BCM0504

Natureza da Informação

Erros

Prof. Alexandre Donizeti Alves

Universidade Federal do ABC

Bacharelado em Ciência e Tecnologia

Bacharelado em Ciências e Humanidades

Terceiro Quadrimestre - 2018

Comunicação

- Até agora trabalhamos com informações e representações e códigos
  - Informações são armazenadas e/ou transmitidas
    - Como garantimos que não ocorrem **erros** no armazenamento ou na transmissão das informações?

Códigos

- Compressão: representamos mensagens com menos bits
  - Com vs sem perda
  - Frequentemente compressão é combinada à codificação

- Menos bits carregando a mesma informação $\Rightarrow$ cada bit é mais importante
  - **Consequência de erros em um único bit é mais séria**

Queremos códigos pequenos e pouco suscetíveis a erros

Veremos técnicas de detecção/correção de erros para geração de representações menos suscetíveis a erros

Comunicação

- Quando transmitimos informações, como temos certeza que o lado receptor recebeu corretamente a informação?

- **SE** **erros** e **perdas** ocorrem, porque isso acontece?

Comunicação

- A **perda de informação** pode ocorrer em todos os processos de comunicação, mas **é possível convivermos com ela**?
  - Depende!!!
  - Existem situações nas quais que é possível convivermos com perdas de informações e outras situações isso não é possível

Comunicação

- **Exemplo**: transmissão de imagem por uma rede de computadores:
  - Quando transmitimos um vídeo pelo YouTube (ou baixamos uma música pela Internet), se alguma mensagem é perdida não há problema, pois o volume de informações de uma mensagem é tão pequeno que a falta de informação de um *frame* (quadro) geralmente não fará diferença
    - Mesmo se fizer, a consequência geralmente não é nem significativa e nem relevante

Comunicação

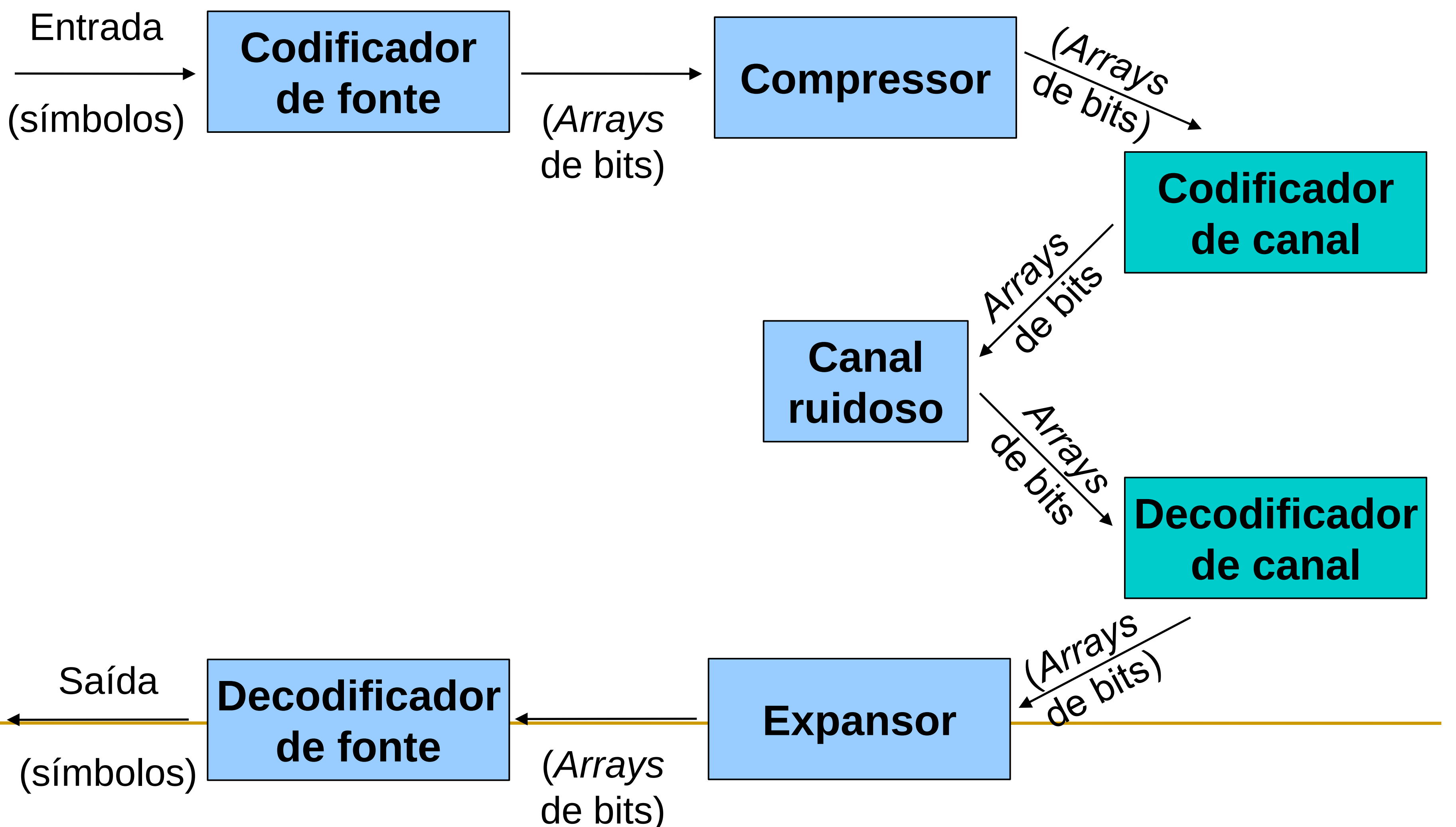
- Quando transmitimos uma imagem médica, como uma ressonância magnética, através de uma rede de computadores, a perda de um único quadro pode representar a detecção de um tumor!

- Portanto, existem situações e aplicações nas quais os erros nos processo de comunicação são tolerados e outras não

- **E como detectar esses erros?**

Modelo de Sistema de Comunicação

- **Modelo estendido**



Detecção/correção de erros

- **Codificador de canal**:
  - Adiciona bits à mensagem para possibilitar detecção/reparos de eventuais ruídos introduzidos

- **Decodificador de canal**:
  - Realiza a decodificação da mensagem transmitida pelo canal ruidoso
    - **Canal**:
      - Ar, fibra ótica, cabo, satélite, telefone, barramentos de computador etc.

Como ocorrem os erros?

- O modelo de comunicação apresentado é geral:
  - Transmissão de informação de um local para outro (comunicação) ou
  - Armazenamento de informação para uso posterior ou
  - Processamento da informação para ser réplica da entrada

Diferentes sistemas $\Rightarrow$ diferentes dispositivos como canal

Exemplos: link de comunicação, pen drive, computador etc.

Vários efeitos físicos podem causar erros

Exemplos: riscos em CD e DVD, memória pode falhar, linha telefônica pode estar ruidosa, quedas de energia etc.

Erros

- Para nossos propósitos, **erros** serão modelados como mudanças em um ou mais bits
  - De 0 para 1 e vice-versa
- Em cadeias de bits: assumiremos que erros são independentes
  - Embora em alguns casos erros em bits adjacentes não sejam independentes uns dos outros
    - Exemplo: quando risca um CD

Erros

- Abordagens para o **tratamento de erros**:
  - Ignorar o erro;
  - Eco (transmissão à origem de reflexos dos dados recebidos);
  - Sinalizar o erro;
  - Detectar e solicitar a retransmissão em caso de erro;
  - Detectar e corrigir os erros na recepção de forma automática

Detecção vs Correção

- Duas abordagens gerais:
  - **Detectar o erro**
    - Informar ao receptor que um erro ocorreu
  - **Corrigir o erro**
    - Decodificador de canal tenta reparar a mensagem

Em ambos casos, bits são adicionados à mensagem, tornando-as maiores (adiciona **redundância**)
Para informar como obter a informação original

Distância de Hamming

- **Como dizer se duas cadeias de bits são diferentes?**
  - Pelo número de bits que são diferentes entre as duas cadeias
    - **Distância de Hamming**
      - Richard W. Hamming (1915-1998)

Exemplo

- Qual é a distância de Hamming entre 0100 e 1110?

0 1 0 0

≠ = ≠ =

1 1 1 0

2 bits

Distância de Hamming

- Definida somente entre **duas** cadeias com o **mesmo número de bits**
- Efeito de erros introduzidos pelo canal pode ser descrito pela distância de Hamming entre
  - Cadeia de entrada e
  - Cadeia de saída

Sem erros $\Rightarrow$ distância = 0
Um erro $\Rightarrow$ distância = 1
Dois erros $\Rightarrow$ distância = 2 (se não ocorrem exatamente no mesmo bit)

Redundância

- Consideremos primeiro a transmissão de um único bit
  - Maneira de proteger: mandá-lo mais de uma vez
    - Na maioria das vezes espera-se que o valor não mude
    - **Exemplo**: mandar duas vezes
      - Mensagem 0 é substituída por 00 pelo codificador de canal
      - Mensagem 1 é substituída por 11 pelo codificador de canal
      - Decodificador alerta erro quando os dois bits recebidos são diferentes
      - **Mas se dois erros ocorrem?**
        - Se no mesmo bit $\Rightarrow$ como se não tivesse ocorrido erro
        - Se nos dois bits $\Rightarrow$ erro não é detectado

Redundância

- Se múltiplos erros são prováveis, uma maior redundância pode ajudar
  - **Exemplo**: para detectar erros duplos, mandar o mesmo bit 3 vezes
    - 000 e 111
    - A menos que os 3 bits recebidos sejam iguais, ocorreu erro $\Rightarrow$ detecta
      - Detecta erros simples e duplos
        - Mas quantos erros podem ter ocorrido?
        - Exemplo: em 011, primeiro errado $\Rightarrow$ bit enviado foi 1
        - Dois últimos bits errados $\Rightarrow$ bit enviado foi 0
      - E erros triplos?
        - **Ainda podem não ser detectados**

Redundância

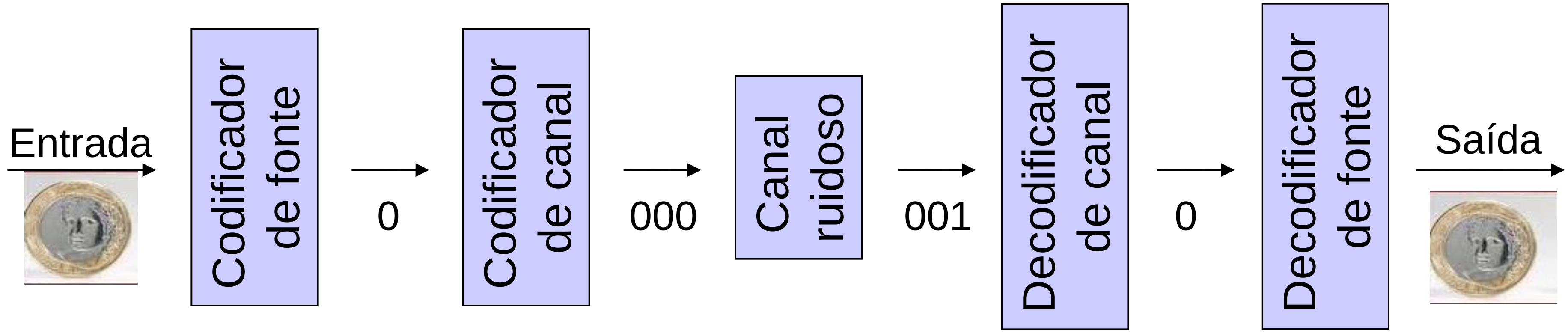
- E se queremos que o decodificador de canal corrija os erros, não só detecte?
  - Se sabe que há um erro no máximo e o bit é enviado 3 vezes, decodificador pode dizer:
    - Se um erro ocorreu
      - Se os 3 bits recebidos não são iguais
    - Qual era o valor original por lógica de maioria
      - Escolhe que valor de bit ocorre mais
  - Chamada **técnica de tripla redundância**
    - Pode ser usada para proteger canais de comunicação, memórias etc.

Redundância

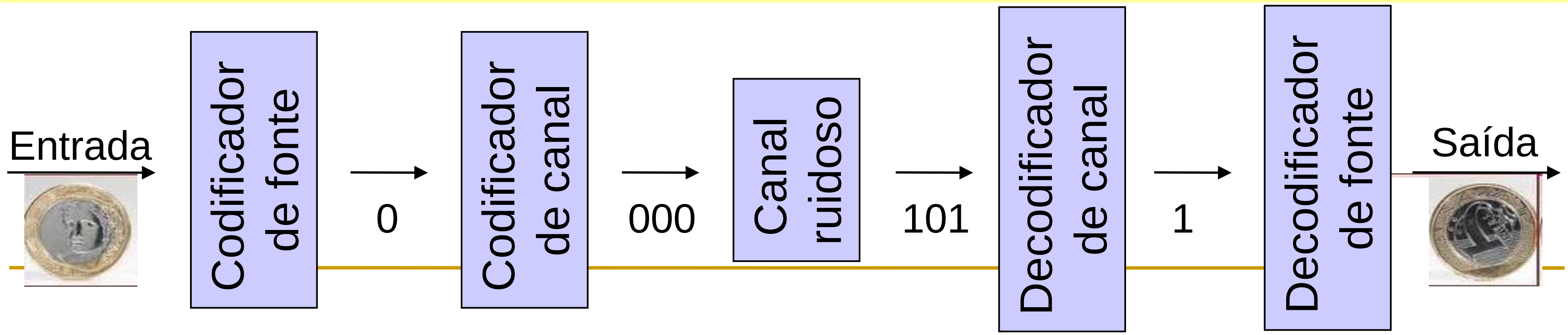
- **Tripla redundância** pode ser usada para corrigir um único erro ou detectar erros duplos
  - Mas não ambos!
    - Correção pressupõe ocorrência de um único erro
- Para ambos, deve aumentar a redundância

Redundância

Codificação de canal por redundância tripla e decodificação por correção de um erro, para canal introduzindo **um bit de erro**



Codificação de canal por redundância tripla e decodificação por correção de um erro, para canal introduzindo **dois bits de erro**



Redundância

- **Eficiência de redundância**
  - **Taxa de código**

$$\frac{\text{Número de bits antes de codificação de canal}}{\text{Número de bits após o codificador de canal}}$$

  - Valor entre 0 e 1
    - Redundância dupla = 0,5
    - Tripla redundância = 0,33
  - Maiores taxas são melhores
    - Representam menos acréscimos à mensagem

Redundância

- **Efetividade de redundância**
  - Se erros são improváveis:
    - Pode ser razoável ignorar o caso ainda mais raro de dois erros ocorrerem juntos e próximos
    - E redundância tripla é bastante efetiva
      - Consegue corrigir um erro
  - Mas erros podem ser concentrados
    - **Exemplo**: risco em CD
    - Neste caso, um erro é então provavelmente acompanhado de erros similares em bits adjacentes
    - E redundância tripla não é efetiva

Múltiplos bits

- Há várias técnicas para detectar erros em uma sequência de vários bits
  - Algumas também realizam correção de erros
- Exemplos:
  - **Paridade**
  - Códigos retangulares
  - **Códigos de Hamming**

Detecção vs correção de erros

- Correção de erros é mais útil que apenas detecção
  - Mas requer mais bits
    - E é então **menos eficiente**

Paridade

- Permite **detecção de erro em um único bit** de uma mensagem
  - Qualquer grupo de bits tem um número par ou ímpar de 1s
    - Bit de paridade é adicionado para tornar o número de 1s no grupo sempre par ou sempre ímpar
      - *Bit de paridade par*: torna número de 1s par
      - *Bit de paridade ímpar*: torna número de 1s ímpar
  - Um sistema opera com **paridade par** ou **ímpar**
    - Exemplo: operando com **paridade par**, verifica-se em grupos de bits recebidos se eles possuem número par de 1s
      - **Se número de 1s for ímpar, houve erro**

Paridade

- Exemplo: código BCD com bits de paridade (*P*)

PARIDADE PAR		PARIDADE ÍMPAR	
P	BCD	*P*	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

Paridade

- Bit de paridade pode ser acrescentado ao início ou fim do código
  - Depende do projeto do sistema

ATENÇÃO:
Número total de 1s, incluindo bit de paridade, é sempre par para a paridade par e sempre ímpar para a paridade ímpar

Paridade

- **Detecção de erro**:
  - Bit de paridade provê detecção de erro em **um único bit**
    - Ou número ímpar de erros, o que é pouco provável
    - Não pode detectar dois erros em um grupo
      - Imagine um sistema com paridade par e transmitindo a sequência: 0101
      - Com o bit de paridade fica 00101
      - Supondo que ocorra um erro no 3º bit, o receptor receberá: 00001 e apontará um erro, pois perceberá um número ímpar de 1s em uma paridade par
      - Agora, se ocorrerem 2 erros e o receptor receber: 01010, **não perceberá o erro**, pois o número de 1s será par

Paridade

- Exemplo: transmitir o código BCD 0101 com **paridade par**
  - Código transmitido = 00**1**01

Código BCD

Bit de paridade par

  - **Erro no terceiro bit a partir da esquerda**
    - Código recebido = 00**0**01

Bit errado

Bit de paridade par

Paridade

- **Exemplo**: considere 1 byte (8 bits)
  - Para permitir a detecção de um único bit errado, um bit de paridade (bit de checagem) pode ser adicionado
    - Cadeia fica com 9 bits
  - Considerando paridade par:
    - Bit de paridade = 1 $\Rightarrow$ número de bits 1 antes é ímpar
    - Bit de paridade = 0 $\Rightarrow$ número de bits 1 antes é par

Cadeia de 9 bits sempre terá número par de bits com valor 1

Exemplo: transmitir 00000001 $\Rightarrow$ 100000001
transmitir 00000011 $\Rightarrow$ 000000011

Paridade

- Decodificador de canal conta número de 1s
  - Se é ímpar (na paridade par) $\Rightarrow$ **há erro**
    - Número ímpar de erros
  - Não consegue corrigir danos
  - Ou mesmo dizer se o erro foi no bit de paridade
  - Não detecta erros duplos
    - Mais genericamente, números pares de erros

Paridade

- **Exemplo**: caractere A em ASCII é 10000001
  - Paridade par: bit de paridade = 0
    - Pois 10000001 tem número par de 1s
    - Transmite-se 010000001
  - Receptor calcula paridade da mensagem e compara-a com o bit de paridade recebido
    - Se forem iguais, transmissão está correta

Paridade

- **Exemplo**: caractere A em ASCII é 10000001
  - Processo vulnerável se houver mais do que um erro
    - Permitindo que esse passe até o destino sem ser detectado
      - Exemplo: 01**1**0**1**0001
        - Paridade par: correto, mas tem dois erros

Por que bit de paridade funciona?

- Para que seja detectado um único erro, é preciso que todos os códigos válidos tenham entre si uma distância de Hamming maior ou igual a 2
  - No caso do bit e paridade temos:
    - Paridade par: 00 e 11 têm d = 2
    - Paridade ímpar: 10 e 01 têm d = 2
    - Paridade par: 000, 101, 110 e 011 têm d = 2
    - Paridade ímpar: 100, 001, 010 e 111 têm d = 2
    - etc.
  - Se a distância fosse 1, um erro faria com que um código válido se transformasse em outro válido
    - Exemplo: 00 e 01

Detectando múltiplos erros

- No caso geral, para criar um código que detecte *L* erros, precisamos que a distância mínima (d_{min}) entre os padrões seja maior ou igual a $L + 1$
  - Exemplo:
    - Para detectar 3 erros, é preciso uma distância mínima de 4

Paridade

- É eficiente
  - Taxa de código no exemplo de um byte = 8/9
  - Usada em muitas aplicações de hardware (onde uma operação pode ser repetida em caso de dificuldade, ou onde é útil a simples detecção de erros)
    - Exemplo: barramento PCI
- Mas é pouco efetiva
  - Mais usada quando probabilidade de erro é muito pequena
  - E quando não há razão para supor que erros em bits adjacentes irão ocorrer em conjunto

Exercícios

- Associe o bit de paridade par apropriado para os seguintes grupos de códigos:
  - 1010
  - 111000
  - 101101
  - 1000111001001
  - 101101011111

Solução

- Associe o bit de paridade par apropriado para os seguintes grupos de códigos:
  - **0**1010
  - **1**111000
  - **0**101101
  - **0**100011100101
  - **1**101101011111

Exercícios

- Um sistema de paridade ímpar recebe os seguintes grupos de códigos: 10110, 11010, 110011, 110101110100 e 1100010101010. Determine quais grupos estão com erro.

Solução

- Um sistema de paridade ímpar recebe os seguintes grupos de códigos: 10110, 11010, **110011**, 110101110100 e **1100010101010**. Determine quais grupos estão com erro.

Como é informado que a paridade é ímpar, qualquer grupo com um número par de 1s está incorreto

Códigos de Hamming

- Supor correção de um único erro
  - Ignorando a possibilidade de erros múltiplos
  - Richard Hamming inventou conjunto de códigos com número mínimo de bits de paridade extra
  - Cada bit adicionado no codificador permite uma checagem de paridade no decodificador
    - Um bit de informação pode ser usado para ajudar a localizar o erro

Códigos de Hamming

- Exemplo: 3 bits extras $\Rightarrow$ 3 testes podem identificar até 8 condições
  - Uma delas será "não erro"
  - 7 remanescentes para identificar a localização de até 7 locais onde erro possa estar
  - Bloco transmitido pode então ter 7 bits
    - 3 para checagem de erros
    - 4 para carregar os dados (carga)
- Exemplo: 4 bits de paridade $\Rightarrow$ blocos com 15 bits
  - 11 bits para carregar mensagem (carga)

Códigos de Hamming

- Número de bits de paridade
  - Se o número de bits de dados for *d*, o número de bits de paridade *p* é determinado por:

$$2^p \geq d + p + 1$$

- Exemplo: 4 bits de dados (mensagens com 4 bits)
  - $p = 2 \Rightarrow 2^p = 2^2 = 4$ e $d + p + 1 = 4 + 2 + 1 = 7$
    - » $4 < 7$, relação não é satisfeita
  - $p = 3 \Rightarrow 2^p = 2^3 = 8$ e $d + p + 1 = 4 + 3 + 1 = 8$
    - » $8 \geq 8$, relação é satisfeita

Precisa de 3 bits de paridade para proporcionar correção de um único erro em quatro bits de dados

Códigos de Hamming

- Detecção e correção são proporcionados por todos os bits no grupo
  - De paridade e de dados
    - $\Rightarrow$ bits de paridade também são verificados

Códigos de Hamming

- Inserção de bits de paridade no código:
  - Arranjar os bits adequadamente no código
    - Exemplo: com 4 bits de dados e 3 bits de paridade
      - Bit mais à esquerda é designado como bit 1, próximo como 2 e assim por diante
        - bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7
      - Bits de paridade são colocados nas posições numeradas em correspondência às potências 1, 2, 4, 8, ...:
        - P_1, P_2, D_1, P_3, D_2, D_3, D_4
        - P_i é um bit de paridade em particular e D_i é um bit de dado em particular

Códigos de Hamming

- Determinação dos valores de bits de paridade
  - Designar apropriadamente o valor 0 ou 1 a cada bit de paridade
    - Cada bit de paridade provê verificação em outros determinados bits no código total
      - Deve saber valor desses outros bits para determinar o de paridade

Códigos de Hamming

- Determinação dos valores de bits de paridade
  - Para determinar o valor do bit, primeiro numere cada posição de bit em binário
    - Escreva o binário equivalente a cada decimal da posição
      - Segunda e terceira linhas da tabela:

Tabela de posicionamento dos bits para um código de correção de erro de 7 bits

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados (D_n)							
Bits de paridade (P_n)							

Códigos de Hamming

- Determinação dos valores de bits de paridade
  - Em seguida, indique a localização dos bits de dados e de paridade
    - Primeira linha da tabela:

Tabela de posicionamento dos bits para um código de correção de erro de 7 bits

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados (D_n)							
Bits de paridade (P_n)							

Número da posição em binário do bit de paridade P_1 tem 1 no dígito mais à direita: esse bit verifica as posições de todos os bits, incluindo ele, que têm 1 na mesma posição nos números de posição em binário (bits 1, 3, 5 e 7)

Códigos de Hamming

- Determinação dos valores de bits de paridade

Número da posição em binário do bit de paridade P_2 *tem 1 no dígito do meio*: esse bit verifica as posições de todos os bits, incluindo ele, que têm 1 na mesma posição (bits 2, 3, 6 e 7)

Tabela de posicionamento dos bits para um código de co… …o de 7 bits

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados (D_n)							
Bits de paridade (P_n)							

Número da posição em binário do bit de paridade P_3 *tem 1 no dígito mais à esquerda*: esse bit verifica as posições de todos os bits, incluindo ele, que têm 1 na mesma posição (bits 4, 5, 6 e 7)

Códigos de Hamming

- Determinação dos valores de bits de paridade
  - Em cada caso, é designado ao bit de paridade o valor que torna a quantidade de 1s, no conjunto de bits que ele verifica, par ou ímpar
    - Dependendo do que for especificado

Exemplo 01

- Determine o código de Hamming para o número BCD 1001, usando paridade par
  - **Passo 1**: determinar número de bits de paridade necessários
    - *p* = 3 são suficientes (visto em slide anterior)
      - Total de bits do código = 4 (dados) + 3 (paridade) = 7

Exemplo 01

- Determine o código de Hamming para o número BCD 1001, usando paridade par
  - **Passo 2**: construir tabela de posições de bits e inserir os bits de dados

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados			1		0	0	1
Bits de paridade							

Exemplo 01

- Determine o código de Hamming para o número BCD 1001, usando paridade par
  - **Passo 3**: Determinar os bits de paridade

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados			1		0	0	1
Bits de paridade	0	0		1			

- P_1 verifica bits das posições 1, 3, 4 e 7: tem que ser 0 para nº de 1s ser par no grupo
- P_2 verifica bits das posições 2, 3, 6 e 7: tem que ser 0 para nº de 1s ser par no grupo
- P_3 verifica bits das posições 4, 5, 6 e 7: tem que ser 1 para no de 1s ser par no grupo

Exemplo 01

- Determine o código de Hamming para o número BCD 1001, usando paridade par
  - **Passo 4**: Código resultante = **0011001**

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados			1		0	0	1
Bits de paridade	0	0		1			

Exemplo 02

- Determinar código de Hamming para bits de dados 10110 usando paridade ímpar
  - **Passo 1**: determinando número de bits de paridade necessários
    - $d = 5$
    - Tentando $p = 4$
      - $2^p = 2^4 = 16$
      - $d + p + 1 = 5 + 4 + 1 = 10$
      - $\Rightarrow$ 4 bits são suficientes
    - Total de bits do código = 5 + 4 = 9

Exemplo 02

- Determinar código de Hamming para bits de dados 10110 usando paridade ímpar
  - **Passo 2**: construir tabela de posições de bits e inserir bits de dados

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
POSIÇÃO DOS BITS	1	2	3	4	5	6	7	8	9
NÚMERO DA POS. EM BINÁRIO	0001	0010	0011	0100	0101	0110	0111	1000	1001
Bits de dados			1		0	1	1		0
Bits de paridade									

P_4 está na posição 8

Exemplo 02

- Determinar código de Hamming para bits de dados 10110 usando paridade ímpar
  - **Passo 3**: determinar bits de paridade

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
POSIÇÃO DOS BITS	1	2	3	4	5	6	7	8	9
NÚMERO DA POS. EM BINÁRIO	0001	0010	0011	0100	0101	0110	0111	1000	1001
Bits de dados			1		0	1	1		0
Bits de paridade	1	0		1				1	

P_1 verifica bits das posições 1, 3, 5, 7 e 9: tem que ser 1 para nº de 1s ser ímpar no grupo

P_2 verifica bits das posições 2, 3, 6 e 7: tem que ser 0 para nº de 1s ser ímpar no grupo

P_3 verifica bits das posições 4, 5, 6 e 7: tem que ser 1 para nº de 1s ser ímpar no grupo

P_4 verifica bits das posições 8 e 9: tem que ser 1 para no de 1s ser ímpar no grupo

Exemplo 02

- Determinar código de Hamming para bits de dados 10110 usando paridade ímpar
  - **Passo 4**: código combinado = **101101110**

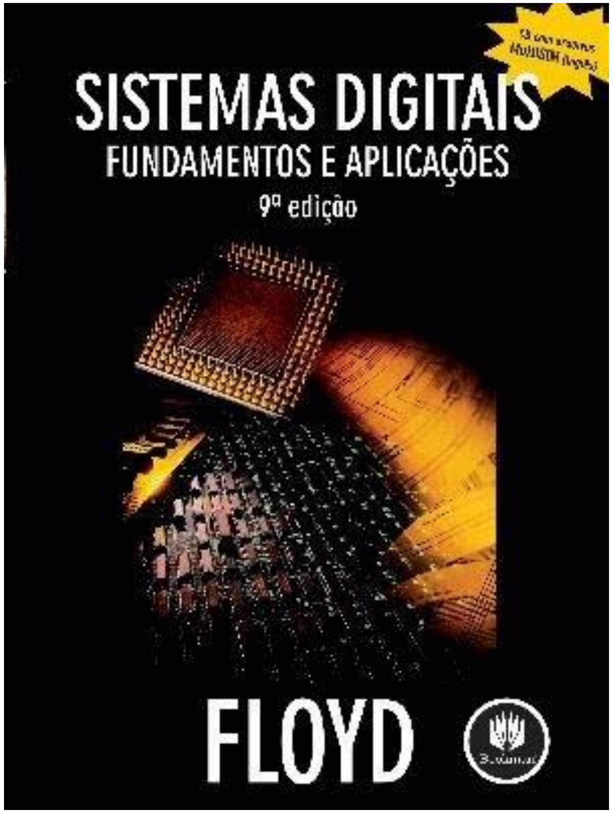
DESIGNAÇÃO DOS BITS POSIÇÃO DOS BITS NÚMERO DA POS. EM BINÁRIO	P_1 1 0001	P_2 2 0010	D_1 3 0011	P_3 4 0100	D_2 5 0101	D_3 6 0110	D_4 7 0111	P_4 8 1000	D_5 9 1001
Bits de dados			1		0	1	1		0
Bits de paridade	1	0		1				1	

Exercícios

- Determinar o código Hamming para o número BCD 1000 usando paridade par

- Determinar o código de Hamming para 11001 usando paridade ímpar

Referência



Capítulo 2