Curso: Engenharia de Computação

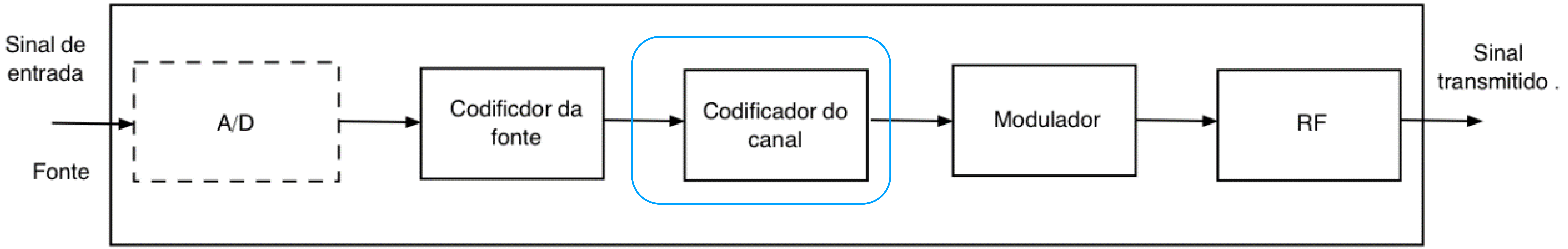
Sistemas de Comunicações Móveis

Prof. Clayton J A Silva, MSc
clayton.silva@professores.ibmec.edu.br

ibmec

Códigos Corretores de Erros

Transmissor



Transmissor

Codificadores de erros

- A transmissão confiável de dados a **altas taxas** tem representado um desafio cada vez maior aos engenheiros
- Os **códigos corretores** têm contribuído de modo significativo para o avanço na área
- Frequentemente, no contexto de comunicações digitais, ocorrem erros de detecção acarretados por problemas de transmissão, implicando a necessidade de correção de erros
- A solução está na **teoria da codificação**

Teorema da codificação

Para qualquer canal cuja entrada é um **alfabeto discreto**, existem códigos com **taxa de informação** *R*, com **palavras** de comprimento de ***n* dígitos**, para os quais a **probabilidade de erro (*Pe*)**, com decodificação por máxima verossimilhança é limitada a

$$Pe < e^{1-nE(R)}$$

, em que ***E(R)>0, 0≤R<C***, é uma função do comportamento estatístico do canal e C é a capacidade do canal.

Teoria da codificação

- Existem códigos que podem tornar a probabilidade de erro na decodificação muito pequena
- A probabilidade decresce exponencialmente quando *n* é aumentado, no entanto o aumento da complexidade do sistema também ocorre
- Objetivos da teoria da codificação:

1. Encontrar **códigos longos** e eficientes
2. Encontrar **métodos práticos** de codificação/decodificação eficientes

Teoria da codificação

- Tipos de erros:

1. Erros **aleatórios**, provocados pelo ruído, como discutimos
2. Erros **em surto**, em caso contrário

Teoria da codificação

- Tipos de códigos:

1. Códigos **lineares** e **não lineares**, em que os **dígitos redundantes** são calculados como combinações lineares ou não lineares dos **dígitos da informação**.

2. Códigos de **bloco** e **convolucionais**. A **redundância** é colocada em um **bloco de dígitos** que verifica a ocorrência ou não de erros. Quando são usados mais blocos configuram-se os códigos convolucionais.

Códigos de blocos *(n, k, d)*

- Processo de codificação

1. **Segmenta-se a mensagem em blocos de *k* dígitos**
2. Acrescenta-se a cada bloco $(\boldsymbol{n}-\boldsymbol{k})$ **dígitos redundantes**
3. Os dígitos redundantes são definidos a partir dos *k* dígitos da mensagens e destinam-se à **detecção simples** ou **detecção e correção** de erros
4. A **eficiência do código** é definida pela razão entre o número de bits de informação e o número total de bits, n/k

Códigos de blocos *(n, k, d)*

- **Distância de *Hamming*** é o **número de posições em que duas palavras de um código diferem**.

Por exemplo, a distância entre as palavras **0**101 010**1** e **1**101 0100 é igual a 2.

- A **distância mínima (*d*) de um código** é a **menor distância de *Hamming*** encontrada entre suas palavras
- Para obter a distância de *Hamming* de um código basta aplicar a operação lógica **ou exclusivo** entre todas as palavras do código.

Códigos de blocos *(n, k, d)*

- Um código (n, k, d) é definido como sendo um conjunto de 2^k *n–uplas* chamadas de **palavras código**, que
  - diferem entre si pelo menos *d* posições,
  - possuem *n–k* bits de redundância e
  - possuem *K* bits de informação

Exemplo
código (7,3,4) de Hamming

k1	k2	k3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$c_1 = k1 \otimes k2$

$c_2 = k2 \otimes k3$

$c_3 = k1 \otimes k3$

$c_4 = k1 \otimes k2 \otimes k3$

k *n-k*

k1	k2	k3	c1	c2	c3	c4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	1	0	1	1
1	0	1	1	1	0	0
1	1	0	0	1	1	0
1	1	1	0	0	0	1

n

Obs. Posteriormente alteraremos a indicação da posição dos bits, na generalização dos códigos de *Hamming*

Códigos de blocos *(n, k, d)*

- **Detecção de erros:** em um código com distância mínima *d*, o menor número de mudanças necessárias para converter uma palavra do código em outra é pelo menos *d*, logo

  é **possível detectar** $\boldsymbol{d-1}$ erros

- **Correção de erros:** após detectar que há erro, é necessário decidir qual é a palavra do código mais provável. Supondo palavras com mesma probabilidade, decide-se pela palavra mais próxima da *n-upla* recebida. **Se houver até *t* erros**,

  decide-se acertadamente se $\boldsymbol{2t+1 \leq d}$

Códigos simples
Códigos de repetição

- Em códigos de repetição, os parâmetros são:

$$k=1,\ c\geq 1 \text{ e } n=k+c=1+c$$

- Como $k=1$, o código tem apenas duas palavras, uma delas é uma sequência de n 0s; a outra uma sequência de n 1s.
- Os dígitos de paridade são todos uma repetição do dígito de informação c vezes.

Códigos simples
Códigos de repetição

- Uma regra simples de decodificação é decidir pelo bit que aparece mais vezes na recepção.
- A distância de *Hamming* do código é $d=n$ e a eficiência é $1/n$.

Códigos simples
Códigos com 1 bit de paridade

- Em códigos com 1 bit de paridade, os parâmetros são:

$$k \geq 1,\ c=1 \text{ e } n=k+c=k+1$$

- Como $c=1$, o código tem apenas um bit redundante, que é definido na transmissão para tornar o número de bits 1 par (ou impar).

Códigos simples

Códigos com 1 bit de paridade

- A regra de decodificação é contar o número de bits recebidos. Se a paridade não estiver correta significa que houve um erro na transmissão.
- A distância de *Hamming* do código é $d=2$ e a eficiência é $k/(k+1)$.
- O código permite a detecção de 1 erro, mas não corrige. Pode ser eficaz quando se dispõe de canal de retorno para retransmissão da palavra.

ibmec.br

Códigos simples
Códigos de *Hamming*

- Em códigos de *Hamming*, os parâmetros são:

$$n \leq 2^c - 1$$

- No algoritmo de *Hamming*,

1. Os bits de redundância e os bits de informação são organizados em uma **sequência a partir de 1**...

$$b_1\ b_2\ b_3\ b_4\ \dots\ b_n$$

2. ..., de modo que os **bits de redundância** ocuparão as **posições** que são **potência de 2**, logo

$$c_1\ c_2\ k_3\ c_4\ \dots$$

ibmec

Códigos simples

Códigos de *Hamming*

- No algoritmo de *Hamming*,

3. **Cada bit** de redundância (bit de paridade) ajusta a **paridade par** (ou impar) dos bits **específicos** de informação...

4. Em geral, cada bit de informação k_j, ou seja, o bit de informação que ocupa a posição j, é verificado pelos bits de redundância cuja soma das suas posições seja j.

Por exemplo, o bit k_3 é verificado pelos bits c_1 e c_2. O bit k_5 é verificado pelos bits c_1 e c_4. O bit k_6 é verificado pelos bits c_2 e c_4.

ibmec.br

Códigos simples
Códigos de *Hamming*

- No algoritmo de *Hamming*,

5. Assim, cada bit de paridade será definido pelo codificador como 0 ou 1 de modo que o número de bits verificados seja par (ou impar)
6. No receptor, a determinação do bit incorreto pode ser obtida **pela soma das posições de todos os bits de paridade incorretos**

Por exemplo, se na recepção os bits c_1 e c_2 estão incorretos, significa que o erro identificado foi no bit k_3.

ibmec

Códigos simples
Códigos de *Hamming*

- Nos códigos de *Hamming*, o número de bits de paridade pode ser dado pela quantidade de divisões inteiras sucessivas de n por 2 até quociente igual a 0.

  Por exemplo, se $n=7$ bits, 7 div 2 = 3, 3 div 2 = 1, 1 div 2 = 0, logo $n-k=3$

Outros códigos

- **Códigos cíclicos:** quando uma permutação de uma das palavras do código permite obter outra palavra do mesmo código...

  Por exemplo, P(*0001*) = *1000*
- Possuem propriedades matemáticas que simplificam a implementação de codificadores e decodificadores.

Outros códigos

- **Códigos convolucionais**: oferecem um enfoque para o controle de erros muito diferente dos códigos de blocos.
- Converte uma sequência inteira (não importa o comprimento) em uma sequência de dados.
- Admite uma **memória** de k bits, logo, para a memória, 2^k combinações possíveis. Cada uma das combinações de memória pode ser representada por um **estado.**
- Cada novo bit a ser transmitido produz um **deslocamento à direita** da palavra da memória, que corresponde a uma **transição de estado**

Códigos convolucionais

- Por exemplo, seja um codificador com memória $k=3$, em um determinado instante (estado) os bits podem ser XYZ, dados por

Estado	XYZ
S_0	000
S_1	001
S_2	010
S_3	011
S_4	100
S_5	101
S_6	110
S_7	111

IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC