Trabalho Prático 1 – Redes de Computadores

Servidor de mensagens – Sistema Supervisório

Gilliard Gabriel Rodrigues
Matrícula: 2019054609

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

gilliard2019@ufmg.br

1. Introdução

O problema proposto foi implementar um sistema supervisório responsável por coletar e armazenar dados dos equipamentos durante processos de produção. Os componentes do sistema são:

- Sensores: responsáveis por mensurar dados de variáveis de processos e enviá-las para a estação remota.
- Rede de comunicação: infraestrutura responsável por prover comunicação entre os nós da rede.
- Estação remota (o servidor): atende às solicitações da central de monitoramento e se comunica diretamente com os sensores distribuídos na indústria.
- Central de monitoramento (o cliente): responsável por gerenciar (adicionar e remover logicamente) os sensores e por solicitar informações deles.

Conforme disponibilizado na especificação do TP, o sistema supervisiona apenas 4 tipos de variáveis e existem 4 equipamentos. A tabela a seguir detalha melhor:

Sensor	SensorId
Temperatura	01
Pressão	02
Velocidade	03
Corrente	04

Tabela I - Sensores e seus identificadores

Equipamento	equipmentId
Esteira	01
Guindaste	02
Ponte Rolante	03
Empilhadeira	04

Tabela II - Equipamentos e seus identificadores

A seguir serão apresentadas as soluções e discutidos os principais desafios e dificuldades durante a implementação.

2. Implementação

No que diz respeito à base do sistema cliente-servidor, a lógica seguida foi a mesma ensinada pelo professor Ítalo Cunha durante as aulas de Introdução à Programação em Redes, disponibilizadas no Moodle. Portanto, não houveram dificuldades em fazer ambos os lados (cliente e servidor) se comunicarem e nem de alternar entre os protocolos IPv4 e IPv6.

As maiores preocupações nesse sentido foram de alterar a lógica de forma que o cliente fosse capaz de continuar enviando mensagens após a conexão ocorrer, sem que ela finalizasse após o envio da primeira mensagem e que a conexão se encerrasse apenas com o comando '*kill*'.

A estrutura de dados utilizada para representar os sensores e equipamentos foi uma matriz booleana 4x4, em que as linhas representam os equipamentos e as colunas, os sensores, tal que se a posição *[i][j]* possui o valor *true* significa que no equipamento de id *i+1* está instalado um sensor de id *j+1* (soma-se 1 para obter o id correto, já que a posição em matrizes começa em 0). Tanto a matriz quanto um contador relativo à quantidade de sensores disponíveis para instalação são variáveis globais.

As próximas seções irão tratar das principais particularidades do sistema, ou seja, as implementações das operações – instalar sensor(es), remover sensor(es), consultar equipamento e consultar variáveis de processo – e os respectivos tratamentos de erros. Essas implementações foram

feitas no arquivo *server.c*, que também conta com uma função que avalia o tipo de comando a partir da entrada no terminal, chamando a função correta para cada caso (*add*, *remove*, *list* ou *read*).

2.1 Instalar sensor

A primeira função que implementa uma das quatro operações chama-se *instalarSensor*, que recebe um ponteiro de *char* e um inteiro representando o socket do cliente. Para avaliar cada parte da entrada do terminal, foi utilizada a função *strtok* da biblioteca <*string.h*>. No corpo da função são inicializadas algumas variáveis relativas à string que guardará a mensagem que será retornada no terminal, o ponteiro de *char* que navega na entrada do usuário, algumas flags referentes ao limite de sensores e a entradas inválidas, assim como um contador de sensores e um array que guarda os id's fornecidos na entrada. Enquanto o ponteiro não apontar para o comando "in" e a entrada ainda for válida (ela é validada pela função *validaId*, que retonar -1 caso o id do sensor ou equipamento inserido não contiver 2 algarismos e não for um id entre 01 e 04), os id's dos sensores são convertidos para inteiros e armazenados em um *array*. Em seguida, se a entrada ainda for válida, a função testa se o id do equipamento é válido e preenche a matriz booleana na(s) posição(ões) relativa(s) aos id's dos sensores informados e no equipamento correto. Há tratamento de erros para os seguintes casos: sensor(es) inválido(s), equipamento inválido, limite máximo de sensores a ser instalados alcançado e sensor repetido. Ao final da função, o servidor envia a mensagem para o cliente.

2.2 Remover sensor

A estrutura da função relativa à operação de remover sensor, cujo nome é *removerSensor*, é muito parecida com a da função *instalarSensor*. As principais diferenças são que ao invés de *settar* como *true* a posição relativa ao sensor em questão e decrementar o contador de sensores disponíveis, ocorre o oposto. Assim como ao invés de lançar a mensagem de erro referente a algum sensor duplicado, é lançada a mensagem referente a algum sensor que não existe no equipamento em questão.

2.3 Consultar equipamento

A função relativa à operação de listar sensores de um equipamento chama-se *consultarEquipamento* e é bem simples também. Além da lógica de percorrer a entrada do usuário, há uma verificação se o id do equipamento é válido e se for, itera na linha referente a ele na matriz, concatenando os id's dos sensores instalados numa *string* e enviando ela na mensagem no final.

2.3 Consultar variável de processo

A função relativa à operação de ler um ou mais sensores de um equipamento chama-se *consultarVariavelDeProcesso* e tem uma estrutura parecida com as funções de instalar e remover sensor, com a diferença de que ao invés de instalar ou remover um sensor após passar pelas validações, ela chama a função *gerarLeituraSensor*, que retorna um valor aleatório com 2 casas decimais após a vírgula referente à leitura de um sensor, concatenando esse valor à mensagem que será enviada ao cliente. A validação exclusiva daqui trata do caso em que o sensor cuja leitura é requisitada não está instalado no equipamento em questão.

3. Conclusão

Os pontos mais importantes para conseguir resolver o trabalho foram, sem dúvidas, aprender a lidar com programação em redes usando sockets, pois essa é a base do TP, aprendizado que foi facilitado pela playlist disponibilizada para os alunos, assim como se familiarizar com as funções da biblioteca <*string.h*> a fim de manipular a entrada do usuário de acordo com as regras de negócio do trabalho.

Após se familiarizar com programação em redes, o maior desafio foi implementar e fazer funcionar a função que instala sensores, assim como implementar as validações necessárias. Tendo feito isso, implementar as outras funções se tornou um processo mais fácil, devido às suas similaridades.

Em suma, foi possível praticar os conceitos de programação em redes vistos em aula e consolidar o aprendizado novo.

Referências

CUNHA, Ítalo. Playlist de Introdução à Programação em Redes. Departamento de Ciência da Computação. Universidade Federal de Minas Gerais. 2022.