Documentação TP1 - Redes de Computadores

Daniel Alves Resende - 2020026834

1. Introdução

O problema proposto para o primeiro trabalho prático foi a implementação de um sistema de controle de operação de sensores de salas de aula. Esses sensores informam a temperatura, humidade e o estado de funcionamento de quatro ventiladores nas salas. O sistema deve ser capaz de cadastrar salas, inserir, atualizar e remover dados dos sensores e mostrar os dados atuais a qualquer momento.

Para a implementação do sistema, uma dupla cliente-servidor foi proposta e desenvolvida. O servidor é responsável pelo armazenamento dos dados do sistema e por qualquer alterações feitas aos mesmos, enquanto o cliente interfaceia com o servidor para envio de comandos. A implementação foi feita usando o protocolo TCP, sendo compatível com ambos IPv4 e IPv6.

2. Arquitetura e funcionamento

2.1 Servidor

Na implementação do servidor, os dados das salas foram armazenados em um array do struct *Room*, enquanto a mensagem recebida é armazenada num struct de tipo *Message*. Seus formatos são:

```c
typedef struct Room{
    int room_state;
    int temperature;
    int humidity;
    int fan_states[4];
}Room;
```

```c
#define MSGSZ 256

typedef struct Message{
    int type;
    char payloadstr[MSGSZ];
} Message;
```

O servidor fica aberto a apenas uma conexão por vez, vinda de um cliente. As mensagens recebidas e enviadas seguem o protocolo TCP. Quando o servidor recebe uma mensagem do cliente em forma de *string*, a mesma é interpretada e a mensagem é reconstruída no formato definido no *struct*. A partir desse ponto, diversas ações podem ser tomadas no código de acordo com o comando que foi enviado. A ação equivalente ao comando é efetuada sobre os dados e uma mensagem de confirmação ou erro é construída e enviada, sendo que nos comandos de *lookup* a mensagem conterá os dados requisitados em caso de sucesso.

Caso o cliente deixe de responder o servidor, o mesmo encerrará a conexão e ficará em *standby*, esperando outro cliente. O fim deliberado da conexão se dará

apenas se o servidor receber uma mensagem com o comando *"kill"*, o que também encerrará a execução do servidor.

2.2 Cliente

O cliente recebe, durante a execução, comandos textuais por meio da linha de comando, que são verificados utilizando expressões regulares, ou *regex*. Dependendo do comando, os argumentos inclusos também são verificados para a identificação da validez dos mesmos *client-side*, e múltiplos comandos podem ser usados em sucessão. Caso um comando não identificado seja usado, o cliente encerrará a execução. No caso de comandos e argumentos válidos, o cliente construirá uma mensagem do tipo *Message* para processamento interno. Antes do envio, esse objeto é usado para a construção da mensagem em formato de *string*, que é prontamente enviada.

A conexão cliente-servidor é mantida durante toda a execução do cliente. Após o envio das mensagens por parte do cliente, o mesmo esperará uma resposta por parte do servidor, que pode tomar os formatos de: código de confirmação; código de erro; dados requisitados. Caso recebida, a resposta será interpretada e a mensagem equivalente será mostrada no terminal. Em caso de *lookup*, essa mensagem inclui os dados requisitados. Caso o servidor deixe de responder após uma requisição enviada, o cliente encerrará sua execução. Da mesma forma, caso o comando *"kill"* seja usado, o cliente enviará uma mensagem equivalente ao servidor e encerrará sua execução.

3. Discussão

A tarefa proposta de implementar um sistema de controle de dados inicialmente parece simples, mas demonstra algumas das dificuldades encontradas no desenvolvimento de um par cliente-servidor. Notoriamente, a necessidade de garantir a chegada dos dados de forma completa implica na necessidade de tratamento de imprevistos por meio de desconexões, comandos inválidos, ou formatos inesperados de mensagens. Assim, o formato de comunicação entre os dois agentes precisa ser definida previamente de forma que acoberte todos os casos possíveis. É necessário então proteger ambos servidor e cliente de casos fora desse padrão, garantindo que o funcionamento siga corretamente.

4. Conclusão

A implementação do sistema, embora inicialmente simples, é um problema que se mostra gradualmente mais complexo. É necessário aplicar um conhecimento considerável de sistemas de rede, ter boas práticas de código e planejamento para evitar problemas de casos inesperados e ter ótimas noções de tratamento de erros. Essas exigências proporcionam uma experiência que traz aprendizados valiosos sobre programação e a área de comunicação em redes.