Trabalho Prático 2

Comunicação Cliente - Servidor TCP

Guilherme Mendes de Oliveira
Departamento de Ciência da Computação

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

guilhermemendes@ufmg.br

1. Introdução

Este trabalho tem como objetivo a implementação de um emulador de camada de enlace para uma rede fictícia, chamada DCCNET. Nela temos a configuração de uma comunicação Cliente Servidor com o protocolo TCP. O Cliente deve ser capaz de ler um arquivo de entrada com determinadas mensagens enquadrá-las "concatenando" as com um cabeçalho, em seguida codifica-las em Base16, isto é, os dados binários nos caracteres ASCII 0-9 e a-f são codificados, a cada quatro bits converteremos em 8 bits numa função *encode16*. Feito a conversão o cliente enviará a mensagem para o servidor que deverá decodificar a mensagem numa função *decode16* e desenquadra-la para que possa prosseguir com as verificações para detectar erros de transmissão e assim receber novos quadros.

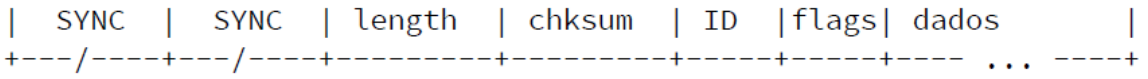

Figura 1 - Exemplo de formação de um quadro.

2. Implementação

2.1 Instruções de compilação e execução

O programa foi desenvolvido na linguagem Python versão 3.9 e executado no ambiente Linux distribuição Ubuntu versão 18.04.02, utilizando bibliotecas padrão da linguagem. Para executar temos os seguintes comandos:

Servidor: python3 main.py -s porta arquivoInput arquivoOutput

Cliente: python3 main.py -c 127.0.0.1porta arquivoInput arquivoOutput

2.2 Estrutura de Dados

Para a implementação não foi utilizado nenhum tipo abstrato de dados pelo motivo de trazer complexidade desnecessária para a implementação do código

2.3 Código

Para a implementação do Cliente e do Servidor foram utilizados como base uma implementação de comunicação padrão conforme documentação da biblioteca socket, utilizando suas função para criar os sockets de comunicação, para conexão após recebimento da porta além de envio e recebimento de mensagens.

2.3.1 Cliente

O Cliente é responsável por iniciar a comunicação, conectando na instância do servidor recebendo como parâmetro o endereço do host, no caso para a implementação de teste o *localhost* 127.0.0.1 e a porta de conexão definida no servidor, no caso para a implementação de teste a 50511. Após ser inicializado através da função open() recebe na entrada (arquivo input) a mensagem (linha do arquivo), após esse início de iteração que termina ao final do arquivo ele realiza todos os processos de enquadramento, codificação e transmissão da mensagem ao servidor, quando todas as linhas do arquivo são enviadas ele encerra a conexão.

2.3.2 Servidor

O Servidor é instanciado definindo a porta de conexão uma vez que o host para teste é o *localhost* após isto, recebe a mensagem do cliente decodifica e desenquadra, caso a mensagem esteja correta, isto é, o valor do *checksum* presente no cabeçalho do quadro esteja de acordo, ele recebe um novo quadro, caso contrário solicita o envio da mensagem novamente ao cliente. Após o término da comunicação ele fecha o *socket* com o cliente e aguarda nova conexão, para que o servidor seja encerrado é necessário utilizar o *"ctrl+c"* no terminal.

3. Desafios, dificuldades e imprevistos

Por se tratar de uma linguagem de mais alto nível a dificuldade em implementar a comunicação básica em Python foi relativamente fácil ao ser comparada com a implementação inicial testada em C/C++. Ao realizar o tratamento dos dados e enquadrá-las tive certa dificuldade em entender a estrutura struct utilizada para criar o quadro e após isso converter a *string* gerada para a Base16 o que levou muito tempo até que a implementação ficasse correta além disso essa *string* gerada para cálculo do checksum também levou um bom tempo para entendimento e implementação, infelizmente o grau de corretude para tal elemento não está completo.

4.Conclusão

Embora o grau de corretude esteja baixo, de modo geral o entendimento e desenvolvimento de todos os princípios básicos da comunicação Cliente-Servidor seguindo o protocolo TCP juntamente com os conceitos de enquadramento e tratamento de erros foi satisfatório. Todos os problemas que existem na execução monitorada pelos testes apontam que os problemas estão na manipulação e transformação dos dados para os diferentes formatos de *bytes* necessários para os cálculos e formação dos quadros.

5.Referências

<http://linguagemc.com.br/a-biblioteca-string-h> .Acesso em: 22 de Junho de 2021

Banahan, M;Brady,D;Doran,M. The C Book. Capítulo 5: Arrays and Pointer.

Banahan, M;Brady,D;Doran,M. The C Book. Capítulo 9:Libraries.