UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Departamento de Ciência da Computação
DCC023 -- Redes de Computadores
Trabalho Prático 1 - Servidor de Mensagens Publish/Subscribe
Guilherme de Abreu Lima Buitrago Miranda - 2018054788

1 - Introdução

O trabalho prático 1 da disciplina de Redes de Computadores tem como objetivo a criação de um servidor e clientes para troca de mensagem seguindo o padrão publish-subscribe, de funcionamento semelhante à plataforma Twitter. Neste padrão, clientes enviam para o servidor suas tags de interesse e, quando algum dos clientes envia uma mensagem contendo uma dessas tags, o servidor faz o encaminhamento para os clientes interessados.

Abaixo, encontram-se detalhes relativos à implementação do trabalho, desafios enfrentados, conclusões e referências bibliográficas.

2 - Implementação

A primeira parte da implementação deste trabalho deu-se acompanhando a aula de Introdução à Programação em Redes disponível no YouTube. Com isso, foi possível desenvolver um servidor que, por meio de threads, consegue receber mensagens de diferentes clientes, independente da ordem em que estes enviam as mensagens.

Posteriormente, foi necessário preparar o servidor para diferenciar uma mensagem comum de notificações de interesse ou desinteresse em tags. Para tal, foram implementados uma série de métodos de manipulação de strings, a fim de observar caracteres como “+”, “-” e #”. Além disso, em um momento futuro, foi também observada a necessidade de implementação de um método para tokenizar a mensagem procurando por quebras de linha (“\n”), pois uma chamada ao método recv pode conter mais de uma mensagem.

Em seguida, mostrou-se fundamental desenvolver uma forma de armazenar as tags, assim como quais usuários estavam inscritos na tag em questão. Com esse objetivo, foram criados os arquivos userList e tagList. Neles, são implementados métodos importantes para a manipulação de estruturas de dados em C, como a alocação de memória, tamanho da lista e destrutor. Além disso, também são implementados métodos de busca por uma tag ou usuário, assim como adição e remoção dos mesmos.

Com as estruturas criadas, era hora de implementar a lógica do protocolo propriamente dito. Conforme pode ser visto no arquivo “servidor.c”, a mensagem é processada e são verificados alguns casos especiais, como mensagens com caracteres inválidos ou longa demais. É também observado se a mensagem é “##kill”, que é a mensagem para fechamento de todas as conexões e terminação da execução.

Logo após, se é uma tag de adição (e o usuário ainda não a adicionou), acresce-se o usuário na lista de interessados na tag. Do contrário, verifica-se se é uma mensagem demonstrando o desinteresse em uma tag. Caso positivo, testa-se se o usuário está inscrito e, se possível, realiza-se a retirada do mesmo da lista da tag. Evidentemente, casos em que é demonstrado o interesse numa tag em que o interesse está ativo e casos em que o usuário se desinteressa numa tag que não estava previamente interessado também são tratados.

Se não se tratar de nenhum desses casos, a mensagem é processada como uma mensagem comum. Assim, vasculha-se pelo caractere “#”, que representa o início de uma tag. Tais tags são separadas e, para cada uma delas, a mensagem é replicada aos interessados. Nesse momento, é também verificada se a atual mensagem já foi enviada para um cliente hipotético x, a fim de que o mesmo não receba uma mensagem duplicada caso a mesma tenha duas tags e ele esteja interessado nas duas.

Por fim, a implementação do cliente foi necessária. Além dos métodos já implementados durante a aula de Introdução à Programação em Redes, a grande adição a ser feita foi a criação de uma thread para ouvir as mensagens enviadas pelo servidor, usando a interface POSIX.

3 - Desafios

Dentre os principais desafios enfrentados durante o trabalho, destacam-se: a criação de estruturas de dados na linguagem C utilizando alocação dinâmica; a gerência de threads e a manipulação de strings em C.

O primeiro desafio enfrentado durante a implementação foi a manipulação de strings em C, que era necessário para identificar se uma mensagem era um pedido de inclusão de tag, de exclusão de tag, de terminação da execução do servidor, inválida ou, por fim: uma mensagem comum. Para tal, métodos da biblioteca padrão foram empregados, como o strcmp e strlen. Além disso, utilizou-se a comparação com caracteres com base na tabela ASCII para determinar a validade das mensagens. Não obstante, como posteriormente foi observado que mais de uma mensagem poderia ser lida com apenas um comando recv (ex: “boa tarde #MaisUmDia bom almoço #DiarioAlimentar\n”), pesquisou-se e encontrou-se o comando strtok para tokenização do conteúdo lido, separando-o em mensagens válidas.

Com relação à criação de estruturas de dados em C utilizando alocação dinâmica, a dificuldade se deu principalmente no debug do programa. Algumas segmentation faults foram comuns e, por vezes, houveram problemas para não se perder as referências dentro das listas encadeadas criadas. Felizmente, com alguma pesquisa e o auxílio do Valgrind, todos os erros percebidos foram corrigidos e métodos como o de busca funcionam bem.

Por fim, após compreender os conceitos da gerência de threads, o trabalho funcionou como esperado. Em particular, estudar a interface POSIX elucidou diversos pontos nebulosos com relação a seu funcionamento e, implementá-las em código não se mostrou uma tarefa atribulada.

4 - Conclusão

Para além da prática com a linguagem C, a implementação do trabalho foi bastante proveitosa para compreender efetivamente os conceitos vistos até o presente momento na disciplina. O aprendizado se deu, sobretudo, no último desafio citado na seção anterior, mas também outras dúvidas dos colegas apresentadas no fórum de discussões. Com isso, considero que o trabalho foi de grande proveito para o processo de aprendizagem.

5 - Referências Bibliográficas

- Introdução à Programação em Redes - https://www.youtube.com/playlist?list=PLyrH0CFXIM5Wzmbv-IC-qvoBejsa803Qk
- Introdução ao POSIX Socket API - https://edisciplinas.usp.br/pluginfile.php/5347724/mod_resource/content/2/76-ProgSockets-v13.pdf
- C library function - strtok() - https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm