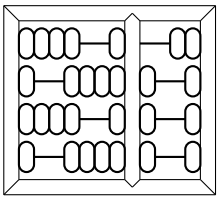
UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE COMPUTAÇÃO



MC833 - Relatório Científico 1

Comunicação Cliente-Servidor Usando Protocolo UDP

135368: Cristina Freitas Bazzano
135749: Flávio Altinier Maximiano da Silva

Campinas - SP
Abril de 2015

Sumário

Resumo

Este trabalho focou-se no estudo temporal de redes cliente-servidor baseadas em comunicação UDP e um comparativo com um estudo anterior baseado em comunicação TCP. Foi implementado um serviço de locadora de filmes baseado em *MySQL* no servidor, e ao cliente foram adicionadas diversas operações de acesso ao banco, baseando-se em operações de consultas pequenas, grandes e de escrita. Observou-se que a escrita no banco é o processo mais lento, enquanto operações de consultas extensas são apenas brevemente mais lentas que buscas mais curtas, o que também ocorreu com o TCP. Outro ponto interessante medido foi a confiabilidade das transmissões, em ambos os protocolos nenhuma mensagem foi perdida. Notamos também que o tempo total de comunicação no protocolo UDP foi cerca de 10 vezes menor que o TCP. Além disso, uma breve análise de tamanho de código mostrou que operações de redes não são as mais presentes em uma implementação desse tipo.

1 Introdução

Na Internet, a grande maioria das montagens de comunicação baseia-se em um sistema cliente-servidor. Nesse contexto, o servidor guarda alguma informação que é relevante ao cliente, que pede essa informação através de uma mensagem na rede; o servidor, por sua vez, responde esse pedido com outra(s) mensagem(ns) contendo a informação requisitada.

O protocolo da camada de transporte escolhido para este projeto é o *User Datagram Protocol* (UDP) [2]. Tal protocolo não garante confiabilidade: é baseado em troca rápida de mensagens, e sem *hand shaking*. Neste trabalho faremos uma comparação com um trabalho anterior que realizou as análises usando o protocolo *Transmission Control Protocol* (TCP) [1], que por outro lado é confiável.

Neste trabalho, num primeiro momento, o foco principal de nossa análise será a medida de tempo total de comunicação entre cliente e servidor, além de uma aproximação do tempo de transmissão das mensagens entre os dois (tempo de as mensagens saírem de um host e chegarem ao outro). Analisaremos também brevemente a confiabilidade das mensagens enviadas, que não é garantida pelo protocolo. Em seguida faremos uma comparação desses resultados com aqueles obtidos em um trabalho passado que analisou o protocolo TCP. O tamanho do código será também analisado, além de um breve parecer sobre o nível de abstração do sistema desenvolvido.

2 Descrição Geral do Sistema

O sistema construído consiste em uma simples montagem de cliente-servidor, com comunicação UDP. Como discutido, tal protocolo não garante que as mensagens enviadas

efetivamente chegarão ao destinatário, nem que chegarão na ordem correta.

O sistema montado é uma emulação de um serviço de locadora online, onde o servidor contém diversas informação sobre os filmes que possui em seu estoque, os quais podem ser acessadas por clientes ou pelo cliente administrador da locadora.

Para sintetizar o problema da melhor forma possível, criamos no nosso banco de dados uma tabela "Locadora" que armazena os seguintes atributos sobre os filmes que possui: título, ano de lançamento, gênero, duração, sinopse, diretor, ator principal, ator coadjuvante principal, exemplares e um id único para cada filme.

Os acessos do cliente ou do administrador do locadora ao servidor são com pedidos de informação ou escrita de dados de cada filme, sendo eles da seguinte forma:

1. Listar todos os títulos dos filmes e o ano de lançamento;
2. Listar todos os títulos dos filmes e o ano de lançamento de um gênero determinado;
3. Dado o identificador de um filme, retornar a sinopse do filme;
4. Dado o identificador de um filme, retornar todas as informações deste filme;
5. Listar todas as informações de todos os filmes;
6. Alterar o número de exemplares em estoque;
7. Dado o identificador de um filme, retornar o número de exemplares em estoque.

Todos esses acessos podem ser feitos em dois modos de consulta: um que realiza a consulta apenas uma vez, para testar sua corretude, e outro que realiza a consulta sequencialmente; cada acesso é repetido 30 vezes e todos os tempos são armazenados em um arquivo *log*, para possibilitar uma futura análise dos mesmos.

Classificamos esses acessos em 3 categorias: as consultas pequenas, consultas grandes e operações de escrita. De acordo com a consulta que cada operação realiza podemos dividi-las da seguinte forma: no primeiro grupo se encontram as operações de 1 a 4 e a operação 7, no segundo grupo temos apenas a operação 5 e no terceiro grupo temos a operação 6.

O primeiro grupo é caracterizado por uma consulta rápida de até 1500 bytes, que é o tamanho máximo MTU (Maximum Transmission Unit) que um pacote pode ter para ser transmitido pela camada Ethernet. O segundo grupo realiza consultas longas, com mais de 1500 bytes, que devem ser quebradas em mais de um pacote para serem enviadas. E o terceiro grupo é caracterizado por uma consulta que carrega um maior atraso do bando de dados, uma vez que no tempo medido está embutido o tempo de escrita no banco.

3 Armazenamento e Estruturas de Dados

Em um sistema real de cliente-servidor os dados são em sua maioria armazenados em Banco de Dados para garantir a persistência e a consistência dos mesmos. Com o objetivo de nos aproximar o máximo possível de um problema real, adotamos o MySQL [3] como o Banco de Dados para armazenar os dados da locadora de filmes no servidor.

Populamos o banco do servidor com dados de 10 filmes retirados do site IMDb [4], os quais estão classificados com as 10 melhores notas no mesmo. Optamos por uma quantia pequena de exemplares para que os tempos medidos nas operações não fossem totalmente influenciados pelo tempo de acesso ao servidor, visto que os tempos de comunicação são o foco da análise do projeto.

No acesso ao banco de dados, os clientes podem apenas acessar a informação, enquanto o cliente administrador da locadora pode, além do acesso, alterar a quantidade de exemplares em estoque. Para garantir esse privilégio de acesso, é necessário uma senha para entrar no modo administrador, a qual é processada pelo programa e só após confirmada o acesso diferenciado ao banco pode ocorrer.

Com o objetivo de fazer um sistema robusto, o lado do Cliente na aplicação desenvolvida também tem uma cópia do banco de dados. Esse banco local serve para verificar os resultados do servidor remoto: se a resposta do servidor for diferente da esperada (que já é conhecida devido ao banco de dados local), essa diferença também é salva pelo experimento.

4 Detalhes de Implementação

Tanto o cliente quanto a servidor foram implementados baseando-se nas descrições encontradas em [5]. A porta do servidor escolhida para comunicação foi a porta "35368".

Para o servidor, o socket de "escuta" é iniciado com o auxílio da operação *getaddrinfo()*; o tipo de socket é o SOCK_DGRAM, que caracteriza uma comunicação UDP.

O servidor entra em um loop no qual fica aceitando mensagens na ordem em que chegam, com a função *recvfrom*(). Ao receber a mensagem, a processa e envia a resposta com *sendto*(), para o mesmo cliente recebido pela chamada a *recvfrom*().

No cliente, seguimos a comunicação também como descrito [5]: o socket de comunicação também é criado com o auxílio de *getaddrinfo()*. Em ambos os casos, como a comunicação é feita em LAN, a função retorna apenas as informações de cliente e servidor diretamente, então não é necessário o loop como feito na literatura [5]. Como estamos trabalhando com apenas um servidor, no lado do cliente podemos usar a função *connect*() como se fosse comunicação TCP. O conceito é diferente, entretanto: enquanto no pro-

tocolo TCP o *connect*() enviava a mensagem de *hand-shaking* ao servidor, no UDP ela apenas salva localmente o destino e remetente de *send*() e *recv*(), respectivamente.

A função *recv*() do cliente, entretanto, é levemente diferente: como no UDP mensagens podem se perder, pode ser que a requisição ao servidor, ou sua resposta ao cliente, se percam na rede. Dessa forma, o cliente ficaria bloqueado esperando uma resposta para sempre. Usamos então em nossa vantagem a função *select*(), que nos permite usar um limite de tempo para esperar a resposta. Colocamos esse tempo como 3.5 s: se a resposta do servidor demorar mais que isso, consideramos que ela se perdeu na rede e partimos para a próxima requisição.

O cliente, a cada vez que é iniciado, joga na tela um *prompt* com as opções de requisição ao servidor. Em seguida, após escolhida a opção, o cliente dá a opção de fazer o acesso apenas uma vez (para verificar a corretude da aplicação) ou trinta vezes. Se o usuário escolher por fazer o acesso apenas uma vez, o resultado da consulta é exibido na tela do cliente. Se escolher por fazer a consulta trinta vezes, um arquivo de *log* de respostas é criado na mesma pasta do cliente, com o nome *testX.txt*, onde X é o número do teste escolhido pelo usuário no *prompt*.

A consulta é feita ao servidor pelo meio de envio de consultas *MySQL*, que são montadas no próprio programa do cliente. Essas consultas são enviadas como mensagens ao servidor que simplesmente as aplica ao seu banco de dados e retorna o resultado das consultas de leitura ou escrita, também em forma de mensagens. Essa mesma consulta é aplicada a um banco de dados local no cliente, que é exatamente igual ao do servidor; o resultado local dessa consulta é usado para verificar a confiabilidade do resultado enviado pelo servidor.

Esse arquivo de *log* gerado tem o seguinte formato: são trinta linhas e duas colunas, onde cada linha representa um dos testes. A primeira coluna representa o tempo total de comunicação entre cliente e servidor (desde imediatamente antes do *send*() até imediatamente depois do *recv*()); a segunda coluna traz um valor numérico da seguinte forma: se a mensagem recebida for igual à gerada no banco de dados local, ou seja, se a transmissão foi confiável e as mensagens chegaram corretamente, esse valor é zero. Qualquer outro valor diferente de zero significa que houve corrupção de mensagens.

O servidor também concatena a um arquivo de *log* dados sobre cada conexão que coordena. Como o servidor concatena informações num mesmo arquivo, quando o cliente fizer, por exemplo, trinta consultas de títulos de filmes, as últimas trinta entradas desse *log* do servidor guardarão os dados sobre essas consultas. Cada entrada nesse *log* é um *float* que guarda o tempo de acesso do servidor ao banco de dados; esse tempo é o intervalo desde imediatamente após o *recvfrom()* da consulta até imediatamente antes do *sendto()* do

resultado. Dessa forma, podemos calcular o tempo de transmissão das mensagens: usando o tempo *tServer* desse *log* do servidor e a segunda coluna do *log* do cliente (*tClient*), o tempo de transmissão de uma mensagem ao outro será dado por:

$$\text{Tempo de Transmissão} = \frac{tClient - tServer}{2} \tag{1}$$

Um outro pequeno detalhe de implementação é que apenas o cliente administrador da locadora deve ter autorização para alterar o número de exemplares em estoque de cada filme. Portanto, quando um cliente decide fazer essa alteração, é perguntado por uma senha. Essa senha é verificada no próprio programa do cliente (é hard-coded como "1234") e, apenas se estiver correta, o programa continua. Essa escrita no banco de dados também é hard-coded para garantir a consistência entre o banco de dados local e o remoto (do servidor), para que posteriormente possa ser verificada a confiabilidade das mensagens. Numa situação de mundo real, onde não há banco de dados local, essa opção não pode ser hard-coded.

O tamanho máximo de buffer adotado é de um milhão de caracteres; o tamanho máximo das consultas são 150 caracteres.

5 Análise dos Resultados

Toda a comunicação foi feita em LAN, com cliente e servidor conectados a um roteador D-Link DIR-600 por Wi-fi.

5.1 Tempo total de Comunicação

Em todos os resultados dos tempos médios utilizamos um intervalo de confiança com nível de confiança de 95%, obtido a partir da soma ou subtração de dois desvios padrão.

Temos nesta seção temos os resultados dos experimentos usando o protocolo UDP; temos também os resultados para os mesmos experimentos usando TCP, de um relatório feito anteriormente, e esses dados serão comparados.

Primeiramente geramos um gráfico com o tempo total de conexão com o servidor para cada uma das operações. O resultado segue demonstrado na Tabela **??** e na Figura 1.

Observe também na Tabela 1 e na Figura 2 os resultados de tempo total de conexão, desde o *connect*() até o *close*(), para o protocolo TCP, para os mesmos experimentos.

Os resultados que obtivemos para o tempo total de conexão dependem bastante das consultas ao banco de dados e do tamanho das mensagens. Essas mensagens nas operações pequenas, como já especificado, podem ser totalmente enviadas em apenas um pacote,

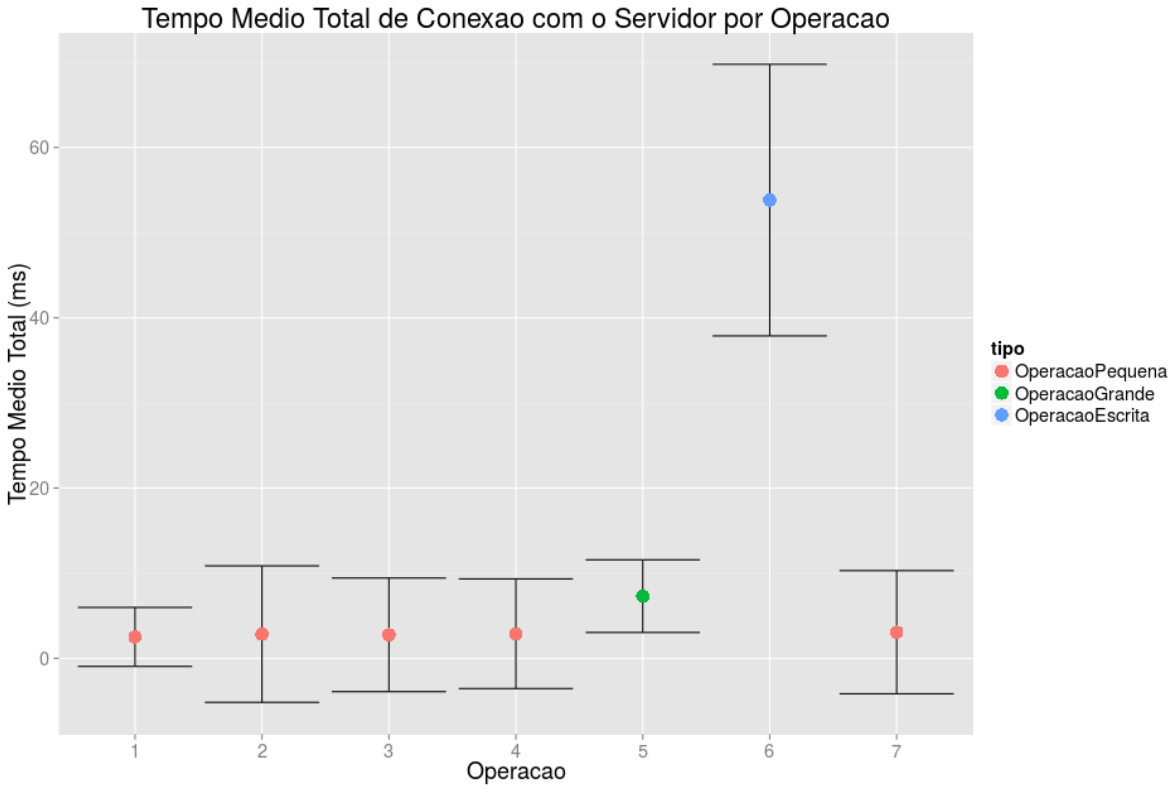


Figura 1: Mostra o tempo médio e o intervalo de confiança de 95% para as sete operações, na mesma ordem em que foram descritas, utilizando-se o protocolo UDP. Tempo calculado no cliente, o envio da mensagem ao servidor, até a recepção da resposta.

enquanto nas operações grandes, vários pacotes são necessários para o envio completo das mensagens. Nas operações de escrita, apesar de só um pacote ser necessário para o envio da mensagem, a consulta ao banco de dados para alteração requer um tempo bem maior de operação.

Com isso em mente, os resultados foram como esperado. Observe que tanto no TCP quanto no UDP as operações de escrita notoriamente tomam mais tempo que as outras. É notável também que as operações grandes levam um pouco mais de tempo que as pequenas - mas ainda têm tempos bem inferiores às de escrita.

O fato a se notar, entretanto, é a escala vertical dos gráficos: embora eles sejam parecidos, em todas as operações o TCP tomou mais tempo que o UDP. Em todos os experimentos realizados, essa afirmação mostrou-se verdadeira.

5.2 Tempo de Transmissão

Em seguida geramos um gráfico com o tempo total de transmissão das mensagens para cada uma das operações, como explicitado pela equação 1. O resultado para o UDP segue na Tabela **??** e Figura 3. Na Tabela 2 e Figura 4 temos os resultados para os mesmos

Tabela 1: Tabela mostrando o tempo médio e intervalo de confiança para cada uma das sete operações, na ordem em que foram descritas, para o protocolo TCP. Tempo calculado no cliente, desde a abertura até o fechamento da conexão.

Operação	Tempo Médio Total (ms)	Intervalo de Confiança (ms)
1	38.42	5.59
2	37.38	9.14
3	37.47	7.59
4	37.10	10.37
5	40.41	9.15
6	114.10	95.27
7	36.49	7.88

experimentos utilizando o protocolo TCP.

Os tempos de transmissão medem o tempo de comunicação entre o cliente e o servidor. Esse tempo deve depender apenas do tamanho das mensagens, que como já especificado, corresponde apenas a um pacote nas operações pequenas e de escrita e de vários pacotes nas operações grandes.

Os tempo obtidos para as operações pequenas e grandes condizem com essa expectativa, tanto no TCP quanto no UDP; no TCP, entretanto, as operações de escrita em alguns experimentos tiveram tempos de transmissão muito maiores (observe o alto desvio): acreditamos que isso se deva às aproximações no cálculo do tempo de transmissão. Para o UDP, entretanto, o tempo de transmissão foi totalmente dentro do esperado: operações pequenas e de escrita levaram quase sempre o mesmo tempo; operações grandes precisaram de um intervalo de tempo maior para serem transmitidas.

Note que os tempos de transmissão das mensagens UDP foram muito inferiores aos tempos do TCP; enquanto uma operação grande demorou cerca de 10 ms, em média, para ser transmitida no TCP, no UDP levou apenas 3 ms, em média.

Tabela 2: Mostra o tempo médio e intervalo de confiança de 95% para transmissão de mensagens segundo a equação 1 para cada uma das operações, na ordem em que foram descritas, utilizando o protocolo TCP.

Operação	Tempo Médio de Transmissão (ms)	Intervalo de Confiança (ms)
1	11.02	2.86
2	10.35	4.09
3	10.79	3.30
4	10.43	4.61
5	11.99	4.45
6	24.19	48.88
7	10.37	3.66

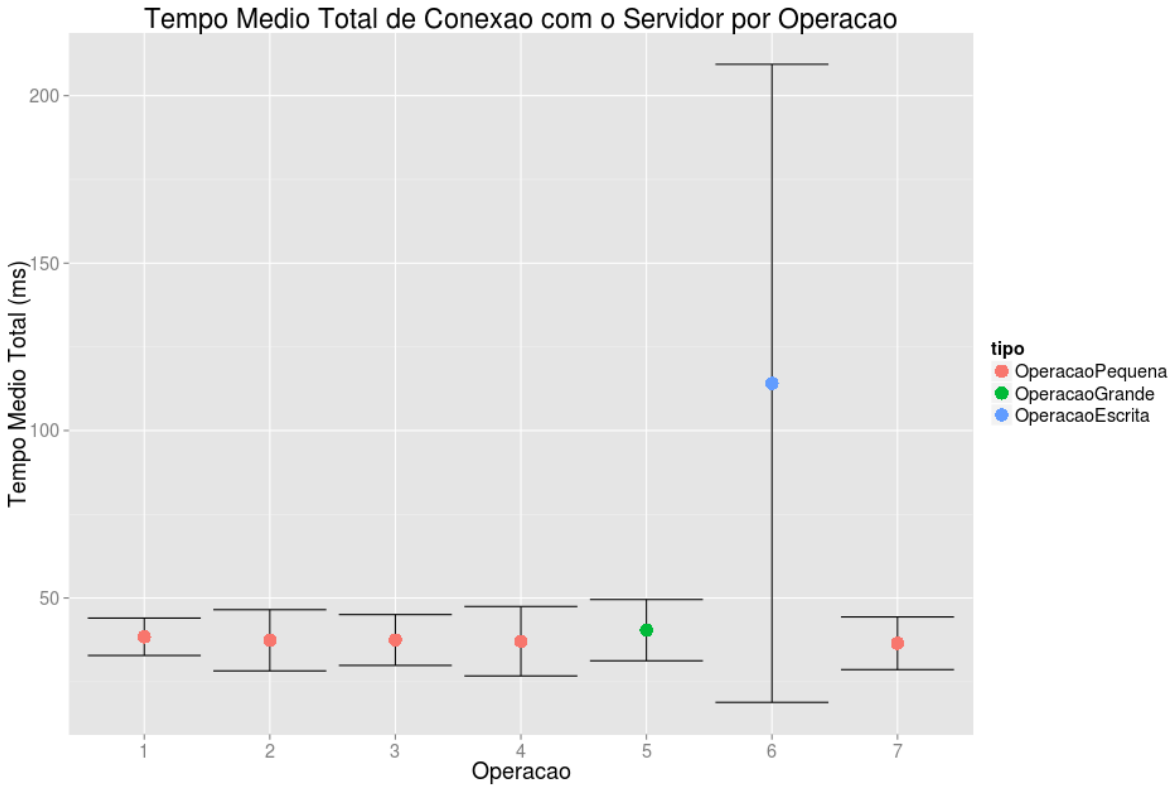


Figura 2: Mostra o tempo médio e o intervalo de confiança de 95% para as sete operações, na mesma ordem em que foram descritas, para o protocolo TCP. Tempo calculado no cliente, desde a abertura até o fechamento da conexão.

5.3 Tamanho de Código

As operações de redes são, na verdade, parte pequena do código total da aplicação. No servidor, a parte mais extensa do código são claramente as operações de consulta *MySQL*. No cliente, além das operações de banco de dados, há dezenas de linhas voltadas apenas para a saída na tela dos *prompts* ou a escrita em arquivo.

As operações de redes, mesmo no servidor que faz algumas operações a mais que o cliente, representam muito pouco do código. Considerando-se o servidor, as operações de redes mais os seus testes de erros somam 51 linhas, enquanto o arquivo todo tem 191 (correspondem a cerca de 26%). Para o cliente, as operações de redes representam 31 de um total de 342 linhas (cerca de 9%).

5.4 Confiabilidade

Como o protocolo escolhido foi o TCP, que garante confiabilidade na transmissão de mensagens, a confiabilidade esperada era de 100%. De fato, todas as consultas feitas foram exatamente iguais às geradas localmente para contra-prova, comprovando experimentalmente a confiabilidade do protocolo.

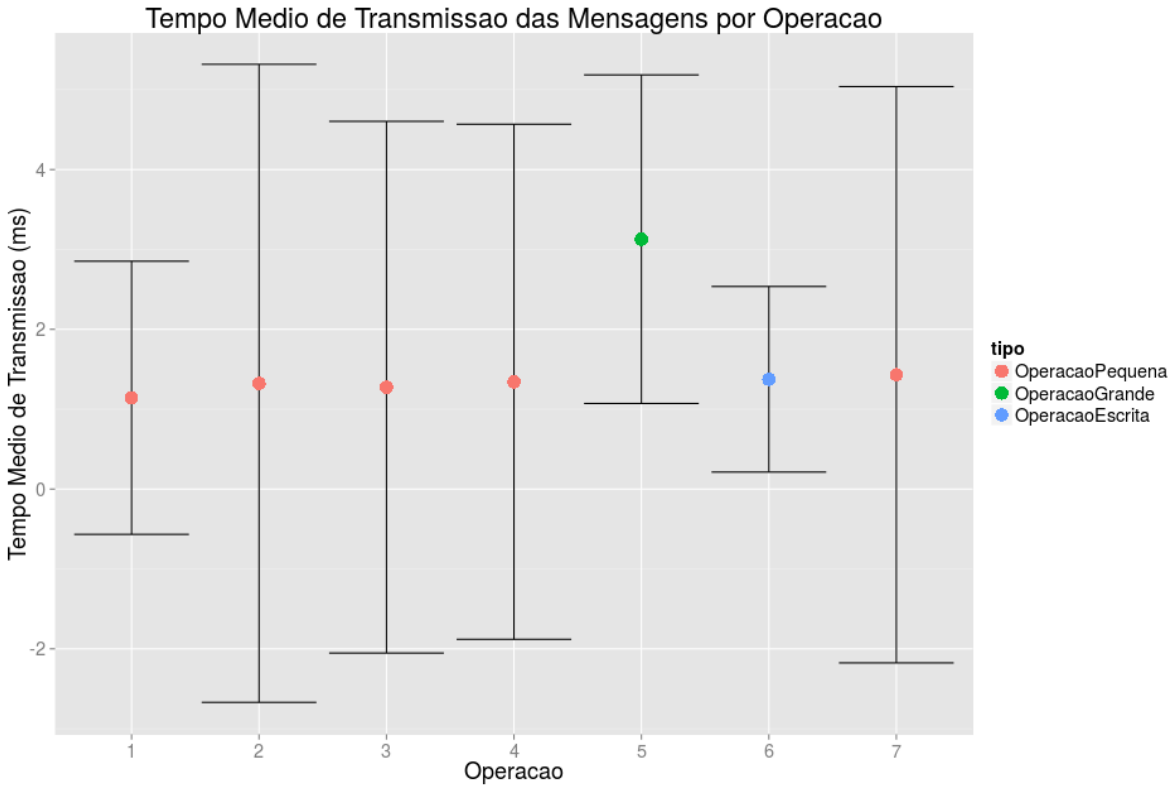


Figura 3: Tempo médio, com intervalo de confiança de 95%, calculado conforme a equação 1, para as sete operações na mesma ordem em que foram descritas, utilizando o protocolo UDP.

5.5 Nível de Abstração

As operações de redes fornecidas pelo sistema são bastante abstratas, apesar de algumas exigirem o entendimento de ponteiros e espaços de memória, meros detalhes da linguagem C. O "diálogo"entre cliente e servidor é resolvido com mensagens, e as operações são tão abstratas quanto imaginamos: baseiam-se em processos de envio (*send()* e *recv()*). Mesmo operações exclusivas a servidores, como *bind()* e *listen()*, embora não sejam tão instintivas, uma vez entendido o protocolo, fazem sentido e tem boa abstração.

Em suma, é possível perceber que toda a complexidade do protocolo está escondida atrás de operações simples.

6 Conclusões

No sistema cliente-servidor implementado neste projeto a informação que o servidor retém é a do seu banco de dados e todas as mensagens se baseiam em consultas *MySQL*. O tempo total está totalmente influenciado pelo tempo de acesso ao banco, como fica visível na operação de escrita.

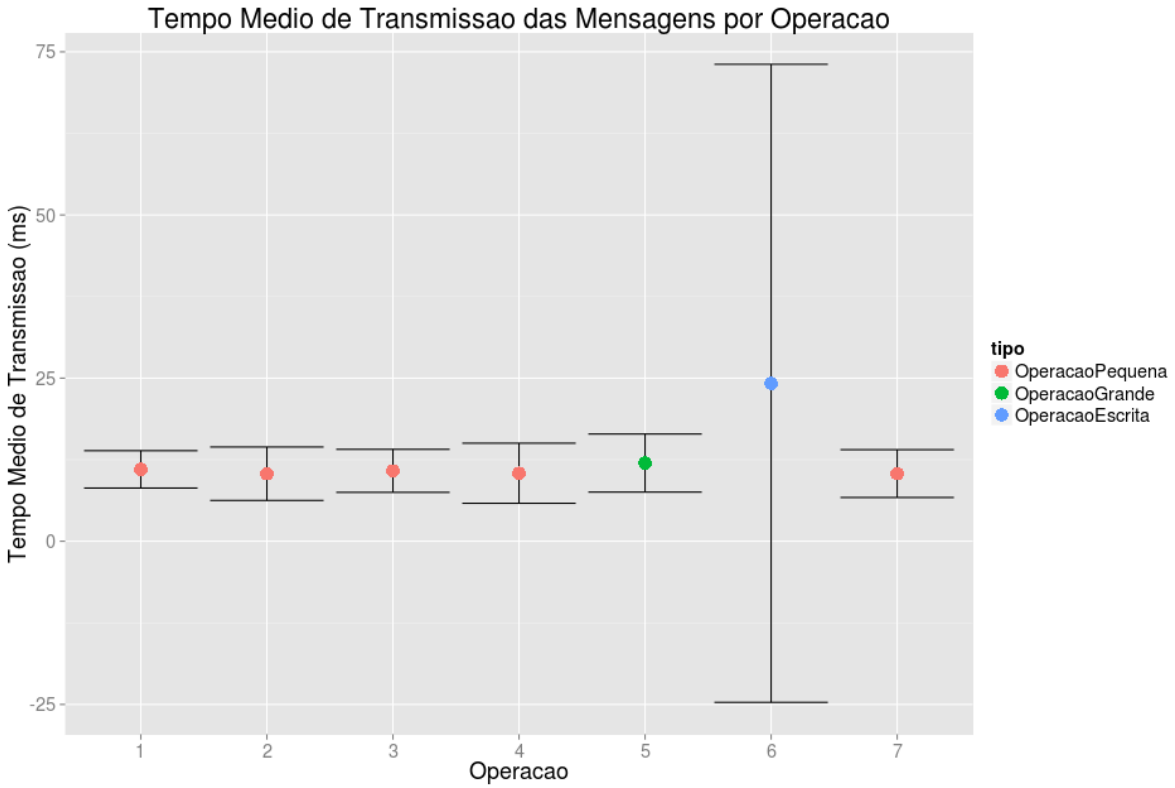


Figura 4: Tempo médio, com intervalo de confiança de 95%, calculado conforme a equação 1, para as sete operações na mesma ordem em que foram descritas, utilizando o protocolo TCP.

Devido a característica desta aplicação apenas uma pequena parte do código do sistema realiza as operações de redes. Apesar disso, essas operações são carregadas pela sua abstração, onde toda a complexidade esta embutida apenas em um comando básico. A abstração condiz com o tempo de execução dessas operações, medidos no tempo de transmissão, que equivale a menos de um quarto do tempo total na maioria das operações.

A confiabilidade garantida pelo protocolo TCP utilizado na comunicação do sistema foi comprovada em todas as consultas. Um futuro trabalho será comparar esses resultados com um sistema que utiliza o protocolo UDP, onde a confiabilidade pode não ser tão perfeita.

Referências

[1] Jon Postel. Transmission control protocol. 1981.

[2] Jon Postel. User datagram protocol. *Isi*, 1980.

[3] AB MySQL. *MySQL: the world's most popular open source database*. MySQL AB, 1995.

[4] Internet Movie Database. Top 250 on imdb: Highest-rated movies based on votes by imdb users., 2015.

[5] Brian Hall. Beej's guide to network programming using internet sockets, 2012. *URL http://beej.us/guide/bgnet/output/html/multipage/index.html*.