Brazilian Journal of Development

Sinais de dados em telecomunicações: Detecção e correção de erros

Data signs in telecommunications: Error detection and correction

DOI:10.34117/bjdv6n11-481

Recebimento dos originais: 23/10/2020
Aceitação para publicação: 23/11/2020

Andréa Cristina Marques de Araújo
Doutoranda em Ciência da Informação - Universidade Fernando Pessoa (Portugal)
Mestre em Ciência da Computação – UFSC
Instituição: Centro Universitário do Estado do Pará CESUPA
Endereço: Av. Gov. José Malcher n.1963 CEP: 66060-232 Belém-PA
E-mail: andreacristinamaraujo@gmail.com

Jacqueline de Fátima Teixeira
Mestre em Ciência da Computação – UFSC
Rua Rosemeri Silva Belto, 319 – Neves CEP 84020-423 Ponta Grossa-PR
Email jackogut@hotmail.com

Guilherme Teixeira Santos
Graduando de Física – Bacharelado
Universidade Estadual de Ponta Grossa - PR
Campus Uvaranas – Av Gal Carlos Cavalcante, 4748 – Ponta Grossa PR CEP 84030-900
Email g.teixeira3399@gmail.com

RESUMO
Qualquer nível de um sistema de transmissão de informação está sujeito a erros, os quais podem ser causados por exemplo, por interferências eletromagnéticas externas. Devido a estes sinais, os dados podem ser recebidos de maneira incorreta, podem ser perdidos ou podem ter seu significado alterado, resultando em informações sem significado. O presente trabalho apresenta portanto, alguns códigos de correção e detecção de erros utilizados para solucionar, na medida do possível, esses tipos de problemas, mantendo assim o sistema funcionando corretamente.

Palavras-chave: Sinais de dados, Distância de Hamming, Códigos de detecção e de correção de erros.

ABSTRACT
Any level of an information transmission system is subject to errors, which can be caused, for example, by external electromagnetic interference. Due to these signals, the data may be received incorrectly, may be lost or may have its meaning changed, resulting in meaningless information. The present work therefore presents some codes of correction and error detection used to solve, as far as possible, these types of problems, thus keeping the system working correctly.

Keywords: Data signals, Hamming distance, error detection and correction codes.

Brazilian Journal of Development

1 INTRODUÇÃO

Um sistema de computação trabalha em função da transferência de informações (dados ou sinais de controle). Essa transferência existe desde o nível de circuito integrado (entre registradores), até os níveis mais altos, como por exemplo gravação em disco ou comunicação entre computadores (MONTEIRO,1995).

Em todo sistema de transmissão de informação há sempre a possibilidade de ocorrerem deformações, ou até mesmo destruição de parte ou de toda a informação transmitida.

No caso de transmissão de informação entre a memória principal e o processador central (CPU), o meio de transmissão suscetível a erros é o barramento de dados, e no caso de sistema de transmissão de sinais à distância (telecomunicação), apesar dos meios de transmissão serem os mais variados possíveis, a possibilidade de erros ocorre da mesma maneira.

Essas interferências no meio de transmissão (geralmente ruídos) que podem alterar o valor de um ou mais bits ou até mesmo destruí-los, podem ser solucionadas por meio de códigos de correção e detecção de erros, possibilitando o funcionamento do sistema de modo correto.

Um conceito importante para determinar códigos de correção e detecção de erros é o da Distância de Hamming, a qual é determinada pelo número de bits diferentes em duas palavras. Assim, em um subconjunto S que contenha palavras de código com distância 2, uma transição para distância 1 pode representar uma palavra de código com erro de um bit.

2 DISTÂNCIA DE HAMMING

Os erros de transmissão, são definidos por Tanembaum (1990, p.34) como “a quantidade de bits que diferem entre uma sequência de bits enviada e uma sequência de bits recebida”. Os chamados códigos de correção e detecção de erros baseiam-se então, na adição de bits extras a cada sequência de bits enviada, possibilitando que haja a checagem da sequência recebida, e assim a extração da sequência original enviada.

O número de posições que uma sequência de bits enviada difere de uma sequência de bits recebida é então chamado de distância Hamming. Isso significa que, se duas palavras (sequências de bits) estão a uma distância Hamming d, serão necessários d erros simples para converter uma na outra. Considere as sequências 00001101 e 01011101. Pela definição acima, a distância Hamming entre as duas palavras é 2, pois há dois bits que fazem com que uma seja diferente da outra.

As propriedades de detecção e correção de erros de um determinado código dependem da distância Hamming definida em sua elaboração. Segundo Tanembaum (1990), para detectar d erros simples, é necessário um código de distância d+1, pois assim não é possível converter

uma sequência de bits válida em outra com d erros simples. Da mesma maneira para corrigir d erros simples é necessária a elaboração de um código cuja distância seja 2d+1, pois assim as palavras transmitidas estão tão distantes, que mesmo com d mudanças, a palavra original ainda assim poderá ser unicamente determinada.

3 CONCEITOS SOBRE CÓDIGOS DE DETECÇÃO E CORREÇÃO DE ERROS:

Em um sistema que utilize códigos de correção ou detecção de erros, um codificador gera uma palavra de código contendo a informação recebida e o código de correção (ou detecção) de erros para essa informação.

As palavras de código fazem parte de um subconjunto em S de um universo U de vetores. Uma falha pode fazer com que uma palavra de código produza uma outra palavra de código contendo erro. Se essa palavra de código fizer parte de S, esse erro não será detectado, e se fizer parte de U-S, então o erro será detectado.

A detecção de erros pode ser feita enviando-se informações adicionais juntamente com os dados, logo, dados incorretos podem ser detectados e rejeitados.

Segundo Carlos Júnior (2000), podemos dividir os códigos de detecção de erros em dois modos: Os separáveis e os não separáveis.

Os considerados como separáveis são aqueles em que as palavras de código são construídas concatenando ao final dos bits de informação o código correspondente. Já os não-separáveis, a palavra de código é obtida com o entrelaçamento do código com a informação.

Na correção de erros, as informações adicionais enviadas são usadas para detectar e corrigir os dados, de modo que estes possam ser aceitos.

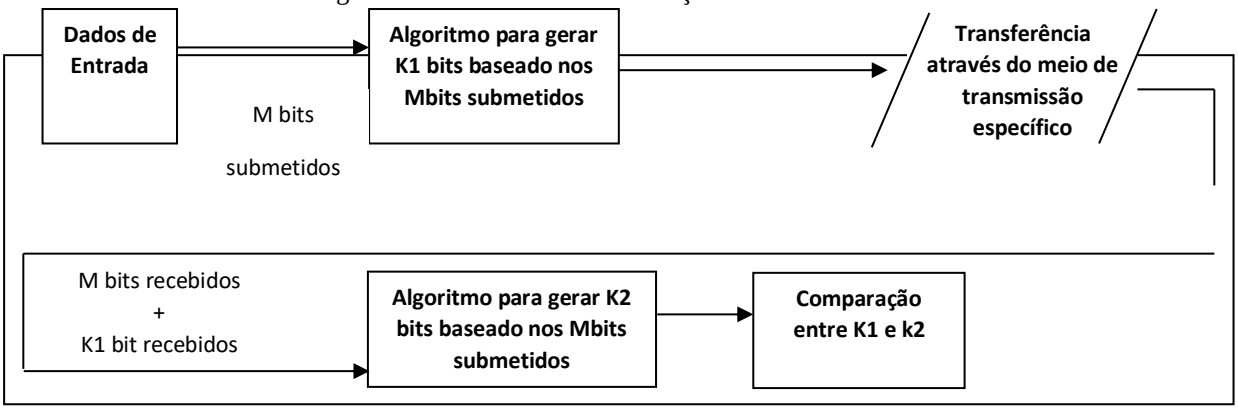
Com o uso dos códigos de detecção de erros, a maneira de se corrigir o erro seria com um pedido de retransmissão. Os códigos de correção de erros possuem no seu "corpo", condições para que o erro seja corrigido sem necessidade de uma retransmissão, porém, a quantidade de informação a mais necessária para que uma informação seja corrigida é muito grande, e esse tipo de código só é usado em casos em que uma retransmissão não é possível, ou é muito difícil, como é o caso de uma comunicação demorada na qual o uso de retransmissão agravaria a situação, ou no caso de uma gravação de dados em fita magnética (uni-direcional).

Segundo Lathi (1998), podemos dividir os códigos de correção de erros em C*í* (cíclicos) e *não –Cí (*não-cíclicos*)*. Os códigos cíclicos são aqueles determinados por estruturas algébricas, como polinômios, sendo seus codificadores e decodificadores projetados de acordo com suas especificações, e, onde uma rotação realizada na palavra de código gera uma nova

palavra de código. O mesmo conceito já não se aplica aos códigos não-cíclicos, uma vez que estes são códigos em concatenado, e em baixa densidade.

Consideraremos abaixo, alguns dos métodos de correção e/ou detecção de erros.

Figura 1- Processo Geral de Detecção de Erros



Fonte: **Adaptado de MONTEIRO (1995. p.108)**

4 MÉTODOS DE DETECÇÃO DE ERROS

4.1 PARIDADE:

Neste tipo de método, um **bit de paridade** é adicionado à informação de acordo com a paridade desejada, ou seja, a palavra código possuirá um número ímpar de bits 1, no caso de paridade ímpar, ou um número par de bits 1, quando a paridade for par.

Entendemos bit de paridade como um bit extra, o qual é transmitido normalmente com o conjunto de dados, cuja função é resultar em uma paridade (par ou ímpar).

Exemplo:

- Paridade par: dados 10010001, bit de paridade é 1.
- Paridade ímpar: dados 10010111, bit de paridade é 0.

O Código de Paridade é um caso do modo separável, utilizado no tratamento serial de informações, como detecção on-line de erros, sendo de fácil implementação. No entanto, conforme podemos verificar no exemplo abaixo, a paridade tem a limitação de somente poder detectar erros em números ímpares de bits.

Exemplo: Se os dados originais forem: 10010001 + 1 como bit de paridade (paridade par) e na transmissão 2 bits se alterarem, 10110011 + como 1 bit de paridade), será entendido como paridade par e o erro não será detectado.

Brazilian Journal of Development

4.2 CHECKSUM:

No Checksum, a palavra de código é formada por todas as palavras de informação concatenadas com um código gerado pela soma dessas palavras, ou seja, o código trata cada 16 bits como um inteiro, adicionando todos os inteiros em um módulo de dados.

Este método é um caso dos códigos separáveis, utilizado em comunicação de dados e armazenamento seqüencial, sendo sua implementação muito simples, pois utiliza apenas adições.

A vantagem é que esse código requer uma aritmética simples e rápida, porque só adiciona 02 bits à mensagem, ou seja, implementações rápidas de 16 bits usam aritmética de 32 bits e adicionam os restos no final, se algum *carry* for gerado, ele será adicionado aos bits menos significativos.

No entanto, muitas combinações diferentes de bits podem produzir o mesmo checksum, disfarçando possíveis erros. Além disso, o código funciona bem apenas com blocos contendo bastante quantidade de informação, e todas as palavras do bloco devem ser lidas para que o checksum seja calculado, consequentemente o tempo de detecção será o mesmo para um erro na primeira palavra ou na última.

Exemplo: Arquivos no formato hexagonal gerados pelos compiladores para uso em processadores da Intel.

4.3 CÓDIGO ARITMÉTICO:

É utilizado para detecção de erros em unidades que utilizam operações aritméticas, quando as mesmas precisarem dos operandos de uma função no seu formato original, portanto, eles não devem ser codificados.

Exemplo: a ULA, as funções de cálculo de endereços, o Código 3N e o Código de Resto.

4.4 CÓDIGO m-entre-n:

Geralmente este código é utilizado no modo não-separável, onde a palavra de código é formada pelos bits de informação entrelaçados com os bits de código, para a detecção de erros em sinais de controle, onde os bits devem ser interpretados separadamente.

Nos códigos m-entre-n, em uma palavra que contenha n bits, m são 1s e os restante são 0s, necessitando portanto, de bastante redundância para ser representado, pois no caso da palavra a ser representada possuir 4 bits (que representam 16 palavras diferentes), no mínimo

4 bits a mais de código são necessários formando o código 4-entre-8, que gera 70 palavras. Uma maneira de melhorar o sistema, ou seja, reduzir a redundância, seria usar um código 3-entre-5.

Exemplo: Dados os seguintes 4 sinais: 1010, 1100, 1001, 0011, 1110, 1011, 0111 e 1101. Qualquer outro código representa um sinal de controle errado, e deve ser detectado.
Se usarmos um código 4-entre-8, existirão 62 palavras que não serão detectadas como erradas. Todavia, com um código 3-entre-5, serão produzidas apenas 10 palavras, as quais 2 não serão detectadas. A detecção dessas duas palavras poderá então, ser realizada com o uso de um circuito simples.

5 MÉTODOS DE CORREÇÃO DE ERROS

5.1 CRC –VERIFICAÇÃO DE REDUNDÂNCIA:

Constitui-se um código *Cí*, onde a mensagem é tratada como um grande número binário, o qual é dividido por outro número binário fixo, sendo o resto da divisão transmitido juntamente com a mensagem. Quando a mensagem é recebida, a mesma divisão é realizada pelo receptor que compara o resto obtido com o transmitido, para validar os dados.

Exemplo: Considerando a transmissão do hexadecimal 0617, ou seja, o número binário 0000-0110-0001-0111, utilizando um registro de verificação de um bit e o divisor 1001, então o registro de verificação é o resto desta divisão. Enquanto neste caso o cálculo pode ser realizado utilizando-se um conjunto de registradores de 32 bits, na maioria dos casos isso não pode ser feito.
Com as devidas divisões realizadas, temos:
...0000010101101 = 00AD = 173 = quociente

0000011000010111 = 0617 = 1559 = dividendo
divisor 1001
0010 = 02 = 2 = resto
Em decimal, teríamos: 1559 dividido por 9 que é igual a 173 e tem como resto 2.
Ao receber a mensagem, o receptor a divide por 9 e verifica se o resultado é 2; se não for os dados foram transmitidos erroneamente.

Brazilian Journal of Development

5.2 CÓDIGO DE HAMMING:

Este código de correção de erros é usado para qualquer tamanho de palavra a ser transmitida, pois bits de paridade são adicionados a palavra original, formando uma nova palavra chamada palavra código.

Os bits são numerados começando de 1, sendo o bit 1 o mais à esquerda. Todos os bits cuja posição seja potência de 2 são bits de paridade, e os outros são de dados. Cada bit de paridade verifica posições específicas, e o bit de paridade é estabelecido (como 0 ou 1) de forma que o número de 1 nas posições verificadas por ele seja par ou ímpar, conforme a paridade definida no código. A palavra original adicionada aos bits de paridade é chamada de palavra-código.

Consideremos um exemplo, dada a palavra original de 8 bits 10010100.

0	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
2^0	2^1		2^2				2^3				

A palavra a ser transmitida seria de 12 bits: 011100110100.

As posições 1, 2, 4 e 8, como são potências de 2, conterão bits de paridade definidos de acordo com a paridade escolhida, por exemplo, consideraremos a paridade par.

Serão verificados pelo bit de paridade 1 (posição correspondente a 2^0) os bits cujas posições possuam 2^0 na sua composição, ou seja, os bits 1, 3, 5, 7, 9 e 11. Da mesma maneira, o bit de paridade 2 (2^1) verifica os bits 2, 3, 6, 7, 10 e 11, o bit de paridade 4 (2^2) verifica os bits 4, 5, 6, 7 e 12, e o bit de paridade 8 (2^3) verifica os bits 8, 9, 10, 11 e 12.

De forma geral, o bit b é verificado pelos bits $b_1, b_2, \ldots b_n$, tal que $b_1 + b_2 + \ldots + b_n = b$.

Por exemplo, o bit 7 será verificado pelos bits de paridade 1, 2 e 4.

Como os bits 3, 5, 7, 9 e 11 são respectivamente 1, 0, 1, 0 e 0, para que a quantidade de bits 1 na lista de posições verificadas pelo bit de paridade 1 seja par, o bit de paridade 1, deverá ser então 0. Da mesma maneira serão determinados os bits de paridade 2, 4 e 8 como 1, 1 e 1, respectivamente.

No momento que a palavra-código é recebida, é possível detectar se há algum erro, calculando os bits de paridade e verificando se alguma das listas verificadas pelos mesmos apresenta paridade diferente da determinada (no nosso exemplo, a paridade é par). Se houver erro, será possível determinar a posição incorreta fazendo-se a soma dos bits de paridade das listas que apresentaram paridade diferente da determinada. Desta maneira, é possível realizar

a correção do erro, simplesmente, alterando o bit de 0 para 1, ou vice-versa, na posição indicada.

Retornando ao exemplo, considere que a palavra-código transmitida de 12 bits 011100110100, sofreu algum tipo de alteração e foi erroneamente recebida como 011101110100, de tal forma que o bit da posição 6 foi alterado de 0 para 1. Quando o meio receptor a recebe, não tem maneira automática de saber que isso ocorreu. Aplicando então, novamente o princípio do código de Hamming, veremos que o bit de paridade 1, que verifica os bits 1, 3, 5, 7, 9 e 11 terá os dígitos 0, 1, 0, 1, 0 e 0, respectivamente, e então apresenta a quantidade de 1s par. Conclui-se então que o erro, se houver, não será em nenhuma dessas posições.

Continuando, o bit de paridade 2, verifica os bits 2, 3, 6, 7, 10 e 11, que apresentam os dígitos 1, 1, 1, 1, 1, e 0, que somados resultarão numa quantidade ímpar de 1s. Conclui-se que há erro, mas não podemos ainda determinar em qual posição ele ocorre, sabemos somente que é numa das posições verificadas pelo bit de paridade 2.

O bit de paridade 4 verifica as posições 4, 5, 6, 7 e 12, que são, 1, 0, 1, 1 e 0. Também apresentam uma quantidade ímpar de 1s. Sabe-se que há erro, e a partir de agora, podemos concluir que é em alguma posição verificada tanto pelo bit de paridade 2, quanto pelo bit de paridade 4. Até agora, podemos concluir que o erro ocorre nas posições 6 ou 7, porém já que a posição 7 foi verificada pelo bit de paridade 1, e o mesmo não apresentou erro, mesmo sem verificar as posições checadas pelo bit de paridade 8, podemos conclusivamente determinar que o erro ocorreu na posição 6.

Nesse momento, o bit da posição 6 pode ser alterado de 1 para 0, e ser extraída da palavra-código a palavra original enviada, e já corrigida: 10010100.

6 CONSIDERAÇÕES FINAIS

O codificador e o decodificador são projetados de acordo com o tipo de erro a ser tratado. Não existe um código de correção de erros genérico, assim para cada tipo de erro deve ser empregado um determinado código.

O processo de decodificação faz uma validação da palavra recebida, e executa uma técnica de correção de erros caso os mesmos forem detectados. Para verificar quando um erro ocorre no processo de transmissão, uma verificação de paridade é feita na palavra recebida em ordem inversa.

Existem diversos tipos de código, a escolha do código a ser utilizado é uma decisão importante, e deve ser tomada com cuidado. Uma escolha errada pode provocar uma degradação muito grande na performance do sistema.

A escolha entre códigos de correção ou detecção é simples, devido a sua enorme diferença de aplicação. Mas a escolha entre os tipos de código é mais complicada, e deve levar em consideração o tipo de erro a ser corrigido, que nem sempre é fácil de ser previsto. Vários códigos podem ser usados para um mesmo tipo de erro, e sua escolha dependerá da complexidade de sua implementação.

Um dos maiores problemas de se usar códigos de correção de erros, é que quanto maior a habilidade de corrigir erros, maior será o tamanho do código. Outro problema é quanto a complexidade matemática para implementação dos codificadores e decodificadores.

REFERÊNCIAS

ARAÚJO, A. C. M.; GOUVEIA, L. B. O digital nas instituições de ensino superior: um diagnóstico sobre a percepção docente em uma instituição de ensino superior em Belém do Pará (Brasil). **Braz. J. of Develop**., v. 6, n. 7, p. 42551-42555, 2020.

LATHI, B. P., **Modern Digital and Analog Communication Systems**. 3rded. New York: Oxford University Press, 1998.

MONTEIRO, Mário A., **Introdução à Organização de Computadores**. 2 ed. Rio de Janeiro: LTC, 1995.

TANEMBAUM, Andrew S. **Organização Estruturada de Computadores**. 3 ed. Rio de Janeiro: Prentice-Hall do Brasil, 1990.