REDES DE COMPUTADORES
TRABALHO PRÁTICO 1A

Rafael Rubbioli : 2014124838 Fernanda Ramalho : 2014106368
Departamento de Ciência da computação, UFMG

2 de setembro 2017

1 Introdução

Este trabalho tem como objetivo a criação de um emulador de enquadramento de dados da camada de enlace. Ele será feito na linguagem C e usará sockets. Contará com sequenciamento e detecção de erros e será full duplex.

2 Desenvolvimento

2.1 Enquadramento

Enquadramento de dados é uma técnica muito usada em redes, na qual se coloca informações adicionais nos pacotes enviados. Essas informações podem ser sobre destinatários, remetentes, tamanho do pacote ou mecanismos de verificação / correção de erro.
Neste trabalho iremos fazer enquadramento por contagem, ou seja, colocaremos no quadro o tamanho do payload. Além disso, usaremos um mecanismo de sentinelas para "avisar" o receptor de que está chegando um quadro. Colocaremos também um campo "reserved" no quadro, mas não o usaremos.
O método de detecção de erros usado será o Checksum. Neste método usa-se um algorítmo que calcula a soma dos dados do quadro antes de enviálos e logo após de recebe-los. Caso a soma seja igual, não houve erros e tudo pode continuar, mas caso ocorra erros, descartaremos todo o pacote até encontrar novos bytes de sincronização. Esse método não é o mais eficiente de detecção de erros que existe, visto que se houvesse um erro que mantivesse a soma igual, o pacote errado seria considerado correto, mas é bastante simples e este não é o objetivo principal deste trabalho.

2.1.1 Formato

O quadro será do seguinte formato:
8 bytes de **synch** (2x DCC023C2)
2 bytes de **checksum** (calculado pelo algorítmo)
2 bytes de **length** (tamanho do payload)
2 bytes de **reserved**
length/8 bytes de **payload** (dados realmente usados)

2.2 Entrada e saída

O programa deverá receber um arquivo de entrada e saída. O arquivo de entrada será lido (em binário), enquadrado e passado pela camada de enlace para que, do outro lado seja verificado e, caso esteja correto escrito no arquivo de saída (em binário). Além disso, o programa deverá receber o IP da rede, a porta (ou porto) e modo que a conexão deverá ser feita (ativa ou passiva). Exatamente nesta ordem :

./frame INPUT OUTPUT IP PORT MODE

2.3 Transmissão

A transmissão dos dados é feita como já foi descrito anteriormente. Primeiro o arquivo é lido de um em um byte, até que o tamanho máximo permitido no payload seja obtido, e enquadrado. O tamanho dos dados lidos é calculado enquanto os bytes são lidos.
Depois disso, é feito o checksum e o quadro inteiro é transmitido. A maneira como a transmissão é feita não importa, desde que todos os dados sejam enviados corretamente.
A função que implementa a transmissão termina quando o arquivo inteiro é lido. A conexão é fechada. Assim, para continuar a mandar os dados, é preciso fazer outra conexão.

2.4 Recepção

A recepção dos dados é feita de maneira que nada é lido até chegarem bytes de sincronização, depois é lido o restante do quadro e feito o checksum. Caso o checksum seja igual ao recebido no quadro, os dados são escritos no arquivo de saída.
O arquivo de saída é aberto em modo "ab", ou seja, permite que escreva no final do arquivo, sem modificar o que já está lá. Em caso de vários quadros, os payloads recebidos serão colocados no final.
A função que implementa o receptor só para de esperar por quadros quando o cliente fecha a conexão. Além disso, ela lê os bytes de sincronização um a um para, caso chegue bytes não esperados, seja mais fácil de enquadrar o quadro de novo. O arquivo de saída é fechado dentro dessa função assim que a conexão com o cliente acaba.

2.5 Full Duplex

O programa rodará da maneira full duplex, ou seja, fará o envio e o recebimento dos dados ao mesmo tempo, independentemente do modo em que esteja rodando (ativo ou passivo). Para isso, usamos threads.
A thread é um "desvio" controlado no fluxo de execução onde criamos mais de um fluxo no programa. Neste caso, precisamos criar apenas um fluxo adicional para fazer o recebimento de dados, já que o fluxo principal estava fazendo a transmissão dos dados.

3 Terminação

O programa é terminado quando o nó remoto fecha a conexão ou quando recebe um sinal de interrupção do teclado (ctrl + c).

4 Dificuldades

As maiores dificuldades do trabalho foram em trabalhar com os tipos da linguagem C. O problema se encontra nos *casts* e nas converções de tipos.
Outra dificuldade foi trabalhar com as threads para poder controlar a transmissão e recepção dos quadros. Como as threads têm comportamentos imprevisíveis (algumas podem acabar antes das outras, impedindo que algumas tarefas sejam executadas) foi preciso usar a função *pthreadjoin*, que suspende a execução de uma thread até que a outra termine, mas encontramos dificuldade em fazer o join funcionar para ambos os transmissor e receptor.
Por fim, houve a dificuldade de lidar com o fechamento do arquivo. Como o código deveria rodar de maneira indeterminada tivemos que escolher o local adequado para a função fclose().

5 Testes

Os testes foram feitos rodando o programa em dois terminais diferentes e com um testbench que envia quadros errados. Além disso, foram usados vários tipos de arquivos como .txt, .pdf, .jpg etc A corretude do programa foi verificada apenas em alguns casos, mas não foram encontrados os erros. Suspeitamos que seja devido erro no uso das threads.

6 Conclusão

Houveram algumas dificuldades, talvez no uso das threads que não havia muita prática, ou talvez na dificuldade de tratamento dos dados na linguagem C. Porém os conceitos foram bem trabalhados e não prejudicou o aprendizado principal do trabalho que era sobre as redes de computadores e o uso de quadros.