Instituto de Informática - UFRGS

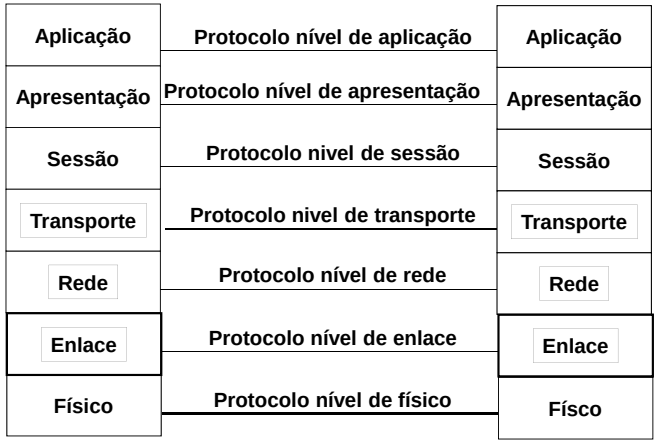
Redes de Computadores

Camada de enlace de dados

Introdução

Aula 06

Nível de Enlace



Instituto de Informática - UFRGS
A. Carissimi -5-avr.-11

Introdução

- Comunicação entre dois dispositivos envolve uma infra-estrutura física composta pela interligação de vários dispositivos entre si
  - Cada interligação é o que se denomina de **enlace**
- Enlace é a interligação entre dois pontos
  - Dedicado (ponto-a-ponto)
  - Compartilhado (*broadcast*)
- Possui um protocolo
  - Normalmente implementado em um adaptador de rede (placa de rede)

Instituto de Informática - UFRGS
A. Carissimi -5-avr.-11

Funções da camada de enlace

- Enquadramento de dados
- Endereçamento
- Detecção de erros ou detecção e correção de erros
- Controle de fluxo
- Controle de erros
- Prover serviços para a camada de rede
  - Orientado a conexão
  - Não orientado a conexão, sem confirmação
  - Não orientado a conexão, com confirmação
- Controle de acesso ao meio
  - Não definido no modelo de referência OSI

Instituto de Informática - UFRGS
A. Carissimi -5-avr.-11

Enquadramento

- Problema: como identificar início (ou final) de um quadro se tudo é sinal eletromagnético (ou óptico) no nível físico?
- Métodos básicos:
  - Marca de início e fim
  - Marca de início e informação do tamanho do quadro
  - Marca de início e um período de silêncio após o quadro
- A marca pode ser, por exemplo:
  - Um caractere específico (flag) como 0111 1110
  - Um conjunto de caracteres (DLE+STX e DLE+ETX)
  - Violação na codificação empregada no nível físico

Bit stuffing* ou *Byte stuffing

- Dependente da tecnologia usada pelo enlace

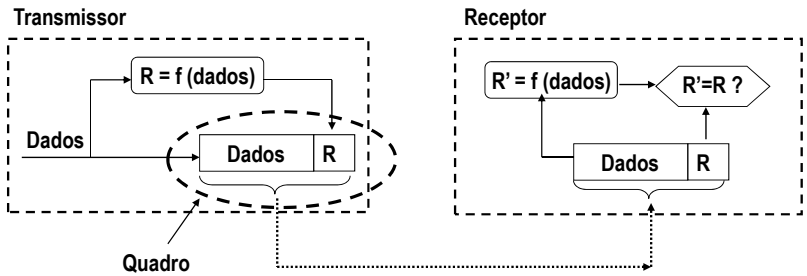
Endereçamento

- Problema em enlaces compartilhados (*broadcast*)
  - Como identificar o destino ?
  - Como saber que uma transmissão é para mim ?
- Solução: definir endereços físicos (endereço MAC)
  - Exemplo: em redes IEEE802.3 tem-se 08:00:46:EC:69:52 (end. *unicast*)
- Questões:
  - Unicidade de endereços (global/local)
  - Endereços de grupo (*multicast,* caso especial, *broadcast*)

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 dos mesmos
- Metodologia básica:
  - Incluir informação junto a cada bloco de dados para possibilitar a detecção de um erro e eventualmente sua correção

Princípio básico para detecção de erros



R = redundância de informação para detecção de erros

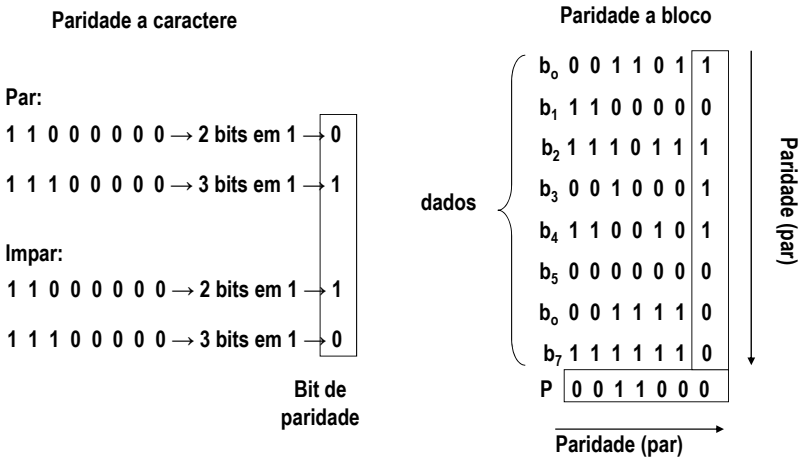
Detecção de erros

- Métodos comuns para detecção de erros:
  - Redundância de caracter (paridade VRC)
  - Redundância de bloco (paridade LRC)
  - *Checksum*
  - Códigos polinomiais ou códigos cíclicos (CRC)
- 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: número de bits caracter + paridade ⇒par
    - Paridade impar: número de bits caracter + paridade ⇒impar
  - **Problema**: inversão de um número par de bits é não detectada
- 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 e corrigir erros em único bit (apenas!!!)
  - **Problema**: pode ocorrer inversões de bits que mascaram o erro

Redundância de caracter (VRC) e de bloco (LRC)



Soma de verificação (*checksum*)

- A informação de redundância é a soma dos dados em aritmética binária
- Transmissor:
  - Os bits de dados são divididos em *k* blocos de *n* bits, cada um
  - Soma os blocos e complementa o resultado (*checksum*)
  - O *checksum* é enviado junto com os dados
- Receptor:
  - Os bits recebidos são divididos 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

	10101001
	00111001
Soma	11100010
Checksum	00011101

Feito pelo transmissor

Transmissão: 1010100100111001 00011101

	10101001
	00111001
	00011101
Soma	11111111
Checksum	00000000

Feito pelo receptor

Se ZERO, OK!

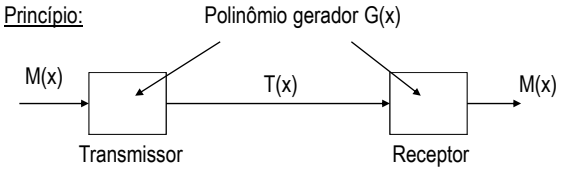
Empregado pelo IP, UDP e TCP

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



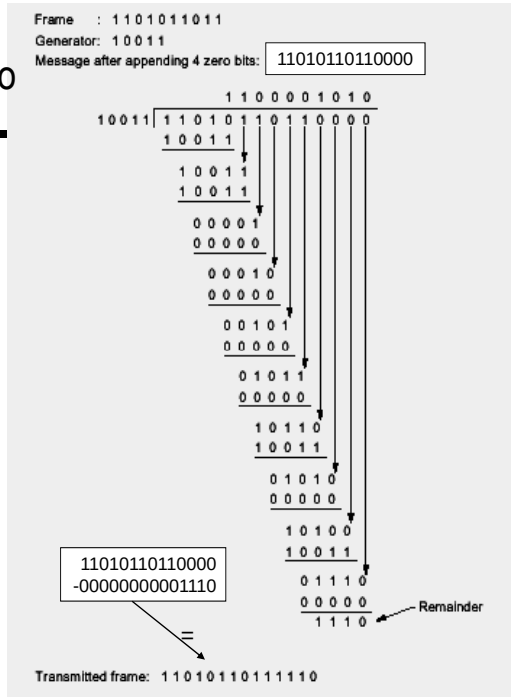
CRC: Algoritmo de Cálculo

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

- Princípio básico (decima)

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

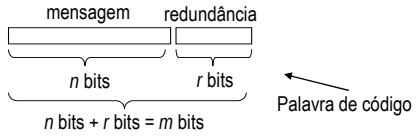
- Polinômios geradores:

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

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

Correção de erros

- Emprega duas estratégias básicas
  - Detecção de erros seguido por retransmissão
  - Detecção com correção automática de erros
- Para correção é necessário incluir informação para determinar se há um erro ou não e quais bits foram afetados
  - Mecanismo de FEC (*Forward Error Correction*)
  - Redundância adicional compensa?
    - Depende da taxa de erros



Código de Hamming

- Define uma regra para determinar a relação entre bits de dados e de redundância
- Distância de Hamming:
  - Quanto bits diferem entre duas palavras de código
  - Se duas palavras de código estão separadas por *d*, serão necessários *d* erros para transformar uma palavra de código em outra.
    - 2^n mensagens de dados são válidas
    - 2^m palavras de código $\rightarrow$ nem todas são usadas
  - Detectar *e* erros se necessita de palavras de código com distância *d=e+1*
  - Corrigir *e* erros se ncessita de palavras de código com distância d=*2e+1*

Código de Hamming (visão simplificada!!)

Exemplo:

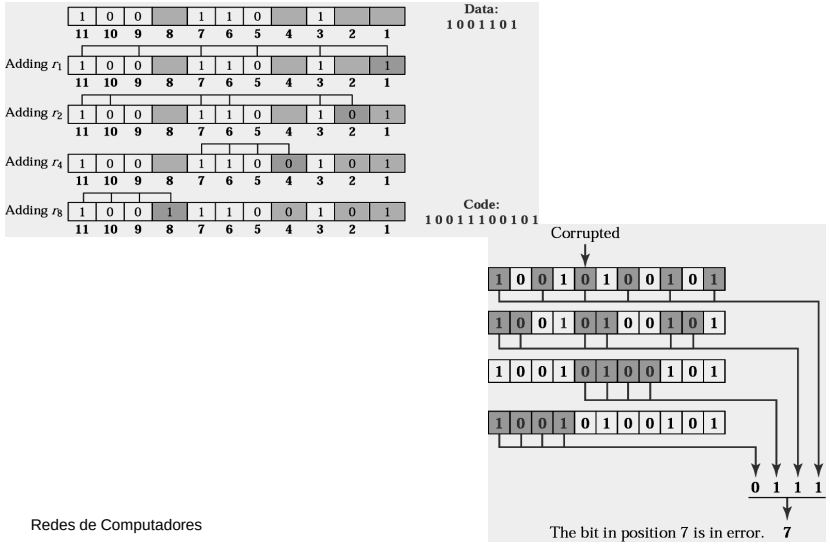
Letra A = 00000 00000
Letra B = 00000 11111
Letra C = 11111 00000
Letra D = 11111 11111

00000 01111 $\rightarrow$ Erro d=1 00000 11111 d=4 00000 00000

00000 00011 $\rightarrow$ Deduz $\rightarrow$ 00000 00000
correção de 2 erros (d=5)

Palavras de código
d=5

Código de Hamming



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