Redes de Computadores

Trabalho Final

Campo Minado Cooperativo

Luis Eduardo Rasch & Gabriel Moura

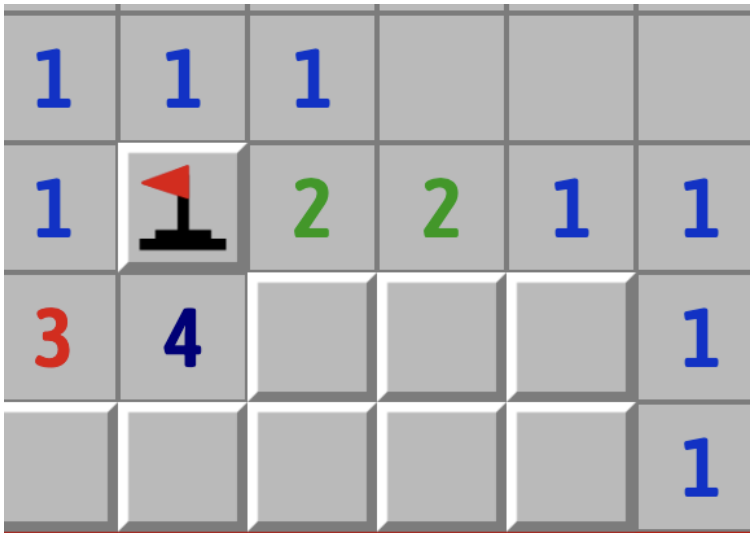

Imagem meramente ilustrativa

1. Objetivo e Função da Aplicação e Protocolo Desenvolvido

O protocolo de aplicação desenvolvido tem como objetivo principal permitir a implementação de um jogo campo minado cooperativo em turnos, onde dois jogadores podem participar simultaneamente de uma mesma partida. A aplicação visa proporcionar uma experiência colaborativa, mantendo a sincronização do estado do tabuleiro entre os participantes conectados durante a jogatina.

A função do protocolo é garantir a comunicação confiável entre os participantes, utilizando uma arquitetura peer-to-peer com coordenação centralizada. Nesse modelo, ambos os computadores são tratados como peers, mas um deles assume o papel de host (servidor lógico) responsável por validar e coordenar todas as ações do jogo, enquanto o outro peer atua como cliente conectado diretamente ao host. Essa abordagem assegura que todas as jogadas sejam processadas de forma consistente e propagadas corretamente para todos os participantes.

2. Características do Protocolo Desenvolvido

O protocolo implementa uma arquitetura peer-to-peer (P2P) híbrida com coordenação centralizada utilizando PacketPeerUDP do Godot. Um dos jogadores assume o papel de host iniciando o listener UDP na porta definida, enquanto outro jogador conecta-se diretamente através do IP e porta do host, mantendo comunicação direta entre os participantes.

- Protocolo de Transporte: O protocolo utiliza UDP como protocolo de transporte subjacente, oferecendo comunicação de baixa latência ideal para jogos em tempo real. Para garantir a entrega de mensagens críticas (como ações de jogo), o protocolo implementa um sistema próprio de acknowledgment para pacotes que requerem confiabilidade.
- Estado do Protocolo: O protocolo é implementado como sem estado, onde cada mensagem carrega todas as informações necessárias para sua interpretação, eliminando dependências de contexto anterior e simplificando a recuperação de erros.
- Modelo de Comunicação: Implementa abordagem push, onde os peers propagam ativamente suas ações para os demais participantes assim que ocorrem eventos relevantes no jogo. Mensagens críticas (revelação de células, bandeiras) utilizam confirmação de recebimento.
- Arquitetura de Canal: A implementação de controle na banda unifica controle e dados no mesmo canal UDP, simplificando a arquitetura de rede. O protocolo utiliza um campo "type" no cabeçalho para distinguir entre diferentes tipos de mensagens.

3. Listagem dos Tipos de Mensagens Possíveis do Cliente para o Servidor

As mensagens enviadas pelos clientes para o host incluem:

- handshake: Solicita conexão inicial ao host, incluindo informações do jogador.
- cell_flag: Informa ação de marcar/desmarcar bandeira em célula.
- player_ready: Sinaliza que o cliente está preparado para iniciar.
- game_action: Solicita execução de ações gerais coordenadas pelo host.
- disconnect: Notifica saída controlada da sessão.
- game_lose: Comunica finalização da partida com derrota.
- game_win: Comunica finalização da partida com vitória.
- ack: Confirma recebimento de mensagens críticas.

4. Listagem e Descrição dos Tipos de Mensagens Possíveis do Servidor para o Cliente

O host envia mensagens para os clientes através do protocolo UDP push sem estado, incluindo:

- handshake_response: Confirma estabelecimento da conexão com ID do peer e comunica início com configurações do tabuleiro.
- player_joined: Notifica entrada de novo participante.
- player_left: Informa desconexão de participante.
- flag_updated: Sincroniza mudanças no estado das bandeiras.
- game_ended: Comunica resultado da partida.
- error: Comunica falhas na validação de ações.
- ack_request: Solicita confirmação para mensagens críticas.

5. Formato de Cada Tipo de Mensagem e Respectivos Campos de Cabeçalho

O protocolo utiliza mensagens estruturadas como Dictionary do Godot, serializadas através da função var_to_bytes() para transmissão via PacketPeerUDP. Cada mensagem segue uma estrutura base contendo campos de cabeçalho padronizados que permitem identificação, roteamento e controle de confiabilidade. As mensagens são representadas em formato JSON para fins de padronização.

A estrutura base inclui o campo "type" como string identificando o tipo específico da mensagem, "sender_id" especificando o peer remetente no formato "IP:PORT", "target_id" indicando o destinatário (string "IP:PORT" específico ou "all" para broadcast), "sequence" com número inteiro incremental utilizado para controle de acknowledgment, "requires_ack" como boolean indicando se a mensagem requer confirmação de recebimento, e "data" como Dictionary contendo os dados específicos de cada tipo de mensagem.

Mensagens de início de jogo transportam no campo "data" as informações "board_size" como Dictionary com "width" e "height" inteiros, "mine_count" especificando o número total de minas. Para mensagens de revelação de células, o campo "data" contém as chaves "x" e "y" como inteiros representando as coordenadas da célula no tabuleiro. Mensagens de atualização do tabuleiro incluem no campo "data" um array "changes" onde cada elemento é um Dictionary contendo "x" e "y" para posição e "revealed" como boolean indicando se a célula foi revelada.

6. Especificação dos Valores Possíveis para Cada Campo de Cabeçalho

Campos de Cabeçalho:

- type: String identificando o tipo da mensagem (valores listados nas seções 3 e 4)
- sender_id: String no formato "IP:PORT" do remetente
- target_id: String "IP:PORT" específico ou "all" para broadcast

Dados Específicos:

- Coordenadas x,y: Inteiros não-negativos dentro dos limites do tabuleiro.
- Estados de célula: Inteiros (-1 para mina, 0 para vazia, 1-8 para contagem de minas adjacentes).
- Flags de bandeira: Boolean (true/false).
- player_id: String única no formato "IP:PORT".
- Códigos de erro: Strings descritivas.

Sistema de Sincronização:

- board_state: Array completo do tabuleiro atual.
- flags_state: Dictionary com todas as bandeiras marcadas.
- game_status: String indicando estado atual da partida ("playing", "won", "lost").