MC833 Programação de Redes de Computadores

Projeto 1 - Socket Servidor e Cliente

Carlos Robson ra135257

Introdução

Neste trabalho foi implementado um servidor e um cliente que se comunicam através de sockets TCP (Transmission Control Protocol). Este é um protocolo utilizado para a transmissão segura de dados entre dois computadores.
O TCP é um protocolo confiável, pois reúne garantia de transmissão ordenada, retransmissão de pacotes perdidos, controle de fluxo e controle de congestão
Essa comunicação é concorrente, ou seja, o servidor pode atender mais de um cliente ao mesmo tempo. Caso ele receba uma requisição de um cliente enquanto processa a requisição de outro cliente, o servidor faz um fork() e passa a tratar as duas requisições paralelamente, sem congelar uma delas.

Sistema

O trabalho simula um sistema acadêmico, que possui uma base de dados com informações de várias disciplinas e permita a realização de algumas operações sobre ela, como consultas e escritas.A operação de escrita de comentário só é permitida pelo professor. As operações estão listadas abaixo com seus devidos comandos:

1 - listar todos os códigos de disciplinas com seus respectivos títulos;
2 - listar todas as informações de todas as disciplinas;
3 - dado o código de uma disciplina, retornar a ementa;
4 - dado o código de uma disciplina, retornar todas as informações desta disciplina;
5 - dado o código de uma disciplina, retornar o texto de comentário sobre a próxima aula
6 - escrever um texto de comentário sobre a próxima aula de uma disciplina (apenas usuário professor);

Armazenamento e estruturas de dados do servidor

Os dados do servidor são armazenados diretamente em um arquivo dataAccess.c, com lista de estruturas definidas em base.h. Isso permite controlar o acesso aos dados mais fácilmente.
Os campos guardados de cada disciplina são: id, title, program, schedule, commentary.

Detalhes da Implementação

Durante a implementação foi focada na conexão entre os servidor e cliente usando o padrão a biblioteca <sys/socket.h>, procurando tratar erros de conexão, leitura e escrita. Por esse motivo, procuramos simplificar ao máximo o sistema. Ao invés de um banco de dados relacional, foi criada uma base de dados simples.

Nossa implementação foi de um servidor TCP concorrente rodando sobre IP versão 4 (IPv4). Utilizamos a biblioteca para importar as funções socket(), connect(), send(), recv(), bind(), listen(), accept() e close(), além das estruturas de dados sockaddr_in (endereço para IPv4) e socklen_t (tamanho de endereço). O projeto foi produzido utilizando as seguintes ferramentas: sublime, gcc, netstat, linux shell e google sheets para gerar os gráficos dos dados.

Para simplificar o script que rodava os vários pedidos,os comando foram usados como inteiros. Ao rodar um cliente, o usuário deve inserir a comando de 1 a 6, dependendo do comando é solicitado os parâmetros da requisição que serão enviados ao servidor.

Para cada requisição que o usuário faz é repetida 30 vezes, para pegar os dados do tempo que a requisição demora. Tais tempos são subtraídos do tempo de execução do servidor, a fim de avaliar unicamente os tempo de transporte das requisições. Os testes foram feitos em ambiente LAN, sendo o servidor conectado via Wifi e o client de forma cabeada.

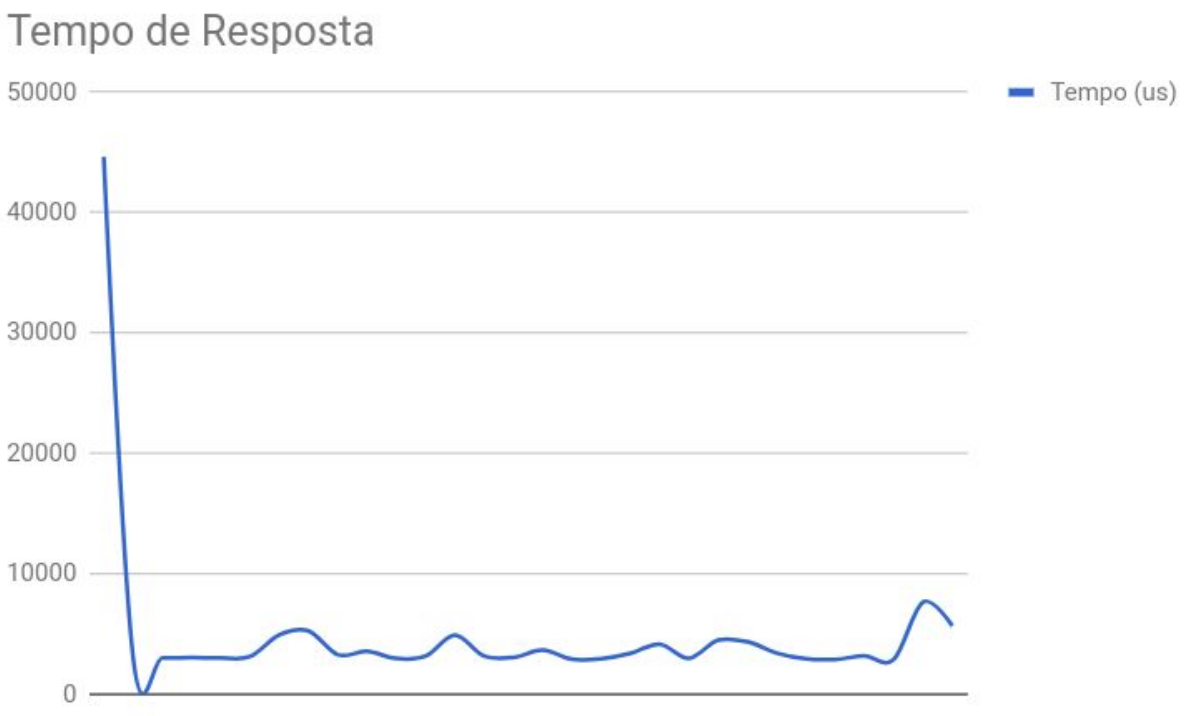
Gráficos e análise de resultados

Foram realizados testes para cada caso de uso. Os tempos médios de comunicação de cada caso, bem como desvio padrão e intervalo de confiança constam na tabela abaixo:

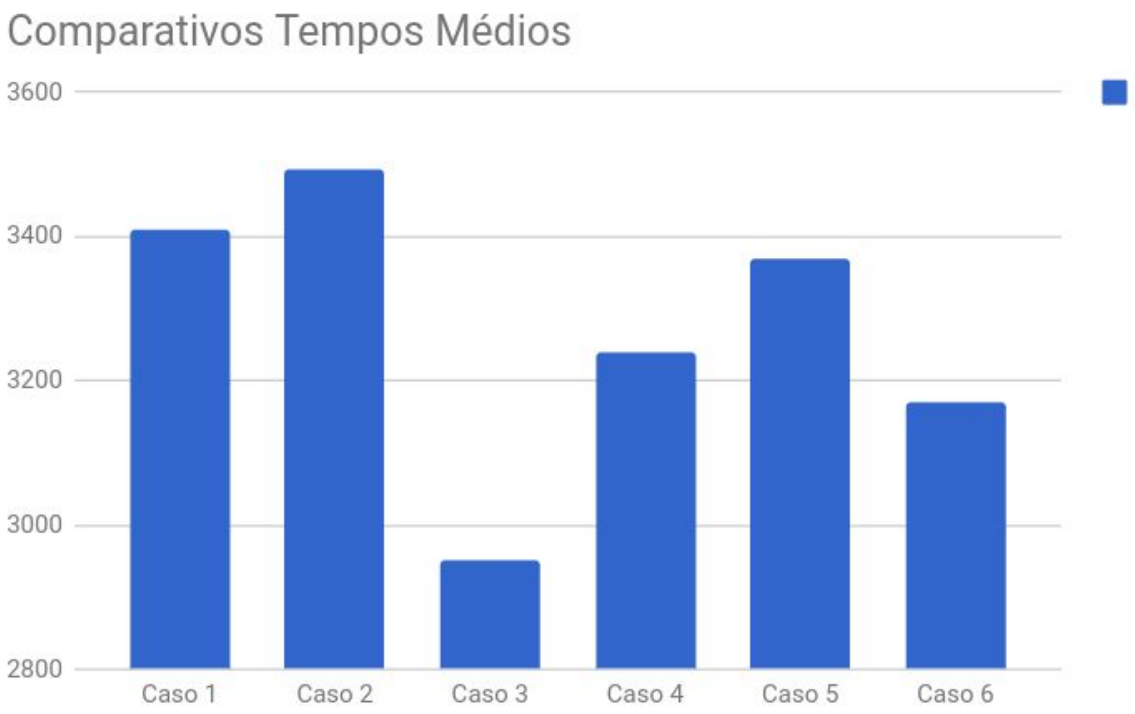
	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
Média	3409.5	3492	2950.5	3240.5	3370	3171.5
Desvio	2813.9	2125.4	6907.7	880.8	4878.6	7557.1
Confiança	9219.8	4252.5	5422.4	3555.7	5115.8	5875.8

Como vemos, em geral os valores da média são próximos, mas os desvios padrões podem ser bem grandes, isso ocorre principalmente devido aos valores de tempo da primeira requisição que em geral é bem maior. Esses valores maiores são ocasionados devido a erros na rede, principalmente na wifi onde o servidor estava funcionando.

No gráfico a seguir que representa o caso 6, onde há maior desvio, podemos ver como ocorrem os tempos dos testes:



Os tempos médio de comunicação de cada caso podem ser comparados no gráfico a seguir:



Conclusão

Analisando os dados obtidos, é possível concluir que os tempos de comunicação se mantém num mesmo intervalo, de forma geral. Também é possível observar que, devido à simplicidade do acesso ao banco, as diferenças de tempo de comunicação dependem basicamente da velocidade de conexão entre servidor e cliente. Atrasos na rede acontecem com uma pequena frequência, podendo ser causadas por vários motivos, como congestionamento na rede, baixa taxa de transmissão da rede, perda de conexão (que ocasiona perda de pacotes), entre outros. Apesar disso, a maior parte das transmissões ocorrem com sucesso e em um tempo bom.

Referencias

https://linux.die.net/man

Beej's Guide to Network Programming

STEVENS R. W., FENNER B., RUDOFF A. M.: UNIX Network Programming The Sockets Networking APIs, Vol. 1, Third Edition, Person Education, 2004