Deteção e Controlo de Erros

Fundamentos de Redes

Mestrado Integrado em Engenharia de Computadores e Telemática

DETI-UA, 2018/2019

CONTROLO DE ERROS

Introdução

- **Problemas no canal de comunicação**
  - **Pacotes corrompidos (recebidos com erros)**
  - **Pacotes perdidos**
  - **Pacotes recebidos fora de ordem**

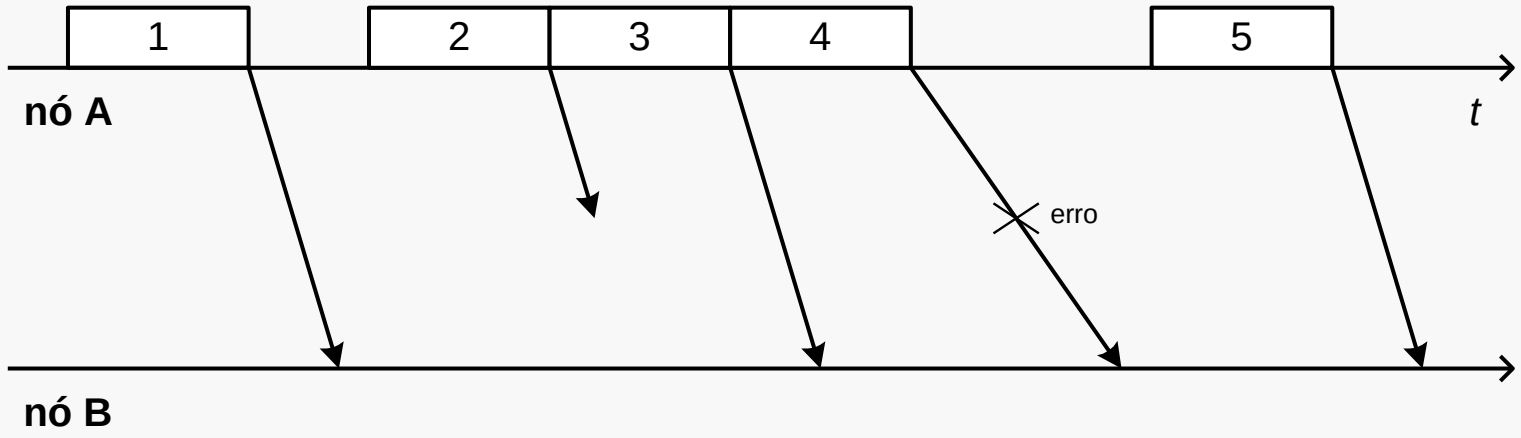
- **Controlo de erros**
  - **O emissor e o recetor serem capazes de coordenar entre si para que os pacotes perdidos (ou recebidos com erros) sejam reenviados**
  - **Protocolos de retransmissão ARQ (*Automatic Repeat reQuest*)**

Controlo de erros

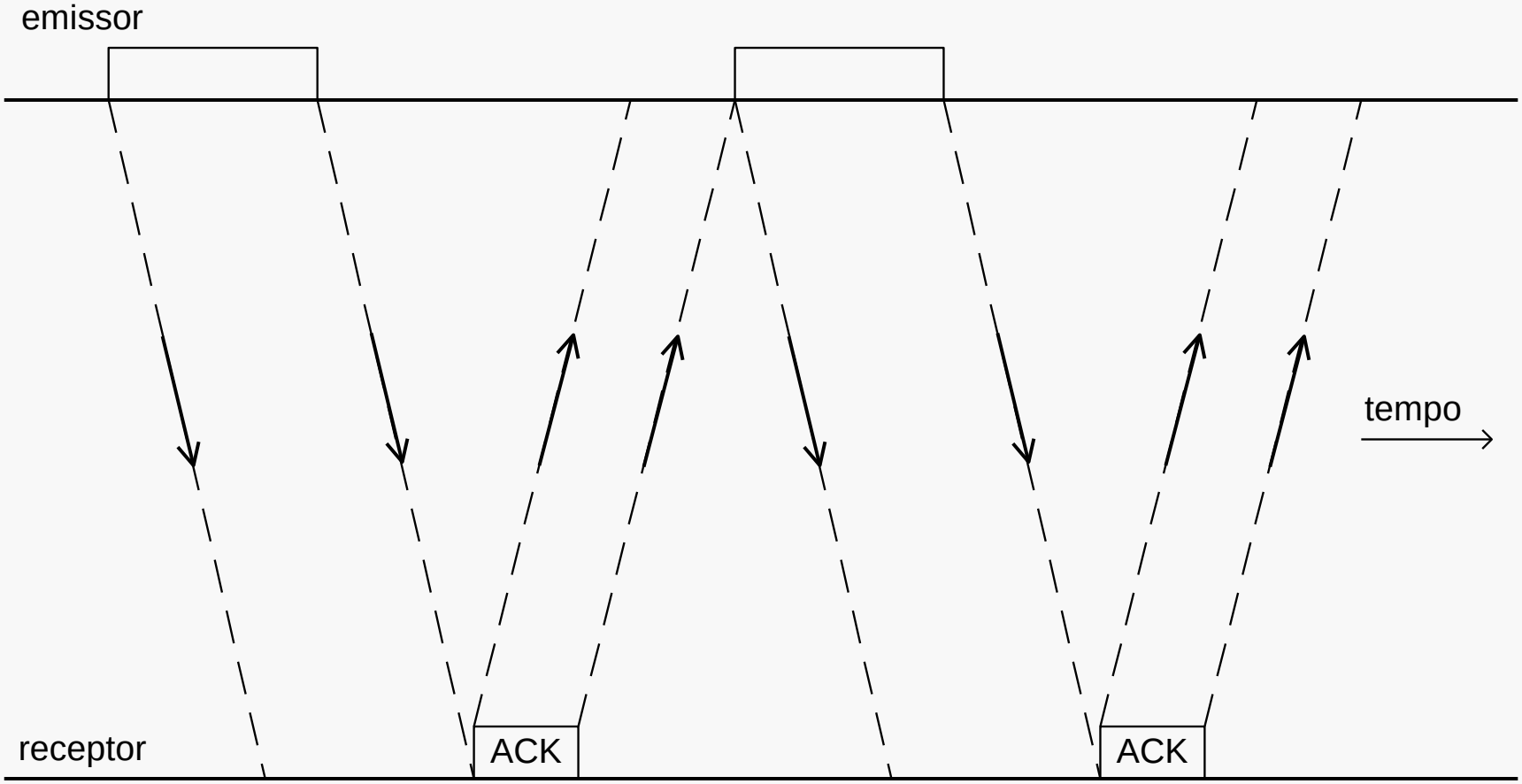
- **Os protocolos de retransmissão ARQ podem ser necessários em diferentes camadas protocolares**
  - **Camada da ligação de dados (exemplo: HDLC)**
  - **Camada de transporte (exemplo: TCP)**
  - **Camada das aplicações (exemplo: TFTP)**
- **A explicação seguinte irá ser conduzida admitindo que os protocolos são implementados na camada da ligação de dados**
  - **Pacote: conjunto de bits entregues para transmissão pela camada de rede**
  - **Trama: conjunto de bits entregues para transmissão à camada física**
  - **Quando são detetados erros numa trama, é transmitida um nova trama contendo o pacote anterior**

Controlo de erros

- **Assume-se:**
  - **Erros (das tramas) são sempre detetados**
  - **Tramas podem sofrer atrasos variáveis mas limitados**
  - **Algumas tramas podem ser perdidas**
  - **Tramas chegam na ordem em que foram transmitidas (pressuposto válido pois estamos a considerar a camada de ligação de dados)**

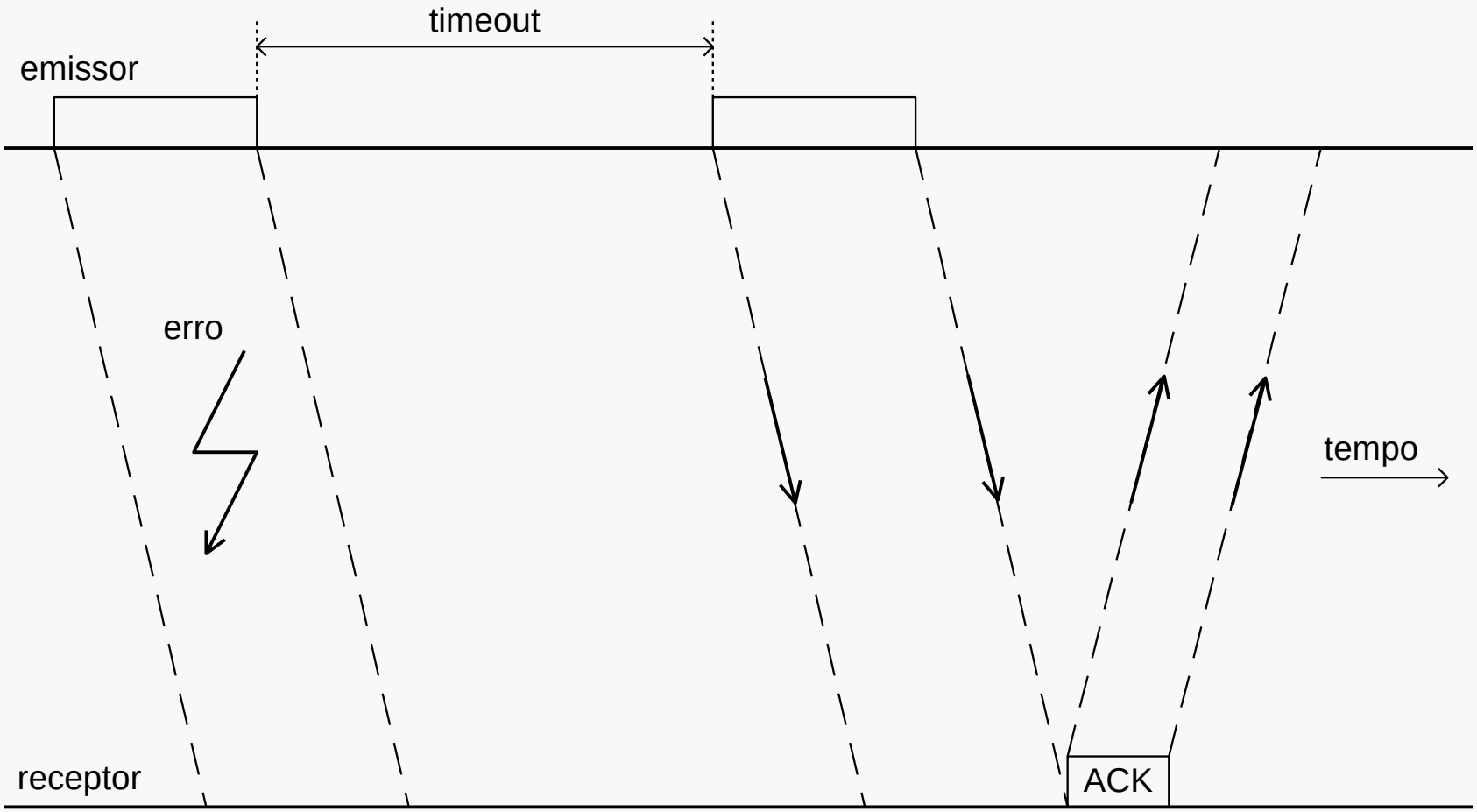


Stop-and-wait (SW)



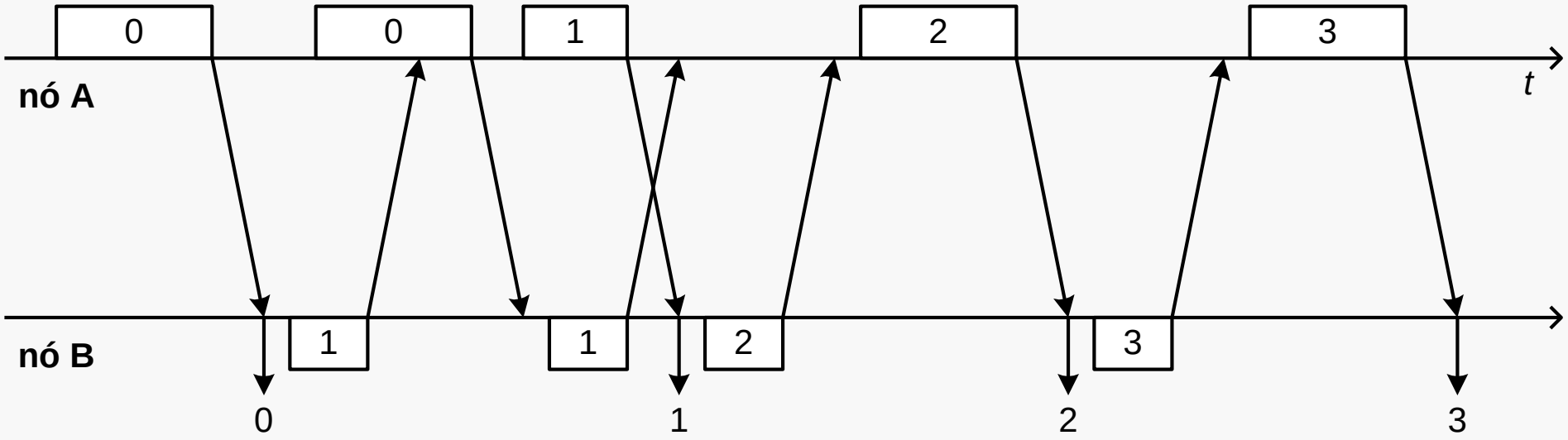
Operação sem erros

Stop-and-wait (SW)



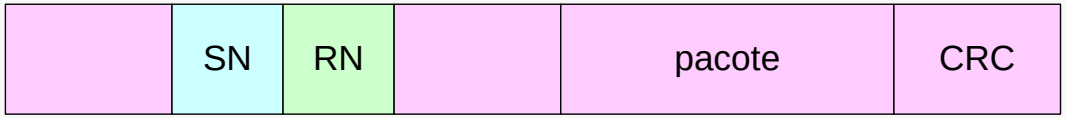
Recuperação de erros

Stop-and-wait (SW)

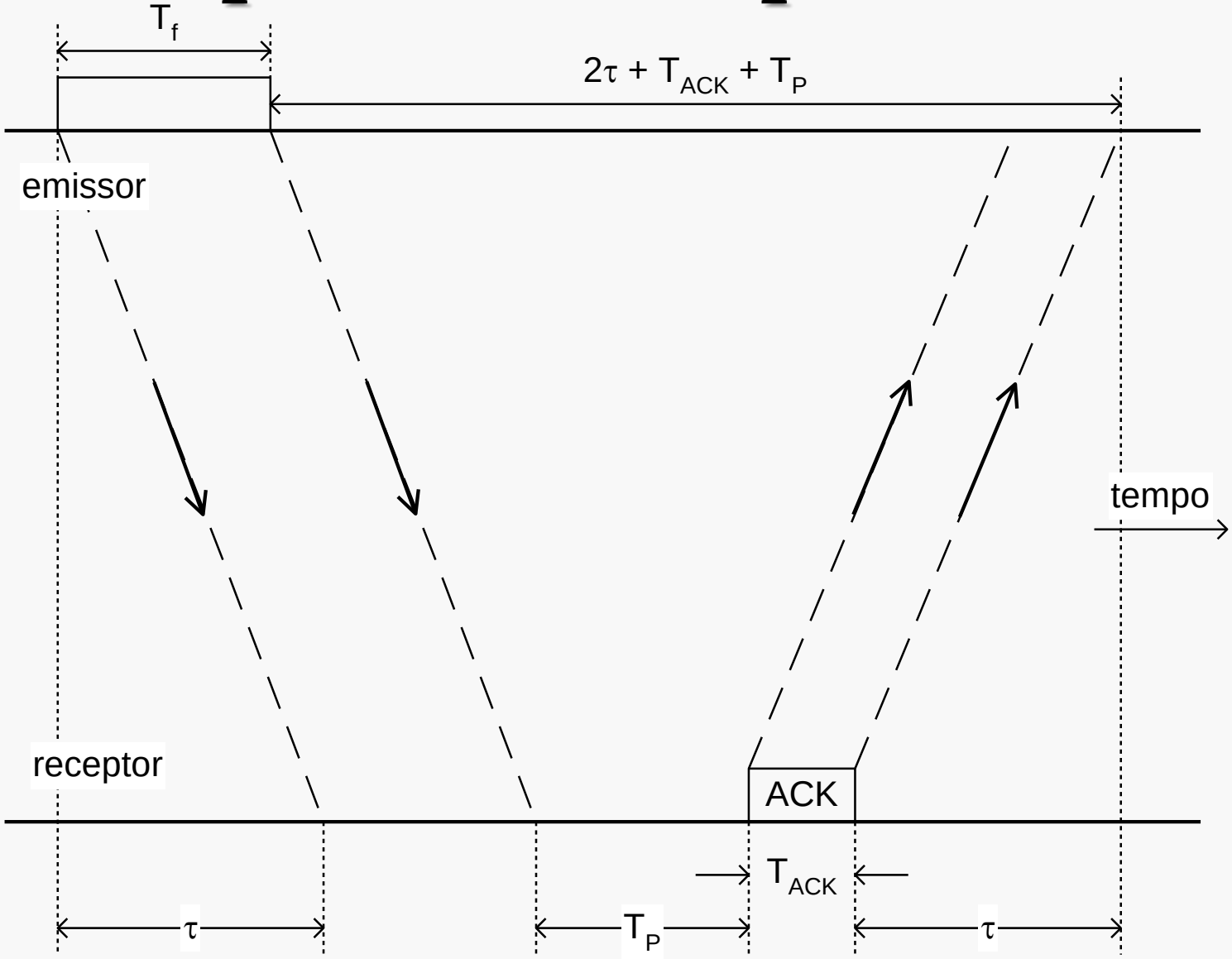


- SN (*Sequence Number*):
  - ❖ nº de sequência das tramas de dados enviadas pelo emissor (nó A)
- RN (*Request Number*):
  - ❖ nº de sequência da próxima trama esperada pelo recetor (nó B)

O RN pode vir encavalitado (*piggybacked*) em tramas de dados que estejam a ser enviadas em sentido contrário (do nó B para o nó A).



Desempenho do Stop-and-Wait



Desempenho do Stop-and-Wait

Utilização máxima sem erros:

$$S = \frac{T_f}{T_f + 2\tau + T_p + T_{ACK}}$$

T_f	tempo de transmissão de uma trama
T_{ACK}	tempo de transmissão de um ACK
τ	atraso de propagação entre emissor e recetor
T_p	tempo de processamento da trama no recetor.

Utilização máxima com erros:

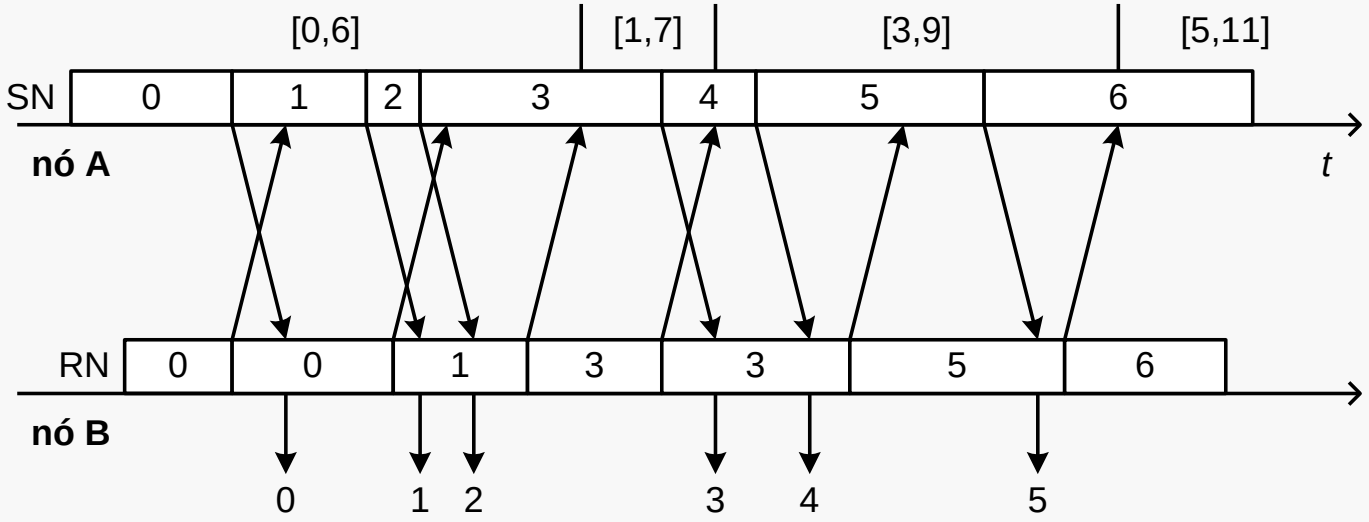
$$S = \frac{T_f}{\left(T + T_f\right)P/\left(1-P\right) + T_f + 2\tau + T_p + T_{ACK}}$$

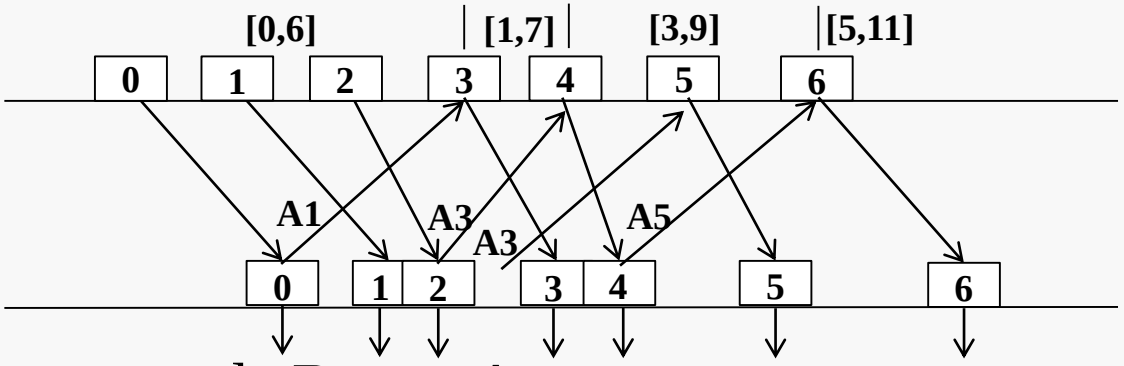
T	*timeout*
P	probabilidade de erro de transmissão de uma trama

Go-Back-N (GB-N)

- Ao emissor é permitido enviar mais do que uma trama antes de receber o ACK da primeira
  - Janela N – nº máximo de tramas que o emissor pode transmitir antes de receber o ACK da primeira
- Após um *timeout*, o emissor que não tenha recebido o ACK de uma trama *n*, retransmite essa trama e todas as seguintes
- Após receber a trama *n*, o recetor apenas aceita receber a trama *n*+1 e descarta a trama se tiver um número superior. Razões:
  - se uma trama não chegou ao destino, essa trama e as seguintes serão retransmitidas
  - simplicidade do recetor (não tem de gerir um buffer de receção)
- No envio de um ACK de uma trama *n*, está implícito que todas as tramas até *n* foram recebidas

Go-Back-N (GB-N)



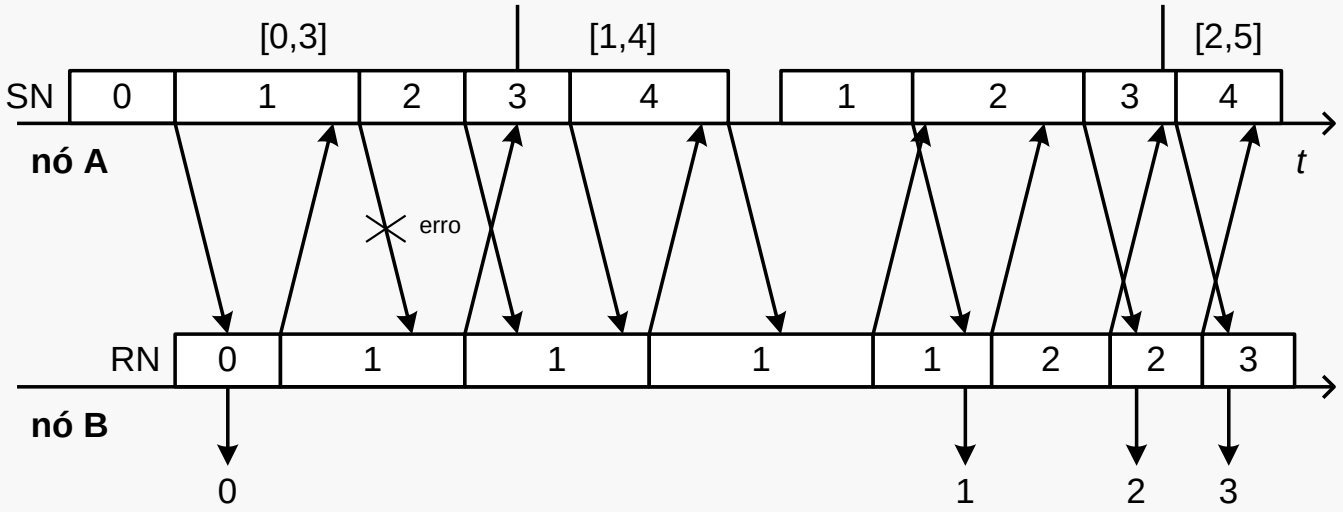


GB-7, dados de A para B:

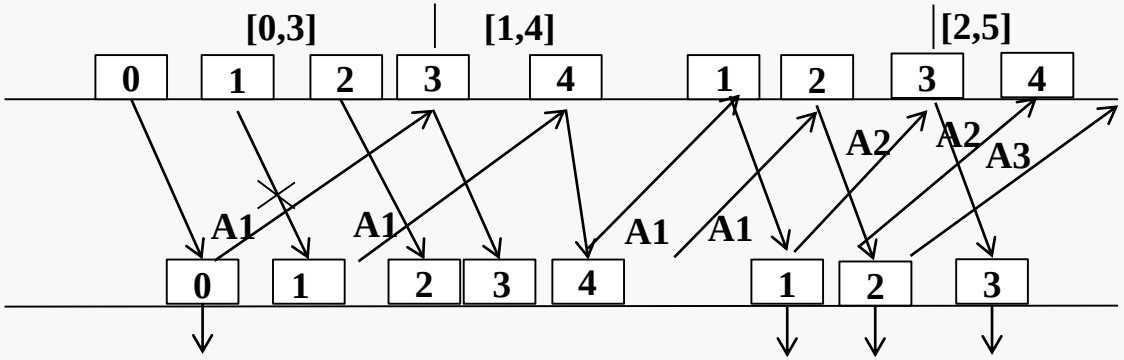
SN de A para B,

RN *piggybacked* em pacotes de B para A.

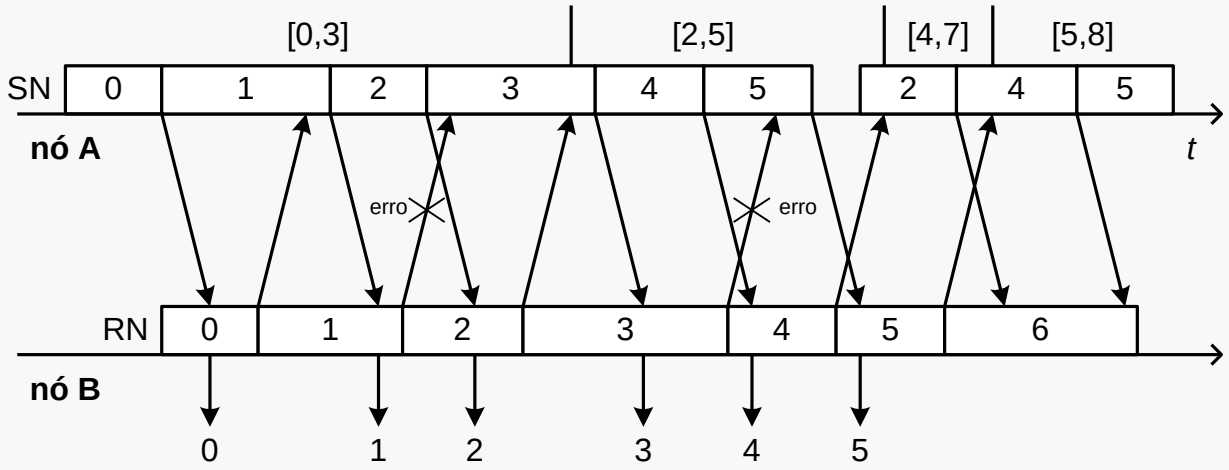
Go-Back-N (GB-N)



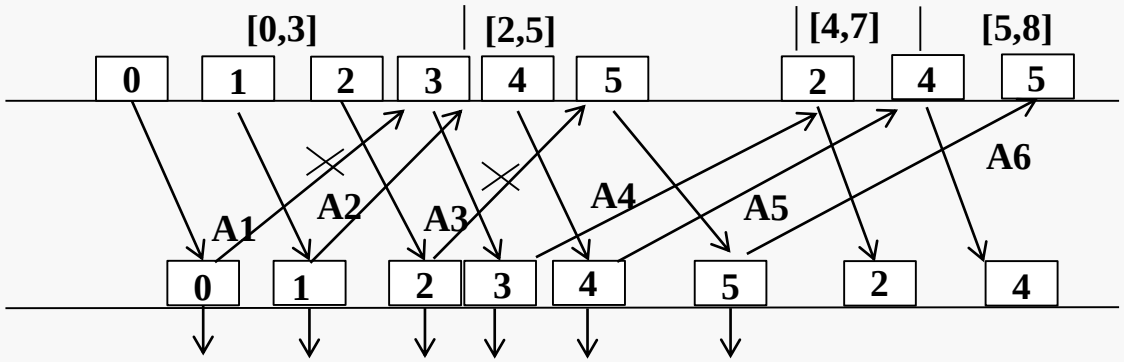
GB-4, erros de transmissão no sentido A→B



Go-Back-N (GB-N)



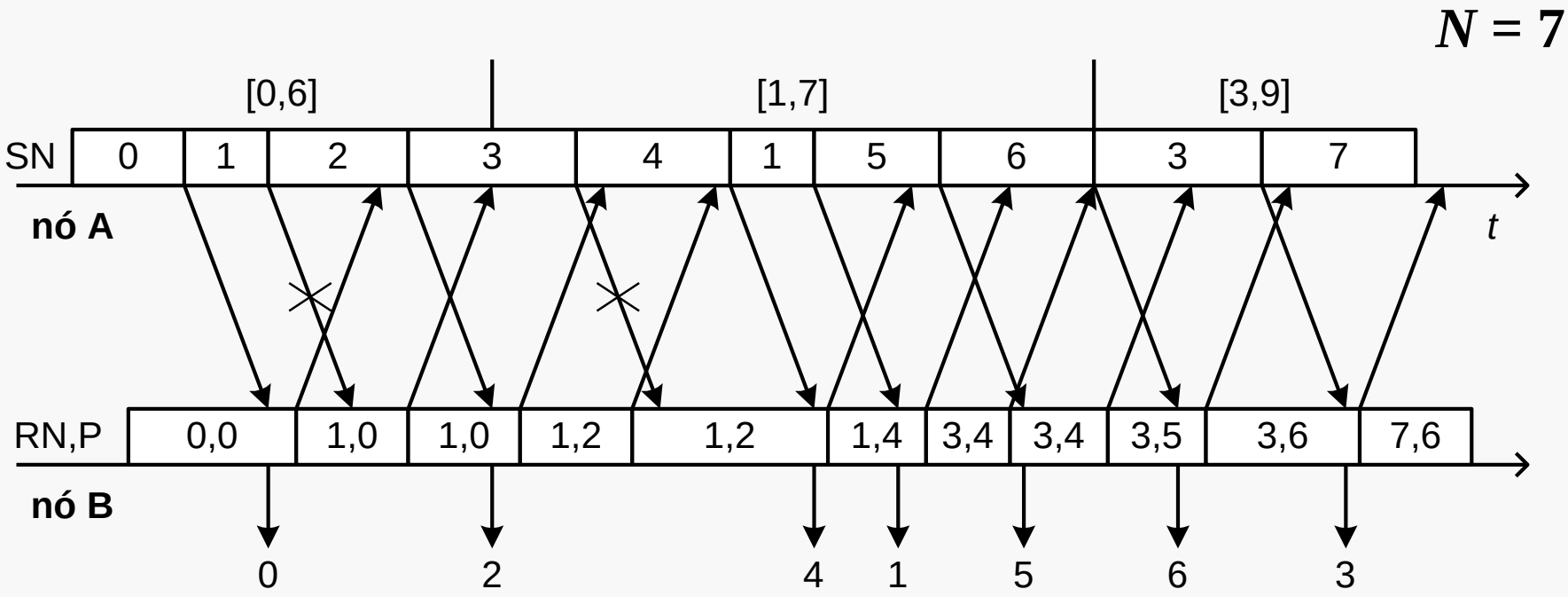
GB-4, erros de transmissão no sentido B→A



Selective Repeat (SR)

- Ao emissor é permitido enviar mais do que uma trama antes de receber o ACK da primeira
  - Janela *N* – nº máximo de tramas que o emissor pode transmitir antes de receber o ACK da primeira
- Após um *timeout*, o emissor que não tenha recebido o ACK de uma trama SN = *n*, retransmite apenas essa trama
- Após receber todas as tramas até ao SN = *n*, o recetor aceita receber qualquer trama cujo SN seja de *n*+1 até *n*+*N*
  - Em geral, apenas as tramas perdidas são retransmitidas
  - Assim, tramas podem ser recebidas fora de ordem mas o buffer de receção permite a reordenação pela ordem original
- O recetor especifica:
  - o RN que é igual a SN + 1
  - *P* - número de sequência da trama com SN mais elevado corretamente recebida

Selective Repeat (SR)



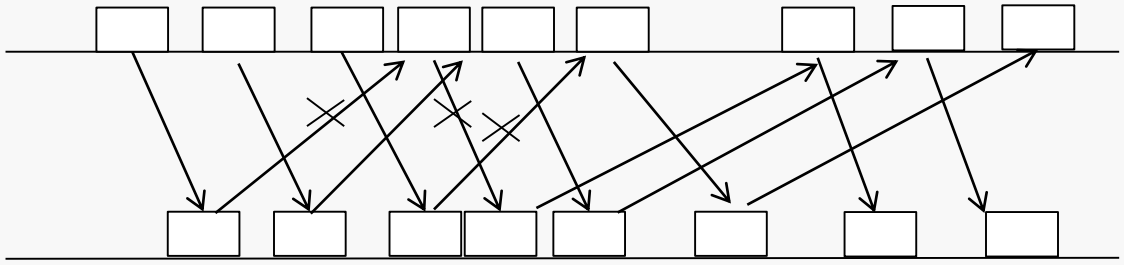
SR-7, dados de A para B:

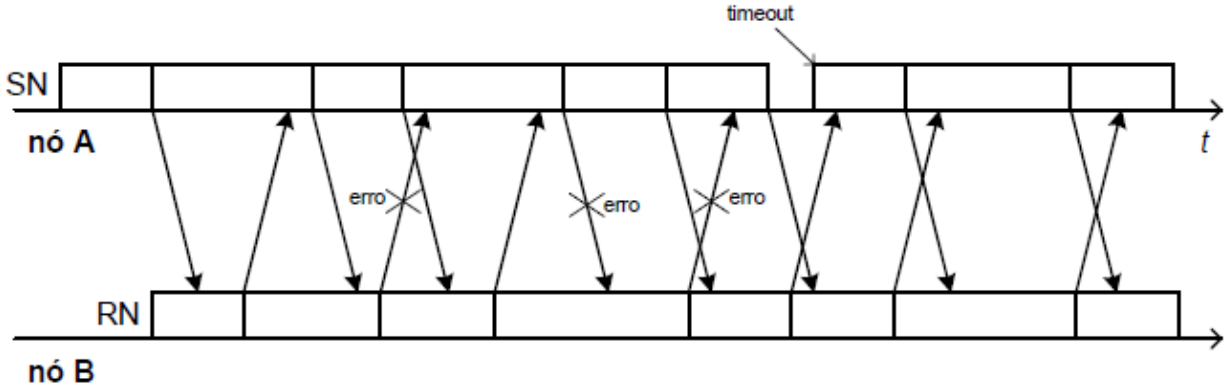
SN de A para B,

RN,P *piggypacked* em pacotes de B para A.

Exemplos

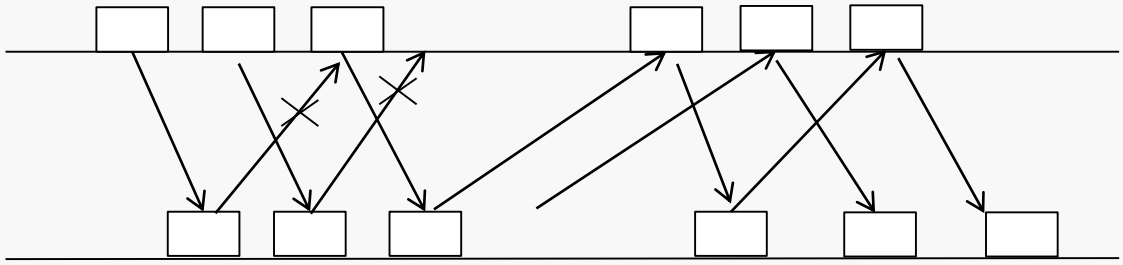
- **GB-4**

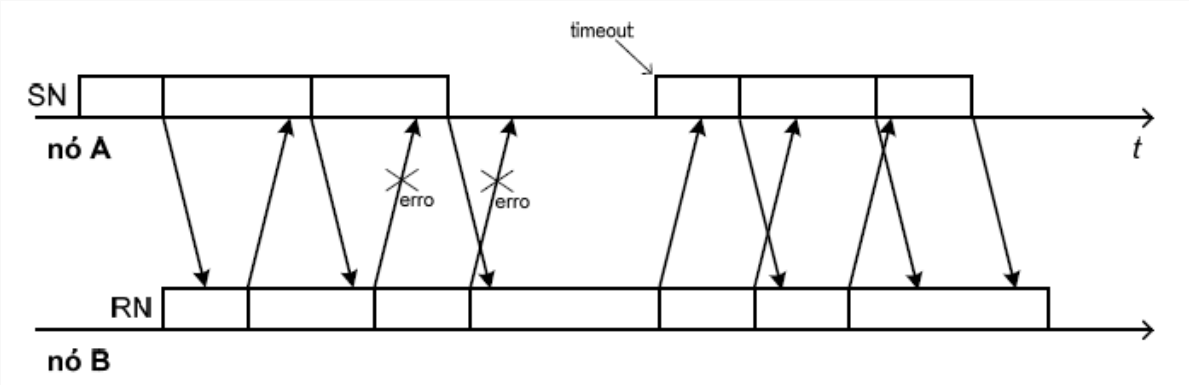




Exemplos

- **GB-3**





DETEÇÃO DE ERROS

Introdução

- **Problemas no canal de comunicação**
  - **Pacotes corrompidos (recebidos com erros)**
  - **Pacotes perdidos**
  - **Pacotes recebidos fora de ordem**

- **Deteção de erros**
  - **O recetor ser capaz de detetar pacotes recebidos com erros**
- **Métodos a abordar:**
  - **Código de verificação de paridade simples**
  - **Código de verificação de paridade em blocos**
  - **CRC (*Cyclic-Redundancy Check*)**

Código de verificação de paridade simples

- **Junta-se um bit adicional, o bit de paridade, à palavra binária a transmitir**
- **Bit de paridade escolhido por forma a garantir que a palavra completa tem um número par de 1s (paridade par) ou um número ímpar de 1s (paridade ímpar)**
- **Exemplo (paridade par):**

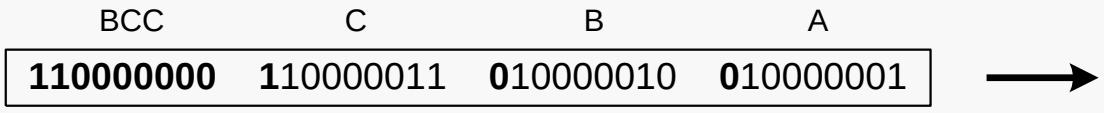
0	1	0	0	0	0	0	0	1	caracter ASCII A com bit de paridade par
0	1	0	0	0	0	0	1	1	com 1 bit errado
0	1	0	0	0	0	1	1	1	com 2 bits errados

- **Detecta todos os erros com um nº ímpar de bits errados; não detecta nenhum erro com um nº par de bits errados**

Código de verificação de paridade em blocos

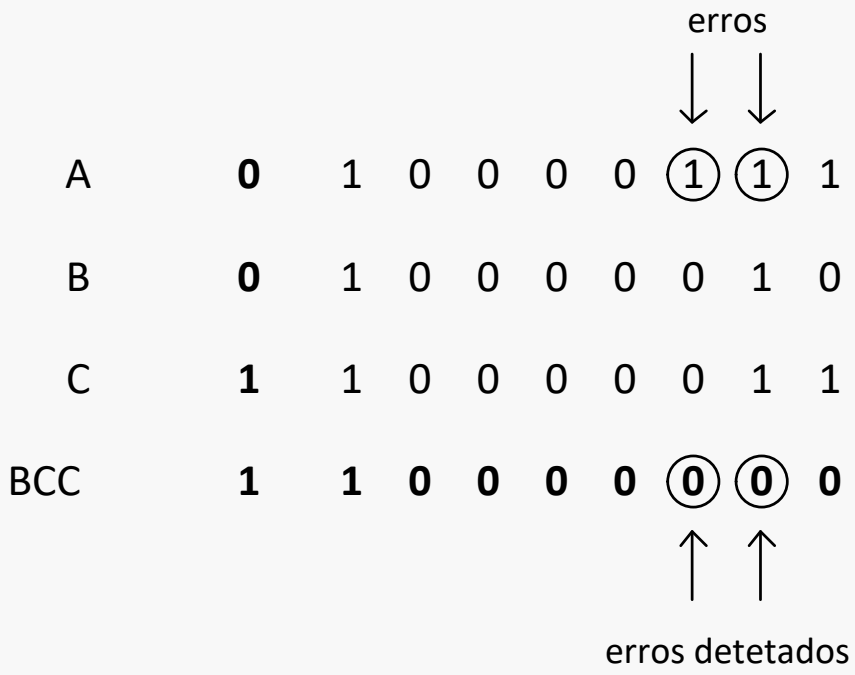
- **Utilizado na transmissão de blocos de palavras binárias**
- **É formado um bit de paridade em cada palavra individual (na horizontal) e também sobre o bloco de palavras (na vertical). É adicionado um carater designado por *Block-Check Caracter* (BCC) no fim do bloco.**
- **Exemplo:**

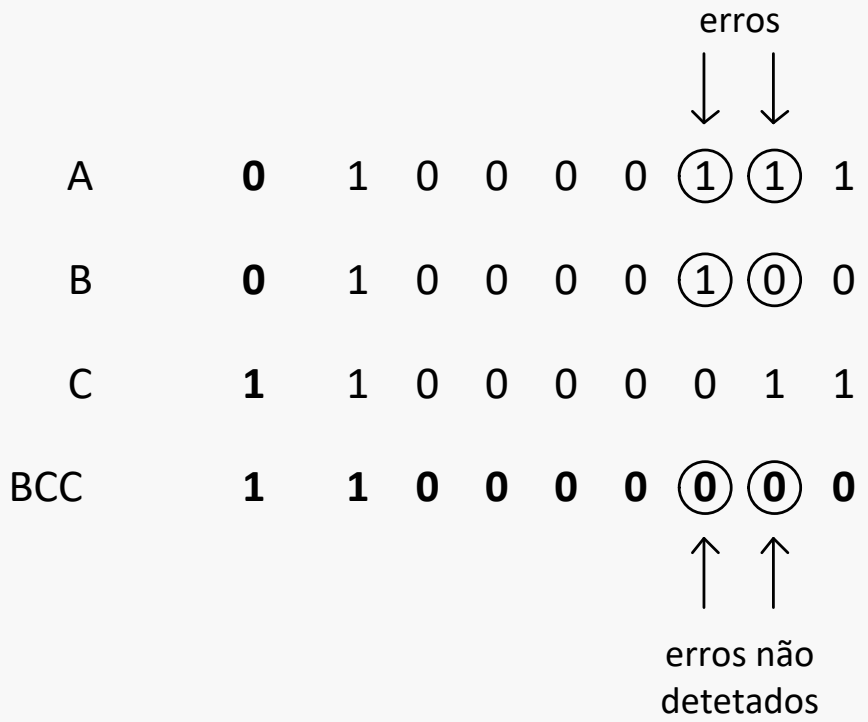
	bit de paridade								
A	**0**	1	0	0	0	0	0	0	1
B	**0**	1	0	0	0	0	0	1	0
C	**1**	1	0	0	0	0	0	1	1
BCC	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**



sequência de bits transmitidos

Código de verificação de paridade em blocos

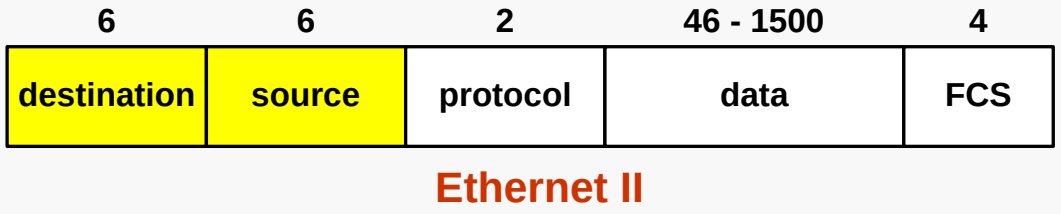




CRC (*Cyclic-Redundancy Check*)

- **Mensagem a transmitir: 57268**
- **Emissor e recetor combinam divisor: 84**
- **No emissor executa-se 57268 / 84 = 681 + 64/84**
- **O emissor transmite 5726864**

- **A mensagem chega com erros ao recetor: 5754864**
- **Agora 57548 / 84 = 685 + 8/84**
- **Como resto é diferente de 64 o erro é detetado!**



CRC (*Cyclic-Redundancy Check*)

- **Represente-se a sequência de bits**

$$b_{n-1}, b_{n-2}, ..., b_3, b_2, b_1, b_0$$

através de um polinómio

$$b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + ... + b_3x^3 + b_2x^2 + b_1x + b_0$$

- **A geração de um CRC para uma mensagem $M(x)$ com m bits segue os seguintes passos:**
  1. **O emissor e o recetor acordam num polinómio gerador $G(x)$, com pelo menos dois termos não-nulos, x^r e 1, onde r é a ordem de $G(x)$.**
  2. **O emissor adiciona r zeros no fim da mensagem a ser transmitida. A mensagem fica então com $m + r$ bits correspondendo ao polinómio $x^rM(x)$.**
  3. **O emissor determina o resto da divisão de $x^rM(x)$ por $G(x)$ (tem sempre r ou menos bits). Este resto designa-se por $R(x)$.**
  4. **A mensagem transmitida é $T(x) = x^rM(x) + R(x)$.**

CRC (*Cyclic-Redundancy Check*)

- **Exemplo:**
  - $M(x) = x^4 + x^3 + x^2 + 1$ ($m = 5$);
  - $G(x) = x^3 + 1$ ($r = 3$);
  - $x^r M(x) = x^7 + x^6 + x^5 + x^3$;
  - $R(x) = x^2 + x$;
  - $T(x) = x^r M(x) + R(x) = x^7 + x^6 + x^5 + x^3 + x^2 + x$.

- **Seja $Z(x)$ o resultado da divisão de $x^r M(x)$ por $G(x)$. Então**

$$x^r M(x) = G(x)Z(x) + R(x)$$

e

$$T(x) = x^r M(x) + R(x) = G(x)Z(x)$$

ou seja, todas as palavras transmitidas são <u>divisíveis</u> por $G(x)$. (Nota: subtração módulo 2= adição módulo 2).

CRC (*Cyclic-Redundancy Check*)

- **A mensagem recebida pode conter erros, isto é, pode ser $T(x) + E(x)$, onde $E(x)$ é o polinómio que representa os erros.**

- **O recetor divide a mensagem recebida por $G(x)$, isto é, executa**

$$[T(x)+E(x)]/G(x)$$

- **Uma vez que**

$$\text{Resto de } [T(x)+E(x)]/G(x) = \text{Resto de } E(x)/G(x)$$

então, o recetor decide que não houve erro se o resto for zero e que houve erros caso contrário.

CRC (*Cyclic-Redundancy Check*)

- **Os erros não serão detetados, se e só se,**

$$E(x) = G(x)Z(x)$$

para algum polinómio não-nulo $Z(x)$.

- **Erros detetados pelos CRC:**
  - **Todos os erros de 1 bit.**
  - **Todos os erros de 2 bits, quando $G(x)$ tem um fator com pelo menos 3 termos.**
  - **Qualquer nº ímpar de erros, quando $G(x)$ tem um fator $(x+1)$.**
  - **Todas as rajadas de erros com um comprimento inferior ao comprimento do CRC.**

CRC (*Cyclic-Redundancy Check*)

Norma	Polinómio Gerador $G(x)$
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16 (ANSI)	$x^{16}+x^{15}+x^5+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
CRC-CCITT (V.41)	$x^{16}+x^{12}+x^5+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Exemplos

Considere um emissor com os 4 bytes 01110101.01100101.01000101.00110000 para enviar com um código de paridade ímpar aplicado a palavras de 8 bits. Diga justificadamente qual a sequência de bits enviada.

Considere a recepção da sequência binária "1011100110" gerada com controle de erros através de um CRC com polinómio gerador $x^3 + x^2 + 1$. Determine justificando se o receptor assume que houve erros de transmissão ou não.

Considere que um emissor tem os 2 bytes 00001101.11100111 para enviar e usa um código CRC com o polinómio gerador x^4+x+1. Indique justificadamente qual a sequência de bits enviada.