv(1024).decode()
                if not message:
                    break

                buffer += message
                while "\n" in buffer:
                    line, buffer = buffer.split("\n", 1)
                    data = json.loads(line)
                    self.update_interface(data)

            except Exception as e:
                print(f"Error: {e}")
                break
```

```
    def disable_buttons(self):
        for button in self.buttons:
            button.config(state=tk.DISABLED)

    def enable_buttons(self):
        for button in self.buttons:
            button.config(state=tk.NORMAL)

if __name__ == "__main__":
    TicTacToeClient()
```

Explicación del Código

- Inicialización: El cliente se conecta al servidor en el puerto 5556.
- Interfaz Gráfica: Utiliza Tkinter para crear la interfaz del juego con botones que representan las posiciones del tablero.
- Comunicación: Envía y recibe el estado del juego utilizando sockets y JSON.
- Actualización de la Interfaz: Actualiza la interfaz gráfica según el estado del juego recibido del servidor.

Librerías Necesarias

- socket
- json
- threading
- Tkinter

Cómo Ejecutar el Programa

- Asegúrate de tener Python instalado en tu sistema.
- Guarda el código en un archivo llamado tic_tac_toe_client.py.
- Ejecuta el archivo desde la línea de comandos con: python tic_tac_toe_client.py.