Códigos de Detecção e Correcção de Erros

Sistemas Distribuídos e Tolerância a Falhas

Manuela Rodrigues M1379

Detecção e Correcção de Erros

- Características dos Erros
- Possíveis Abordagens
- Códigos de Detecção de Erros:
  - Paridade, Checksum, CRC (Cyclic Redundancy Check)
- Códigos de Detecção e Correcção de Erros
  - Correção de erros isolados: Código de Hamming
  - Correção de erros em "rajada" (burst): BCH, Reed-Solomon, Reed-Muller, Golay

Erros (1)

- Um sistema de computação funciona em função da transferência de informação desde o nível de circuito integrados até aos níveis mais altos, como por exemplo gravação no disco ou comunicação entre computadores.

- Está sujeito a diversos erros, como os causados por interferências electromagnéticas, envelhecimento de componentes, curto-circuitos, ...

Erros (2)

- **Características dos erros**
  1. São inevitáveis em qualquer sistema de comunicação real;
  2. A distribuição dos erros não é homogénea: bits isolados ou em "rajadas" (bursts) de erros, com 8 ou mais bits sucessivos errados;
  3. Deve-se levar em conta o meio físico de transmissão de dados, para incluir maior ou menor redundância na transmissão, a fim de garantir que a informação recebida seja confiável.

Erros (3)

- **Possíveis abordagens no tratamento de erros:**

1. Ignorar o erro;
2. Eco (transmissão à origem de reflexos dos dados recebidos);
3. Sinalizar o erro;
4. Detectar e solicitar a retransmissão em caso de erro;
5. Detectar e corrigir os erros na recepção de forma automática.

Códigos de Detecção de Erros

- **Códigos de Detecção de Erros**
  - Detectar um erro é uma tarefa mais simples do que detectar e corrigir;
  - Nem sempre é possível solicitar uma retransmissão;
  - Todos os métodos utilizam a inserção de bits extras;

    (Esses bits podem ser obtidos a partir da informação original e o receptor recalcula os bits extras)
  - Um método ineficiente mas muito utilizado para detectar erros é a Paridade;
  - Um método mais eficiente é o uso de um código polinomial ou CRC (Cyclic Redundancy Check);

Detecção de Erros – Paridade (1)

- **Paridade**
  - Consiste basicamente no acto do transmissor adicionar um bit de redundância após um determinado número de bits (normalmente um byte):
    - nº par de 1's → paridade par
    - nº impar de 1's → paridade impar
  - 000, 011, 101, 110 → são mensagens transmitidas sem erro, tendo em conta que o último bit é o de paridade

Detecção de Erros – Paridade (2)

- **Exemplo1:**
  - O caracter A no código ASCII é representado por 1000001
  - O bit P de paridade é calculado e transmitido:
    - 1000001P → nº par de 1's → P = 0 (paridade par), logo transmite-se 10000010
  - O receptor calcula a paridade da mensagem e compara-a com o bit P recebido: P = paridade → transmissão correcta

Detecção de Erros – Paridade (4)

- Este processo pode ser vulnerável se houver mais do que um erro, permitindo assim que este passe até ao destino sem ser identificado.

  Exemplo: 11010010 – devolve valor 0 mas existe erro

- 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: Bus PCI e SCSI.

Detecção de Erros – Checksum (1)

- **Checksum**
  - Consiste na transmissão de todas as palavras juntamente com o resultado da sua soma binária.
    - Inclui o bit de transporte.
    - Inversão do valor dos bits do checksum.

A	B	Soma	Transporte
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Detecção de Erros – Checksum (2)

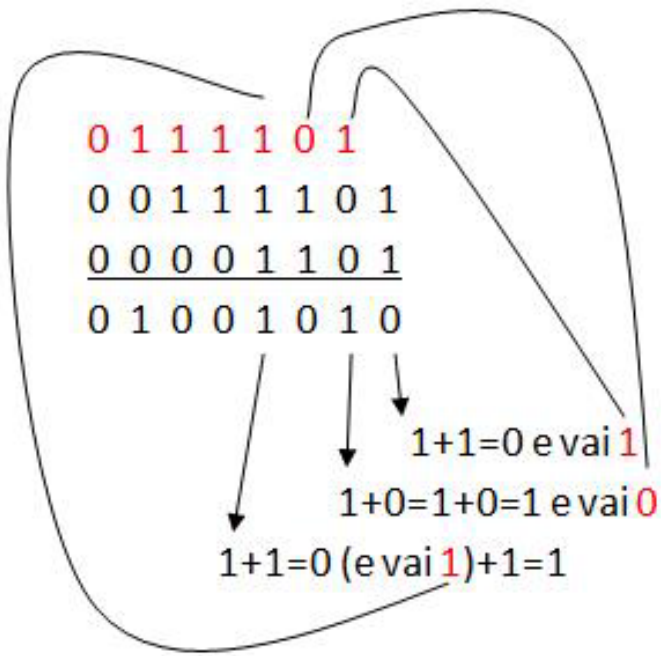
- **Exemplo:**

  checksum de 2 palavras de 8 bits

  - Dados iniciais: 00111101 00001101
  - Checksum é:

    01001010
  - Checksum invertido:

    10110101



Detecção de Erros – Checksum (3)

- Dados enviados:
  - 00111101
  - 00001101
  - 10110101 (checksum – invertido)
- No receptor, as palavras são novamente somadas e comparadas com o checksum enviado:
- Se qualquer um dos dados transmitidos, incluindo o checksum, sofrerem algum erro então, a soma do novo checksum com o checksum enviado, será diferente de 1.

  0 1 0 0 1 0 1 0 → novo checksum (das palavras iniciais recebidas)

  1 0 1 1 0 1 0 1 → checksum enviado

  1 1 1 1 1 1 1 1 → sem erro

Detecção de Erros – Checksum (3)

- **Exemplo com erro:**

0 0 1 1 0 0 0 1

0 0 0 0 1 1 0 1

0 0 1 1 0 1 1 0 → novo checksum

1 0 1 1 0 1 0 1 → checksum enviado

1 1 1 0 1 0 1 1 ≠ 1 1 1 1 1 1 1 1

→ valor recebido incorrectamente, com erro no 3º ou 5º bit (de qualquer uma das palavras enviadas, incluindo o checksum)

Detecção de Erros – CRC (1)

- **CRC (Cyclic Redundancy Check)**
  - Esquema mais eficiente
  - Emissor/receptor concordam num **polinómio gerador** *G(x)*, em que quanto maior for o seu grau maior será a capacidade de detecção de erros
  - Neste polinómio tanto o bit de maior ordem quanto o de menor ordem devem ser iguais a 1
  - Palavra inicial de ***k*** bits é representado por um polinómio de X de ordem ***k-1***
    - palavra inicial = 10110001
    - polinómio = $X^7+X^5+x^4+1$

Detecção de Erros – CRC (2)

- Execução: o polinómio p(x) é representado pela palavra inicial somada aos bits de paridade e deve ser divisível por G(x);
- O receptor tenta dividir p(x) por G(x). Se houver resto ≠ 0, houve um erro de transmissão;
- Se houver um erro, em vez de se receber o polinómio T(x), recebe-se T(x)+E(x);
- Cada bit 1 em E(x) corresponde a um bit invertido;
- T(x)/G(x) é sempre zero, logo o resultado é E(x)/G(x).

Detecção de Erros – CRC (3)

- **Exemplo:**
  - Mensagem a transmitir:**10111011**
  - Polinómio gerador ***G(X)*** **= x^4+x+1 → 10011**
  - Acrescenta-se à mensagem inicial, a quantidade de zeros equivalentes ao grau de G(x), ficando:

    10111011 0000
  - Seguidamente divide-se a mensagem (ponto anterior) pelo polinómio gerador
    - A divisão de dois polinómios (na sua forma binária) é feita recorrendo à operação XOR (⊕)

Detecção de Erros – CRC (4)

1 0 1 1 1 0 1 1 0 0 0 0
1 0 0 1 1
0 0 1 0 0 0 1
  1 0 0 1 1
  0 0 0 1 0 1 0 0
    1 0 0 1 1
    0 0 1 1 1 0 0
      1 0 0 1 1
      0 1 1 1 1 ← Resto da Divisão

- O resto que da divisão é finalmente adicionado à mensagem original, pelo que a mensagem transmitida será:

101110111111

Detecção de Erros – CRC (5)

- Para descodificar a mensagem, o procedimento deve ser repetido.

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

  1 0 0 1 1

  0 0 1 0 0 0 0

  1 0 0 1 1

  0 0 0 1 1 1 1 1

  1 0 0 1 1

  0 1 1 0 0 1

  1 0 0 1 1

  0 1 0 1 0 1

  1 0 0 1 1

  0 0 1 1 0 ← ≠ 0, logo a mensagem foi recebida com erro

Códigos de Detecção e Correcção de Erros

- **Códigos de Correcção de Erros**
  - Códigos de correcção podem recuperar o dado original a partir do código com erros;
  - Consistem na criação de códigos extras que detectam situações inválidas mas que mantém a identidade do dado original;
  - O conceito mais básico e mais importante desses códigos é a distância de Hamming, utilizada para a criação de códigos de correcção.

Detecção e Correcção de Erros - Hamming (1)

- **Erros isolados**
- **Códigos de Hamming - Codificação:**
  1. Os bits da palavra de código são numerados a partir da esquerda, começando por 1;
  2. É acrescentada informação redundante em posições pré-definidas, ou seja, os bits que são potência de 2 vão ser bits de controlo (2^n);
  3. Os restantes são preenchidos com k bits de dados conhecidos, isto é, com a mensagem a transmitir;

Men

	1=2^0	**2=2^1**	**3**	**4=2^2**	**5**	**6**	**7**
1º	1	0	0	1			
2º	?	?		?			
3º	?	?	1	?	0	0	1

passo

Detecção e Correcção de Erros - Hamming (2)

4. Cálculo dos bits de controlo, isto é, conversão para binário das posições $\neq 2^n$ e valor=1;

   3 – 011

   7 – 111

5. Aplicação do OR Exculsivo (XOR - $\oplus$) aos valores obtido no ponto anterior

   0 1 1

   1 1 1

   1 0 0

   3^a 2^a 1^a

Detecção e Correcção de Erros - Hamming (3)

6. Inserção dos valores obtidos nas respectivas posições do bits de paridade (ponto 2.)

	$1=2^0$	**$2=2^1$**	**3**	**$4=2^2$**	**5**	**6**	**7**
1º passo	1	0	1	0			
2º passo	?	?		?			
3º passo	?	?	1	?	0	0	1
6º passo	0	0	1	1	0	0	1

7. Mensagem a transmitir: 0011001

Detecção e Correcção de Erros - Hamming (4)

- **Códigos de Hamming - Descodificação:**

1. Conversão para binário das posições = 1;
2. Aplicação do OR Exculsivo (XOR - $\oplus$) aos valores obtido no ponto anterior:
   - Se o resultado for = 0, não houve erros na transmissão
   - Se for $\neq$ 0, o resultado obtido convertido para decimal é igual à posição do erro

Detecção e Correcção de Erros - Hamming (5)

- Sem erro: 0 0 1 1 0 0 1

3ª 4ª 7ª

3 - 011
4 - 100
7 - 111

011
⊕ 100
111
⊕ 111
000

- Com erro: 0 0 0 1 0 0 1

4ª 7ª

4 - 100
7 - 111

100
⊕ 111
011 → Erro na posição 3

⇒ Código Correcto: 0011001

Detecção e Correcção de Erros – Burst errors

- **Erros em rajada – Burst Errors**

  - Conjunto de símbolos, recebidos sobre um canal de transmissão de dados, em que o primeiro e último símbolo são erros, e entre eles existe uma sequência de símbolos correctamente recebidos.
  - O tamanho da "rajada" (burst) de erros é definida pelo nº de bits, desde o 1º erro até ao último (inclusive).

    O vector de erros B = (0 0 0 0 1 0 1 1 0 1 0 0 0 0 0) tem tamanho 6

Detecção e Correcção de Erros - BCH

- **Códigos BCH (Bose - Chaudhuri - Hocquenghem):**
  - Um código BCH é um código polinomial, cíclico, detector e corrector de erros de tamanho variável.
  - Estes códigos são a generalização dos códigos de Hamming, ou seja, $t \geq 1$
  - Aplicações: Telefone VoIP, Modems Digitais
  - Referências:
    - Farrel, Patrick Guy – Essentials of Error – Control Coding
    - Purser, Michael – Introduction to Error Correcting Codes
    - Wallace, Hank – Error Detection and Correction Using the BCH Code

Detecção e Correcção de Erros– Reed-Solomon

- **Códigos Reed-Solomon:**
  - Os Códigos Reed-Solomon constituem uma sub-classe dos Códigos BCH.
  - Aplicações: Gravação de CD's e DVD's, Modems de alta velocidade (ADSL), Televisão digital (DVB)
  - Referências:
    - Farrel, Patrick Guy – Essentials of Error – Control Coding
    - Purser, Michael – Introduction to Error Correcting Codes

Detecção e Correcção de Erros– Reed-Muller

- **Códigos Reed-Muller:**
  - Os códigos Reed-Muller fazem parte dos códigos de correcção de erros mais antigos.
  - Aplicações: Inventados em 1954 e utilizados em 1972 pelo Mariner 9 para transmitir fotografias, a preto e branco, de Marte.
  - Referências:
    - http://www-math.mit.edu/phase2/UJM/vol1/COOKE7FF.PDF
    - http://ocw.usu.edu/Electrical_and_Computer_Engineering/Error_Control_Coding/lecture9.pdf

Detecção e Correcção de Erros– Golay

- **Códigos Golay:**
  - O código Golay é um código perfeito linear de correcção de erros.
  - Aplicações: Voyager 1 (1979) e 2 (1980), que precisavam de transmitir centenas de fotografias a cores de Júpiter e Saturno, com telecomunicações de largura de banda muito restritas.
  - Referências:
    - http://www.quadibloc.com/crypto/mi0602.htm
    - http://mathworld.wolfram.com/GolayCode.html