Relatório do Trabalho 1 da Disciplina de Redes de Computadores 1

Fernando de Barros Castro[1], **Ruibin Mei**[1]

[1]Universidade Federal do Paraná

{fbc23,rm23}@inf.ufpr.br

Resumo. *O trabalho consiste na implementação de um jogo conhecido como "Caça ao Tesouro", utilizando protocolos de rede de nível 2 (camada de enlace) na comunicação entre dois computadores, no modelo cliente-servidor, baseado na utilização de raw sockets e inspirado no Kermit Protocol para envio de mensagens curtas e transferência de arquivos. Este relatório tem como objetivo descrever as decisões tomadas e as implementações realizadas no desenvolvimento do trabalho.*

1. Introdução

O cliente e o servidor devem ser executados em máquinas diferentes, utilizando os seguintes comandos:

- **sudo ./bin/src/client**
- **sudo ./bin/src/server**

O objetivo do cliente é exibir na tela o tabuleiro e a posição atual do jogador, além de informar ao usuário sempre que um tesouro for encontrado. Dessa forma, o cliente funciona como uma interface de comunicação entre o usuário e o servidor.

Já o servidor tem como objetivo receber as solicitações enviadas pelo cliente e responder informando se a ação solicitada é possível ou, caso um tesouro tenha sido encontrado, enviá-lo ao cliente.

GitHub: `https://github.com/rm1003/treasure-hunt`

2. Decisões

Optou-se pela linguagem C++ para a implementação do trabalho, principalmente por suas características de Programação Orientada a Objetos, que auxiliam na modelagem e no projeto da aplicação.

2.1. Hierarquia dos arquivos

A divisão atual dos diretórios foi pensada para abstrair ao máximo os detalhes da rede, de forma que tanto o cliente quanto o servidor não tenham conhecimento direto do que ocorre na comunicação em nível de rede. Seguindo essa lógica, o diretório *libs* contém a abstração da classe `NetworkProtocol`, que fornece APIs utilizadas tanto pelo cliente quanto pelo servidor, ou seja, são funções chamadas por ambos para envio e recebimento de pacotes. Todo o controle da rede e de pacotes mandados pela rede estão sendo manipulados pela `NetworkProtocol` utilizando as funções do `PackageHandler` que por consequência as funções do `RawSocket`. Dessa forma é possível abstrair a rede, e o cliente ou servidor conseguem obter os dados dos pacotes e mandar pacotes sem se preocupar com operações na rede. A hierarquia esta separado nos seguintes diretórios (tabela 1):

2.1.1. Estrutura de Arquivos

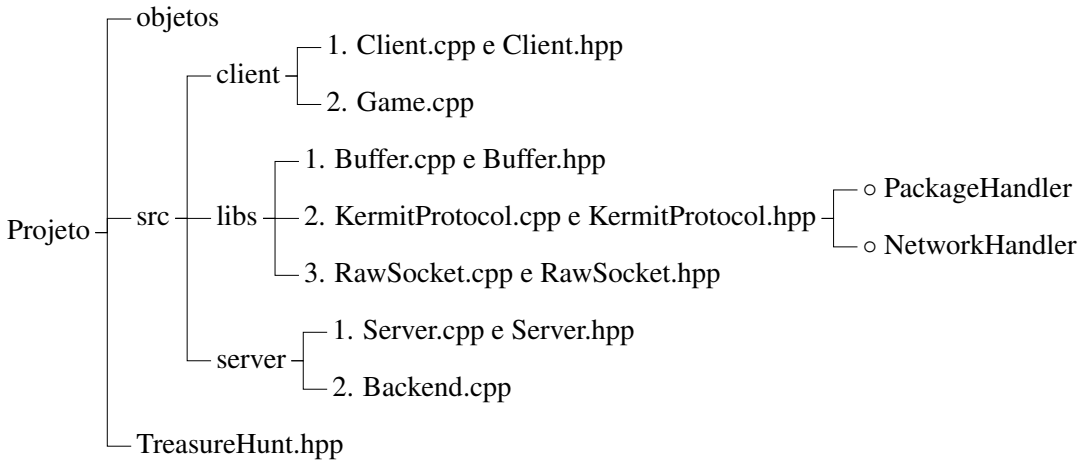

Figura 1. Estrutura hierárquica do projeto

2.1.2. Descrição dos Componentes

objetos : Contém os arquivos de dados dos tesouros do jogo.
src : Diretório raiz contendo todo o código fonte do projeto.
Client : Contém a classe do cliente.

- **Client.cpp/hpp**: Contendo toda a lógica do jogo no lado do cliente.
- **Game.cpp**: Arquivo principal do cliente, contendo a função `main()` e chamada das funções do `Client.hpp` para manipulação da interface.

libs : Bibliotecas compartilhadas contendo código fonte das APIs utilizados pelo cliente e servidor.

- **Buffer.cpp/hpp**: Contém a classe de manipulação de *buffer* para o envio de arquivos, e é utilizado pelo servidor para mandar tesouro para o cliente.
- **KermitProtocol.cpp/hpp**: Contém a classe do protocolo inspirado no *Kermit Protocol*, na qual possui todas as funções de manipulação do protocolo. É utilizado por ambos (cliente e servidor) e suas funções são chamadas via `NetworkHandler`.
  - **PackageHandler**: Manipulação de pacotes, faz o ”envelope”e todas operações de verificação de Checksum e adição/remoção do `0xff` para os casos de bytes `0x88` e `0x81`. Além disso, indica se o pacote é válido ou não e entre outras resultados como time-out, mensagem repetida.
  - **NetworkHandler**: Fornece a API de comunicação para envio e recebimento de mensagens.
- **RawSocket.cpp/hpp**: Contém as funções de mais baixo nível, apenas manda e recebe o pacote vindo da rede. O acesso é restrito à classe `PackageHandler`.

server : Contém a classe do servidor.

- **Server.cpp/hpp**: Contendo toda a lógica do jogo no lado do servidor.
- **Backend.cpp**: Arquivo principal do servidor, contendo a função `main()` e chamada das funções do `Server.hpp` para manipulação do servidor.

TreasureHunt.hpp : Contém as estruturas de manipulação do *grid*.