Universidade Federal de São João del Rei - UFSJ

Trabalho Prático Redes de Computadores

Jogo de Truco

Carlos Magno

Lucas Geraldo

São João del Rei/Maio -2017

Sumário

1 Introdução

O presente trabalho tem como objetivo exercitar os conceitos básicos de programação em rede e de redes de computadores, por este motivo o mesmo se baseia na criação de uma aplicação que utiliza a API de Sockets e o conceito de cliente servidor para permitir uma comunicação entre diferentes processos que podem estar sendo executados em máquinas distintas. Esta documentação esta focada na descrição da solução para o problema apresentado, portanto alguns conceitos de redes podem não ser abordados.

1.1 O problema

Este primeiro trabalho tem como finalidade implementar um protocolo da camada de aplicação para possibilitar a execução e comunicação entre processos distintos que estarão executando um jogo de truco. Portanto, o mesmo tem como o seu principio básico criar uma aplicação que utiliza protocolos da camada de aplicação, transporte e das demais camadas para possibilitar o funcionamento de uma jogo de turnos em rede.

1.2 Api Sockets

A API Socket foi criada para facilitar o envio de mensagem em redes sobre os protocolos da camada de transporte e de rede em sistemas BSD UNIX. Porém posteriormente a mesma foi adota como padrão para comunicação entre máquinas.

1.3 Regras Consideradas

Para facilitar a implementação do jogo de truco, foi considerada uma combinação das regras do Truco Mineiro com as regras do Truco paulista, ficando definido que o sistema de pontuação adotado seria o do truco paulista e as demais regras seriam originadas do truco mineiro. Sendo assim, as manilhas do jogo são fixas, e a pontuação segue o modelo de truco valendo 3 pontos.

Resumo das regras [2]:

- **Manilhas**: São fixas. Da mais forte para mais fraca: 4 de paus/7 de Copas/Ás de Espadas/7 de ouro.
- **Ordem das Cartas**: 3, 2, A, K, J, Q, 7, 6, 5, 4.

2 Desenvolvimento

2.1 Visão geral da implementação

O trabalho considerou somente um jogo de truco com quatro jogadores, portanto a implementação está direcionada somente para o funcionamento de um jogo de duplas divididas em duas

equipes. Foi implementado um servidor rodando na linguagem C e um cliente com interface gráfica em python. O cliente e o servidor se comunicam por meio de uma string de 32 bytes que será explicada detalhadamente no tópico de protocolo.

2.2 O servidor

Como determinado na especificação do trabalho, o servidor foi implementando na linguagem C. O mesmo foi desenvolvido para funcionar em turnos com quatro jogadores, portanto o seu funcionamento pode ser divido em algumas etapas distintas, que podem ser descritas como uma maquina de estados. Podemos destacar as seguintes etapas:

- **Conexão**: Nessa etapa é realizada a conexão com os clientes, atribuindo as equipes e um id fixo para cada jogador. Somente ocorre uma transição para o próximo estado quando todos clientes já estiverem conectados.
- **Embaralhar cartas**: Nesta etapa é realizado o embaralhamento das cartas a serem distribuídas para cada jogador.
- **Comunicação inicial e Distribuição de cartas**: Nessa etapa ocorre uma comunicação inicial, distribuindo as cartas e os detalhes iniciais de cada jogador(id, equipe).
- **Inicio do Jogo**: Nesta etapa todos os jogadores ficam bloqueados até receberem a mensagem indicando que é sua vez de jogar. Quando o jogador realiza uma ação, ocorre uma transição de estado indicando a vez do próximo jogador.
- **Analise do Vencedor de Turno**: Após os quatro jogadores jogarem suas cartas, ocorre a análise do vencedor do turno, onde é feita a verificação de qual jogador jogou a carta mais alta dentre as que estão na mesa. O jogador vencedor é escolhido para iniciar o próximo turno.
- **Analise do Vencedor da Rodada**: Após os quatro jogadores jogarem todas as suas cartas, ou se uma dupla fugiu de um pedido de truco, é feita uma análise de qual equipe venceu a rodada. Obtida essa informação, o servidor modifica o trecho do placar correspondente à equipe vencedora da rodada. O jogo é encerrado quando uma pontuação maior ou igual a 12 pontos é alcançada por uma das duas equipes.
- **BroadCast**: Ao final da ação de cada jogador é disparada uma mensagem para todos os clientes, para que cada um possa atualizar o estado do jogo.

2.2.1 Métodos Principais

- **embaralhar**: Atribui nomes e valores de cada carta em uma lista, para facilitar a distribuição aleatória das mesmas.

- **distribuir**: Distribui as cartas de maneira aleatória entre os quatro clientes.
- **setToken**: Altera o trecho delimitado de uma string.
- **getToken**: Retorna um trecho delimitado de uma string.
- **entregar**: Essa função dispara uma mensagem padrão para todos os clientes, atribuindo a cada um suas cartas (mão).
- **vencTurno**: Retorna o id do jogador que venceu o turno.
- **vencRodada**: Retorna 1 se uma equipe já venceu a rodada corrente. Se "sim"a função atualiza o placar do jogo.
- **vencJogo**: Retorna 1 se alguma das equipe tenha atingido 12 pontos ou mais.
- **truco**: Aumenta o valor da rodada.
- **broadcast**: Dispara uma mensagem atualizando o estado de todos os clientes. Essa função é acionada ao final de cada ação de um jogador.

2.3 O cliente

O cliente foi escrito em python utilizando a biblioteca pygame, o mesmo somente renderiza a tela e realiza a atualização de poucos campos da mensagem vinda do servidor. O cliente possui duas threads, [1]permitindo que um mesmo processo realize diferentes tarefas simultaneamente. A primeira thread ficou encarregada de desenhar e atualizar a tela, e a segunda thread ficou responsável por monitorar as mensagens enviadas pelo servidor e repassar as mesmas para a primeira thread. Portanto a segunda thread tem como sua principal função ouvir a porta que está sendo usada na conexão com o servidor.

A interface é atualizada a medida que servidor realiza uma transição de estados. O funcionamento do cliente com interface gráfica pode ser resumido nos seguintes estados:

- **Conexão**: Realiza a conexão com servidor.
- **Recebe dados iniciais**: Recebe uma mensagem com o id, a equipe e as cartas.
- **Permissão de jogo**: Após o cliente receber suas cartas, ele recebe um broadcast atualizando suas permissões de jogo.
- **Atualização da mesa e placar**: A cada jogada os clientes recebem uma mensagem geral atualizando o estado da mesa, placar e rodada.
- **Sua vez de Jogar**: Quando o cliente recebe autorização para jogar, ele pode solicitar truco ou jogar uma carta e terminar a sua jogada.

2.3.1 Métodos Principais

O cliente está dividido em módulos, portanto vamos destacar apenas os módulos e métodos principais.

- **Modulo de Conexão**: Este arquivo contem as funções de conexão, envio e recepção de mensagens.
- **Modulo de GUI**: Nesse modulo estão definidas as funções para desenhar a janela gráfica.
- **Modulo Principal**: Esse modulo utiliza os modulos Gui e Conexao para renderizar o jogo e permitir as interações com usuário.
- **conectar**: Realiza a conexão com o servidor de acordo com as informações contidas no arquivo de configuração config.
- **ler_socket**: Recebe uma mensagem do servidor.
- **envia_mensagem**: Dispara uma mensagem para o servidor.
- **encerrar_conexao**: Encerra a conexão com o servidor.
- **iniciar**: Define as configurações da tela inicial do jogo. Carrega a imagem de fundo.
- **tela _ adrao**: Define as configurações da tela principal do jogo.
- **carrega_ cartas**: Carrega a imagem das cartas que estão na mão do jogador.
- **update_ card**: Carrega a imagem das cartas.
- **jogar_cartas**: Renderiza na mesa a carta jogada pelo jogador.
- **recebe_cartas**: Recebe as cartas do servidor.
- **Verifica _resposta servidor**: Essa função fica a todo momento monitorando e esperando mensagens vindas do servidor. A mesma é utilizada na segunda thread.
- **Processa resposta**: Processa a string obtida pela função verifica reposta servidor.
- **Renderiza mesa**: Utiliza as funções do módulo de interface para renderizar as cartas na mesa.
- **Envia carta**: Dispara uma menagem para o servidor indicando a carta jogada.
- **prepara mensagem**: Prepara mensagem a ser enviada para o servidor.

- **Main**: Utiliza o módulo de conexão e de interface para desenhar a tela e permitir a interação com o jogador, essa função é utilizada pela primeira thread e tem como objetivo desenhar e manter atualizado os objetos da tela.
- **Funções pygame**: Além das funções apresentadas, o módulo principal e o módulo GUI utilizam as funções da biblioteca pygame [3], biblioteca de código aberto para criar aplicações multimídia. Neste software as funções da pygame são utilizadas para desenhar os objetos da tela bem como para carregar as imagens das cartas.

3 O protocolo

O protocolo implementado é baseado na comunicação utilizando strings de tamanho fixo com 32 bytes. O presente protocolo é utilizado na camada de aplicação rodando em cima do protocolo de transporte TCP.

3.1 Descrição

Com o intuito de facilitar a comunicação entre cliente e servidor optou-se por utilizar uma string de tamanho fixo, com trechos de bytes para representar cada estado do servidor e da partida.

3.2 Etapas para estabelecer conexão

O servidor aguarda os quatro clientes se conectarem para disparar as primeiras mensagens. Quando os clientes estão conectados é disparada uma mensagem para cada um informando o seu id, equipe e suas cartas iniciais. Posteriormente é disparada uma mensagem indicando a vez do primeiro jogador que irá jogar.

Exemplo de primeira mensagem: **0a00qo3c4o00000000000000000000**

Jogador ID: 0, Equipe: a, Cartas na mão: qo3c4o (Q ouro, 3 copas, 4 ouro).

3.3 Formato da Mensagem

O protocolo criado define que uma mensagem deve possuir 32 bytes, sendo que para o cliente e o servidor se comunicarem corretamente, a mesma deve sempre possuir esse tamanho e seguir o padrão apresentado na tabela 1 .

Padrão da String: 00000000000000000000000000000000

Tamanho [0:31] 32 bytes.

Tabela 1: Formato da mensagem do protocolo

Posição	Nome	Valores	Tamanho	Descrição
[0:1]	id_Jogador	0/1/2/3	1 byte	Id do jogador.
[1:2]	equipe	a/b	1 byte	Marcador da equipe.
[2:3]	vez	0/1	1 byte	Indicador da vez.
[3:4]	rodada	0/1/2/3	1 byte	Indicador da rodada.
[4:10]	mao	4pkc4p	6 bytes	Cartas recebidas.
[10:14]	placar_jogo	0000	4 bytes	Placar do jogo.
[14:17]	placar_rodada	aba	3 bytes	Mostra o placar da rodada.
[17:19]	valor_rodada	00	2 bytes	Valor atual da rodada.
[19:20]	question	0/1	1 byte	Indicador de Truco. Proposta de truco.
[20:21]	equipe_question	a/b	1 byte	Indica a equipe que pode pedir truco.
[21:22]	resposta_question	0/1	1 byte	Resposta da proposta de truco.
[22:30]	mesa	4p3c2a7o	8 bytes	Mostra as cartas que estão na mesa.
[30:31]	virada	0/1	1 byte	Indica que a carta jogada está virada.

3.4 Atualização dos Campos

O cliente somente atualiza poucos campos da mensagem recebida do servidor, portanto grande parte dos campos somente podem ser atualizados pelo servidor. Podemos destacar que o cliente atualiza somente os campos question, resposta_question, mesa e virada. Os demais campos destacados na tabela 1 são atualizados pelo servidor.

3.5 Funcionamento do protocolo

O funcionamento da aplicação e a utilização do protocolo podem ser exemplificados na figura 1 e nos seguintes estados:

- **Estabelecer conexão**: Aguarda a conexão dos clientes.
- **Broadcast Inicial**: Envia mensagem para todos os clientes com os detalhes iniciais.
- **Estado do jogo**: Envia uma mensagem indicando a vez do cliente a jogar. O cliente atualiza os campos que ele pode editar e retorna a mensagem para o servidor.
- **Broadcast**: O servidor dispara um mensagem, atualizando o estado de todos os clientes.
- **Repete estado**: Após o broadcast é disparada uma nova mensagem indicando o próximo jogador a jogar, volta ao estado do jogo.

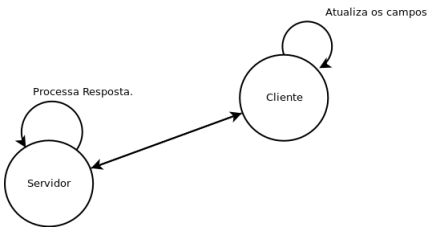

Figura 1: Cliente Servidor

4 Conclusão

Com a realização deste trabalho, o problema proposto foi resolvido, bem como foi implementado um protocolo da camada de aplicação que viabilizou o funcionamento do jogo em rede. O formato de mensagem que o protocolo utiliza é simples, e se adaptou bem às necessidades de comunicação entre cliente e servidor. Porém pelo fato de possuir um tamanho fixo, a mensagem pode gerar um tráfego de rede elevado mesmo enviando somente dados básicos.

O cliente com interface gráfica apresentou um alto consumo de processamento ao executar mais de três proce na mesma máquina, porém o mesmo se mostrou adequado para ser utilizada em hospedeiros finais distintos ou no mesmo hospedeiro, rodando apenas 1 cliente com interface e os demais utilizando um cliente terminal (que foi implementado para auxilio nos testes) ou a ferramenta netcat (nc localhost 5000).

Podemos destacar, que a principal dificuldade encontrada neste trabalho foi a definição do protocolo de comunicação, bem como conseguir abstrair o problema de maneira a observar cada uma das suas particularidades.

Pelo que foi apresentando nesse trabalho podemos concluir que o problema proposto foi resolvido, além disso foi possível enriquecer nosso conhecimento sobre programação em rede e as suas características essências.

Referências

[1] Butenhof, D. R. *Programming with POSIX threads*. Addison-Wesley Professional, 1997.

[2] megajogos.com.br. Como jogar truco mineiro, may 2017.

[3] pygame. Pygame, may 2017.