Universidade Federal de Minas Gerais
Redes de Computadores - 2019/01

Trabalho Prático 1
DCCNET: Camada de Enlace

Deiziane Natani da Silva - 2015121980

1. Introdução

O presente trabalho consiste na implementação de um emulador para a camada de enlace para uma rede fictícia denominada DCCNET. Ele aplica os conceitos de codificação, enquadramento, detecção de erro, sequenciamento e retransmissão de dados. A partir de uma entrada o programa deve ser capaz de identificar o padrão referência de sincronização para realizar envio e recebimento de informações entre dois componentes conectados entre si.

A técnica utilizada é o enquadramento por sentinelas para que o dado seja lido de forma correta. Pelo mesmo código é possível rodar uma instância de conexão passiva (servidor) e uma de conexão ativa (cliente) onde ambos enviam os dados lidos a partir de um arquivo e escrevem os dados recebidos em outro arquivo.

2. Desenvolvimento

O trabalho foi implementado utilizando a linguagem de programação Python, construído como uma conexão TCP. Como dito anteriormente, o emulador desenvolvido funciona como transmissor e receptor simultaneamente. Para a execução do programa, são necessários alguns parâmetros com informações básicas para sua execução. A partir do segundo parâmetro, é possível identificar se o emulador deverá se comportar na ponta de forma passiva - como o servidor, ou de forma ativa - como cliente. Esta distinção é feita objetivando identificar quais serão os próximos parâmetros a serem lidos para cada um dos casos, bem como a criação de suas conexões. Abaixo estão os modelos de inicialização para cada um dos casos, servidor e cliente, respectivamente:

```
./dccnet -s <PORT> <INPUT> <OUTPUT>
./dccnet -c  <IP>:<PORT> <INPUT> <OUTPUT>
```

Para abordar o enlace *full-duplex* tanto no cliente quanto no servidor foram criadas duas threads para trabalharem simultaneamente, uma para enviar e outra para receber as informações. As threads de recebimento dos dados recebem tanto dados do arquivo quanto confirmações de envio (ACK).

No envio dos dados, ocorre a transmissão de SOF (*Start-Of-Frame*) necessário para identificar o início do quadro. Também, o dado é coletado do arquivo, os outros campos do enquadramento são construídos, e tudo é enviado. O campo EOF (*End-of-Frame*) indica o fim do recebimento de dados. Como o sinal EOF pode estar contido no meio dos dados, ele é escapado através do *byte stuffing*, sendo colocado um sinal DLE (*Data-Link-Escape*) antes do EOF. No recebimento dos dados é realizada a sincronização pelo SOF e as outras informações são recebidas uma a uma para facilitar as verificações dos parâmetros *checksum, id, flag e data*, porque eles são utilizados para validação das informações enviadas.

Para a transmissão de dados são necessários alguns passos:

1. Ler os dados a serem enviados, através do arquivo passado como parâmetro, e, a partir dele, montar o quadro a ser transmitido.

2. São criados os campos de sincronização **SOF, ID, flags, checksum, dados** e o caractere **EOF**. Por fim, é feita a concatenação de cada uma das variáveis citadas acima gerando, assim, o enquadramento de envio. Entretanto, antes de transmitir o quadro, ainda são necessários mais alguns ajustes de modo a garantir que o receptor possa identificar uma possível ocorrência de erros durante o envio dos dados. O checksum é um artifício inserido no quadro de envio com o intuito de verificar a integridade dos dados recebidos. Antes do envio de um pacote, é calculada a soma de verificação dos dados contidos no quadro e este valor é armazenado no campo checksum do mesmo.

3. Quando o quadro chega ao receptor, a soma é realizada novamente para verificar a consistência dos dados recebidos. Após o cálculo do checksum e a substituição deste valor no quadro gerado anteriormente, tem-se os dados enquadrados e prontos para serem transmitidos.

A recepção foi implementada da seguinte forma:

1. Os dados são recebidos por meio da função ***recv***

2. É calculado o ***checksum*** do quadro recebido para saber se o mesmo foi corrompido durante a transmissão

3. Se dados foram entregues com sucesso e não houve corrompimento durante a transmissão. Neste caso, um **ACK** é enviado para o remetente da mensagem confirmando o êxito da conexão (o ACK é um quadro sem dados).

4. Após o envio do ACK, o quadro recebido anteriormente é escrito no arquivo de saída

5. Caso o quadro recebido esteja corrompido, um erro ocorreu durante o envio dos dados. Nenhuma confirmação é recebida e o receptor fica aguardando pela retransmissão do quadro.

3. Conclusão

O trabalho foi importante para complementar o conteúdo aprendido em sala de aula. Ao conhecer a teoria, já é possível ter uma boa noção do funcionamento das camadas de redes, entretanto, o trabalho prático é, sem dúvida, um mecanismo excelente para visualizar, de forma concreta, a maneira com que os processos acontecem, mais especificamente, na camada de enlace. Apesar dos desafios, conclui-se que o objetivo final foi atingido