Relatorio Trabalho 1

João Caboclo GRR20221227, Iago Bariuka GRR20244409)

23 de junho de 2025

Resumo

Este relatório descreve a utilização de timeouts e tipos específicos de mensagens na implementação do trabalho T1 da disciplina CI1058 – Redes 1, baseado em comunicação via raw sockets com protocolo personalizado de camada 2. São detalhadas as estratégias de controle de tempo e confiabilidade, bem como os tipos 3 e 14 do protocolo.

1 Introdução

No contexto do Trabalho T1 da disciplina CI1058 – Redes 1, foi implementado um jogo cliente-servidor com comunicação direta via raw sockets, utilizando um protocolo inspirado no Kermit. O protocolo implementa controle de fluxo do tipo stop-and-wait e inclui mecanismos para tratar falhas na comunicação através de retransmissões condicionadas por timeouts.

2 Uso de Timeout

O mecanismo de timeout é empregado para garantir que mensagens sejam recebidas dentro de um intervalo de tempo razoável, evitando bloqueios indefinidos na comunicação. Foi adotada uma abordagem de **backoff exponencial** para aumentar a robustez do sistema frente a falhas temporárias na rede.

2.1 Timeout no Loop Principal

No servidor, a função `recebe_mensagem` recebe um parâmetro `timeoutMillis`, configurando o tempo máximo de espera por uma mensagem válida. Esse tempo é inicialmente 3000 milissegundos. Caso nenhuma mensagem válida seja recebida nesse intervalo, o tempo de espera é duplicado e uma nova tentativa é feita. Esse processo se repete até o número máximo de 5 tentativas:

```
int status = recebe_mensagem(soquete, timeout, &f, buffer, sizeof(buffer));
if (status <= 0) {
    printf("  ␣Timeout␣aguardando␣comando...␣Tempo␣de␣espera␣%d␣ms\n", timeout);
    timeout *= 2;
    tentativas++;
}
```

2.2 Timeout em Envio com Confirmação

Na função `envia_com_ack`, utilizada para garantir a entrega confiável de quadros, o timeout começa com 500ms e é duplicado a cada falha de recebimento. O envio é tentado no máximo 5 vezes:

```
timeout *= 2;  // Backoff exponencial
```

Esse comportamento está alinhado com práticas comuns em protocolos confiáveis, como o TCP.

3 Tipo 14: Fim de Jogo

O tipo de mensagem `0x0E` (14 em decimal) é utilizado para indicar ao cliente que **todos os tesouros foram encontrados**. Após detectar essa condição, o servidor altera o tipo do ACK de resposta para esse valor:

```
if (todos_tesouros_encontrados(&mapa)) {
    ack_type = 0x0E;
}
```

Essa sinalização permite ao cliente encerrar o jogo de forma limpa e sincronizada com o servidor.

4 Tipo 3: Envio da Posição Inicial

A mensagem do tipo `0x03` (3 em decimal) serve para **informar ao cliente a posição inicial do jogador no mapa**. Ela contém dois bytes de dados com as coordenadas (linha e coluna):

```
unsigned char tipo = 0x03;
dados[0] = m->linha;
dados[1] = m->coluna;
```

O cliente reconhece esse tipo de mensagem e responde com um ACK para confirmar o recebimento correto.

5 Transferência de Arquivos

O protocolo também contempla a transferência de arquivos multimídia (texto, imagem e vídeo) quando o jogador encontra um tesouro no mapa. Essa transferência é iniciada pelo servidor e finalizada pelo cliente, com confirmação de entrega garantida via stop-and-wait.

5.1 No Servidor

Ao detectar que o jogador alcançou a posição de um tesouro, o servidor localiza o arquivo correspondente na pasta `./objetos`, identificando seu tipo (`texto`, `jpg`, `mp4`) com base na assinatura do arquivo. O tipo determina o código da mensagem de envio (6, 7 ou 8).

O servidor então envia:

- O nome do arquivo

- O tamanho total (em 4 bytes)
- O conteúdo em blocos de até 127 bytes (tipo 5)
- Uma mensagem final com tipo 9, indicando fim de arquivo

Cada envio é acompanhado de espera por ACK antes de prosseguir com o próximo bloco, garantindo entrega ordenada e confiável.

5.2 No Cliente

O cliente, ao receber o nome do arquivo, envia um ACK e aguarda o frame com o tamanho do arquivo. Antes de prosseguir, ele verifica se existe espaço livre em disco suficiente utilizando a função `statvfs`. Se o espaço for insuficiente, o cliente aborta a transferência e responde com um frame de erro (tipo 15).

Caso haja espaço suficiente, o cliente abre o arquivo para escrita binária, envia um novo ACK e inicia o processo de recepção dos dados. A cada bloco recebido, os dados são gravados no disco e um novo ACK é enviado. O processo se encerra ao receber o frame de tipo 9, indicando o fim do arquivo. Um processo filho é criado para abrir automaticamente o tesouro com o aplicativo padrão do sistema.

Esse mecanismo garante que a transferência ocorra apenas se for segura e que cada parte do arquivo seja confirmada, mantendo a confiabilidade do protocolo proposto.

6 Conclusão

O uso controlado de timeouts com backoff exponencial proporciona maior confiabilidade na comunicação entre cliente e servidor, especialmente em redes com perdas. Além disso, a definição clara dos tipos de mensagem permite uma semântica rica no protocolo. As mensagens dos tipos 3 e 14 demonstram como o protocolo pode ser estendido para transmitir comandos de controle específicos do jogo.

Referências

[1] Albini, L.; Todt, E. (2025). *Trabalho T1 – Redes 1*. Universidade Federal do Paraná.