

UNIVERSIDADE FEDERAL DE PELOTAS

Centro de Desenvolvimento Tecnológico

Barachelado em Computação



Jogo UNO em Arquitetura P2P (TCP Socket)

Protocolo da Camada de Aplicação com Aplicações em Jogos
Distribuídos

Eduardo Schwantz, Isabele Santos Scherdien, Thiago Dias Mazzoni

Protocolo da Camada de Aplicação Jogo UNO - Arquitetura Peer-to-Peer

Eduardo Schwantz, Isabele Santos Scherdien, Thiago Dias Mazzoni

5 de agosto de 2025

Sumário

1	Objetivo da Aplicação	2
2	Características do Protocolo	2
3	Tipos de Mensagens Enviadas pelo Cliente (Peer)	2
4	Tipos de Mensagens Recebidas pelos Peers	2
5	Formato das Mensagens e Campos de Cabeçalho	3
6	Casos de Uso e Fluxo de Comunicação	4
7	Considerações Finais	4

1 Objetivo da Aplicação

Este trabalho tem como objetivo apresentar o protocolo de camada de aplicação desenvolvido para um jogo de UNO digital, utilizando uma arquitetura peer-to-peer. O protocolo permite que múltiplos jogadores participem de uma partida de forma distribuída, mantendo o estado do jogo sincronizado via troca de mensagens TCP.

2 Características do Protocolo

Característica	Valor	Justificativa
Arquitetura	Peer-to-Peer (P2P)	Todos os jogadores atuam como peers e se conectam entre si.
Estado	Com estado (stateful)	O jogo mantém informações persistentes como cartas, turnos e direção.
Persistência	Persistente	As conexões TCP são mantidas durante toda a partida.
Comunicação	Push	Atualizações são enviadas automaticamente aos peers.
Controle	Na banda	Dados de controle e jogo trafegam juntos no mesmo canal.
Formato	JSON delimitado por nova linha	Facilita parsing contínuo e compatibilidade entre linguagens.
Protocolo de Transporte	TCP	Garante confiabilidade na entrega e ordem das mensagens.

3 Tipos de Mensagens Enviadas pelo Cliente (Peer)

- **JOIN_GAME**: solicita entrada de um jogador na partida.
- **PLAY_CARD**: envia uma jogada realizada (carta jogada).
- **BUY_CARD**: solicita a compra de uma carta.
- **CHAT_MSG**: envia mensagem de chat para os demais jogadores.

4 Tipos de Mensagens Recebidas pelos Peers

- **JOIN_GAME**: novo jogador ingressando na partida.
- **GAME_STATE**: sincronização do estado atual do jogo.
- **PLAY_CARD**: jogada recebida de outro jogador.
- **BUY_CARD**: compra realizada por outro peer.
- **CHAT_MSG**: exibição de mensagem no terminal.

5 Formato das Mensagens e Campos de Cabeçalho

Formato geral

Listing 1: Estrutura padrão da mensagem

```
{
  "type": "TIPO_DA_MENSAGEM",
  "sender": "Nome do jogador",
  "timestamp": "2025-08-04T14:00:00.000Z",
  "payload": { ... }
}
```

Campos do payload por tipo

JOIN_GAME

```
{
  "player_name": "nome_do_jogador"
}
```

PLAY_CARD

```
{
  "card": { "color": "blue", "value": "reverse" },
  "color": "red" // usado em cartas coringa
}
```

BUY_CARD

```
{}
```

CHAT_MSG

```
{
  "message": "mensagem do chat"
}
```

GAME_STATE (resumo)

```
{
  "deck": [...],
  "players": {
    "Jo o": [...],
    "Maria": [...]
  },
  "current_turn": "Maria",
  "top_card": { "color": "blue", "value": "+2" },
  "direction": 1,
  "turn_order": ["Jo o", "Maria"]
}
```

6 Casos de Uso e Fluxo de Comunicação

Início do jogo

1. Peer A abre socket TCP e espera conexões.
2. Peer B se conecta e envia `JOIN_GAME`.
3. Peer A reconhece e inicia a partida com `GAME_STATE`.

Jogada

1. Jogador envia `PLAY_CARD`.
2. Peers validam jogada e retransmitem estado atualizado.

Chat

1. Qualquer peer envia `CHAT_MSG`.
2. Todos os peers exibem a mensagem.

7 Considerações Finais

O protocolo proposto demonstrou-se eficaz para partidas simples de UNO entre múltiplos jogadores, mesmo sem um servidor central. Futuras melhorias podem incluir:

- Criptografia ponta a ponta nas mensagens (ex: TLS).
- Detecção e recuperação de falhas de peers desconectados.
- Persistência de estado em disco ou banco de dados.

Dificuldades encontradas:

- Manter consistência de estado em múltiplos peers simultaneamente.
- Gerenciamento de ordem de jogadas e eventos paralelos.
- Diagnóstico de erros silenciosos nas conexões TCP.