Relatório - Jogo da Velha com Sockets

Este relatório descreve o projeto de um Jogo da Velha em Python utilizando sockets e threads.

Funcionalidades implementadas:

- Comunicação cliente-servidor usando socket TCP/IP;
- Uso de duas threads no servidor, uma para cada cliente;
- Controle de jogadas alternadas e verificação de vitória ou empate;
- Interface de linha de comando para jogadores.

Protocolos e conceitos de rede aplicados:

- Criação de sockets com a primitiva `socket()`;
- Comunicação bidirecional com `send` e `recv`;
- Controle de concorrência com `threading` e `lock`;
- Utilização das funções primitivas como `input()` e `print()`.

Possíveis melhorias futuras:

- Interface gráfica;
- Suporte a múltiplas partidas consecutivas;
- Armazenamento de histórico de partidas.

Dificuldades encontradas:

- Sincronização entre os clientes e controle de turno;
- Manipulação de erros em entradas inválidas;
- Finalização adequada das conexões após término do jogo.

Segue, abaixo, o código-fonte impresso do projeto.

Código: server.py

```
import socket
import threading

board = [" " for _ in range(9)]
players = []
lock = threading.Lock()
current_player = 0

def print_board():
    return f"""
 {board[0]} | {board[1]} | {board[2]}
-----------
 {board[3]} | {board[4]} | {board[5]}
-----------
 {board[6]} | {board[7]} | {board[8]}
"""

def check_winner():
    wins = [(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 a, b, c in wins:
        if board[a] == board[b] == board[c] != " ":
            return True
    return False

def handle_player(conn, addr, player_id):
    global current_player
    symbol = "X" if player_id == 0 else "O"
    conn.sendall(f"Você é o jogador {symbol}.\n".encode())

    while True:
        with lock:
            if current_player != player_id:
                continue
            conn.sendall(print_board().encode())
            conn.sendall("Escolha uma posição (0-8): ".encode())

        try:
            pos = int(conn.recv(1024).decode())
        except:
            conn.sendall("Entrada inválida.\n".encode())
            continue

        with lock:
            if board[pos] == " ":
                board[pos] = symbol
                if check_winner():
                    for p in players:
                        p.sendall(print_board().encode())
                        p.sendall(f"Jogador {symbol} venceu!\n".encode())
                    break
                elif " " not in board:
                    for p in players:
                        p.sendall(print_board().encode())
                        p.sendall("Empate!\n".encode())
                    break
                current_player = 1 - current_player
            else:
                conn.sendall("Posição ocupada, tente outra.\n".encode())
```

```
    conn.close()

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("localhost", 5555))
    s.listen(2)
    print("Servidor aguardando jogadores...")

    while len(players) < 2:
        conn, addr = s.accept()
        players.append(conn)
        print(f"Jogador {len(players)} conectado: {addr}")
        threading.Thread(target=handle_player, args=(conn, addr, len(players)-1)).start()

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

Código: client.py

```
import socket
import threading

def receive_messages(sock):
    while True:
        try:
            data = sock.recv(1024).decode()
            if not data:
                break
            print(data)
        except:
            break

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("localhost", 5555))

    threading.Thread(target=receive_messages, args=(s,), daemon=True).start()

    while True:
        try:
            msg = input()
            s.sendall(msg.encode())
        except:
            break

    s.close()

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