

Camada de Enlace

Controle de Erros

Tipos e Natureza dos Erros

- Um erro ocorre quando um bit é alterado entre a transmissão e a recepção
- Os erros podem ser:
 - Erros de bit Simples (Isolados)
 - Erros em Salvas (Rajadas)
- Erros tendem a acontecer em rajadas

Erros de bit Simples

- Ou erros de bit isolados
- Apenas um bit é alterado
- bits adjacentes não são afetados
- Ruído Branco

Erros em Salvas (**Burst**)

- Comprimento B
- Sequência contínua de B bits onde o primeiro, o último e um grande número de bits intermediários contém erro
- Ruído impulsivo
- Pode ser causado por perdas em sistemas **wireless**
- Maior para taxas de dados elevadas

Exemplo

- Dados transmitidos em blocos de 1000 bits, taxa de erros de 0,001 erro por bit (1 a cada 1000 bits)
- Erros independentes (isolados) comprometeria a maior parte dos quadros
- Erros em rajada (ex.: 100 bits seguidos) afetaria apenas um ou dois blocos em cada 100 (em média)

Natureza de Erros

- Erros em rajadas são mais comuns
- Erros simples ou isolados são mais destrutivos (em média) que os em rajada
- Por outro lado: erros em rajada são mais difíceis de se detectar e corrigir do que erros isolados

Detecção e Controle de Erros

- Meios de transmissão comumente utilizados são sujeitos a erros
- Ex.: sistema telefônico (**loops** locais) e meios de transmissão sem fio
- Obs.: Controle de erros envolve os tratamentos necessários uma vez que um erro foi detectado (mas não corrigido)

Detecção e Controle de Erros

- Duas possibilidades:
 - Detecção de erros: apenas **detecta**-se o erro, indicando a necessidade de retransmissão
 - Correção de erros: técnicas que permitem **detectar** e corrigir bits errôneos em um quadro recebido

Detecção e Controle de Erros

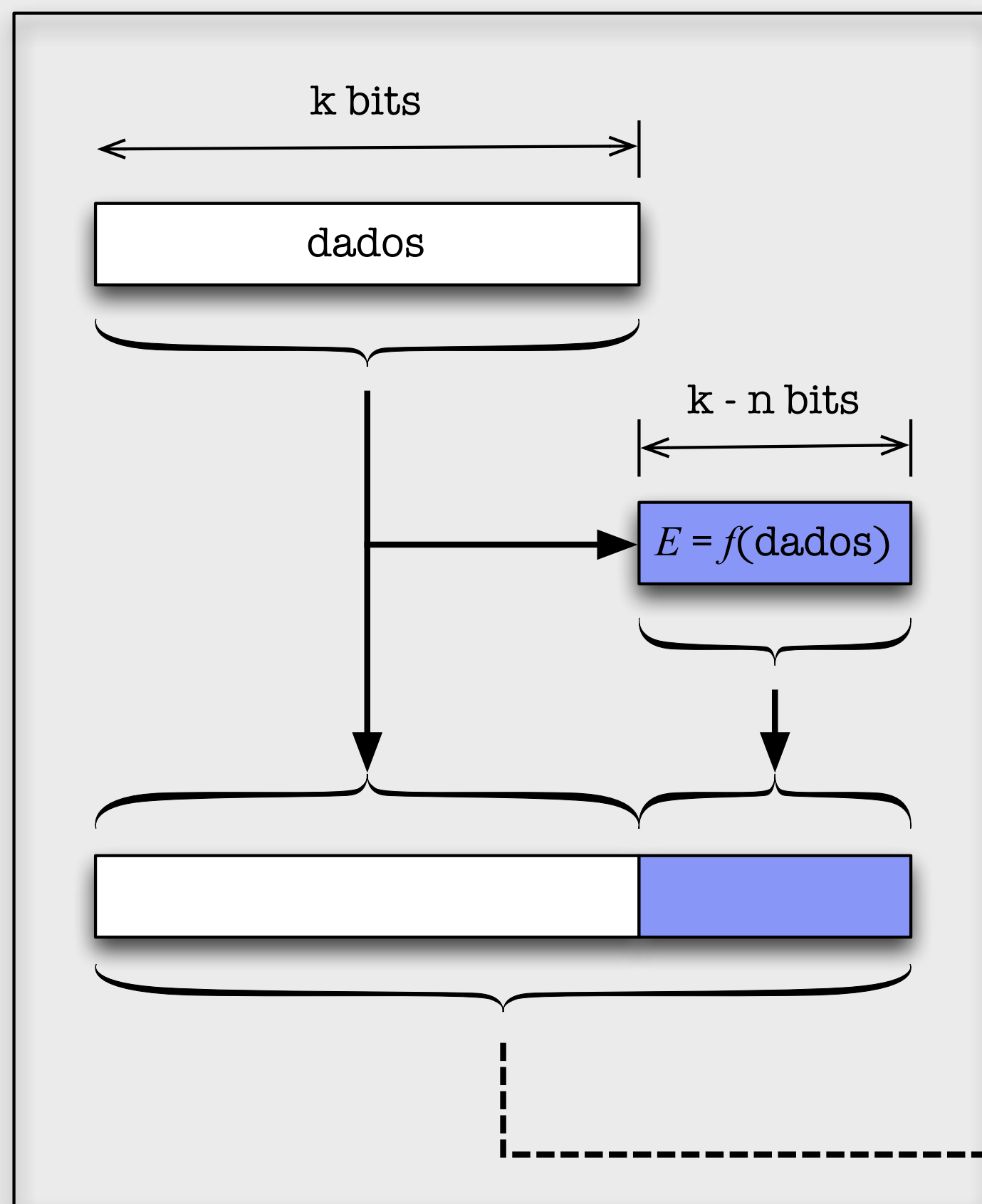
- Correção de erros é aplicável (prática) quando:
 - O canal de transmissão é simplex
 - Retransmissões não são possíveis
 - Os atrasos de transmissão são muito grandes
 - Ex.: conexões de satélite ou enlaces interplanetários
 - Taxa de erros é muito grande

Detecção e Controle de Erros

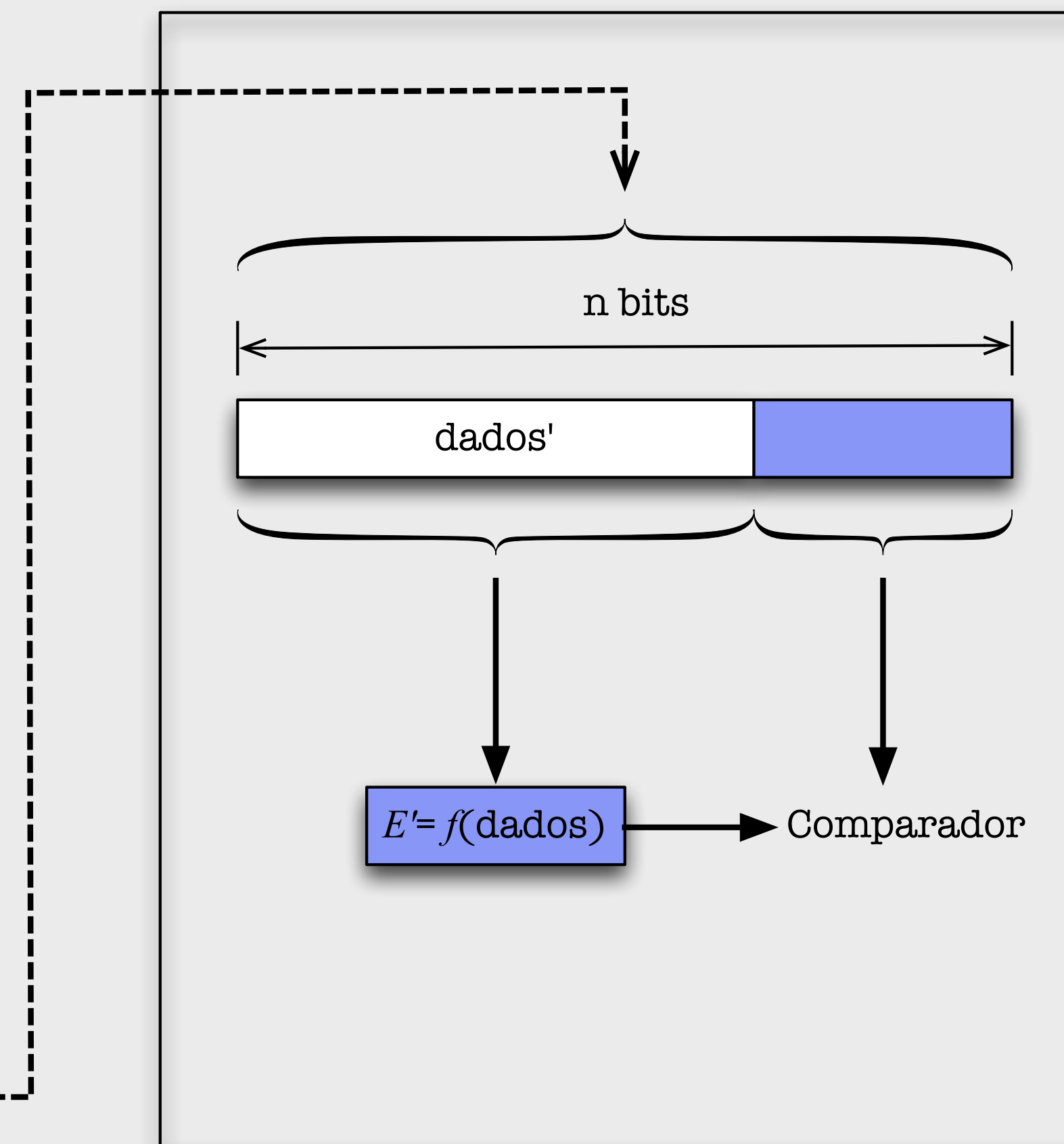
- Na maioria das situações comuns, contudo, detecção seguida de **retransmissão** é mais eficiente
- Em geral, detecção de erros e retransmissões geram menos bits de **overhead** do que códigos de correção de erros
- Assumindo que erros ocorrem esporadicamente

Diagrama do Processo de Detecção de Erros

Transmissor



Receptor



Detecção de Erros

- bits adicionais são acrescentados pelo transmissor formando o código de detecção de erros
- Paridade
 - Valor do bit de paridade é tal que o caractere tem sempre um número par (paridade par) ou ímpar (paridade ímpar) de uns
 - Número par de erros não pode ser detectado



bits de Paridade

- Confiabilidade 100% apenas nos casos de erros isolados
 - 1 bit para cada bloco protegido por um bit de paridade
- Para erros em rajada (vários bits no mesmo bloco):
 - Probabilidade de detecção do erro é de apenas 50%

bits de Paridade

- Melhoria:
 - Organizar os dados transmitidos em uma matriz (k linhas \times n colunas)
 - Última linha: bits de paridade
 - Paridade coluna-por-coluna
 - Cada bit de paridade checa uma posição de bit em cada linha
 - Capaz de corrigir rajadas de erros de até n bits

bits de Paridade

		Ordem de transmissão (uma linha de cada vez)							
									
$k = 7$ linhas		1	0	0	1	1	1	0	0
$n = 8$ colunas		0	1	0	0	0	1	1	1
Capaz de detectar surtos de erros de até 8 bits		1	1	0	0	0	0	1	1
		0	0	1	1	0	1	1	0
		1	1	1	0	0	0	1	1
		0	0	1	1	0	1	0	1
		1	1	0	0	1	0	1	1
bits de Paridade		0	0	1	1	0	0	1	1

01001011 00101100 11001010

010010110001011001110010100

001101010000111010101100010

001

101

010

000

111

010

101

100

010

Cyclic Redundancy Check (CRC)

- Para um bloco de k bits transmitidos, é gerada uma sequência de n bits
- Transmite $k+n$ bits os quais são perfeitamente divisíveis por algum número
- Receptor divide o quadro por aquele número
 - Se não há resto, assume que não haja erros

Códigos Polinomiais

- Checagem de Redundância Cíclica (CRC)
- Um quadro (seqüência de bits) a ser transmitido é visto como um polinômio $M(x)$ binário (i.e., com coeficientes 0 e 1 apenas)
- Ex.: $110001 \rightarrow x^5 + x^4 + x^0$ ou $x^5 + x^4 + 1$

Códigos Polinômiais

- Polinômio gerador: $G(x)$
- Utilizado para a geração de um **checksum** (CRC) a ser concatenado ao final de cada quadro original
- Polinômio resultante, $M(x) + \text{checksum}$, deve ser divisível por $G(x)$

Algoritmo de Geração

- Concatenar um número de bits “0” (equivalente ao grau r do polinômio gerador) ao final do quadro a ser transmitido
- Resultando no polinômio $x^r.M(x)$
- Dividir (módulo 2) o polinômio resultante por $G(x)$

Algoritmo de Geração

- Subtrair (módulo 2) o resto da divisão acima da sequência de bits correspondente ao polinômio $x^r.M(x)$
- O resultado é o quadro com **checksum** a ser transmitido

Exemplo de Códigos Polinomiais

- Quadro original: 1 1 0 1 0 1 1 0 1 1
- Gerador: 1 0 0 1 1 ($x^4 + x + 1 = \text{grau } 4$)
- Quadro após adicionar 4 bits:
1 1 0 1 0 1 1 0 1 1 0 0 0 0
- Resto da divisão: 1 1 1 0
 - (1 1 0 1 0 1 1 0 1 1 0 0 0 0 dividido por 1 0 0 1 1)
- Quadro transmitido:
1 1 0 1 0 1 1 0 1 1 1 1 1 0

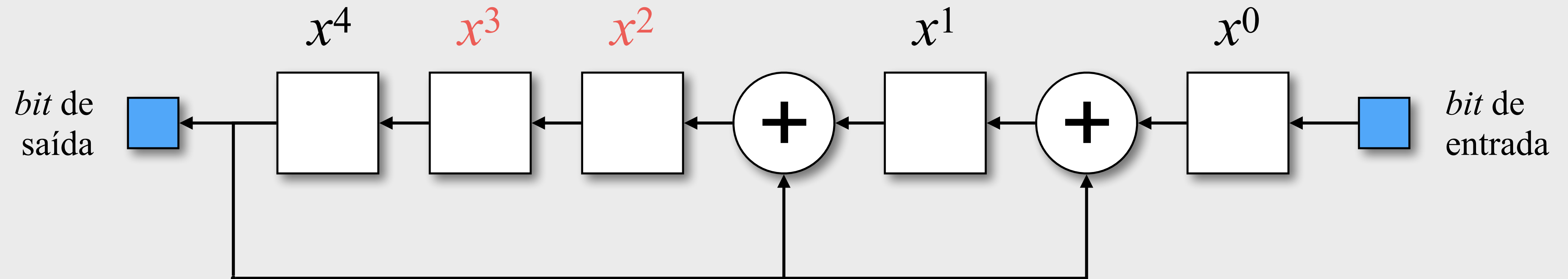
Resto da divisão: 1 1 1 0

	1	1	0	1	0	1	1	0	1	1	0	0	0	0		1	0	0	1	1
-	1	0	0	1	1															
	0	1	0	0	1	1														
	-	1	0	0	1	1														
		0	0	0	0	0														
							1	0	1	1	0									
						-	1	0	0	1	1									
							0	0	1	0	1	0	0							
								-	1	0	0	1	1							
									0	0	1	1	1	0						

Solução Prolixa

	1	1	0	1	0	1	1	0	1	1	0	0	0	0		1	0	0	1	1
-	1	0	0	1	1															
	0	1	0	0	1	1														
	-	1	0	0	1	1														
		0	0	0	0	0	1													
		!	1	0	0	1	1													
			0	0	0	0	1	0												
			!	1	0	0	1	1												
				0	0	0	1	0	1											
				!	1	0	0	1	1											
					0	0	1	0	1	1										
					!	1	0	0	1	1										
						0	1	0	1	1	0									
						-	1	0	0	1	1									
							0	0	1	0	1	0								
							!	1	0	0	1	1								
								0	1	0	1	0	0							
								-	1	0	0	1	1							
									0	0	1	1	1	0						

Solução por Hardware



Tipos de Erros Detectados

- Erros de um único bit (100%)
- Erros duplos, desde que $G(x)$ tenha pelo menos três bits 1
- Qualquer número ímpar de erros, desde que $G(x)$ contenha um fator $x + 1$

Tipos de Erros Detectados

- Qualquer surto de erros cujo comprimento (entre o primeiro e o último bits invertidos) seja menor que o comprimento do polinômio gerador
- A maior parte dos erros de rajada
- Probabilidade de não detectar: $1/2^r$

Códigos Polinômiais

- Implementação simples e eficiente por **hardware**
- Registradores de deslocamento e portas XOR

Códigos Polinomiais Padronizados

- CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
 - Transmissão de seqüências de caracteres de 6 bits, gera CRCs de 12 bits
- CRC-16: $x^{16} + x^{15} + x^2 + 1$
 - Transmissão de caracteres de 8 bits, gera CRCs de 16 bits
- CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
 - Idem (Europa)

Eficiência CRC-16 e CRC-CCITT

- Todos os erros de 1 ou 2 bits
- Todos os erros de um número ímpar de bits
- Todos os surtos de erros de até 16 bits
- 99,997% dos surtos de erros de 17 bits
- 99,998% dos surtos de erros de 18 bits ou mais

Correção de Erros

- Correção de erros detectados, usualmente requer a retransmissão do bloco de dados
- Não é apropriado para aplicações **wireless**
- Precisa corrigir os erros com base na seqüência de bits recebida

Retransmissão em Wireless

- Taxa de erros de bit é alta
 - Várias retransmissões seriam necessárias
- Atraso de propagação pode ser longos (satélite) comparado com o tempo de transmissão do quadro
- Poderia resultar na retransmissão do quadro com erros mais muitos outros quadros subsequentes

Códigos de Correção de Erros

- Transmissor inclui informação redundante o suficiente para permitir ao receptor deduzir quais foram os dados corretos transmitidos
- bits de redundância permitem determinar a posição do(s) bit(s) invertido(s)
- bits errôneos são corrigidos antes que os dados sejam repassados para a camada de rede

Processo de Correção de Erros

- Cada bloco de k bits é mapeado em um bloco de n bits ($n > k$)
- Palavra-código (**codeword**)
- Codificador para correção de erros em avanço — **Forward Error Correction** (FEC)
- Normalmente todos os k bits originais são incluídos na palavra-código

Processo de Correção de Erros

- Algumas FECs mapeiam os k bit de entrada em n bit da palavra-código onde os bits k originais não figuram entre eles
- Envio da palavra-código
- Cadeia de bits recebida é similar à transmitida mas pode conter erros

Processo de Correção de Erros

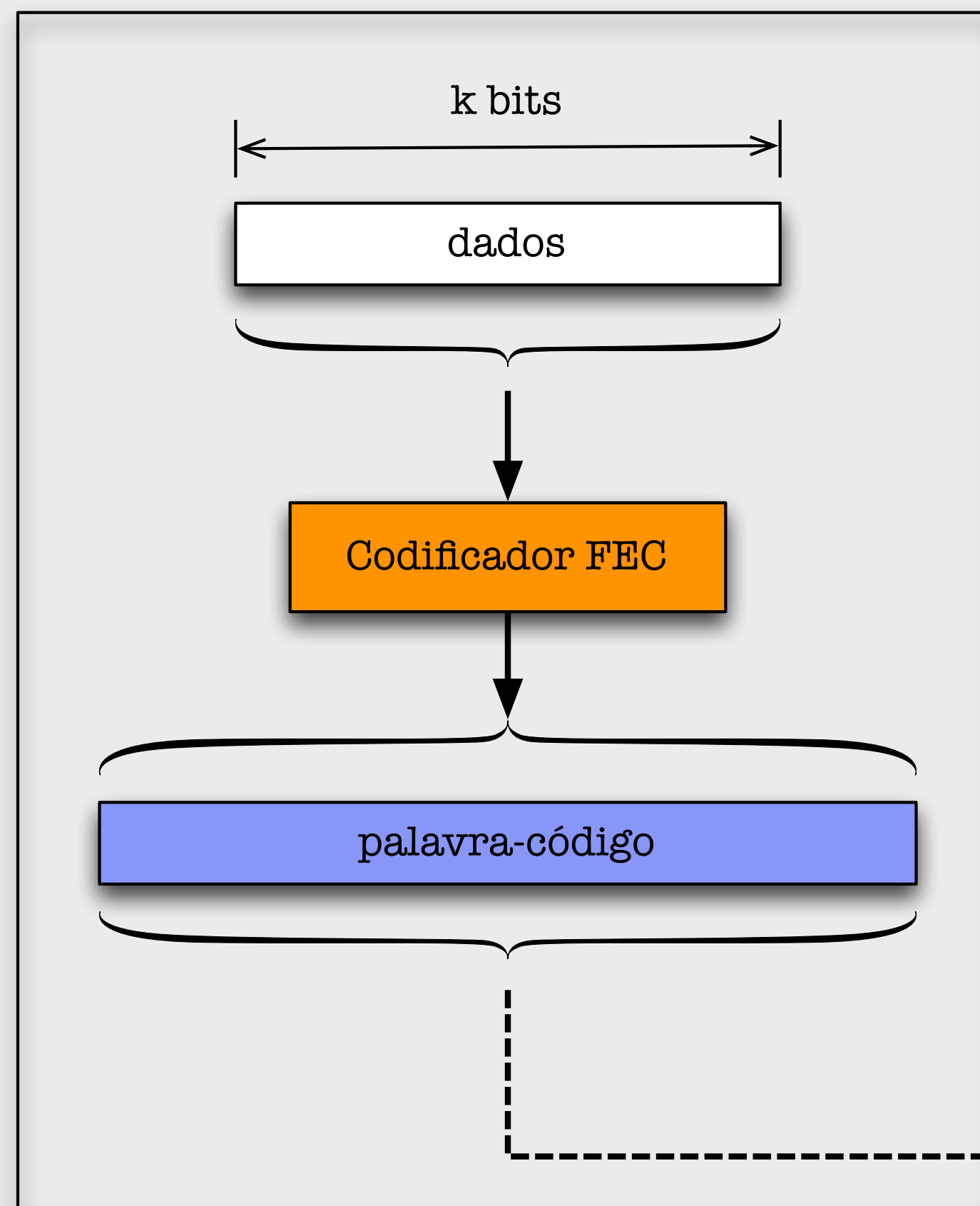
- Palavra-código recebida é passada para o decodificador FEC
- Se não contém erros, o bloco de dados original é obtido na saída
- Muitos padrões de erros podem ser detectados e corrigidos
- Alguns padrões de erros podem ser detectados, mas não corrigidos

Processo de Correção de Erros

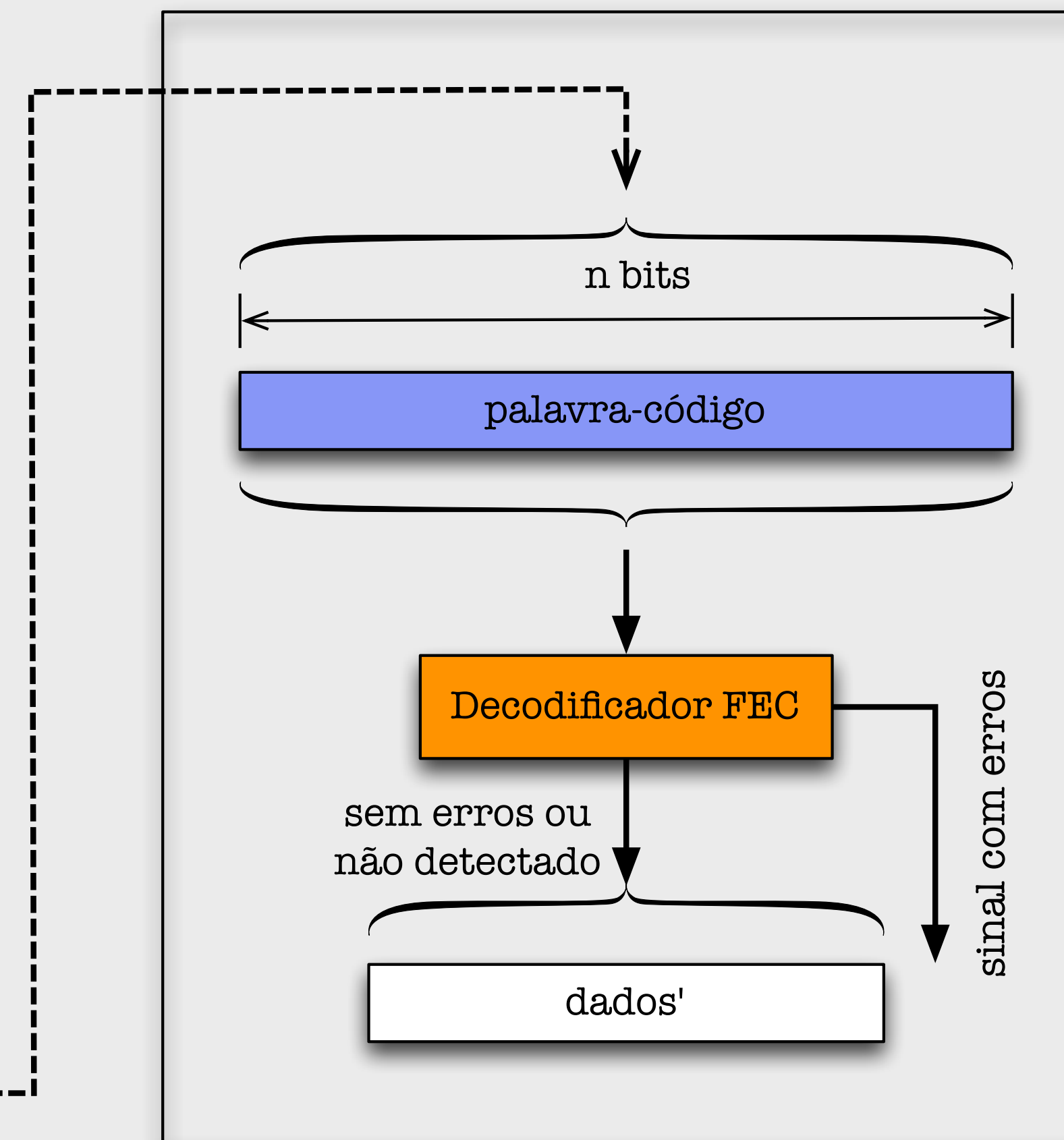
- Alguns (raros) padrões de erros não são detectados
- Resulta em uma saída de dados incorreta, não correspondendo aos dados originais
- Deve ser detectado por algum nível superior

Diagrama do Processo de Correção de Erros

Transmissor



Receptor



Distância de Hamming

- Distância de Hamming entre duas palavras-código:
 - Número de posições de bits nas quais as duas palavras diferem entre si
 - Ex.: 10001001 e 10110001: distância Hamming = 3
 - São necessários 3 erros de bit para transformar uma palavra na outra

Distância de Hamming

- k bits de dados: mensagem
- r bits de redundância: checagem
- Palavra-código (**codeword**): mensagem + bits de checagem
 - comprimento: $n = k + r$
- 2^k mensagens válidas
- Nem todas as 2^n palavras-código são válidas

Distância de Hamming

- Conjunto de todas as palavras-código: código
- Dado o algoritmo para computar os bits de checagem, é possível:
 - Enumerar todas as palavras código válidas
 - Encontrar as duas palavras-código cuja distância mínima (dentro do código) = Distância de Hamming do código em si

Distância Hamming

	Código com distância 3				Código com distância 2			
código	000000	010000	100000	110000	000000	010000	100000	110000
	000001	010001	100001	110001	000001	010001	100001	110001
	000010	010010	100010	110010	000010	010010	100010	110010
	000011	010011	100011	110011	000011	010011	100011	110011
	000100	010100	100100	110100	000100	010100	100100	110100
	000101	010101	100101	110101	000101	010101	100101	110101
	000110	010110	100110	110110	000110	010110	100110	110110
	000111	010111	100111	110111	000111	010111	100111	110111
	001000	011000	101000	111000	001000	011000	101000	111000
	001001	011001	101001	111001	001001	011001	101001	111001
	001010	011010	101010	111010	001010	011010	101010	111010
	001011	011011	101011	111011	001011	011011	101011	111011
	001100	011100	101100	111100	001100	011100	101100	111100
	001101	011101	101101	111101	001101	011101	101101	111101
	001110	011110	101110	111110	001110	011110	101110	111110
	001111	011111	101111	111111	001111	011111	101111	111111

Distância de Hamming

- As propriedades de detecção e correção de erros de um código dependem da sua distância de Hamming
- Para detectar d erros: código com distância $d + 1$
- Não há como d erros converterem uma palavra-código válida em outra palavra-código também válida

Distância de Hamming

- Para corrigir d erros: código com distância $2d + 1$
- Mesmo com d bits errôneos, a palavra-código original ainda estará mais próxima da palavra recebida do que qualquer outra
- Palavra correta pode ser deduzida unicamente

bit de Paridade

- Um bit adicional é adicionado ao bloco de bits a ser transmitido de forma que a soma total dos bits 1s seja par (ou ímpar)
- Exemplo: dados originais: 10110101
 - paridade par: 101101011
 - paridade ímpar: 101101010

bit de Paridade

- Distância de Hamming do código = 2
 - Apenas um erro de bit: gera uma palavra-código ilegal (com a paridade incorreta)
 - Permite detectar erros de um único bit

Correção de Erros

- Código com as seguintes palavras-código válidas:
 - 0000000000, 0000011111,
1111100000, 1111111111
- Distância de Hamming do código = 5
 - Capaz de corrigir erros duplos (dois bits)

Correção de Erros

- Ex.: palavra recebida: 0000000111
- Receptor deduz que a palavra correta é 0000011111
- Mas se o erro for de três bits, convertendo a palavra 0000000000 em 0000000111, o erro não será corrigido corretamente!
- Não há como ter certeza (pode-se apenas fazer suposições, com base em observações)

Aspectos Gerais de Projeto

- Correção de erros simples (de 1 único bit)
- Tamanho da mensagem: m bits
 - 2^m mensagens válidas
- Número de bits de checagem: r
 - Total de bits em uma palavra-código: $n = m + r$
 - 2^n palavras-código (i.e., padrões de bits possíveis)

Aspectos Gerais de Projeto

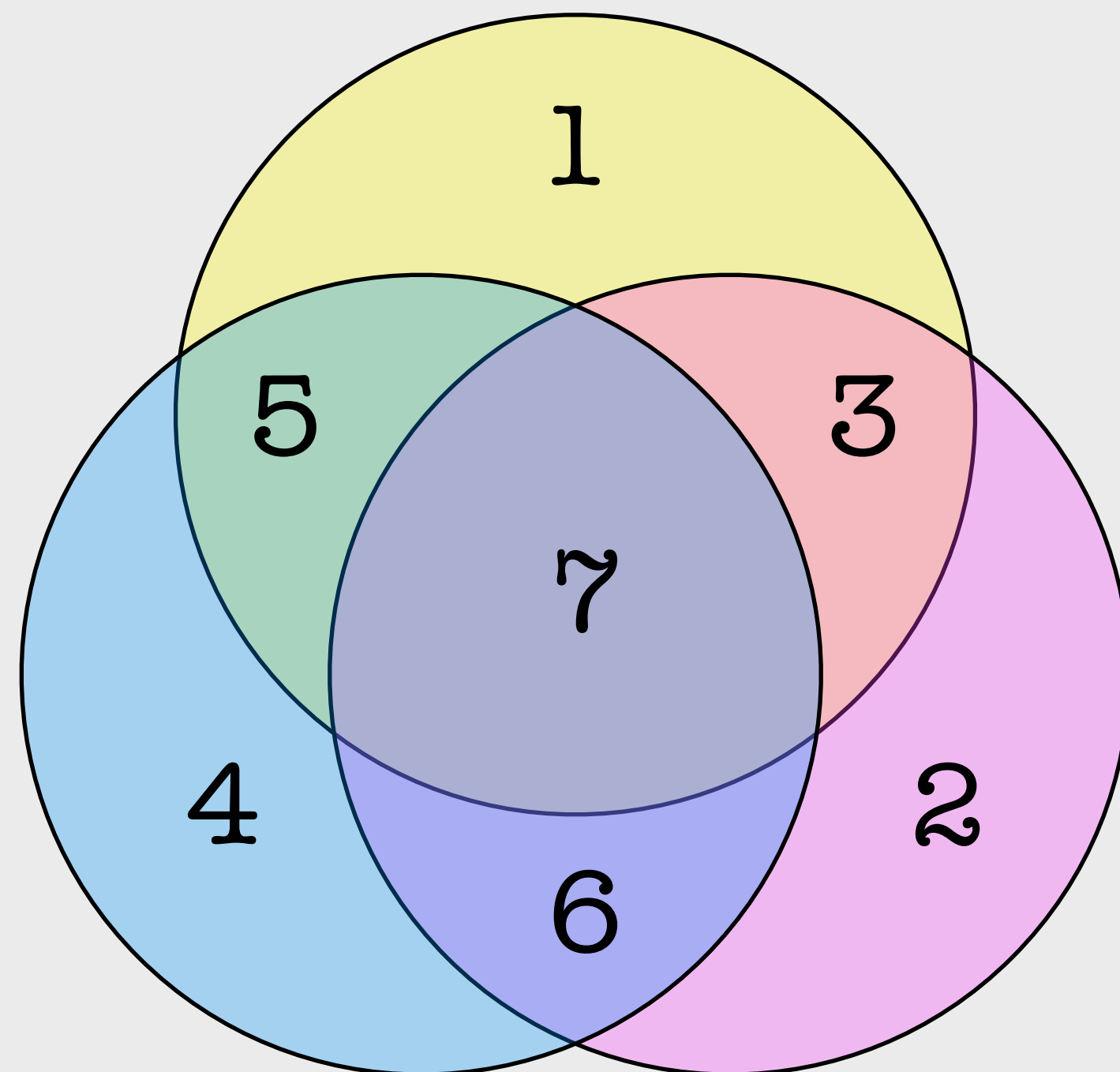
- Para cada palavra-código válida:
 - n palavras ilegais (a uma distância 1 da palavra válida)
 - Portanto, cada palavra válida requer $n + 1$ padrões de bits dedicados a ela
 - Logo, o número mínimo de **checkbits** necessários é dado por: $(n + 1) 2^m \leq 2^n$ ou $(m + r + 1) \leq 2^r$

Código de Hamming

- Numera-se os bits seqüencialmente
 - Começando com o bit 1 como o bit mais à esquerda
- bits numerados como potências de 2 (ex.: 1, 2, 4, 8, 16, etc.) representam os r **checkbits**
- Demais bits (3, 5, 6, 7, 9, etc.) representam os m bits de dados

Código de Hamming

- Cada **checkbit** determina a paridade de um sub-conjunto dos bits da palavra-código
- O mesmo **checkbit** pode estar envolvido na paridade de vários sub-conjuntos de bits



Código de Hamming

- Para determinar os **checkbits** que fazem a verificação de um determinado bit de dados na posição k :
- Reescrever k como uma soma de potências de 2

Código de Hamming

- Exemplos:
 - $k = 11 = 1 + 2 + 8$
 - bit 11 é checado pelos bits 1, 2 e 8
 - $k = 29 = 1 + 4 + 8 + 16$
 - bit 29 é checado pelos bits 1, 4, 8 e 16
- Transmissor calcula cada **checkbit** e os insere na palavra-código a ser transmitida

Código de Hamming

- Ao receber uma palavra-código, o receptor
 - Inicializa um acumulador (em zero)
 - Examina cada **checkbit** k ($k = 1, 2, 4, 8, \dots$) para determinar se o mesmo tem a paridade correta
 - Se paridade do **checkbit** k está incorreta:
 - Adiciona k ao acumulador

Código de Hamming

- Se, ao final, o valor do acumulador for zero
 - Não houve erro na palavra recebida
- Se o valor do acumulador for diferente de zero:
 - Acumulador contém o número do bit errôneo

Código de Hamming

– Exemplo –

- Blocos de dados transmitidos: 7 bits
- 4 **checkbits**: 1, 2, 4, 8
- bits de dados: 3, 5, 6, 7, 9, 10, 11

<i>bit</i>	Checado por
3	1, 2
5	1, 4
6	2, 4
7	1, 2, 4
9	1, 8
10	2, 8
11	1, 2, 8

1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	----	----

Código de Hamming

– Exemplo –

- Checkbits 1, 2 e 8 estão com paridade incorreta
- bit 11 foi invertido (é o único bit checado pelos bits 1, 2 e 8)

Correção de Surtos de Erros

- Grupo de k palavras-código a serem transmitidas
 - Arranjadas como uma matriz $k \times n$
- Transmitir os dados coluna por coluna
- Uma rajada de erros de até k bits afetaria, no máximo um bit em cada palavra-código

Correção de Surtos de Erros

- Código de Hamming em cada palavra-código seria usado para corrigir cada erro individual
- Resultado: múltiplos ($\leq k$) erros consecutivos corrigidos com $k \times r$ **checkbits**

Código de Hamming

– Exemplo –

Caractere	ASCII	Código
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	11111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	00101011111
d	1100100	11111001100
e	1100101	00111000101

Ordem da transmissão dos bits

Controle de Erros

- Erros não-recuperáveis, i.e., que não podem ser corrigidos no receptor
- Deve ser maior, especialmente em serviços orientados a conexão
- A camada de enlace deve tratar os problemas decorrentes de erros em bits nos quadros

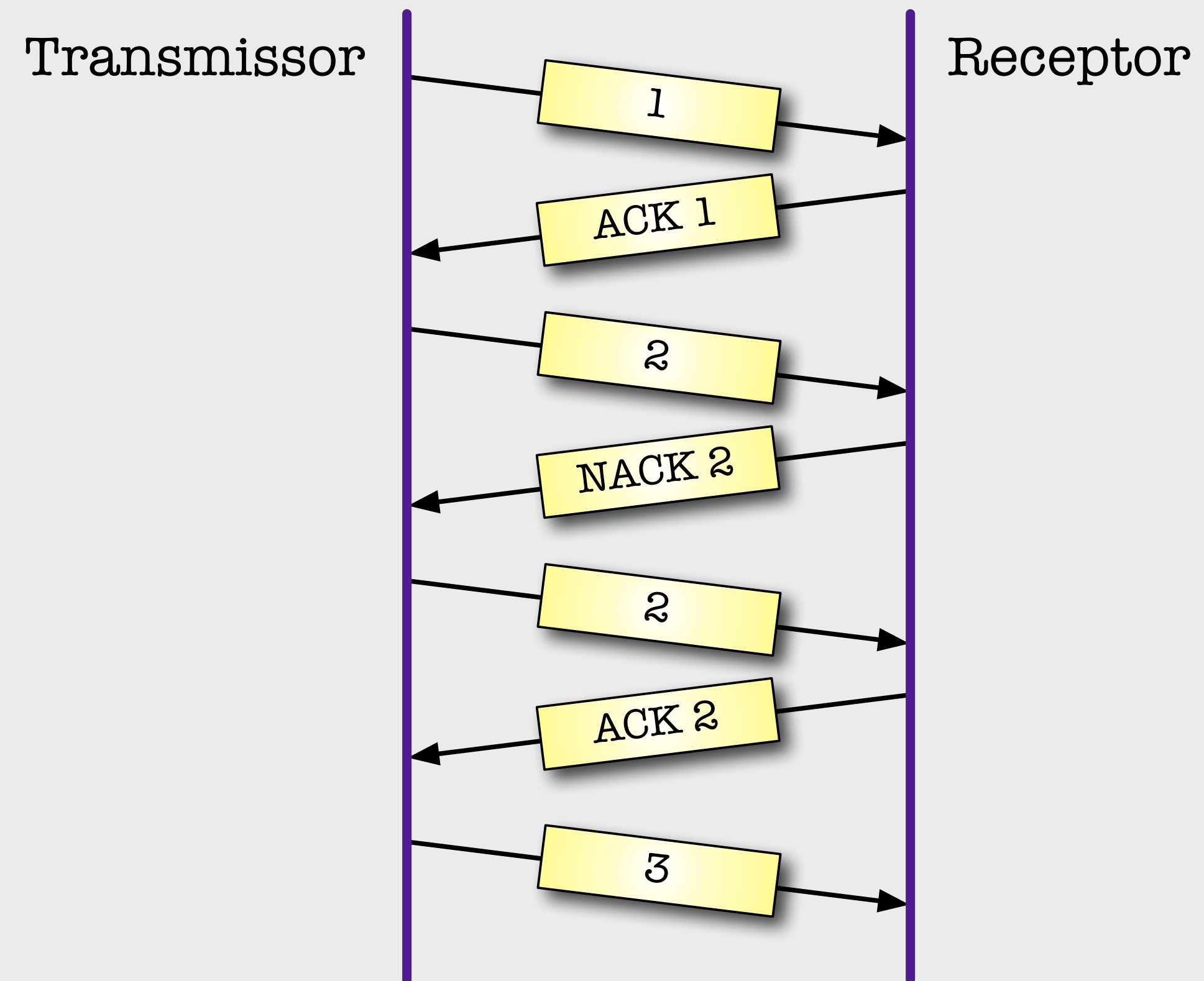
Controle de Erros

- Quadros perdidos
 - Ex.: devido a ruídos na transmissão
- Quadros recebidos com erros de **checksum** que não possam ser corrigidos
- Quadros recebidos fora de ordem
- Perda de quadros de reconhecimento
 - E, em consequência, a duplicação de quadros

Controle de Erros

- Tipos de Quadros de Reconhecimentos (**acknowledgement**)
- Receptor informa ao transmissor o estado do quadro recebido:
 - **ACK**: o quadro chegou sem problemas, prossegue normalmente
 - **NACK**: o quadro chegou, mas com erro, deve ser retransmitido

Controle de Erros



Controle de Erros

- Problema: Quadros perdidos devido a erros de transmissão
 - O quadro não é recebido de forma alguma
 - portanto, **NACK** não será enviado pelo receptor
- Transmissor poderia ficar bloqueado para sempre à espera do reconhecimento

Controle de Erros

- Solução: Uso de temporizadores
- Permite que o transmissor atribua um limite máximo (**timeout**) ao tempo que esperará por um reconhecimento de um quadro pelo receptor

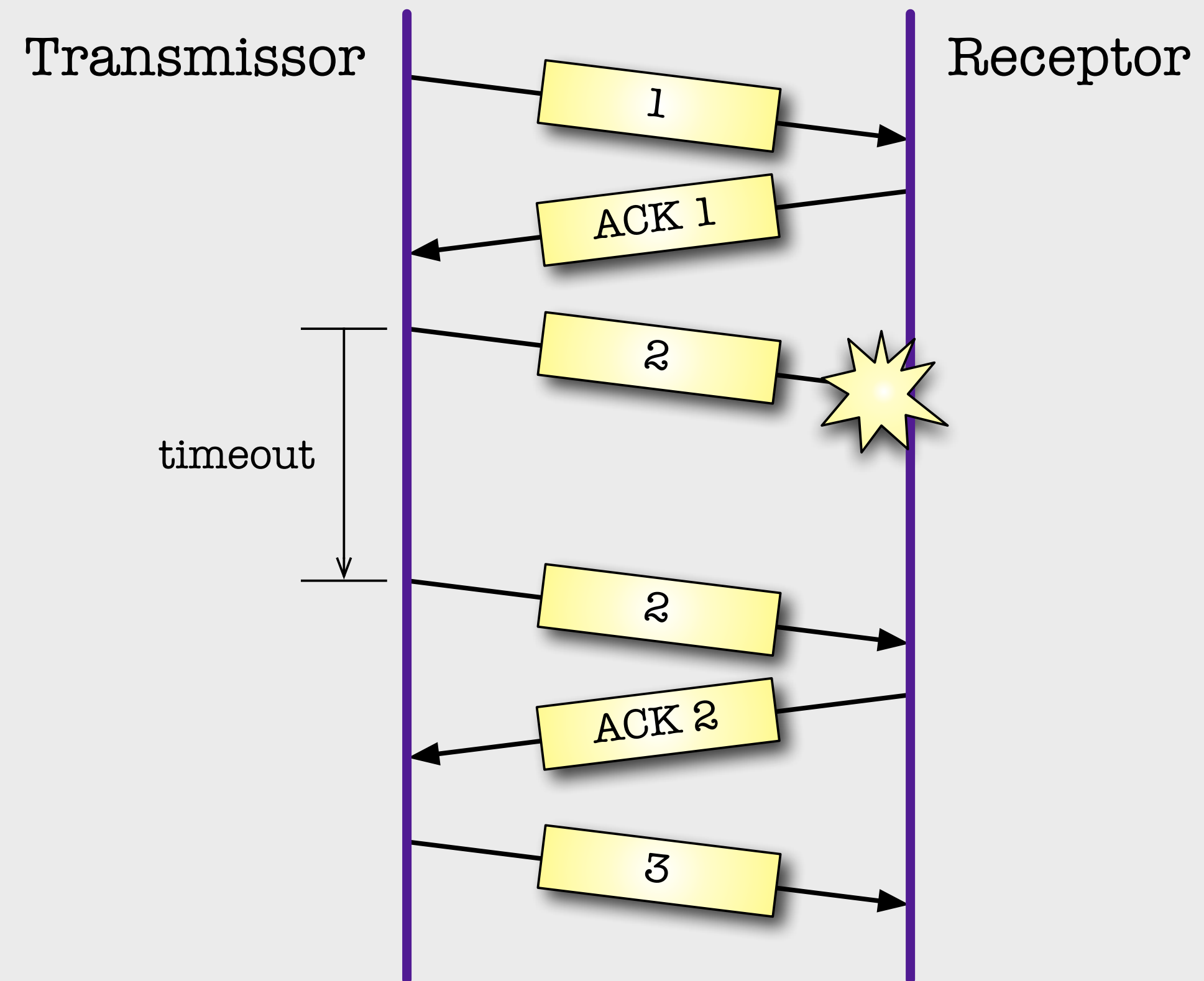
Controle de Erros

- Ao enviar um quadro o transmissor dispara um temporizador
- Alarme “soará” (**timeout**) após um tempo considerado suficiente para
 - o quadro se propagar e ser recebido do outro lado
 - o receptor processar o quadro
 - o reconhecimento propagar de volta até o remetente

Controle de Erros

- Caso o reconhecimento não chegue antes do **timeout** ocorrer
- Transmissor reenvia o quadro, assumindo que o mesmo não foi recebido do outro lado do enlace

Controle de Erros



Controle de Erros

- Mas e se um reconhecimento se perder?
 - Ocorrerá um **timeout**
 - Transmissor se comportará como se o quadro original houvesse sido perdido
 - Retransmitirá uma duplicata do quadro

Controle de Erros

- O mesmo quadro será ser processado duas (ou mais) vezes pelo receptor
- Cópias do mesmo pacote poderão ser passadas para a camada de rede como se fossem pacotes diferentes
- Podendo gerar resultados indesejáveis
 - Ex.: operações como saque em uma conta corrente

Controle de Erros

- Solução:
 - Associar números de seqüência aos quadros
 - Receptor saberia se um quadro já foi recebido, descartando duplicatas
- Esta solução é também válida para o problema da ordenação dos quadros:
 - Quadro só é repassado à camada de rede se a ordem estiver correta

Controle de Erros

