Universidade Federal de Minas Gerais
Redes de Computadores - DCC023
Trabalho Prático - Sockets
Luan Alves Oliveira - 2019027008

Introdução

O objetivo deste trabalho foi criar um sistema supervisório responsável por coletar e armazenar dados dos equipamentos durante os processos de produção. Desta forma, foi implementado um sistema com o protocolo TCP no modelo cliente/servidor, com possibilidade de utilização em IPv4 e IPv6. Foram implementadas quatro tipos de mensagens que podem ser enviadas pelo cliente para o servidor, ‘add’ para instalar sensores, ‘remove’ para remover sensores, ‘list’ para consultar equipamentos e ‘read’ para consultar variáveis de processo.

Implementação

A implementação foi realizada utilizando o software Visual Studio Code e testado utilizando o software WSL (Windows Subsystem for Linux), a programação usou de base a linguagem C com inclusão das bibliotecas específicas para lidar com sockets e strings.

Os códigos do sistema ficam subdivididos em quatro arquivos, sendo eles client.c, server.c, common.c e common.h. O cliente é responsável por enviar as mensagens e, com isso, servir de interface para o usuário operar o sistema, já o server é onde estão armazenados os dados dos sensores e onde a mensagem enviada será tratada para executar o comando solicitado, seja ele instalar, remover ou consultar.

Para controlar os sensores adicionados e removidos dos equipamentos é criada uma matriz 4x4 que é inicializada com 0 em todos os elementos, onde as linhas representam os equipamentos e as colunas representam os sensores. À medida que os sensores são adicionados, os elementos correspondentes são alterados para 1 representando que há um sensor com o id da coluna no sensor com id da linha.

Para tratar as mensagens recebidas no servidor foi utilizada a função strtok que divide a mensagem recebida em tokens com o delimitador definido, no caso desde programa o delimitador padrão é o espaço ‘ ‘. Antes da mensagem ser identificada para a função que deve exercer é identificado quantas palavras compõem a mensagem, assim é possível identificar onde está cada parâmetro passado.

Para a mensagem ‘add’ primeiro é identificado o tamanho da mensagem, e, a partir disso, temos informação de quantos sensores serão adicionados. Logo após salvamos os parâmetros dos ID’s dos sensores e do ID do equipamento em buffers e verificamos para cada um deles se o mesmo já existe no equipamento, a partir disso é enviada a mensagem correspondente a ação para o cliente.

Para a mensagem ‘remove’ o processo é similar ao descrito acima, porém a verificação é se o equipamento possui o sensor a ser removido. Já para a mensagem list e read é identificado quais sensores existem no equipamento solicitado e a partir daí é enviada a mensagem de retorno para o cliente com os ID’s ou o valor aleatório representando a leitura de um sensor.

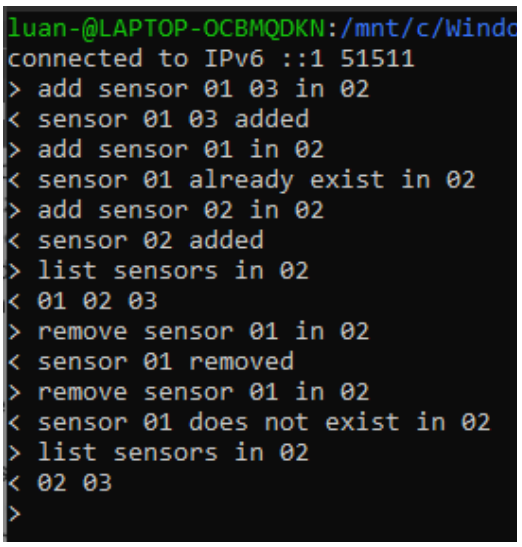

Imagem 1 - Exemplo 1

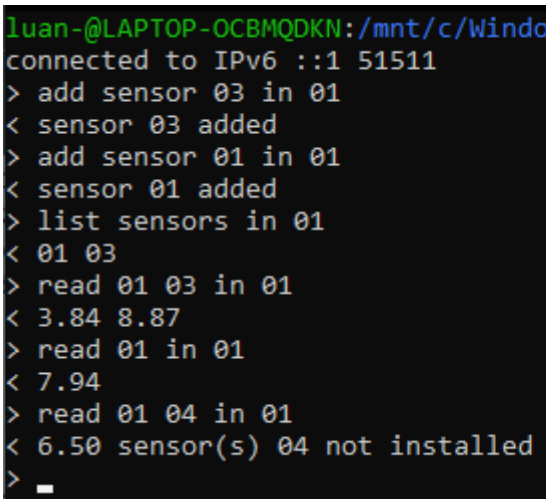

Imagem 2 - Exemplo 2

Acima estão apresentados os dois exemplos disponíveis nas especificações do trabalho prático, é importante salientar as duas diferenças presentes em relação às respostas do servidor. A primeira se trata da ordem em que os sensores são reportados no comando ‘list’, no código implementado eles são apresentados em ordem de ID, e não em ordem de inclusão como no exemplo, com isso a visualização do usuário fica mais intuitiva. A segunda é em relação a resposta do comando ‘read’ no exemplo 2, e a implementação do código segue a tratativa apresentada na seção de tratamento de erros da especificação do trabalho.

Instruções de compilação e execução

- Acesse o diretório: .\TP1_2019027008;
- Utilizando o terminal, execute o arquivo makefile utilizando o comando ‘make’ ;
- Com esse comando a compilação do programa deve ser executada gerando os arquivos a serem utilizados a seguir.
- Utilizando o terminal do servidor, execute o binário *server* utilizando o comando ‘./server **<v4|v6> <server port>**’ ; Obs.: A parte em negrito é correspondente tipo de endereço que será utilizado e o número de porta.
- Utilizando o terminal do cliente, execute o binário *client* utilizando o comando ‘./client **<server IP> <server port>**’ ; Obs.: A parte em negrito é correspondente ao endereço IP que será utilizado e o número de porta.

Desafios

Os principais desafios encontrados foram na aplicação do protocolo TCP para um problema real, e nesta parte as aulas disponíveis em vídeo e o livro de programação com sockets ofereceram suporte suficiente para implementação.

Após o processo descrito acima foi necessário modificar o programa para que a conexão mantivesse ativa e fosse possivel enviar mais de uma mensagem do cliente para o servidor, e, para isso, foram incluídas estruturas de repetição fazendo com que a troca de mensagens fique ativa até que o comando kill ou um comando inválido seja enviado.

Os processos já descritos foram os de mais complexidade, porém o que demandou mais tempo foi a parte de tratamento das strings e todas as possibilidades de mensagens. Visto que cada uma das variações teria de apresentar um resultado diferente isso resultou em inúmeras condicionais no código.

Conclusão

Neste trabalho prático foi realizada a solução do problema proposto utilizando os conhecimentos adquiridos nas aulas de Redes de Computadores, durante seu desenvolvimento foi possível notar vários pontos que não haviam ficado claros e que foi possível encarar na prática com um problema real, sendo assim muito proveitoso para mim. Com a solução adotada consegui entender melhor as aplicabilidades de um TCP, principalmente sendo utilizada para a comunicação ponto a ponto. Ao final do trabalho durante a produção do relatório também verifiquei alguns pontos de complexidades que poderiam ser desenvolvidos para o código ficar ainda mais dinâmico e algumas funções que poderiam ser otimizadas.