Trabalho Prático 1

Redes de Computadores

João Antonio Oliveira Pedrosa
Matrícula: 2019006752

[1] Universidade Federal de Minas Gerais
Belo Horizonte - MG - Brasil

1. Introdução

O trabalho propõe a implementação de um sistema modelo cliente servidor para simular a interação entre uma Estação Remota (o servidor) e uma Central de Monitoramento (o cliente). A Estação Remota deve ser capaz de atender às seguintes solicitações da Central de Monitoramento:

1. Instalar sensor: Adiciona um novo sensor à um equipamento na Estação Remota
2. Remover sensor: Remove um sensor existente de um equipamento na Estação Remota
3. Consultar equipamento: Informa os tipos de sensores instalados no equipamento
4. Consultar variável de processo: Solicita dados de um tipo de variável de processo de um equipamento (e.g. pressão, temperatura, velocidade, etc)

Para tal, a Estação Remota e a Central de Monitoramento trocam mensagens curtas de até 500 bytes utilizando o protocolo TCP.

2. Ferramentas e Linguagem de Programação

O trabalho foi desenvolvido utilizando a linguagem Python em sua versão 3.9.7 e as únicas bibliotecas utilizadas foram:

- random
- sockets
- sys

Como o trabalho foi desenvolvido em Python, não há um Makefile para compilação que gera os arquivos binários. Para executar os arquivos basta fazer, no terminal, a chamada:

python arquivo.py [arg1] [arg2]

3. Decisões e Modelagem Computacional

Como o retorno dos valores das variáveis de processo não precisa ser consistente com nenhum contexto, é retornado, quando solicitado pela Central de Monitoramento, um número aleatório de 0 a 10 com 2 casas de precisão. Sendo assim, não é necessário guardar valores para essas variáveis mas apenas uma indicação de quais sensores estão instalados em cada equipamento. Para tal, os sensores de cada equipamento são armazenados em um *Set*, estrutura padrão do Python capaz de remover inserções e deleções em tempo constante utilizando Hash Tables. Tais *Sets* são armazenados em um dicionário no

qual as chaves são o *id* do equipamento e os valores são os *Sets* que represeta os sensores já instalados no equipamento.

Tal solução se mostraria eficiente até para casos mais complexos do que o proposto no trabalho, pois seria capaz de permitir que um número maior de equipamentos ou sensores possam ser monitorados, sem comprometer o tempo de resposta do servidor.

Algumas decisões foram feitas como parte da interpretação do que é pedido no trabalho:

- No caso de uma mensagem não reconhecida ser enviada ao servidor, o servidor desconecta o cliente sem retornar mensagem alguma.
- Caso uma lista de sensores seja passada com o comando *add* e algum desses sensores já esteja instalado no equipamento, a mensagem retornada é a especificada na especificação, porém a estação remota instala todos os outros sensores requisitados que ainda não estavam instalados.
- Apesar de não ser especificado, a função *remove* é capaz de, assim como a função *add*, receber uma lista de sensores a ser removido. Tal suposição foi feita com base no fato de que essa funcionalidade não interefere em nenhum teste que seja feito c
- Na função *list*, como não foi especificado em qual ordem os sensores devem aparecer na mensagem, assumiu-se que qualquer ordem é válida.

4. Implementação

Segue uma breve explicação de todas as funções implementadas no servidor. No caso do cliente, a única função implementada é a **main**, que simplesmente realiza a solicitação de uma mensagem pela entrada padrão, envia para o servidor e imprime a mensagem que o servidor retorna.

4.1. `terminate`

Recebe como argumento um cliente e uma string. Converte a mensagem para bytes, envia ao cliente e finaliza a conexão.

4.2. `send`

Recebe como parâmetros um cliente e uma string. Converte a mensagem para bytes e envia ao cliente.

4.3. `receive`

Recebe como parâmetro um cliente. Faz uma requisição de mensagem para o cliente e retorna a mensagem convertida para uma string.

4.4. `parseCommand`

Recebe como parâmetro uma string. Baseado na suposição de que a string contém uma mensagem do cliente no formato especificado na documentação, faz o parsing da mensagem e retorna uma string contendo a ação que deve ser performada, uma lista de sensores e a id do equipamento.

4.5. `checkCommand`

Recebe como parâmetro uma lista de sensores e a id de um equipamento. Retorna um valor booleano indicando se os sensores e o equipamento passados estão dentre os monitorados pela Estação Remota.

4.6. `addSensors`

Recebe como parâmetro uma lista de sensores e a id de um equipamento. Checa se algum dos sensores da lista já está instalado e instala todos os que não foram instalados ainda. Retorna uma string representando a mensagem que deve ser enviada ao cliente como resultado da ação.

4.7. `readSensors`

Recebe como parâmetro uma lista de sensores e a id de um equipamento. Checa se todos os sensores da lista estão instalados no equipamento. Caso estejam, retorna uma mensagem contendo um número aleatório gerado para cada sensor. Caso algum dos sensores não esteja instalado, retorna uma mensagem contendo o primeiro sensor não instalado encontrado.

4.8. `removeSensors`

Recebe como parâmetro uma lista de sensores e a id de um equipamento. Checa se todos os sensores já estão instalados. Remove todos os sensores da lista que estejam instalados. Retorna uma string representando a mensagem que deve ser enviada ao cliente como resultado da ação.

4.9. `listSensors`

Recebe como parâmetro a id de um equipamento. Retorna uma string contendo a mensagem especificando todos os sensores instalados no equipamento.

5. Conclusão

Com a implementação dessae sistema pude adquirir bastante familiaridade com como um sistema que opera recebendo e enviando mensagens através de um protocolo opera, afinal, isso é algo que eu nunca tinha feito antes e outrora não sabia como funcionava. Não houve nenhuma grande dificuldade encontrada durante a implementação do trabalho, a biblioteca Sockets do Python realmente torna o trabalho de enviar e receber mensagens algo relativamente simples de ser implementado e a possibilidade de aceitar os dois tipos de endereço de IP também se torna relativamente simples, pelo mesmo motivo.