Universidade Federal de São João del Rei
Ciência da Computação
Redes de Computadores

Truco em Rede local

Alunos: Rafael Campos, Welton Santos, Ygor Magalhães

Professor: Rafael Sachetto

São João del rei
Outubro, 2018

Conteúdo

1 Introdução

A documentação apresentada tem como objetivo trabalhar os conceitos de programação em redes de computadores, baseando-se na criação de uma aplicação que utiliza API de Sockets no conceito cliente servidor, permitindo a comunicam entre diferentes processos que podem ser executados em maquinas diferentes. Com o foco em apresentar uma solução para o problema proposto.

1.1 Problema Proposto

Esta proposta consiste em desenvolver um jogo de truco baseado em turnos, com a finalidade de implementar um protocolo de camada de aplicação para criar a comunicação e execução entre processos em máquinas distintas. Toda transmissão será feita entre cliente servidor, com o objetivo de criar uma aplicação que utilize o protocolo da camada de transporte TCP, para realizar entraga confiável de dados.

1.1.1 Sockets e TCP/IP

1. TCP/IP
   cada ponto final é identificado por uma tupla: (porta TCP, endereço IP)
   a conexão entre dois pontos finais é identificada pelo par [(IP, porta)origem, (IP, porta)destino]

2. Em sistemas Unix todo fluxo de E/S é identificado por um descritor
   socket mapeia um descritor para um ponto final
   através da conexão entre sockets é possível conectar pontos finais e fazer operações de E/S

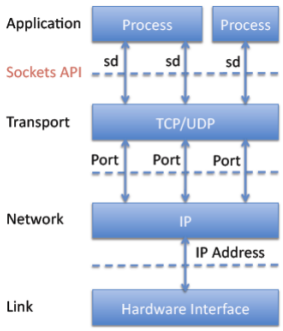

Figura 1: Protocolo.

1.2 API Sockets

Uma API de soquetes (API sockets) é uma interface de programação de aplicativos (API), normalmente fornecida pelo sistema operacional, que permite que softwares da camada de aplicação controlem e usem soquetes de rede. APIs de soquete de Internet geralmente são baseados no padrão Berkeley sockets. Para este trabalho utilizou-se a API de soquetes em C provinda pelo sistema Ubuntu (GNU/Linux) com a finalidade de desenvolver um protocolo na camada de aplicação sem a necessidade de se preocupar com detalhes da camada de transporte e demais abaixo existentes na pilha TCP/IP.

Todo sistema operacional atual tem sua implementação do padrão TCP/IP, tanto para uso interno, como para o uso em software através de uma API. A API vai nos permitir criar um socket, envia dados pela rede, receber dados do mundo externo, "escutar" em uma porta à espera de conexões, etc. Cada sistema operacional que suporta rede tem algum tipo de Stack de rede. Linux não é exceção. A pilha de rede é o que permite que os aplicativos sejam capazes de acessar a rede por meio de um dispositivo de rede física. Dispositivos de rede pode ser modems, cable modems, ISDN, dispositivos Wi-Fi, placas Ethernet, placas Token Ring dentre outros.

1.3 Regras de Truco Consideradas

Para Implementação do trabalho foram considerado as regras do Truco Mineiro com o objetivo de fazer 12 pontos para alcançar a vitoria. As cartas são alocadas vetor de Cartas (estrutura dedicada), de forma que a sua hierarquia é definida por um atributo inteiro presente dentro da estrutura que define o "poder"de cada carta. Desta forma o "Zap" (4 de paus) possui o maior valor "14", seguido pelo "7 copas" (7 de coração) com valor 13, "Espadilha" (Az de espada) com valor 12 e por final "pica fumo"(7 de ouros) completando o conjunto de cartas denominado manilha. Posteriormente, segue as demais cartas, sem destinção de nipe na atribuição de seu valor, deste modo a carta 3 de paus possui valor equivalente a 3 de ouros.
Ordem das Cartas:

1. 4 de paus, valor 14.
2. 7 de Copas, valor 13.
3. Às de Espadas, valor 12.
4. 7 de ouro, valor 11.
5. 3, valor 10.
6. 2, valor 9.
7. A, valor 8.

8. K, valor 7.
9. J, valor 6.
10. Q, valor 5.
11. 7, valor 4.
12. 6, valor 3.
13. 5, valor 2.
14. 4, valor 1.

O programa não possui interface grafica sendo executado no terminal, as cartas são representadas por seu numero e uma letra que representa seu nipe:

1. Ouro: o
2. Copas: c
3. Espada: e
4. Paus: p

Desta forma a carta Rei de Ouros é representada por **ko**.

2 Desenvolvimento

2.1 Visão Geral da Implementação

A implementação deste trabalho está direcionada para o funcionamento de um jogo de truco com suporte para duplas com 1 ou 2 integrantes, com poder de escalabilidade para mais jogadores. A hierarquia adotada para a aplicação foi a tradicional *cliente-servidor*, com um host (servidor) responsável por sicronizar os clientes e direcionar o fluxo do jogo, centralizando maior parte do processamento consigo. Do outro lado encontra-se os clientes (jogadores) que por vez realizam funções simples como formatar saída de informações e permitir a interação do usuário com o jogo enquanto aguardam pelas instruções provindas do mesmo, permanecendo desta forma, insentos de altos gastos de recursos com processamento. Tanto o cliente e servidor foram implementados na linguagem C, com a comunicação entre eles realizada através do envio de bytes em texto plano que serão detalhados no tópico de protocolo.

2.2 Servidor

2.2.1 Sistema de Gerenciamento do Jogo

O servidor é responsável por gerenciar o fluxo do jogo e consequentemente definir qual jogador deve jogar e em qual momento. O sistema de permissão funciona através de "sinais", de modo que, enquanto um jogador faz uma jogada os demais estão a espéra de um sinal do servidor que definirá suas próximas ações. Desta forma qualquer dado inserido por um usuário durante seu perído de espera, é armazenado no buffer e então quando receber um sinal de permissão para jogar o buffer será lido e interpretado.

2.2.2 Fluxo de operações do servidor

Estado de estabelecimento de conexões.

1. Servidor configura estruturas (socket, bind, liste etc...).
2. Entra no estado de espera por conexões (padrão é 4 conexões).
3. Recebe solicitação de conexão.
4. Autentica usuário.
5. Insere o usuário em uma das duplas.
6. Retorna ao passo 2.1 até obter 4 conexões.
7. Servidor distribui as cartas.
8. Envia cartas aos participantes.
9. Envia carta para usuário.

10. Retorna ao passo 3 até que todos os usuários estejam com cartas.

Inicio e decorrer da partida.

1. Servidor sorteia um jogador para iniciar partida.
2. Envia sinal de permissão para jogar para jogador.
3. Aguarda pelo jogador.
4. Recebe sinal do jogador.
5. Interpreta o sinal.
6. Escolhe uma ação (E.: aguardar por uma carta).
7. Atualiza os demais jogadores.
8. Passa vez para o próximo jogador.
9. Retorna ao passo 2 até que o jogo termine.

2.2.3 Principais Funções

Neste tópico será discutido o papel das principais funções do módulo cliente. ***escutaSolicitacao:*** Esta função é invocada logo ao inicio da aplicação no servidor e faz todo o processo de estabelecimento de conexão (autenticação, inserção em dupla, atribuição de id etc..) com os usuários, trasnmitindo o controle para a função de *controleJogo* após todos as duplas estarem completas.
autenticaUsuarios: Invocada dentro da função *escutaSolicitacao* esta função é disparada por meio de threads, para que desta forma, seja possível aos usuários logarem no sistema de forma pararela.
controleJogo: Consiste na principal função do servidor, nesta função a sicronização e controle do fluxo do jogo é realizado. Através do envio e decodificação de sinais esta função informar aos jogadores qual o momento de realizar suas jogadas e de mesma forma os mantem atualizados sobre os movimentos dos demais jogadores.
fechaConexoes: Após a partida ser incerrada ou o servidor por algum motivo sazonal tenha que encerrar seu funcionamento, esta função é invocada e fecha os sockets abertos para escuta dos clientes.
controleJogo: A partir desta função o fluxo do jogo é controlado.Os sinais de bloqueio e de permissão para jogar são enviados para os players. Além de toda contabilização de pontos e também do exercimento das regras do jogo.
enviarCartas: Envia as cartas dos jogadores do servidor para o cliente.
enviarMesa: Envia as cartas que estão na mesa para os jogadores.
enviarAnuncioAumentoAposta: Envia sinal de aumento de aposta para os jogadores. ***enviarAnuncioAceitaAposta:*** Envia que a aposta foi aceita. ***enviarValorRodada:*** Envia a pontuação da rodada. ***enviarResultado:*** Exibe o resultado para os jogadores. ***enviarBloqueio:*** . ***enviarSinal:*** Envia sinal .

2.3 Clientes

2.3.1 Fluxo de operações do cliente

Processo de estabelecimento de conexão.

1. Jogador acessa o server.
2. Inicia processo de conexão (abre socket, se autentica, entra em uma dupla, etc...) com o servidor.
3. Aguarda o servidor iniciar a partida.

Fluxo de operação ao iniciar a partida.

1. Jogador recebe catas.
2. Aguarda por suas cartas.
3. Recebe cartas do servidor.
4. Aguarda pelo início da partida.
5. Espera sua vez de jogar, enquanto visualiza as jogadas dos demais jogadores.
6. Recebe sinal do servidor para proceder com sua jogada.
7. Escolhe uma ação (jogar um carta, aumentar aposta, recusar aumentos de aposta).
8. Retorna ao passo 1 até que o jogo termine.

2.3.2 Principais Funções

abreConexao: Funcionando em sicronia com a função escutaSolicitação operante no servidor, esta função abre conexão com o servidor e oferece ao jogador mecanismos de interação para que o mesmo consiga se autenticar e ser inserido em uma das duplas. Após o final de estabelecimento de conexão, a função é encerrada e retorna na estrutura do jogador o valor do socket aberto *file descriptor* utilizado na autenticação.
decodificador: Trabalhando paralelamente a função controleJogo presente no servidor, esta função também realiza trabalho parecido. Dentro desta função o controle do fluxo de operações no cliente é realizado. Através de um menu de decodificação de códigos as mensagens do servidor são interpretadas e então ações são tomadas de acordo com o código recebido, trabalhando sicronamente com o servidor e demais jogadores.
jogar: a partir desta função o jogador é capaz de enviar uma carta ao servidor. Primeiramente esta função envia o código **"00"** e então o servidor após interpretar o código entra em estado de espera e aguarda pelo envio da carta pelo jogador.

receberMesa: Recebe as cartas na mesa atraves da conexão do servidor enviadas anteriomente pelo jogador.

3 Protocolo

3.1 Descrição

A principal ideia por trás da arquitetura cliente-servidor é estruturar o sistema como um conjunto de processos cooperativos que oferecem serviços aos processos de usuário. No modelo apresentado a comunicação entre processos se dá exclusivamente através da troca de mensagens baseado na comunicação utilizando strings. O presente protocolo é implementado na camada de aplicação, e opera sobre o protocolo da camada de transporte TCP. Adotado como modelo adequado para implementação do cliente-servidor para o jogo de truco.

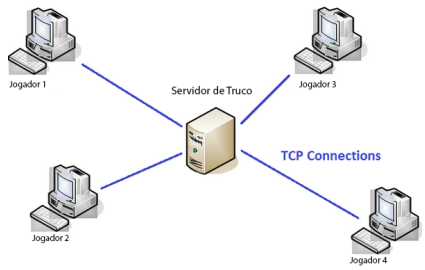

Figura 2: Figura Cliente-Servidor.

3.2 Sistema de Códigos

Como solução para este trabalho, optou-se por utilizar um sistema padronizado com finalidade de sicronizar o servidor e seus clientes durante o decorrer da partida. Os sinais são compostos por 2 bytes, inicialmente abrangendo somente a utilização de números, obtendo então com essa combinação 99 sinais disponíveis. além de ser flexível a adoção de caracteres do alfabeto e outros simbolos, desta forma expandindo a gama de possibilidades para representação. Deste modo cada funcionalidade foi assimilada a um código e através de menus de decodificação os cliente esperam ou realizam ações apartir de um código identificado pelo menu de decodificação.

Sinais enviados pelos clientes.

1. **Código de envio de carta, "00":** ao receber este código do cliente o servidor entra em estado de espera por uma carta do cliente.

2. **Código de solicitação de aumento da aposta, "01":** Ao receber este código o servidor o servidor envia uma mensagem, para o próximo jogador avisando que este esta sendo intimado a aumentar o valor da aposta. Depois de avisar, ele transfere o jogo para o jogador intimado e entra em estado de espera pela resposta.

3. **Código para aceitar aumento de aposta, "02":** ao receber este código de um jogador o servidor sabe que o jogador acaba de aceitar um pedido de aumento de aposta, então o valor da aposta atual é atualizado e enviado aos jogadores. Este código também é utilizado quando uma equipe atinge 10 pontos "mão de 10"e então os jogadores enviam este código para o servidor indicando que aceitaram jogar a mão de 10.

4. **Código para recusa de aumento de aposta, "03":** ao receber este código de um jogador o servidor sabe que o jogador acaba de recusar um pedido de aumento de aposta e a rodada é encerrada e os pontos em aposta são adicionados ao placar da equipe vencedora.

5. **Código para anúncio de aumento de aposta, "04":** este código é enviado pelo server ao jogadores que ao receberem são atualizados que um jogador aceitou uma solicitação de aumento de aposta e a partir desta rodada esta valendo mais. Este código também é utilizado quando uma equipe atinge 10 pontos "mão de 10"e então os jogadores enviam este código para o servidor indicando que decidiram não jogar a mão de 10.

Sinais enviados pelo servidor.

1. **Código de permissão para o jogador realizar sua jogada, 10:** este código é enviado do do server para o jogador. Ao receber este código o jogador sabe que está em seu momento de realizar sua jogada e então o server entra em um estado de espera por outro código.

2. **Código de envio de mesa, "11":** ao receber este código do server jogador entra em estado de espera para receber as cartas da mesa, que seram enviadas a todos os jogadores (broadcast).

3. **Código de envio de cartas, "12":** ao receber este código do server jogador entra em estado de espera para receber as suas cartas e então armazena-las em sua estrutura.

4. **Código para envio do valor da rodada, "13":** ao receber este código do server jogador entra em estado de espera para receber o novo valor da rodada e então formata-lo e apresentar para o usuário.

5. **Código para envio de resultado (vitória ou empate de turno ou rodada): "14":** este código é utilizado pelo servidor tanto para envio de mensagem de vitória quanto de empate. Desta forma ao receber este código o cliente entra em estado de espera, aguardando por uma string com a mensagem configurada pelo servidor.

6. **Código para aviso de mão de 10, "15":** em seguia a uma equipe atingir 10 pontos é enviado pelo servidor este código a mesma. Ao receber este código o jogadores são redirecionados para um menu com as opções de aceitar ou recusar jogar a mão de 10.

7. **Código de Bloqueio de aumento de aposta, "16":** após uma equipe solicitar aumento de aposta, a mesma deve ser impedida de solicitar outro aumento consecutivo, podendo solicitar novamente somente quando a rodada acabar ou caso se a equipe adversária também solicitar aumento de aposta novamente. Para realizar o bloqueio este código é enviado aos jogadores e um variável reponsável por registrar o bloqueio é configurada para tal.

8. **Código de Desbloqueio de auemnto de aposta, "17":** complementar ao código "16", este código é utilizado para alterar o estado de uma equipe de modo que a mesa possa solicitar aumento de aposta caso desejar.

9. **Código de correção para valor da rodada durante a mão de 10: "18":** durante a mão de 10 o valor da rodada é dobrado (valendo 4 pontos) e é através deste código que o servidor atualiza os clientes sobre o novo valor e configura outros detalhes necessários.

10. **Código de final de partida, "30:** ao finalizar a partida o servidor envia este código a todos os jogadores avisando que vai fechar as conexões (sockets de escuta) e encerrar as operações. Portanto ao receber este código o cliente também encerra conexão com o servidor fechando o socket dedicado a escuta.

11. **Código de desistência de jogador, "31:** caso um jogador aborte a partida de forma abrupta, existe um esforço empregado pelo servidor para tentar contornar a situação e então visar os demais jogadores sobre a desistência do jogador. Deste modo este código ao ser interpretado pelo cliente possui efeito parecido ao código *30* citado anteriormente.

A partir deste protocolo, a necessidade da utilização de recursos para sicronização e controle, como mutex e variáveis condicionais foi reduzido consideravelmente, além de oferecer escalabilidade para implementação de novas ações e também praticidade na correção de erros. Como exemplo pode-se citar o caso corriqueiro em que o cliente recebe um código "10", este então sabe que sua vez de jogar e procede respondendo ao servidor com outro código que indica a ação a ser tomada como no caso de enviar uma carta, onde o servidor receberia o código "00"do cliente e então aguardaria pela envio de uma carta.

3.3 Funcionamento do Protoclo

3.3.1 Aplicação

O funcionamento da aplicação e a utilização do protocolo são demonstrados nos seguintes estados:
Estabelecendo conexão: Aguarda a conexão dos clientes.
Broadcast Inicial: É Enviada uma 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 envia uma mensagem, atualizando os estados de todos os clientes.
Repete estado:Após o broadcast envia uma nova mensagem indicando o próximo jogador a jogar, volta ao estado do jogo.

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 as necessidades de comunicação entre cliente e servidor. Podemos destacar, que a principal diculdade encontrada neste trabalho foi a definição do protocolo de comunicação visto que a sicronia entre o servidor e clientes é dinâmica, e novas funcionalidades surgiram conforme o desenvolvimento do projeto o que forçou o protocolo a se adapatar através de expansões constantes em sua complexidade. Este proroblema foi amenizado através do sistema de códigos e menus de decodificação, sem perder a consistencia e performance possibilitando mapear e abstrair o problema de maneira a observar cada uma das suas particularidades com maior clareza. Vale salientar também que a utilização de estruturas de controle de concorrência entre threads como *mutex* e *semáforos*, foi reduzido de forma considerável e desta forma diminuiu a apartição de problemas como dead locks.

5 Referências

(a) Regras do Truco Mineiro , megajogos.com.br , Outubro 2018.

(b) KUROSE, J. F. e ROSS, K. - Redes de Computadores e a Internet - 5ª Ed., Pearson, 2010.

(c) VINICIUS M., Programação de Sockets para Linux, disponivel em: http://softwarelivre.blog.br/2014/05/30/programacao-de-sockets-para-linux/ , acesso em : 20 de outubro 2018