Trabalho Prático 2 da disciplina de Redes

Lorenzo Carneiro Magalhães - 2021031505

Belo Horizonte, 06 de janeiro de 2025

1. Introdução

Nesse trabalho prático, foi desenvolvido um sistema de comunicação via sockets em um esquema produtor-consumidor. Nesse cenário, um cliente possui um sensor e envia mensagens de registros do sensor para o servidor, paralelamente, o sensor recebe atualizações de outros sensores via servidor e atualiza a medição do próprio sensor com base nos adjacentes. O servidor tem o papel de receber os registros dos sensores e fazer um broadcast desses registros para os sensores do mesmo tipo que o sensor remetente.

2. Implementação

A implementação desse trabalho foi mais desafiadora do que a do trabalho anterior, pois exigiu o uso de threads e mutexes, funcionalidades com as quais geralmente não tenho muita proficiência.

A implementação seguiu um fluxo linear, em que o funcionamento dos sockets, especialmente o servidor multi-threaded, foi priorizado inicialmente, e depois o desenvolvimento concentrou-se em implementar o sistema de atualização de medição dos clientes sensores e por final no desenvolvimento do sistema de comunicação inteligente do servidor.

Além disso, ocorreu uma última fase de correção de bugs e formatação de saídas.

2.1. Construção dos sockets

A construção base dos sockets foi totalmente implementada a partir do vídeo do professor Ítalo, o qual foi colocado como uma das referências do trabalho. Dessa maneira, não houve grandes dificuldades na implementação básica dos sockets.

Essa implementação do servidor multi-threaded serviu como base para minhas outras funcionalidades que necessitavam de mais threads.

2.2. Funcionalidades do cliente

O cliente sensor tem duas funções:

1. enviar registros do próprio sensor para o servidor
2. receber registros de sensores via servidor

Esse funcionamento faz com que seja necessário duas threads, uma para cada função. Além do funcionamento padrão, é necessário uma comunicação simples entre as threads, pois a thread 1 necessita conhecer a atualização do valor do sensor computada em 2 para enviar para o servidor.

A thread 2 recebe todos os registros de sensores do mesmo tipo via servidor e atualiza a medição com base nos registros adjacentes e deve informar a thread 1 da nova atualização na medição interna, de maneira que a thread 1 sempre envie o valor de medição correto.

Na thread 2 também foi implementado o tratamento dos casos em que as coordenadas são iguais, ou seja, sensores sobrepostos, e em que sensores do mesmo tipo são removidos.

A implementação do sistema de atualização da medição baseado nos 3 sensores vizinhos mais próximos foi feita de maneira simples. O ideal seria implementar através de um *min-heap*, mas decidi implementar através de uma busca em um vetor de vizinhos pela facilidade de codificação.

Os logs também são feitos pela thread 2.

2.3. Funcionalidades do servidor

Após implementar o cliente, a lógica de implementar o servidor ficou mais fácil e rápida, pois parte do código pôde ser aproveitado.

A lógica do servidor é muito similar porém mais simples. O servidor possui um loop principal de aceitação de novas conexões e cada nova conexão acarreta a criação de uma thread específica para essa conexão.

Essa thread, que recebe o struct *client_data*, que contêm o socket e uma mensagem *sensor_message*, fica em um loop próprio de recebimento de mensagens e seu consequente *broadcast* para sensores do mesmo tipo que o remetente da mensagem recebida, além do registro das mensagens.

Esse controle do envio de mensagens é definido no mesmo arquivo. A implementação desse sistema assume 3 partições que representam cada tipo de sensor: temperatura, umidade e qualidade do ar.

Para uma abordagem mais generalista, seria interessante definir as partições por meio de vetores, mas por facilidade de implementação decidi explicitar no código cada partição e seus respectivos ponteiros e índices.

Cada partição é um vetor do struct *client_data* de tamanho 20, máximo de clientes associados, que armazena os sensores do tipo específico que estão atualmente conectados ao servidor.

A partir desse vetor, criei funções de adicionar e remover novos clientes de acordo com o tipo, realizar o *broadcast* de uma mensagem pela partição e verificar se um sensor identificado dinamicamente já foi registrado.

É importante ressaltar que a identificação e diferenciação de sensores ocorre por meio do número do socket. Isso confere a possibilidade de diferenciação de sensores sobrepostos pelo servidor.

2.4. Correção de erros

Quando a integração base foi concluída, comecei a tratar os erros encontrados. Em sua maioria eram relacionados ao alinhamento da ordem dos processos do meu programa com a especificação e à formatação de registros.

Em geral, não houve muitos problemas relacionados com os sistemas em si, o que agilizou a implementação completa.

3. Conclusão

O trabalho foi executado com sucesso apesar da maior complexidade do trabalho.

Por se tratar de um documento curto, não é possível detalhar todas as decisões de implementação, mas todas as decisões importantes foram discutidas.

Por fim, achei muito interessante trabalhar com sockets e threads em conjunto e ressalto que a aula do Ítalo foi essencial para a realização do trabalho.