Trabalho Prático 1 - Redes de Computadores

Luiz Philippe Pereira Amaral

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)

1. Introdução

O trabalho desenvolvido em C representa o protocolo de comunicação via rede elaborado para gerenciar sensores em equipamentos via uma central de monitoramento que se comunica com uma estação remota. No ecosistema simulado existem 4 tipos de equipamentos e 4 tipos de sensores. Cada equipamento pode ter 0 ou 1 sensor de cada tipo. O protocolo consiste das seguintes mensagens:

1. Adicionar sensor: add sensor [sensor_id]+ in [equipement_id]
2. Remover sensor: remove sensor [sensor_id]+ in [equipement_id]
3. Listar sensores: list sensors in [equipement_id]
4. Lêr sensor: read [sensor_id]+ in [equipement_id]

O sistem tem ao todo capacidade para 15 sensores e as mensagens de sucesso e erro apropriadas são enviadas para cada situação.

2. Implementação

A implementação utiliza a interface padrão POSIX de sockets do C. Para compilar o programa, utilize o comando:

```
$ make
```

Algumas função utilitárias foram utilizadas para interpretar as entradas (tipo de endereço a ser utilizado e porta), para lêr do buffer de entrada e para formatar mensagens para serem enviadas via rede e para formatar mensagens recebidas da rede.

2.1. Servidor

O servidor é iniciado com a seguinte linha de comando:

```
$ ./server [v4|v6] [porta]
```

O servidor então espera até que uma conexão com o cliente seja estabelecida e a partir daí começa a responder as mensagens recebidas. Caso o cliente se disconecte, o servidor volta a aguardar uma conexão. Caso o cliente envie a mensagem ”kill”para o servidor, ele finaliza todas as sua operações.

O armazenamento do estado dos equipamentos e sensores é feito in-memory por meio de uma matriz, ou seja, não existe persistência de dados entre as seções.

A interpretação dos comandos é feita por meio de expressões regulares, cada mensagem tem a função que reconhece seu tipo e faz a conversão dos seus parâmetros de execução.

O maior desafio na implementação do servidor foi um comportamento não previsto do socket que, por padrão, permanece ativo por até 4 minutos após o encerramento da sua conexão com o programa para garantir que todos os dados foram transmitidos. Isso fazia com que iniciar o servidor várias vezes seguidas fazia com que a ciração do socket falhasse após a primeira vez. Para evitar esse compoertamento e liberar o socket imediatamente, é preciso configurá-lo com os parâmetros SO_REUSEADDR e SO_REUSEPORT.

2.2. Cliente

Para inicar o cliente, executamos o comando:

```
$ ./client [host] [porta]
```

O programa detecta se trata-se de uma conexão IPv4 ou IPv6 analisando o endereço host com uma expressão regular. Em seguida, tenta se conectar ao endereço e porta fornecidos, caso não conseiga, finaliza sua execução, caso consiga, ele lê as mensagens da entrada e envia para o servidor. Após receber a resposta do servidor, ele a imprime na tela e espera o novo comando. Esse loop é mantido até que o comando ”kill”seja enviado ou o buffer de entrada chegue em estado EOF (Ctrl+D para o terminal).