JOGO DE ADIVINHAÇÃO COM IMPLEMENTAÇÃO DE CLIENTE-SERVIDOR

Daniel Luiz de Araújo Marques, Lucas Ferreira dos Santos.
Outubro, 2023.

1. Introdução

O objetivo deste trabalho é implementar um jogo multiusuário de turnos, através dos protocolos da camada de aplicação e transporte. A implementação do código foi feita em Linguagem C e o protocolo utilizado para a troca de pacotes entre as máquinas foi o TCP. A porta padrão utilizada pela aplicação é a porta 8080.

Foi-se implementado um jogo de adivinhação em turnos, onde é gerado um número aleatório de 1 até 15 e os jogadores devem adivinhar qual é este número. O jogo é composto por 2 jogadores e cada um tem até três chances para acertar o número gerado.

2. Implementação

O servidor é o responsável pela organização da partida e por toda a comunicação durante ela. Os jogadores só poderão fazer sua jogada quando houver a liberação do servidor. Por exemplo: o servidor irá enviar uma mensagem para um dos jogadores avisando que é sua vez de jogar, em paralelo, envia uma mensagem para o outro jogador indicando que o outro jogador está jogando. Quando o jogador é autorizado a jogar, o servidor realiza as verificações necessárias (se o número jogado foi o correto; se foi maior que o número gerado; se foi menor; etc), após isso, envia mensagens de atualização para os jogadores, para mantê-los atualizados.

2.1 Cliente

Módulo responsável por receber, processar e enviar mensagens para o servidor. Contém apenas a função main, com todas as operações a serem realizadas, algumas delas são:

- **Criação do Socket do Cliente:**
  - Ele especifica que o cliente utilizará uma comunicação TCP e associa o socket com a variável ‘sock’.
- **Configuração do Endereço do servidor:**
  - A estrutura ‘serv_addr’ é configurada com informações sobre o servidor, incluindo a família de endereços (IPv4) e a porta na qual o servidor está ouvindo. A função ‘inet_pton’ é usada para converter o

endereço IP do servidor em um formato binário adequado para o armazenamento na estrutura ‘serv_addr’.

- **Conexão com o Servidor:**
  - A função ‘connect()’ é usada para estabelecer uma conexão com o servidor. Se a conexão falhar, uma mensagem de erro é exibida.
- **Leitura de Mensagens do Servidor:**
  - O cliente lê as mensagens do servidor usando a função ‘read()’. O servidor envia mensagens que indicam quando é a vez do jogador ou fornece feedbacks sobre as tentativas do jogador.
- **Entrada do Jogador:**
  - Quando o servidor indica que é a vez do jogador, o cliente solicita ao jogador para digitar um número entre 1 e 15. Isso é feito através da entrada do teclado usando ‘fgets()’ e, em seguida, o número é enviado ao servidor usando ‘send()’.
- **Limpeza do Buffer:**
  - O buffer é limpo após o envio de um palpite bem-sucedido para evitar confusões com mensagens futuras.
- **Verificação do Resultado:**
  - O cliente lê a próxima mensagem do servidor e verifica se o jogador adivinhou corretamente o número, encerrando o loop se o jogador acertou ou se o jogo terminou.

2.2 Servidor

Módulo responsável por gerenciar toda a comunicação entre os jogadores. Contém três funções principais, dentre elas:

- **void enviar_mensagem:** essa função recebe como parâmetro o socket e a mensagem. É a responsável por enviar mensagens para os clientes.
- **void receber_tentativa:** essa função recebe como parâmetro o socket e o buffer. É responsável por receber as tentativas dos jogadores.
- **int main:** função mais extensa do módulo servidor. Contém as seguintes operações:
  - **Declaração de Variáveis:**
    - Armazenamento das variáveis dos sockets do servidor e dos clientes, também contém a estrutura ‘sockaddr_in’ para armazenar detalhes de endereço. Além disso, contém variáveis para os buffers que armazenam mensagens recebidas dos jogadores e uma variável ‘mensagem’ que envia mensagens para os clientes.
  - **Criação do socket do servidor:**
    - Cria um socket do servidor usando a função ‘socket()’. Utilizando o protocolo ‘SOCK_STREAM’ para comunicação,

que é apropriado para a comunicação TCP. O socket é associado à variável ‘socket_server’.

- **Configuração do endereço do servidor:**
  - A estrutura ‘address’ é configurada com informações sobre o endereço do servidor. Ela é preenchida com IPv4, o endereço de IP e a porta do servidor que irá escutar.
- **Vinculação do socket à porta:**
  - O socket do servidor é vinculado à porta especificada através da função ‘bind()’. Isso permite com que o servidor escute conexões na porta especificada.
- **Colocando o servidor para escutar**
  - O servidor é colocado para escutar conexões através da funções ‘listen()’. Ele pode aceitar até três conexões em fila.
- **Aceitando conexões de jogadores:**
  - O servidor entra em um loop onde aguarda a conexão dos jogadores. Através da função ‘accept()’ ele aceita as conexões, assim preenchendo os sockets.
- **Implementação do Jogo:**
  - Após a conexão dos jogadores, o servidor irá gerar um número aleatório que os jogadores tentarão adivinhar. Quem começará a partida é definido aleatoriamente.
  - O jogo consiste em loop onde os jogadores alternam suas tentativas de adivinhar o número gerado. Cada jogador recebe uma mensagem informando se é a sua vez de jogar ou a vez do adversário. Eles enviam sua tentativa e recebem o feedback se o número está correto ou se é maior ou menor que o número gerado.
  - O jogo acaba quando as tentativas de ambos jogadores acabam, ou quando o número é adivinhado.

3. Conclusões

Este projeto exemplifica a criação de uma aplicação de rede que permite a interação entre cliente e servidor. Alguns dos pontos a serem notados neste projeto são:

- **Comunicação cliente e servidor:** foi implementada com sucesso através do uso de sockets, permitindo que os jogadores participem do jogo e compartilhem informações.
- **Lógica de jogo baseada em turnos:** foi desenvolvida corretamente, permitindo que os jogadores consigam fazer suas tentativas de jogadas em turnos e recebam os feedbacks sobre as mesmas.

- **Feedbacks aos Jogadores:** foram implementadas mensagens informativas e feedbacks para os jogadores, tornando o jogo mais interativo e envolvente.
- **Tratamento de Exceções e Erros:** foram implementados para assegurar que o jogo funcione de maneira consistente, mesmo quando haja entradas inválidas ou problemas de comunicação.
- **Encerramento Limpo:** o jogo fecha de maneira ordenada, fechando os sockets e liberando recursos quando o mesmo chega ao fim.

Vale ressaltar que este projeto é apenas um exemplo básico. Logo, existem inúmeras maneiras de aprimorá-lo e expandi-lo, como, por exemplo: adicionando uma interface, autenticação de jogadores, aumento de número de jogadores simultâneos, um chat para os jogadores se comunicarem entre si e tratamentos mais avançados de erros.

Referências

[1] Jim Kurose and Keith W. Ross. Redes de Computadores e a Internet 5ª edição, 2009.
[2] Rafael Sachetto. Código Cliente TCP simples.
[3] Thalisson de Almeida. Desenvolvimento de jogos em rede, 2017. Disponível em: https://imasters.com.br/desenvolvimento/desenvolvimento-de-jogos-em-rede-parte-01.
[4] Kennedy Tedesco. Uma introdução a TCP, UDP e sockets, 2019. Disponível em:https://www.treinaweb.com.br/blog/uma-introducao-a-tcp-udp-e-sockets.