Problema 3: "*Maranhon*"
TEC502 - MI - Concorrência e Conectividade

Allen Hichard, Cássio Santos

[1]Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana – BA – Brasil

{allenhichard21,sss.cassio}@gmail.com

Abstract. *This report describes the resolution process of the third Mi - Concurrence and Connectivity problem of the 2015.2 period. This report aims to present the development of an network shopping application for books. The application designed for computers has features to allow connections scalable way servers and balancing client connections, enabling information sharing and file management between the network servers. The focus of this paper is the application of concurrence and connectivity concepts associated with the development of the software as the communication protocols TCP, Unicasting application implemented by Peer to Peer, file synchronization and concurrent access to resources. The software was developed based on the concepts mentioned. In this report are presented the steps taken to implement the program as well the obtained results.*

Resumo. *Este relatório descreve o processo de resolução do Problema 3 do MI-Concorrência e Conectividade do semestre 2015.2. Este relatório tem como objetivo apresentar o desenvolvimento de um aplicação para compras de livros em rede. A aplicação criada para computadores possui características de permitir conexões de servidores de forma escalonável, e balancear os clientes que se conectem ao sistema, possibilitando o compartilhamento de informações e gerenciamento de arquivos entre os servidores operantes da rede. O enfoque desse artigo encontra-se na aplicação de conceitos de concorrência e conectividade associados ao desenvolvimento do software sendo eles os protocolos de comunicação TCP, aplicação Unicasting implementado por Peer to Peer, sincronização de arquivos e acesso concorrente a recursos. O software foi desenvolvido com base nos conceitos citados. Neste relatório serão apresentados os passos adotados para a implementação do programa junto aos resultados obtidos.*

1. Introdução

Ao longo do tempo, vem se descobrindo cada vez mais aplicações para a internet. Uma das atualmente mais utilizadas e lucrativas atividades disponibilizadas na rede mundial é o mercado de compra e venda de materiais. Empresas físicas expandem seus negócios e consequentemente seus lucros ao permitirem que usuário efetuem compras em suas lojas virtuais e recebam os seus produtos com a comodidade de não precisarem sair de casa.

Uma das empresas que adotou este tipo de mercado foi a Maranhon, uma loja online de venda de livros pela internet concorrente da famosa Amazon. Com o crescimento

das vendas e a popularização do sistema da Maranhon, o seu, até então, único servidor passou a não suportar a grande quantidade de acessos de seus usuários. Analisando as vantagem do comércio online, a empresa decidiu por investir na expansão de sua arquitetura. Para isso, a sua infraestrutura passaria a contar com mais dois servidores capazes de fornecer serviço aos seus clientes.

Para sustentar esta nova infraestrutura, fez-se necessário a elaboração de um sistema capaz manter sincronizada e consistente a base de dados de livros dos servidores e o qual fosse escalonável, ou seja, capaz de ser ampliada com a adição de outros servidores.

O sistema consiste em uma aplicação na qual clientes efetuam compras identificando o livro que deseja comprar e a quantidade desejada, com isso, o servidores são capazes de efetuar operações de escrita em arquivo para manter a base de dados consistente e armazenar a nova configuração do estoque de livros, bem como manter salvo uma lista de clientes com o valor total de suas compras.

Uma exigência deste novo sistema é que fosse reduzido o número de processos de clientes executados no servidor, sendo necessário portando fazer-se uso de conceitos novos de distribuição de recursos e balanceamento. Além disso, era de extrema importância que os clientes conseguissem continuar realizando suas compras, mesmo que o servidor ao qual ele estivesse conectado sofresse algum problema e saísse da rede, ou seja, o cliente deveria ser realocado para um novo servidor caso algum problema no seu servidor atual fosse detectado. Como proposta para solução do problema a dupla decidiu desenvolver um sistema hibrido com a presença de um DataCenter. O sistema é composto de três pacotes e estruturas básicas: o distribuidor, responsável por realocação de clientes e manipulação de servidores além de manter um backup dos dados alterados nos servidores; os servidores, responsáveis por permitir ou não a alteração de um recurso no arquivo (no caso os livros) e tratar todas as requisições feitas pelo cliente; e os clientes, responsáveis por efetuar as operações de compra as quais são a base da lógica do sistema.

Vale lembrar que para o desenvolvimento deste sistema distribuído de compras foi utilizado a linguagem de programação Java, com versão de JDK 8. Também foram utilizados conceitos como: arquiteturas de aplicação híbrida, protocolo de transporte TCP, sockets, threads, modelo de transmissão unicast, distribuição de carga e recurso, acesso concorrente em arquivo além de outros conceitos relacionados a redes de computadores.

2. Fundamentação teórica

Nesta seção serão tratados os principais conceitos utilizados para a resolução do problema, conceitos estes voltados principalmente ao objetivo do Modulo Integrado, por isso, não serão tratados aspectos como interface gráfica e conceitos de Java de forma detalhada.

2.1. Protocolo de Transmissão

Para que dois ou mais computadores se comuniquem, é necessário a utilização de algum tipo de protocolo que permita a troca de informação entre eles. Os protocolos de transporte tem as seguintes principais funções:

- Identificar apenas as várias aplicações de cada máquina com acesso à rede
- Detectar e, se possível, corrigir erros ocorridos na transmissão dos dados
- Controlar o fluxo de dados na comunicação entre a origem e o destino

- Controlar os congestionamentos na rede
- Segmentar e remontar mensagens fragmentadas pela rede

A rede identifica apenas as máquinas, atribuindo a elas endereços IP. Para identificar aplicações, as aplicações que utilizam a rede devem ser associadas a portas de comunicação, portas estas que são os pontos de acesso ao serviço de transporte.

As portas de origem e de destino devem ser identificadas no estabelecimento de conexões TCP ou no envio de datagramas usando UDP.

2.1.1. Protocolo UDP

O protocolo UDP, que significa User Datagram Protocol, trata-se de um dos protocolos referentes a camada de transporte pertencentes as camadas da comunicação em redes de computadores. Por [Forouzan 2009], o UDP e caracterizado por ser mais simples que o TCP em virtude de sua flexibilidade, pois, enquanto o TCP dedica-se a conexão e a chegada dos dados ao destino, o UDP presa apenas pela velocidade de propagação. Tal protocolo não verifica recebimentos de dados, não possui uma opção de reenvio, não ordena mensagens, não verifica integridade de informações transitadas e também não possui controle de fluxo.

Por, diante de tantas desvantagens perante o protocolo TCP, o qual efetua inúmeras medidas de segurança e preservação de dados para montar uma boa comunicação, o UDP tem como característica a sua velocidade de envio, propagação e recepção, pois possui menos quesitos a serem analisados. Tal característica torna-se essencial para aplicar em situações que prezem pela transmissão em tempo real, como vídeo conferencias, streams e jogos.

O datagrama UDP, nome do pacote com os dados, segundo [Forouzan 2009], possui os campos porta de origem e porta de destino, os quais especificam as portas que serão utilizadas na comunicação. Já o campo tamanho descreve quantos bytes terá o pacote completo. O campo checksum e opcional e faz uma soma verificadora para garantir que os dados estarão livres de erros. Alguns protocolos conhecidos que utilizam o UDP são TFTP, SNMP, DHCP e DNS.

2.1.2. Protocolo TCP

O Transmission Control Protocol, ou TCP, foi projetado especificamente para oferecer um fluxo de bytes fim a fim confiável numa inter-rede não confiável [Tenenbaum 2003]. Ao contrario do UDP, o TCP tem garantia de envio dos pacotes, possui ferramentas de controle de fluxo de pacotes e erros, alem 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, memoria e processamento das entidades envolvidas do que o UDP

Para aplicação no sistema foi escolhido o protocolo TCP, por mais se adequar aos requisitos proposto envolvendo segurança e estado consistente dos dados. Além de algumas outras vantagens nesses aspectos perante o UDP, sendo eles:

- Serviço orientado por conexão; uma sessão é estabelecida entre os hosts;
- Garante a entrega através do uso de confirmações;
- Entrega sequenciada dos dados;
- Garantia de transporte confiável de dados.

2.2. Endereçamento e transmissão

Em uma tecnologia de rede na qual existe o envio de dados através de mensagens, faz-se necessário executar uma série de tarefas responsáveis por estabelecer uma transmissão correta da mensagem. Uma tarefa importante que diferencia o tipo de método a ser utilizado é como as mensagens são endereçadas. Qual método é o mais correto de ser utilizado depende, neste caso, se o remetente conhece previamente o endereço de quem ele está tentando entrar em contato, ou se simplesmente deseja enviar uma mensagem de forma geral para a rede.

2.2.1. Multicast

Multicasting é a forma de entrega de informações para múltiplos destinatários simultaneamente. Cada usuário em uma rede pode se conectar a um determinado grupo multicasting, e compartilhar toda informação emitida pelo mesmo.

O multicast oferece uma forma eficiente de oferecer suporte a aplicativos um-para-muitos de grande largura de banda em uma rede [Rosenzweig et al. 1998]

- O multicast pode reduzir drasticamente o trafego da rede ao enviar uma única copia dos dados.
- Os hosts podem ser configurados para multicast sem a necessidade de atualização de hardware.
- Como os roteadores mais novos já oferecem suporte ao encaminhamento de multicast e aos protocolos de roteamento de multicast, habilitar o multicast em uma rede sera prático e econômico.

2.2.2. BroadCast

Segundo [Tenenbaum 2009], Broadcast é uma transmissão de difusão na qual existe apenas um canal de comunicação, compartilhado por todas as máquinas da rede. Este tipo de sistema oferece a possibilidade de endereçamento de um pacote de dados para todas as máquinas na rede, que vão, por sua vez, receber e processar este pacote.

2.3. UniCast

Unicast é o termo usado para descrever a comunicação na qual um pacote de informação é enviado de um ponto a outro. Neste tipo de transmissão existe apenas um transmissor e um receptor. Este tipo de transmissão com um transmissor e um receptor é conhecida comumente por transmissão “ponto a ponto” [Fairhurst 2009].

Para o tipo de sistema sugerido se compararmos a nível de funcionalidade, o multicasting se sobressaem perante o Unicasting, pois seria possível todos os servidores se

comunicarem de forma estantânea mandando mensagem para um grupo específico, ou seja, o objetivo dessa opção é acelerar o processo de envio de dados.

Contudo o multicasting é baseado no Protocolo UDP, não possuindo controle de fluxo, podendo haver perda do dado na transmissão, por esse motivo foi identificado que o TCP atenderia melhor para a solução do problema. O sistema tem como características a consistência dos dados e todos os servidores têm que permanecer constantemente atualizado a cada compra do cliente, caso uma informação fosse perdida o sistema não sincronizaria, por isso não foi escolhido o serviço como Multicasting.

2.4. API Socket

Os sockets são as portas pelas quais dados passam da rede para o processo e o processo para a rede. O que acontece é que a camada de transporte do hospedeiro destinatário não entrega os dados diretamente a um processo, mas sim a um socket intermediário. Este destinatário pode conter mais de um socket ao mesmo tempo a qualquer momento, o que gera a necessidade de identificação destes sockets de maneira exclusiva.

Um Socket TCP é identificado por quatro elementos:(endereço IP da fonte, número da porta da fonte, endereço IP do destino, número de porta do destino).

A API de sockets está disponível para muitos sistemas operacionais, incluindo sistema usados em computadores pessoais (por exemplo, Windows NT e Windows 98 da Microsoft) como também vários sistemas UNIX (por exemplo, Solaris da Sun Microsystems)[Gracioli 2007].

Sockets se trata de uma API do java disponível no pacote java.net. Através do protocolo TCP, é possível criar um fluxo entre dois computadores, sendo possível até mesmo conectar mais de um cliente ao mesmo servidor.

Para o caso de mais de um cliente se conectar simultaneamente ao servidor, o servidor utiliza-se de uma manobra para evitar a concorrência que é redirecionar o cliente de uma porta para outra, liberando assim a sua porta inicial e permitindo que outros clientes se conectem novamente. Em Java, essa operação é feita através de *threads*.

2.5. Threads

Em diversas aplicações ocorrem múltiplas atividades ao mesmo tempo, porém, algumas dessas atividades podem ser bloqueadas de tempos em tempos. Dessa forma, uma maneira de contornar os problemas que podem ser gerados nessa operação é decompor a aplicação em diversas threads sequenciais que executam de maneira quase em paralelo. [Tenenbaum 2009]

A Thread é um conceito usado que torna possível executar várias instruções de forma simultânea em um mesmo processo. A diferença mais marcante entre utilizar processos paralelos e utilizar threads em um mesmo processo, é que no segundo, as entidades tem a capacidade de compartilhamento de um espaço de endereçamento e de todos os seus dados entre elas.

Uma outra diferença que explica sua alta utilização nas mais variadas aplicações é a sua fácil manipulação, são mais fáceis de criar e destruir em comparação aos processos, visto que não existem quaisquer recursos associados a eles.

Em java para criar linhas de execução paralela utilizamos a classe Thread do pacote java.lang. Esta classe recebe como argumento um objeto com o código que desejamos rodar para iniciar a execução de uma *Thread* é necessário utilizar o método start da instância da *Thread*. Ex.: *threadgame.start();*

Para uma classe ser adicionada a uma *Thread* ela deve inicialmente implementar a interface *Runnable*. A interface Runnable contém apenas um método chamado run, basta implementá-lo e quando executar o método *start()* da *Thread*, já saberá que deve executar o método *run()* da classe.

As classes também podem extender a classe Thread, podendo assim usar o método *start()* diretamente da classe. Este forma não é recomendada pois ao extender a classe Thread a classe em questão herda muito mais que apenas o método *run()*.

O escalonador (*scheduler*) basicamente pega todas as threads que precisam ser executadas e faz o processador ficar alternando a execução de cada uma delas.

2.6. Exclusão Mútua

A exclusão mútua é um modo de assegurar que outros processos sejam impedidos de usar uma variável ou um arquivo compartilhado que já estiver em uso em um processo [Tenenbaum 2009]. Os servidores no sistema faz exclusão mútua no arquivo local, impedindo o acesso a um recuso em uso, enquanto o DataCenter(Distribuidor) garante que apenas um servidor modifique o arquivo de backup por vez.

2.6.1. Semáforo

Os recursos no sistema possuem semáforos para indicar a possibilidade de compra e tratar a concorrência no sistema, impossibilitando a compra de um recurso em uso, ele se comporta da seguinte forma: Um semáforo pode conter o valor 0 - indicando que nenhum sinal de acordar foi salvo ou algum valor positivo se um ou mais sinais de acordar estiver pendentes [Tenenbaum 2009].

Na aplicação cada livro tem o seu valor de semáforo correspondente, o sinal 1 indica que o recurso está disponível e pode ser acessado pelo cliente, possibilitando a compra caso aja quantidade suficiente, caso o sinal for 0 o processo ficará bloqueado ocorrendo uma espera ociosa.

2.7. DeadLock

Um conjunto de processos está em situação de impasse se todo o processo pertencente ao conjunto estiver esperando por um evento que somente outro processo desse mesmo conjunto poderá fazer acontecer [Tenenbaum 2009]. O DeadLock (Impasse) foi tratado ao permitir que um processo só faça acesso a um recurso por vez, não gerando portando impasses no sistema através da eliminação da espera circular que é uma das condições para que haja impasses.

3. Métodologia

Por se tratar de um sistema que contém três tipos diferentes de serviços a dupla decidiu por adotar um protocolo para identificação de máquinas o qual é necessário para que o re-

ceptor da conexão identifique a máquina que se conectou e execute os devidos tratamentos dos serviços que apresentam-se modularizados em classes diferentes.

Na tabela **Tabela 1** podem ser vistos os valores escolhidos para identificação das máquinas.

Table 1. Códigos identificadores das máquinas

Código Identificador	Tipo Máquina
0	Servidor
1	Cliente
2	Distribuidor

Como o sistema utiliza apenas conexões *unicast* com o protocolo TCP, a cada conexão entre os atores da aplicação (servidor, cliente e distribuidor) aquele que requisitou a conexão envia inicialmente o seu identificador para que o outro lado da conexão saiba como tratar esta maquina.

Para um melhor entendimento do funcionamento do sistema este será dividido em 4 partes: Interação Servidor-Distribuidor, Interação Cliente-Distribuidor, Interação Cliente-Servidor e Interação Servidor-Servidor.

3.1. Interação Servidor-Distribuidor

A comunicação entre Servidor e distribuidor acontece de forma bidirecional, portando, fez-se necessário a criação de uma conexão Peer-to-Peer entre estes dois serviços. Ao iniciar sua aplicação, o servidor é perguntado sobre a porta que as máquinas devem utilizar para conectar-se a ele além de ser perguntado sobre o Ip do distribuidor ao qual ele deve se conectar. Tendo estas duas informações, o servidor estabelece uma conexão com o distribuidor e informa seu IP e a porta de acesso, o distribuidor, por sua vez, armazena estes dados em um lista de servidores, e conecta-se ao servidor enviando, em sequência, o código identificador "2" para mostrar-se como distribuidor.

Depois deste momento o servidor e o distribuidor podem trocar informações. A seguir, na **Tabela 2** é possível ver o protocolo criado para esta comunicação juntamente com a ação.

Table 2. Protocolo de Aplicação para comunicação entre servidor e distribuidor

Código Operação	Ação	Direção
3	Enviar IP Porta do Servidor	Servidor ->Distribuidor
4	Enviar Lista de IPS	Distribuidor ->Servidor
5	Enviar Livros com semáforo	Distribuidor ->Servidor
6	Guardar dados do Cliente	Servidor ->Distribuidor

Além deste protocolo de aplicação, vale ressaltar que o protocolo de transporte utilizado para esta interação foi o TCP (Transport Control Protocol) devido a garantia de envio e recebimento de dados.

Com o conhecimento do protocolo de aplicação, pode-se avançar ao funcionamento de cada ação presente nele, e este é o espaço dedicado a essa explanação.

3 - Enviar IP Porta do Servidor

Esta operação é utilizada quando o servidor conecta-se ao sistema, e é responsável por enviar para o distribuidor as informações necessárias para o acesso a este novo servidor. O distribuidor, por sua vez, tem o papel de armazenar estas informações em uma lista para em um outro momento informar ao cliente com quem ele deve se conectar e para informar aos servidores com quais outros eles devem se conectar.

4 - Enviar Lista de IPS

Como comentando na operação anterior, os servidores necessitam estabelecer conexão com os outros servidores. Visto que a verificação de possibilidade de acesso ao recurso pode não necessariamente ter como responsável o servidor atual. Esta conexão é feita basicamente para remover o ponto central de falha caso fosse necessário sempre acessar o distribuidor para Alterar a linha do arquivo em outro servidor.

Esta operação é chamada sempre que houve uma alteração nos servidores conectados ao distribuidor, ou seja, sempre que um servidor entrar ou sair do sistema.

5 - Enviar Livros com Semáforos

Após diversas discussões em sessão sobre as formas que poderiam ser manipulados os recursos dos livros, a dupla decidiu distribuir a posse das variáveis de manipulação entre os servidores, fazendo com que não exista uma sobrecarga de operações em nenhuma das máquinas, pois elas serão bloqueadas apenas nos recursos que possuírem.

Sendo assim, esta operação é executa sempre que houver uma alteração nos servidores conectados ao distribuidor, e é responsável por enviar a todos os servidores a lista de livros incluindo a informação de qual servidor manipula o *semáforo* de cada livro. Na **Tabela 3** a seguir, pode ser visualizada a estrutura desta tabela.

Table 3. Exemplo da tabela de livros com 3 servidores conectados, 6 livros na lista e na visão do servidor 2

Id	Nome do Livro	Preço do Livro	Quantidade Disponível	IP-Porta	Semáforo
0	50 tons de cinza	R$ 50,00	0	IP-Porta0	x
1	O código da Vince	R$ 15,00	10	IP-Porta1	x
2	Praticamente Inof.	R$ 7,00	35	IP-Porta2	1
3	A culpa é das estr.	R$ 20,00	4	IP-Porta0	x
4	Sistemas Operac.	R$ 120,00	50	IP-Porta1	x
5	Crepúsculo	R$ 50,00	2	IP-Porta2	0

6 - Guardar Dados do Cliente

A dupla optou por manter o arquivo de Clientes apenas no distribuidor, devido ao fato de que não haverá concorrência nas linhas (recursos) pois já que a lista de Clientes está sendo carregada e manipulada na memória antes de salvar no arquivo, e como apenas o Servidor que tem o cliente conectado pode manipular esse recurso, dois servidores nunca tentarão modificar as informações de um mesmo cliente ao mesmo tempo.

Esta operação é executa toda vez que um cliente efetuar um compra, e tem como função enviar do servidor ao distribuidor o nome do cliente que efetuou a compra e o valor comprado para que o distribuidor, por sua vez, armazene na memória e no seu arquivo o novo valor total de compras feita pelo cliente.

3.2. Interação Cliente-Distribuidor

A conexão entre Cliente e Distribuidor acontece apenas em um sentido e é feita quando o cliente inicia o seu sistema e informa o *IP* do distribuidor que deseja acessar, neste tipo de interação, o cliente tem dois papeis básicos, requisitar conexão com servidor e Informar queda de servidor, os códigos de operação bem como as ações representadas podem ser vistas na **Tabela 4**.

Table 4. Protocolo de Aplicação para comunicação entre cliente e distribuidor

Código da Operação	Ação	Direção
7	Conectar com servidor	Cliente - >Distribuidor
8	Informar queda Servidor	Cliente - >Distribuidor

A seguir, serão explicadas mais a fundo as operações entre o Cliente e o Distribuidor.

Conexão

Ao iniciar o sistema o cliente é perguntado sobre o IP do distribuidor que deseja se conectar e conecta-se a porta padrão "11111" do mesmo, com isso é notado a questão de transparência de conexão para o cliente pois o mesmo só precisará conhecer o endereço da máquina distribuidora sem ter que precisar saber da estrutura interna da rede ou dos endereços dos servidores inseridos nela. Como mostra na figura **[Figura 1]**.

```
Informe o IP do distribuidor
localhost
Cliente se conectou ao distribuidor
Enviou codigo para Distribuidor para sinalizar que é cliente
Cliente recebeu Servidor para conectar: 127.0.0.1-2222
Cliente se conectou ao Servidor
Enviou codigo para Servidor para sinalizar que é cliente
Informe seu nome:
Allen
id 0 livro 50 tons de cinza valor 50.0 qtd 0
id 1 livro O codigo da Vince valor 15.0 qtd 2
id 2 livro Praticamente Inof. valor 7.0 qtd 30
id 3 livro Sistemas Operac. valor 120.0 qtd 130
id 4 livro Crepusculo valor 50.0 qtd 1

Informe a opção desejada
1 - Comprar um livro
```

Figure 1. Conexão do cliente no servidor - Distribuidor [Próprio Autor]

7 - Conectar com servidor

Logo após iniciar a conexão com o distribuidor e o distribuidor reconhecer a máquina conectada como um Cliente, o cliente tem a função de requisitar um *IP* e Porta de acesso para conectar-se a um servidor. Neste caso, o distribuidor analisa, através de seu algoritmo de balanceamento, qual dos servidores é o mais indicado para que o cliente se conecte e envia ao cliente as informações de *IP* e porta para que seja feita a conexão direta entre Servidor e Cliente, não mais sendo necessário manter conexão com o distribuidor, e portando, encerrando seu *Socket* **[Figura 2]**.

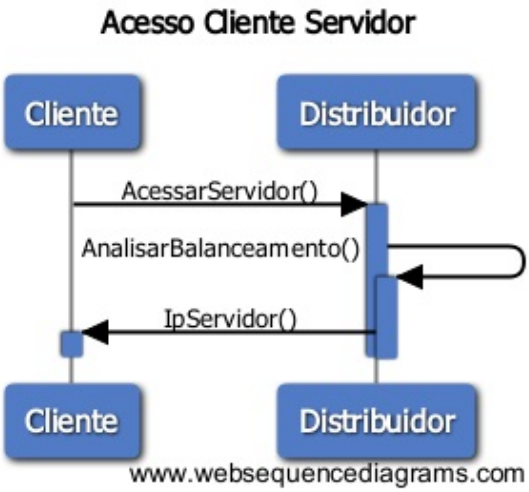

Figure 2. Diagrama de sequência Cliente - Distribuidor [Próprio Autor]

8 - Informar queda Servidor

Um dos papeis de extrema importância dos clientes é informar ao distribuidor quando os mesmos não conseguirem enviar alguma mensagem ao servidor que lhe foi indicado. Através desta operação, o cliente envia ao distribuidor o IP e Porta do servidor foi desconectado do sistema. Assim, o distribuidor pode tomar as providencias necessárias que seriam:

1. Reorganização da lista de Ips-Porta dos servidores conectados no sistema
2. Envio aos servidores restantes, a nova lista de IPs-Porta para que os mesmos estabeleçam suas conexões.
3. Reorganização da tabela de semáforos atribuindo a posse dos recursos(livros) que o servidor desconectado tinha posse para os servidores ainda conectados.
4. Envio da nova tabela de posse dos semáforos a todos os servidores para que estes saibam quais recursos(livros) eles tem a posse da manipulação.

3.3. Interação Cliente-Servidor

A interação Cliente-Servidor é onde estão definidos todas as ações básicas do sistema, principalmente a compra de um livro. Nesta interação o cliente recebe o Ip-Porta do servidor que lhe é indicado para conectar e estabelece a conexão com o mesmo. A seguir, o cliente fica possibilitado de requisitar uma das 3 seguintes operações apresentadas na **Tabela 5**.

Table 5. Protocolo de Aplicação para comunicação entre cliente e servidor

Código da Operação	Ação	Direção
9	Receber Lista Livros	Cliente ->Servidor
10	Efetuar Compra	Cliente ->Servidor
11	Enviar Nome Cliente	Cliente ->Servidor

9 - Receber Lista Livros

O cliente envia este código de operação para o Servidor e espera por receber a versão mais atualizada da String contendo as informações dos livros, sendo elas: Id do livro, Nome do livro, valor e quantidade disponível.

Esta operação é feita sempre que o cliente efetuar um operação de compra e no inicio do serviço do seu serviço.

10 - Efetuar Compra

O cliente envia para o servidor ao qual ele está conectado o Id do livro que o mesmo deseja comprar e a quantidade desejada. Antes de efetuar esta operação o cliente deve enviar o seu nome usando o código de operação 11, para que o servidor possa atualizar tanto a lista de livros quanto enviar a requisição para o distribuidor pedindo para atualizar o recurso referente ao cliente que diz respeito ao valor total comprado pelo cliente.

Quando efetuar a compra o servidor tem por sua vez o papel de verificar a disponibilidade de livros requisitados no estoque e a posse do semáforo do recurso para enfim efetuar o que for necessário, esta verificação será explanada na sub seção a seguir quando falaremos sobre comunicação entre os servidores.

Depois de verificar o semáforo, o servidor fica responsável por atualizar o seu arquivo e propagar a alteração em todos os Servidores inclusive no Distribuidor para que todos os arquivos fiquem sincronizados na ultima versão.

11 - Enviar Nome Cliente

Quando conectado ao servidor, o Cliente tem o dever de informa-lo qual o seu nome. Este nome ficará armazenado na instância de *TratamentoCliente* no *Controller* do Servidor em questão, e será utilizado para fazer as devidas alterações no arquivo de Clientes no servidor sempre que necessário, ou seja, sempre que uma requisição de compra feita pelo cliente, seja bem sucedida.

3.4. Interação Servidor-Servidor

A interação Servidor-Servidor é feita basicamente em dois momentos. O primeiro, para verificar a possibilidade de manipulação de um recurso, e o segundo, para replicar a alteração nos demais arquivos. A seguir, a **Tabela 6** representando o protocolo adotado.

11 - Verificar Semáforo

Antes de aceitar a requisição de compra de um cliente, e após verificar a quantidade de livros no estoque o servidor verifica se o mesmo tem o poder de manipulação do semáforo, se tiver, ele verifica o estado do semáforo: Caso seja 1, ele decrementa o semáforo, faz a operação em seu arquivo, propaga a operação utilizando o código de

Table 6. Protocolo de Aplicação para comunicação entre servidores

Código da Operação	Ação	Direção
11	Verificar Semaforo	Servidor ->Servidor
12	Atualizar Linha Livro	Servidor ->Servidor Servidor ->Distribuidor

operação 12 e finalmente incrementa novamente o semáforo; Caso o semáforo tenha valor 0, a thread de tratamento cliente no servidor é bloqueada e fica aguardando em uma fila verificando o semáforo constantemente até que o mesmo volte para o valor 1 e ele efetue as operações necessárias.

Uma outra possibilidade é o servidor ao qual o cliente está conectado, não ter o poder de manipulação do semáforo do recurso o qual o cliente deseja comprar. Neste caso, entra em uso o protocolo de aplicação referente ao código de operação 11, que vai acessar diretamente o servidor que possui o recurso através das conexões já estabelecidas e fazer toda a operação já comentada, visto que o método que será chamado no outro servidor é o mesmo que foi chamado neste.

12 - Atualizar Linha Livro

Este protocolo de aplicação é utilizado para manter a consistência e sincronização entre os arquivos nos servidores. Sempre que uma alteração é feita localmente no arquivo do servidor, este tem como obrigação replicar a operação nos outros servidores e no *DataCenter*(Distribuidor) como forma de *backup*.

Assim, é enviado aos servidores o Id do livro e a quantidade a ser decrementada, para que os mesmos atualizem-se.

4. Resultados e Discussões

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

4.1. Produto

O software "Maranhon" produto final em resposta ao Problema 3 do MI - Concorrência e conectividade, simula um processo de compra online de livros. O sistema conta com 3 operadores distintos:

- Os servidores - que atendem as requisições do usuário;
- O servidor Data Center (Distribuidor)- utilizado para balanceamento de requisições e backup geral dos servidores.
- Os Clientes - usuários que utilizam dos serviços prestados.

O sistema é iniciado pelo Data Center(Distribuidor) , no qual o mesmo fica esperando comunicação de servidores ou clientes. Quando um servidor conecta é enviado um sinal de operação indicando que o mesmo é servidor, o mesmo critério é utilizado para conexão do cliente.

O servidor após enviar seu sinal, faz uma comunicação bidirecional com todos os servidores ativos, e fica a espera de conexão de novos usuários. Os clientes quando

entram no sistema é direcionado ao Data Center(Distribuidor), onde é verificado o melhor servidor para atendê-lo e redirecionado para o mesmo.

O sistema garante a compra do cliente mesmo após a inoperação do servidor ao qual pertence. Como os servidores estão interligados ao Data Center, é identificado pelo cliente a queda do servidor e enviado uma solicitação de reconexão com outro servidor na rede para finalizar a compra.

Cada servidor possui seu próprio arquivo de livros, onde qualquer modificação efetuada é atualizado os arquivos dos demais servidores e também feita uma atualização geral em um arquivo de backup no Data Center, que faz o controle de fluxo dos dados e garante a atualização geral de compras dentre os servidores.

O Data Center tem como papel fundamental salvar os registros de compras dos clientes. Cada compra efetuada com sucesso é registrada em um arquivo de clientes, arquivo esse utilizado para efetuar *login* ou cadastrar novo cliente.

4.2. 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.

4.2.1. Teste de Balanceamento de Carga

A cada nova conexão de um cliente, o sistema Distribuidor analisá o servidor com menos requisições de usuário sendo atendidas, realoca de forma transparente. Foi verificado a distribuição de carga inserindo múltiplos servidores e clientes, havendo três servidores operante e nove cliente enviando requisições de compra, cada servidor atenderá três requisições.

4.2.2. Teste de Transparência

O cliente precisa conhece o Endereço IP apenas do servidor com características únicas de redistribuir os clientes na rede analisando o melhor servidor apto e operante para atendê-lo. Ao se conectar com este unico distribuidor conhecido, o cliente recebe a informação do Ip e Porta do servidor que ele deve se conectar para iniciar suas solicitações.

4.2.3. Teste de Inoperação do Servidor

Caso qualquer servidor fique inoperante, o mesmo fica impossibilitado de fazer atualizações constantes referente as novas compras efetuadas no sistema durante o período de sua inoperação, caso ele volte a ficar operante, o mesmo faz a requisição do serviço de *backup* e atualiza seus registros, garantindo a consistência dos dados.

Nesse processo foi testado outro requisito referente aos cliente: caso o servidor esteja tratando mil clientes e durante as requisições o mesmo fica inoperante, supondo que ainda tenha dois outros servidores ainda disponíveis, cada servidor recebe conexões

de quinhentos novos clientes, ou seja, há distribuição de carga não só na inserção, mas também quanto um servidor fica inoperante.

4.2.4. Teste de escalonamento dos servidores

Utilizando a comunicação peer to peer entre os servidores, foi possível notar que o sistema se comporta de forma adequada a cada nova inserção de um servidor na rede. Quando um novo servidor faz conexão, a comunicação é feita de forma dinâmica entre ele e os demais na rede.

O escalonamento, é feito de forma sincronizada entre os servidores na rede e o novo servidor a ser inserido, os servidores usam comunicação peer to peer para trocar mensagens entre si. Quando um novo servidor se conecta ao sistema, a aplicação funciona da seguinte maneira:

- O novo servidor se conecta com todos os servidores na rede;
- Todos os servidores na rede se conectam com o novo servidor.

4.2.5. Teste de concorrência

Caso o dois clientes tentem comprar o livro ao mesmo tempo, o recurso do livro é repassado apenas para um, enquanto o outro fica bloqueado aguardando o recurso, ou seja, não a concorrência na compra de um mesmo livro. O fato interessante é que toda compra do livro é feito em memória e não diretamente no arquivo, o que possibilita compras múltiplas desde que sejam de livros diferentes.

5. Conclusão

Através das análises, discussões e estudos foi desenvolvido uma aplicação que, através de testes manuais, chegou a atingir resultados satisfatórios, pois o programa é capaz de distribuir igualmente a quantidade de clientes ativos nos servidores operantes.

Os servidores do sistemas operam em conjunto, garantindo maior qualidade e eficiência na manipulação dos clientes, garantindo assim as compras dos mesmo. O maior benefício de ter um grupo de servidores é que mesmo que um deles fique inoperante por um período de tempo todos os clientes conectados no mesmo poderão efetuar a compra sem problemas.

O sistema tem como característica a consistência dos dados e o gerenciamento eficaz dos recursos disponíveis para o cliente, dentre eles, efetuar uma compra e armazenar pontos de compra. Além disso, o sistema possui um único servidor(Distribuidor) com funcionalidade particulares para garantir a atualização dos dados de compra e histórico de compras do cliente, contudo é capaz de fazer backup de todas as requisições de compras feitas pelos clientes independente de qual servidor eles estejam conectados.

Os principais problemas encontrados foram na comunicação entre os servidores e tratamento envolvendo inoperação de um servidor:

Na comunicação o grande desafio foi fazer um sistema completamente escalonável, sendo assim se o sistema receber uma grande quantidade de requisições o

mesmo pode aumentar a capacidade de servidores operantes de forma simples, apenas indicando qual o endereço do Distribuidor.

Na inoperação de um servidor, caso haja clientes conectados, todos serão redirecionados a novos servidores operantes, garantindo a compra para o cliente. Porém, como esse servidor não está operando, o grande problema é a falta de atualização no seu arquivo. Caso torne a ficar disponível novamente, o mesmo é atualizado pelo arquivo de backup geral do sistema.

Dentre as funcionalidades propostas pela *Maranhon* , todos os requesitos propostos foram atendidos, garantindo a confiabilidade na utilização do sistema.

References

Fairhurst, G. (2009). Unicast. Disponível em:http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/uni-b-mcast.html. Acesso em: 25 fev. 2016.

Forouzan, B. A. (2009). *Comunicação de dados e redes de computadores*. AMGH Editora.

Gracioli, G. (2007). Api sockets. Disponível em:`http://www.usr.inf.ufsm.br/~giovani/sockets/api_socket.txt`. [Acesso em: 25 fev. 2016].

Rosenzweig, P., Kadansky, M., and Hanna, S. (1998). *The java reliable multicast service: A reliable multicast library*. Sun Microsystems Laboratories SML Technical Report Series.

Tenenbaum, A. S. (2003). *Redes de computadoras*. Pearson Prentice Hall, 4rd edition.

Tenenbaum, A. S. (2009). *Sistemas Operacionais Modernos*. Pearson Prentice Hall, 3rd edition.