MI Concorrência e Conectividade 2015.2
Primeiro Problema: Roletrando

Allen Hichard, Daniel Andrade

[1]Profª Elisângela Oliveira Carneiro
Universidade Estadual de Feira de Santana – UEFS

Abstract. *This report describes the development process of the first MI - Concurrence and Connectivity of the 2015-2 period. The problem's context is the former Brazilian TV program "Roletrando" The problem's solution contains concepts related to Communication Protocols, Sockets and Threads. Furthermore, we also talk about difficulties encountered, like concurrence problems. In the end, we present the conclusions reached during the development process.*

Resumo. *Esse relatório descreve o processo de desenvolvimento do Problema 1 do MI - Concorrência e Conectividade do período de 2015-2. O problema se trata do desenvolvimento do jogo "Roletrando", antigo programa de sucesso da TV Brasileira. O Jogo deve ser desenvolvido com base na arquitetura cliente-servidor e a solução para o problema aborda conceitos como Protocolos de Comunicação, Sockets e Threads, conexão de computadores em rede e problemas de concorrência. Além disso, são relatados no processo de desenvolvimento as dificuldades encontradas, tais como problemas de concorrência. Ao final, são apresentadas as conclusões obtidas durante o processo de desenvolvimento.*

1. Introdução

A popularidade dos computadores pessoais tem aumentado crescentemente nas últimas décadas. Com tal crescimento, detectou-se a necessidade de criar aplicações que se utilizassem de informações disponíveis em outro computador, e para isso precisava-se conectar tais computadores. Diante dessas necessidades surgiram as Redes de Computadores.

Utilizando-se da literatura disponível na área de Redes de Computadores, o Primeiro Problema do MI Concorrência e Conectividade traz a proposta de implementar o jogo "Roletrando", exibido pelo SBT nas décadas de 1980 e 1990, em redes de computadores. O Problema "Roletrando" tem como objetivo, além de obter um produto final divertido, contribuir para aprendizagem de conceitos iniciais de programação em redes.

A solução para o problema foi construída na Linguagem Java na versão 8, sendo que para a interface gráfica foi utilizado o framework Swing. Para a coleta de dados a serem utilizados no jogo foi elaborado um script na linguagem Ruby versão 2.2.

2. Fundamentação Teórica

Na seção que segue, serão descritos os conceitos utilizados para a resolução do problema proposto. Os itens abordados foram de fundamental importância para a resolução do problema, tendo contribuição direta ou indireta na solução.

2.1. Arquitetura Cliente-Servidor

A enciclopédia Britânica define a Arquitetura Cliente-Servidor como:

> Arquitetura de uma rede de computadores na qual clientes (computadores remotos) solicitam e recebem serviços de um servidor centralizado (computador hospedeiro). Computadores clientes disponibilizam uma interface para possibilitar ao usuário realizar solicitações ao servidor e exibir o resultado assim que o servidor retornar. O servidor, por sua vez, espera pela chegada de novos clientes e os responde. Idealmente, um servidor provê uma interface padronizada e transparente para os clientes, independente do hardware ou software que está provendo o serviço.[Britannica 2015]

Na arquitetura cliente-servidor, o cliente é a parte que interage com o usuário da aplicação. Possuindo uma interface para que o usuário possa requisitar tarefas, o lado cliente é chamando de *front-end* da aplicação. Nesse tipo de arquitetura, os clientes são responsáveis por gerenciar as atividades dos usuários e validar os dados informardos pelos mesmos [Fileto 2006].

O outro componente fundamental desta arquitetura, o Servidor (também chamado de *back-end*), é o responsável ofececer serviços a muitos clientes. Tais serviços podem ser realizados diretamente pelo processo servidor ou por processos escravos, criados para atender os pedidos do cliente, liberando assim o processo mestre do servidor para receber outras solicitações [Fileto 2006].

2.2. Protocolos de Transporte

Um software que se comunica em rede necessita de um meio para transmitir informações. Essa comunicação entre as entidades cliente e servidor é realizada pelos chamados Protocolos de Transporte. Esse tipo de protocolo transfere dados para aplicações executando em um IP hospedeiro. Essa comunicação é realizada através de *portas* [Parziale et al. 2006].

Segundo [Tanenbaum 2003], a internet dois protocolos principais na camada de transporte, um protocolo sem conexões e outro orientado a conexões. Ambos serão abordados a seguir.

2.2.1. User Datagram Protocol

O User Datagram Protocol, ou simplesmente UDP, oferece um meio para as aplicações enviarem informações encapsuladas sem que seja necessário estabelecer uma conexão. Esse protocolo se preocupa somente em oferecer um meio para que os dados sejam enviados. O UDP não garante que os dados chegaram ao seu destino, não controla o fluxo de pacotes ou controle de erros [Tanenbaum 2003]. Por outro lado, o UDP é bastante veloz comparado ao protocolo seguinte.

2.2.2. Transmission Control Protocol

O Transmission Contro Protocol, ou TCP, foi projetado especificamente para oferecer um fluxo de bytes fim a fim confiável numa inter-rede não confiável [Tanenbaum 2003]. Ao

contrário do UDP, o TCP tem dá garantia de envio dos pacotes, possui ferramentas de controle de fluxo de pacotes e erros, além de outras ferramentas como retransmissão de pacotes perdidos. No contexto dos protocolos de transporte, mais segurança e confiabilidade significa mais consumo de recursos, ou seja, o protocolo TCP consome mais banda, memória e processamento das entidades envolvidas do que o UDP.

2.3. Threads

Nos sistemas operacionais antigos, cada processo possuía um único fluxo de controle, ou seja, os programas executavam uma instrução de cada vez. Porém, nos sistemas operacionais modernos cada processo pode dar início a um ou vários subprocessos, chamados de *Threads*. As threads são executadas em paralelo com o processo pai e apresentam as mesmas características de um processo qualquer [Tanenbaum and Machado Filho 1995].

Na programação moderna as Threads são muito úteis quando um processo possui diversas tarefas independentes para realizar. Por exemplo, num servidor web, múltiplas threads permitem que diversas solicitações sejam atendidas simultaneamente sem que seja preciso abrir um processo para cada nova solicitação [Silberschatz et al. 1998].

3. Metodologia

Tradicionalmente, seguindo as primitivas da metodologia do *Problem Based Learning*, o primeiro contato com o problema *Roletrando* foi acompanhado de uma enxurrada de ideias.

3.1. Primeiras Discussões

Por se tratar de um primeiro contato, as discussões iniciais envolveram questões relacionadas a temática do jogo. As primeiras constatações foram que a roleta do jogo deveria ser girada, sorteando um valor, antes de dar a chance o usuário de tentar acertar alguma letra da palavra escondida. Além disso, também foi notado que o jogo deveria exibir a maior pontuação de cada usuário quando o mesmo estivesse jogando e, também, diponibilizasse um ranking com os 3 melhores jogadores. Com uma noção básica do funcionamento do jogo, o foco passou a ser como fazê-lo jogável em uma rede local, tal como exige o problema.

3.2. Questões Iniciais

Visando a implementação do jogo em rede, o centro das discussões passou a ser como transmitir dados entre computadores via rede. Sabe-se que o jogo deve possuir seu servidor e seu cliente rodando em máquinas distintas. A partir desse momento, as questões foram relacionadas a qual protocolo de transporte utilizar e onde centralizar a lógica do jogo, cliente ou servidor.

3.3. Estratégia de Desenvolvimento

Com poucas informações sobre protocolos de transporte e outras informações importantes relacionadas, a dupla resolveu se dedicar primeiramente a implementar o jogo de forma local, e, enquanto isso, procurar tirar as suas dúvidas. Tal decisão foi tomada pelo fato de que o jogo poderia ter seu desenvolvimento iniciado mesmo sem o conhecimento

necessário sobre a área de redes, sendo necessário apenas centralizá-lo na entidade a dupla achar mais coerente depois. Pensando dessa forma, o desenvolvimento do jogo foi iniciado.

O primeiro passo na implementação do jogo foi a descrição de sua *engine*, ou seja, suas funcionalidades básicas. Funcionalidades tais como sortear um número, substituir as letras da palavra por um símbolo ou caractere qualquer, elaboração do ranking do jogo e entre outros. Todas essas funcionalidades foram implementadas sem problemas, dada a experiência adquirida no MI Programação.

3.4. Elaboração do banco de palavras

Para testar a engine construída, e também para o funcionamento efetivo do jogo, é necessário um conjunto de palavras acompanhadas de suas respectivas dicas. Para obter esses dados, a dupla se utilizou do conjunto de palavras disponibilizado em (http://alcor.concordia.ca/ vjorge/Palavras-Cruzadas/Lista-de-Palavras.txt). Para cada palavra, utilizou-se como dica a sua segunda definição, ou a primeira caso a segunda inexistisse, presente no Dicionário Aurélio Online. Todo o procedimento foi automatizado por um script escrito na linguagem de programação Ruby e resultou na obtenção de um banco de 21862 palavras com suas respectivas dicas. Tal quantidade de palavras torna a chance de repetição quase inexistente numa partida e torna o jogo mais dinâmico.

3.5. Escolha do protocolo de transporte

Dispondo dos dois protocolos de transporte mais populares, a dupla preocupou-se em escolher o protocolo que mais se adaptasse a aplicação em desenvolvimento. Segundo [Kurose et al. 2010], o protocolo TCP é um serviço "Orientado para Conexão", sendo assim oferece um nível de garantia na entrega dos pacotes enviados, além de outras vantagens como controle de congestionamento. Apesar dessas vantagens acarretararem em um maior consumo de banda e, também, maior consumo dos recursos físicos do servidor, a dupla julgou que o mesmo é mais vantajoso para a aplicação a ser desenvolvida.

3.6. Centralização da Lógica do Jogo

Após a escolha do protocolo, o passo seguinte foi determinar em qual entidade seria centralizada a lógica do jogo. Analisando ambas as situações, foi notado que haveriam duas possibilidades:

• **Lógica centralizada no Cliente** - Colocando a lógica do jogo no lado do cliente, tem-se a vantagem que o cliente pode se conectar apenas quando for enviar sua pontuação final. Dessa forma, o servidor seria apenas uma central de dados, armazenando a pontuação final de cada cliente. Esse prática apresenta lados positivos, como o baixo consumo de banda e recursos do servidor, uma vez que a conexão com o mesmo e aberta somente para o envio dos dados já processados pelo cliente. Porém, apresenta também lados negativos como a independência demasiada do cliente, resultando numa falta de segurança na qual o cliente pode encerrar o jogo antes que os dados sejam enviados ou ainda, num pior caso, alterar esses dados antes que sejam enviados ao servidor, tornando o jogo fraudulento.

• **Lógica centralizada no Servidor** - Ao centralizar o a lógica do jogo no servidor, o mesmo efetua todas as operações do jogo, seja ela girar a roleta ou informar o

SOLICITAÇÃO	CARACTERE
Enviar nome de usuário	1
Receber Highscore de Usuário	2
Receber palavra	3
Receber dica	4
Verificar se a roleta está disponível	5
Verificar se o round acabou	6
Enviar caractere como tentativa	7
Receber valor acumulado	8
Receber número da rodada	9
Verificar se tem próxima rodada	10
Receber valor da roleta	11
Receber Ranking	12
Desconectar do servidor	13
Separador Padrão	-

Table 1. Protocolo de Comunicação

saldo total após uma rodada. Dessa forma, há um maior tráfego de dados entre o cliente e o servidor, já que todas grande parte das informações que o cliente tem são obtidas através de solicitações feitas ao servidor. Essa opção apresenta uma maior segurança dos dados, uma vez que eles são processados no servidor e com interferência quase nenhuma do cliente. Por outro lado, esse método requer um tráfego maior de informações, característica que faz necessária uma conexão aberta durante tudo o tempo do jogo ou a abertura de uma série de conexões não simultâneas durante a partida. Seja abrindo e fechando uma conexão ou mantendo a mesma aberta, a número maior de requisições acarretará num maior consumo de banda e e recursos do servidor, dada as características do protocolo TCP.

Diante dos fatos evidenciados, a dupla adotou a prática de centralizar a lógica no servidor, optando por uma maior segurança dos dados e, por isso, consumindo mais banda e recursos do servidor.

3.7. Protocolo de transmissão de dados do jogo

Para transmitir os dados entre o cliente e o servidor é necessário um protocolo de comunicação próprio do jogo. Assim como o protocolo HTTP apresenta um código *200* para requisições *GET* efetuada corretamente, por exemplo, o jogo deve possuir um protocolo que permita ao cliente solicitar suas informações ao servidor.

Para criar o protocolo de comunicação, a dupla fez o levantamento de todas as solicitações que o cliente pode fazer ao servidor. Tendo em vista que as informações que representam cada solicitação vão trafegar pela rede, foi determinado que cada solicitação seria representada por um numero inteiro. Tal decisão foi tomada com base no tamanho do tipos de dados primitivos, sendo que um int é o relativamente pequeno e abrange o número necessário de possibilidades requeridos para o protocolo. O resultado pode ser conferido na tabela 1.

Como pode ser visualizado na tabela1, além das operações está definido também

um separador padrão. Tal separador é utilizado para formatar strings de objetos não nativos do Java, tornando assim possível enviar esses objetos em rede. Ao enviá-los diretamente, simplesmente serializando-os, o Java acusa classes incompatíveis no cliente e no servior.

3.8. Comunicação Cliente-Servidor

Com o protocolo de comunicação determinado, o jogo modelado e a consciência que o mesmo seria centralizado no servidor, deu-se início a fase de criação do servidor do jogo.

Como uma primeira determinação, a dupla escolheu manter a conexão cliente-servidor aberta durante todeo o período do jogo. Apesar de não ser a melhor prática, a mesma é mais simples e se torna a mais viável dado o tempo para a elaboração da solução do problema.

Para haver comunicação entre o Cliente e o Servidor, o servidor deve possuir um Socket especializado em ouvir conexões de entrada. O cliente, por sua vez, deve possuir um socket capaz de se conectar ao socket do servidor e, também, o endereço IP e porta na qual o servidor está alocado. Uma vez conectado, para atender as solicitações de diversos clientes, o servidor deve ser capaz de criar diversos processos escravos. Cada processo escravo, aqui achamado de ClientHost, é uma thread que redireciona o cliente para o jogo depois do mesmo ser recebido pelo servidor.

Para ser capaz de identificar e diferenciar as solicitações dos clientes, o servidor possui uma cópia do protocolo. A entidade ClientHost faz uso desse protocolo e ouve todas as solicitações do cliente. A mesma é capaz de atender tais solicitações pelo fato de possuir a engine do jogo consigo. Por exemplo, para uma solicitação de *Receber Palavra*, o ClientHost executa a função da engine responsável por recuperar a palavra, e, ao possuí-la em mãos, o ClientHost a encaminha ao cliente, atendendo a sua solicitação. Todos os passos do exemplo são feitos em cada uma das solicitações possíveis ao servidor.

3.9. Ranking e Concorrência

O ranking do jogo foi elaborado como uma entidade a parte do jogo. Um dos motivos para tal, além da modularização, foi a previsão de problemas de concorrência relacionados ao mesmo. Um dos problemas que poderia acontecer, segundo discutido em sessão, foi que se dois jogares diferentes atualizarem simultaneamente suas pontuações, uma das duas pode se perder.

Como uma identidade independente, o sistema de ranking deve estar presente em todos os jogos, e consequentemente em todas as instâncias da entidade ClientHost. Para garantir um ranking íntegro para todos os clientes, a dupla optou por forçar o sistema a manter somente um ranking e, sendo assim, cada cliente esperar por sua vez de atualizá-lo quando necessário. Para implementar a solução descrita, foi utilizado o padrão de projeto Singleton, tornando possível a existência de apenas uma instância do ranking. Através disso, o encarregado por determinar a vez de cada um manipular o ranking é determinado pela própria fila de operações do sistema operacional [Tanenbaum and Machado Filho 1995].

3.10. Interface Gráfica

Por decisão da dupla, a interface gráfica foi deixada como última tarefa a ser realizada. Durante o processo de desenvolvimento foi implemantada uma *CLI (Command Line In-*

terface) para os testes e visualização do jogo. Tal escolha foi feita pelo fato da CLI ser algo mais rápido de ser feito e não deixar de ser funcional. Ao finalizar todo o *background* do jogo, foi iniciada a produção da Interface Gráfica utilizando o Swing.

A Interface Gráfica é composta por quatro telas:

• **SettingsScreen** - Tela de configurações na qual o usuário deve inserir o ip e a porta na qual o servidor está rodando.

• **RankingScreen** - Disponibiliza o ranking dos Três melhores jogadores. Um jogador quando entra no jogo pode visualizar diretamente o ranking não precisando está logado ou jogando para ter acesso. O ranking é acessível apenas através da Registration Screen.

• **RegistrationScreen** - O jogador pode informar seu nome, caso o nome já esteja cadastrado, todas as informações serão carregadas senão é adicionado na base de dados do jogo. A tela Contpem também um link para o ranking.

• **ScreenGameMode** - Tela que possibilita o jogador à jogar o Roda Roda. Estão contidos na tela dados do Jogador: Recorde Pessoal; Nome do Jogador; Valor ganho na partida atual, e dados do Jogo como palavra, dica, valor da roleta e quanto ganharam ao indicar uma letra. Além disso, a tela conta com um teclado virtual com as letras que podem ser indicadas.

4. Resultados e Discussões

Nessa seção serão abordados os principais resultados obtidos e os testes realizados no software produzido.

4.1. Testes

Durante todo o desenvolvimento do software foram feitos diversos testes para garantir a integridade do produto. Os testes foram realizados em situações que simulavam o comportamento de um usuário normal através da linha de comando. Casos de fronteira também foram inseridos nos casos de teste.

4.1.1. Teste da engine do jogo

Primeira entidade a ser elaborada, a engine do jogo foi também a primeira a ser testada. Através de interface de linha de comando simulando o jogo em modo single-player offline, foram analisados o comportamento da engine em todos os níveis e diversas situações. Não foram detectados problemas na engine desenvolvida.

4.1.2. Teste de comunicação

Depois da engine do jogo devidamente construída, o próximo passo era construir a interface de comunicação entre o cliente e o servidor. Para testar a comunicação foram construídas interfaces em linhas de comando no cliente e no servidor. Quando um cliente enviava uma solicitação, o servidor imprimia em tela o nome da solicitação. Dessa forma,

foi possível verificar se a solicitação encaminhada era recebida corretamente. Os problemas identificados nessa fase de testes foram somente relativos a erros nas cópias dos protocolos, facilmente identificados e corrigidos.

4.1.3. Teste geral do jogo

Após os testes do jogo e de comunicação, os dois foram integrados e somados a interface gráfica. A partir desse momento foi testada o comportamento do jogo como um todo. Além de ser testado com o diversos clientes e o servidor rodando na mesma máquina, também foi testado com um cliente e um servidor em máquinas diferentes. Não foram detectados problemas nessa fase de testes.

4.1.4. Resultado final

O software "Roda a Roda", produto final em resposta ao Problema 1 do MI - Concorrência e conectividade, simula virtualmente o jogo Roletrando. Distribuído entre 4 telas diferentes, o jogo funciona em uma rede local na qual um dos computadores atua como um servidor e os demais podem se conectar a ele para jogar.

Durante o período do jogo, o jogador não precisa girar a roleta. A mesma é girada automaticamente assim que o mesmo escolhe uma letra. Quando um usuário cai no "Perdeu Tudo", a interface mostra um aviso e o valor conquistado naquela rodada é perdido. As quantias das outras rodadas são mantidas como valor acumulado.

Para inicializar o servidor do jogo, o usuário deve informar o nome de dois arquivos nos quais o ranking geral e os três melhores jogadores serão armazenados, respectivamente. Para inicializar o cliente, primeiro deve ser informado o IP na qual o servidor está rodando. O próximo passo é inserir o nome de usuário que gostaria que aparecesse no jogo, e após isso, o jogo é inicializado.

O software desenvolvido conta com uma interface simples e amigável ao usuário, buscando tornar o jogo o mais fácil de entender e jogar possível.

5. Conclusão

Observando os requisitos do problema e os resultados obtidos, é perceptível que todos os quesitos propostos pelo problema foram atendidos com exatidão, apesar das dificuldades iniciais em encontrar as funcionalidade do jogo e aprender os conceitos de concorrência e conectividade. Foi gratificante desenvolver um dos jogos mais populares da televisão brasileira.

O programa pode ter um avanço considerável referente ao protótipo atual, sendo assim podendo tornar ainda mais atrativo e mais divertido para os jogadores.De acorco com a problemática envolvida é possível pensar diferentes sugestões para potencializar o jogo, dentre elas:

• **Modo versus:** Um determinado jogador pode jogar contra outros jogadores onde o melhor é o vencedor;

• **Multiplayer:** O jogo pode ser jogado por vários jogadores mas diferente do modo versus o multiplayer permite que jogadores possam auxiliar em uma mesma partida jogando em duplas ou trios por exemplo;

6. Referências

Nessa seção constam os trabalhos utilizados como base teórica que foram necessários para chegar a solução descrita nesse relatório.

References

Britannica, E. (2015). ”client-server architecture”.

Fileto, R. (2006). ”sistemas cliente-servidor”.

Kurose, J. F., Ross, K. W., Marques, A. S., and Zucchi, W. L. (2010). *Redes de computadores ea Internet: Uma abordagem top-down*. Pearson.

Parziale, L., Liu, W., Matthews, C., Rosselot, N., Davis, C., Forrester, J., Britt, D. T., et al. (2006). *TCP/IP tutorial and technical overview*. IBM Redbooks.

Silberschatz, A., Galvin, P. B., Gagne, G., and Silberschatz, A. (1998). *Operating system concepts*, volume 4. Addison-Wesley Reading.

Tanenbaum, A. S. (2003). *Redes de computadoras*. Pearson Educación.

Tanenbaum, A. S. and Machado Filho, N. (1995). *Sistemas operacionais modernos*, volume 3. Prentice-Hall.