Universidade Federal de Minas Gerais
TP 02 - Redes de Computadores [2022/2]

Augusto Carvalho Porto Pereira - 2019054358

Introdução

O trabalho prático 2 da disciplina de Redes de Computadores teve como objetivo exercitar e colocar em prática o conhecimento acerca de comunicação entre aplicações em rede. No caso, várias aplicações cliente e uma aplicação servidor deveriam se comunicar via rede para cumprirem seus objetivos.

Para garantir que essa comunicação funcione de forma correta entre as aplicações, mesmo que elas estejam sendo executadas em uma mesma máquina, é necessário que analisemos cautelosamente todo o aprendizado relacionado a protocolos de comunicação, portas, endereços, conexões além de outros temas centrais da disciplina.

Embora realizando ações individualmente simples como adicionar e remover itens de uma lista (no caso, manuseando vários equipamentos que se comunicam por intermédio de um servidor), o trabalho possui seu real valor no ensino prático das teorias de redes de computadores para as linguagens de programação.

Arquitetura

A solução do problema foi realizada utilizando uma estrutura modesta, com cerca de 600 linhas de código divididos em 3 arquivos da linguagem C, sendo eles um servidor (server.c), um cliente-equipamento (equipment.c) e um arquivo contendo funções comuns para ambos (common.c).

Toda a parte de conexão entre equipamentos e o servidor foi majoritariamente tomada como base no código exibido e escrito pelo professor Ítalo Cunha na playlist de Introdução à Programação em Redes disponível na plataforma moodle, no qual estão explicados individualmente qual a responsabilidade de cada método e variável do código. Sem esta aula rica, acredito que seria complexo a construção a partir do zero destas aplicações. Em especial, a explicação guiada sobre como utilizar de threads para permitir múltiplas conexões foi vital para esse trabalho prático em comparação ao anterior, visto que esta era a maior transição entre eles.

Servidor

O servidor foi construído para centralizar os equipamentos interconectados utilizando do protocolo construído para trocar mensagens com cada um dos equipamentos, que frequentemente precisarão comunicar entre si. Ao adicionar um equipamento, por exemplo, o servidor recebe a mensagem do novo equipamento

conectado e envia a todos, inclusive ao novo equipamento, o código da nova instalação. Para realizar a comunicação com todos os clientes envolvidos, o servidor utiliza de multicast desenvolvido manualmente, visto que o protocolo TCP não permite broadcast na rede e não valeria a pena na construção gerar dupla implementação adicionando também o UDP. A interação com múltiplas conexões simultâneas é feita por uso de multithread, com uma execução em paralelo para cada novo equipamento.

Equipamento

Por sua vez, o código criado para os equipamentos utiliza em cada execução duas threads paralelas, uma para ativamente receber mensagens do servidor e trabalhar sobre elas e outra para ler o prompt de comando que recebe informações do usuário e realiza as interações necessárias com isso. Por exemplo, o comando de fechar a conexão, close connection, gera uma mensagem para o servidor que por sua vez propaga para outros equipamentos, enquanto o comando list equipments não gera interações no servidor e somente exibe no terminal as informações do banco de dados daquele equipamento.

Discussão

A realização do trabalho conforme a especificação apresentada gerou desafios guiados pelo processo de decisão da melhor maneira de definir o que e como os equipamentos e o servidor se comunicam. Além disso, lidar com o trade-off entre armazenar informações redundantes em cada equipamento ou centralizá-las no servidor e aumentar a quantidade de mensagens trocadas possui em ambos os casos seus pontos positivos e negativos. Um dos pontos facilitadores para evitar conflitos de informações em cada sistema foi o uso de códigos em todas as mensagens do protocolo, que seriam resolvidos centralizando constantes como o significado dos códigos das requisições, as mensagens de erro e sucesso e outros pontos no arquivo common.

Uma das maiores dificuldades do trabalho, que projetam dificuldades reais vivenciadas em projetos que lidam com redes, foi realizar verificações de erros que são muito dificilmente testáveis, como por exemplo verificar erros de máquinas que, por algum motivo, não tiveram seu id corretamente assimilado, ou então garantir que as mensagens chegaram de forma correta. Por exemplo, duas das quatro mensagens de erro não poderiam ser testadas por simples mensagens do terminal de comando, exigindo alterações temporárias no código para assegurar o funcionamento correto das checagens.

Conclusões

Ao final do trabalho é possível avaliar que houve um grande aprendizado acerca de programação em redes na linguagem C e principalmente sobre de que forma

aplicações diferentes comunicam mensagens de forma sincronamente, embora seja possível relacionar também o conhecimento adquirido para tomar como base na análise de sistemas mais complexos que utilizem, por exemplo, comunicação assíncrona, multithreads, entre outros avanços tecnológicos do ferramental de desenvolvimento. É interessante também perceber que alguns problemas experienciados no processo de testes e implementação como erros que encerravam a aplicação servidor sem liberar o endereço para que ela pudesse ser executada novamente ou a finalização de somente uma das aplicações sem que a outra tenha finalizado sua ação fizeram ser compreendidos os motivos pelos quais os protocolos são tão densos e repletos de validações e confirmações necessárias.

Desenvolver, com a mão na massa, aplicações similares às que usamos diariamente sem nos preocuparmos com como ocorrem por debaixo dos panos se mostrou uma experiência interessante e enriquecedora garantida pela disciplina.

Referências

- https://www.youtube.com/playlist?list=PLyrH0CFXIM5Wzmbv-IC-qvoBejsa803Qk
- https://stackoverflow.com/
- https://en.cppreference.com/w/c
- https://www.ibm.com/docs/en/i/7.4
- https://cboard.cprogramming.com/c-programming/