Implementação de Jogo Caça ao Tesouro com Comunicação por Raw Sockets

Pedro Henrique Marques de Lima [*1] **, Felipe Gonçalves Pereira** [†1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

{phml23,fgp23}@inf.ufpr.br

Abstract. *This paper describes the implementation of a treasure hunt game using raw sockets for network communication. The project implements a client-server architecture where players navigate through an 8x8 grid searching for treasures. Communication between client and server uses a custom protocol inspired by Kermit, implementing flow control with stop-and-wait mechanism. The system handles file transfer of different types (text, images, videos) and includes error handling and timeout management.*

Resumo. *Este artigo descreve a implementação de um jogo de caça ao tesouro utilizando raw sockets para comunicação em rede. O projeto implementa uma arquitetura cliente-servidor onde jogadores navegam por um grid 8x8 em busca de tesouros. A comunicação entre cliente e servidor utiliza um protocolo customizado inspirado no Kermit, implementando controle de fluxo com mecanismo stop-and-wait. O sistema gerencia transferência de arquivos de diferentes tipos (texto, imagens, vídeos) e inclui tratamento de erros e gerenciamento de timeouts.*

1. Introdução

O projeto consiste na implementação de um jogo de caça ao tesouro que utiliza comunicação de baixo nível através de raw sockets. O objetivo é demonstrar conceitos fundamentais de redes de computadores, especificamente protocolos de nível 2, controle de fluxo e transferência confiável de dados.

O jogo implementa uma arquitetura cliente-servidor onde o servidor controla o estado do jogo e gerencia os tesouros, enquanto o cliente fornece a interface gráfica para interação do usuário. A comunicação entre as partes utiliza um protocolo customizado inspirado no protocolo Kermit. Link repositório .

2. Arquitetura do Sistema

2.1. Modelo Cliente-Servidor

O sistema foi projetado seguindo o modelo cliente-servidor tradicional:

- **Servidor**: Responsável por manter o estado do jogo, incluindo posições dos tesouros e localização atual do jogador. Gerencia também a transferência dos arquivos de tesouro.
- **Cliente**: Implementa a interface gráfica usando SDL2, permitindo ao usuário navegar pelo mapa e visualizar tesouros encontrados.

2.2. Comunicação por Raw Sockets

A comunicação utiliza raw sockets, permitindo controle direto sobre os pacotes de rede no nível 2 (enlace). Esta abordagem foi escolhida para:

- Implementar controle de fluxo customizado
- Gerenciar detecção e correção de erros manualmente

3. Protocolo de Comunicação

3.1. Estrutura do Pacote

O protocolo implementado é inspirado no Kermit e possui a seguinte estrutura:

```
1 typedef struct Package {
2     unsigned char start;        // Marcador inicio (0x7E)
3     unsigned char info_upper;   // Tamanho (7 bits) + Seq
          (1 bit)
4     unsigned char info_down;    // Seq (4 bits) + Tipo (4
          bits)
5     unsigned char checksum;     // Checksum dos campos
6 } Package;
```

Listing 1. Estrutura do pacote de comunicação

Os campos do protocolo são:

- **Marcador de início**: Byte fixo 0x7E para sincronização
- **Tamanho**: 7 bits indicando bytes de dados (0-127)
- **Sequência**: 5 bits para controle de sequência
- **Tipo**: 4 bits identificando o tipo da mensagem
- **Checksum**: 8 bits para detecção de erros

3.2. Tipos de Mensagem

O protocolo define 16 tipos de mensagem diferentes:

- **Controle**: ACK (0), NACK (1), OK_ACK (2), ERRO (15)
- **Transferência**: TAMANHO (4), DADOS (5), FIM_ARQUIVO (9)
- **Arquivos**: TEXTO_ACK_NOME (6), VIDEO_ACK_NOME (7), IMAGEM_ACK_NOME (8)
- **Movimento**: DIREITA (10), CIMA (11), BAIXO (12), ESQUERDA (13)

3.3. Controle de Fluxo

Foi implementado o mecanismo stop-and-wait para controle de fluxo:

```
do {
    package_assembler(buffer, 0, sequencia, tipo, NULL);
    if (send(socket, buffer, sizeof(buffer), 0) < 0) {
        perror("Erro ao enviar\n");
        exit(0);
    }
} while (recebe_mensagem(socket, TEMPO_REENVIO,
         buffer, sizeof(buffer), sequencia) < 0);
```

Listing 2. Implementação do controle de fluxo

4. Funcionalidades Implementadas

4.1. Interface Gráfica

A interface foi implementada usando SDL2, proporcionando:

- Marcação de posições visitadas
- Indicação de tesouros encontrados

4.2. Transferência de Arquivos

O sistema suporta transferência de três tipos de arquivo:

- Textos (.txt)
- Imagens (.jpg)
- Vídeos (.mp4)

5. Tratamento de Erros

5.1. Detecção de Erros

O sistema implementa checksum para detecção de erros:

```
unsigned char checksum(unsigned char *buffer) {
    unsigned char sum = 0;
    sum += get_size(buffer);
    sum += get_sequence(buffer);
    sum += get_type(buffer);

    for (int i = 0; i < get_size(buffer); i++) {
        sum += buffer[DATA + i];
    }
    return sum;
}
```

Listing 3. Cálculo do checksum

5.2. Recuperação de Erros

Quando erros são detectados, o protocolo implementa:

- Retransmissão automática em caso de timeout
- Envio de NACK para pacotes corrompidos
- Códigos de erro específicos para diferentes situações

6. Resultados e Testes

O sistema foi testado em ambiente Linux utilizando conexão direta por cabo ethernet entre duas máquinas. Os testes demonstraram:

- Comunicação confiável entre cliente e servidor
- Transferência correta de arquivos de diferentes tipos
- Recuperação adequada de erros de transmissão

7. Conclusão

O projeto demonstrou com sucesso a implementação de um protocolo de comunicação customizado utilizando raw sockets. O jogo de caça ao tesouro fornece uma aplicação prática para conceitos fundamentais de redes de computadores, incluindo controle de fluxo, detecção de erros e transferência confiável de dados.

As principais características do trabalho incluem:

- Implementação completa de protocolo inspirado no Kermit
- Sistema de transferência de arquivos
- Interface gráfica com SDL2
- Tratamento adequado de erros e timeouts