Instituto de Informática - UFRGS

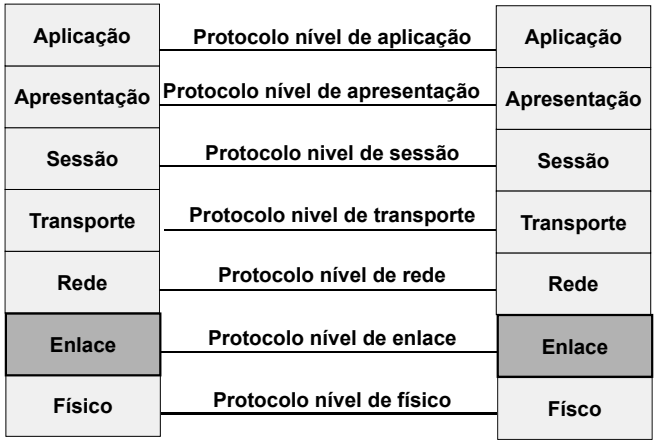
Redes de Computadores

Camada de enlace de dados

Introdução

Aula 06

Nível de Enlace



Instituto de Informática - UFRGS
A. Carissimi -19-mars-18

Introdução

- O que é um enlace?
  - Meio físico que compõe uma rede local (*Local Area Network* – LAN)
    - Pode ser com fio (coaxial, par trançado, fibra óptica) ou sem fio (ar)
  - Interliga nós (computadores, roteadores, *switches, bases wireless,* etc)
- Um enlace pode ser
  - Difusão (*broadcast*): composto por múltiplos nós
  - Ponto a ponto: composto por dois nós

Instituto de Informática - UFRGS
A. Carissimi -19-mars-18

Característica do enlace

- Enlace *half-duplex*
  - Os dispositivos podem transmitir e receber dados
  - A comunicação é bidirecional, porém NÃO PODE haver transmissões simultâneas.
  - Compartilhado e o sinal é difundido em todo o meio
- Enlace *full-duplex*
  - Os dispositivos podem transmitir e receber dados
  - A comunicação é bidirecional e PODE haver transmissões simultâneas
  - Meio dedicado entre um dispositivo e outro

Instituto de Informática - UFRGS
A. Carissimi -19-mars-18

Funções da camada de enlace (MR-OSI)

- Enquadramento de dados
- Endereçamento
- Detecção de erros ou detecção e correção de erros

Aula de hoje

- Entrega de quadros
  - Confiável: orientado a conexão (controle de fluxo e de erros)
  - Não confiável: não orientado a conexão (com ou sem confirmação)
- Controle de acesso ao meio
  - Não definido no modelo de referência OSI

Procedimentos implementados por protocolos da camada de enlace

Enquadramento

- Definição de uma unidade de transmissão: quadro de dados
  - Basicamente, cabeçalho + área de dados + rabeira (opcional)
    - Envolve a definição de campos (sintaxe e semântica)
  - Compõe a 2-PDU
- Como delimitar quadros?
  - Tamanho fixo
  - Tamanho variável: marcas de inicio e de fim
- Marcas de inicio e fim
  - Um caractere específico (*flag*) como 0111 1110 (caractere *til*)
  - Um conjunto de caracteres (DLE+STX e DLE+ETX

O que acontece se a marca aparece na área de dados?

Bit stuffing* ou *Byte stuffing

Bit stuffing e *Byte stuffing*

- Princípio básico:
  - O transmissor sabe o que vai transmitir na sua área de dados e evita o envio de dados que coincidam com a marca (substituição de símbolos)
  - Padrão de substituição é combinado com o receptor (protocolo)
- *Bit stuffing*
  - Transmissão serial: evita seis bits consecutivos em 1 na área de dados
    - Transmissor Após 5 bits consecutivos em 1, insere1 bit em zero
    - Receptor: após receber 5 bits em 1, descarta, se zero, o 6º bit; se recebe o 6º bit em 1, espera o próximo como zero (marca de fim 1111110)
- *Byte stuffing*
  - Transmissão paralela: evita a ocorrência única de DLE na área de dados
    - Transmissor: insere 1 DLE após a ocorrência de DLE na área de dados
    - Receptor: ao receber dois DLEs, descarta um deles

Endereçamento

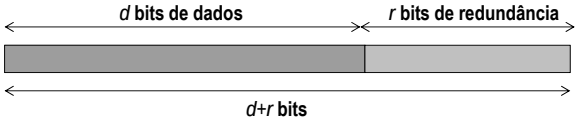
- Problema em enlaces compartilhados (*broadcast*)
  - Como identificar o destino ?
  - Como saber que uma transmissão é para mim ?
- Solução:
  - Definir endereços físicos no enlace (endereço MAC)
    - Ex.: em redes IEEE802.3 tem-se 08:00:46:EC:69:52 (end. *unicast*)
- Particularidades:
  - Endereços no enlace devem ser únicos
    - Unicidade de endereços (contexto global/local)
  - Possibilidade de definir endereços de grupo (*multicast*)
    - Caso especial: endereço de *broadcast* (ex.: FF:FF:FF:FF:FF:FF)

Detecção e correção de erros

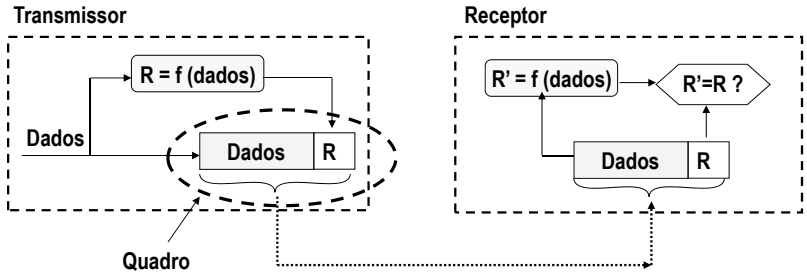
- Erros acontecem!!
  - Necessário tratar essa situação para fornecer um canal lógico livre de erros → controle de erros
- Controle de erros é baseado na capacidade de;
  - Detecção e correção
  - Detecção, descarte e retransmissão

Palavra-chave é DETECÇÃO !

- Metodologia básica:
  - Incluir informação (redundância) junto a cada bloco de dados para possibilitar a detecção de um erro ou sua detecção e correção



Princípio básico para detecção de erros



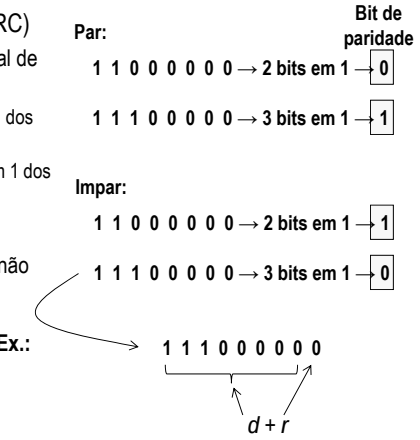
R = redundância de informação para detecção de erros

Métodos para correção e detecção de erros

- Detecção de erros:
  - Redundância de caracter (paridade VRC)
  - *Checksum*
  - Códigos polinomiais ou códigos cíclicos (CRC)
- Detecção e correção de erros
  - Paridade bidimensional
  - Códigos de Hamming, Reed-Solomon, LDPC (*Low-density parity-check*), etc
- Características básicas desses métodos
  - Quantidade de redundância inserida
  - Método utilizado para calcular a redundância
  - Cobertura (bits com erros podem passar despercebidos?)
    - Depende dos fatores anteriores

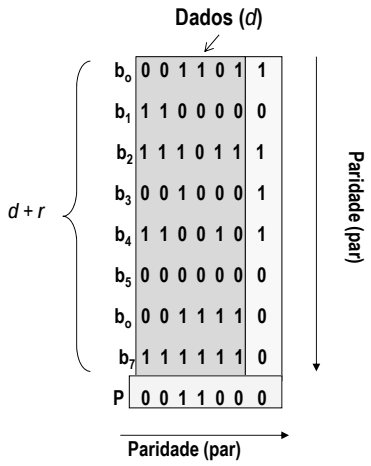
Cálculo de paridade

- Redundância de caracter (paridade VRC)
  - Inserção de um bit de paridade ao final de cada caracter
    - Paridade par: quantidade de bits em 1 dos dados e da paridade deve ser par
    - Paridade impar: quantidade de bits em 1 dos dados e da paridade deve ser ímpar
- Problema:
  - inversão de um número par de bits é não detectada

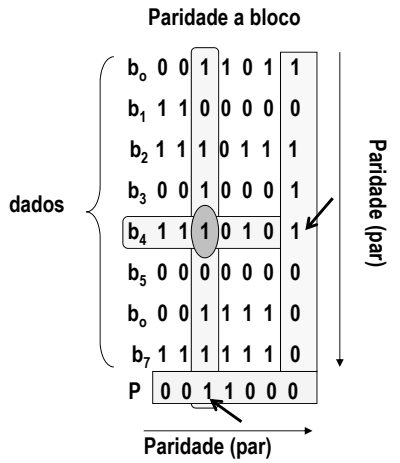


Cálculo de paridade bidimensional

- Redundância de bloco (paridade LRC)
  - Divide os bits de dados a serem enviados em um bloco com *i* linhas e *j* colunas e calcula a paridade (par ou ímpar) de cada linha e coluna
  - Permite detectar erro em qualquer combinação de dois bits
  - Permite detectar e corrigir erros em único bit (apenas!!!)
- Problema:
  - pode ocorrer inversões de bits que mascaram o erro



Exemplo capacidade de detecção e correção de erros



Detecção de erro: linha b4; coluna 3

Soma de verificação (*checksum*)

- Redundância é a soma dos dados em aritmética binária
- Transmissor:
  - Divide os dados em *k* blocos de *n* bits,cada um
  - Efetua a soma os blocos e complementa o resultado (*checksum*)
  - Envia *checksum* junto com os dados
- Receptor:
  - Recebe os dados e divide em k blocos de *n* bits
  - Soma os blocos e complementa o resultado (*checksum*)
    - Se *checksum* for zero, os dados são aceitos → não houve erro!!

Exemplo de *checksum*

Dados a serem transmitidos: 1010100100111001

Soma 10101001 + 00111001 = 11100010
Checksum 00011101

Feito pelo transmissor

Transmissão: 1010100100111001 00011101

10101001
00111001
00011101
Soma 11111111
Checksum 00000000

Feito pelo receptor

Se ZERO, OK!

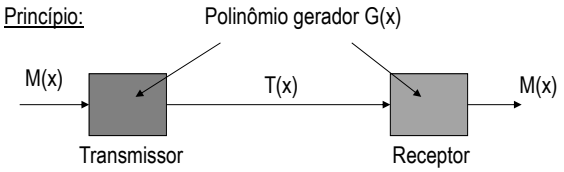
Empregado pelo IP (cabeçalho), UDP e TCP (cabeçalho e área de dados)

Verificação de redundância cíclica (CRC)

- *Cyclic Redundancy Check* (CRC)
- Código polinomial pois considera a cadeia de bits a ser transmitida como um polinômio cujos coeficientes são 1 e 0

$$110101 \equiv x^5 + x^4 + x^2 + 1$$

- Dados (quadro) a serem transmitidos são n bits
  - polinômio de **n** termos
  - x $^{n-1}$ até x^0



CRC: Algoritmo de Cálculo

- Multiplicar M(x) por x^r
- Dividir x^{r}M(x) por G(x)
- Subtrair o resto da divisão de x^{r}M(x)

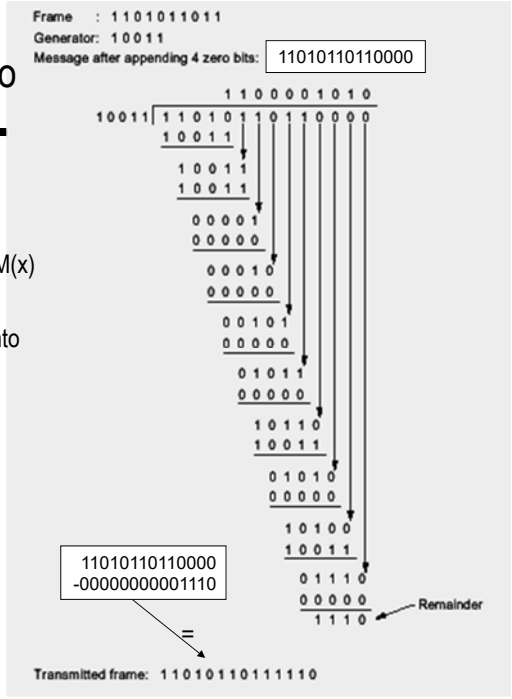
- Princípio básico de funcionamento (decimal)

210278 | 10941

2399 →resto

(210278-2399) | 10941

zero →resto



CRC: erros detectados (alguns números...)

- Todos erros de um bit
- Todos erros duplos, se o polinômio possuir pelo menos 3 termos em 1
- Qualquer número ímpar de erros se polinômio for fatorável por x+1
- Qualquer erro em seqüências de *n* bits ou menos (*n*=grau do polinômio)
- CRC-16 e CRC-CITT
  - 100% das falhas em sequências de 16 ou menos bits, 99.997% das falhas em sequências de 17 bits, 99.998% em sequências de 18 bits ou mais
- CRC-32
  - Chance de receber dados ruins é de 1 em 4.3 bilhões

Se chegou sem erro de CRC, talvez esteja correto!!

CRC: Polinômios geradores

- Polinômios geradores:

$$CRC-12 = X^{12} + X^{11} + X^3 + X^2 + X + 1$$
$$CRC-16 = X^{16} + X^{15} + X^2 + 1$$
$$CRC-CCITT = X^{16} + X^{12} + X^5 + 1$$
$$CRC-32 = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

- Implementação via hardware
  - Combinação de portas lógicas ou-exclusivo com *shift registers*
  - Empregado na Ethernet (IEEE 802.3): CRC-32

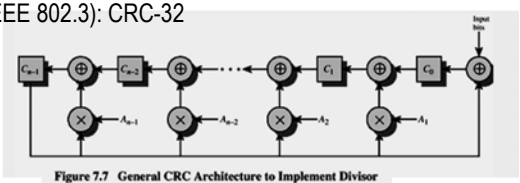


Figure 7.7 General CRC Architecture to Implement Divisor $1 + A_1X + A_2X^2 + \ldots + A_{n-1}X^{n-1} + X^n$

Peterson & Browson (1961)

Leituras complementares

- Stallings, W. *Data and Computer Communications* (6th edition), Prentice Hall 1999.
  - Capítulo 7 seção 7.2
- Tanenbaum, A. *Redes de Computadores* (4ª edição), Campus, 2003.
  - Capítulo 3, seções 3.1 e 3.2
- Kurose, J.; Ross, K.; *Redes de Computadores e a Internet* (6ª edição), Pearson Education do Brasil, São Paulo, 2013.
  - Capítulo 5, seções 5.1 e 5.2