REDES DE COMPUTADORES

UNIDADE 7 – Métodos de Detecção de Erros
(Aula 11 – Técnicas Convencionais e CRC)

Prof. Ivan Nunes da Silva

1. Métodos de Detecção de Erros

1.1 Erros na Transmissão de Dados

- Na conexão inter-redes, a transmissão de dados estão sujeitas a erros.
- Raios, surtos de energia e outras interferência eletromagnéticas podem introduzir correntes elétricas indesejadas nos componentes ou fios usados para comunicação.
- A interferência que é séria (especialmente raio) pode causar danos permanentes ao equipamento de rede.
- Uma pequena mudança no sinal elétrico pode fazer com que o receptor interprete mal um ou mais bits de dados.
- A interferência pode destruir completamente um sinal, significando que, embora o remetente transmita, o receptor não detecta a chegada de quaisquer dados.
- A interferência em um circuito de transmissão completamente inativo pode criar o efeito oposto, embora o remetente não transmita qualquer coisa, um receptor poderia interpretar a interferência lida como uma seqüência válida de bits ou caracteres.

2

1. Métodos de Detecção de Erros

1.2 Bits de Paridade e Verificação de Paridade

- **Processo de Verificação de Paridade**
  1. Mecanismo faz com que o remetente compute um bit adicional, denominado de bit de paridade (*parity bit*), e anexe-o a cada caractere antes do envio.
  2. Após todos os bits de um caracter serem recebidos, o receptor remove o bit de paridade.
  3. Receptor executa a mesma computação que o remetente e verifica se o resultado está de acordo com o valor de bit de paridade.
  4. A computação de paridade é escolhida de forma que se um dos bits do caracter é danificado em trânsito, a computação do receptor não concordará com o bit de paridade e o receptor indicará que aconteceu um erro.
  5. Existem duas formas de paridade, ou seja, **par e ímpar**.
  6. Ambos remetente e receptor devem concordar em qual forma será utilizada.

3

1. Métodos de Detecção de Erros

1.2 Bits de Paridade e Verificação de Paridade

- **Processo de Computação da Paridade**
  - **Paridade Par** → remetente fixa o bit de paridade para 0 ou 1 de forma que faça o número total de bits (inclusive o bit de paridade) um número par. Exemplos:
    - 01001010 → Bit de Paridade = 1 (Caracter contem um número ímpar de bits 1.
    - 01011010 → Bit de Paridade = 0 (Caracter já contem um número par de bits 1.
  - **Paridade Ímpar** → remetente fixa o bit de paridade para 0 ou 1 de forma que faça o número total de bits 1 (inclusive o bit de paridade) um número ímpar. Exemplos:
    - 11011010 → Bit de Paridade = 0 (Caracter já contem um número ímpar de bits 1.
    - 01110001 → Bit de Paridade = 1 (Caracter contem um número par de bits 1.
  - **A Paridade não detecta erros de transmissão que mudam um número par de bits.**

4

1. Métodos de Detecção de Erros

1.3 Detectando Erros com *CheckSum*

- **Características de Detecção de Erro com *CheckSum***
  - Muitos sistemas de rede enviam um *CheckSum* junto com cada pacote para ajudar o receptor a detectar erros.
  - Para calcular um *CheckSum*, o remetente trata os dados como uma seqüência de números e computa sua soma.
  - Os dados não são restritos a valores inteiros, podendo conter caracteres, números em ponto flutuante ou uma imagem.
  - O sistema de redes trata os dados meramente como um uma seqüência de inteiros com o propósito de calcular o *CheckSum*.

H	e	l	l	o		w	o	r	l	d	.
48	65	6C	6C	6F	20	77	6F	72	6C	64	2E

4865 + 6C6C + 6F20 + 776F + 726C + 642E = 71FC

5

1. Métodos de Detecção de Erros

1.3 Detectando Erros com *CheckSum*

- **Vantagens da Detecção de Erro com *CheckSum***
  - As principais vantagens derivam do tamanho e da facilidade de computação dos *CheckSums*.
  - O tamanho pequeno do *CheckSum* significa que o custo de transmissão do *CheckSum* é normalmente muito menor do que o custo de transmitir os dados.
  - O *CheckSum* só exige adição e o processamento necessário para criar ou verificar um *CheckSum* é pequeno.
  - A maioria das redes que empregam uma técnica de *CheckSum* usam um *CheckSum* de 16 ou 32 bits e geram *CheckSum* único para o pacote inteiro.

6

1. Métodos de Detecção de Erros

1.3 Detectando Erros com *CheckSum*

- **Desvantagens da Detecção de Erro com *CheckSum***
  - Tem a desvantagem de não detectar todos os erros comuns.
  - Por exemplo, a tabela abaixo mostra que um *CheckSum* não é suficiente para detectar erros de transmissão que inverte o segundo bit em cada um dos quatro bits enviados.

Item de Dados em Binário	Valor do Checksum	Item de Dados em Binário	Valor do Checksum
0001	1	0011	3
0010	2	0000	0
0011	3	0001	1
0001	1	0011	3
totais	7		7

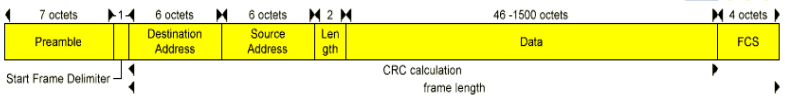
  - Para estender o exemplo para um pacote inteiro, imagine que os quatro itens modificados acontecem no meio de vários outros.

7

1. Métodos de Detecção de Erros

1.4 Detectando Erros com CRC

- **Características da Técnica:**
  - Mecanismo baseado em códigos de Verificação de Redundância Cíclica (*Cyclic Redundancy Checks*).
  - Técnica que permite detectar erros em múltiplos bits (erros por rajada).
  - É provado matematicamente que a CRC consegue detectar mais erros que um *CheckSum*.
  - Dada uma mensagem de **n** bits, em lugar de se acrescentar 1 bit de paridade será adicionada uma seqüência de **k** bits denominada **FCS (*Frame Check Sequence*).**
  - O FCS é determinado por um polinômio gerador **P(x)** tal que os **n**+**k** bits transmitidos sejam divisíveis por **P(x).**



8

1. Métodos de Detecção de Erros

1.4 Detectando Erros com CRC

- **Passos do Mecanismo CRC (Parte I):**
  - O transmissor acrescenta o FCS e o receptor verifica se a Mensagem acrescida do FCS é divisível por **P(x)** com resto zero. Caso não seja, ocorreu erro de transmissão.
  - Sendo **M** a mensagem de **n** bits a ser transmitida, esta mensagem pode ser considerada como um polinômio **M(x)** de grau **n-1** que possui o termo x^i se o **i-ésimo** bit de **M** é **1**.
  - Como por exemplo:

    Se M = 1010001101

    Então n = 10 e M(x) = $x^9 + x^7 + x^3 + x^2 + 1$

1. Métodos de Detecção de Erros

1.4 Detectando Erros com CRC

- **Passos do Mecanismo CRC (Parte II):**
  - Considera-se uma seqüência de **k** bits correspondente a um **Polinômio Gerador P(x)** de grau **k-1** como o seguinte:

    Se P = 110101

    Então k = 6 e **P(x)** = $x^5 + x^4 + x^2 + 1$
  - **O Polinômio Gerador P(X) de grau r=k-1 deve ser conhecido tanto pelo transmissor como pelo receptor.**
  - Deve-se calcular **F(x)** tal que o quadro **T(x)** a ser transmitido:

    $T(x) = x^r.M(x) + F(x)$, seja então divisível por **P(x)**.
  - **F(x)** pode ser obtido como o resto da divisão de **$x^r.M(x)$** por **P(x)**.

1. Métodos de Detecção de Erros

1.4 Detectando Erros com CRC

- **Algoritmo Para Geração da Soma Verificadora (FCS) e do Quadro Completo de Transmissão T(x):**
  1. Definir a cadeia **M** de **n** bits a ser transmitida.
  2. Montar o polinômio **M(x)** de grau **n-1** a partir de **M**, sendo que os bits de **M** corresponde aos coeficientes de **M(x)**.
  3. Definir a cadeia **P** de **k** bits que gerará o polinômio gerador **P(x)**.
  4. Obter o polinômio **P(x)** de grau **r = k-1** a partir de **P**, sendo que os bits de **P** corresponde aos coeficientes de **P(x)**.
  5. Obter o quadro completo **T(x)** a ser transmitido, o qual é formado por **T(x)** = x^r**.M(x)** + **F(x)**, onde **F(x)** é o Frame Check Sequence (FCS) de grau **r-1** (que contem então **r** bits), sendo dado pelo resto da divisão de x^r**.M(x)** por **P(x)**.
  6. Transmitir o quadro **T** que é formado pela cadeia original **M** concatenada com a cadeia **F** representando o FCS.

11

1. Métodos de Detecção de Erros

1.4 Detectando Erros com CRC

- **Algoritmo Para Recepção do Quadro Completo T(x) e Verificação de Ocorrência de Erro:**
  1. Reconstituir o polinômio **T**(**x**) a partir do quadro de bits **T** que foi transmitido.
  2. Obter o polinômio **R**(**x**) que é dado pela divisão do polinômio **T**(**x**) por **P**(**x**).
  3. Se **R**(**x**) for um polinômio nulo, então não houve erros durante a transmissão; caso contrário, houve erros durante a transmissão.
  4. Se não houve erros, a cadeia **M** de bits originais poderá ser obtida a partir de **T**, bastando retirar os **k** bits mais a direita do quadro total **T**.

12