Implementação de uma Aplicação de Rede: Jogo da Memória

Samuel Damiani Frigotto

21 de abril de 2025

Introdução

Este relatório descreve a implementação de um jogo da memória utilizando sockets TCP, como parte da disciplina de Redes de Computadores 1. O projeto visa demonstrar o funcionamento de uma aplicação de rede baseada em comunicação cliente-servidor, com troca de mensagens estruturadas e controle de turnos. A aplicação foi desenvolvida em Python, com suporte a interface gráfica para os jogadores.

Descrição do Jogo

O jogo da memória é composto por 30 cartas organizadas em uma grade, contendo 15 pares de animais distintos. Participam dois jogadores, que se conectam ao servidor e jogam alternadamente. A cada turno, o jogador seleciona duas cartas:

- Se as cartas forem iguais, o jogador marca um ponto e pode jogar novamente.
- Se forem diferentes, as cartas são ocultadas novamente após um breve intervalo e a vez passa ao adversário.
- O jogo termina quando um jogador encontra 8 pares.

O tabuleiro e o placar são sincronizados entre os dois clientes por meio de mensagens trocadas com o servidor.

Estrutura de Arquivos

- **cliente.py**: Contém a lógica da interface gráfica do jogo usando `tkinter`, além da comunicação com o servidor via sockets. Esse arquivo é executado pelo jogador.
- **servidor.py**: Contém a lógica principal do jogo e gerencia toda a comunicação entre os dois clientes conectados. Ele controla os turnos, valida jogadas e envia atualizações para os jogadores.

- **comum.py**: Arquivo auxiliar utilizado tanto pelo cliente quanto pelo servidor. Contém enums e constantes compartilhadas, como a enumeração dos animais que compõem os pares de cartas.
- **Pasta `imagens/`**: Contém as imagens JPEG dos 15 animais e do verso das cartas. Cada imagem é nomeada conforme o nome do animal, em letras minúsculas, para ser carregada dinamicamente.

Protocolo de Comunicação

A comunicação entre cliente e servidor é realizada por meio de mensagens binárias. Cada mensagem começa com um *opcode*, que define seu tipo, seguido dos dados correspondentes. A tabela abaixo apresenta os *opcodes* utilizados:

Opcode	Descrição	Dados Adicionais
0	Conexão inicial	[0, id_jogador]
1	Atualização do tabuleiro	30 bytes com os IDs dos animais
2	Sua vez de jogar	Nenhum
3	Jogada do cliente	[3, posição]
4	Resultado da jogada	[acerto, pos1, id1, pos2, id2]
5	Atualização do placar	[5, pontos_jogador_0, pontos_jogador_1]
6	Fim do jogo	[id_vencedor]
7	Jogada do adversário	[pos1, pos2, id1, id2]
8	Carta revelada temporariamente	[posição, id_animal]
9	Mensagem de erro	[tamanho_mensagem, mensagem_em_bytes]
10	Vez do adversário	[1] se ele acertou, [0] se errou, ou nenhum dado

Diferença entre Opcode 2 e Opcode 10

Embora ambos estejam relacionados à indicação de turno, o **opcode 2** e o **opcode 10** possuem funções distintas e são tratados de forma diferente no cliente.

Opcode 2 - Sua vez de jogar

Este opcode é enviado pelo servidor quando é a vez do cliente jogar. Ele sinaliza que o cliente pode clicar nas cartas para realizar uma jogada. No cliente, isso ativa a variável `vez_de_jogar = True` e exibe a mensagem "Sua vez de jogar!"na interface.

Implementação no cliente:

- Define o status da vez.
- Reseta as cartas selecionadas.

Opcode 10 - Vez do adversário

Este opcode é enviado ao cliente quando é a vez do oponente jogar. Pode ser enviado com um único byte adicional indicando o resultado da jogada anterior do adversário:

- `1` – o adversário acertou um par.
- `0` – o adversário errou.
- Nenhum byte extra – apenas uma notificação de que não é a vez do jogador.

Implementação no cliente:

- Atualiza a interface para exibir " Vez do adversário...".
- Impede o jogador de clicar nas cartas.
- Opcionalmente, mostra se o adversário acertou ou errou.

Fluxo de Mensagens

O fluxo de comunicação entre cliente e servidor ocorre da seguinte maneira:

1. Cliente se conecta ao servidor e recebe seu ID (opcode 0).
2. Quando dois clientes estão conectados, o servidor inicia o jogo:
    - Envia o tabuleiro para ambos (opcode 1).
    - Informa ao jogador inicial que é sua vez (opcode 2) e ao outro que é a vez do adversário (opcode 10).
3. O jogador realiza a jogada (opcode 3), selecionando duas cartas.
4. O servidor revela temporariamente as cartas para ambos (opcode 8).
5. Após verificar se as cartas são um par:
    - Envia o resultado ao jogador (opcode 4).
    - Envia os dados da jogada para o adversário (opcode 7).
    - Atualiza o placar (opcode 5).
    - Envia o novo tabuleiro (opcode 1).
    - Se necessário, informa o fim do jogo (opcode 6).
    - Caso não haja acerto, troca a vez (opcode 2 e 10).

Implementação Técnica

Servidor

O servidor é executado em `servidor.py`, utilizando as bibliotecas `socket` e `threading`. Ele aceita dois jogadores, gera um tabuleiro com 15 pares aleatórios e inicia o jogo. A cada jogada, verifica a validade, revela cartas, atualiza o placar e envia mensagens apropriadas aos jogadores. A lógica do jogo é centralizada no servidor.

Cliente

O cliente é executado em `cliente.py` e implementa uma interface gráfica com `tkinter`. Ele se conecta ao servidor, exibe o tabuleiro com botões gráficos que mostram imagens dos animais, e responde aos comandos recebidos do servidor. Toda a interação com o usuário acontece nesse arquivo.

Arquivo Comum

O arquivo `comum.py` contém definições compartilhadas, como a enumeração dos animais e funções auxiliares, como a que gera o tabuleiro inicial. Isso permite manter consistência entre servidor e cliente, evitando duplicação de código.

Conclusão

A implementação do jogo da memória em rede permitiu aplicar os conceitos de comunicação entre processos utilizando o protocolo TCP. A definição de um protocolo de mensagens próprio e a implementação da lógica do jogo em um servidor centralizado proporcionaram uma experiência prática valiosa sobre desenvolvimento de aplicações distribuídas. A utilização de uma interface gráfica no cliente tornou a aplicação mais intuitiva e próxima de um produto real.