

III-1

Códigos detetores e corretores de erros

Comunicações
(16 de dezembro de 2018)



Sumário

1. Aspetos gerais sobre a comunicação digital

- Comportamento do canal
- Causas da existência de erros

2. Códigos detetores e corretores de erros

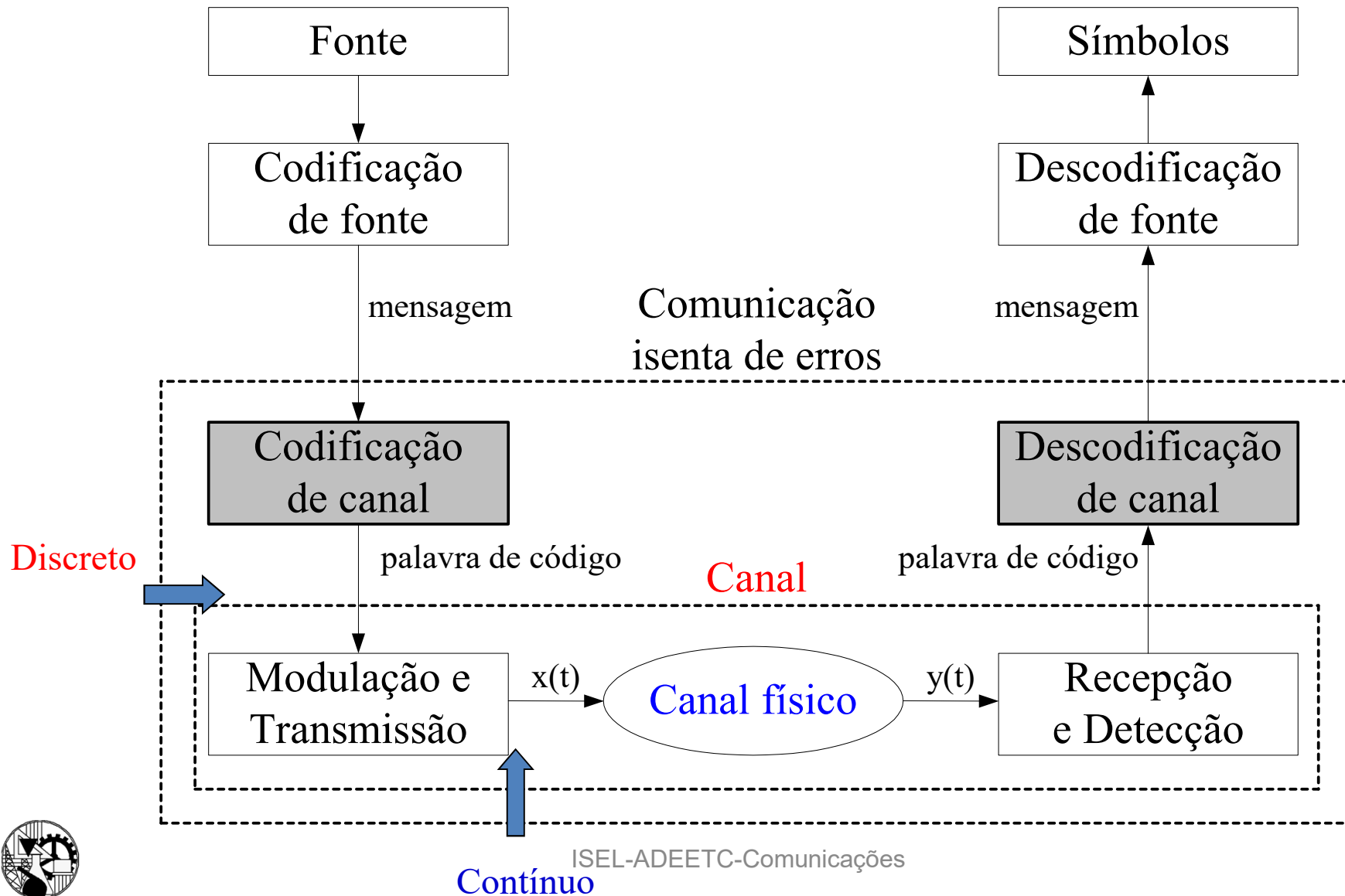
- Códigos de bloco linear (n,k)
- Características dos códigos
- Capacidades de deteção e correção
- Códigos de repetição e bit de paridade
- Código de *Hamming*
- CRC – *Cyclic Redundancy Check*

3. Deteção e Correção

4. Exercícios

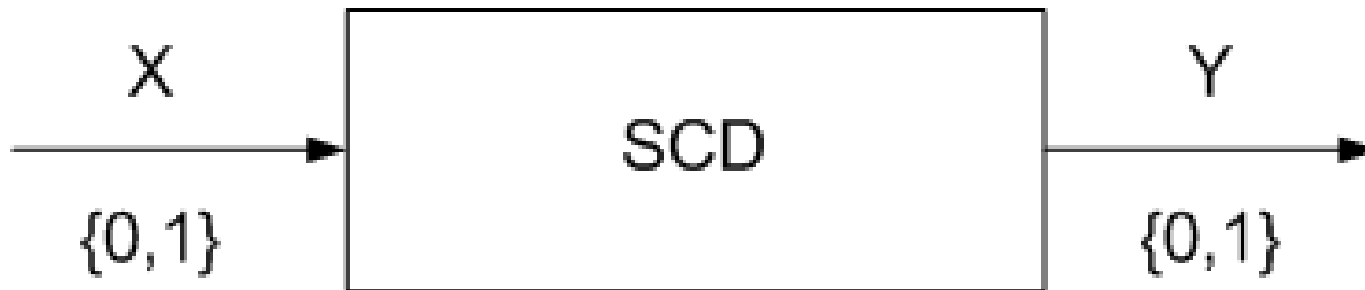


1. Cenário de utilização



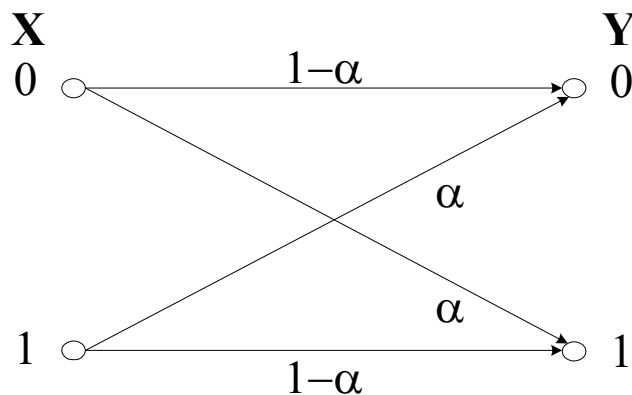
1. Modelo de canal discreto

- O canal é analisado através de modelo discreto usando variáveis aleatórias (v.a.)
- Do ponto de vista da transmissão, um SCD pode ser visto através de modelo probabilístico
- A probabilidade de erro por troca de bit não é nula



1. Modelo de canal discreto

- O canal é analisado através de modelo discreto usando variáveis aleatórias (v.a.)
- Modelo BSC - *binary symmetric channel*



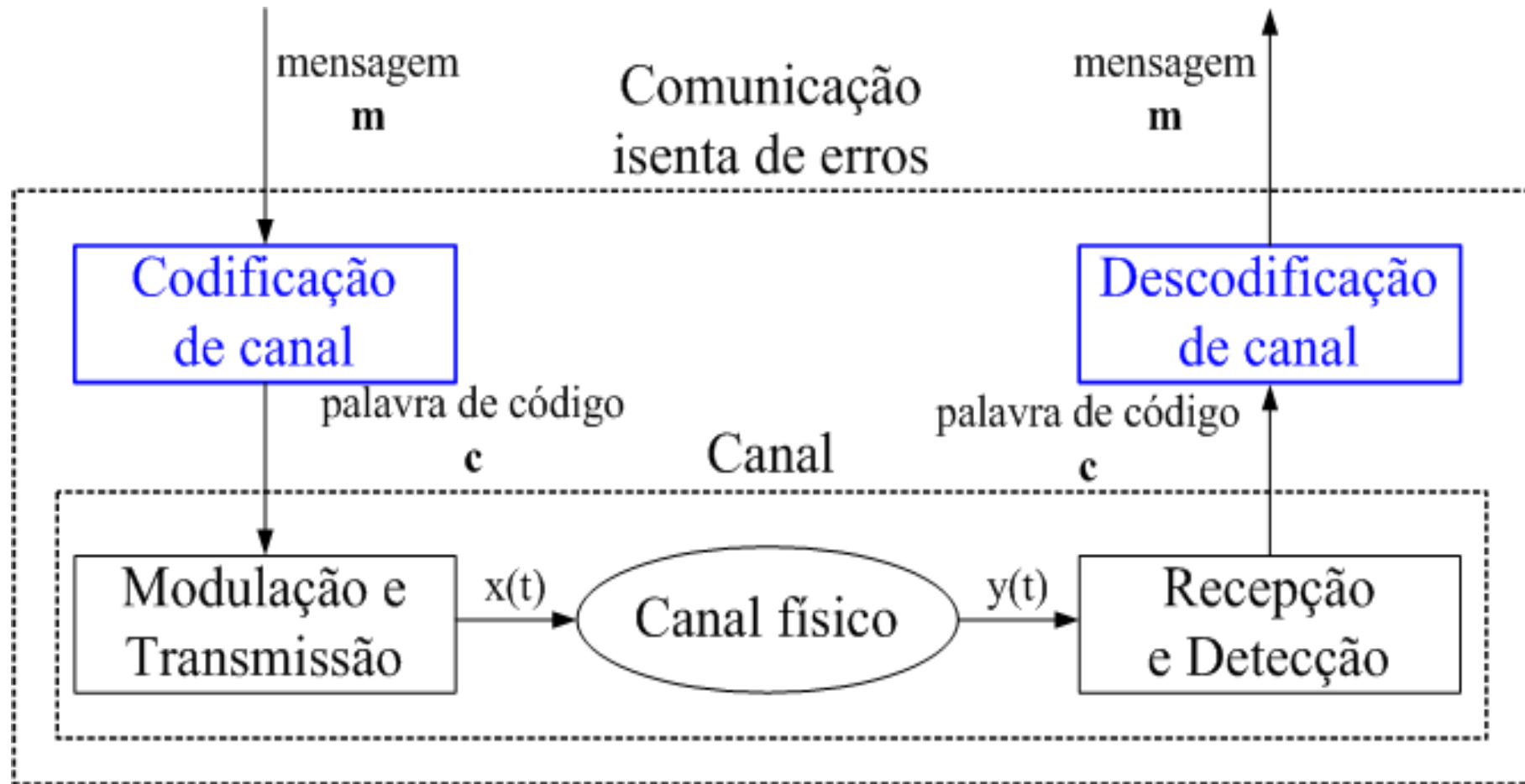
Probabilidade de erro de bit

$$\begin{aligned} P_e &= P(y_0, x_1) + P(y_1, x_0) \\ &= P(y_0|x_1)P(x_1) + P(y_1|x_0)P(x_0) \\ &= \alpha P(x_1) + \alpha P(x_0) \\ &= \alpha \end{aligned}$$

A probabilidade de erro define o **BER (*Bit Error Rate*)** do canal. É a taxa de erros por bit.



1. Cenário de Utilização: detalhe



2. Códigos de controlo de erros

- A deteção e correção são obtidas pela introdução de redundância na mensagem original
- Essa redundância é função da mensagem
- Códigos a analisar: repetição; bit de paridade par; Hamming e CRC
- Os códigos de canal são utilizados nos modos:
 - FEC - **F**orward **E**rror **C**orrection
 - ARQ - **A**utomatic **R**epeat **R**e**Q**uest



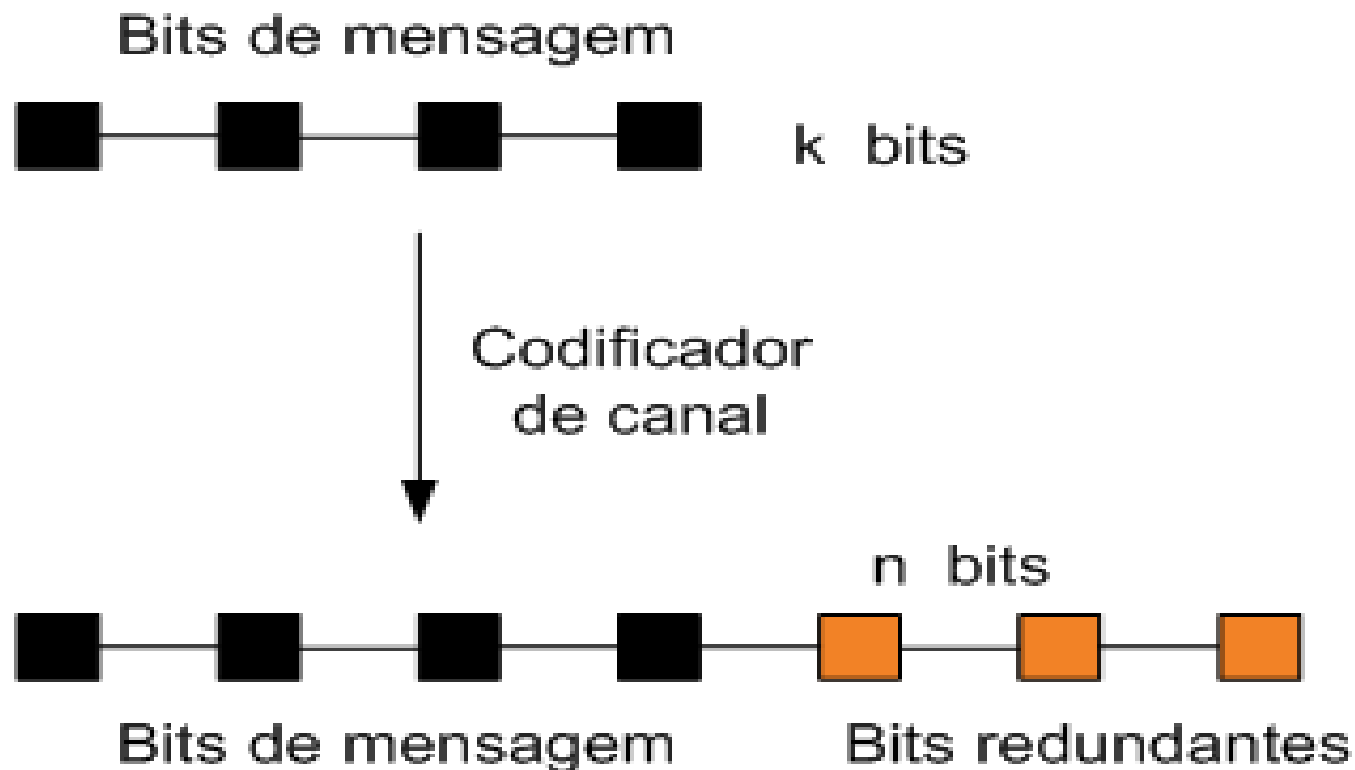
2. Modos de funcionamento

- **FEC - Forward Error Correction**
 - Modo de correção de erros
 - O recetor recebe as palavras, deteta eventuais erros e corrige-os
- **ARQ - Automatic Repeat ReQuest**
 - Modo de deteção de erros
 - O recetor recebe as palavras e deteta eventuais erros; em caso de erro, solicita a retransmissão




2. Códigos de bloco (n,k)

- Codificador de bloco
- Cada bloco de k **bits de mensagem** origina uma **palavra de código** com n bits
- k = número de bits de mensagem
- n = número de bits de palavra de código



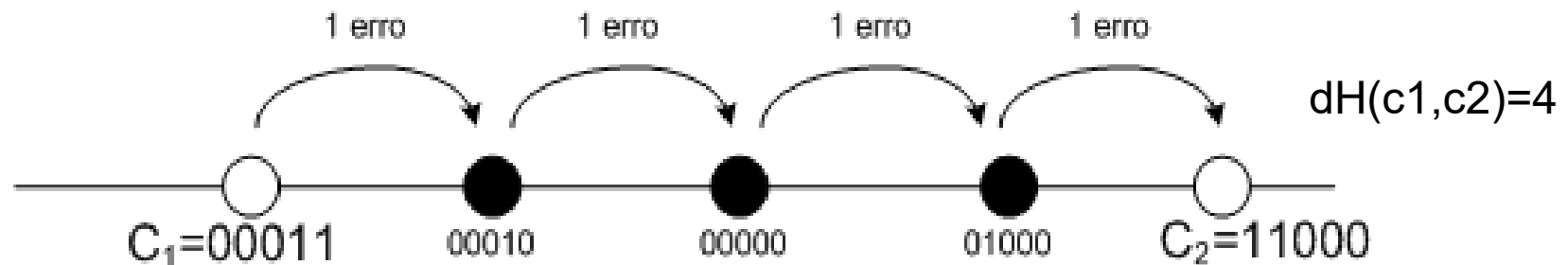
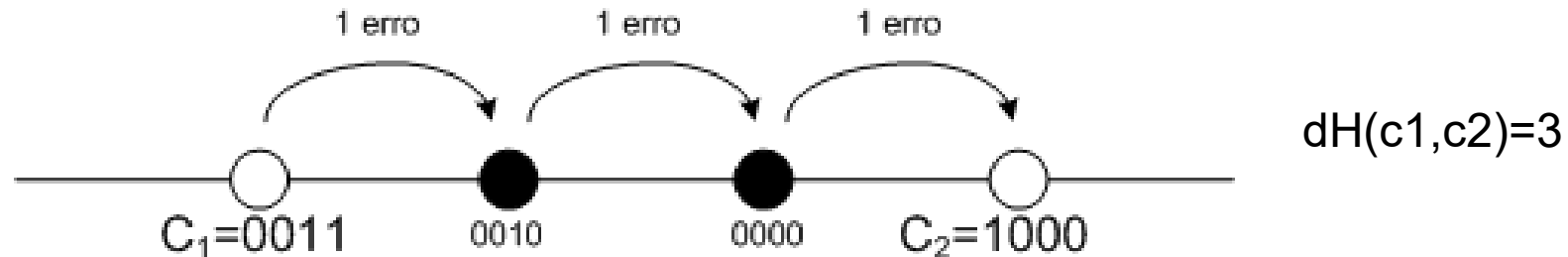
2. Códigos de bloco (n,k): propriedades

1. *Code rate* (ritmo) $R = \frac{k}{n}$, medida de eficiência
2. Distância de Hamming (**dH**): número de dígitos em que diferem duas quaisquer palavras do código
3. **Distância mínima (dmin)**: é a menor distância de Hamming entre duas quaisquer palavras do código; depende da redundância:
Majorante  $d_{\min} \leq 1 + q, \quad q = n - k$
4. Deteta todos os padrões até “l” erros: $l \leq d_{\min} - 1$
5. Corrige todos os padrões até “t” erros: $t \leq \lfloor \frac{d_{\min} - 1}{2} \rfloor$
6. Deteta “l” erros e corrige “t” erros: $d_{\min} \geq l + t + 1, \quad \text{com } l > t$



2. Códigos de bloco (n,k): distância

- Distância de Hamming entre palavras



2. Códigos lineares de bloco (n,k)

- Bloco: todas as palavras têm a mesma dimensão
- Linear:
 - o vetor nulo pertence ao código
 - a soma modular de quaisquer duas palavras do código é ainda uma palavra do código

n = número de bits da palavra de código

2^n palavras possíveis

k = número de bits da mensagem

2^k palavras de código

$q = n - k$, é o número de bits redundantes

Seja $\mathbf{m} = [m_0 \ m_1 \ \dots \ m_{k-1}]$ a mensagem e \mathbf{c} a palavra de código

Podem ser sistemáticos ou não sistemáticos; exemplos destas formas:

• sistemática: $\mathbf{c} = [m_0 \ m_1 \ \dots \ m_{k-1} \ b_0 \ b_1 \ \dots \ b_{q-1}]$

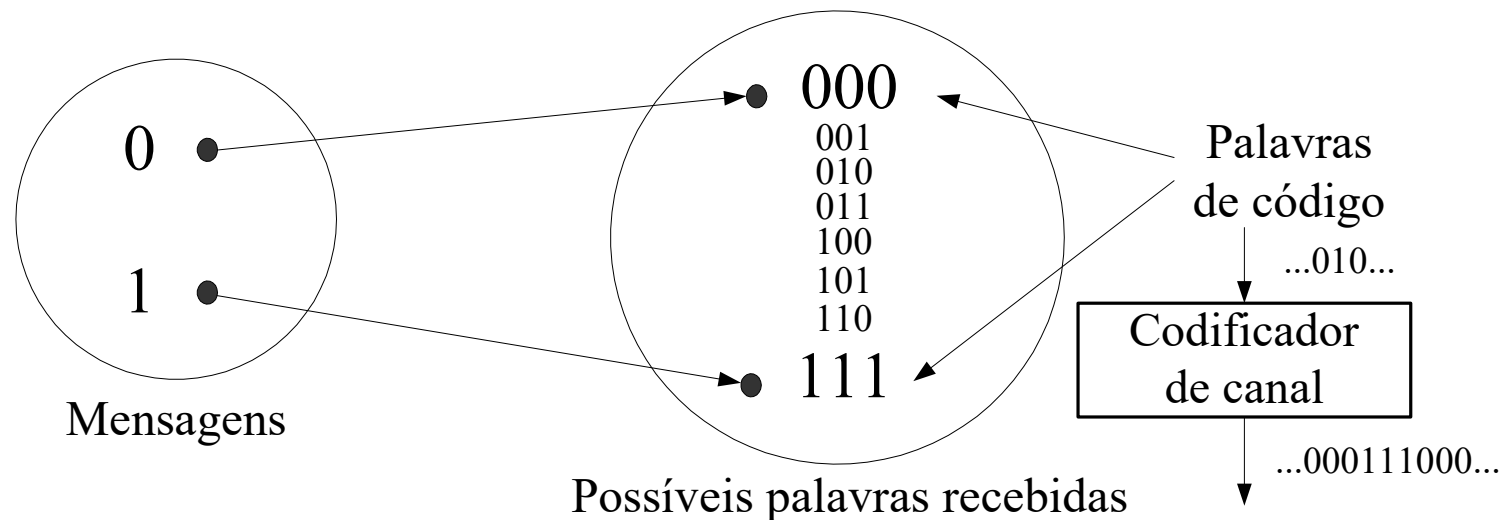
• não sistemática: $\mathbf{c} = [m_0 \ b_1 \ b_0 \ m_1 \ \dots \ m_{k-1} \ \dots \ b_{q-1}]$



2. Código de repetição (3,1)

- Consiste na repetição da mensagem
- Exemplo: código (3,1), na forma (n,k) com k=1 bit de mensagem e n=3 bit na palavra de código

m	c
0	000
1	111



Usa $2^k = 2^1 = 2$ palavras de $2^n = 2^3 = 8$ possíveis



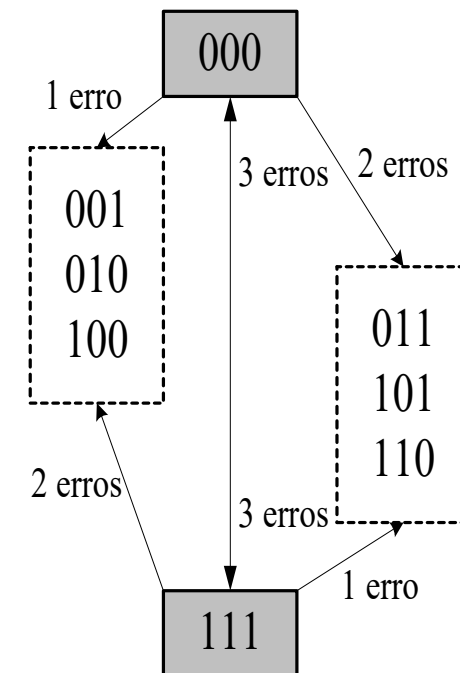
2. Código de repetição (3,1)

- Descodificação realizada por maioria
- A **distância** entre as palavras de código, garante que:
 - Deteta todos os erros de 1 e 2 bit
 - Corrige todos os erros de 1 bit

Considerando um BSC com $\alpha = 10^{-5}$,
tem-se que:

$$\begin{aligned} P(1, 3) &= C_1^3 \alpha^1 (1 - \alpha)^2 = \frac{3!}{2!1!} \alpha (1 - \alpha)^2 \\ &= 3\alpha - 6\alpha^2 + 3\alpha^3 \approx 3 \times 10^{-5} \end{aligned}$$

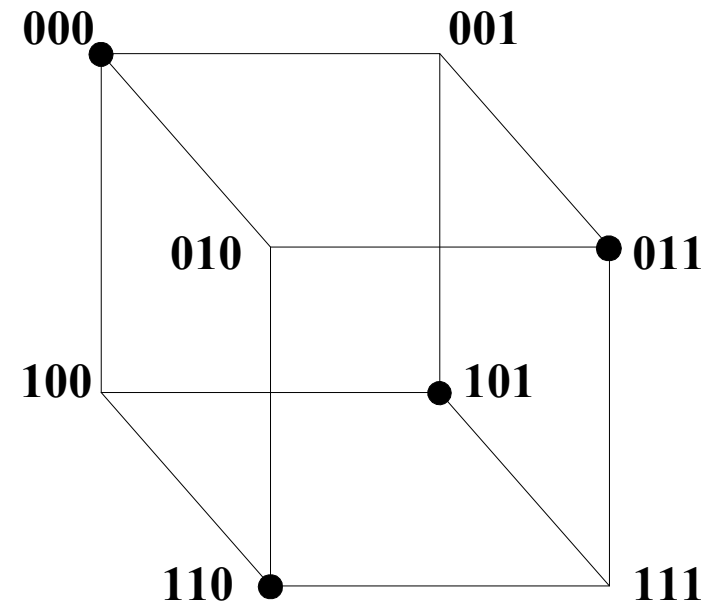
$$\begin{aligned} P(2, 3) &= C_2^3 \alpha^2 (1 - \alpha)^1 = \frac{3!}{1!2!} \alpha^2 (1 - \alpha) \\ &= 3\alpha^2 - 3\alpha^3 \approx 3 \times 10^{-10} \end{aligned}$$



2. Código bit de paridade (3,2) - paridade par

- Adicionar um bit no final da mensagem; este bit é a soma módulo 2 dos bits da mensagem
- A palavra de código é $\mathbf{c} = [m_0 \ m_1 \ m_0 \oplus m_1]$

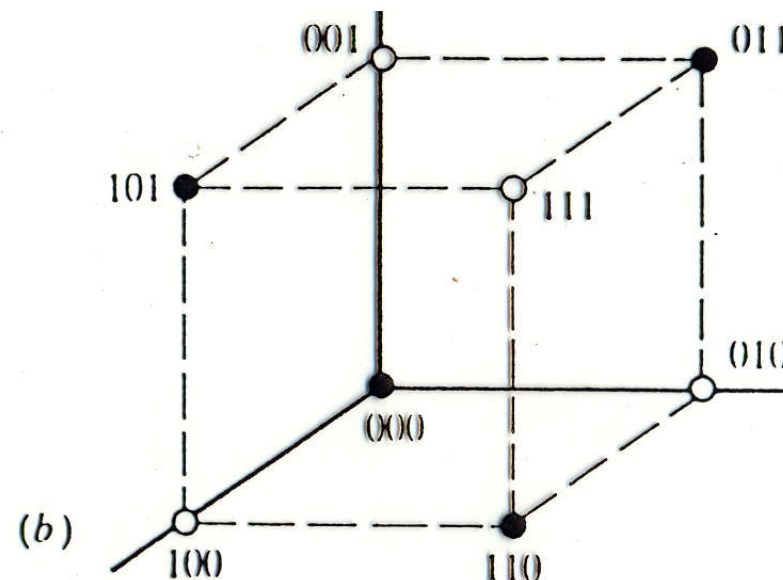
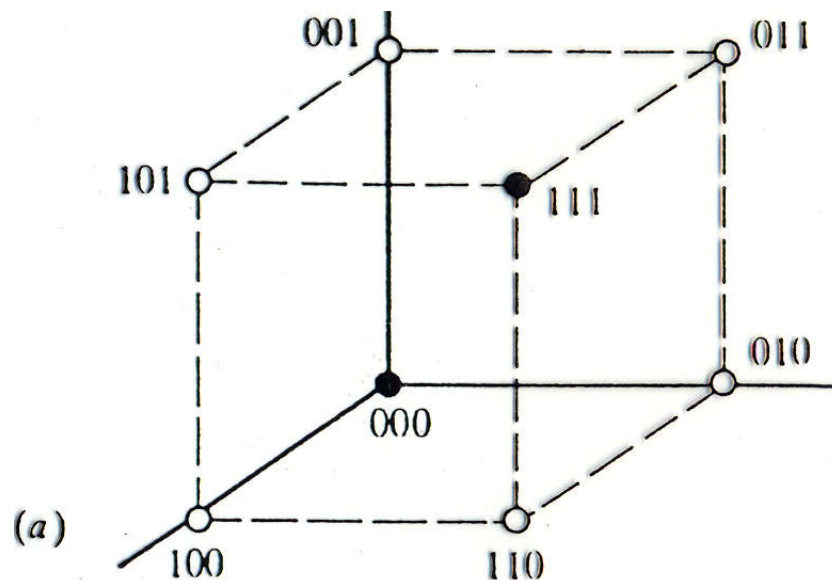
m	c
00	000
01	011
10	101
11	110



- Deteta a presença de 1 e 3 bits errados
- Não tem capacidade de correção; não realiza FEC



2. Palavras de código: vetores



- Palavras de 3 bit

(a) código de repetição (3,1); 3 arestas entre as 2 palavras de código

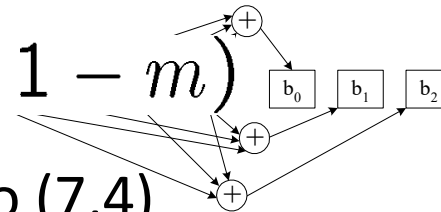
(b) código de bit de paridade (3,2); 2 arestas entre 2 palavras de código mais próximas



2. Códigos de Hamming

- Família de códigos lineares de bloco
- Têm $d_{\min}=3$, logo corrigem todos os erros de 1 bit
- A motivação: $P(2, n) \ll P(1, n)$
- Definidos por um parâmetro inteiro $m (\geq 2)$ tal que:

$$(n, k) = (2^m - 1, 2^m - 1 - m)$$



Por exemplo, com $m=3$ tem-se o código (7,4)

$$\mathbf{c} = [m_0 \ m_1 \ m_2 \ m_3 \ b_0 \ b_1 \ b_2]$$

Equações de paridade:

$$b_0 = m_1 \oplus m_2 \oplus m_3$$

$$b_1 = m_0 \oplus m_1 \oplus m_3$$

$$b_2 = m_0 \oplus m_2 \oplus m_3$$



2. Hamming (7,4): todas as palavras

Listagem das 16 palavras de código e respetivos pesos de Hamming

<u>Palavra de código</u>							<u>Peso</u>	<u>Palavra de código</u>							<u>Peso</u>
0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	3
0	0	0	1	1	1	1	4	1	0	0	1	1	0	0	3
0	0	1	0	1	0	1	3	1	0	1	0	1	1	0	4
0	0	1	1	0	1	0	3	1	0	1	1	0	0	1	4
0	1	0	0	1	1	0	3	1	1	0	0	1	0	1	4
0	1	0	1	0	0	1	3	1	1	0	1	0	1	0	4
0	1	1	0	0	1	1	4	1	1	1	0	0	0	0	3
0	1	1	1	1	0	0	4	1	1	1	1	1	1	1	7

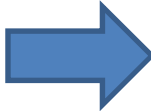
O menor peso de Hamming para palavras não nulas é 3, logo:

$$d_{\min} = 3, l = 2 \text{ e } t = 1$$



2. Códigos Cíclicos - CRC

- Os códigos cíclicos são uma sub-classe dos códigos lineares de bloco
 - **Linear:** o vetor nulo pertence ao código; a soma modular de duas palavras do código é ainda uma palavra do código
 - **Bloco:** todas as palavras têm a mesma dimensão de n bits
- Nos códigos cíclicos tem-se que qualquer rotação cíclica de qualquer ordem sobre uma palavra de código é ainda uma palavra de código
- Exemplo: código de bit de paridade par (3,2)



m	c
00	000
01	011
10	101
11	110



2. Códigos Cíclicos

- Tem-se $c(X) = m(X)g(X)$ em que:
 - $c(x)$ é a palavra de código – polinómio de grau $n-1$
 - $m(x)$ depende da mensagem – polinómio de grau $k-1$
 - $g(x)$ – polinómio gerador de grau q
- As palavras de código $c=[c_{n-1} \ c_{n-2} \ \dots \ c_1 \ c_0]$ podem ser analisadas como polinómios:
 - $c(X) = c_{n-1} X^{n-1} + c_{n-2} X^{n-2} + \dots + c_1 X + c_0$
- O número de bits redundantes (de paridade) corresponde ao grau do polinómio gerador



2. Polinómio Gerador

- Determinado polinómio $g(X)$ de grau q é gerador de um código (n,k) , com $q=n-k$, caso seja factor de X^n+1
- Ser fator de X^n+1 implica que $\text{resto}\left[\frac{X^n+1}{g(X)}\right]=0$
- Assim, a fatorização do polinómio X^n+1 é importante, neste contexto
- Através desta fatorização, conseguimos obter polinómios geradores para códigos de diferentes dimensões



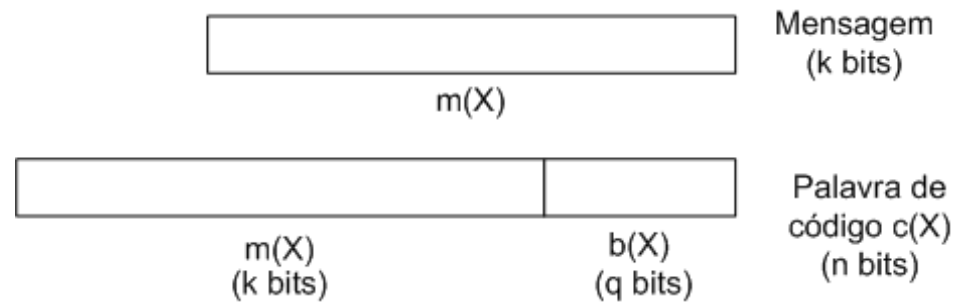
2. Polinómios Geradores

Código	Polinómio gerador $g(X)$
CRC4	$X^4+X^3+X^2+X+1$
CRC7	$X^7+X^6+X^4+1$
CRC12	$X^{12}+X^{11}+X^3+X^2+X+1$
CRC16	$X^{16}+X^{15}+X^2+1$
CRC-CCITT	$X^{16}+X^{12}+X^5+1$
CRC32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$



2. CRC – Cyclic Redundancy Check

- Num código cíclico sistemático, as palavras têm a seguinte organização



- Os bits $b(X)$, que constituem um polinómio de grau $q-1$ designam-se por **CRC-Cyclic Redundancy Check**
- A palavra de código é dada por

$$c(X) = m(X)X^q + b(X) = m(X)X^q + \text{resto} \left[\frac{m(X)X^q}{g(X)} \right]$$



2. CRC – Cyclic Redundancy Check

- O CRC resulta do resto da divisão de polinómios entre:
 - A mensagem deslocada de q bits para a esquerda
 - O polinómio gerador do código

$$CRC = b(X) = \text{resto} \left[\frac{m(X)X^q}{g(X)} \right]$$

- Dado que $g(X)$ tem grau q, resulta que $b(X)$ terá grau q-1, sendo constituído por q bits
- Assim, temos palavra de código com n bits (k de mensagem e q de paridade)



2. CRC – Cyclic Redundancy Check

- Exemplo de cálculo do CRC para código (7,4)

- $m(X) = X^3 + 1 = [1 \ 0 \ 0 \ 1]$

- $g(X) = X^3 + X^2 + 1 = [1 \ 1 \ 0 \ 1]$

$$CRC = b(X) = \text{resto} \left[\frac{m(X)X^q}{g(X)} \right] = \text{resto} \left[\frac{(X^3 + 1)X^3}{X^3 + X^2 + 1} \right] = \text{resto} \left[\frac{X^6 + X^3}{X^3 + X^2 + 1} \right]$$

$$= X + 1$$

1 0 0 1 0 0 0	1 1 0 1
1 1 0 1	1 1 1 1
0 1 0 0 0	
1 1 0 1	
0 1 0 1 0	
1 1 0 1	
0 1 1 1 0	
1 1 0 1	
0 0 1 1	

$$c(X) = m(X)X^3 + b(X) = (X^3 + 1)X^3 + (X + 1).$$

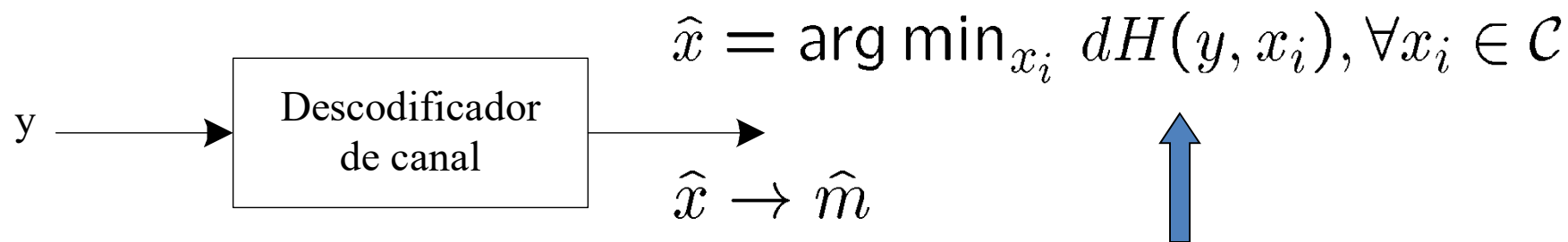
$$= X^6 + X^3 + X + 1$$

$$= [1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1]$$



2. Descodificador de canal: características

- O decodificador:
 1. recebe a palavra \mathbf{y} (possivelmente com erros)
 2. estima a palavra de código \hat{x} que lhe deu origem
 3. estima a mensagem \hat{m}

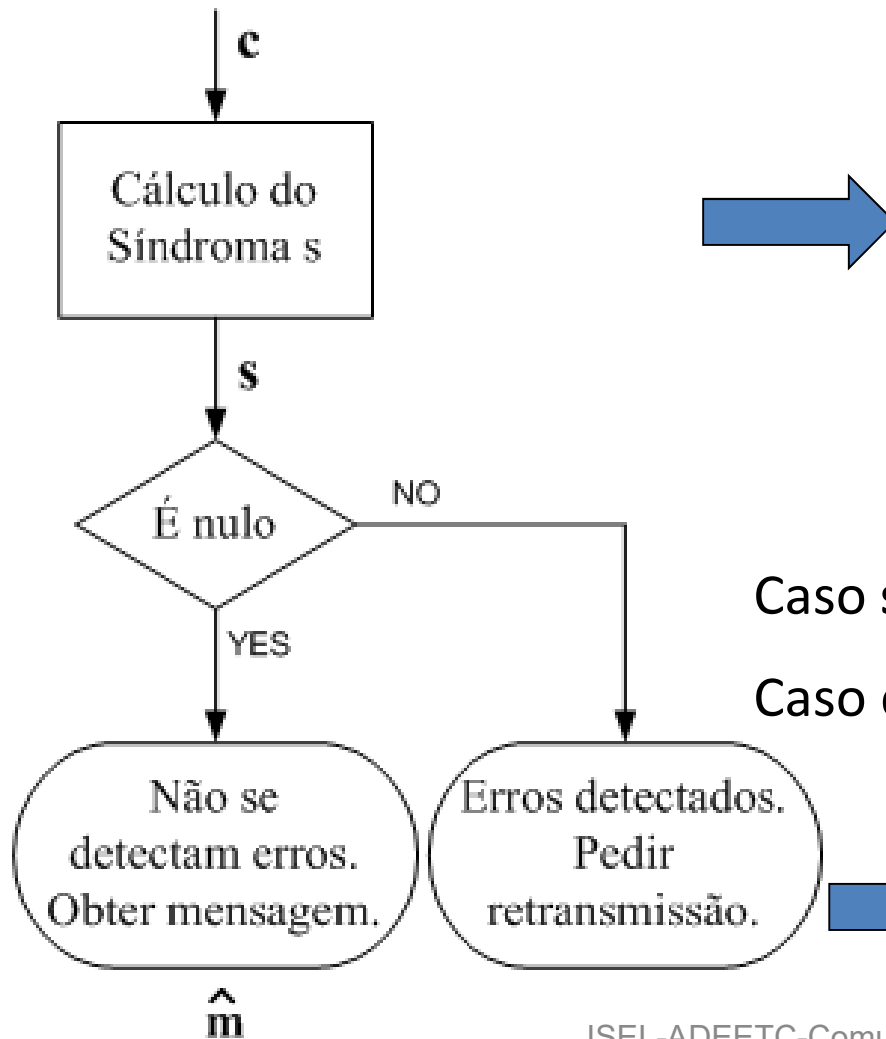


- Funciona num dos modos:
 1. deteção
 2. correção
 3. deteção e correção
- Se a palavra recebida \mathbf{y} não pertence ao código, houve erro(s)



3. Descodificação: deteção

- Processo de descodificação em modo **deteção (ARQ)**



Síndroma

= conjunto de sintomas

= comparação bit a bit entre os bits de paridade transmitidos e recalculados no decodificador

Caso s seja nulo, não se detetam erros

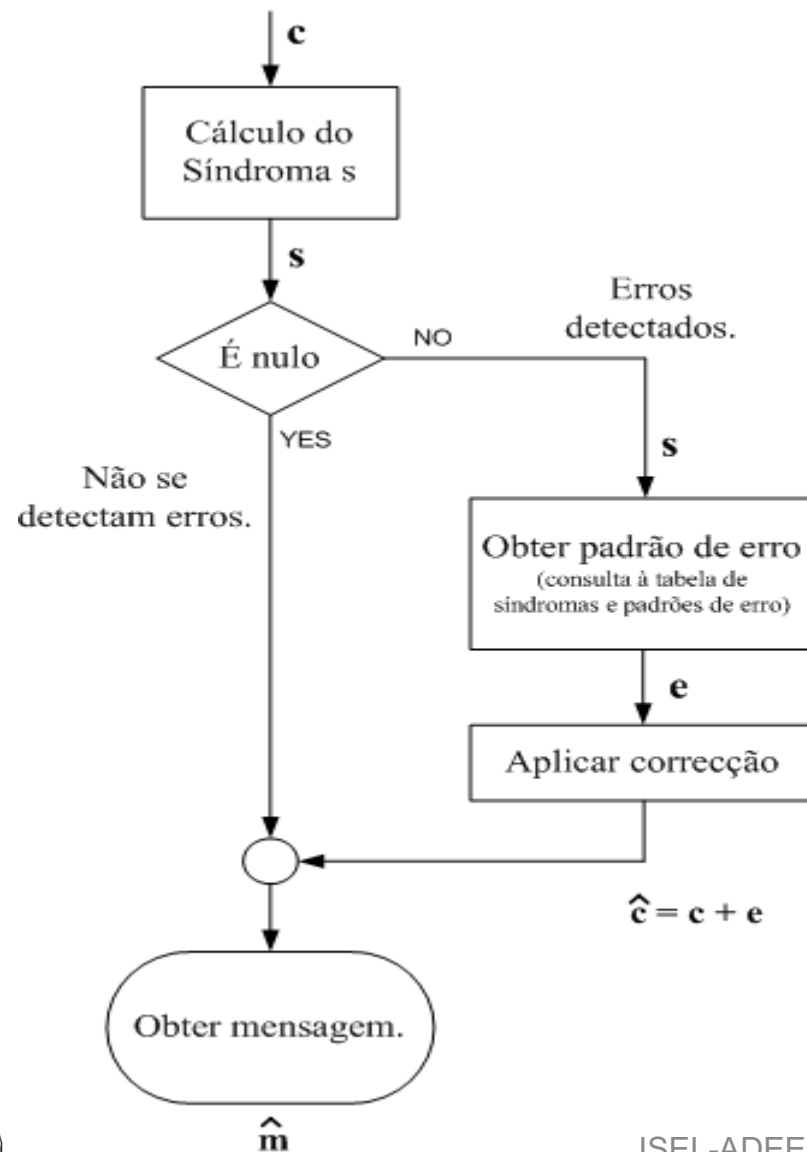
Caso contrário, existem erros detetados

É necessário que o SCD possibilite este pedido de retransmissão



3. Descodificação: correção

- Processo de descodificação em modo **correção (FEC)**



Mecanismo de
correção



3. Descodificação: correção

- Tabela de síndromas para o código Hamming(7,4)
- O código tem $2^3=8$ síndromas: síndrome nulo - ausência de erro; os outros 7 correspondem aos padrões de um bit em erro por palavra

Síndrome	Padrão de Erro	Observações
000	0000000	Ausência de erro
011	1000000	1.º bit em erro
110	0100000	2.º bit em erro
101	0010000	3.º bit em erro
111	0001000	4.º bit em erro
100	0000100	5.º bit em erro
010	0000010	6.º bit em erro
001	0000001	7.º bit em erro



3. Descodificação: correção

- Sejam as palavras de código
 - $c_1 = [1\ 0\ 0\ 0\ 0\ 1\ 1]$
 - $c_2 = [0\ 0\ 1\ 1\ 0\ 1\ 0]$
- Sejam as palavras recebidas no decodificador
 - $y_1 = c_1 + [1\ 0\ 0\ 0\ 0\ 0\ 0] = [\underline{0}\ 0\ 0\ 0\ 0\ 1\ 1]$
 - $y_2 = c_2 + [0\ 0\ 1\ 0\ 0\ 0\ 0] = [0\ 0\ \underline{0}\ 1\ 0\ 1\ 0]$
 - $y_3 = c_1 + [1\ 1\ 0\ 0\ 0\ 0\ 0] = [\underline{0}\ \underline{1}\ 0\ 0\ 0\ 1\ 1]$
- Os síndromas obtidos são
 - $s_1 = [0\ 1\ 1]$
 - $s_2 = [1\ 0\ 1]$
 - $s_3 = [1\ 0\ 1]$

s	e
000	0000000
011	1000000
110	0100000
101	0010000
111	0001000
100	0000100
010	0000010
001	0000001



Dois erros na palavra



3. Descodificação: correção

s	e
000	0000000
011	1000000
110	0100000
101	0010000
111	0001000
100	0000100
010	0000010
001	0000001

- Os padrões de erro associados são

- $e_1 = [1\ 0\ 0\ 0\ 0\ 0\ 0]$

- $e_2 = [0\ 0\ 1\ 0\ 0\ 0\ 0]$

- $e_3 = [0\ 0\ 1\ 0\ 0\ 0\ 0]$

- As palavras estimadas são

- $c_1 = y_1 + e_1 = [\underline{0}\ 0\ 0\ 0\ 0\ 1\ 1] + [1\ 0\ 0\ 0\ 0\ 0\ 0] = [1\ 0\ 0\ 0\ 0\ 1\ 1]$

- $c_2 = y_2 + e_2 = [0\ 0\ \underline{0}\ 1\ 0\ 1\ 0] + [0\ 0\ 1\ 0\ 0\ 0\ 0] = [0\ 0\ 1\ 1\ 0\ 1\ 0]$

- $c_3 = y_3 + e_3 = [\underline{0}\ \underline{1}\ 0\ 0\ 0\ 1\ 1] + [0\ 0\ 1\ 0\ 0\ 0\ 0] = [0\ 1\ 1\ 0\ 0\ 1\ 1]$

- As mensagens obtidas após correção

- $m_1 = [1\ 0\ 0\ 0]$

- $m_2 = [0\ 0\ 1\ 1]$

- $m_3 = [0\ 1\ 1\ 0]$

Os dois erros na palavra implicaram erro após correção (t=1)



3. CRC – Cyclic Redundancy Check

- O decodificador, em modo de deteção calcula o síndrome $s(X)$
- Dado que $c(X)=m(X)g(X)$, tem-se que qualquer palavra de código é fator do polinómio gerador
- Seja $y(X) = c(X) + e(X)$ a palavra recebida, em que $e(X)$ é o padrão de erro

- Caso $e(X)$ seja nulo o síndrome é nulo

$$s(X) = \text{resto} \left[\frac{y(X)}{g(X)} \right] = \text{resto} \left[\frac{c(X)}{g(X)} \right] = \text{resto} \left[\frac{m(X)g(X)}{g(X)} \right] = 0$$

- Caso $e(X)$ seja não nulo o síndrome é não nulo e depende do valor de $e(X)$

$$s(X) = \text{resto} \left[\frac{y(X)}{g(X)} \right] = \text{resto} \left[\frac{m(X)g(X) + e(X)}{g(X)} \right] = \text{resto} \left[\frac{e(X)}{g(X)} \right]$$



3. CRC – Cyclic Redundancy Check

- Na descodificador temos divisão de polinómios $s(X) = \text{resto} \left[\frac{c(X)}{g(X)} \right]$
- Recorrendo ao MATLAB, podemos usar a função *deconv*
- Sejam $c(X) = X^6 + X^3 + X + 1$ $g(X) = X^3 + X^2 + 1$
 $= [1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1]$ $= [1 \ 1 \ 0 \ 1]$

```
>> c = [1 0 0 1 0 1 1];
```

```
>> g = [1 1 0 1];
```

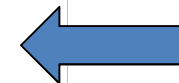
```
>> [q, s] = deconv(c, g);
```

```
>> mod(s,2)
```

```
ans =
```

```
0 0 0 0 0 0 0
```

Síndrome nulo



**Ausência de
erros**



3. CRC – Cyclic Redundancy Check

- Introduzindo 1 erro no penúltimo bit na palavra $c(X)$ temos

$$y(X) = c(X) + e(X) = (X^6 + X^3 + X + 1) + (X)$$

$$= X^6 + X^3 + 1$$

$$= [1\ 0\ 0\ 1\ 0\ 0\ 1]$$

$$g(X) = X^3 + X^2 + 1$$
$$= [1\ 1\ 0\ 1]$$

```
>> c = [1 0 0 1 0 0 1];
```

```
>> g = [1 1 0 1];
```

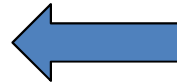
```
>> [q, s] = deconv(c, g);
```

```
>> mod(s,2)
```

```
ans =
```

```
0 0 0 0 0 1 0
```

Síndrome não nulo



Erros detetados



3. CRC - *Cyclic Redundancy Check*

- Tipicamente é utilizado em modo de **deteção** de erros
- Quando a distância mínima do código for maior ou igual a 3, também pode ser usado em modo **correção**
- Tipicamente temos um número reduzido de bits de paridade calculado para elevado número de bits de mensagem
 - $n \gg q > 1$
- O CRC tem elevada capacidade de deteção de erros, especialmente de *burst* de erros (rajada de erros)
- Um *burst* ou rajada de erros define-se como um bloco contíguo de bits recebidos em erro; o primeiro e último *bit* distam B bits entre si, sendo B o comprimento do *burst*



3. CRC - *Cyclic Redundancy Check*

- Elevada capacidade de deteção de erros:
 - todos os *burst* de dimensão q ou menor
 - uma fração dos *burst* de dimensão $q+1$; a fracção é $1-2^{-(q-1)}$
 - uma fração dos *burst* de dimensão superior a $q+1$; a fracção é $1-2^{-q}$
 - todas as combinações de d_{\min} ou menos erros
 - todos os padrões com número ímpar de erros, quando o gerador tem número par de coeficientes não nulos
- Por exemplo, para o código CRC7 com $g(X)=X^7+X^6+X^4+1$ temos
 - todos os *burst* de dimensão 7 ou menor
 - $1-2^{-(q-1)} = 1 - 2^{-(7-1)} = 98,44\%$ dos *burst* de dimensão 8
 - $1-2^{-(q)} = 1 - 2^{-(7)} = 99,22\%$ dos *burst* de dimensão superior a 8
 - todos os padrões com número ímpar de erros



3. Comparação de códigos

Análise comparativa de códigos: ritmo e capacidades de deteção e correção de erros.

Código	$R = k/n$	dmin	Deteta l	Corrige t
Repetição (2,1)	0.500	2	1	0
Repetição (3,1)	0.333	3	2	1
Repetição (4,1)	0.250	4	3	1
Repetição (5,1)	0.200	5	4	2
Paridade (3,2)	0.666	2	1	0
Paridade (8,7)	0.875	2	1	0
Hamming (7,4) m=3	0.571	3	2	1
Hamming (15,11) m=4	0.733	3	2	1
Hamming (31,26) m=5	0.838	3	2	1



4. Exercícios

Tenha em conta os mecanismos de deteção e correção de erros usados nos códigos de bloco (n, k) .

- a) Quais as vantagens e desvantagens da utilização destes códigos? Justifique.
- b) Indique, justificando, quais as técnicas normalmente utilizadas para estabelecer os bits redundantes para proceder à deteção/correção de erros. Exemplifique e relacione o número de bits redundantes com as capacidades de deteção e correção de erros.
- c) Considere que o ficheiro f demorou 5 segundos a ser transmitido, sem a utilização de códigos detetores e corretores de erros. Passando a transmitir o ficheiro f , no mesmo sistema, usando um código $(8, 4)$, quanto tempo demorará essa transmissão?



4. Exercícios

Solução

- a) Vantagens: controlo de erros, diminuição de BER, aumento da qualidade de serviço (QoS).

Desvantagens: maior complexidade, mais tempo necessário para a transmissão (e retransmissão, quando necessário) e correção.

- b) Técnica de repetição e técnica de bits de paridade (XOR)

Exemplo de repetição: mensagem=010 -> palavra de código=000 111 000

Exemplo de paridade par: mensagem=0110 -> palavra de código=011 101

As capacidades de deteção e correção de erros são diretamente proporcionais ao número de bits redundantes.

- c) Demorará o dobro do tempo, 10 segundos (no melhor caso).



4. Exercícios

Considere o código de controlo de erros cujas palavras estão organizadas na forma $c = [m_0 \ m_1 \ b_0 \ b_1]$, tais que $b_0 = m_0 \oplus m_1$ e $b_1 = m_1$.

- a) Apresente todas as palavras de código.
- b) Calcule a distância mínima de Hamming.
- c) Calcule as capacidades de deteção e correção de erros.
- d) Suponha que se transmite a mensagem 01 e que sobre a palavra de código resultante é aplicado o padrão de erro 1010. Qual a mensagem decodificada? Comente.



4. Exercícios

Solução.

a) Código (4,2), com 4 palavras de código

0 0 0 0

0 1 1 1

1 0 1 0

1 1 0 1

b) $d_{\min}=2$

c) detecção $l=1$ bits por bloco, correção $t=0$ bits (não tem).

d) $m=01 \rightarrow c=0111 \rightarrow y = c + e = 0111 + 1010 = 1101$. A mensagem decodificada é 11 (os dois primeiros bits do bloco). Os dois erros introduzidos na transmissão não foram detetados.



4. Exercícios

Assuma uma transmissão digital com código Hamming (7,4), cujas palavras estão organizadas na forma $c = [m_0 \ m_1 \ m_2 \ m_3 \ b_0 \ b_1 \ b_2]$, com equações de paridade

$$b_0 = m_1 \oplus m_2 \oplus m_3 \quad b_1 = m_0 \oplus m_1 \oplus m_3 \quad b_2 = m_0 \oplus m_2 \oplus m_3$$

- a) Sabendo que o número de bits a transmitir antes da aplicação do código é 40000, qual o número de bits a transmitir após a aplicação do código?
- b) Qual a sequência transmitida quando se enviam os bits de informação 10100011?
- c) Caso seja recebida a sequência 1010001, existem erros nesta sequência?



4. Exercícios

Solução

- a) São transmitidos $40000 + 30000 = 70000$ bits, no total.
- b) A sequência transmitida é 1010 110 0011 010.
- c) A palavra 1010 001 não pertence ao código. Logo, existem erros detetados nesta sequência.



4. Exercícios

Considere o código de bloco linear com palavras definidas por $c = [m_0 \ m_1 \ m_2 \ b_0 \ b_1 \ b_2 \ b_3]$, em que $b_0 = m_0 \oplus m_1$, $b_1 = m_2$, $b_2 = m_1 \oplus m_2$ e $b_3 = m_0 \oplus m_2$.

- a) Indique as dimensões (n,k) .
- b) Qual a distância mínima do código e as respectivas capacidades de deteção e correção de erros?
- c) Exemplifique uma deteção de erros.



4. Exercícios

Solução

- a) $(n,k) = (7,3)$.
- b) Listando as 8 palavras de código, conclui-se que a palavra de código com menor peso de Hamming tem peso igual a 3. Logo $d_{\min}=3$, $l=2$ e $t=1$.
- c) Por exemplo, se a palavra de código 0010111 sofrer um erro no último bit, temos que a palavra recebida é 0010110. Esta palavra não pertence à lista de palavras de código, logo temos a presença de erro detetada.

Em alternativa, assumindo que os bits de mensagem recebidos 001 estão corretos, e se recalcularmos os bits de paridade teremos 0111, o que difere da configuração recebida 0110, detetando-se assim o erro



4. Exercícios

Considere o polinómio gerador $g(X) = X^4 + X^3 + X^2 + X + 1$ de código (10,6).

- a) Apresente a palavra de código $c(X)$, quando a mensagem é 1 0 0 0 0 1.
 - b) A palavra 1111111111 pertence ao código?
-

Considere o polinómio gerador $g(X) = X^3 + X + 1$ do código (7,4).

- a) Quais das palavras 0000000, 1011000 e 0000011 pertencem ao código?
- b) Apresente todas as palavras de código.



4. Exercícios

Solução

a) $c(X) = [1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$.

b) Sim. A palavra 1111111111 pertence ao código. Se dividirmos esta palavra pelo polinómio gerador, temos resto nulo. Isto significa que a palavra pertence ao código.

a) As palavras 0000000 e 1011000 pertencem ao código. Se dividirmos estas palavras pelo polinómio gerador, temos resto nulo. Isto significa que estas palavras pertencem ao código. Para a palavra 0000011 não se obtém resto nulo, pelo que não pertence ao código.



4. Exercícios

Solução (continuação)

b) As 16 palavras de código.

0	0	0	0	0	0	0	1	0	0	0	1	0	1
0	0	0	1	0	1	1	1	0	0	1	1	1	0
0	0	1	0	1	1	0	1	0	1	0	0	1	1
0	0	1	1	1	0	1	1	0	1	1	0	0	0
0	1	0	0	1	1	1	1	1	0	0	0	1	0
0	1	0	1	1	0	0	1	1	0	1	0	0	1
0	1	1	0	0	0	1	1	1	1	0	1	0	0
0	1	1	1	0	1	0	1	1	1	1	1	1	1



4. Exercícios

Suponha uma transmissão digital em que são enviados os bits de informação 1010. Considere que o controlo de erros é realizado através de CRC com polinómio gerador $g(X) = X^3 + X + 1$.

- a) Apresente a sequência binária transmitida.
 - b) Provoque um erro nesta sequência binária e ilustre o funcionamento da deteção de erros.
-

Quanto tempo demora a transmissão de um ficheiro com 1 024 000 bytes, considerando a utilização de modulação 16-QAM, com tempo de símbolo $T_s = 10 \mu s$, nos cenários:

- a) Ausência de códigos detetores e corretores de erros?
- b) Deteção de erros com código CRC7, estabelecido por $g(X) = X^7 + X^6 + X^4 + 1$, aplicado a blocos de mensagem com dimensão 1024 bits?



4. Exercícios

Solução

a) 1010011

b) 1010001, é a sequência anterior com erro no penúltimo bit. Se dividirmos esta sequência pelo polinómio gerador obtemos o resto 010. Dado que o resto não é nulo, o decodificador/recetor deteta o erro.

Por outro lado, o resto da divisão de 1010011 pelo polinómio gerador, é nulo.

a) Demora 20,48 segundos.

b) Demora 20,62 segundos.



4. Exercícios

- Exercícios sugeridos (de enunciados de testes de semestres anteriores):
 - Exercício #4, alínea iv), do segundo teste parcial, verão 2016/2017, 6 de junho de 2017
 - Exercício #8, alínea iv), do teste de época normal, verão 2016/2017, 5 de julho de 2017



4. Exercícios

- Exercícios sugeridos (de enunciados de testes de semestres anteriores):
 - Exercício #8, alínea iv), do teste de época normal, inverno 2016/2017, 30 de janeiro de 2017
 - Exercício #6, alínea iv), do teste de época de recurso, inverno 2016/2017, 16 de fevereiro de 2017



4. Exercícios

- Exercícios sugeridos (de enunciados de testes de semestres anteriores):
 - Exercício #4, alínea iii), do 2.º teste parcial, verão 2015/2016, 15 de junho de 2016
 - Exercício #10, alínea iii), do teste de época normal, verão 2015/2016, 30 de junho de 2016
 - Exercício #6, alínea ii), do teste de época de recurso, verão 2015/2016, 19 de julho de 2016
 - Exercício #4, alíneas i), ii), iii), do segundo teste parcial, verão 2014/2015, 18 de junho de 2015



4. Exercícios

- Exercícios sugeridos (de enunciados de testes de semestres anteriores):
 - Exercício #8, alínea ii), do teste de época normal, verão 2014/2015, 9 de julho de 2015
 - Exercício #6, alínea ii), do teste de época de recurso, verão 2014/2015, 23 de julho de 2015
 - Exercício #5, alíneas ii) e iii), do segundo teste parcial, inverno 2014/2015, 14 de janeiro de 2015
 - Exercício #6, alínea ii), do teste de época de recurso, inverno 2014/2015, 20 de fevereiro de 2015

