Cap. 03 – Controle de Erro

Referências Bibliográficas

- Gerard J. Holzmann – “Design and Validation of Computer Protocols” – Prentice Hall; Englewood Clifs; New Jersey; 1991.
- Andrew S. Tanenbaum - “Computer Networks” - Prentice Hall; Englewook Cliffs; New Jersey; 1989; ISBN 0-13-166836-6

- Paulo Coelho - “Material de Aula” - Arquitetura de Redes de Computadores (FACOM49070 - Mecatrônica)
- Pedro Frosi - “Material de Aula” - Arquitetura de Redes de Computadores (GBC056 - Ciência da Computação)

3.1 - Introdução

- “**estatísticas**” - … nro de erros causados na transmissão de dados são muitas ordens de grandeza do nro de erros causados por falhas de “hardware” nos sistemas computacionais.
- … diferença de magnitude entre uma probabilidade de erro de 10^{-15} e outro de 10^{-4} não pode ser subestimada.

- e.g., … considere uma taxa de erros de bits de 10^{-15} em uma linha de transmissão com capacidade de 9600 bps.
- ... tem-se 01 erro de bit a cada 3303 anos de operação contínua.

… 3.1 - Introdução

- e.g., … considere uma taxa de erros de bits de 10^{-15} em uma linha de transmissão com capacidade de 9600 bps.
  - ... teremos 01 erro de bit a cada 3303 anos de operação contínua.
  - 10^{15} / 9600 bps = 104.166.666.667 segundos →
  - 104.166.666.667 / 60 = 1.736.111.111,11 minutos →
  - 1.736.111.111,11 / 60 = 28.935.185,18 horas →
  - 28.935.185,18 / 24 = 1.205.632,71 dias →
  - 1.205.632,71 / 365 = 3.303,10 anos.

- "**alguns valores**" .. para taxas de erros de bits como funçao do meio de transmissão, p.ex., "hardware" = 10^{-15}; fibra ótica = 10^{-19}; cabo coaxial = 10^{-6} e par trançado = 10^{-4} a 10^{-5}

… 3.1 - Introdução

- Observação … diferença de magnitude entre uma probabilidade de erro de 10^{-15} e outro de 10^{-4} não pode ser subestimada.
- e.g., … diferença entre taxas 10^{-15} e 10^{-4} é enorme se considerarmos a taxa de transmissão de 100.000.000 bps (100 Mbps) ?!
  - 10^{-15} .. 01 bit será corrompido a cada 115 dias
  - 10^{-4} .. 10.000 bits serão corrompidos a cada segundo

- e.g., para transmitir 10^{15} bits a uma taxa de transferência de 100 Mbps, são necessários $10^{15} / 10^{8} = 10^{7}$ segundos.
- 10^{7} segundos corresponde $10^{7} / 60 = 166.666,6667$ minutos, que por sua vez corresponde $166.666,6667 / 60 = 2.777,7778$ horas, que por sua vez corresponde $2.777,7778 / 24 = 115,7407$ dias.

… 3.1 - Introdução

- Observação … diferença de magnitude entre uma probabilidade de erro de 10^{-15} e outro de 10^{-4} não pode ser subestimada.
- e.g., … diferença entre taxas 10^{-15} e 10^{-4} é enorme se considerarmos a taxa de transmissão de 100.000.000 bps (100 Mbps)
  - 10^{-15} .. 01 bit será corrompido a cada 115 dias.
  - 10^{-4} .. 10.000 bits serão corrompidos a cada segundo

- … dependendo das características da linha de transmissão, até ruídos podem ser inseridos como se fossem dados.
- … dependendo da rede e dos elementos de interconexão, os dados podem ser reordenados, distorcidos ou removidos.

… 3.1 - Introdução

- "**controle de erro**" … objetivo é aumentar a confiabilidade da transmissão, preferivelmente para um nível de confiabilidade de operação "standalone" de um sist. computacional.
  - … não se deve esperar dos métodos de controle de erro a captura de todos os possíveis erros que a princípio podem ocorrer !!
  - ... um controle de erro efetivo deve considerar as características de erro do canal a ser usado … "questão frequentemente negligenciada".

- Obs.: … se a taxa de erro de um canal é << que a de um equip. periférico, a inclusão de controle de erro é a solução ?!
- … não necessariamente, pois neste caso a inclusão do controle de erro pode degradar desnecessariamente o desempenho e até diminuir a confiabilidade em vez de aumentá-la.

3.2 – Modelo de Erro

- "**Discrete Memoryless Channel**" - modelo formal para este tipo de canal é o canal discreto sem memória.
- "**canal discreto**" .. canal é dito discreto porque ele reconhece apenas um número finito de níveis de sinais distintos.
- "**canal sem memória**" .. canal é dito sem memória porque a probabilidade de um erro é suposta ser independente dos erros anteriores.

- "**Canal Simétrico**" - p (bit error 0) = p (bit error 1)

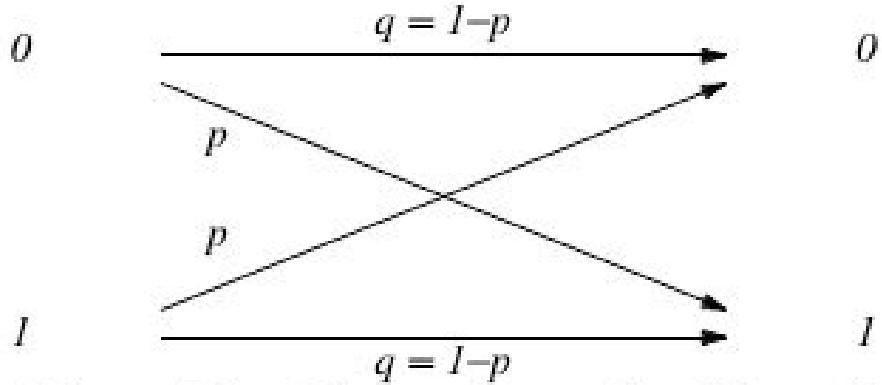

Figure 3.1 — Discrete Memoryless Channel

… 3.2 – Modelo de Erro

- **Canal Simétrico Binário** … estabelece a probabilidade “Pr” de se ter ao menos “n” bits contíguos livres de erro EFI >= “n”
- Pr(EFI >= n) = $(1 – b)^{n}$ onde EFI “Error Free Interval”
- … onde “n” >= 0 e “b” = taxa média de erros de bits.

- e.g., considere b = 0.5, então (1-b)^n = 0.5^n, o que implica que a razão entre 2 resultados consecutivos é linear.
  - n = 1 então Pr(EFI >= 1) = 0.5
  - n = 2 então Pr(EFI >= 2) = 0.25
  - n = 3 então Pr(EFI >= 3) = 0.125
  - n = 4 então Pr(EFI >= 4) = 0.0625

… 3.2 – Modelo de Erro

- "**Canal Simétrico Binário**" … estabelece que a probabilidade "Pr" de se ter ao menos "n" bits contíguos livres de erro EFI >= "n" é:
- Pr(EFI >= n) = $(1 – b)^{n}$ onde EFI "Error Free Interval"
- … onde "n" >= 0 e "b" = taxa média de erros de bits.

- e.g., conside b = 0.5, então (1-b)^n = 0.5^n o que implica que a razão entre 2 resultados consecutivos é linear.
- … probabilidade decresce linearmente a medida que o tamanho do intervalo livre de erros decresce (EFI);
- … de maneira similar, probabilidade que a duração de uma rajada exceda "n" bits decresce linearmente com "n".

… 3.2 – Modelo de Erro

- Para expressar “Pr” como uma função exponencial do tamanho do intervalo livre de erros (EFI >= n), substitui-se a equação …
- Pr(EFI >= n) = e ^ [-b*(n-1)]
- … onde “n” >= 1 e “b” = taxa média de erros de bits.

- “**acurácia da predição**” .. comparar os resultados da equação com valores empíricos ou experimentais (melhor forma de avaliar a acurácia da predição)
- … estudos monstram que a equação acima (exponencial) prediz intervalos livres de erros melhor que a equação anterior (linear).
- … mas, a equação pode ainda ser melhorada com a adição de mais um fator que determina o fator de agrupamento » Benoit Mandelbrost.

3.3 – Erros de Transmissão

- “**principais erros de transmissão**” .. como aparecem / apresentam ?!

- **inserção / remoção de dados** – … normalmente causado por perda temporária de sincronização entre transmissor e receptor.
- **remoção de dados** - … podem ser causados artificialmente por disciplinas inadequadas de controle de fluxo.
- **duplicação de dados** - … podem ser gerados intencionalmente, p.ex., pelo transmissor ao implementar protocolo com retransmissão.
- **distorção de dados** - … podem ser gerados por variações nas condições de operação do canal, inserindo distorções nos dados.
- **reordenação de dados** - … podem potencialmente ser causados quando os dados percorrem diferentes rotas.

… 3.3 – Erros de Transmissão

- “**solução**” .. esquemas de controle de fluxo resolver os problemas de “**remoção**” .. “**duplicação**” e “**reordenação de dados**”.

- … não obstante, em todos os casos, inserções e distorções podem ocorrer, assim, fazem-se necessários métodos para verificar a consistência dos dados.
- **inserção / remoção de dados –** … normalmente causado por perda temporária de sincronização entre transmissor e receptor.
- **distorção de dados -** … podem ser gerados por variações nas condições de operação do canal, inserindo distorções nos dados.

3.4 – Redundância de Dados

- ”**Detecção de Erro**” .. somente funciona se aumentarmos a redundância de dados de algum modo na mensagem.

- Além da detecção de erros de transmissão, o receptor pode também ser capaz de corrigir os erros segundo 02 maneiras ..
- “**Forward Error Control**” .. redundância se dá de tal modo que o receptor reconstroi a mensagem a partir da mensagem distorcida.
- “**Feedback Error Control**” .. redundância de dados se dá de tal modo que o receptor apenas detecta que a mensagem contém distorções;

… 3.4 – Redundância de Dados

- "**Feedback Error Control**" .. redundância de dados se dá de tal modo que o receptor é capaz de apenas detectar erros (se houver)
- … códigos de transmissão correspondentes são denominados "Error Detecting Codes" ou Códigos de Detecção de Erros.
- … aplicáveis em redes de computadores e no processamento para-lelo onde custo do reenvio da informação é viável.

- "**Foward Error Control**" .. ECCs capazes de detectar e corrigir erros em primitivas apresentam complexidade / computabilidade bem maior que em ("Feedback Error Control").

… 3.4 – Redundância de Dados

- "**Feedback Error Control**" ... há duas possibilidades:

- transmissor com alta taxa de erro » retransmissão pode ser requerida explicitamente, através de uma confirmação negativa;
- … neste caso o receptor simplesmente descarta a primitiva corrompida e espera por uma retransmissão da mensagem;

- transmissor com taxa de erro suficientemente baixa » ausência de confirmação indica sucesso.

… 3.4 – Redundância de Dados

- "**Controle de Erro**" - tem o objetivo de diminuir a taxa de erro de um determinado canal de comunicação.
- … contudo, nem todos os erros podem ser detectados, então, sempre existe uma 'Taxa de Erro Residual' (RER – "Residual Error Rate").

- e.g., suponha-se que a probabilidade de erros de transmissão em msgs. seja "p", e que o método de controle de erro identifique uma fração "f" deste conjunto de erros, então:

$$RER = p * (1 – f)$$

- ... esta equação implica que, por instância, a taxa residual de erros é da ordem de 10^{-9} (10^-9) ou menos.

3.5 – Tipos de Codificação

- São 02 os tipos básicos de codificação:
- “Block Codes”
  - todas as “code words” têm possivelmente o mesmo tamanho;
  - codificação para cada msg. de dado pode ser definida estaticamente.

- “**Convolution Codes**” .. há uma relação entre a codificação da primitiva corrente e a codificação das primitivas anteriores …
- “**code word**” produzida depende da mensagem de dados e de um nro. prévio de mensagens codificadas.
- codificador muda seu estado com cada msg. que é processada, mas o comprimento da “code word” é usualmente estático.

… 3.5 – Tipos de Codificação

- Palavras Códigos ou “Code Words” podem ser classificadas:

- “**Linear Codes**” .. toda combinação linear de palavras códigos válidas produz uma outra palavra código válida (mod-2 sum).

- “**Cyclic Codes**” .. cada deslocamento cíclico (cyclic shift) de uma palavra código válida produz uma palavra código válida (CRC).

- “**Systematic Codes**” .. toda palavra código inclui bits da primitiva original, seguida ou precedida por um grupo de bits de “checks”.

… 3.5 – Tipos de Codificação

- Em todos os casos, as palavras códigos são maiores que as palavras de dados sobre as quais se baseiam;
- … se o nro. de bits originais é “d” e o nro. de bits adicionais é “e”, a razão “d / (d + e)” é chamada “code rate”.
- … melhorar a “code rate” frequentemente significa aumentar a redundância e por consequência diminuir a “code rate”.

- e.g. … reduzir taxa de erro de um canal por um fator de 5 * 102 usando, p.ex., “Forward Error Control” pode exigir um código com um “code rate” <= 0.5.
- “code rate” <= 0.5 » “redundância é maior do que a própria msg.”

3.6 – Verificação por Paridade

- Se a probabilidade de erros de múltiplos bits por msg. é baixa, tudo que é necessário para o controle de erro em um canal simétrico binário é o código de verificação por paridade.
- … para toda mensagem que adicionarmos 1 único bit, teremos a soma módulo 2 naquela mensagem igual a 1 (um).
- … se um único bit, incluindo o bit de verificação sofre mudança, o cálculo da paridade no receptor indica erro, mas não pode ser corrigido.

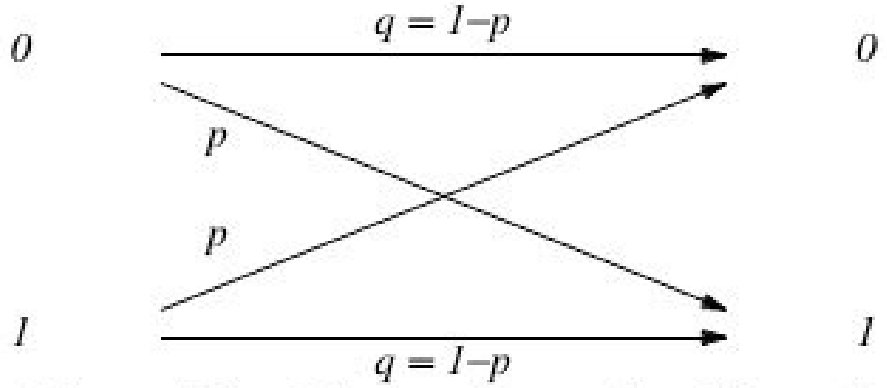

Figure 3.1 — Discrete Memoryless Channel

… 3.6 – Verificação por Paridade

- Se “q = 1 – p”, então a probabilidade de transmissão livre de erro para msgs. de “n + 1” bits, sendo 1 bit de paridade, é “$q^{(n+1)}$”.
- … probabilidade de erros de 1 bit em “n + 1” bits transmitidos é a probalidade binomial = “$(n + 1) * p * q^n$ ”

- “**Canal Simétrico**” .. p (bit error 0) = p (bit error 1)

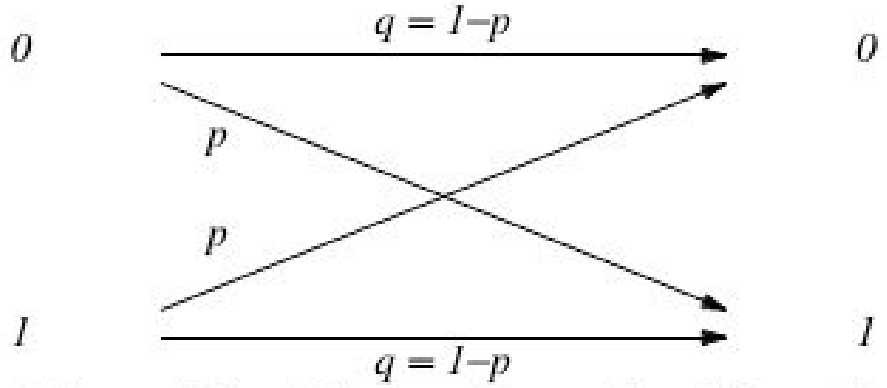

Figure 3.1 — Discrete Memoryless Channel

… 3.6 – Verificação por Paridade

- … probabilidade de erros de 1 bit em “n + 1” bits transmitidos é a probalidade binomial, ou seja, … $(n + 1) * p * q^n$
- … sob tais circunstâncias, e.g., canal sem memória, a taxa de erro residual da verificação por paridade de 1 bit é :

$$1 – q^{(n + 1)} – (n + 1) * p * q^n$$

- e.g., … para n = 15 e $p = 10^{-4}$ teremos taxa de erro residual da ordem de 10^{-6} por mensagem ou 10^{-7} por bit.

… 3.6 – Verificação por Paridade

- … linha contínua mostra como a taxa de erro residual por “code word” aumenta como uma função da taxa “p” de erro de bits.

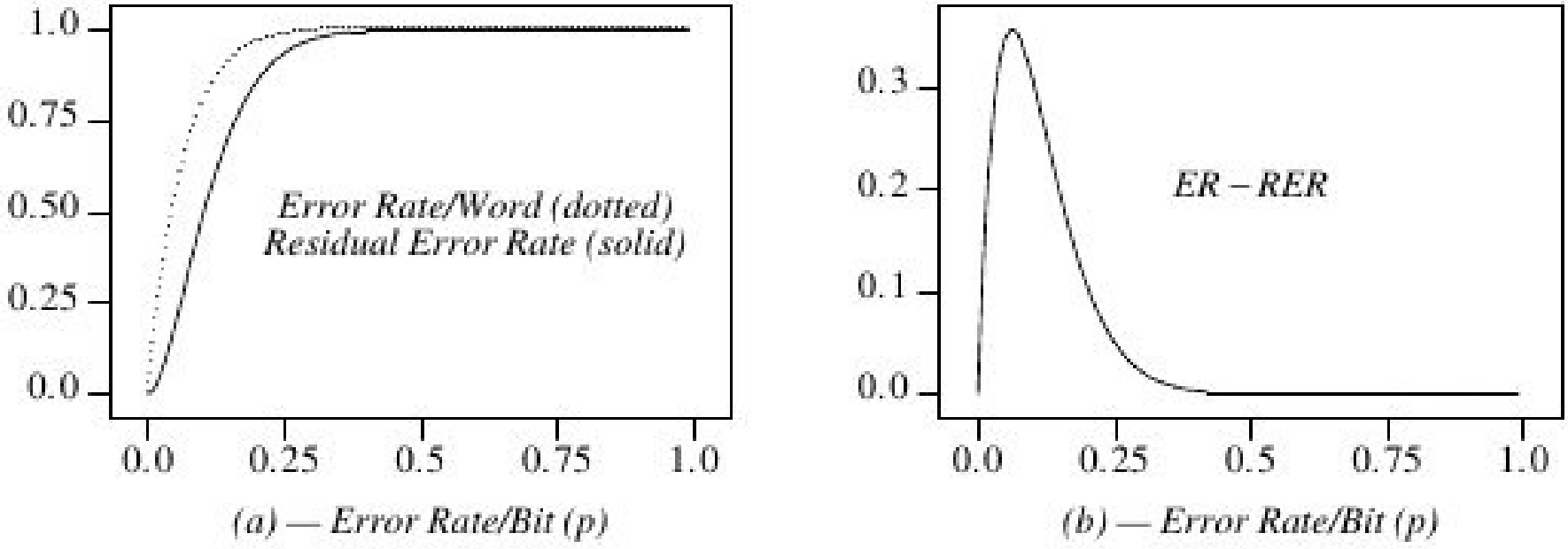

Figure 3.2 — Residual Error Rate of a 1-bit Parity Check, n=15

… 3.6 – Verificação por Paridade

- Nem todos os erros podem ser detectados e, então, sempre existe uma "Taxa de Erro Residual" (RER) = p * (1 – f) …
  - "f" - fração de erros que é identificada; "p" - probabilidade de erro.

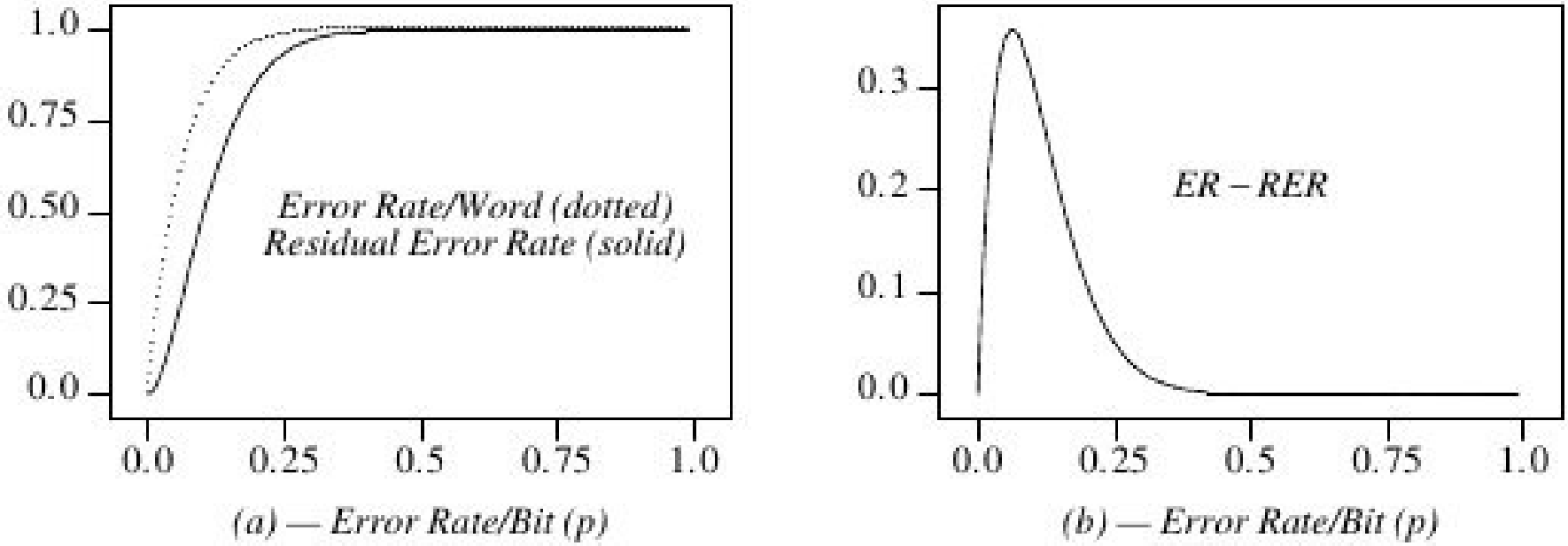

Figure 3.2 — Residual Error Rate of a 1-bit Parity Check, n=15

3.7 – Correção de Erro

- “Forward Error Control” usa somente um conjunto pequeno de combinações disponíveis de bits para codificar as msgs.
  - … códigos são escolhidos de modo que se tenha relativamente um nro. grande de erros de bits para converter uma msg válida em outra.

- “code rate” - … para um código de correção de erro é em geral menor que aquele para um código de detecção de erro.

- Lembre-se que “d” - nro. de bits originais é “d”; “e” - nro. de bits adicionais na “code word”; e “d / (d + e)” - chamada “code rate”;
  - … logo que aumentamos “e” → “d / (d + e)” diminui, ou seja, maior é a redundância para códigos de correção do que somente de detecção.

… 3.7 – Correção de Erro

- “Forward Error Correction” - … por apresentar “code rate” <<< que o “Feedback Error Control”, quando se deve utilizar ?!

- Deve ser considerado somente quando as msgs. de controle do RX para o TX for um problema por razões tais como:
  - valores altos para o atraso de transmissão;
  - taxa de erros de bits alta;
  - ausência de canal de retorno (receptor → transmissor).

… 3.7 – Correção de Erro

- e.g., … problema de comunicação de uma estação espacial (receptor) e o centro de controle (transmissor) na terra;
  - … sinal de controle para liberar o obturador da câmera ou ajustar um curso, pode exigir vários minutos até alcançar a estação;
  - … neste cenário, pode não haver tempo suficiente para repetir o sinal no caso de erro de transmissão (sinal pode chegar como pode se perder).

- e.g., taxa alta de erros de bits → neste caso, até mesmo a probabilidade de uma requisição para uma retransmissão ser recebida corretamente é inaceitavelmente baixa.
  - … coloca em dúvida a própria requisição para retransmissão.

… 3.7 – Correção de Erro

- Até mesmo a Verificação de Paridade Simples por “code word” pode ser estendida de um código de detecção de um único erro para código de correção de um único erro.
  - e.g., sequência de 7 bits é estendida de 1 bit, o que torna o nro. de bits na sequência um nro. par … “Longitudinal Redundancy Check” ou LRC.
  - 1000100 + 0 onde “0” é o bit de paridade.
  - … na sequência inclui-se redundância para a série de “n” códigos, ou seja, “Vertical Redundancy Check” ou VRC

		LRC	
D	=	1000100	0
A	=	1000001	0
T	=	1010100	1
A	=	1000001	0

		0010000	1 VRC

… 3.7 – Correção de Erro

- “Hamming Distance” - nro. de bits a serem alterados na msg. de modo a obter um outra msg. válida - mínima diferença entre “code words”
  - … diferença entre 02 “code words” é definida como o nro. de bits nos quais as “code words” se diferem, ou seja, em nro de bits.

- Código com “Hamming Distance” de “n”, implica que qualquer combinação de até “n – 1” erros de bit pode ser detectada;
  - … qualquer combinação até “(n – 1)/2” de erros de bits por código pode ser corrigida se o receptor interpretar toda palavra de código não válida como a palavra de código válida mais próxima.
  - … este método é formalmente chamado de “Maximum Likelihood Decoding” ou “Nearest Neighbor Decoding”.

… 3.7 – Correção de Erro

- Aumentando a Distância de Hamming, ou seja, escolhendo “code words” mais longas, aumenta-se a confiabilidade de um código tanto mais quanto maior a “code word”.

- Obs.: Redundância de um Código determina seu poder de detecção e correção de erros de transmissão.
- … redundância pode ser redefinida como o nro de bits sobre o mínimo exigido para codificar inequivocamente uma mensagem.

3.8 – “Linear Block Code”

- Para codificar uma de “n” msgs. precisamos de “m” bits, sendo, que “m” = inteiro superior e mais próximo (log “n” na base “2”)
- … pode-se proteger os “m” bits adicionando “c” bits de verificação e escolhendo “n” códigos de um total de $2^{(m + c)}$ códigos;
- ... de tal forma que cada combinação de 02 códigos válidos contemple a maior diferença em bits quanto possível.

- e.g., considere 17 msgs diferentes, então são necessários “m” bits para $\log_2 17$ (log 17 na base 2) → 5 bits (inteiro mais próximo).

3.8 – “Linear Block Code”

- … pode-se proteger os “m” bits adicionando “c” bits de verificação e escolhendo “n” códigos de um total de $2^{(m + c)}$ códigos;
- ... de tal forma que cada combinação de 02 códigos válidos contemple a maior diferença em bits quanto possível.

Table 3.3 — Parity Protection

c	m	m/(m+c)
1	0	0.00
2	1	0.50
3	4	0.57
4	11	0.73
5	26	0.84
6	57	0.90
7	120	0.94
8	247	0.97

… 3.8 – “Linear Block Code”

- Com uma boa aproximação, o nro de bits de dados que podem ser protegidos cresce exponencialmente com o nro. de bits de verificação, ou seja, “c” bits de verificação.

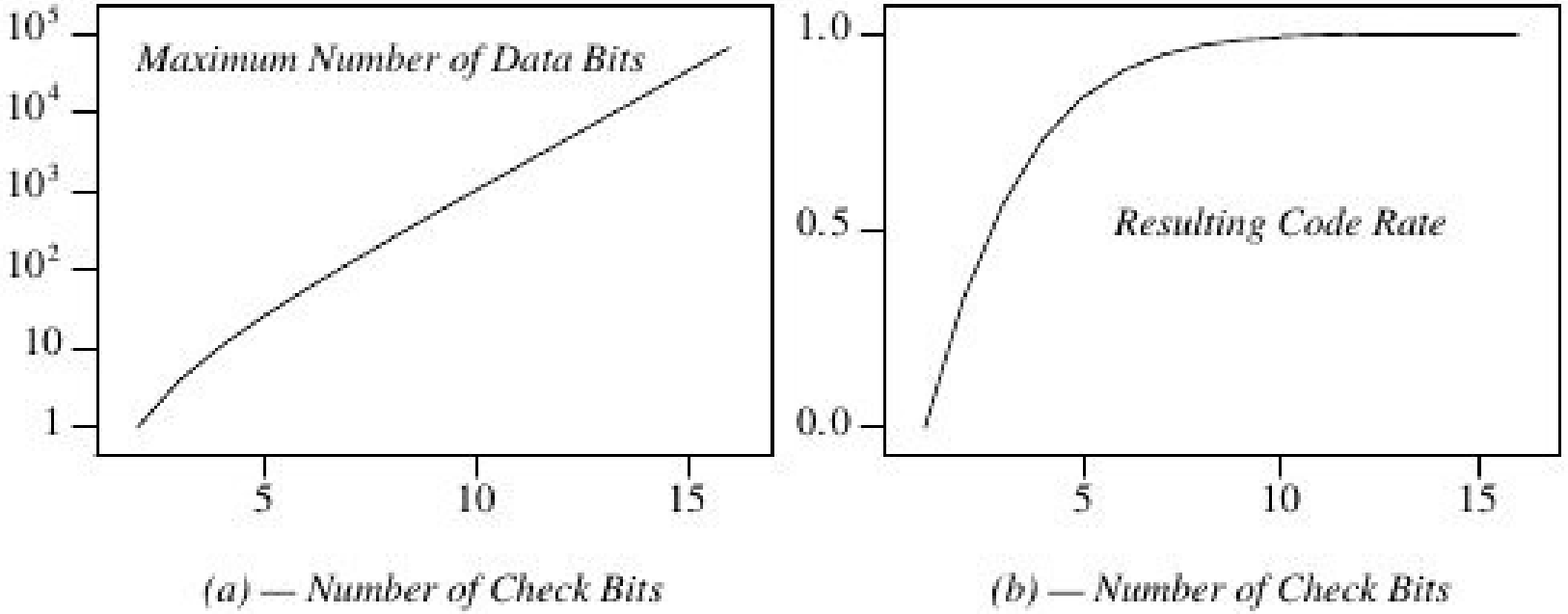

Figure 3.3 — Parity Protection

… 3.8 – “Linear Block Code”

- Para corrigir erros de 1 bit, precisa-se de uma distância no Código de Hamming de ao menos 3 entre “code words”;
- … neste contexto, quantos bits “c” de verificação precisamos ??

- … para toda “code word” de “m+c” bits, tem-se precisamente “m+c” códigos resultantes com erros de 1 bit;
- … para cada palavra no espectro de “2^{m}” possíveis códigos de dados, precisamos de “m+c+1” palavras para protegê-la contra erros de 1 bit → total de palavras no código é $(m + c + 1) * 2^{m}$
- $(m + c + 1) * 2^{m} = 2^{(m + c)}$ → “$m + c + 1 = 2^{c}$” .. permite calcular o nro mínimo de bits de verificação dado o nro de bits de dados.

… 3.8 – “Linear Block Code”

- $(m + c + 1) * 2^m = 2^{(m + c)}$ → “$m + c + 1 = 2^c$” o que permite calcular o nro mínimo de bits de verificação dado o nro de bits de dados.
- … também podemos calcular o nro máximo de bits de dados para um dado nro de bits de verificação “c”

Table 3.3 — Parity Protection

c	m	m/(m+c)
1	0	0.00
2	1	0.50
3	4	0.57
4	11	0.73
5	26	0.84
6	57	0.90
7	120	0.94
8	247	0.97

… 3.8 – “Linear Block Code”

- Mesmo efeito da Tab. 3.3 (anterior) é ilustrado na Fig. 3.3 para até 16 bits de verificação.

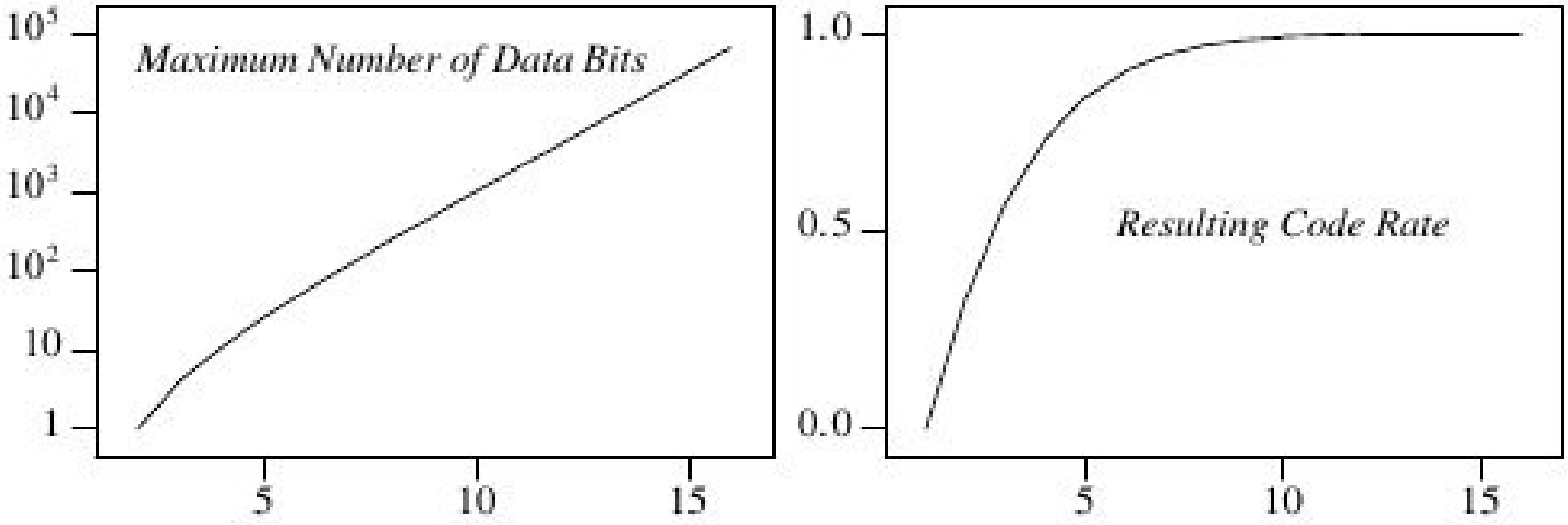

Figure 3.3 — Parity Protection

… 3.8 – “Linear Block Code”

- e.g. “hamming code” - … em uma mensagem de “m” bits, inclui-se “c” bits de verificação => “m + c” bits na mensagem;
- “idéia” - sobrepor os bits de paridade, de forma que eles consi-gam verificar-se uns aos outros, bem como os bits de dados.
- … código de Hamming é obtido a partir da palavra de dados, inserindo pontos de controle, denominados bits de paridade

Posição do bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
bits de paridade	p1	X		X		X		X		X		X		X		X		X		X		…
	p2		X	X			X	X			X	X			X	X			X	X		
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

… 3.8 – “Linear Block Code”

- e.g. “hamming code” - … em uma mensagem de “m” bits, inclui-se “c” bits de verificação => “m + c” bits na mensagem;
- … para “c” bits de verificação, teremos “m + c = 2^c -1”, ou seja, nro máximo de bits na mensagem será “m = (2^c - 1) - c”;
- … e.g., para “m = 4” e “c = 3” (bits de paridade) => 7 bits na msg; para “m = 11” e “c = 4” (bits de paridade) => 15 bits na msg.

Posição do bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
bits de paridade	p1	X		X		X		X		X		X		X		X		X		X		
	p2		X	X			X	X			X	X			X	X			X	X		…
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

… 3.8 – “Linear Block Code”

- “Hamming Code” - … bits na palavra de código são numerados de “1” a “m+c” sendo que “i-ésimo” bit de verificação é colocado na posição “2^i” … $0 <= i <= \log_2^{(m+c)}$
  - … bits de verificação são colocados no palavra de código de tal maneira que a soma da posição dos bits que eles ocupam aponte para o bit errado para qualquer erro de 1 único bit (mas não para 2 bits).

Posição do bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
bits de paridade	p1	X		X		X		X		X		X		X		X		X		X		…
	p2		X	X			X	X			X	X			X	X			X	X		
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

… 3.8 – “Linear Block Code”

- … bits de verificação são colocados no palavra de código de tal maneira que a soma da posição dos bits que eles ocupam aponte para o bit errado para qualquer erro de 1 único bit;
  - … qdo um posição é escrita como a soma de potências de 2, p.ex., (1 + 2 + 4), estas potências também apontam para os bits de verificação que cobrem o bit em questão … neste caso bit de dados – 7.

Posição do bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
bits de paridade	p1	X		X		X		X		X		X		X		X		X		X		…
	p2		X	X			X	X			X	X			X	X			X	X		
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

… 3.8 – “Linear Block Code” (LBC)

- “Representação Matricial” - método conveniente para “LBC”
  - e.g., … considere um código com 3 bits de dados (D1, D2 e D3) e 03 bits de verificação (C4, C5 e C6).
  - … seja C4 = D1 + D2; C5 = D1 + D3; e C6 = D2 + D3, então com algum rearranjo podemos representar as equações na forma de matricial.

$$\begin{bmatrix} C4 \\ C5 \\ C6 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} D1 \\ D2 \\ D3 \end{bmatrix}$$

$$\begin{aligned} D1 + D2 + \quad + C4 \quad\quad\quad &= 0 \\ D1 + \quad + D3 \quad + C5 \quad\quad &= 0 \\ \quad D2 + D3 \quad\quad + C6 &= 0 \end{aligned}$$

… 3.8 – “Linear Block Code” (LBC)

- C^t é a matriz transposta de palavras de dados, escrita como um vetor de bits e que de acordo com a equação anterior.
  - Matriz de Dados + Verificação * C^t produz o vetor “zero”.

- Podemos representar a matriz na forma: $H * C^t = 0$
  - … onde H é a Matriz de Paridade (Dados + Verificação).

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot C^t = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

… 3.8 – “Linear Block Code” (LBC)

- … erros de transmissão podem ser formalizados pela adição de um vetor E, ou seja, $H * (C^t + E) = s$
  - … onde S é chamado de Síndrome.

- Neste código, toda soma módulo 2 de palavras de código válidas produz uma outra palavra de código válida;
  - … assim, se o vetor de erros E coincide com alguma palavra de código válida, a síndrome é zero e o erro não é detectável !!

3.9 – Cyclic Redundancy Checks

- “Cyclic Redundancy Check” - algoritmo de código cíclico mais utilizado em redes de computadores;
  - ISO especifica um algoritmo similar denominado FCS (Frame Check Sequence) utilizado em seus protoclos IEEE802.

- CRC contempla o produto de bits de verificação de uma primitiva, cuja amostragem é definida segundo um critério.
  - … bits escolhidos fazem parte do fator (polinômio gerador) e, portanto, é transmitido o produto, ou seja, primitiva * fator;
  - .. no destino faz-se a divisão do que é recebido pelo fator, se resto é zero, significa ausência de erro ou erro não detectável;
  - … método de divisão é específico e o fator (polinômio gerador) usado determinam o leque de erros de transmissão que podem ser detectados.

… 3.9 – Cyclic Redundancy Checks

- Seja uma primitiva - sequência de “n” bits que pode ser repre- sentada por um polinômio de grau “n-1”;
  - Somatório $b_i * x^i$ com i = [0 .. (n-1)]
  - b_i é o coeficiente do bit na posição “i”
  - x^i indica o literal do bit na posição “i”

- e.g., … considere a sequência de bits “10011”.
  - Como esta sequência pode ser representada por polinômio ?

“$1*x^4 + 0*x^3 + 0*x^2 + 1*x + 1$” ou “$x^4 + x + 1$”

… 3.9 – Cyclic Redundancy Checks

- Sejam as operações binárias (ou-exclusivo):
  - 0 + 0 = 0 - 0 = 0
  - 0 + 1 = 0 - 1 = 1
  - 1 + 0 = 1 - 0 = 1
  - 1 + 1 = 1 - 1 = 0

- … não há “carry bit” na adição e nem “borrow bit” na subtração, ou seja, para todo “i” … $x^i + x^i = 0$
- …. para multiplicar 02 códigos de dados, basta multiplicar os polinômios correspondentes aos códigos de dados.

… 3.9 – Cyclic Redundancy Checks

- Dado codificado em 4 bits (p.ex., $x^2 + 1$) é multiplicado por "x + 1"
- Código gerado é o de paridade cuja "code rate" de ¾.
- É também um código cíclico, mas não é simétrico.

Table 3.4 — A Cyclic Code

Data Word	Polynomial	Multiplied By	Produces	Code Word
0 0 0	0	$x+1$	0	0 0 0 0
0 0 1	1	$x+1$	$x+1$	0 0 1 1
0 1 0	x	$x+1$	x^2+x	0 1 1 0
0 1 1	$x+1$	$x+1$	x^2+1	0 1 0 1
1 0 0	x^2	$x+1$	x^3+x^2	1 1 0 0
1 0 1	x^2+1	$x+1$	x^3+x^2+x+1	1 1 1 1
1 1 0	x^2+x	$x+1$	x^3+x	1 0 1 0
1 1 1	x^2+x+1	$x+1$	x^3+1	1 0 0 1

… 3.9 – Cyclic Redundancy Checks

- e.g., Calcule o polinômio cuja primitiva é “1 0 0 1 1” e o fator (polinômio gerador) é “1 1 0 0”

  “1 0 0 1 1” → $x^4 + x + 1$

  “1 1 0 0” → $x^3 + x^2$

  $(x^4 + x + 1) * (x^3 + x^2) = x^7 + x^6 + x^4 + x^3 + x^3 + x^2$

  … $x^7 + x^6 + x^4 + \underline{x}^3 + \underline{x}^3 + x^2$ → $x^3 + x^3$ é 0 …

  $(x^4 + x + 1) * (x^3 + x^2) = x^7 + x^6 + x^4 + x^2$

… 3.9 – Cyclic Redundancy Checks

- “fator” - utilizado para gerar o “checksum”, também é denominado Polinômio Gerador (Generator Polynomial) do código.
  - … multiplica-se o polinômio da primitiva por um termo igual à parcela de mais alto grau do polinômio gerador de modo que a primitiva seja deslocada para a esquerda tantos bits quantos do “checksum”;
  - … na sequência, divide-se o polinômio da primitiva pelo polinômio gerador ou fator obtendo-se um quociente e o resto.

- Como o CRC é um código linear, todo padrão de erro E deve ser igual a alguma palavra código T;
  - … para um código conhecido esta propriedade pode ser usada para calcular a taxa residual de erro.

… 3.9 – Cyclic Redundancy Checks

- “Controle de Erro” - tem o objetivo de diminuir a taxa de erro de um determinado canal de comunicação;
  - … contudo, nem todos os erros podem ser detectados, e então sempre existe uma ‘Taxa de Erro Residual’ (RER – “Residual Error Rate”);

- Suponha-se que a probabilidade de erros de transmissão em uma msg. seja “p”, e que o método de controle de erro identifique uma fração “f” deste conjunto de erros, então:
  - RER = p * (1 – f) … esta equação implica que, por instância, a taxa residual de erros é da ordem de 10^{-9} (10^-9) ou menos.

… 3.9 – Cyclic Redundancy Checks

- Seja P - polinômio da primitiva e G - polinômio gerador de grau "r", então o resto da divisão é polinômio R com grau "r-1", onde:

$$R = P * x^r / G$$

- Palavra Código "$T = P * x^r – R$"

- Um erro de transmissão adiciona um polinômio E ao código … e o receptor descobre o erro por:

$$(T + E) / G = T/G + E/G = E/G$$

… 3.9 – Cyclic Redundancy Checks

- Um erro de comunicação é indetectável se E/G = 0
  - … se E é diferente de zero e de grau menor que G, a divisão sempre tem resto, ou seja, todas as rajadas menores ou iguais a “r” são detectáveis.
  - ... posição dentro da primitiva T “code word” onde ocorre a rajada de erro é irrelevante para detecção do erro;

- Padrão de Erro “E” não se transforma em um múltiplo de G pela simples multiplicação de um fator x^i (obviamente para $x^i \neq G$).
  - … como a ocorrência do erro é aleatória, então para um código de comprimento “n+r” bits é possível calcular a probabilidade de ocorrência de erro (independente da posição).

… 3.9 – Cyclic Redundancy Checks

- Existem “2^(n + r)” possibilidades de erros, sendo que o nro. de múltiplos inteiros de um polinômio gerador de grau “r” em uma palavra de código é “2 ^ n”
- … cada múltiplo pode ser considerado como uma soma finita de “n” fatores, no qual cada fator é obtido por um deslocamento à esquerda do polinômio gerador.
- $2^n / 2^{(n+r)} = 1 / 2^r$

- e.g., considere r = 16 $\rightarrow$ 10^{-5} de todos os erros.

… 3.9 – Cyclic Redundancy Checks

- Projetar polinômios geradores para detectar a maior classe possível de erros de comunicação não é uma “tarefa fácil”.
  - … um bom polinômio gerador tem pelo menos um fator (x + 1), além disso, em sistemas distribuídos isto deve ser compartilhado com todos os pares.

- CRC-12 é um polinômio largamente utilizado, para gerar soma verificação - “checksum” de grau 12

$$x^{12} + x^{11} + x^3 + x^2 + 1$$

- Obs.: … grau é 12, então este código pode detectar rajadas de erros de até 12 bits, não somente rajadas de 12 bits.

… 3.9 – Cyclic Redundancy Checks

- A CCITT (hoje ITU-T) recomenda polinômio gerador de grau 16.

$$x^{16} + x^{12} + x^5 + 1$$

  - … grau é 16, então este código pode detectar rajadas de erros de até 16 bits, não somente rajadas de 16 bits.

- Em aritmética binária, este código pode ser escrito como:

$$(x+1) * (x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1)$$

  - … pode-se ver que qualquer polinômio multiplicado pelo fator (x+1) tem um número par de parcelas (ie, bits não zero).
  - … verificar o resultado (nro par de parcelas) … !!!

… 3.9 – Cyclic Redundancy Checks

- Isto significa que todo “E” com um número ímpar de parcelas, produzido por um número ímpar de erros de bits é detectável;
- CRC-CCITT detecta
  - 100% em caso de 2 bits de erro;
  - 99,997% em caso de rajadas de 17 bits;
  - 99,998% em rajadas maiores que 17 bits.

- O polinômio utilizado pelo protocolo BSC (Binary Synchronous Communication) da IBM é muito próximo deste polinômio:

$$x^{16} + x^{15} + x^2 + 1$$

… 3.9 – Cyclic Redundancy Checks

- Comitê IEEE 802 padronizou um CRC-32 bits:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

- Codificação e Decodificação do “checksum” CRC exigem tempo e podem degradar o desempenho do autômato;
- … implementação é tipicamente feita em “hardware”, por motivos óbvios de necessidade de esforço computacional.

- Observe-se que o desempenho do algoritmos tem uma linearidade com o grau do polinômio gerador.