DCCNET: Camada de Enlace

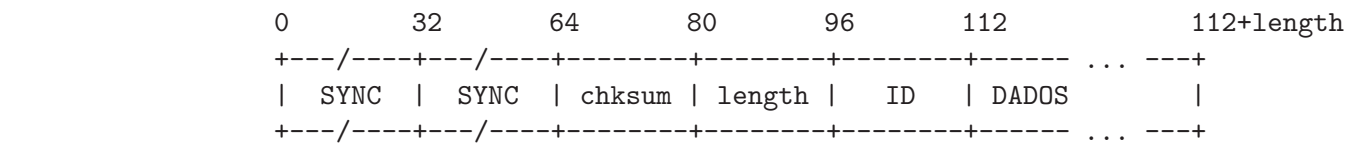
Introdução

Neste trabalho iremos desenvolver um emulador de camada de enlace para uma rede fictícia chamada DCCNET. O emulador tratará do sequenciamento, enquadramento, detecção de erro e retransmissão dos dados.

Enquadramento

O formato de um quadro DCCNET é mostrado no diagrama 1. A sequência utilizada para sincronização entre os pontos finais (`SYNC`, 32 bits) é sempre `0xDCC023C2`. O campo de detecção de erro (`chksum`, 16 bits) utiliza o algoritmo de *checksum* da Internet para detecção de erros. O campo de tamanho (`length`, 16 bits) conta a quantidade de *dados* transmitidos no quadro; o campo de tamanho *não* conta bytes do cabeçalho do quadro. O campo `ID` (16 bits) identifica o quadro transmitido ou confirmado. Os campos `length` e `ID` devem ser transmitidos usando *network byte order* (*big-endian*).

DIAGRAMA 1: Formato de um quadro DCCNET



Sequenciamento

Nesta versão do trabalho iremos emular a camada de enlace DCCNET usando o algoritmo *stop-and-wait* de controle de fluxo. As janelas de transmissão e recepção terão tamanho um. Consequentemente, os valores do campo `ID` se restringem a zero ou um. Abaixo detalhamos o funcionamento de um nó para transmitir e receber dados. Note que, como os enlaces DCCNET são *full-duplex* (dados podem ser transmitidos nas duas direções), cada nó funciona como transmissor e receptor simultaneamente.

Transmissor

Para transmitir dados, o nó transmissor cria um quadro como especificado no diagrama 1. Quadros de transmissão de dados possuem valor maior do

que zero no campo `length`. Quando `length > 0`, o campo `ID` contém o identificador do quadro. Enquanto o quadro transmitido não for confirmado, ele deverá ser retransmitido (periodicamente) a cada segundo. Após recebimento da confirmação, o transmissor deve trocar o valor do campo `ID` (de zero-para-um ou de um-para-zero) e transmitir o próximo quadro.

Receptor

Um quadro de dados só pode ser aceito se ele tiver identificador (`ID`) diferente do identificador do último quadro recebido[1] e se nenhum erro for detectado (ver abaixo). Ao aceitar um quadro de dados, o receptor deve criar um quadro de confirmação. Quadros de confirmação não carregam dados e têm `length = 0`. O campo `ID` do quadro de confirmação deve ser idêntico ao do quadro confirmado.

O receptor deve manter em memória o identificador (`ID`) e o *checksum* (`chksum`) do último quadro recebido. Se um quadro recebido for uma retransmissão do último quadro recebido, o receptor deve reenviar o quadro de confirmação. Um quadro é considerado uma retransmissão do último quadro recebido se tiver o mesmo identificador (`ID`) e o mesmo *checksum* (`chksum`).

Deteção de Erros

Erros de transmissão são detectados usando o *checksum* presente no cabeçalho do pacote. O *checksum* é o mesmo utilizado na Internet e é calculado sobre todo o quadro, incluindo cabeçalho e dados. Durante o cálculo do *checksum*, os bits do cabeçalho reservados ao *checksum* devem ser considerados com o valor zero. Pacotes com erro não devem ser aceitos pelo destino nem confirmados.

Para recuperar o enquadramento após a detecção de um erro, seu emulador deve esperar por duas ocorrências do padrão de sincronização (`SYNC`) que marcam o início de um quadro. O transmissor deve manter qual o identificador (`ID`) do próximo quadro a ser enviado (inverso ao identificador do último quadro confirmado), para retransmitir em caso de erro.

Implementação

Você deverá implementar o DCCNET sobre uma conexão TCP.[2] Seu emulador do DCCNET deve ser capaz de funcionar como transmissor e receptor simultaneamente. Seu emulador deve imprimir na tela, em hexadecimal, todos

[1] Inicialize o identificador do último quadro com o valor um, de forma que o primeiro quadro transmitido tenha o identificador zero.

os dados recebidos. Seu emulador deve interoperar com outros emuladores (teste com emuladores dos colegas), inclusive com o emulador de referência implementado pelo professor.

Avaliação

Este trabalho deve ser realizado em grupo de até quatro alunos. O trabalho pode ser implementado em Python, C, C++ ou Java, mas deve interoperar com emuladores escritos em outras linguagens. O mesmo grupo (ou subconjuntos disjuntos do) utilizado para realizar este trabalho deverá ser utilizado para realização da segunda parte (implementação de janela deslizante). Qualquer incoerência ou ambiguidade na especificação deve ser apontada para o professor; se confirmada a incoerência ou ambiguidade, o aluno que a apontou receberá dois pontos extras. O aluno deverá entregar documentação em PDF de *até* 4 páginas (duas folhas), sem capa, utilizando fonte tamanho 10, e figuras de tamanho adequado ao tamanho da fonte. A documentação deve discutir desafios, dificuldades e imprevistos de projeto, bem como as soluções adotadas para os problemas.

Testes

Pelo menos os testes abaixo *serão* realizados durante a avaliação do seu emulador:

- Transmissão e recepção de dados em paralelo.
- Recuperação do enquadramento após erros de transmissão.

Exemplo

Para transmitir quatro bytes com valores 1, 2, 3 e 4 (nesta ordem), os seguintes bytes terão de ser transmitidos na camada de enlace (assumindo que o identificador (`ID`) do quadro é zero):

```
dcc023c2dcc023c2faef0004000001020304
```

[2]Utilize `socket(AF_INET, SOCK_STREAM, 0)` para criar o *socket*. Isto simplifica o desenvolvimento do trabalho.