Nivel enlace de datos

Tema 3

Alcaraz, S., Roig, P.J.

Fundamentos de Redes de Telecomunicación
Grado en Ingeniería de Tecnologías de la Telecomunicación

Area de Arquitectura y Tecnología de Computadores
Departamento de Física y Arquitectura de Computadores
Universidad Miguel Hernández

{salcaraz,proig}@umh.es

05/01/2025

Objetivos de aprendizaje

Después de abordar la unidad, el estudiante será capaz de:

- Entender el uso del direccionamiento físico.
- Explicar los mecanismos básicos para el uso de códigos detectores de error.
- Entender la operación del chequeo de redundancia cíclica.
- Definir la distancia Hamming.
- Explicar los principios de la corrección hacia adelante mediante códigos de bloque.
- Presentar los diferentes tipos de entramado y el problema de la transparencia de datos.
- Explicar la necesidad de control de flujo y control de error.
- Presentar los aspectos fundamentales de los mecanismos de control de flujo basados en parada y espera y ventana deslizante. b
- Presentar los aspectos fundamentales de los mecanismos de recuperación de errores ARQ parada, ARQ retroceso-N y ARQ rechazo selectivo.
- Presentar un resumen de HDLC.

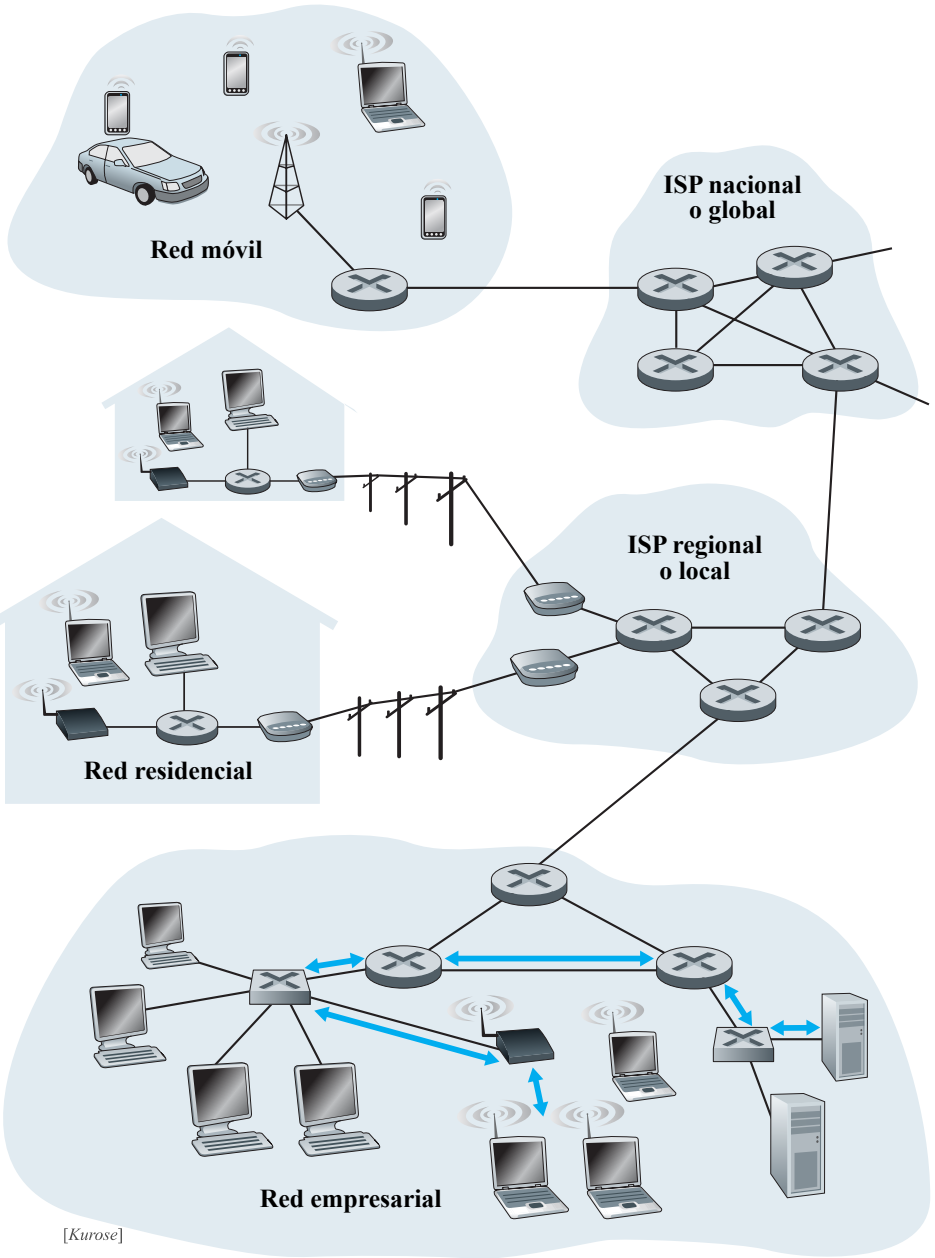
Nivel de enlace de datos

1. Funciones de nivel de enlace
2. Direccionamiento
3. Control de errores
4. Entramado
5. Control de flujo
6. HDLC

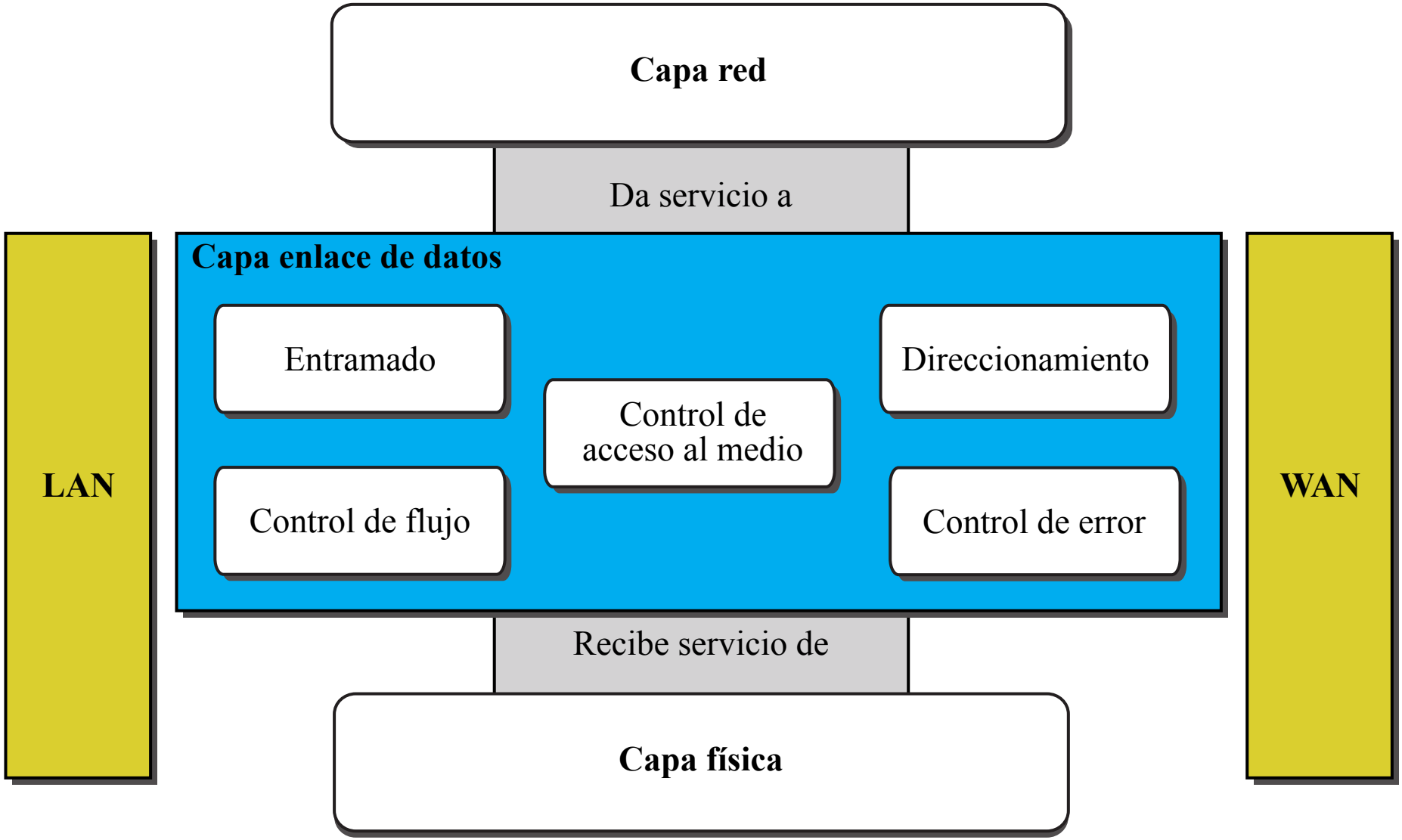
Nivel de enlace de datos

1. ***Funciones de nivel de enlace***
2. Direccionamiento
3. Control de errores
4. Entramado
5. Control de flujo
6. HDLC

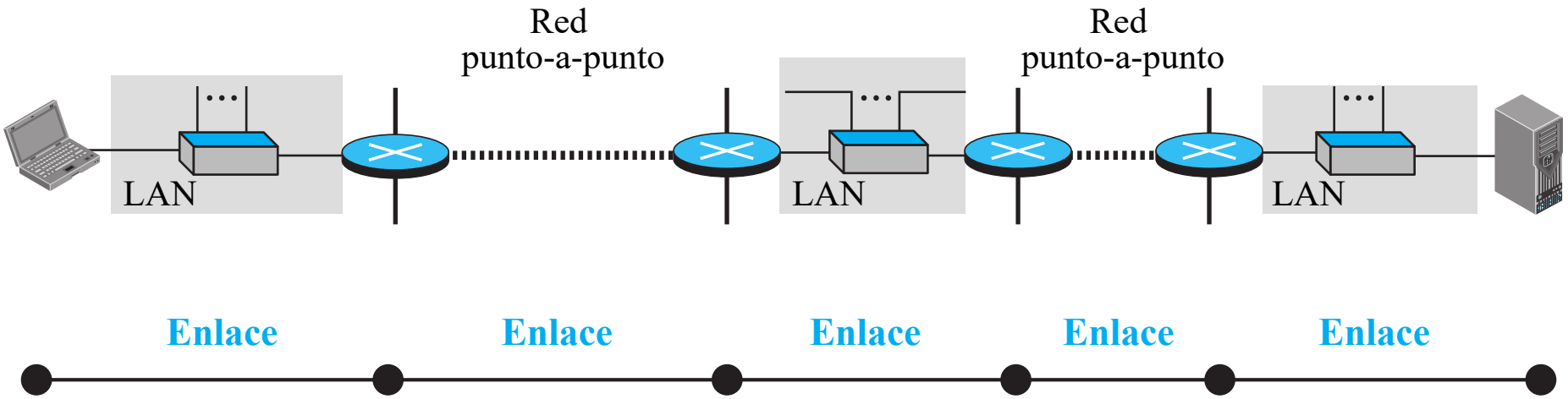
Introducción al nivel de enlace



Funciones del nivel de enlace

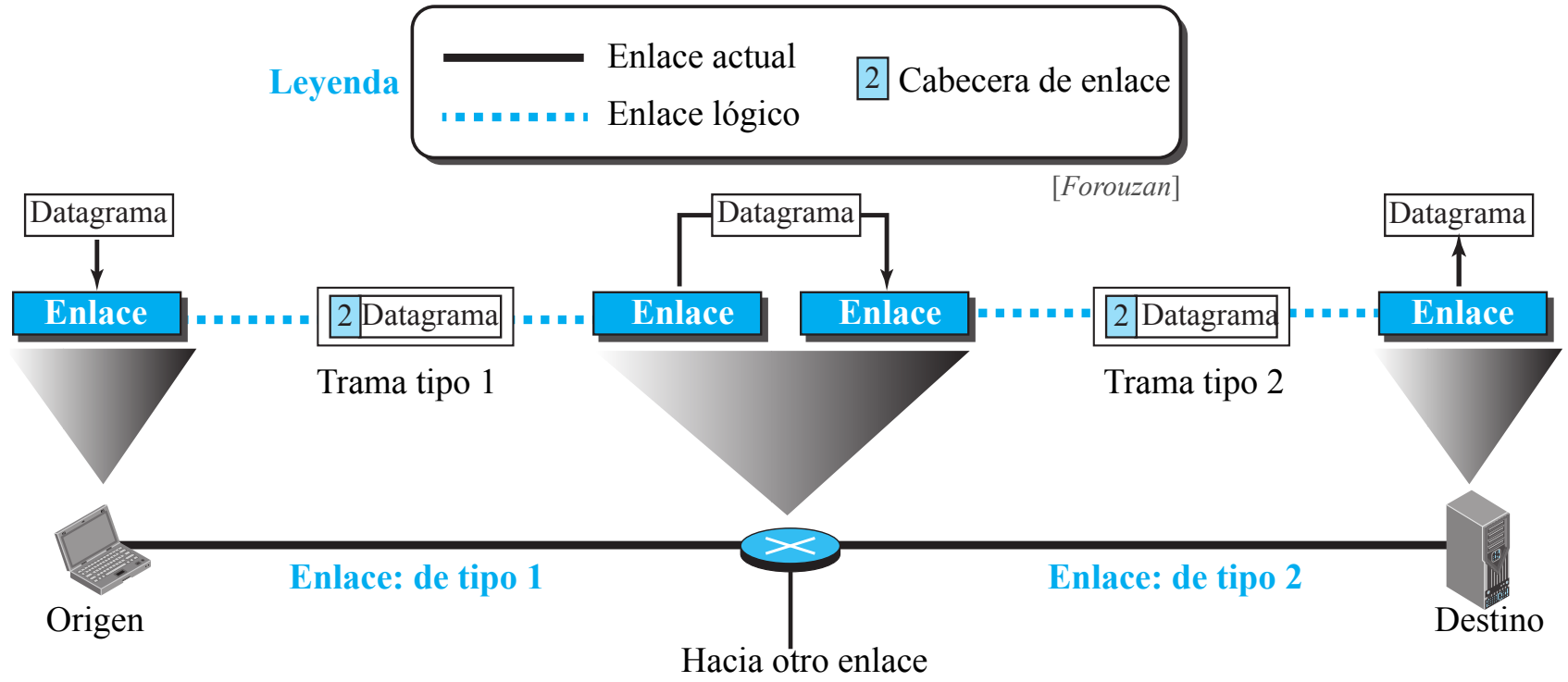


Nodos y enlaces



A nivel de enlace, la **comunicación** es **nodo a nodo**.

Servicios



En cada segmento, cada dispositivo realiza las operaciones de desencapsulado y encapsulado, según la tecnología de red, nodo a nodo, hasta que la trama alcance el destino.

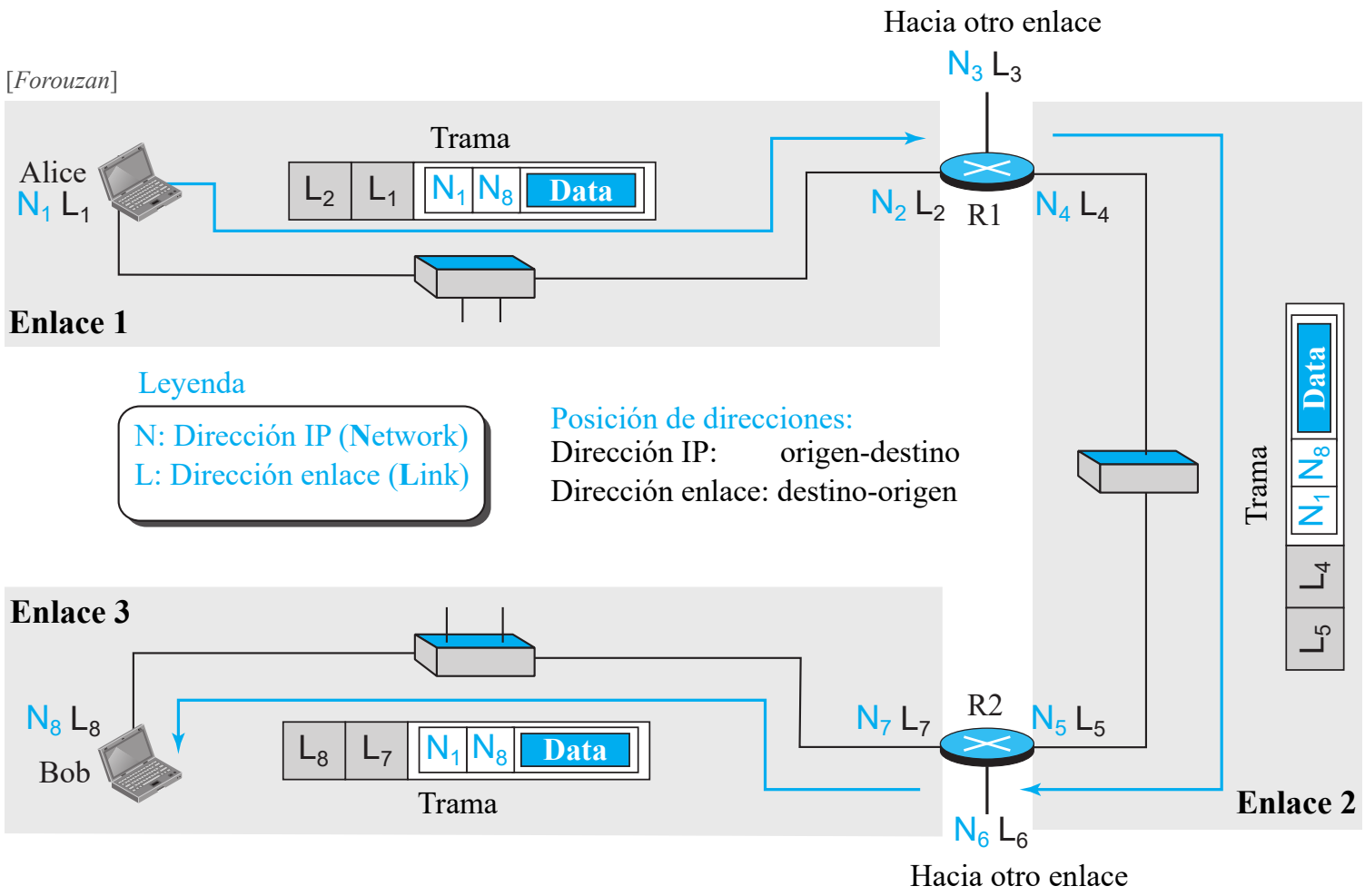
El datagrama de nivel superior permanece invariable.

La unidad de dato de protocolo a nivel enlace de datos se denomina **trama** (*frame*).

Nivel de enlace de datos

1. Funciones de nivel de enlace
2. ***Direccionamiento***
3. Control de errores
4. Entramado
5. Control de flujo
6. HDLC

Direccionamiento a nivel de enlace



Las direcciones a nivel enlace de cada segmento de red corresponden al origen y destino de la trama, permaneciendo invariante el datagrama encapsulado.

Es evidente, que el datagrama y las direcciones IP no deben variar a lo largo de los diferentes encapsulados.

Tipos de direcciones a nivel enlace

Unicast: Las direcciones unicast se asignan a host o interfaces de router. Unicasting significa comunicación **uno-a-uno**. Una trama con dirección unicast va dirigida a un sólo nodo de la red.

`A3:34:45:11:92:F1`

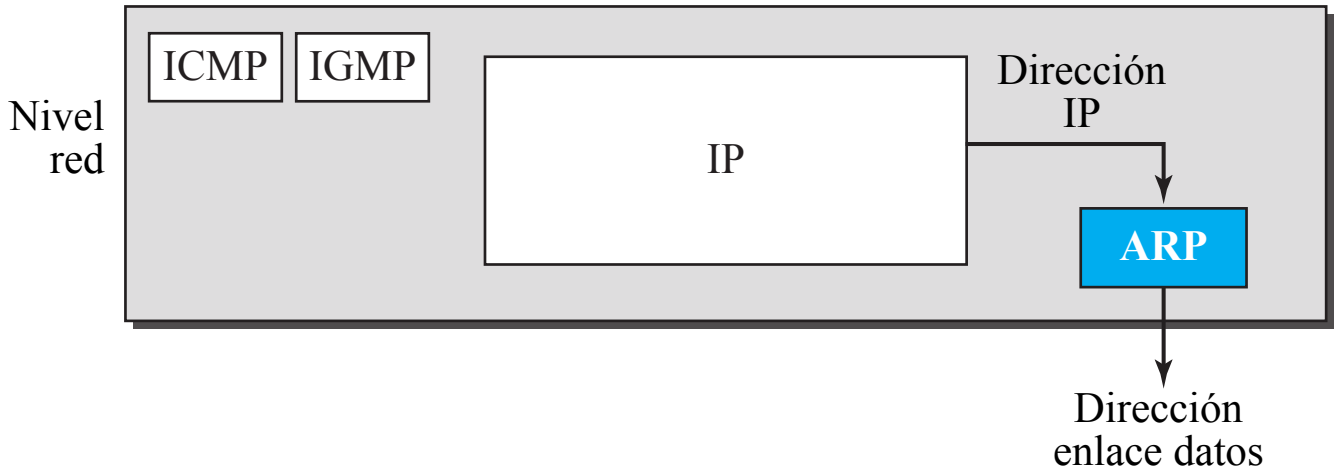
Multicast: Algunos protocolos a nivel enlace definen direcciones multicast. Multicasting significa **uno-a-muchos**.

`01:00:5E:11:92:F1`

Broadcast: Algunos protocolos a nivel enlace definen direcciones broadcast. Broadcasting significa **uno-a-todos**. Una trama con dirección broadcast irá destinada a todos los hosts del enlace.

`FF:FF:FF:FF:FF:FF`

Address Resolution Protocol (ARP)

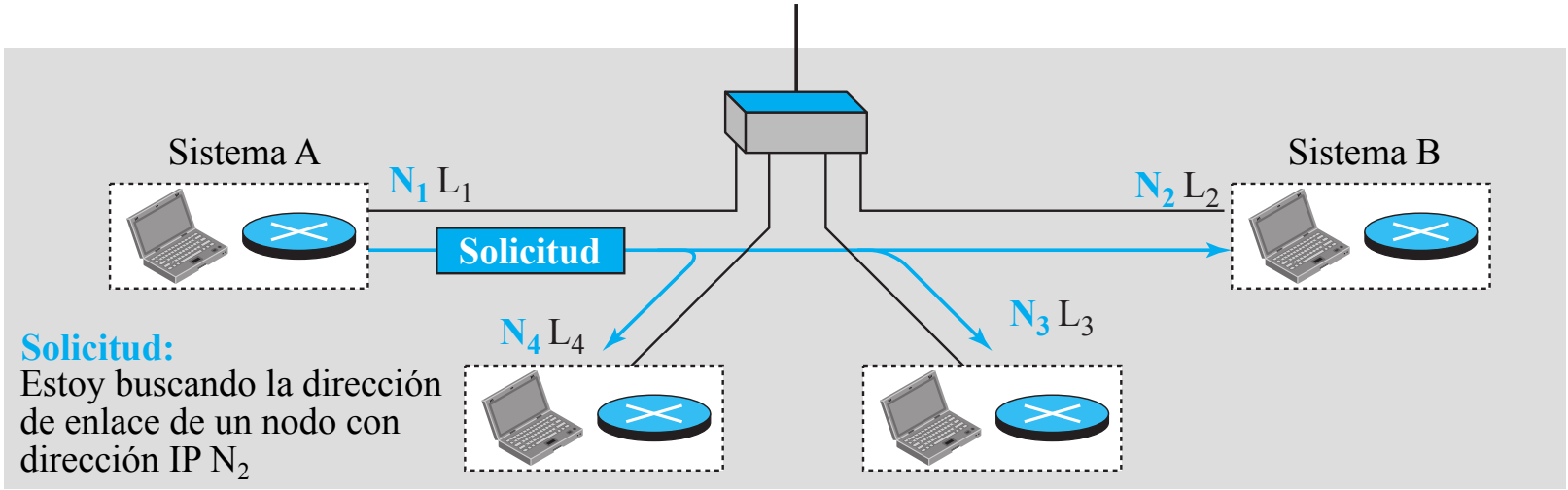


En el envío de datagramas IP ocurre que se conoce la dirección IP (red), pero no la dirección física (enlace), por eso hay que descubrirla, mediante ARP.

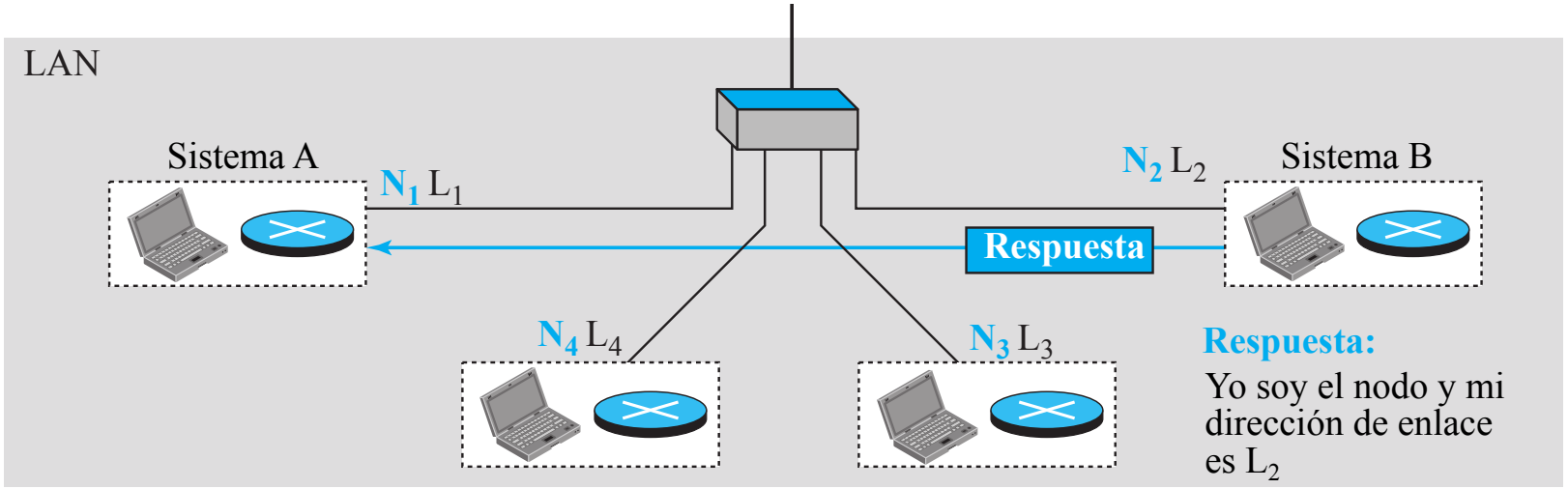
ARP es un protocolo auxiliar a nivel de red, que relaciona una dirección IP con su correspondiente dirección de enlace.

ARP es un protocolo que no sigue estrictamente los principios de niveles en el modelo TCP/IP, ya que, a partir de una dirección IP de un host remoto, descubre su dirección de enlace, y se la pasa a su nivel de enlace.

ARP: solicitud y respuesta

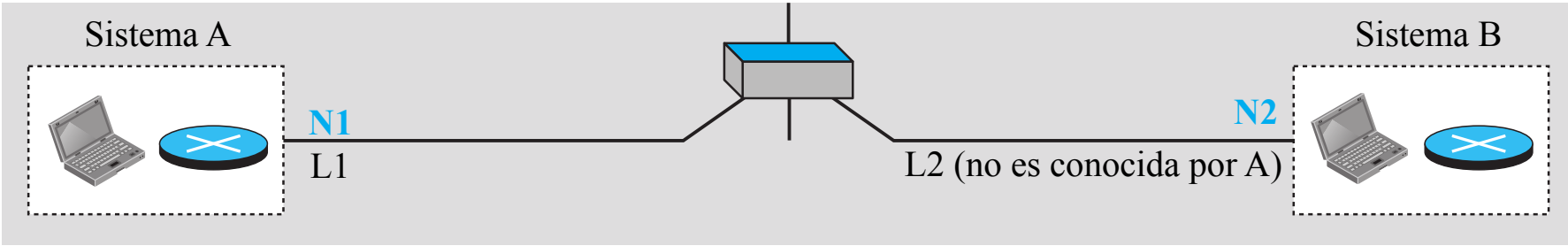


(a) La solicitud ARP es broadcast. [*Forouzan*]



(b) La respuesta ARP es unicast.

Ejemplo con cabeceras ARP



[*Forouzan*]

Solicitud ARP

0x0001	0x0800
0x06 | 0x04	0x0001
L1	
N1	
Todo 0s	
N2	

Trama broadcast

B	A		**Data**	
Destino	Origen			

1 De A hacia B

Respuesta ARP

0x0001	0x0800
0x06 | 0x04	0x0002
L2	
N2	
L1	
N2	

Trama unicast

A	B		**Data**	
Destino	Origen			

2 De B hacia A

B: dirección broadcast

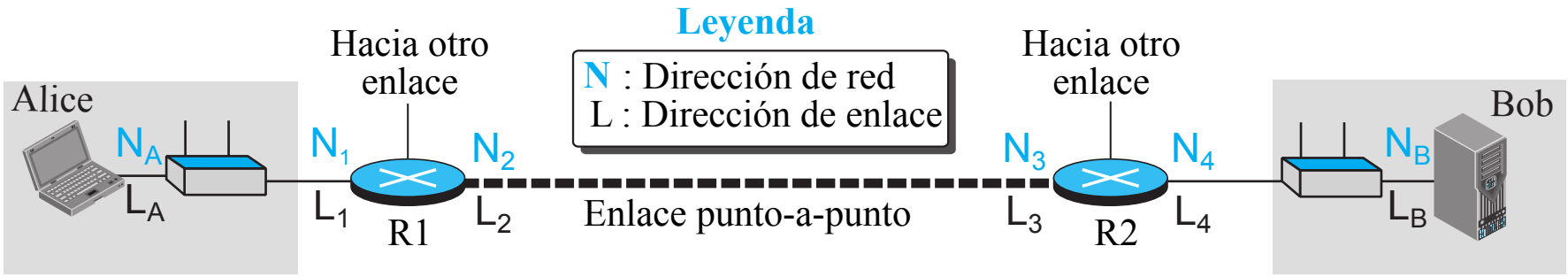
0	8	16	31
Hardware Type		Protocol Type	
Hardware length	Protocol length	Operation **Request:1, Reply:2**	
Source **hardware** address			
Source **protocol** address			
Destination **hardware** address (empty in request)			
Destination **protocol** address			

Hardware: LAN or WAN protocol
Protocol: Network-layer protocol

[RFC 826]

Un ejemplo de comunicación a nivel de enlace

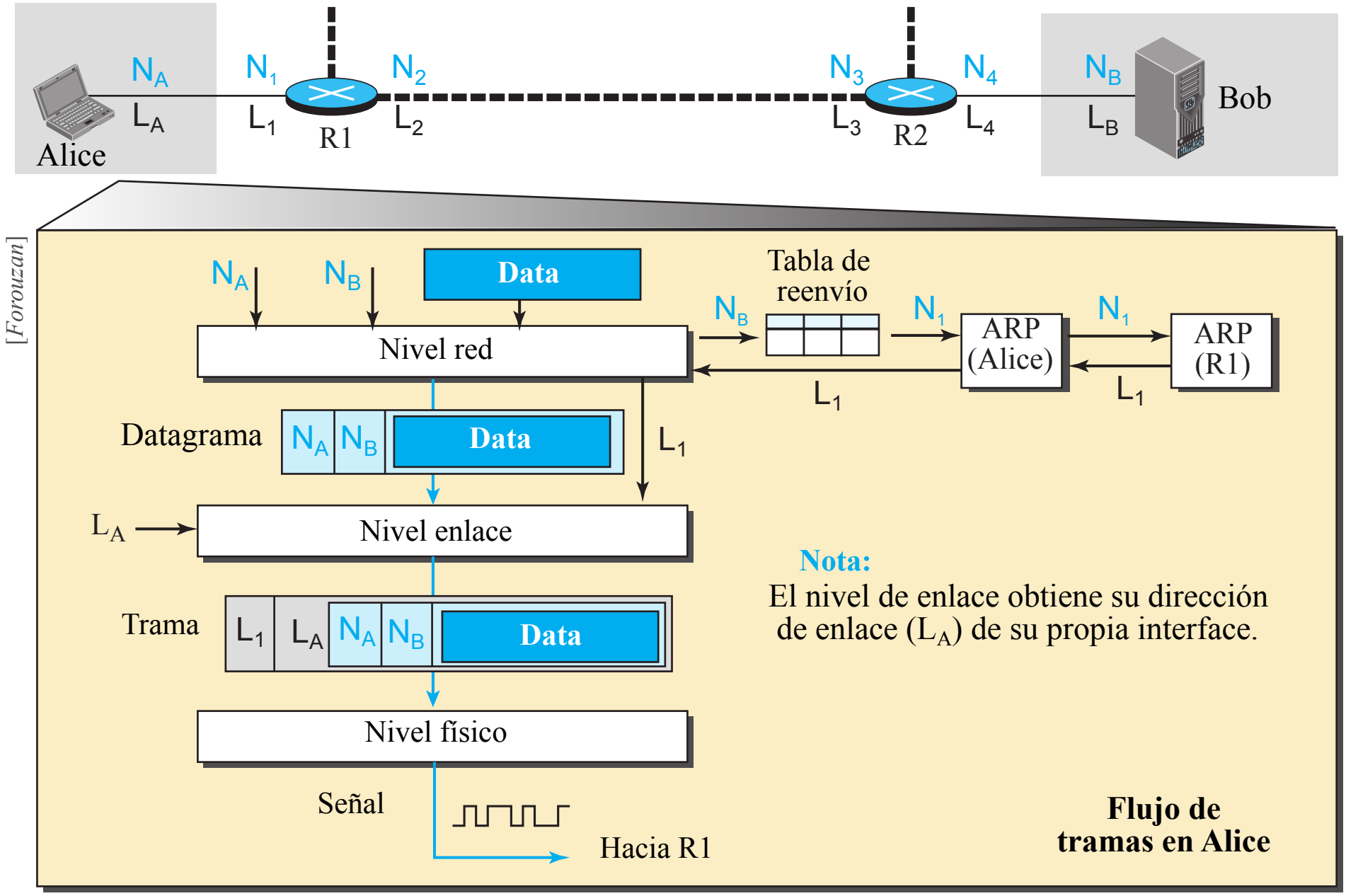
Ejemplo: envío de trama **unicast** desde nodo *Alice* a nodo *Bob*:



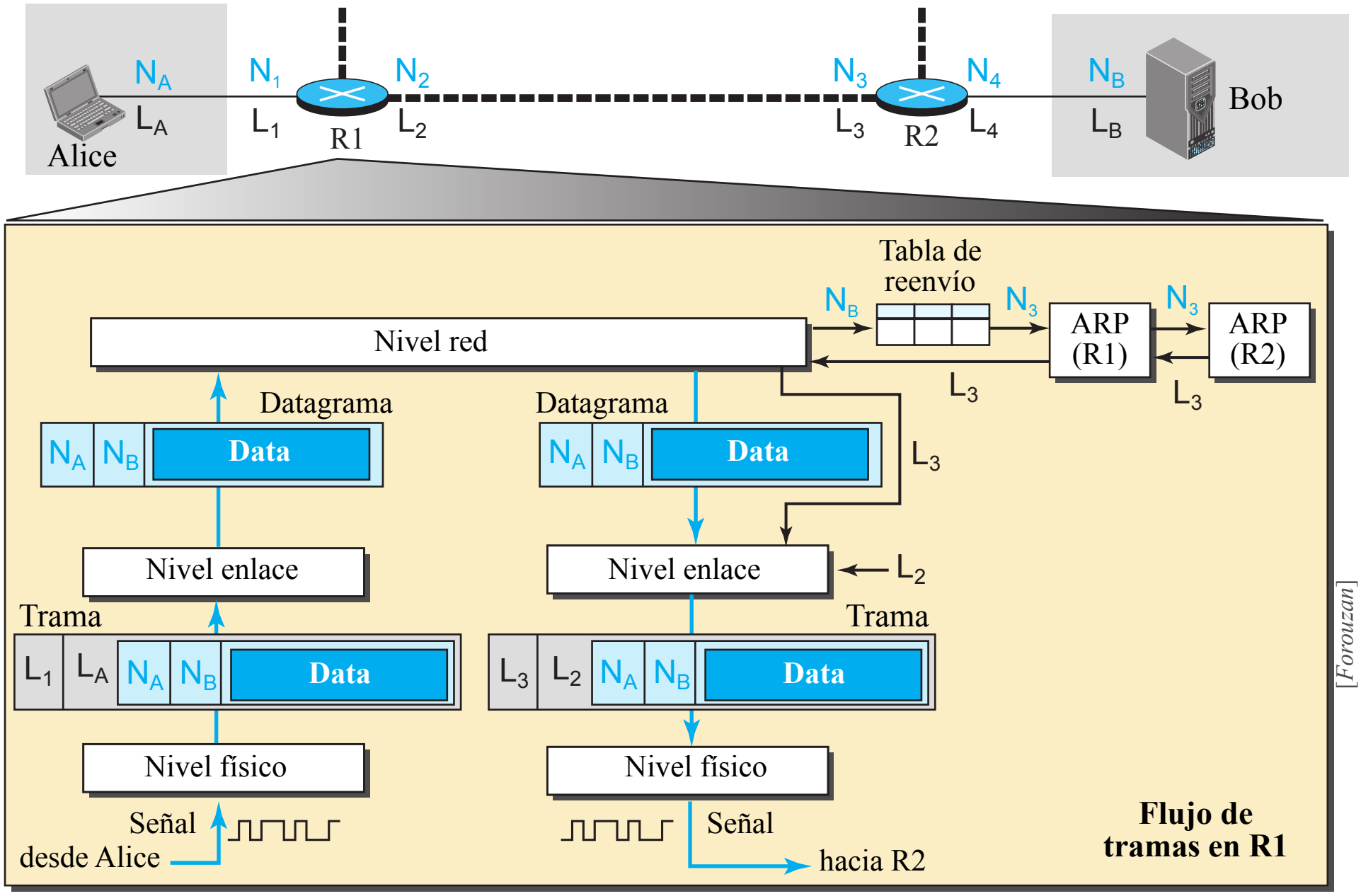
Suposiciones:

- Las direcciones de red (network) y enlace (data link) vienen identificadas con N_x y L_x, respectívamente.
- Alice conoce la dirección de red (IP) de Bob, definida por N_B.
- Todos los hosts e interfaces de routers están dotados de direcciones de red y enlace.
- Los hosts conocen sus *gateways* (interfaces de router).
- El enrutamiento está bien establecido y las tablas de reenvío completas, por lo que los hosts conocen cómo alcanzar el resto de redes.

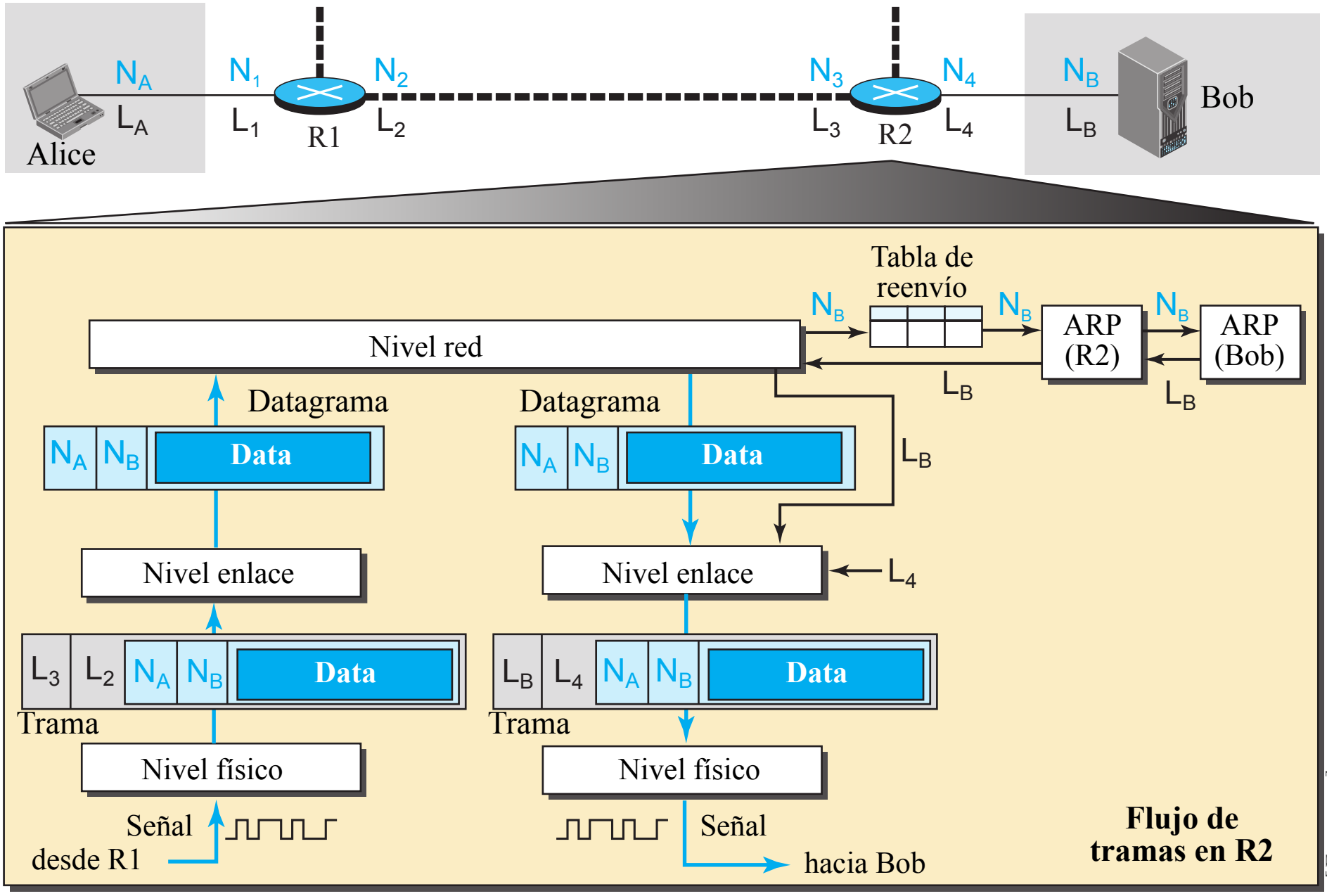
Ejemplo: Alice



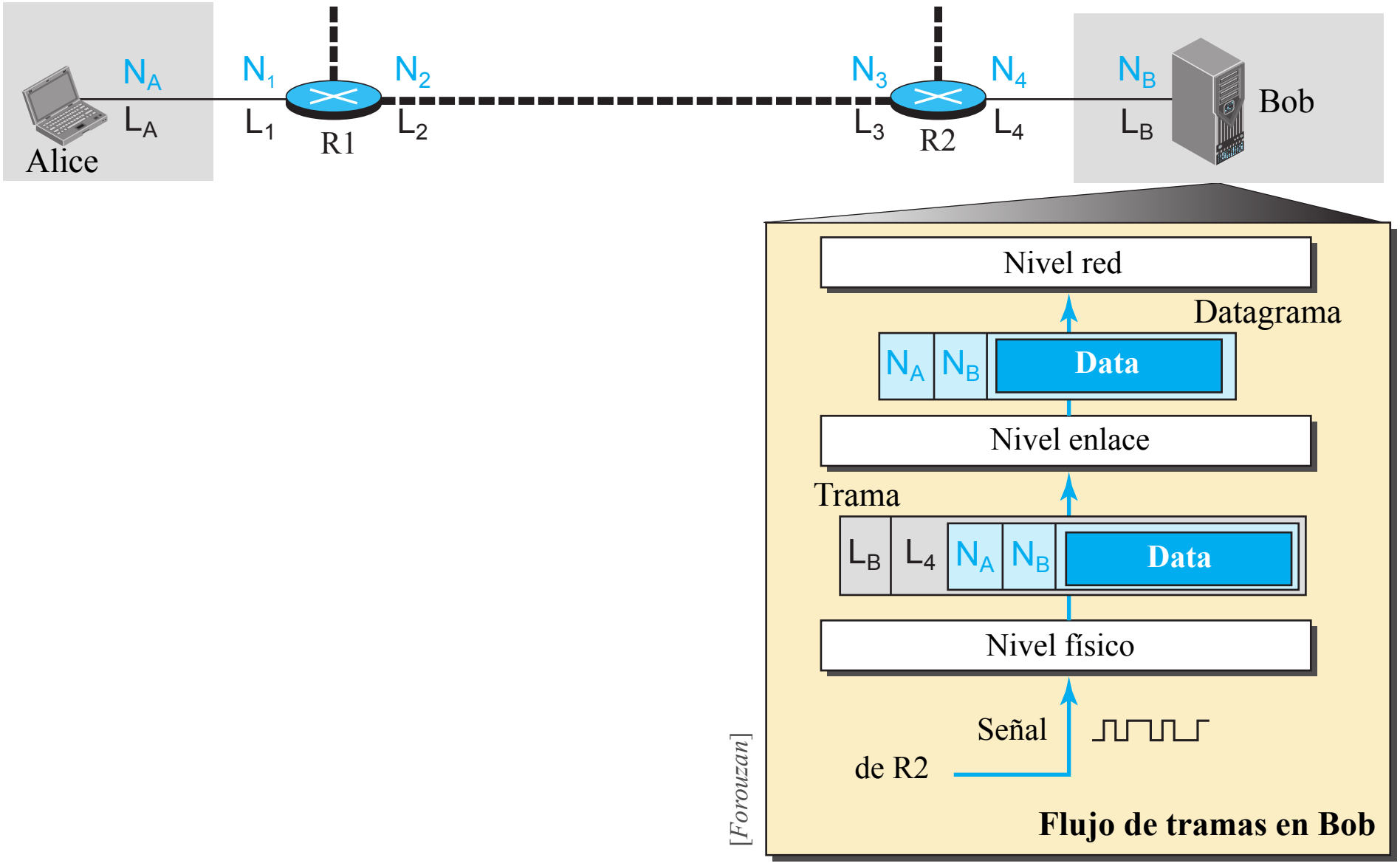
Ejemplo: R1


Alice
N_A
L_A
N_1
L_1
R1
N_2
L_2
N_3
L_3
R2
N_4
L_4
N_B
L_B
Bob
Tabla de reenvío
N_B
N_3
ARP (R1)
N_3
ARP (R2)
L_3
L_3
Nivel red
Datagrama
Datagrama
N_A N_B Data
N_A N_B Data
L_3
Nivel enlace
Nivel enlace
L_2
Trama
Trama
L_1 L_A N_A N_B Data
L_3 L_2 N_A N_B Data
Nivel físico
Nivel físico
Señal
desde Alice
Señal
hacia R2
Flujo de tramas en R1
[Forouzan]

Ejemplo: R2


Alice
N_A
L_A
N_1
L_1
R1
N_2
L_2
N_3
L_3
R2
N_4
L_4
N_B
L_B
Bob
Tabla de reenvío
Nivel red
ARP (R2)
ARP (Bob)
Datagrama
Data
Nivel enlace
Trama
Nivel físico
Señal
desde R1
hacia Bob
Flujo de tramas en R2
[Forouzan]

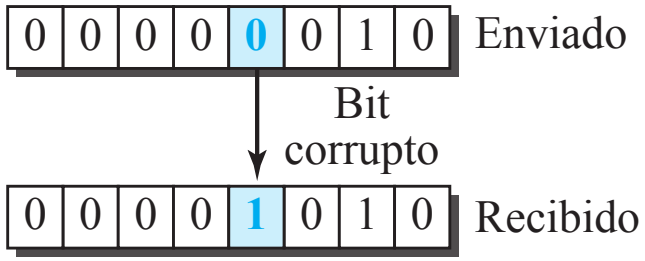
Ejemplo: Bob


Alice
N_A
L_A
N_1
L_1
R1
N_2
L_2
N_3
L_3
R2
N_4
L_4
N_B
L_B
Bob
Nivel red
Datagrama
N_A N_B Data
Nivel enlace
Trama
L_B L_4 N_A N_B Data
Nivel físico
Señal
de R2
Flujo de tramas en Bob
[Forouzan]

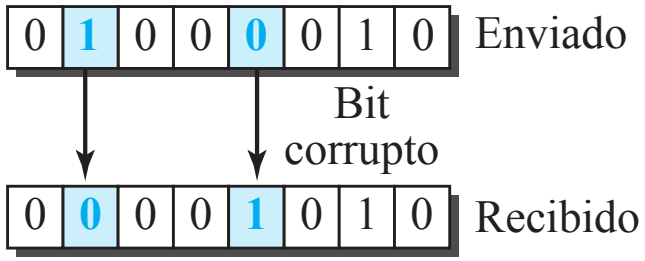
Nivel de enlace de datos

1. Funciones de nivel de enlace
2. Direccionamiento
3. ***Control de errores***
4. Entramado
5. Control de flujo
6. HDLC

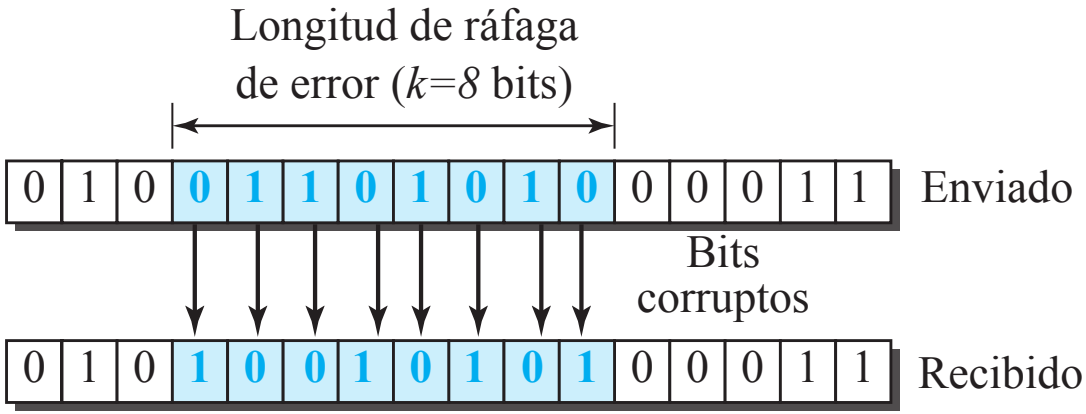
Errores



a) Error simple



b) Error doble



c) Error en ráfaga

Detección vs corrección

¿Que es más apropiado la detección o corrrección de errores?

Depende:

- La corrrección es mucho más costosa que la detección.
- En **detección**, sólo interesa descubrir si se ha producido algún error, independientemente del número de bits erróneos.
- A efectos de trama errónea, es indiferente que tenga un bit erróneo o muchos (ráfaga de bits), es igualmente erróneo.
- En **corrección**, se necesita averiguar cuántos bits erróneos se han producido, y además, sus posiciones.

Por ejemplo, en corrección de errores, según la longitud del código y la cantidad de errores a corregir, obtenemos una complejidad:

- Un error simple en una código de 8 bits: $\binom{8}{1} = 8\ posiciones.$
- Dos errores en el mismo código de 8 bits: $\binom{8}{2} = 28\ posiciones.$
- Diez errores en un código de 1000 bits: $\binom{1000}{10} = 2{,}63 \times 10^{23}\ posiciones.$

Probabilidades de error

Los errores se producen cuando uno o más bits cambian en una secuencia de L bits (que llamaremos **trama**). Sea:

- P_b, la probabilidad (constante e independiente) que un bit recibido sea erróneo (también se conoce como **BER**, *Bit error rate*).
- P_1, probabilidad que una trama, en el receptor, esté libre de error.
- P_2, probabilidad que la trama llegue con algún bit erróneo y, mediante alguna técnica de detección de error, es detectado, y por tanto, la consideramos trama libre de error.
- P_3, probabilidad que la trama llegue con algún bit erróneo, pero aún utilizando alguna técnica de detección de error, no es detectado (suponemos que $P_3 = 0$).

Entonces:

$$P_1 = \overbrace{(1-P_b)(1-P_b)\cdots(1-P_b)}^{L} = (1-P_b)^L$$

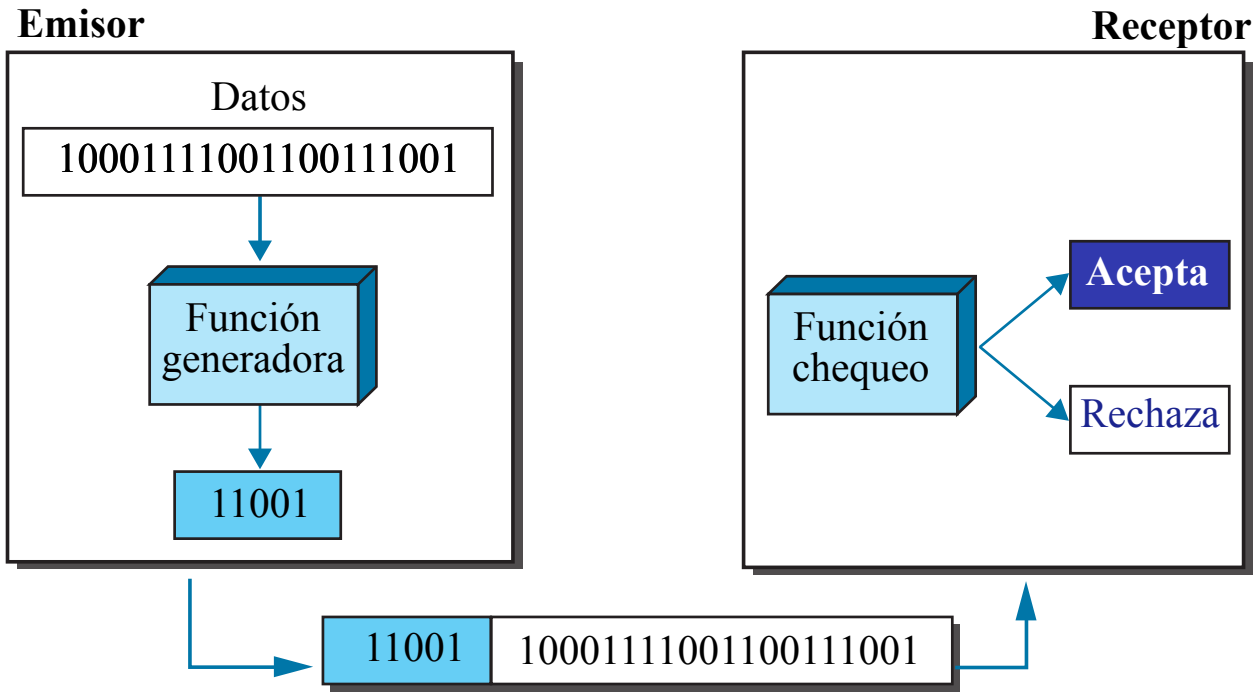
por lo que:

$$P_2 = 1 - P_1 = 1 - (1-P_b)^L$$

de donde concluímos que, la probabilidad de trama libre de error (P_2), disminuye cuando:

- La probabilidad de bit (P_b) erróneo aumenta.
- La longitud de la trama (L) aumenta.

Redundancia y codificación



El concepto de **redundancia** se utiliza tanto en detección como en corrección de errores y se implementa mediante técnicas de **codificación**.

- El emisor añade redundancia mediante un proceso de codificación de datos.
- El receptor decodifica y chequea la relación entre ambos conjuntos de bits para determinar la presencia de información corrupta.

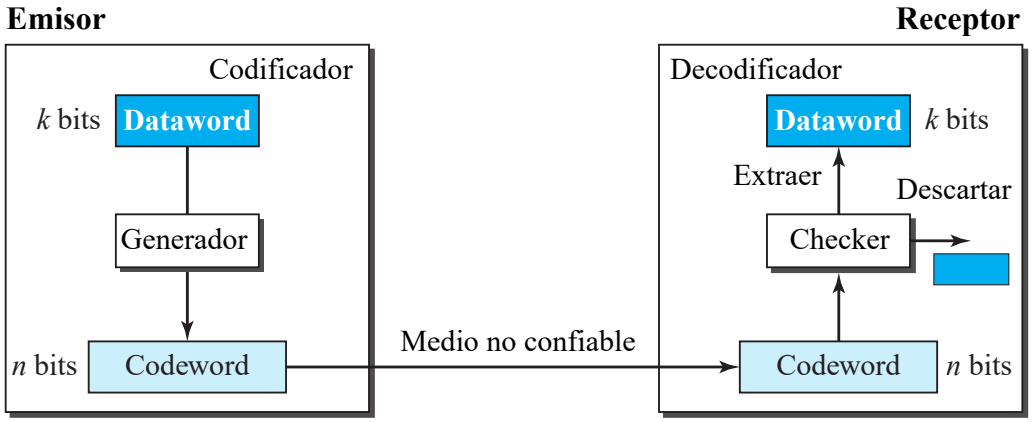
Existen dos tipos de técnicas de codificacion:

Códigos de bloque: bit de paridad, Hamming, CRC, etc.

Códigos de convolución: (fuera del ámbito de esta asignatura)

Códigos de bloque

Códigos de bloque



Una forma de representar los **códigos de bloque** es:

$$C(n, k)$$

El mensaje se divide en bloques de k bits, denominado **dataword**.

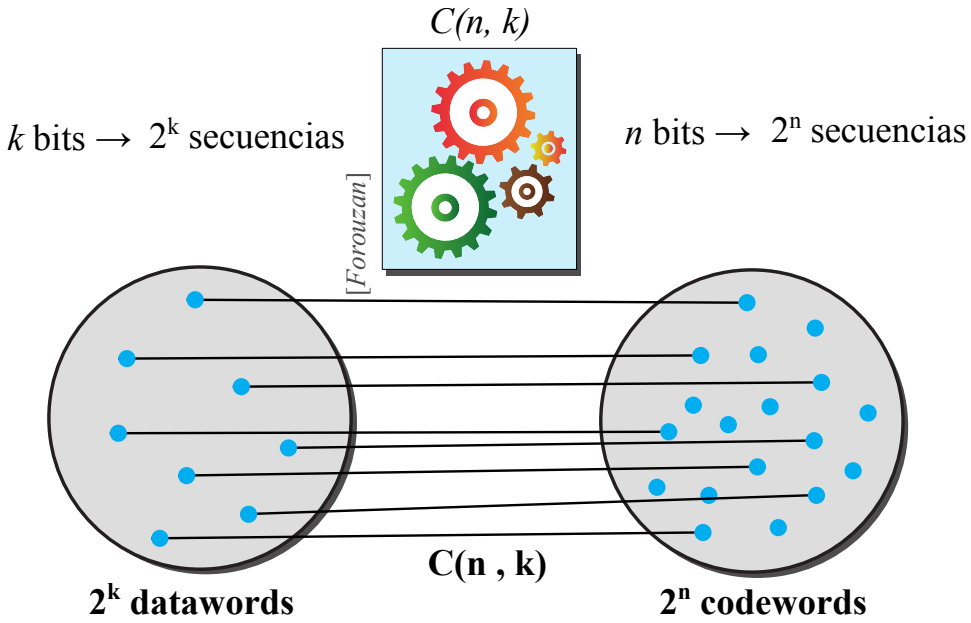
Se añaden r bits redundantes a cada bloque, con lo que la longitud final es $n = k + r$.

El bloque resultante final de n bits se denomina **codeword**.

El receptor puede detectar un cambio en el codeword original si se cumplen las siguientes condiciones:

1. El receptor tiene (o puede encontrar, o generar) una lista de los codewords válidos.
2. El codeword original ha cambiado a un codeword inválido.

Detección de errores (I)



Sea un conjunto de *datawords* de tamaño k y un conjunto de *codewords* de tamaño n:

- Con k bits se pueden definir 2^k *datawords*.
- Con n bits se pueden definir 2^n *codewords*.

Como $n > k$, el conjunto de *codewords* es más grande que el conjunto de *datawords*.

El proceso de codificación es uno-a-uno, por lo tanto, un *dataword* siempre se codificará con el mismo *codeword*.

Por lo tanto, tendremos $(2^n - 2^k)$ *codewords* no utilizados (**ilegales** o **inválidos**).

La tarea consiste en detectar la existencia de **códigos inválidos**.

Detección de errores (II)

Sea el código $C(3,2)$ definido en la siguiente tabla:

Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

Supongamos el **emisor** codifica el dataword 01 como 011 y lo envía al receptor (a través de un medio no confiable).

En el **receptor** se pueden producir los siguientes casos:

1. El receptor recibe 011 ⇒ codeword **válido** ⇒ extrae el dataword 01.
2. El codeword se corrompe ⇒ se recibe 111 (el bit de más a la izquierda) ⇒ codeword **inválido** ⇒ descartado.
3. El codeword se corrompe ⇒ se recibe 000 (dos bits a la derecha) ⇒ codeword **válido** ⇒ el receptor extrae (incorrectamente) el dataword 00.

Un código detector de error sólo detecta errores para los que ha sido diseñado; otros tipos pueden ser no reconocidos.

Distancia Hamming

La **distancia Hamming** entre las secuencias binarias, x e y, del mismo tamaño, es el número bits diferentes, comparado bit a bit y se representa por:

$$d(x, y)$$

En relación a la detección de error, la distancia Hamming entre los codewords enviados y recibidos es el número de bits alterados durante la transmisión.

Por ejemplo, si los codewords enviado y recibido son 00000 y 01101, respectivamente, tenemos que han sido alterados 3 bits durante la transmisión, por lo tanto:

$$d(00000, 01101) = 3$$

La distancia Hamming se calcula aplicando la operacion XOR:

$$\mathrm{d}(00000, 01101) = 3 \quad \Leftrightarrow \quad \begin{array}{r} 00000 \\ \oplus\ 01101 \\ \hline 01101 \end{array}$$

Un código de bloque es **lineal** si la XOR (suma módulo 2) de dos códigos válidos produce otro código válido.

Distancia Hamming mínima para detección de error

En un conjunto de codewords, la **distancia Hamming mínima** (d_{min}) es la menor distancia Hamming entre todos los posibles pares de codewords.

A efectos prácticos, la distancia Hamming mínima de un código lineal es el número de 1s en el códeword válido, distinto de cero, y con el menor número de 1s.

Si ocurren s errores durante la transmisión (s bits alterados), entonces la distancia Hamming entre codeword enviado y recibido es s.

En un sistema diseñado para detectar s errores, la distancia Hamming mínima debe ser $(s+1)$, de esta forma, un codeword recibido con error, no será un codeword válido, y por lo tanto, el error será detectado.

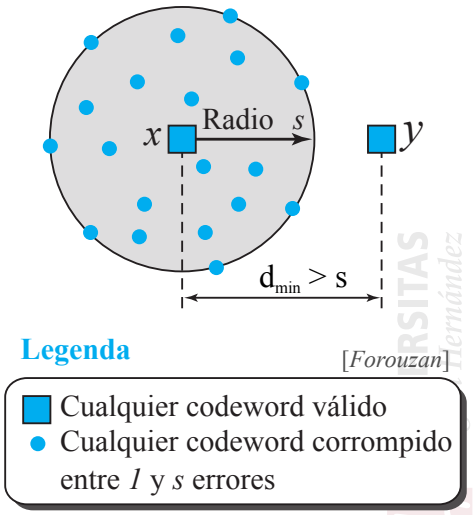
Desde un punto de visto geométrico, supongamos que el codeword enviado x está en el **centro** del círculo de radio s.

Todos los codewords recibidos con 0 hasta s errores son puntos **dentro** o en el perímetro del circulo.

El resto de codewords válidos deben estar **fuera** del círculo.

Por lo tanto, d_{min} debe ser un entero mayor que s, por lo tanto:

$$d_{min} = s + 1$$



Código de chequeo de bit de paridad

El **Código de chequeo de paridad** es un código de bloque lineal, donde a partir de un dataword de k bits se obtiene un codeword $n = k + 1$ bits.

El bit extra se denomina **bit de paridad** y se selecciona de forma que, el total de *1s* del codeword sea **par**.

Por ser $d_{min} = 2$, se trata de un código detector de error de un sólo bit.

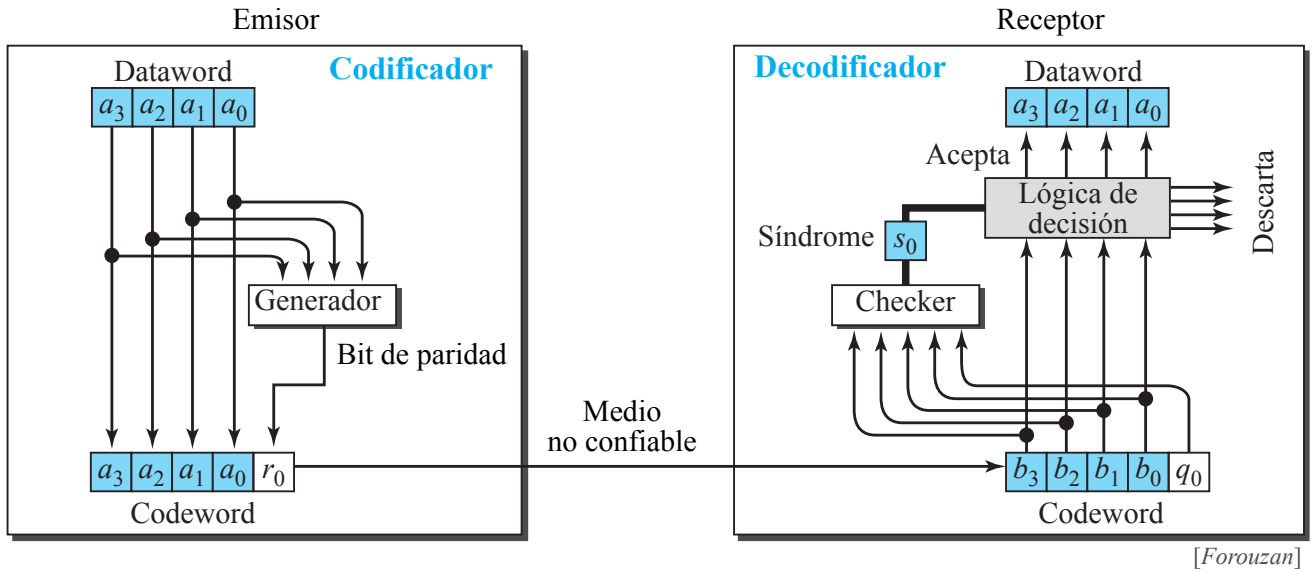
- Código chequeo de paridad $C(3, 2)$

Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

- Código chequeo de paridad $C(5, 4)$

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000**0**	1000	1000**1**
0001	0001**1**	1001	1001**0**
0010	0010**1**	1010	1010**0**
0011	0011**0**	1011	1011**1**
0100	0100**1**	1100	1100**0**
0101	0101**0**	1101	1101**1**
0110	0110**0**	1110	1110**1**
0111	0111**1**	1111	1111**0**

Codificación del chequeo de bit de paridad *C(5,4)*



En el **codificador**, a partir del dataword, se genera el bit de paridad mediante arimética módulo 2 (sumas sin acarreo), de forma que el total de 1s es par:

$$r_0 = (a_3 + a_2 + a_1 + a_0)_{|2}$$

El codeword se transmite por un medio no confiable (puede sufrir alguna distorsión).
En el receptor (**decodificador**), realiza la suma (módulo 2) sobre la totalidad del codeword.
El resultado, denominado **síndrome**, es 1 bit, de forma que, el síndrome es 0 cuando el total de 1s del codeword recibido es par, en otro caso es 1.

$$s_0 = (b_3 + b_2 + b_1 + b_0 + q_0)_{|2}$$

Si el síndrome es 0, no se ha detectado error en el codeword, y por lo tanto, los bits correspondientes al dataword son aceptados, en otro caso, rechazados.

El bit de paridad puede detectar un número impar de errores.

Códigos de redundancia cíclica (CRC)

Códigos de redundancia cíclica (CRC)

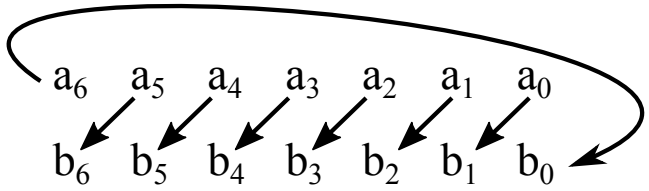
Los **códigos cíclicos** (*CRC, Check Redundancy Cyclic*) pertenecen al conjunto de códigos de bloque lineales con una propiedad adicional:

*en un código **cíclico**, si un codeword es cíclicamente rotado, el resultado es otro codeword válido.*

Por ejemplo, a partir del codeword definido por 1011000, desplazando los bits hacia la izquierda, y de forma cíclica, se obtienen los siguientes codewords válidos:

$$1011000 \overset{\text{Desplazamiento izquierda}(\leftarrow)}{\cdots} 0110001 \overset{\text{Desplazamiento izquierda}(\leftarrow)}{\cdots} 1100010 \overset{\text{Desplazamiento izquierda}(\leftarrow)}{\cdots} 1000101\cdots$$

Si denominamos $a_6 \cdots a_0$ y $b_6 \cdots b_0$ los bits del primer y segundo codeword, respectivamente, entonces podemos realizar el siguiente desplazamiento cíclico:



Si $a_6 \cdots a_0$ es un codeword válido entonces $b_6 \cdots b_0$ es otro codeword válido.

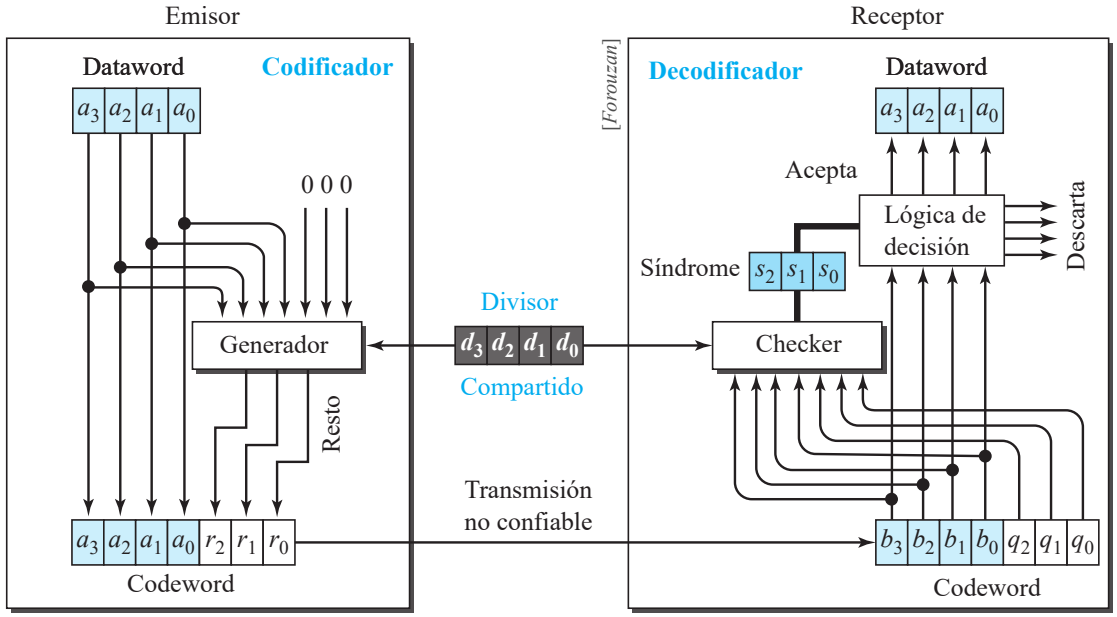
Código CRC *C(7, 4)*

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000**000**	1000	1000**101**
0001	0001**011**	1001	1001**110**
0010	0010**110**	1010	1010**011**
0011	0011**101**	1011	1011**000**
0100	0100**111**	1100	1100**010**
0101	0101**100**	1101	1101**001**
0110	0110**001**	1110	1110**100**
0111	0111**010**	1111	1111**111**

¿Es cíclico? Si. ¿Por qué?

¿Es lineal? Si. ¿Por qué?

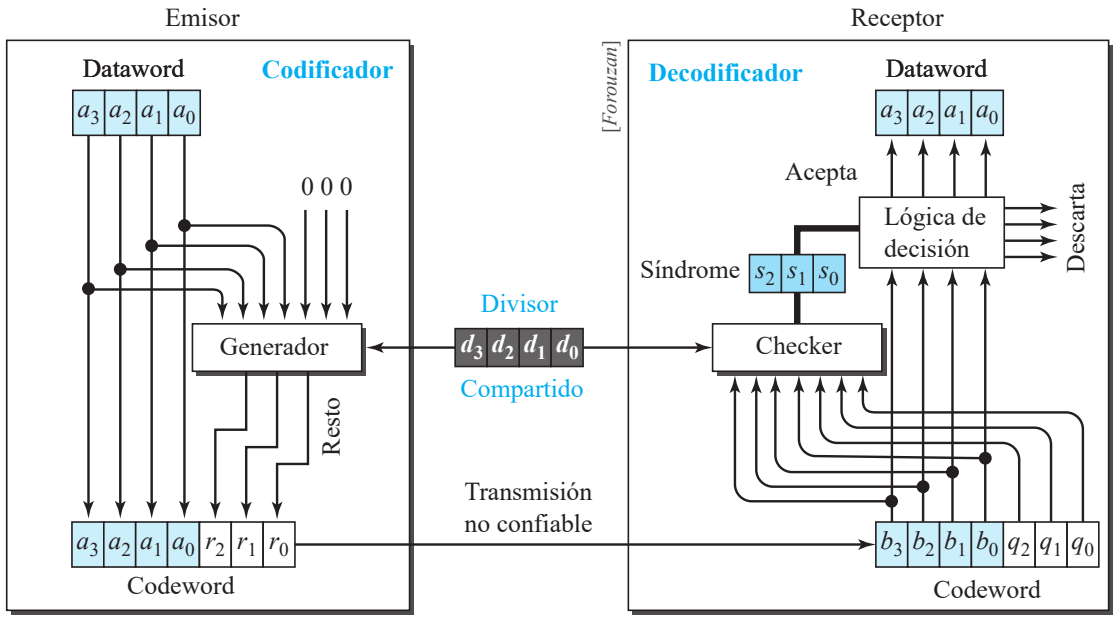
CRC: codificador *C(7,4)*



En el proceso de **codificación**:

- El dataword tiene k bits ($k = 4$) y el codeword tiene n bits ($n = 7$).
- El dataword se incrementa añadiendo $n-k$ bits 0 (en este caso, $n-k = 3$) por la derecha de la palabra e inyectamos los n bits resultantes al generador.
- El generador utiliza un divisor de tamaño $n-k+1$ (en este caso, 4) y divide el dataword (junto con los bits añadidos) entre el divisor mediante división módulo 2.
- El cociente de la división se decarta y el resto ($r_2 r_1 r_0$), denominado **FCS** (*Frame Check Sequence*), se añade al dataword para crear el codeword.
- El codeword se transmite a través del medio no confiable.

CRC: decodificador *C(7,4)*



En el proceso de **decodificación**:

- Se recibe el codeword.
- Una copia de los bits se inyectan al checker (que es el mismo que el generador).
- El resto producido por el checker, de longitud $n-k$ bits (en este caso, 3), se denomina **síndrome**.
- El síndrome se inyecta al analizador de lógica de decisión:
  1. Si los bits de síndrome son todo 0s, los 4 bits más a la izquierda del codeword se aceptan como dataword (es decir, interpretado como no error).
  2. En otro caso, los 4 bits se descartan (se ha producido un error en la transmisión).

CRC: justificación algebraica

Supongamos:

T, trama de n bits a transmitir (*codeword*),

M, mensaje de k bits de datos, son los primeros k bits de T (*dataword*),

F, los últimos $(n-k)$ bits T (es el FCS),

P, patrón de $n-k+1$ bits (*generador*).

Vamos a definir:

$$T = 2^{n-k}M + F$$

es decir, multiplicar M por 2^{n-k} es desplazar a la izquierda $n-k$ bits, añadiendo ceros al resultado. Finalmente, sumar F es concatenar M y F.

El **objetivo** es hacer T divisible por P, es decir, que T/P no tenga resto (i.e. resto 0).

Supongamos que se divide $2^{n-k}M$ entre P:

$$\frac{2^{n-k}M}{P} = Q + \frac{R}{P} \qquad (1)$$

Hay un cociente y un resto. El resto será siempre al menos un bit más corto que el divisor, ya que es división módulo 2.

La secuencia de comprobación de la trama, FCS, será igual al resto. Entonces:

$$T = 2^{n-k}M + R$$

¿Satisface R la condición exigida de que la división T/P tenga resto cero?

Para comprobarlo, consideremos que:

$$\frac{T}{P} = \frac{2^{n-k}M + R}{P} = \frac{2^{n-k}M}{P} + \frac{R}{P} \qquad (2)$$

sustituyendo (1) en (2), y considerando que la suma módulo 2 de números binarios iguales es 0:

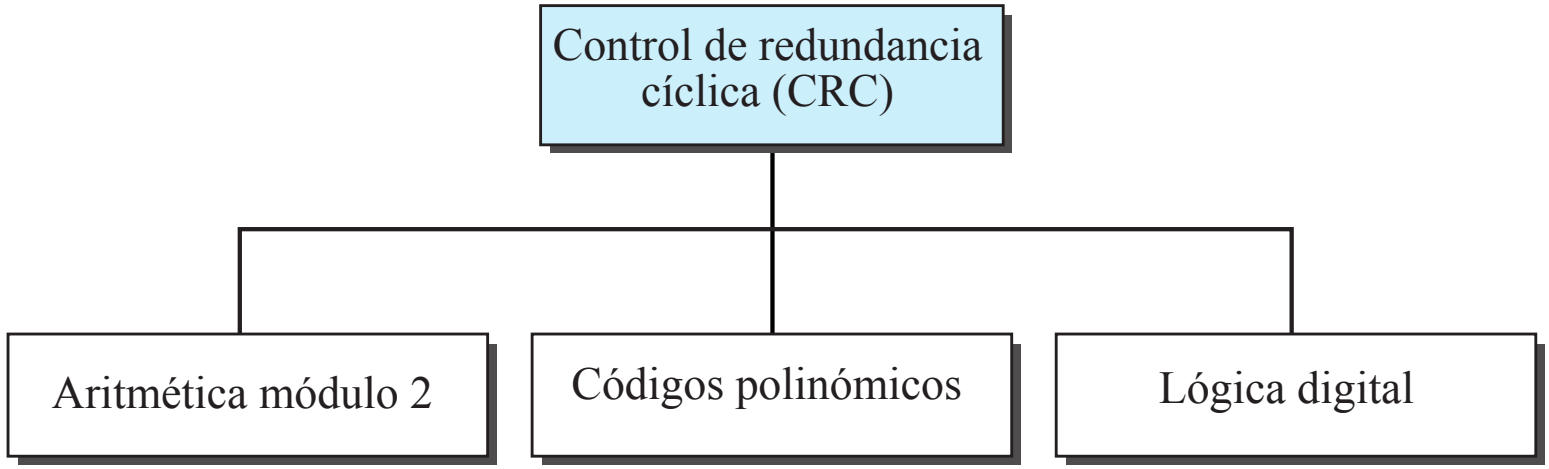
$$\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P} = Q$$

No hay resto, y por tanto, T es divisible por P.

En el emisor, la FCS se genera fácilmente: simplemente se divide $2^{n-k}M$ entre P y se usan los $n-k$ bits del resto como FCS.

En el receptor se divide T entre P y, si no se han producido errores, entonces, el resto será 0.

CRC


Control de redundancia cíclica (CRC)
Aritmética módulo 2
Códigos polinómicos
Lógica digital

CRC: Aritmética módulo 2

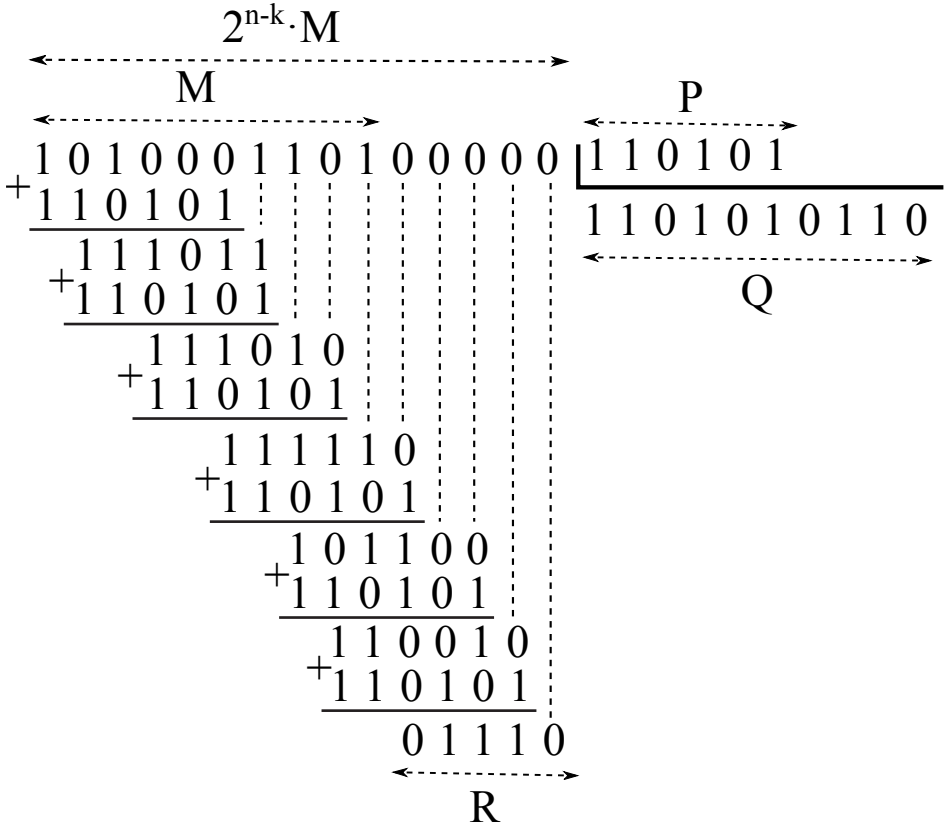
Aritmética módulo 2: ejemplo (emisor)

Sean:

- mensaje $M = 1010001101$ (10 bits)
- patrón $P = 110101$ (6 bits)
- FCS R = a calcular (5 bits)

Por lo tanto:

$$n = 15$$
$$k = 10$$
$$(n - k) = 5$$

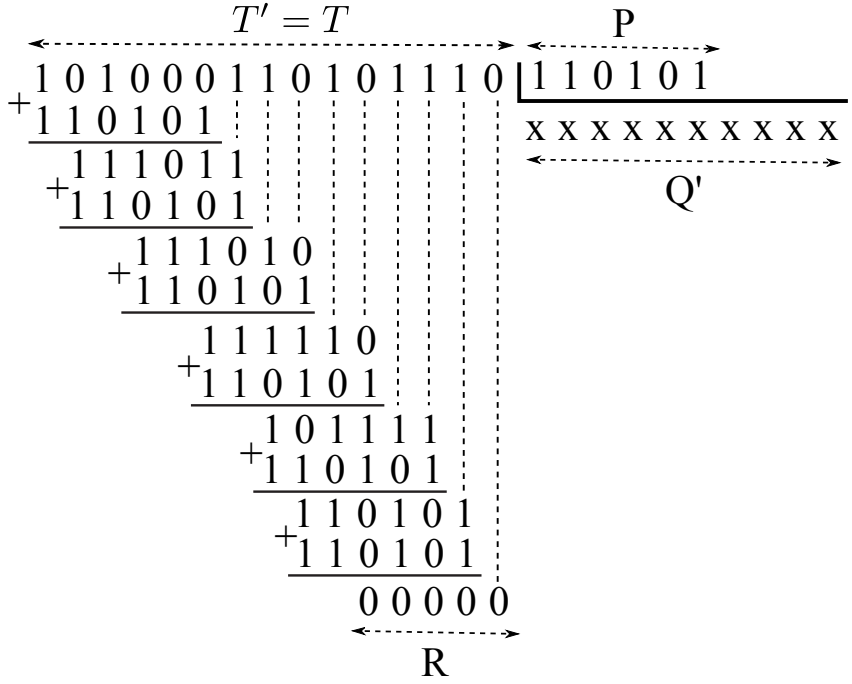


El mensaje M se multiplica por 2^5, resultando 101000110100000, se divide entre P, obteniendo un resto R.

El resto de la división módulo 2 se suma a $2^5 \times M$ y se obtiene $T = 101000110101110$, que es la secuencia transmitida (*codeword*).

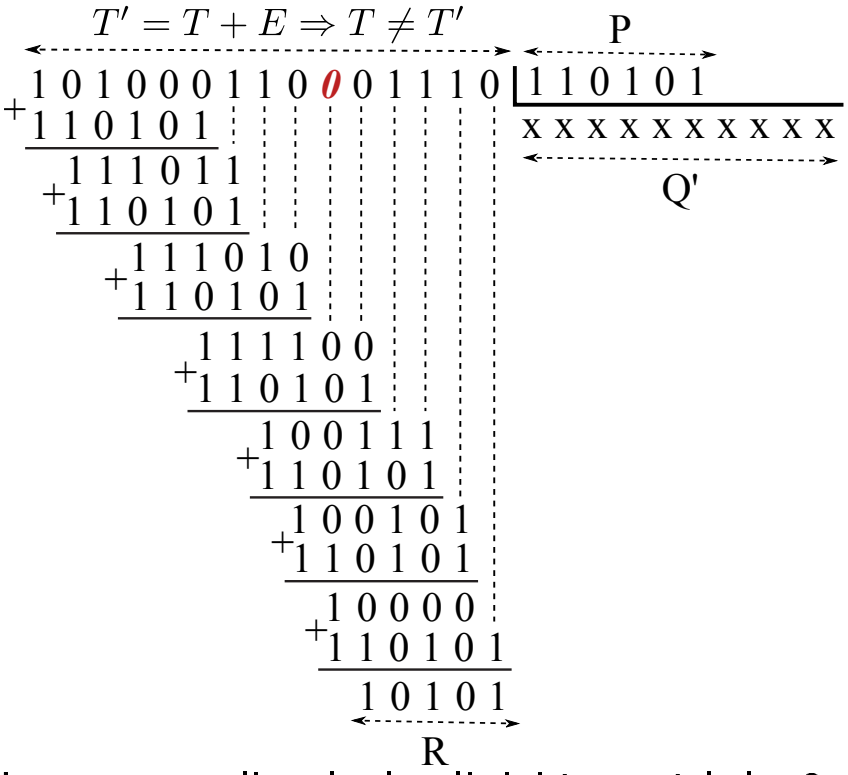
Aritmética módulo 2: ejemplo (receptor)

Si **no hay errores**, la cadena de bits recibida en el receptor T', es la misma que fue generada por el emisor T.



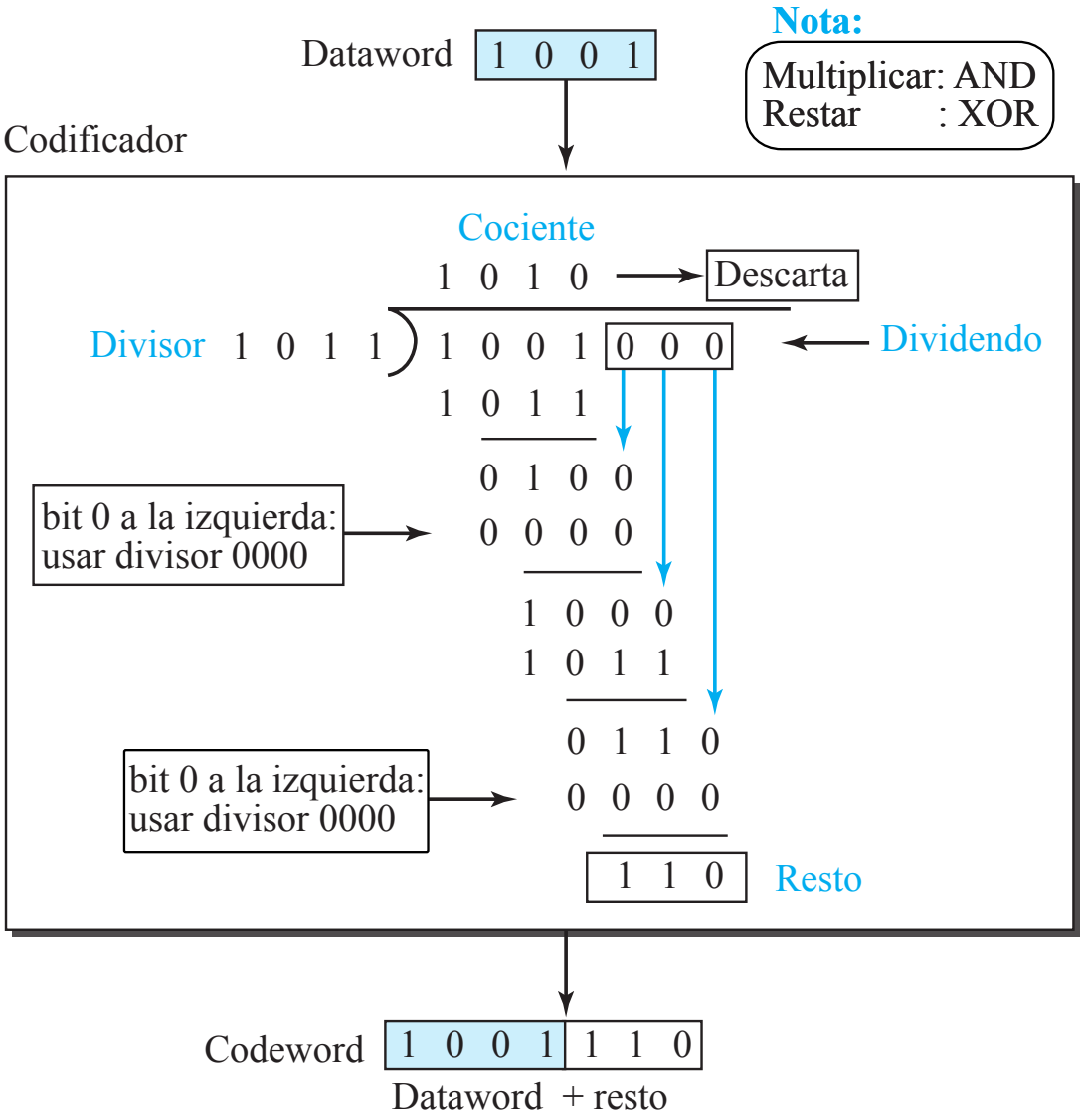
Y como no hay resto, se determina que no se han producido errores de transmisión.

Si se produce **algún error** en la transmisión, tendremos que la trama recibida T' será diferente a T.

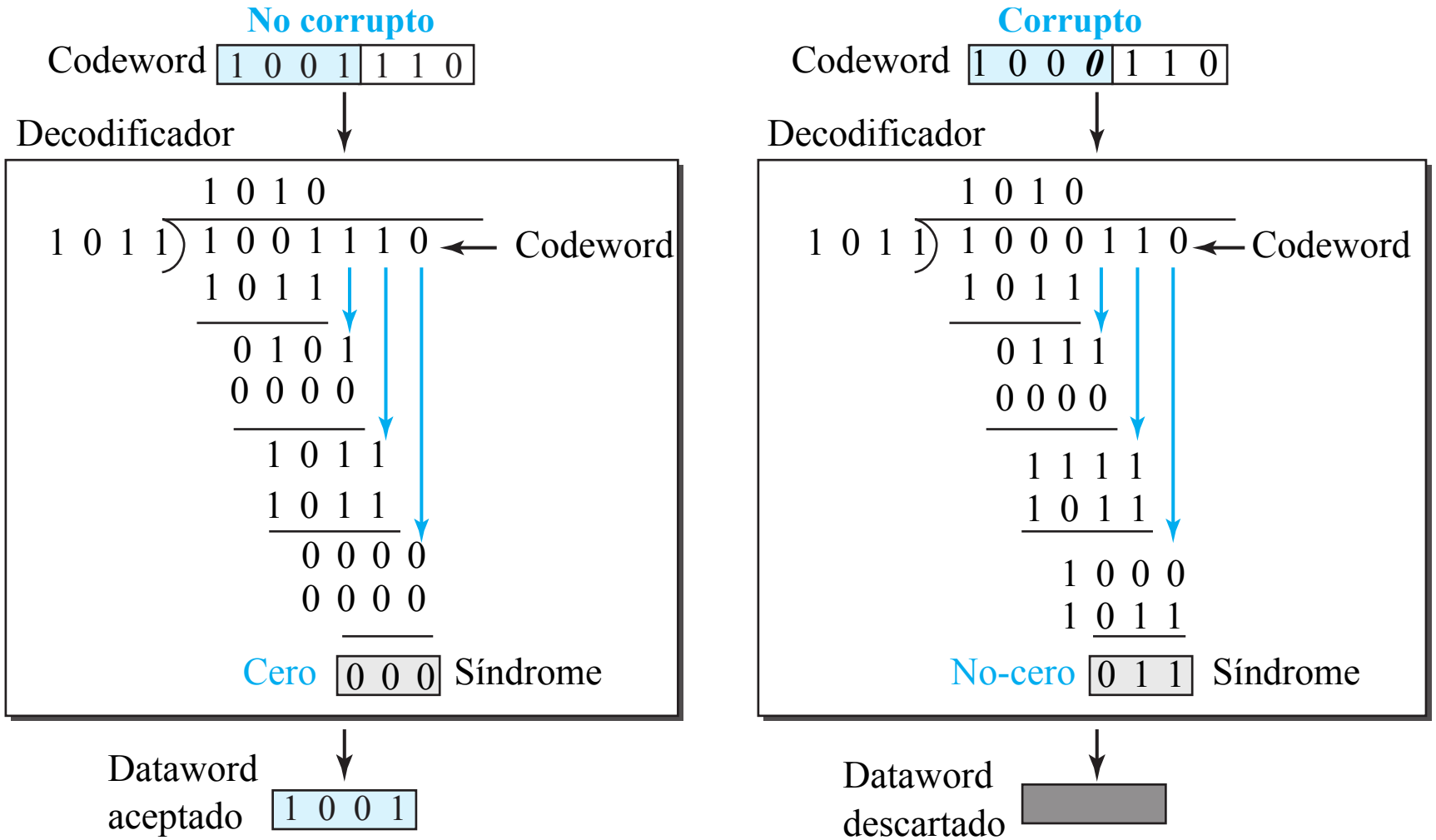


Una vez realizada la división módulo 2, se obtiene $R \neq 0$, por lo tanto, se determina que la cadena recibida es errónea.

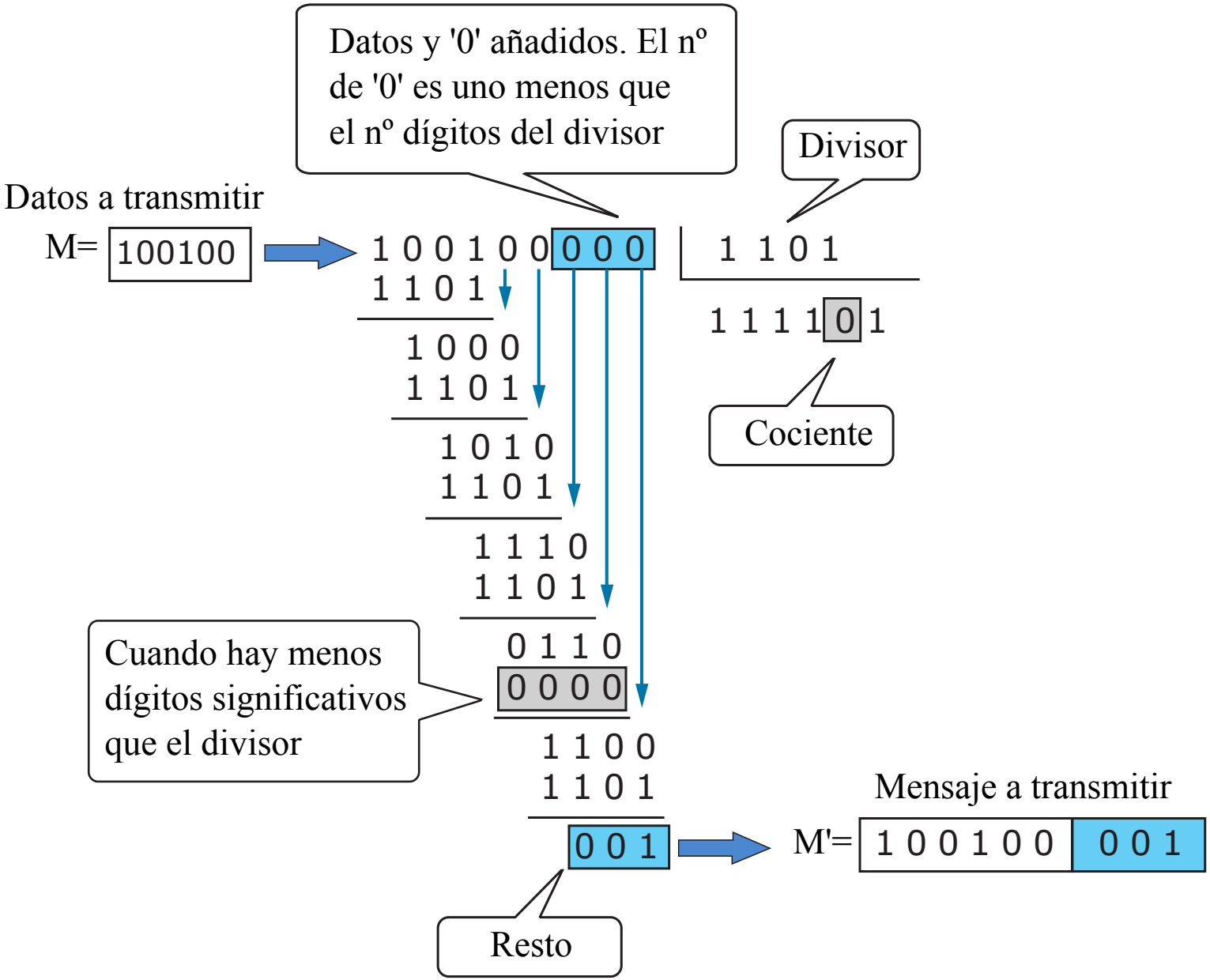
CRC: codificador



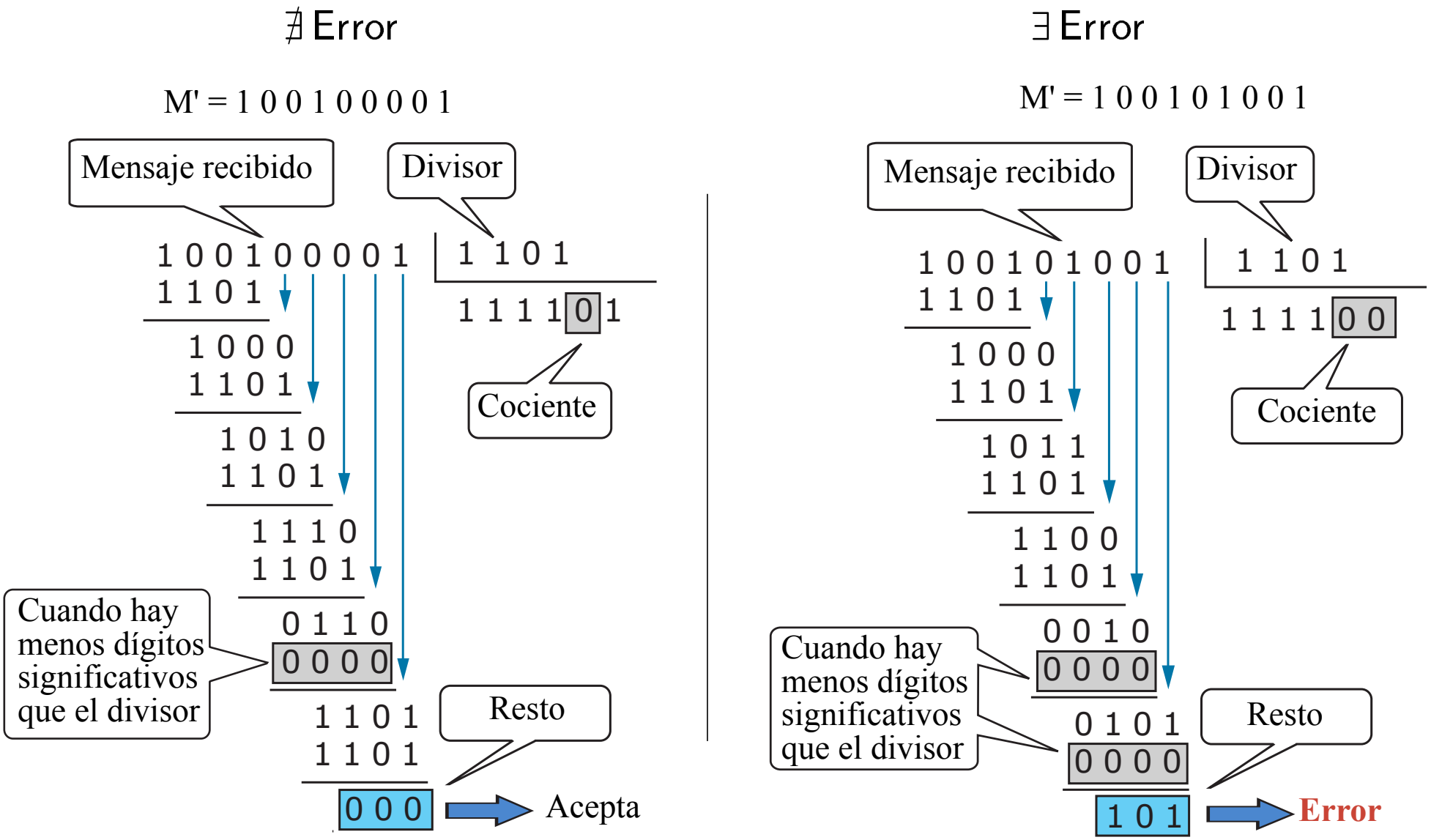
CRC: decodificador



Aritmética módulo 2: emisor (ejemplo 2)

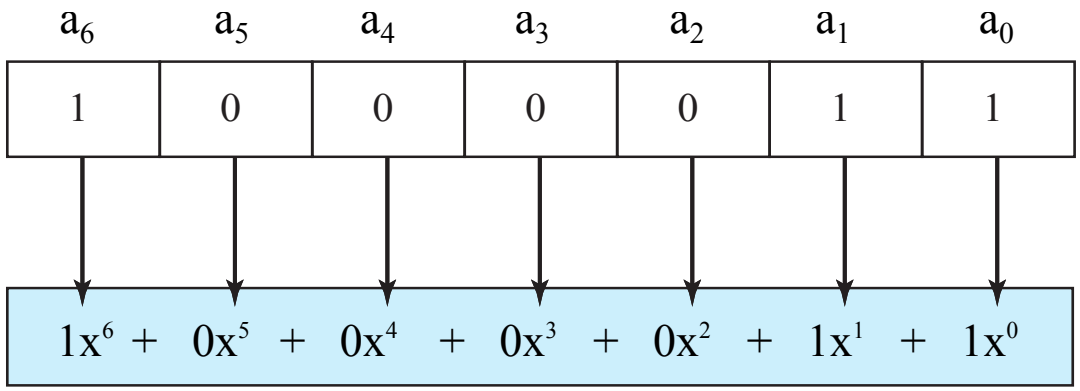


Aritmética módulo 2: receptor (ejemplo 2)

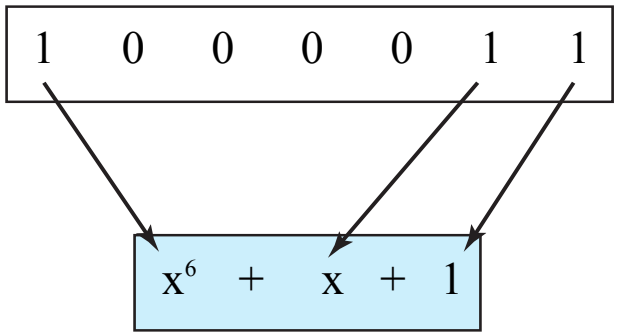


CRC: Códigos polinómicos

Representación polinómica de códigos cíclicos (I)



(a) Patrón binario y polinomial

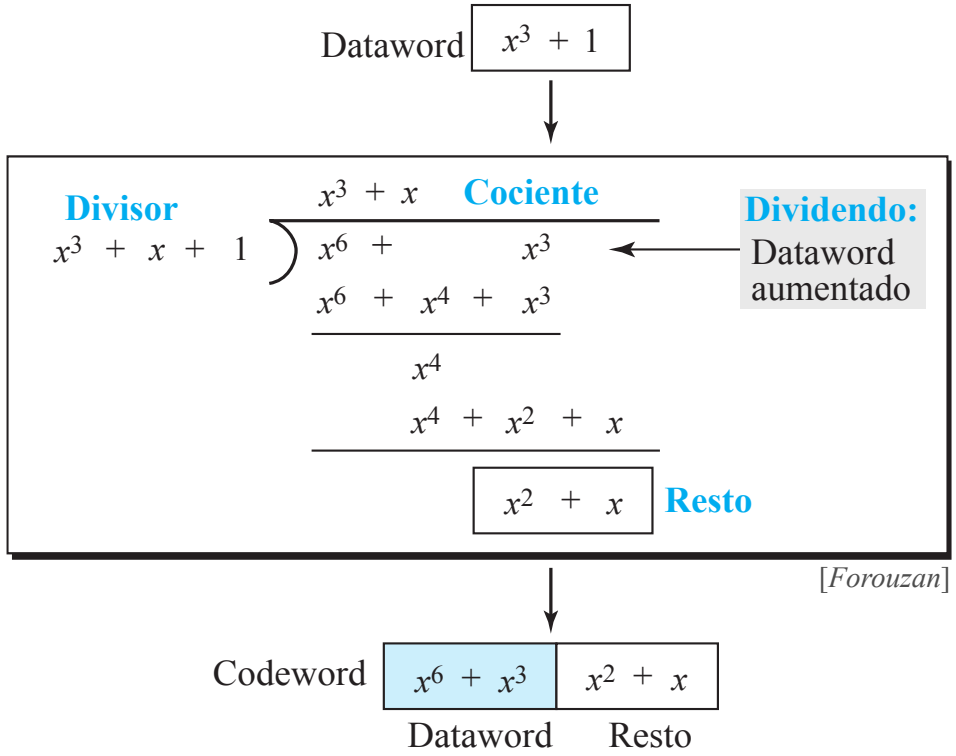


(b) Formato corto

El formato polinómico facilita la representación ya que elimina los bits 0, obteniendo un formato más corto y manejable.

Representación polinómica de códigos cíclicos (II)

Elemento	Binario	Polinómico
Dataword	1001	$x^3 + 1$
Dataword aumentado	1001000	$x^6 + x^3$
Generador	1011	$x^3 + x + 1$
Resto	110	$x^2 + x$
Codeword	1001110	$x^6 + x^3 + x^2 + x$



Las operaciones aritméticas polinómicas se siguen realizando en módulo 2.

Desplazamiento de m bits a la izquierda → multiplicar cada término por x^m.

La sustracción o adición de términos polinómicos → combinación de todos los términos y eliminación de los repetidos.

El divisor en un código cíclico se conoce como **polinomio generador** o **generador**.

Análisis polinómico de códigos cíclicos (I)

Sea:

$M(x)$ el mensaje a transmitir.

$G(x)$ el polinomio generador de grado r.

$T(x)$ la secuencia de bits a transmitir.

$R(x)$ el resto de la división módulo 2 entre el mensaje a transmitir $M(x)$ desplazado r bits y $G(x)$:

$$R(x) = \left| \frac{x^r M(x)}{G(x)} \right|$$

La secuencia a transmitir $T(x)$ se obtiene concatenando $R(x)$ a $x^r M(x)$, por tanto:

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

Por definición, sabemos que $\left| \frac{T(x)}{G(x)} \right| = 0$

Análisis polinómico de códigos cíclicos (II)

Sea $T'(x)$ la secuencia de bits recibida en el receptor y, una vez aplicada la decodificación utilizando el polinomio generador $G(x)$, se pueden producir los siguientes casos:

$$\text{si } \nexists \text{ error} \Rightarrow R'(x) = \left|\frac{T'(x)}{G(x)}\right| = 0$$

$$\text{si } \exists \text{ error} \Rightarrow R'(x) = \left|\frac{T'(x)}{G(x)}\right| \neq 0 \tag{3}$$

Si se produce (3), es decir, el codeword recibido ha sufrido alguna perturbación, entonces la secuencia recibida se puede expresar como concatenación del error producido al codeword original (**ruido aditivo**):

$$T'(x) = T(x) + E(x)$$

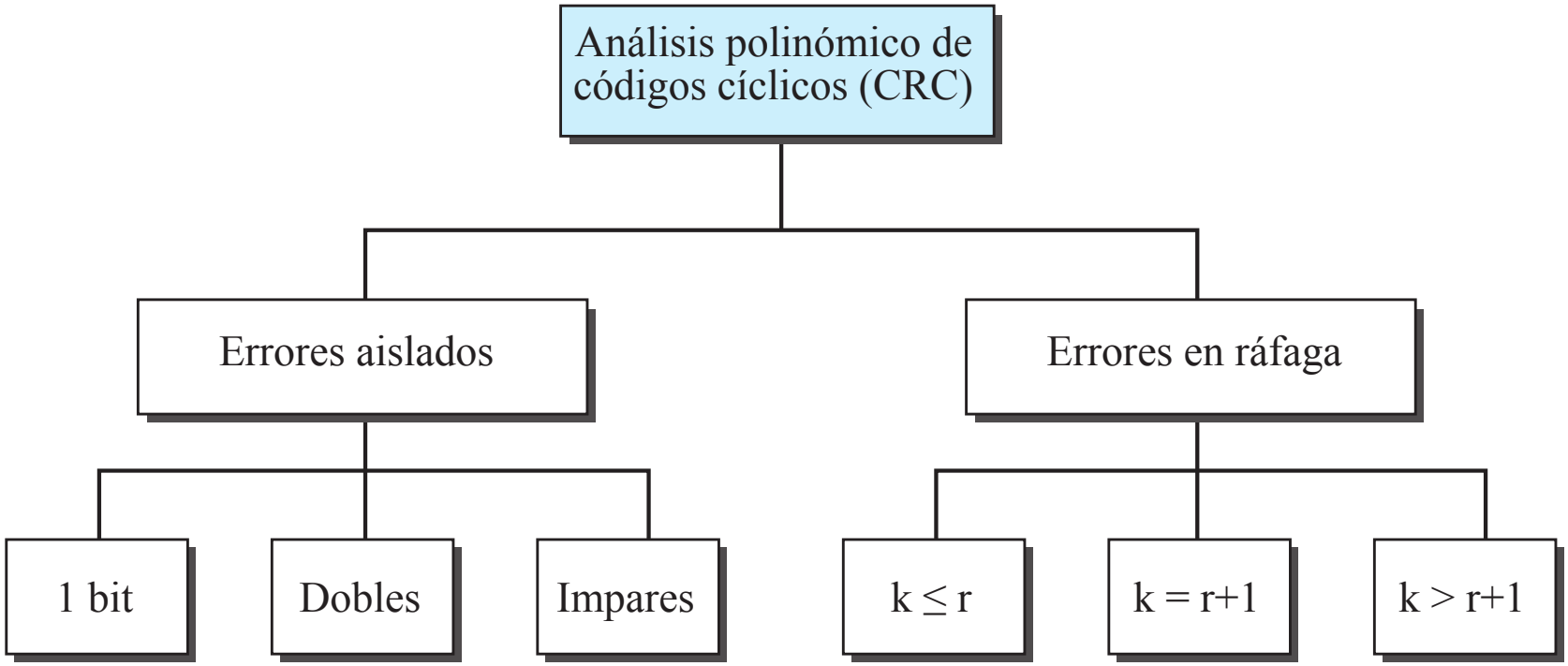
por tanto:

$$R'(x) = \left|\frac{T'(x)}{G(x)}\right| = \left|\frac{T(x) + E(x)}{G(x)}\right| = \left|\frac{T(x)}{G(x)} + \frac{E(x)}{G(x)}\right| \neq 0$$

Sabemos que $\left|\frac{T(x)}{G(x)}\right| = 0$, entonces, para detectar el error se tiene que cumplir que $\left|\frac{E(x)}{G(x)}\right| \neq 0$, en caso contrario, se habrá producido un error, y no habrá sido detectado.

En códigos cíclicos, aquellos patrones de error $E(x)$ que son divisibles por $G(x)$ no son detectados.

Análisis de errores



k ≡ Longitud de ráfaga
r ≡ Grado del polinomio generador

Códigos polinómicos: *errores de 1 bit*

Los errores de 1 bit vienen definidos por $E(x) = x^i$, tal que, $i \geq 0$. Recordemos que para detectar los errores se tiene que cumplir:

$$\left|\frac{E(x)}{G(x)}\right| \neq 0$$

Ejemplo: Dado el error $E = 00010000$, expresado en formato polinomial:

$$E(x) = x^4$$

- Si $G(x)$ es **monomio**, entonces, el error E **no** es detectado, ya que sería divisible por el generador, por ejemplo:

$$G(x) = x^3 \Rightarrow \left|\frac{E(x)}{G(x)}\right| = \left|\frac{x^4}{x^3}\right| = 0$$

- Si $G(x)$ es **superior a monomio** (y contiene el término x^0) entonces, **si** detecta el error, ya que siempre deja resto, por ejemplo:

$$G(x) = x^3 + 1 \Rightarrow \left|\frac{E(x)}{G(x)}\right| = \left|\frac{x^4}{x^3 + 1}\right| = x \neq 0$$

Los errores de 1 bit son detectados si $G(x)$ contiene al menos dos términos y el x^0.

Códigos polinómicos: *errores de 1 bit* (ejemplo)

Sea el generador:

$$G(x) = x^3 + 1$$

Sea el dataword:

$$M(x) = x^4 + x^2 + 1$$

por tanto, el dataword aumentado:

$$x^3 \cdot M(x) = x^7 + x^5 + x^3$$

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + 1 \\ \cline{2-2} x^7 + x^4 & x^4 + x^2 + x + 1 \\ \cline{1-1} x^5 + x^4 + x^3 & \\ x^5 + x^2 & \\ \cline{1-1} x^4 + x^3 + x^2 & \\ x^4 + x & \\ \cline{1-1} x^3 + x^2 + x & \\ x^3 + 1 & \\ \cline{1-1} x^2 + x + 1 & \end{array}$$

Finalmente, el codeword:

$$C(x) = x^7 + x^5 + x^3 + x^2 + x + 1$$

Supongamos se produce error en el 6º bit, reflejado en el patrón de error:

$$E = 00100000$$

y en formato polinómico:

$$E(x) = x^5$$

En relación a la capacidad de detección de $G(x)$, dividimos $E(x)$ entre $G(x)$:

$$\begin{array}{r|l} x^5 & x^3 + 1 \\ \cline{2-2} x^5 + x^2 & x^2 \\ \cline{1-1} x^2 & \end{array}$$

Comprobando que:

$$\left| \frac{E(x)}{G(x)} \right| \neq 0$$

por lo tanto, $E(x)$ no es divisible por $G(x)$, luego, SI detecta el error.

También podemos comprobar que el codeword **distorsionado** por el ruido aditivo:

$$C'(x) = x^7 + x^3 + x^2 + x + 1$$

tampoco es divisible por $G(x)$.

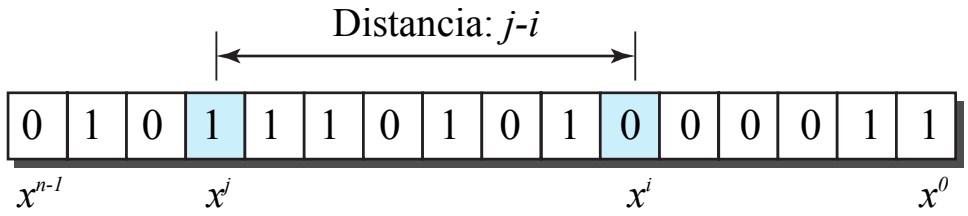
$$\begin{array}{r|l} x^7 + x^3 + x^2 + x + 1 & x^3 + 1 \\ \cline{2-2} x^7 + x^4 & x^4 + x + 1 \\ \cline{1-1} x^4 + x^3 + x^2 + x & \\ x^4 + x & \\ \cline{1-1} x^3 + x^2 + 1 & \\ x^3 + 1 & \\ \cline{1-1} x^2 & \end{array}$$

Se tiene que:

$$\left| \frac{C(x) + E(x)}{G(x)} \right| \neq 0$$

por lo tanto, $C(x) + E(x)$ no es divisible por $G(x)$.

Códigos polinómicos: *errores dobles aislados*



Sea $G(x)$ un generador de grado r y $E(x)$ un patrón de error doble aislado, en las posiciones i, j, tal que $j > i$, donde $(j - i)$ define la **distancia** entre errores:

$$E(x) = x^j + x^i$$

que se puede expresar como:

$$E(x) = x^i(x^{j-i} + 1)$$

El error será detectado si no es divisible por el generador:

$$\left|\frac{E(x)}{G(x)}\right| = \left|\frac{x^i(x^{j-i}+1)}{G(x)}\right| \neq 0 \Rightarrow \begin{cases} \left|\frac{x^i}{G(x)}\right| \neq 0, \text{y además} \\ \left|\frac{x^{j-i}+1}{G(x)}\right| \neq 0 \end{cases}$$

Los patrones de error dobles aislados con distancia $(j - i)$ se detectan si el generador $G(x)$, de grado r, no es monomio, contiene el término x^0 y $j - i < r$, ya que siempre dejará resto.

Errores dobles aislados: ejemplo

Sea el polinomio generador definido por:

$$G(x) = x^4 + x^3 + 1$$

1. Sea el patrón de error doble $E_1 = 010100000 \rightarrow E_1(x) = x^7 + x^5 = x^5(x^2 + 1)$. Para que $E_1(x)$ sea detectado, se tienen que cumplir las siguientes condiciones:

   - Que x^5 no sea divisible por $G(x)$:

     $\left|\frac{x^5}{x^4 + x^3 + 1}\right| \neq 0$ **siempre deja resto**, y por tanto, no es divisible.
   - Que $(x^2 + 1)$ no sea divisible por $G(x)$:

     $\left|\frac{x^2 + 1}{x^4 + x^3 + 1}\right| \neq 0$ como la distancia entre errores $(7 - 5 = 2)$ es **menor** que el grado del generador (4), siempre dejará resto.

   Por lo tanto, el patrón $E_1(x)$ **siempre será detectado** con el polinomio $G(x)$.

2. Sea el patrón de error doble $E_2 = 010000010 \rightarrow E_2(x) = x^7 + x = x(x^6 + 1)$. Para que $E_2(x)$ (mayor distancia de errores que $E_1(x)$) sea detectado, se tienen que cumplir las siguientes condiciones:

   - Que x no sea divisible por $G(x)$:

     $\left|\frac{x}{x^4 + x^3 + 1}\right| \neq 0$ **siempre deja resto**, y por tanto, no es divisible.
   - Que $(x^6 + 1)$ no sea divisible por $G(x)$:

     $\left|\frac{x^6 + 1}{x^4 + x^3 + 1}\right| \neq 0$ como la distancia entre errores $(7 - 1 = 6)$ **no es menor** que el grado del generador (4), **no se tiene la certeza** de que el error sea detectado.

   Por lo tanto, el patrón $E_2(x)$ **no siempre tenemos la certeza** que sea detectado con $G(x)$.

Códigos polinómicos: *errores impares*

Axioma: Si $E(x)$ tiene un nº impar de términos, entonces $(x+1) \notin E(x)$.

Consideremos el error polinomial $E(x)$, con un número impar de errores:

$$E(x) = x^3 + x^2 + x$$

Si

$$G(x) = x + 1$$

entonces, sabemos que:

$$\nexists\, T(x) \text{ tal que } T(x) \times G(x) = E(x)$$

en otras palabras, no existe ningún polinomio que tenga un número impar de términos cuando lo multiplicas por $(x+1)$, es decir:

$$T(x) = \frac{E(x)}{G(x)} \Rightarrow \left|\frac{E(x)}{G(x)}\right| \neq 0$$

Esta observación nos sugiere que $(x+1)$ debe ser un factor del polinomio generador para detectar errores impares. Con la inclusión del factor $(x+1)$ también se detectan los errores de 1 bit, ya que son impares.

Los errores impares se detectan siempre que el polinomio generador contenga el factor $(x+1)$.

Códigos polinómicos: *errores en ráfaga* ($k \leq r$)

Supongamos un patrón de error E, con una ráfaga de k bits de duración e inicio en la posición j:

$$E = 0000\overbrace{1_{j+(k-1)}1_{j+(k-2)}\cdots 1_{j+1}1_j}^{k}00000$$

que expresado en formado polinómico:

$$E(x) = x^{j+(k-1)} + x^{j+(k-2)} + \cdots + x^j = x^j(x^{k-1} + \cdots + 1)$$

Por lo tanto, las condiciones de detección de error son:

$$Si \left|\frac{E(x)}{G(x)}\right| = \left|\frac{x^j(x^{k-1} + \cdots + 1)}{G(x)}\right| \neq 0 \Rightarrow \begin{cases} \left|\frac{x^j}{G(x)}\right| \neq 0 \\ \\ \left|\frac{x^{k-1}+\cdots+1}{G(x)}\right| \neq 0 \end{cases}$$

El generador $G(x)$, de grado r, detecta ráfagas de k bits si $G(x)$ cumple las condiciones de detección de 1 bit y $k \leq r$ (ya que siempre dejaría resto).

Códigos polinómicos: *errores en ráfaga* ($k = r + 1$)

Sean el polinomio generador de grado r definido por:

$$G(x) = x^r + x^{r-1} + \cdots + x + x^0$$

y una ráfaga de longitud k, a partir del bit j, y donde $k = r + 1$:

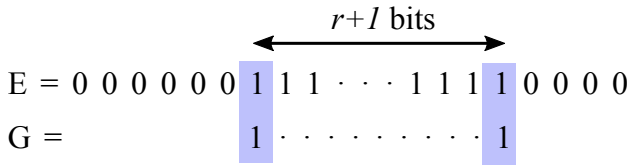
$$E(x) = x^{j+(k+1)} + x^{j+k} + \cdots + x^{j+1} + x^j$$

A nivel de bits, sabemos que $G(x)$ empieza y acaba en 1, por lo tanto:

$$G = \overbrace{1 \cdots\cdots\cdots\cdots 1}^{r+1}$$

$$G(x) = x^r + \cdots\cdots\cdots + x^0$$

$$E = 0000\overbrace{1111\cdots1111}^{k=r+1}00000$$



Si el resto de bits de $G(x)$ fueran 1, entonces tendríamos que $E(x) = G(x)$, y por tanto, $\left|\frac{E(x)}{G(x)}\right| = 0$, en cuyo caso, el error no sería detectado.

Por tanto, la expectativas de detección de error hay que expresarlas **en términos de probabilidad** y en función del tamaño del vector $G(x)$, definido por r.

¿Cuál es la probabilidad p de que $E(x)$ y $G(x)$ sean diferentes?

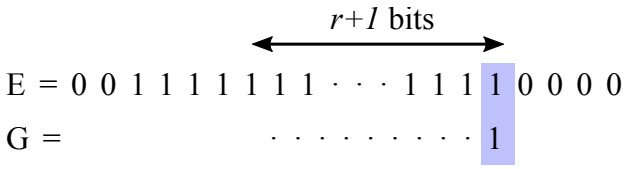
Por definición, los bits primero y último de $E(x)$ y $G(x)$ son 1, entonces, los restantes $(r+1)-2$ bits serán iguales con probabilidad q (siendo p y q, complementarios):

$$q = \left(\frac{1}{2}\right)^{(r+1)-2} = \frac{1}{2^{r-1}} \rightarrow p = 1 - q = 1 - \frac{1}{2^{r-1}}$$

$G(x)$, de grado r, captura ráfagas de longitud $k = (r + 1)$ bits con probabilidad $p = 1 - 1/2^{r-1}$.

Códigos polinómicos: *errores en ráfaga* ($k > r + 1$)

La longitud de ráfaga es mayor que la longitud del generador, es decir, $k > r + 1$, por lo que sólamente podemos asegurar que coincidirán en un sólo bit 1:



¿Cuál es la probabilidad q de que el resto de r bits sean iguales?

Suponiendo que todos los patrones de bits son igualmente probables, y que el sólamente el último bit de ambas cadenas es igual, entonces, la probabilidad q de que un trozo de $E(x)$ sea idéntica a toda la secuencia $G(x)$:

$$q = \left(\frac{1}{2}\right)^{(r+1)-1} = \frac{1}{2^r}$$

Entonces, la probabilidad de que ambas cadenas no sean iguales, y por tanto, el error sea dectectado:

$$p = 1 - q = 1 - \frac{1}{2^r}$$

El generador $G(x)$, de grado r, detecta ráfagas de longitud $k > (r + 1)$ bits con probabilidad $p = 1 - 1/2^r$.

Polinomios estándard

Nombre	Polinomio	Tecnología
CRC-8	$x^8 + x^2 + x + 1$ 1 0 0 0 0 0 1 1 1	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ 1 1 0 0 0 1 1 0 1 0 1	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1	LANs

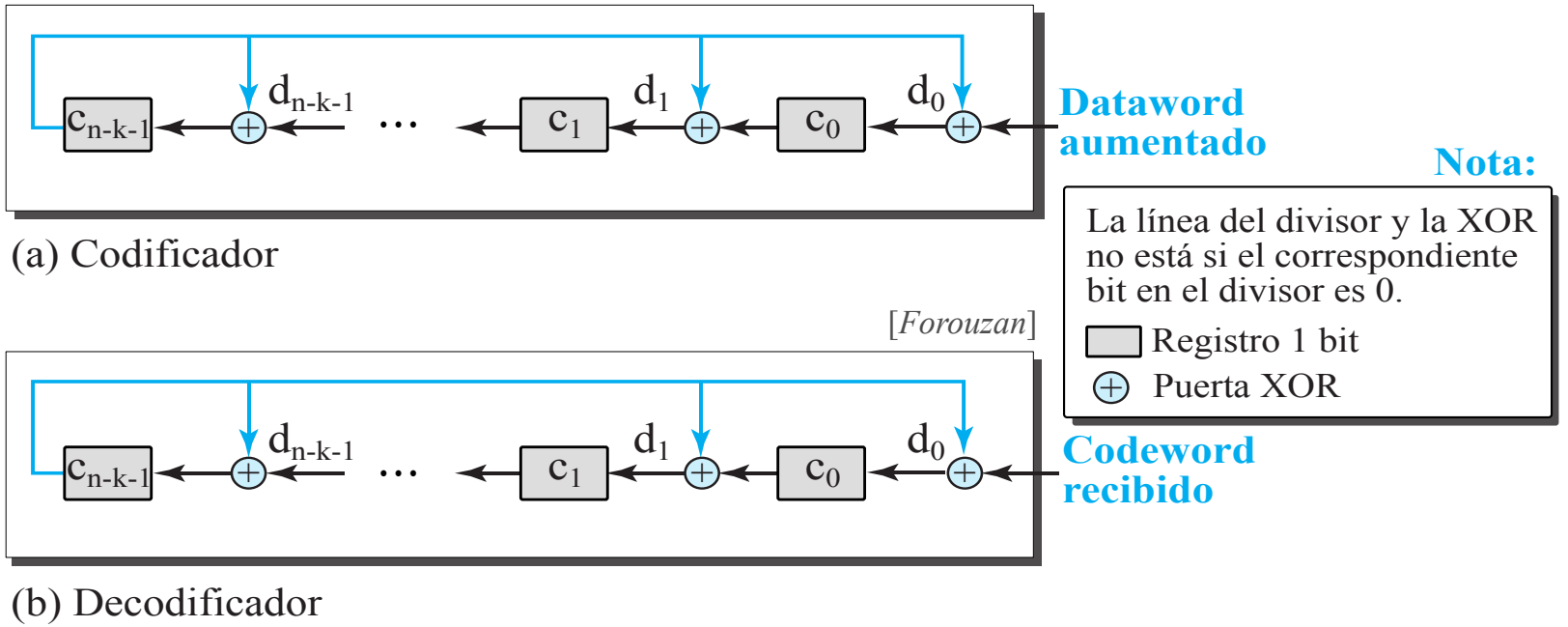
[*Forouzan*]

Los códigos cíclicos han sido estándarizados en multitud de tecnologías debido a sus características:

- Buen rendimiento en la detección de errores simples, dobles, impares y ráfagas.
- Fácil implementación en hardware y software.
- Especialmente indicado para su implementación en hardware (sistemas digitales).

CRC: Lógica digital

Implementación digital de códigos cíclicos



(a) Codificador

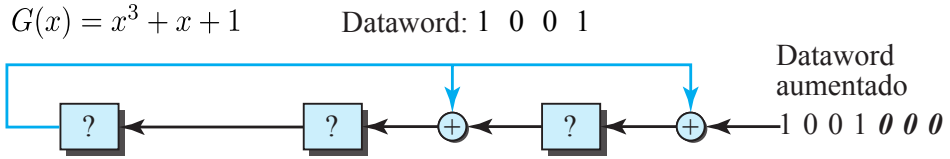
[*Forouzan*]

(b) Decodificador

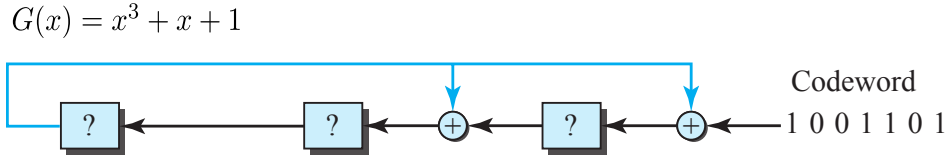
A partir del **polinomio generador**, se diseña el código cíclico con las siguientes especificaciones:

- Se dispone de $(n-k)$ **registros de desplazamientos de 1-bit** en el codificador y decodificador.
- Pueden tener hasta $(n-k)$ **puertas XOR**, pero los divisores tienen, normalmente, varios 0s en sus patrones, lo que reduce drásticamente el nº de puertas.

En el **codificador**, una vez han sido inyectados los n bits del dataword aumentado, el síndrome estará ubicado en los diferentes registros de 1-bit.

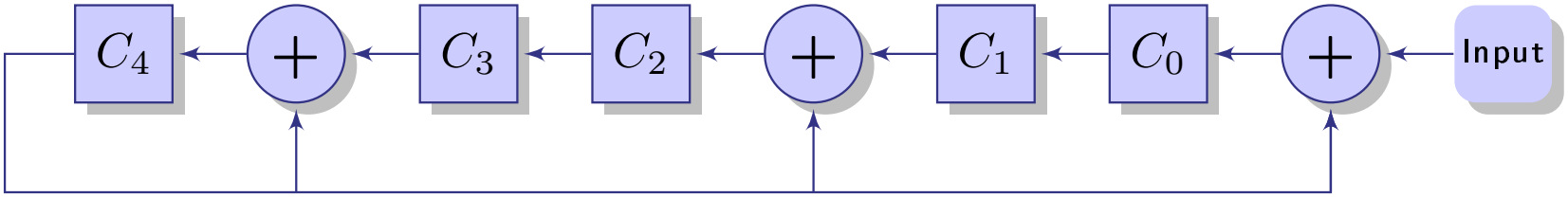


En el **decodificador**, una vez es inyectado el codeword (n pasos), los registros de 1-bit contienen el síndrome.



Obtención de codeword (emisor/codificador)

$G(x) = x^5 + x^4 + x^2 + 1$ Dataword = 1010001101



t	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3$	$C_4 \oplus C_1$	$C_4 \oplus Input$	Input
	0	0	0	0	0	0	0	1	1
1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	0	0	1	1	1
3	0	0	1	0	1	0	0	0	0
4	0	1	0	1	0	1	1	0	0
5	1	0	1	0	0	1	1	1	0
6	1	1	1	0	1	0	1	0	1
7	0	1	1	1	0	1	1	1	1
8	1	1	1	0	1	0	1	1	0
9	0	1	1	1	1	1	1	1	1
10	1	1	1	1	1	0	0	1	0
11	0	1	0	1	1	1	1	0	0
12	1	0	1	1	0	1	0	1	0
13	1	1	0	0	1	0	1	1	0
14	0	0	1	1	1	0	1	0	0
15	0	1	1	1	0				

Resto

Dataword aumentado

Codeword = 1010001101011110

Chequeo de codeword *sin* error (receptor/decodificador)

Codeword = 1010001101011110

t	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3$	$C_4 \oplus C_1$	$C_4 \oplus Input$	Input
	0	0	0	0	0	0	0	1	1
1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	0	0	1	1	1
3	0	0	1	0	1	0	0	0	0
4	0	1	0	1	0	1	1	0	0
5	1	0	1	0	0	1	1	1	0
6	1	1	1	0	1	0	1	0	1
7	0	1	1	1	0	1	1	1	1
8	1	1	1	0	1	0	1	1	0
9	0	1	1	1	1	1	1	1	1
10	1	1	1	1	1	0	0	1	0
11	0	1	0	1	1	1	1	1	1
12	1	0	1	1	1	1	0	0	1
13	1	1	0	1	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0				

Codeword

No hay residuo

Dataword = 1010001101

Chequeo de codeword *con* error (receptor/decodificador)

Codeword = 1010001101010110

t	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3$	$C_4 \oplus C_1$	$C_4 \oplus Input$	Input
	0	0	0	0	0	0	0	1	1
1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	0	0	1	1	1
3	0	0	1	0	1	0	0	0	0
4	0	1	0	1	0	1	1	0	0
5	1	0	1	0	0	1	1	1	0
6	1	1	1	0	1	0	1	0	1
7	0	1	1	1	0	1	1	1	1
8	1	1	1	0	1	0	1	1	0
9	0	1	1	1	1	1	1	1	1
10	1	1	1	1	1	0	0	1	0
11	0	1	0	1	1	1	1	0	0
12	1	0	1	1	1	1	0	0	1
13	1	1	0	0	0	0	1	0	1
14	0	0	1	0	0	0	0	0	0
15	0	1	0	0	0				

Codeword

Residuo

Dataword descartado

Corrección de errores

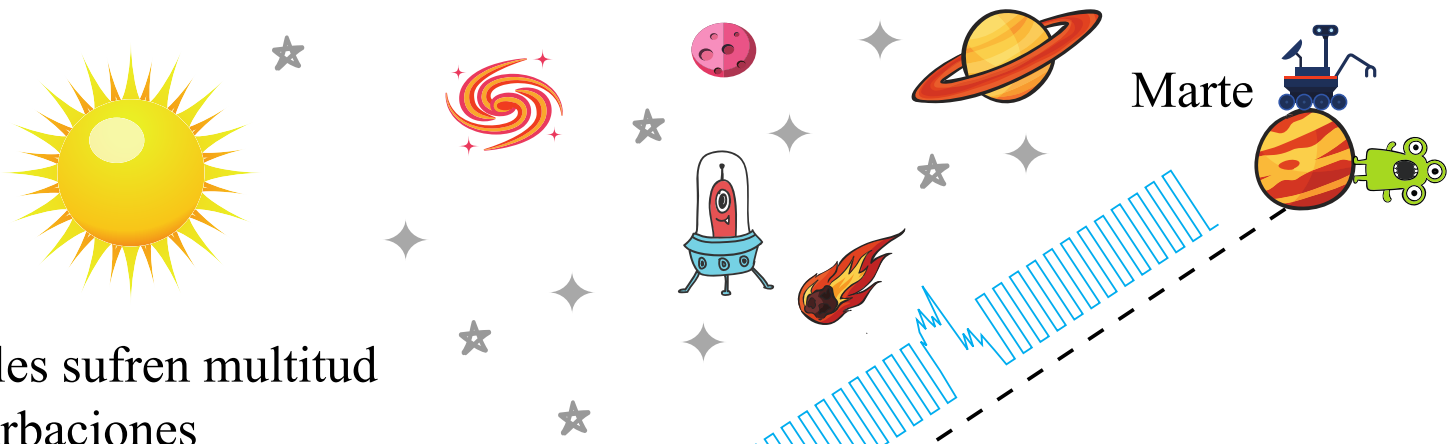
Tipos de códigos

ARQ
(*Automatic Repeat Request*)

BEC
(*Backward Error Control*)

FEC
(*Forward Error Control*)

Corrección de errores hacia adelante

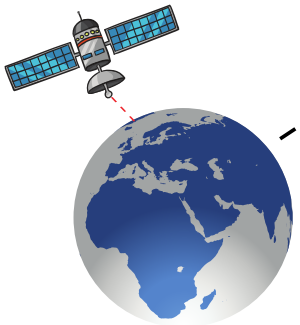


La señales sufren multitud de perturbaciones

¿Qué ocurre si se produce un error?
¿Se solicita retransmisión de mensaje?
¿Y si el mensaje es una alarma?

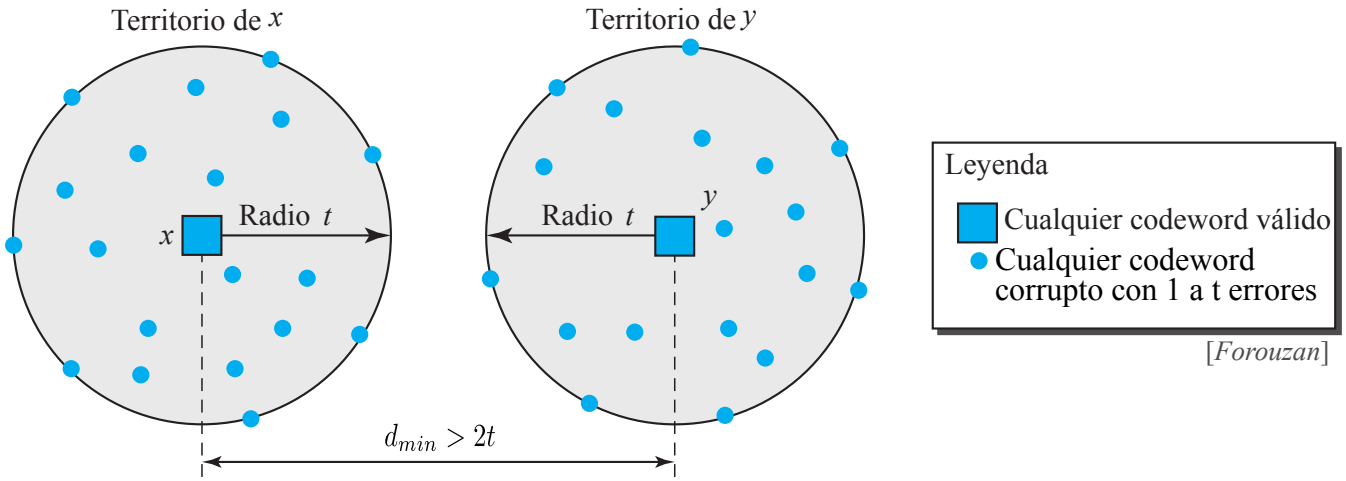
$t_p \uparrow\uparrow\uparrow$

La señal electromagnética tarda 10-15 minutos-luz en recorrer la distancia



Código corrector de error (FEC):
Consiste en incluir bits redundantes en los datos a transmitir. Esta información permitirá al receptor determinar la ubicación del bit (o bits) erróneos, de forma que posibilite su corrección en el destino, ya que la retransmisión es inviable.

Distancia Hamming en correccion de errores



La distancia Hamming en corrección de errores es mayor que en detección de errores.

Provoca una gran sobrecarga en los protocolos (*overhead*).

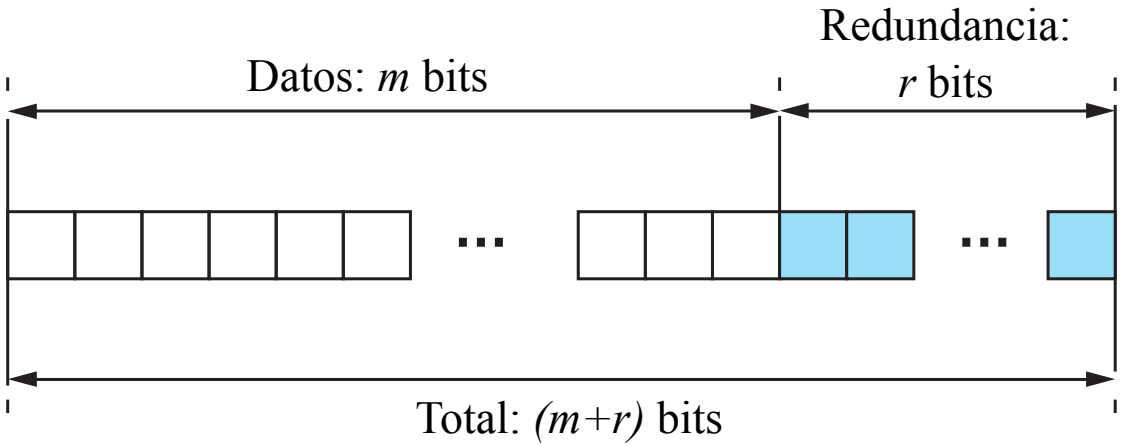
Aplicación es exclusiva de entornos muy concretos: sistemas en tiempo real, comunicaciones espaciales, canales simplex, etc.

Un código con capacidad de corrrección de hasta t errores, necesita una distancia mínima:

$$d_{min} = 2t + 1$$

Diseño de codificador Hamming (I)

Codificador Hamming con capacidad para detectar 1 error:



En el total de $m+r$ bits pueden suceder las siguientes casos:

- Error en posición 1
- Error en posición 2
- ...
- Error en posición $i \in [1, m]$
- Error en posición $j \in [m+1, r]$
- No error

En total, pueden suceder:

$$m + r + 1$$

posibles casos ($m + r$ *de error y* 1 *de no error*) que deben se codificados con r bits.

Por lo tanto, tiene que cumplir:

$$\boxed{2^r \geq m + r + 1}$$

Diseño de codificador Hamming (II)

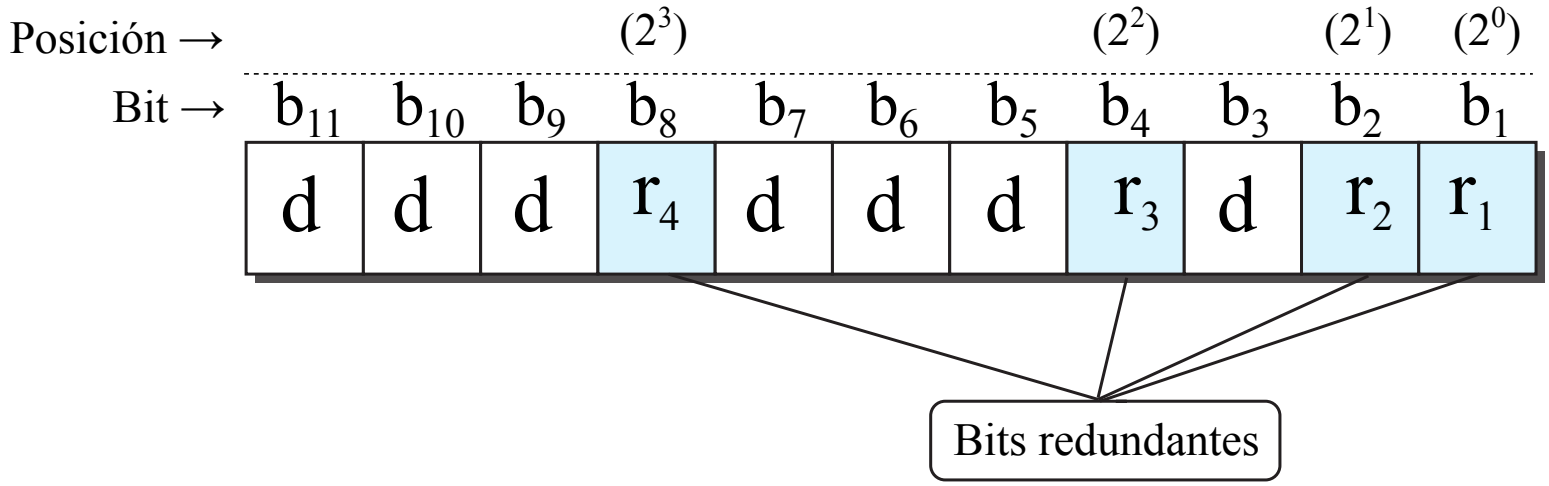
Si definimos:

$b_i \rightarrow$ bit posición i

$r_j \rightarrow$ bit redundante j

$2^k \rightarrow$ posición 2^k de la trama

$d \rightarrow$ bit de datos



Los r bits redundantes se colocan en las posiciones potencia de 2.

El resto de m bits de datos, se reparten por las posiciones libres de la trama.

Diseño de codificador Hamming (III)

Si definimos:

$b_i \rightarrow$ bit posición i.

$\oplus \rightarrow$ XOR.

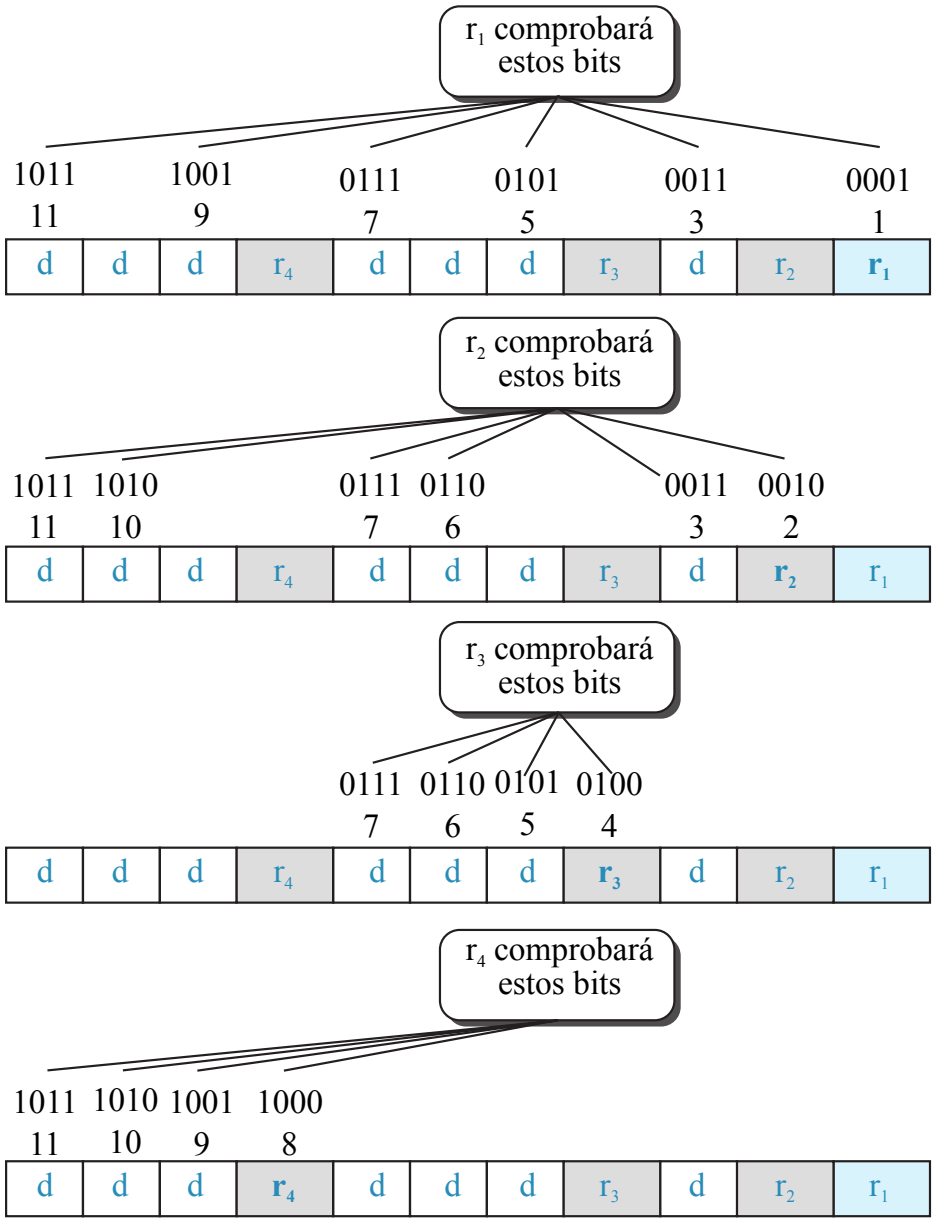
entonces, se definen los siguientes **bits de paridad** (redundantes)

$$r_1 = b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11}$$

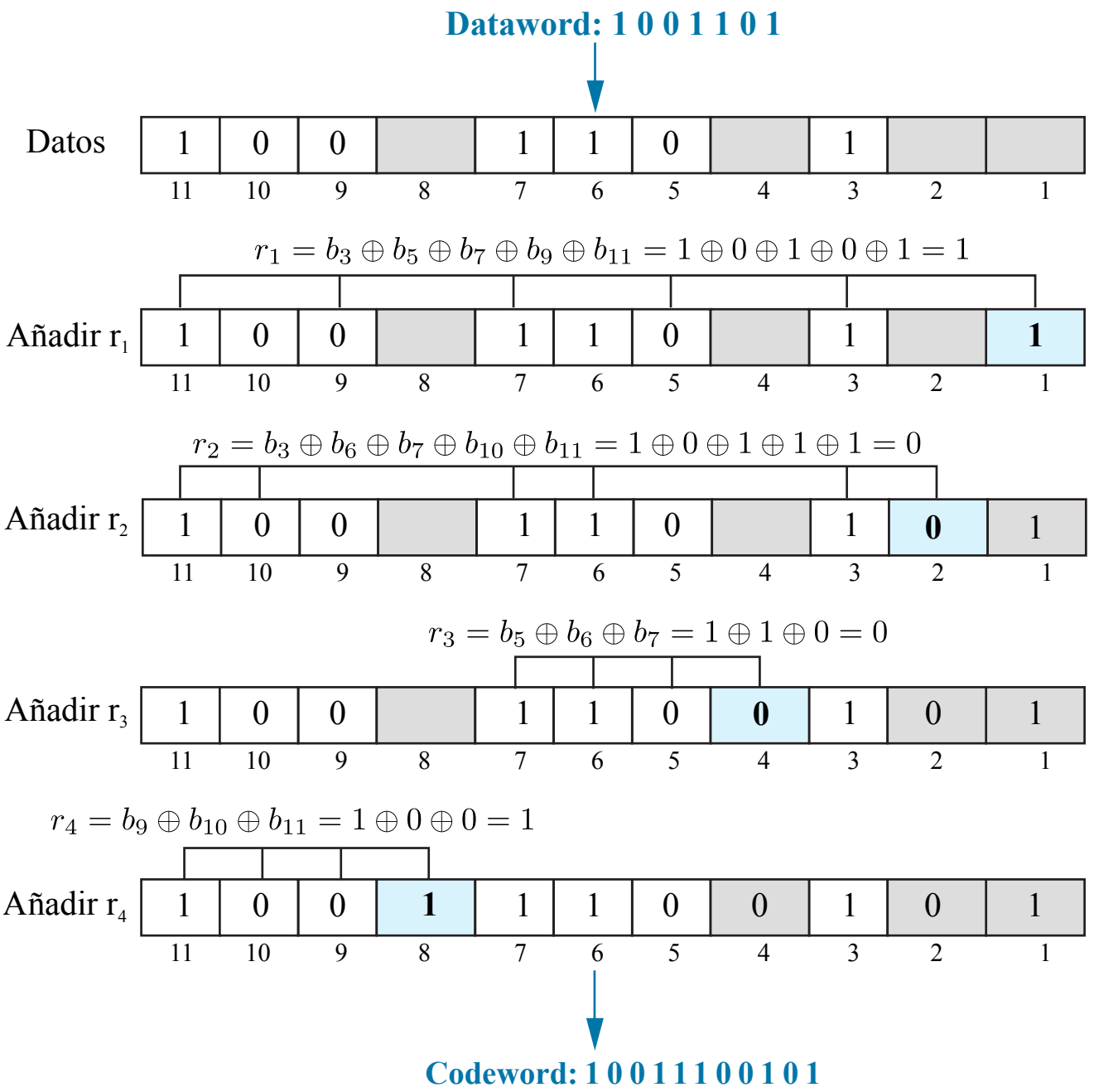
$$r_2 = b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11}$$

$$r_3 = b_5 \oplus b_6 \oplus b_7$$

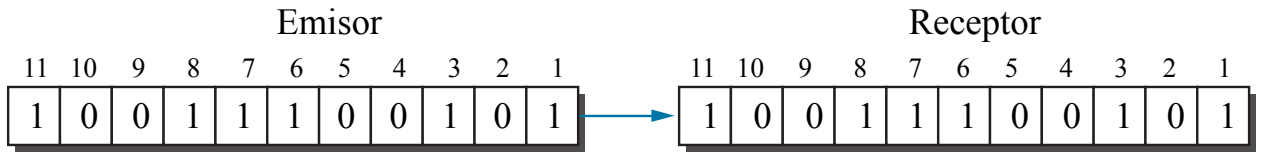
$$r_4 = b_9 \oplus b_{10} \oplus b_{11}$$

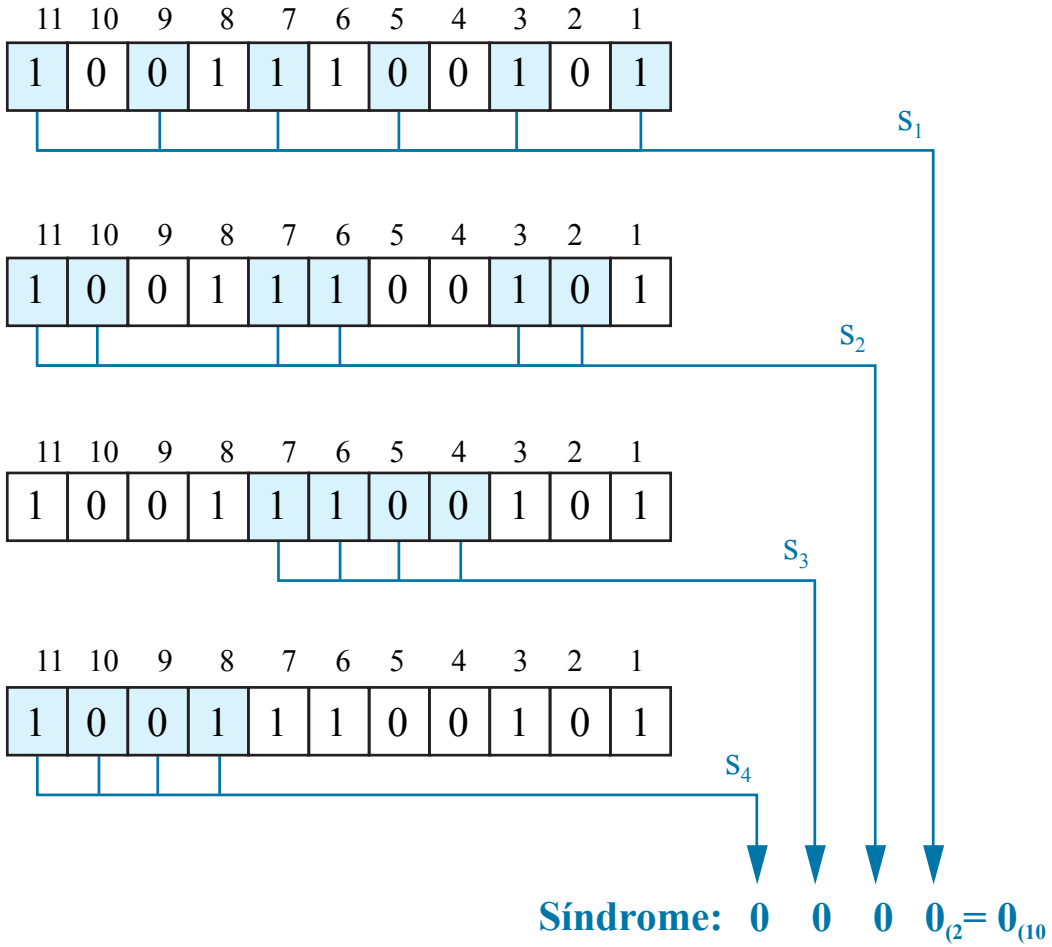


Diseño de codificador Hamming (IV)



Decodificador Hamming (sin) error (receptor)





Se calculan los bits de **síndrome**:

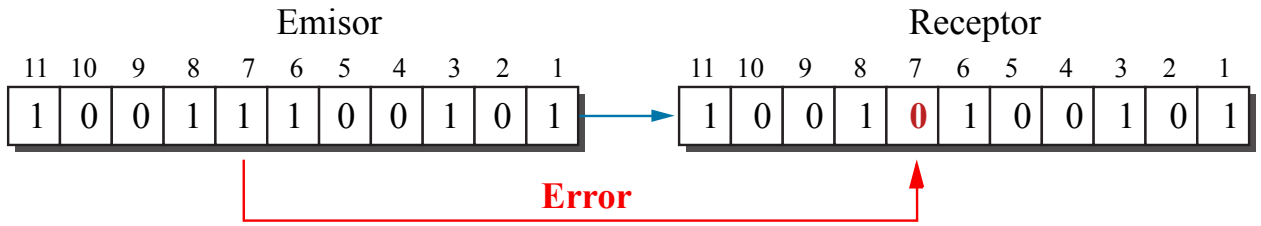
$$
\begin{aligned}
s_1 &= b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \\
&= 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \\
&= 0
\end{aligned}
$$

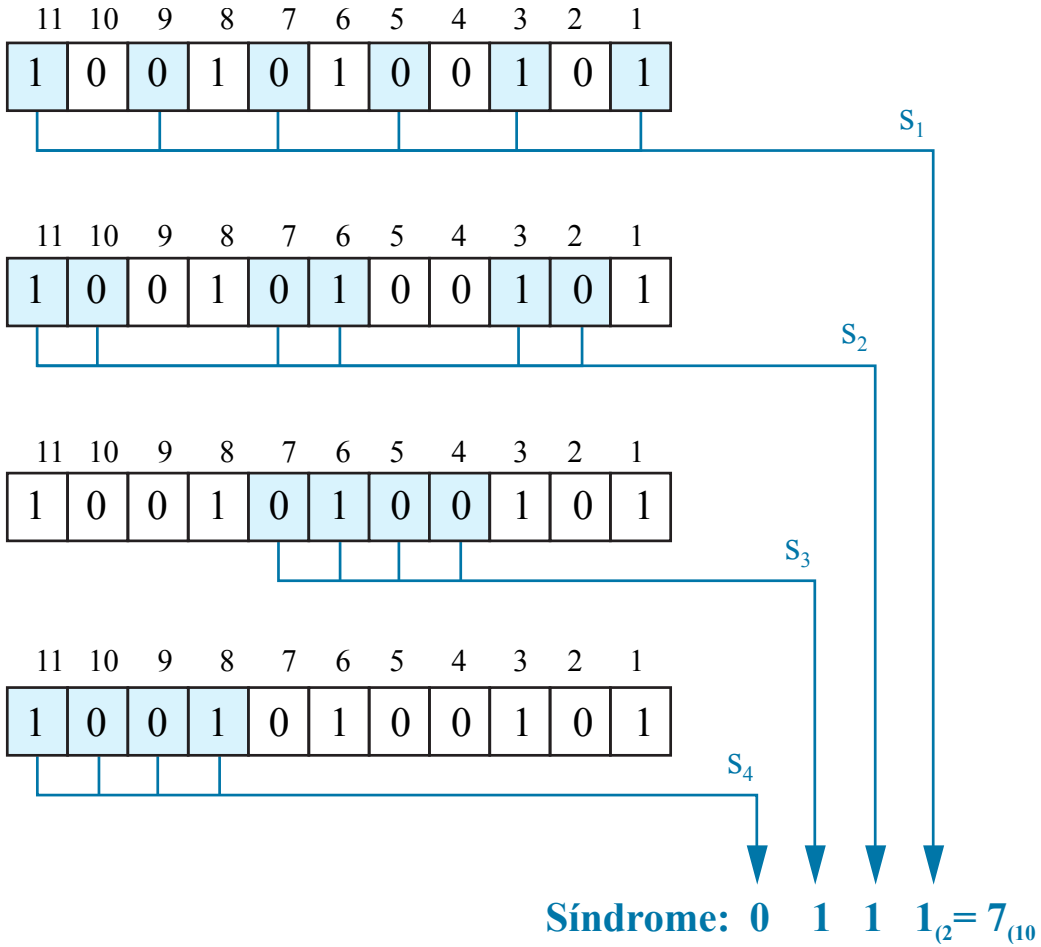
$$
\begin{aligned}
s_2 &= b_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \\
&= 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
s_3 &= b_4 \oplus b_5 \oplus b_6 \oplus b_7 \\
&= 0 \oplus 0 \oplus 1 \oplus 1 \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
s_4 &= b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \\
&= 1 \oplus 0 \oplus 0 \oplus 1 \\
&= 0
\end{aligned}
$$

Decodificador Hamming (con) error (receptor)





Se calculan los bits de **síndrome**:

$$
\begin{aligned}
s_1 &= b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \\
&= 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
s_2 &= b_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \\
&= 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
s_3 &= b_4 \oplus b_5 \oplus b_6 \oplus b_7 \\
&= 0 \oplus 0 \oplus 1 \oplus 0 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
s_4 &= b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \\
&= 1 \oplus 0 \oplus 0 \oplus 1 \\
&= 0
\end{aligned}
$$

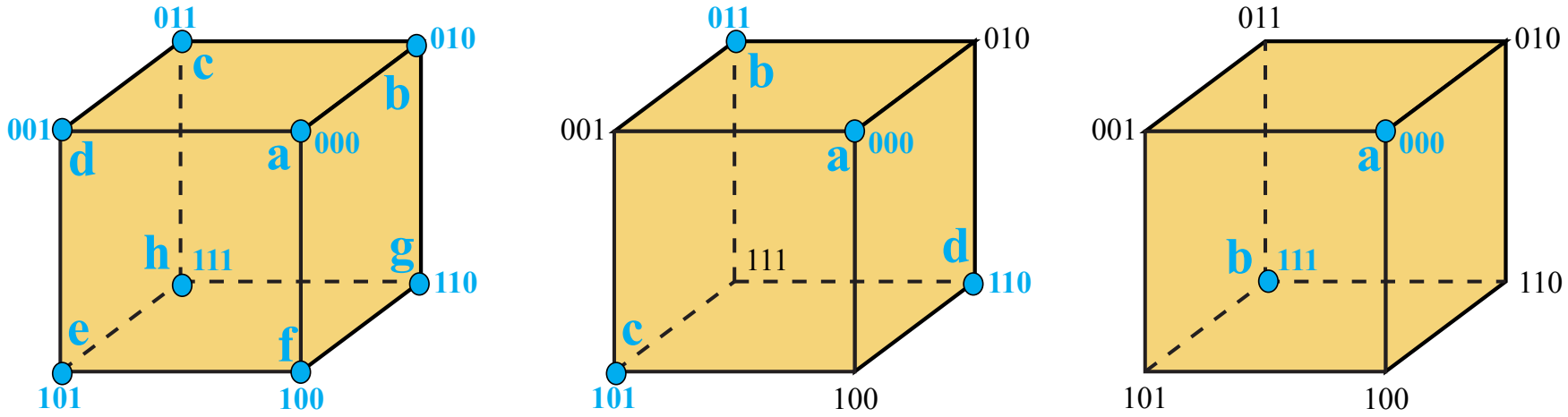
Código Hamming $C(n, k)$

Dado el siguiente código Hamming $C(9, 5)$:

Dataword	*Codeword*	*Dataword*	*Codeword*
00000	**0**0000**0**00	10000	**1**1000**0**01
00001	**0**0000**1**11	10001	**1**1000**1**10
00010	**0**0001**1**01	10010	**1**1001**1**00
00011	**0**0001**1**10	10011	**1**1001**1**11
00100	**0**0010**1**10	10100	**1**1010**1**11
00101	**0**0010**1**01	10101	**1**1010**1**00
00110	**0**0011**0**11	10110	**1**1011**0**10
00111	**0**0011**0**00	10111	**1**1011**0**01
01000	**0**0100**1**11	11000	**1**1100**1**10
01001	**0**0100**1**00	11001	**1**1100**1**01
01010	**0**0101**0**10	11010	**1**1101**0**11
01011	**0**0101**0**01	11011	**1**1101**0**00
01100	**0**0110**0**01	11100	**1**1110**0**00
01101	**0**0110**0**10	11101	**1**1110**0**11
01110	**0**0111**1**00	11110	**1**1111**1**01
01111	**0**0111**1**11	11111	**1**1111**1**10

$d_{min} = 3 \Rightarrow 2$ bits de detección y 1 bit de corrección

Modelo geométrico del cubo



(a) 8 bytes (*d=1*) (b) 4 bytes (*d=2*) (c) 2 bytes (*d=3*)

El **modelo geométrico del cubo**, indica de una forma intuitiva, que a medida que aumenta la distancia Hamming (d), y por tanto, la capacidad de detección y/o corrección de los códigos, disminuye el rendimiento.

$d = 1$: podemos representar 8 caracteres, pero cualquier codeword es válido $\Rightarrow$ no tiene capacidad de detección ni corrección.

$d = 2$: hemos disminuido la codificacion a 4 caracteres $\Rightarrow$ tiene capacidad para detectar 1 error, pero no corrige ningún error.

$d = 3$: ahora sólo se pueden codificar 2 caracteres $\Rightarrow$ puede detectar hasta dos bits erróneos y corregir un bit.

Nivel de enlace de datos

1. Funciones de nivel de enlace
2. Direccionamiento
3. Control de errores
4. ***Entramado***
5. Control de flujo
6. HDLC

Entramado

El ***entramado (framming)*** es el proceso de empaquetar una secuencia de bits en una trama, de forma que, cada trama sea distinguible de otra.

Añade unos bits de **inicio** y **fin** de trama, así como direcciones hardware y control de errores.

La dirección **destino** define dónde debe ir la trama, y la dirección **origen** es útil para localizar el emisor en el proceso de confirmación de tramas.

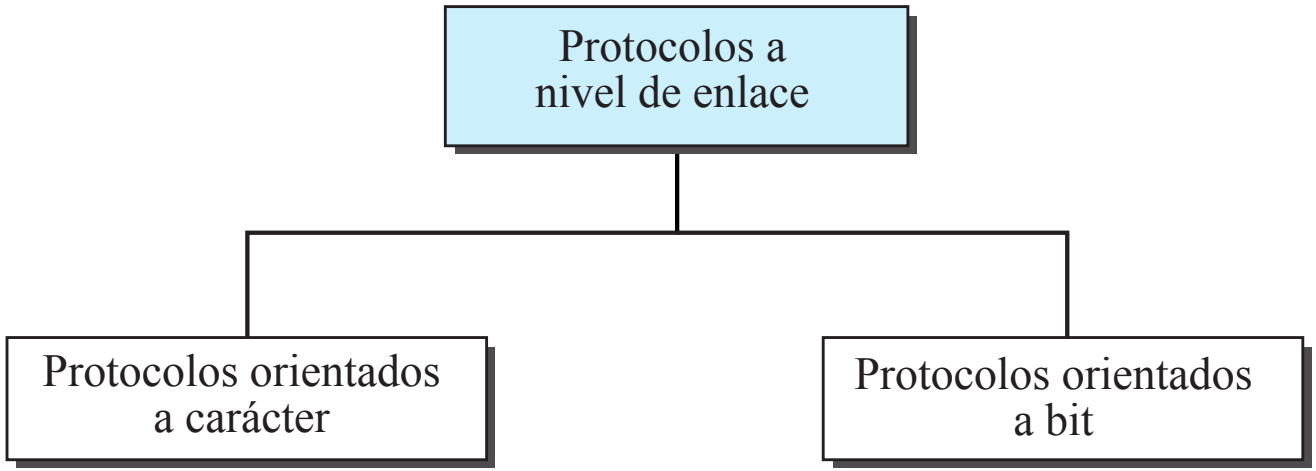
Aunque el mensaje global podría ir empaquetado en una sóla trama, no es lo normal, debido a que puede originar tramas muy grandes, por lo que el proceso de control de error se vuelve ineficiente.

Cuando un mensaje se transporta en una trama muy larga, incluso un error de 1-bit requiere la retransmisión completa de toda la trama.

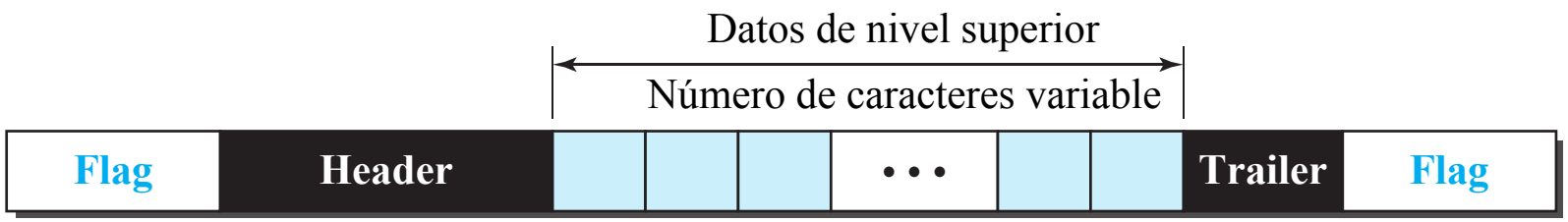
Por ello, el mensaje se suele dividir en tramas más pequeñas, y así, un error de un bit afecta sólo a una trama pequeña.

En el entramado de datos de nivel superior aparece el problema de la **transparencia de datos**, que surge cuando es necesario insertar algún bit/byte extra para que no sean interpretados erróneamente.

Protocolos a nivel de enlace



Protocolo orientado a carácter



BSC (*Binary Synchronous Communication*), IBM.

Las tramas orientadas a carácter (o byte) tienen la siguiente disposición:

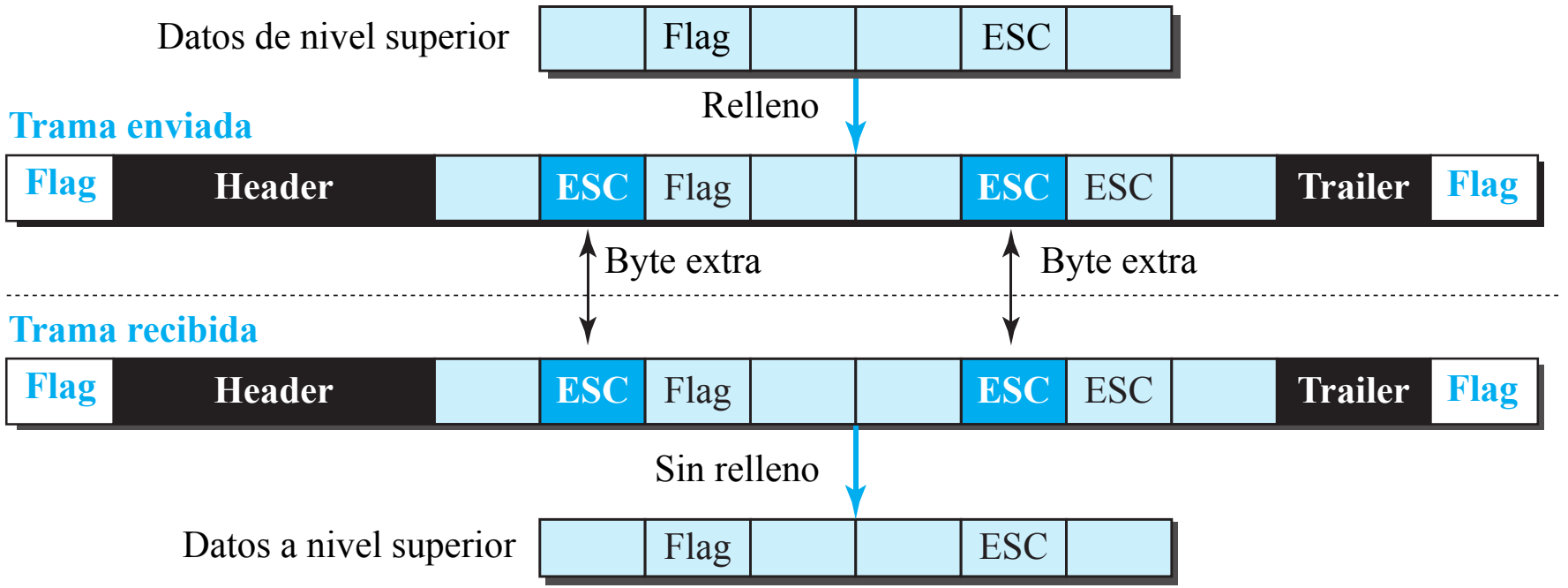
- Los **datos** de nivel superior son caracteres de 8 bits codificados ASCII.
- La **cabecera** (**header**), contiene direcciones origen y destino e información de control.
- El **trailer**, con los bits redundantes para control de errores, es múltiplo de 8 bits.
- Para la separación de tramas, se añade un **flag** al inicio y fin de la trama.

Los protocolos orientados a carácter se utilizaban cuando sólo se intercambiaban caracteres de texto, por lo que se utilizaba un carácter flag que fuera imposible de aparecer en los datos.

Posteriormente, surgió la necesidad de codificar imágenes, vídeo, audio, etc, por lo que el flag utilizado podía ser parte de los datos, en consecuencia, sería interpretado de forma errónea por el receptor.

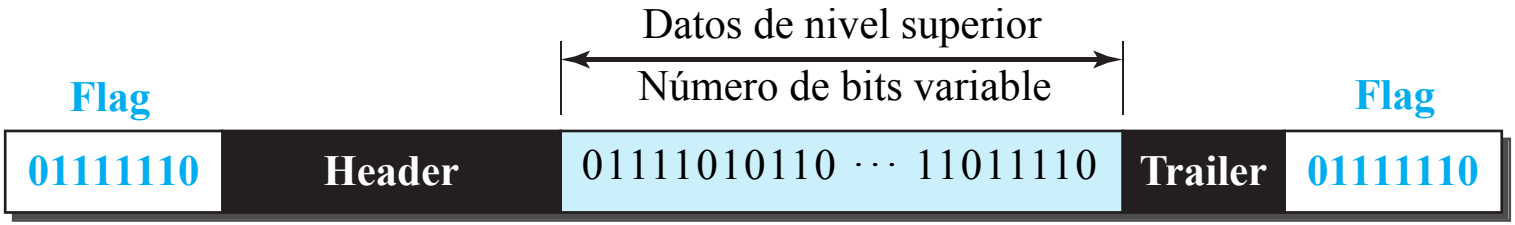
Para solucionarlo, se utiliza el mecanismo de **inserción de carácter ESC**, denominado, **byte-stuffing**.

Protocolo orientado a carácter: relleno de caracteres



El proceso de *byte stuffing* es el proceso por el que se añade un byte extra si hay un flag o carácter ESC en el texto.

Protocolo orientado a bit



HDLC (*High-level Data Link Communication*), ISO.

En protocolos orientados a bit, la parte de datos es una secuencia interpretada por el nivel superior, como datos, vídeo, audio, etc.

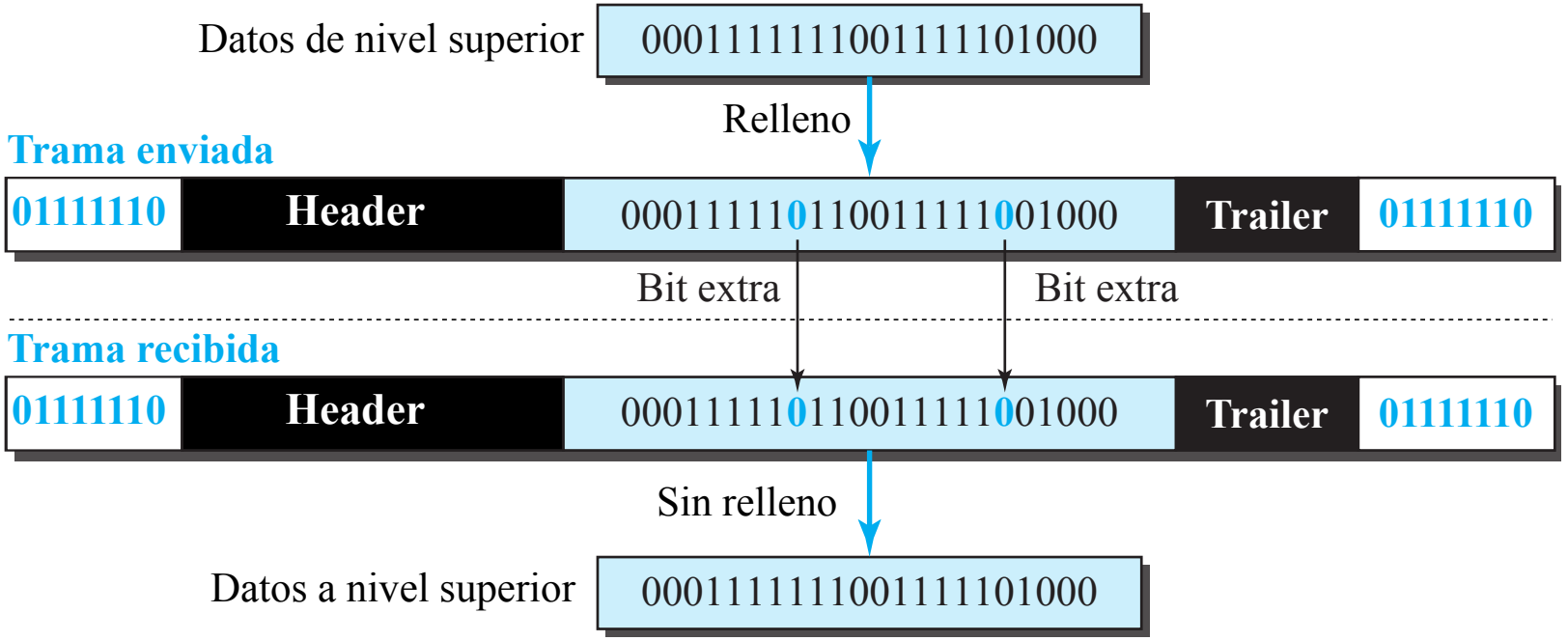
Para la delimitación de tramas se utiliza un patrón de bits denominado **flag**.

Habitualmente, el patrón flag de 8 bits es 01111110, como delimitador de inicio y final de trama.

Puede surgir el mismo problema, que el patrón de bits del flag aparezca en los datos.

Mediante el mecanismo de **inserción de bits** se elimina la posibilidad de que el patrón de bits del flag aparezca en la parte de datos.

Protocolo orientado a bit: relleno de bits



Siempre que aparezcan cinco ó más 1s consecutivos, se añade un bit 0. El receptor eliminará los bit 0 extra añadidos.

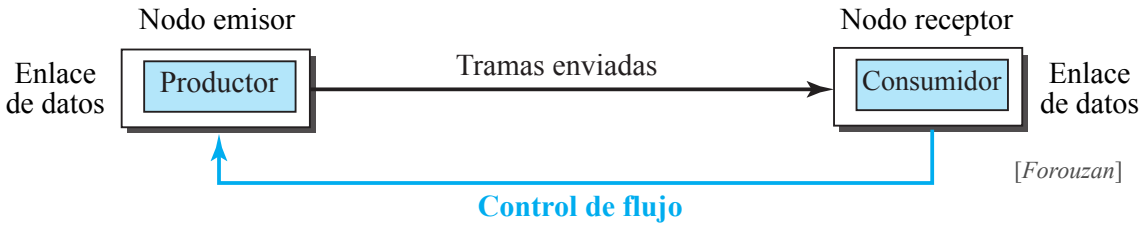
Nivel de enlace de datos

1. Funciones de nivel de enlace
2. Direccionamiento
3. Control de errores
4. Entramado
5. ***Control de flujo***
6. HDLC

Nota aclaratoria

- La función de Control de flujo se implementa en los niveles 2 y 4 (Modelo OSI y TCP/IP).
- El nivel 4 incluye Control de flujo, así como otras funciones y protocolos, que quedan **fuera del ámbito de esta asignatura**.
- Hay que tener en cuenta que en las prácticas, cuando se configura la función de ventana deslizante, es necesario hacerlo a nivel 4, TCP (está perfectamente indicado en el guión de la práctica).

Control de flujo



Hay una relación **productor-consumidor**, donde una entidad produce (**emisor**) items (tramas) que son consumidas por otra entidad (**receptor**), y donde debería existir un balance entre la tasas de producción y consumo:

- Si los items se producen **más rápido** de lo que pueden ser consumidos, el consumidor se sobrecarga y se producirá **pérdida** de items, que será necesario volver a retransmitir, con el consiguiente gasto de tiempo, ancho de banda, pérdida de rendimiento, etc.
- Si los items se producen a una tasa **más lenta** de lo que se podría consumir, se produce un uso **ineficiente** del sistema.

Las técnicas de control de flujo están orientadas a la prevención de descartes, pero también es muy importante un uso eficiente del sistema y balanceo entre las diferentes entidades. De forma simplificada:

- El nivel de enlace del emisor envía tramas hacia el nivel de enlace del receptor.
- Si en el receptor no se pueden procesar y enviar los paquetes hacia su nivel de red a la misma tasa en la que están llegando las tramas, alcanzará un estado de congestión.
- En este caso, el control de flujo debe de informar desde el nodo receptor hacia el nodo emisor de que pare o baje la tasa de envío de tramas.

Recuperación de errores

Debido a que el medio físico es **no confiable**, y se producen errores de transmisión, es necesario implementar mecanismos de control de errores a nivel enlace con el fin de evitar que tramas distorsionadas en el medio físico, sean enviadas al nivel red del nodo receptor.

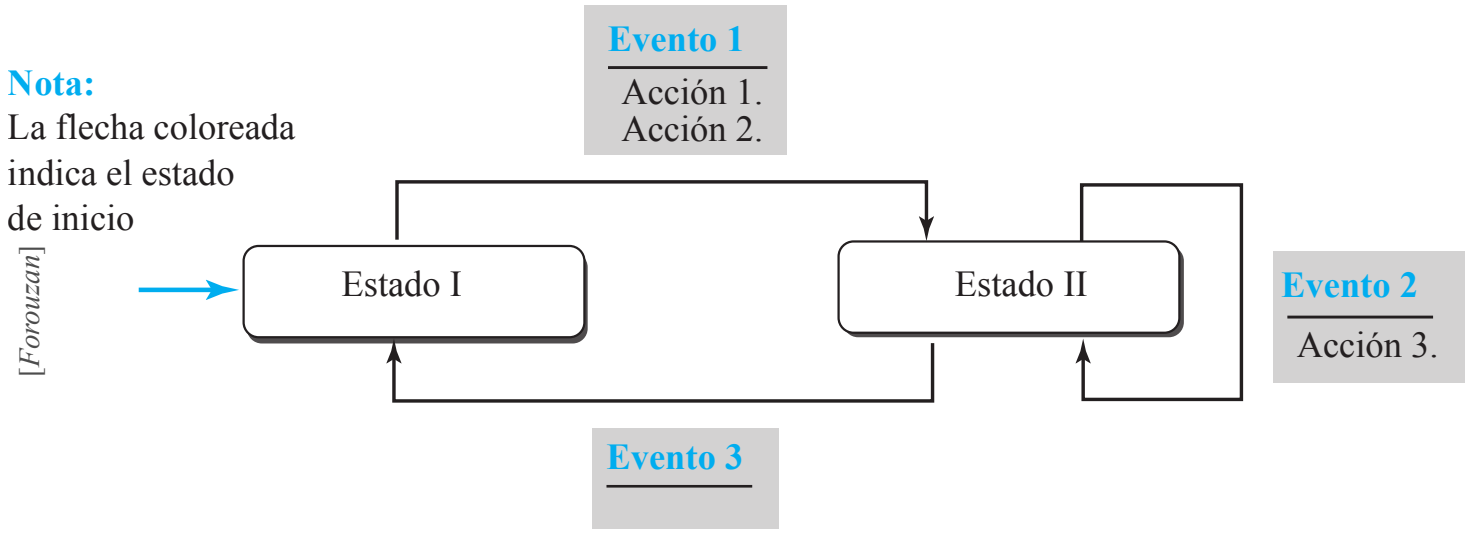
Habitualmente, la detección se realiza mediante CRC, añadido por el emisor y chequeado por el receptor.

Diferentes **estrategias**:

- Si la trama se detecta corrupta, entonces se descarta de forma silenciosa, en caso contrario, el paquete se envía a su capa red. En caso de descarte, será el emisor, mediante el uso de temporizadores, el encargado de la retransmisión. Por ejemplo: LANs, Ethernet, etc.
- Y además, se pueden enviar confirmaciones al receptor. Por ejemplo: Stop-and-Wait y ventana deslizante.
- Y una tercer método, combinación de control de flujo y control de errores: una vez detectada la trama errónea, informar al nodo emisor mediante una confirmación negativa, obteniendo, en general, un incremento del rendimiento Por ejemplo: TCP.

En adelante, utilizaremos la denominación de tramas `ACK` y `NACK` para confirmación positiva y negativa, respectivamente.

Máquinas de Estado Finito (FSM)



Para modelar el comportamiento del nivel de enlace se utilizan las **Máquinas de Estados Finitos** (*FSM, Finite State Machine*).

Una FSM es una máquina con un número finito de **estados**.

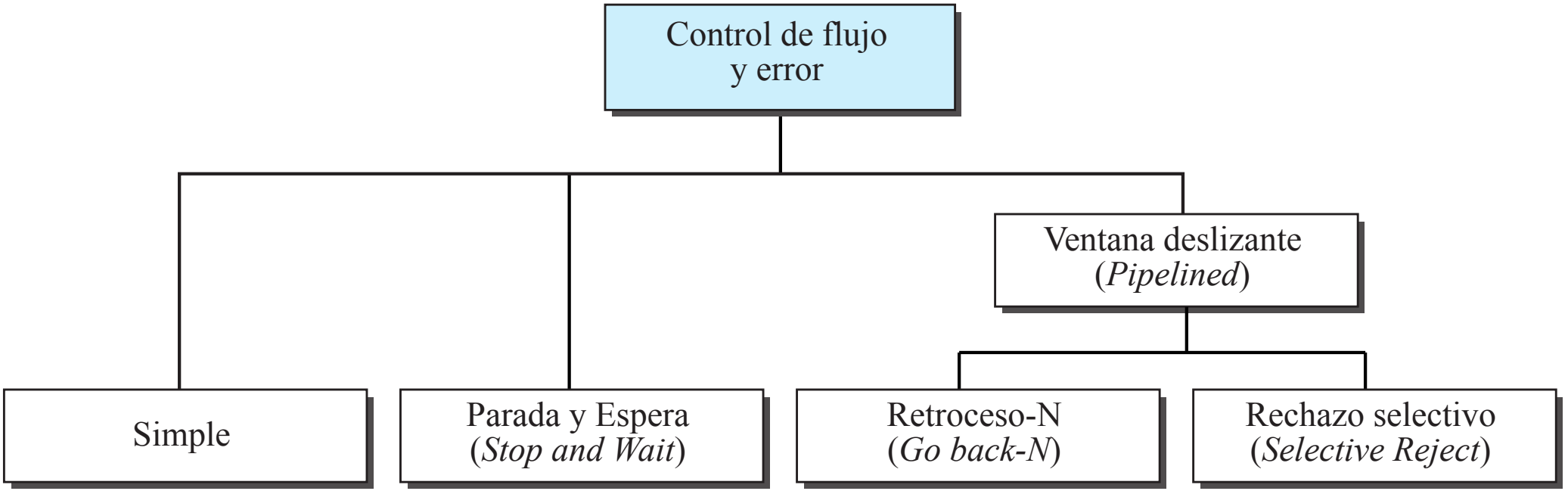
La máquina siempre está en algún estado hasta que ocurra un **evento**.

Cada evento es asociado con dos reacciones:

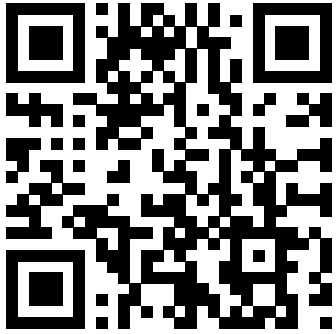
1. La lista (posiblemente vacía) de posibles acciones a realizar.
2. Siguiente estado.

Uno de los estados debe ser marcado como **estado inicial**, y es el estado en el que la máquina arranca cuando se inicia.

Técnicas de control de flujo y error

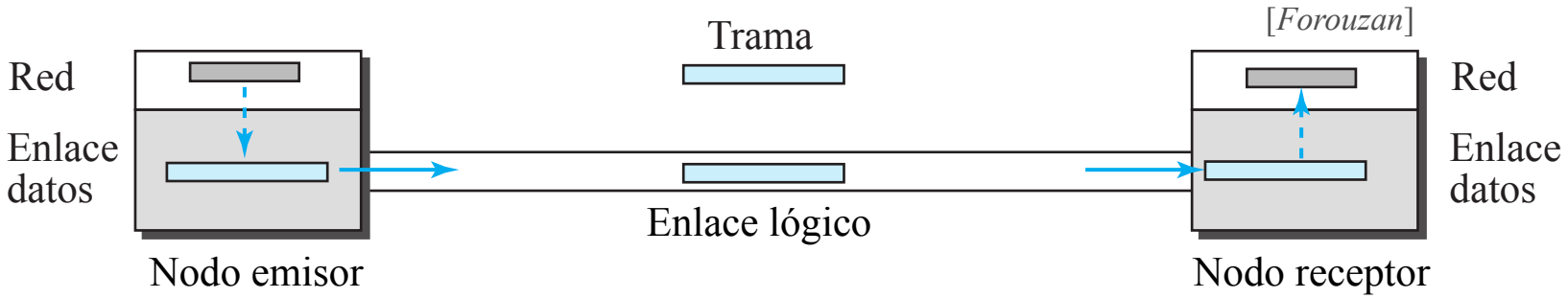


Protocolo simple



Protocolo simple

El **protocolo simple** no implementa control de flujo ni control de errores:



Se supone que el receptor gestiona las tramas conforme le van llegando, por lo tanto, suponemos que nunca será sobrecargado.

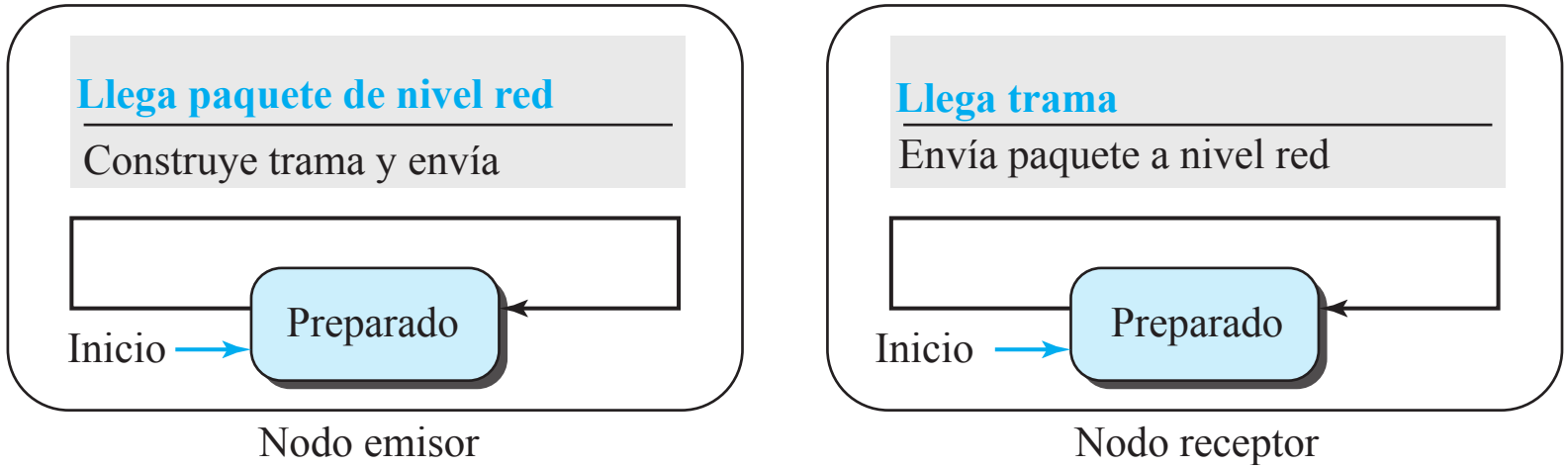
- En el **emisor**, el nivel de enlace acepta los paquetes de nivel de red, conforma la trama, y la envía al medio.
- En el **receptor**, el nivel de enlace recibe la trama, extrae el paquete y lo entrega a su nivel de red.

Es una estrategia simple, muy eficiente, pero no implementa control de errores (suponemos medios perfectos).

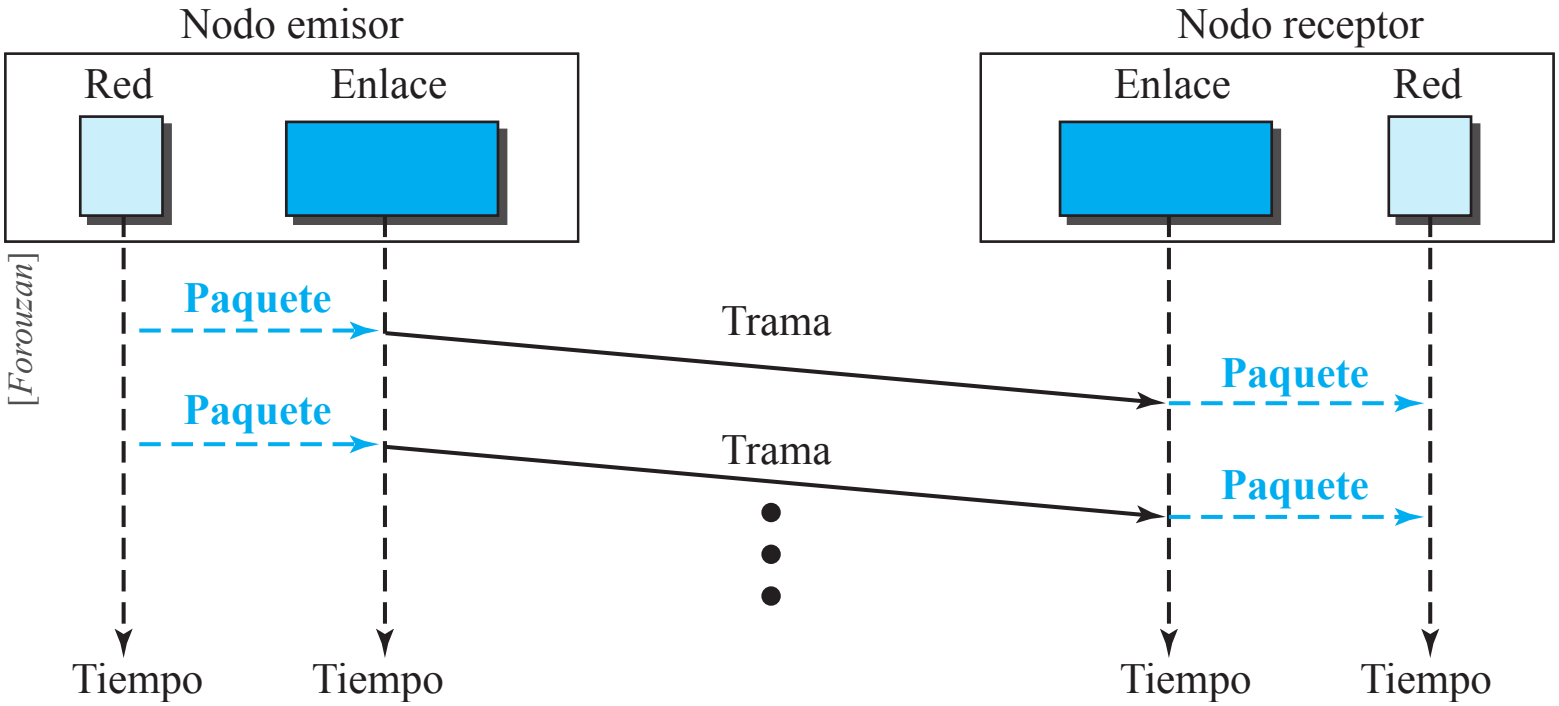
La tasa de procesamiento de tramas en el receptor debe de ser igual o superior a la tasa de envío de tramas por parte del emisor.

Protocolo simple: FSM

[*Forouzan*]

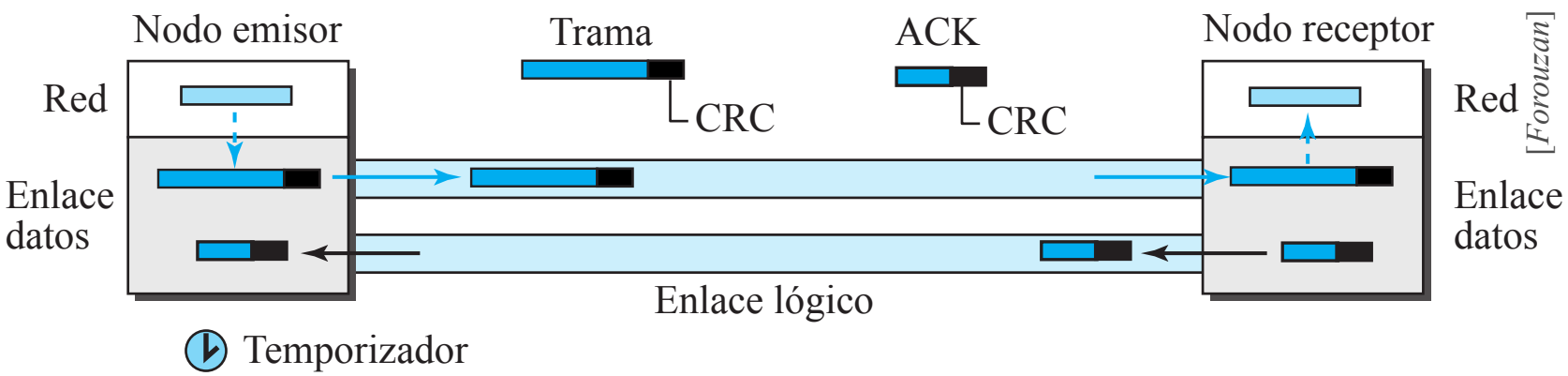


Protocolo simple: ejemplo



Parada y espera

Stop-and-wait



El protocolo **Stop-And-Wait** implementa control de flujo y control de error.

El **emisor** envía una trama y espera la confirmación antes de enviar la siguiente trama.

Las tramas van provistas de CRC. Cuando una trama llega al **receptor**, se hace el chequeo de CRC y si la trama es corrupta, se descarta silenciosamente (no hace nada).

Cada vez que el emisor envía una trama, inicia un **temporizador**:

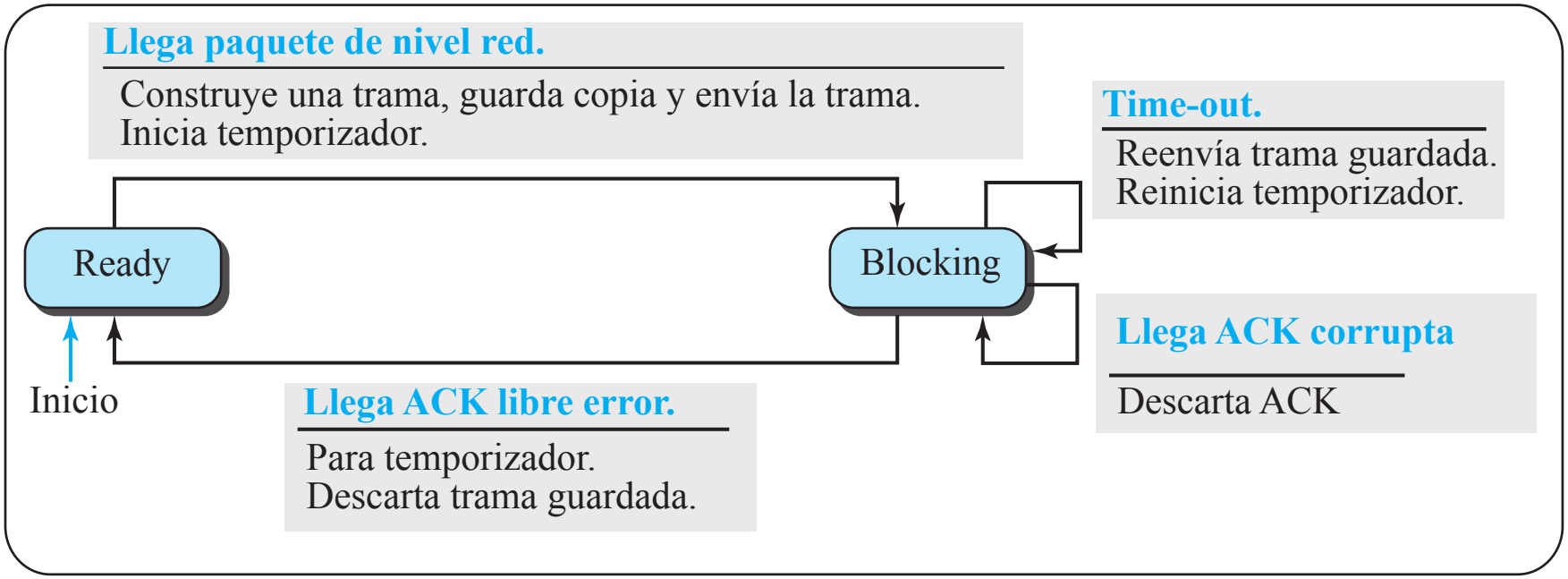
- Si la **confirmación** llega antes que el temporizador expire, entonces se detiene y envía la siguiente trama (caso de trama disponible).
- Si el temporizador expira, el emisor reenvía la trama previa, asumiendo que la trama ha sido perdida o estaba corrupta ⇒ necesita mantener copia en memoria de la trama hasta que la confirmación llegue.

Cuando la confirmación llega, el emisor elimina la copia y envía siguiente trama (si está preparada).

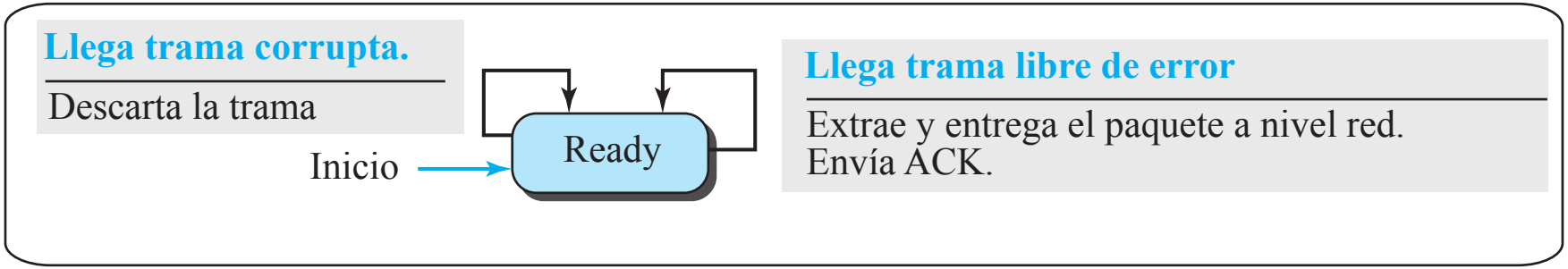
Stop-and-wait: FSM

Nodo emisor

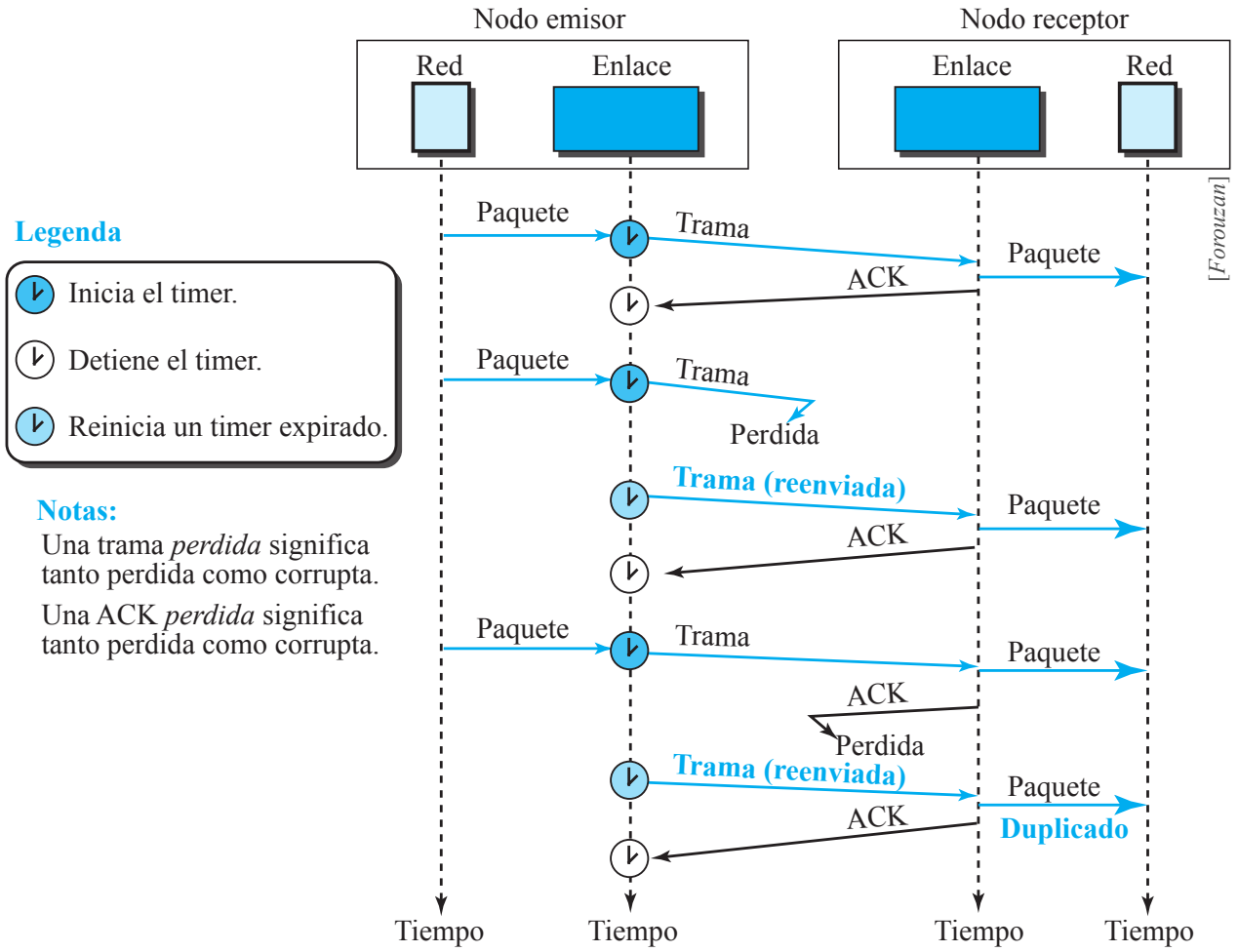
[*Forouzan*]



Nodo receptor



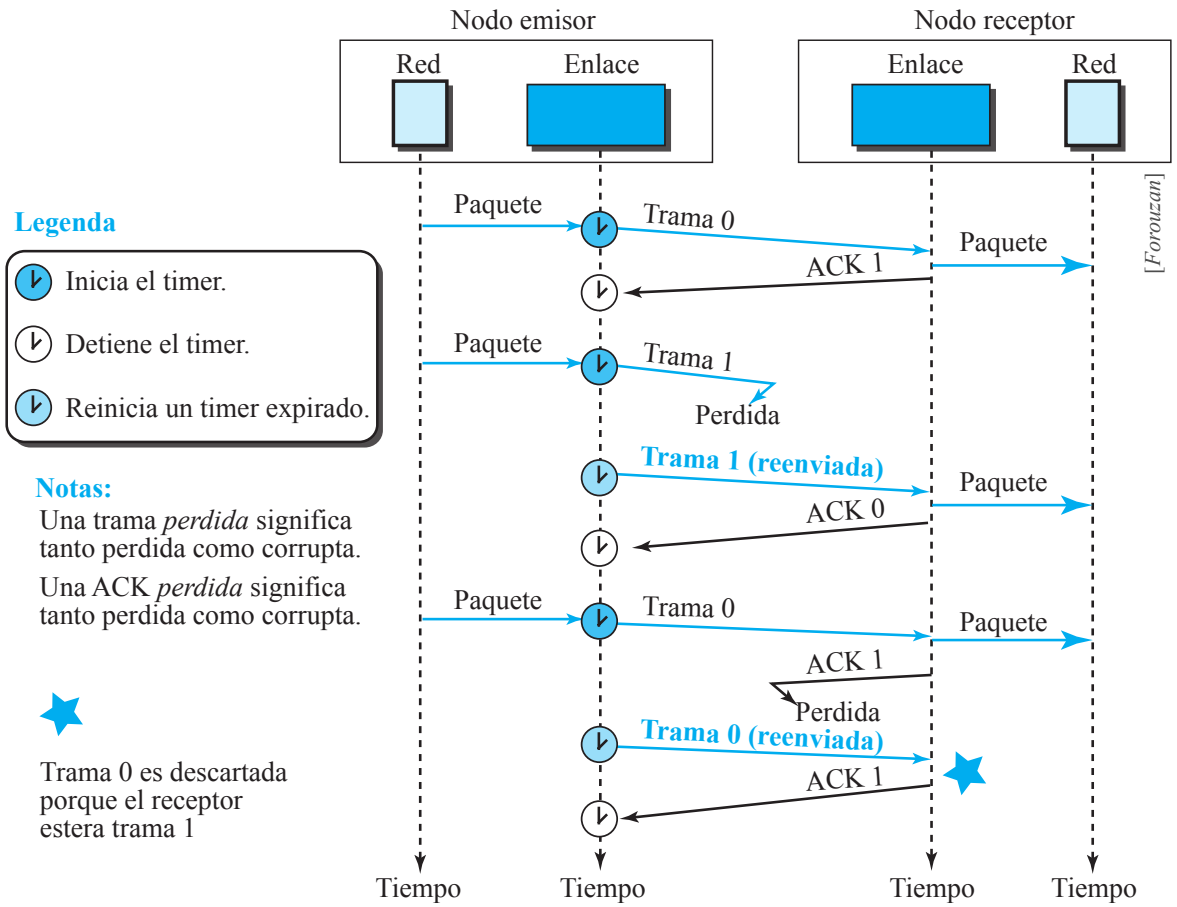
Stop-and-wait: ejemplo



Hay un problema con el esquema propuesto: la capa red del receptor recibe (incorrectamente) dos copias del mismo paquete.

Para solucionarlo, se introduce la numeración de tramas y ACKs en el siguiente esquema.

Stop-and-wait (0/1): ejemplo



`ACK x`, indica que la próxima trama que espera recibir es la numerada `x`.

Con la numeración de tramas 0 y 1 es suficiente, ya que sólamente puede haber una trama en circulación.

Stop-and-wait está diseñado para comunicación unidireccional: los datos viajan en un sentido (→) y las confirmaciones viajan en sentido contrario (←).

Medidas de rendimiento

Es necesario obtener un balanceo entre productor y consumidor: que la tasa de transmisión no sea tan elevada que desborde al receptor y, por tanto, se produzcan descartes, y no que sea tan baja, que se obtenga un rendimiento muy bajo.

El proceso total de comunicación, desde que se envía la trama hasta que se recibe la confirmación, depende de varios factores:

- Tasas de transmisión de datos al medio $[bits/s]$
- Longitud de las tramas $[bits]$
- Velocidad de transferencia en el medio $[m/s]$
- Longitud del enlace $[m]$

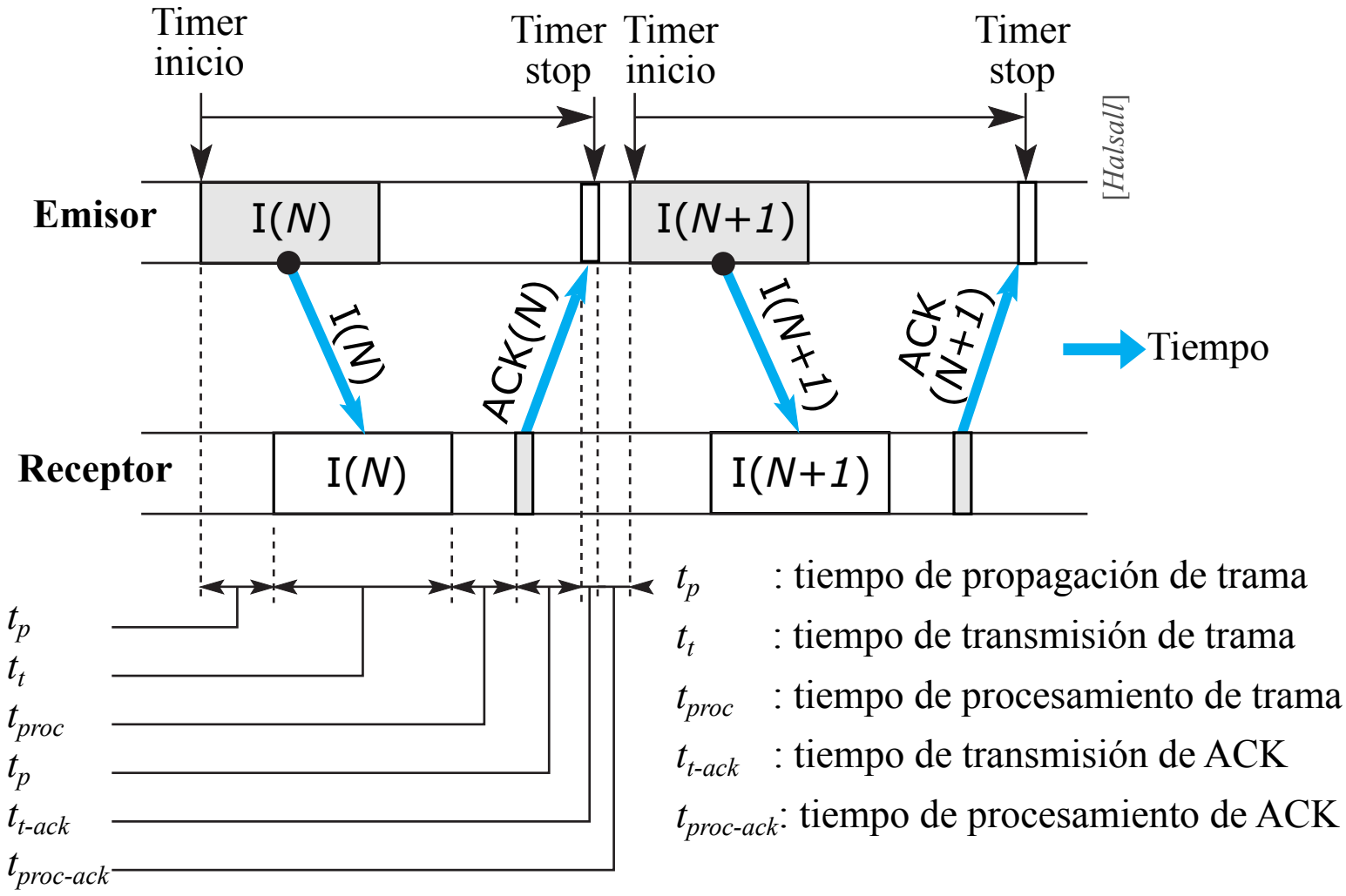
Utilización: relación entre el tiempo en el que el nodo emisor está útil ocupado en transmitir datos y el tiempo total en que el sistema está ocupado (enviando datos y esperando confirmación). También se denomina *eficiencia*. Es una medida porcentual.

$$U = \frac{T_{util}}{T_{total}} \tag{4}$$

Throughput: cantidad de información enviada por unidad de tiempo.

También se denomina *productividad* $[bits/s]$

Retardos



t_p : tiempo de propagación de trama

t_t : tiempo de transmisión de trama

t_{proc} : tiempo de procesamiento de trama

$t_{t\text{-}ack}$: tiempo de transmisión de ACK

$t_{proc\text{-}ack}$: tiempo de procesamiento de ACK

Stop-and-wait (medio *perfecto*): eficiencia (I)

Sea T_{total} el tiempo que se tarda en transmitir una trama, y recibir confirmación:

$$T_{total} = t_t + t_p + t_{proc} + t_{ack} + t_p + t_{proc}$$

donde:

t_p, tiempo que tarda una trama en propagarse por el medio *(tiempo de propagación)*.
t_t, tiempo que una estación tarda en transmitir una trama *(tiempo de transmisión)*.
t_{proc}, tiempo de procesamiento de trama.
t_{ack}, tiempo de transmisión de trama ACK.

Y consideramos las siguientes suposiciones:

- El medio es perfecto, ni se pierden ni se corrompen tramas.
- Tiempo de procesamiento de tramas despreciable $\Rightarrow t_{proc} = 0$.
- Tiempo de transmisión de *ack* despreciable (mucho más pequeñas que las tramas de datos) $\Rightarrow t_{ack} = 0$.

$$T_{total} = t_t + t_p + \cancel{t_{proc}} + \cancel{t_{ack}} + t_p + \cancel{t_{proc}} = t_t + 2t_p \quad (5)$$

Si definimos la relación entre el medio de transmisión y el dispositivo mediante el parámetro a, definido como:

$$a = \frac{t_p}{t_t}$$

A efectos de simplificación, suponemos el tiempo de transmisión normalizado a la unidad, $t_t = 1$, entonces:

$$a = t_p$$

Finalmente, estableciendo que el tiempo útil es el tiempo de transmisión de trama, a partir de (4) y (5), obtenemos la expresión de la **Utilización**:

Stop-and-wait: $U = \dfrac{1}{1 + 2a}$

Stop-and-wait (medio *perfecto*): eficiencia (II)

$$a = cte \Rightarrow \begin{cases} t_p = cte \\ t_t = cte \end{cases} \Leftrightarrow \begin{cases} \text{Enlaces punto a punto} \\ \text{Tramas de longitud fija} \end{cases}$$

$$a \neq cte \Rightarrow \begin{cases} t_p = \frac{d}{V} \\ t_t = \frac{L}{R} \end{cases} \Leftrightarrow \begin{cases} \text{Enlaces multipunto} \\ \text{Tramas de longitud variable} \end{cases}$$

donde:

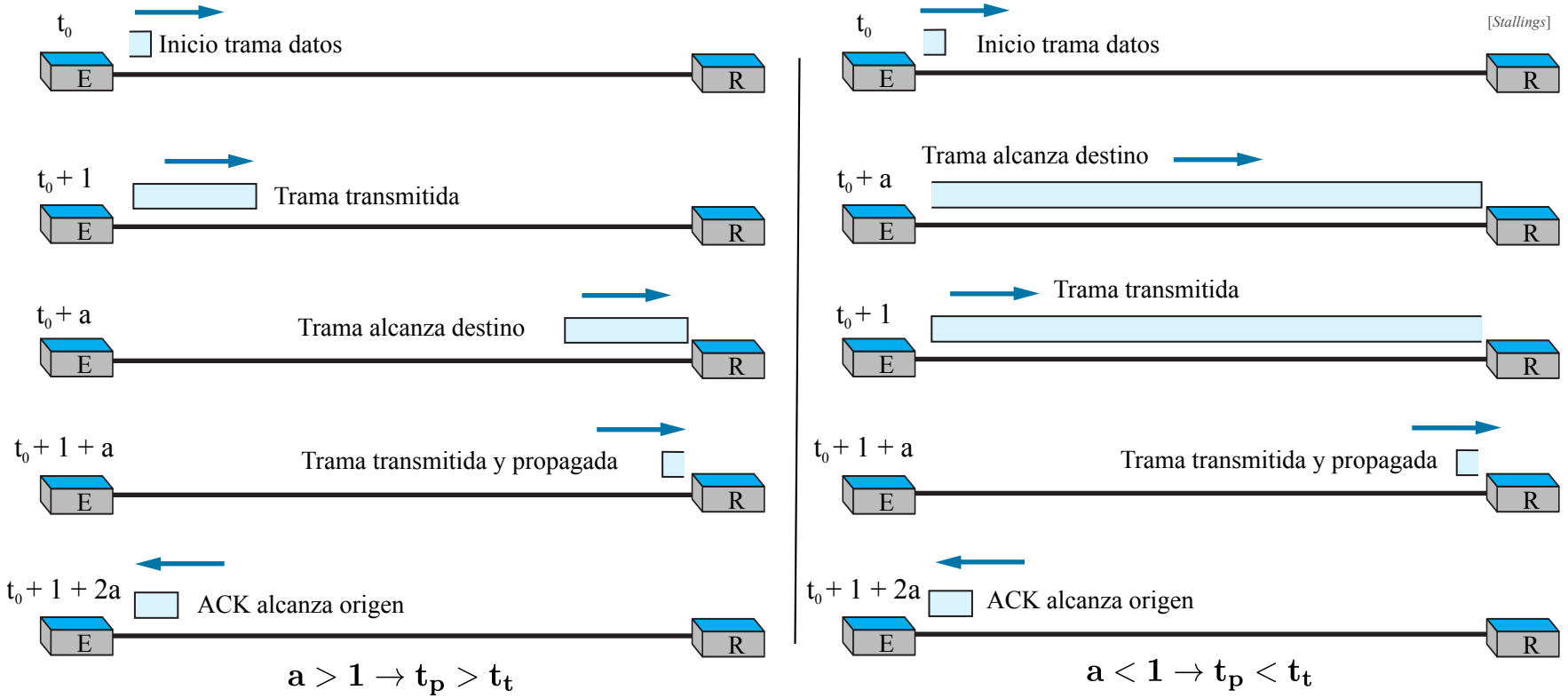
d : distancia del enlace $[m]$

V : velocidad de transmisión del medio: $\begin{cases} V_{medio\_no\_guiado} = V_{luz} = c = 3 \times 10^8 \,\mathrm{m/s} \\ V_{medio\_guiado} = 2 \times 10^8 \,\mathrm{m/s} \end{cases}$

L : longitud de la trama $[bits]$

R : tasa de transmisión de bits al medio $[bps]$

Stop-and-wait (medio *perfecto*): eficiencia (III)



$a > 1$: la trama es totalmente transmitida antes de que empiece a llegar al receptor
→ *infrautilización del dispositivo*.

$a < 1$: la longitud del enlace en bits es menor que la longitud de trama
→ *infrautilización del medio*.

En ambos casos, se tiene que:

$$U = \frac{1}{1 + 2a}$$

Stop-and-wait: ejemplos

Ejemplo-1: LAN Ethernet 10BaseT

$d = 100\,\text{m}$

$L = 1\,\text{Kbytes}$

$R = 10\,\text{Mbps}$

$V = 2 \times 10^8\,\text{m/s}$

Solución:

$$t_t = \frac{L}{R} = 800\,\mu\text{s}$$

$$t_p = \frac{D}{V} = 0{,}5\,\mu\text{s}$$

por lo tanto:

$$a = \frac{t_p}{t_t} = \frac{0{,}5\,\mu\text{s}}{800\,\mu\text{s}} = 6{,}25 \times 10^{-4}$$

entonces:

$$U = \frac{1}{1+2a} = \frac{1}{1+2\cdot 6{,}25\cdot 10^{-4}} \approx 0{,}998 \rightarrow 99{,}8\ \%$$

Conclusión: utilización **muy alta**, cercana al 100 %, lo que nos indica que la mayor parte del tiempo está transmitiendo.

Ejemplo-2: WAN ATM fibra óptica

$d = 1000\,\text{km}$

$L = 53\,\text{bytes}$

$R = 155\,\text{Mbps}$

$V = 2 \times 10^8\,\text{m/s}$

Solución:

$$t_t = \frac{L}{R} = 2{,}73\,\mu\text{s}$$

$$t_p = \frac{D}{V} = 5\,\text{ms}$$

por lo tanto:

$$a = \frac{t_p}{t_t} = \frac{5\,\text{ms}}{2{,}73\,\mu\text{s}} = 1831$$

entonces,

$$U = \frac{1}{1+2a} = \frac{1}{1+2\cdot 1831} \approx 0{,}00027 \rightarrow 0{,}027\ \%$$

Conclusión: utilización **muy baja**, lo que nos indica que la mayor parte del tiempo NO está transmitiendo.

Stop-and-wait (medio *imperfecto*): eficiencia

Recordamos que la utilización U, donde, t_t es el tiempo de transmisión de trama, y T_{total} es el tiempo total de ocupación del canal:

$$U = \frac{t_t}{T_{total}} \tag{6}$$

En el caso de existencia de errores, necesitamos modificar (6), considerar que N_r es el valor esperado del número de transmisiones por trama y, a partir de la expresión T_{total} obtenida en (5), tenemos que:

$$U = \frac{t_t}{N_r T_{total}} = \frac{1}{N_r(1+2a)} \tag{7}$$

Si suponemos P la **probabilidad de trama errónea**, y que las `ACK` están libres de error, la probabilidad de que se necesiten k intentos para transmitir con éxito una trama es $Pr[\text{k transmisiones}] = P^{k-1}(1-P)$, entonces:

$$N_r = E[\text{transmisiones}] = \sum_{i=1}^{\infty}(iPr[\text{i transmisiones}]) = \sum_{i=1}^{\infty}\left(iP^{i-1}(1-P)\right) = \frac{1}{1-P} \tag{8}$$

Por lo que finalmente, a partir de las ecuaciones (7) y (8), obtenemos:

Stop-and-wait: $U = \dfrac{1-P}{1+2a}$

El término de la probabilidad de error

El término P corresponde a la **probabilidad de trama errónea**, definida a partir de la Probabilidad de bit erróneo (P_{bit}) y la longitud de la trama (L), calculada a partir de:

$$P = L \cdot P_{bit}$$

o para tamaños de trama elevados:

$$P = 1 - (1 - P_{bit})^{L}$$

Los términos P_{bit} (*Probabilidad de bit erróneo*) y BER (*Bir error rate*) son equivalentes, y utilizados indistintamente.

Stop-and-wait: ejemplos

IEEE 802.11 (WiFi)

- Distancia hasta 250 m
- Tamaño de paquete 400-18000 bits
- Tasas de transmisión 2-11-54-150 Mbps

USB

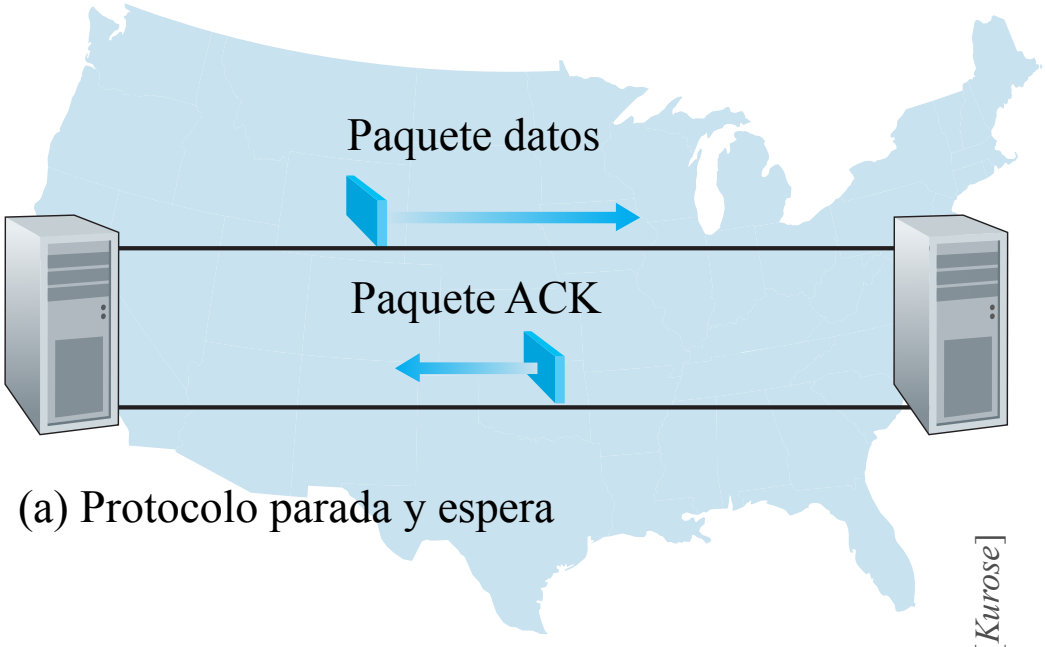
- Distancia + hub de 30 m
- Tasas max de 480 Mbps
- Tamaño de paquete 0-8216 bit

IEEE 1394 (Firewire) versiones *a* y *b*

- Distancia $4{,}5$ m
- Tasa de transmisión 400-1600 Mbps
- Tamaño de paquete 0-16384 bit (a 400 Mbps)
  0-32768 bit (a 1600 Mbps)

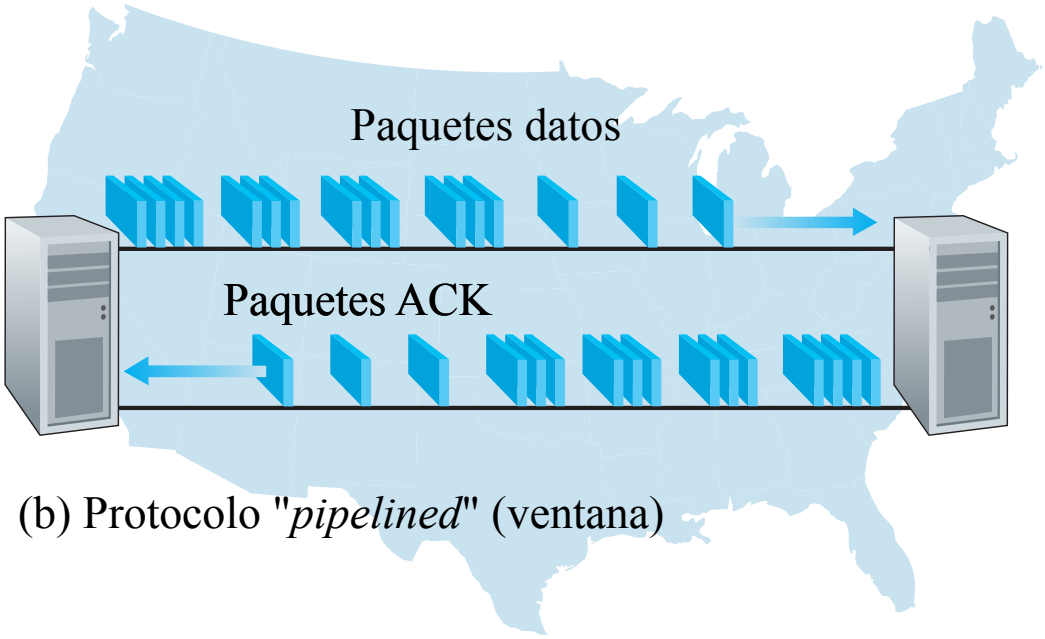
Ventana deslizante

Control de flujo *pipelined*



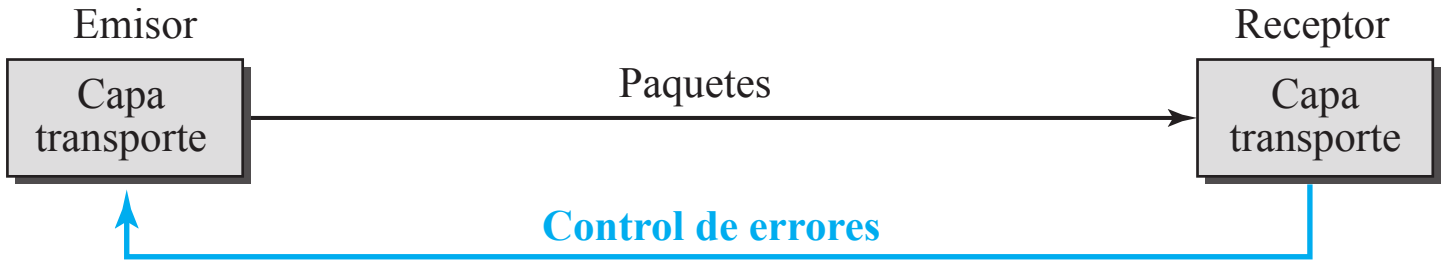
(a) Protocolo parada y espera

[*Kurose*]



(b) Protocolo "*pipelined*" (ventana)

Control de errores



Como el medio físico es no confiable, es necesario mantener el control de errores, con las siguientes funciones:

- Detectar y descartar paquetes corruptos.
- Registrar y reenviar paquetes perdidos y descartados.
- Reconocer y descartar paquetes duplicados.
- Mantener los paquetes fuera de orden hasta que lleguen los paquetes perdidos.

Números de secuencia

El control de errores necesita saber, en el **emisor**, qué paquetes son necesarios reenviar, y en el **receptor** necesita saber qué paquete ha sido duplicado, ha llegado fuera de orden y detectar los huecos entre paquetes.

Para ello, los paquetes se numeran mediante un **número de secuencia**.

Es necesario establecer un límite, de forma que si el campo secuencia son m bits, entonces el rango de secuencia estará entre 0 y $2^m - 1$, es decir, **módulo $\mathbf{2^m}$**.

Ejemplo: Con un campo de secuencia de longitud $m = 4$, los números de secuencia serán:

$$0, \cdots, 2^m - 1 = 0, \cdots, 2^4 - 1 = 0, \cdots, 15$$

por lo tanto:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \cdots$$

por lo que se produce el denominado **reciclado** de números de secuencia.

Confirmaciones (ACK)

En general, se utilizan señales positivas, **ACK**, para control de error:

- El receptor puede enviar una confirmación (ACK) para indicar que uno o varios paquetes han llegado libres de error.
- El receptor puede descartar paquetes corruptos, sin avisar.

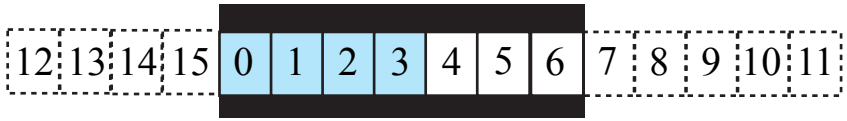
El uso de **temporizadores**:

- El emisor detectará la pérdida mediante el uso de temporizadores.
- Cada vez que envía un paquete, activa un temporizador, si no llega un ACK antes que el temporizador expire, reenviará el paquete.

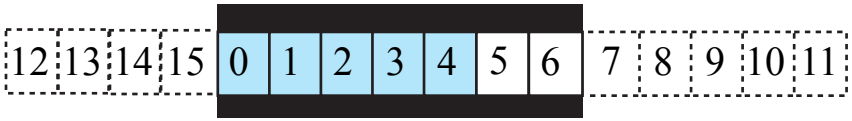
En relación al **descarte de paquetes**:

- Los paquetes duplicados también pueden ser descartados por el receptor de forma silenciosa.
- Los paquetes fuera de orden pueden ser descartados (y ser tratados como perdidos por el emisor) o almacenados a la espera que lleguen los paquetes que faltan.

Ventana deslizante (formato lineal)

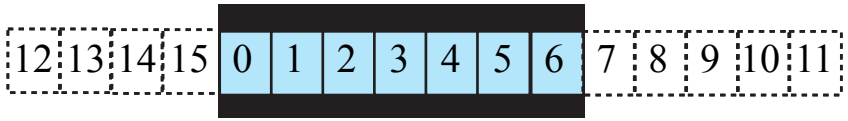


(a) Cuatro paquetes han sido enviados

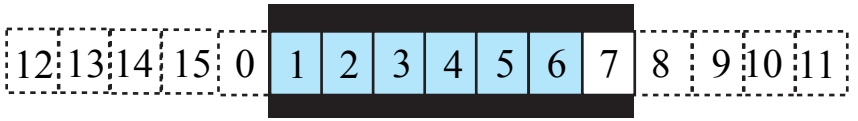


(b) Cinco paquetes han sido enviados

[Forouzan]

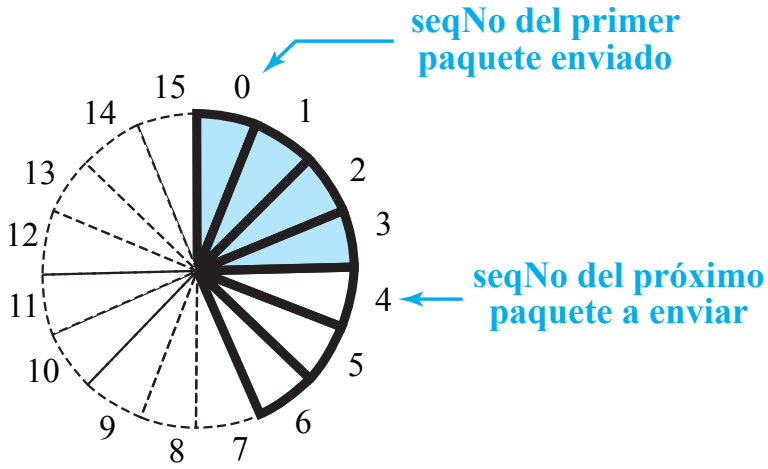


(c) Siete paquetes enviados: ventana llena

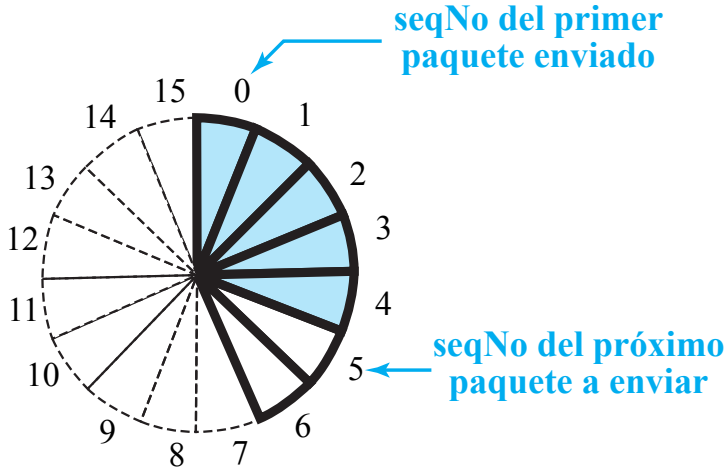


(d) Confirmación paquete 0, desplazamiento ventana

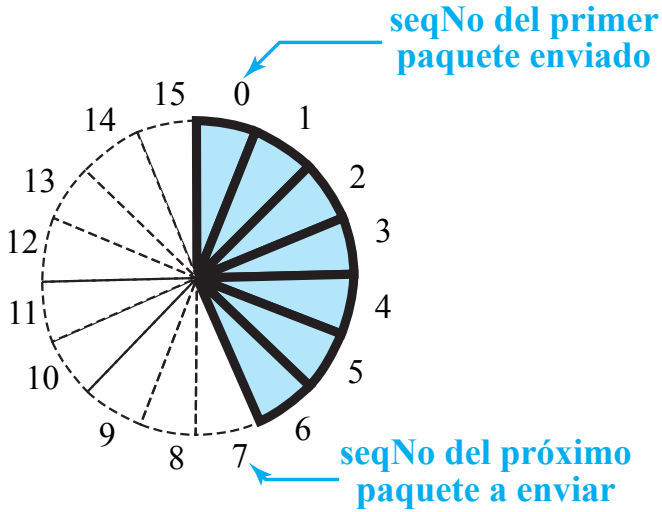
Ventana deslizante (formato circular)



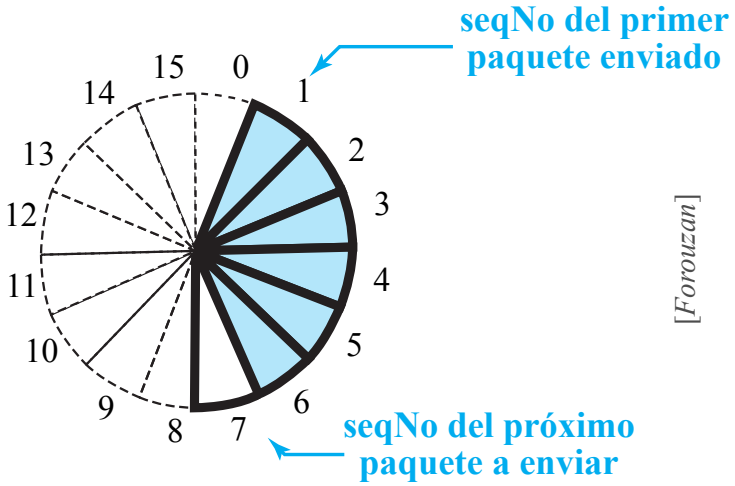
(a) Cuatro paquetes han sido enviados



(b) Cinco paquetes han sido enviados



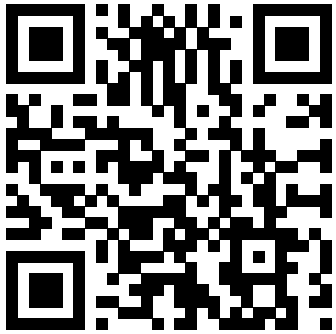
(c) Siete paquetes enviados: ventana llena



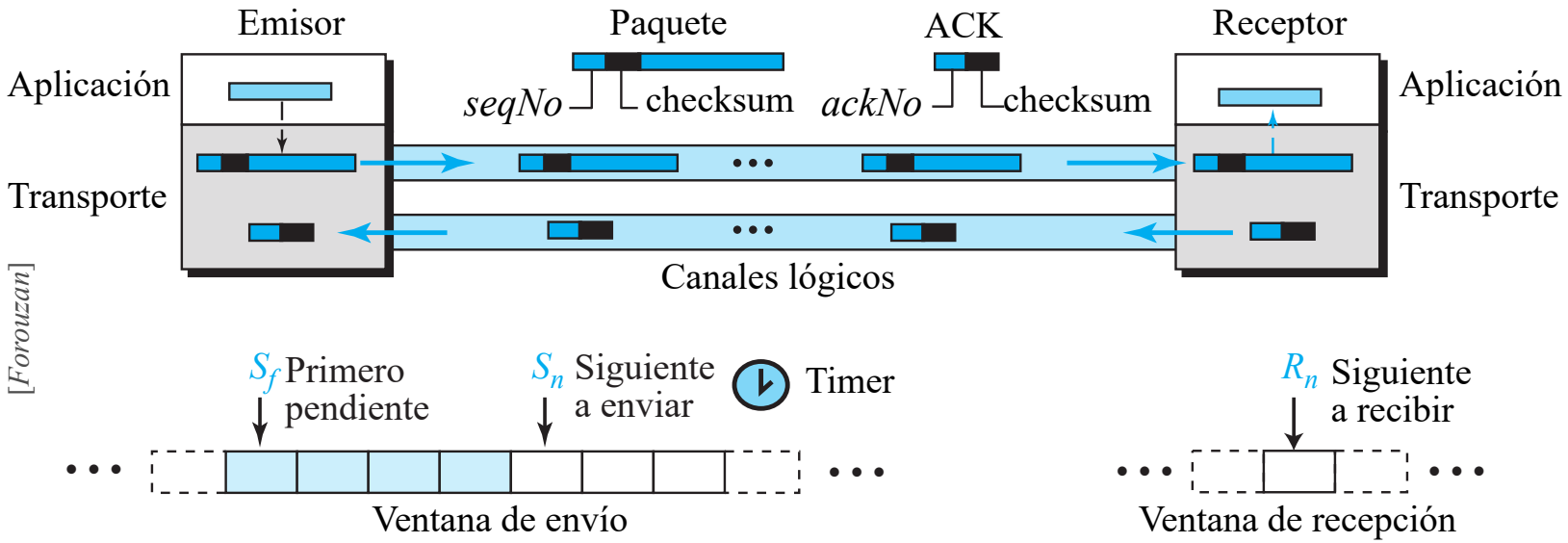
(d) Confirmación paquete 0, desplazamiento ventana

[Forouzan]

Retroceso-N



Protocolo *Retroceso-N*

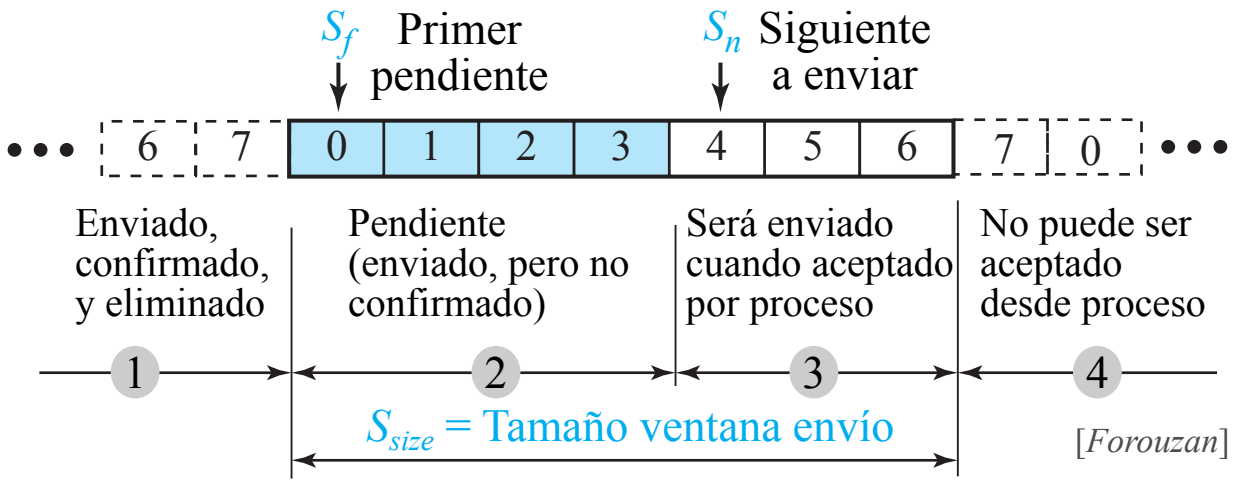


Para aumentar la eficiencia, varios paquetes deben de estar en transición mientras que el emisor espera la confirmación (*pipelining*).

- El emisor mantiene copia de los mensajes enviados mientras espera la confirmación.
- Números de secuencia módulo 2^m, donde m es el tamaño en bits del campo secuencia.
- Confirmación es **acumulativa** y define número de secuencia del siguiente paquete que espera recibir.

Por ejemplo, un valor de confirmación de 7, indica que todos los paquetes con secuencias anteriores hasta el 6 han llegado correctamente, y el receptor espera el paquete con número de secuencia 7.

Protocolo *Retroceso-N*: ventana emisor (I)



La **ventana del emisor** es una caja imaginaria que cubre los números de secuencia de los paquetes que pueden estar en tránsito o que pueden ser enviados.

Cada posición de la ventana define:

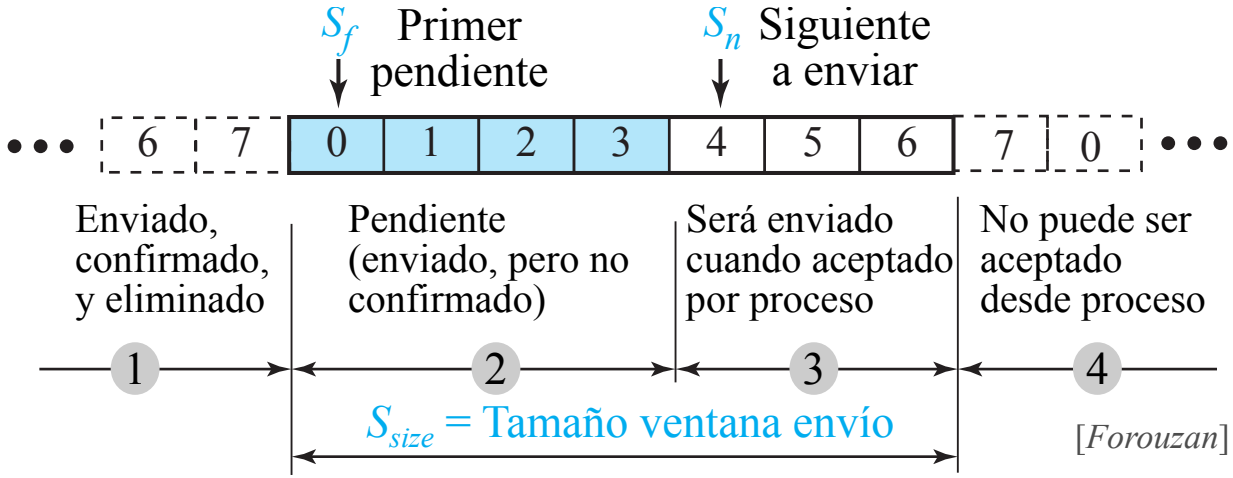
- Paquetes que ya han sido enviados, ó,
- Paquetes que pueden ser enviados.

El **tamaño máximo de la ventana de emisión** es $2^m - 1$.

La figura muestra una ventana deslizante de tamaño 7 ($m = 3$).

El mecanismo de ventana deslizante es una abstracción para modelar envío eficiente de paquetes.

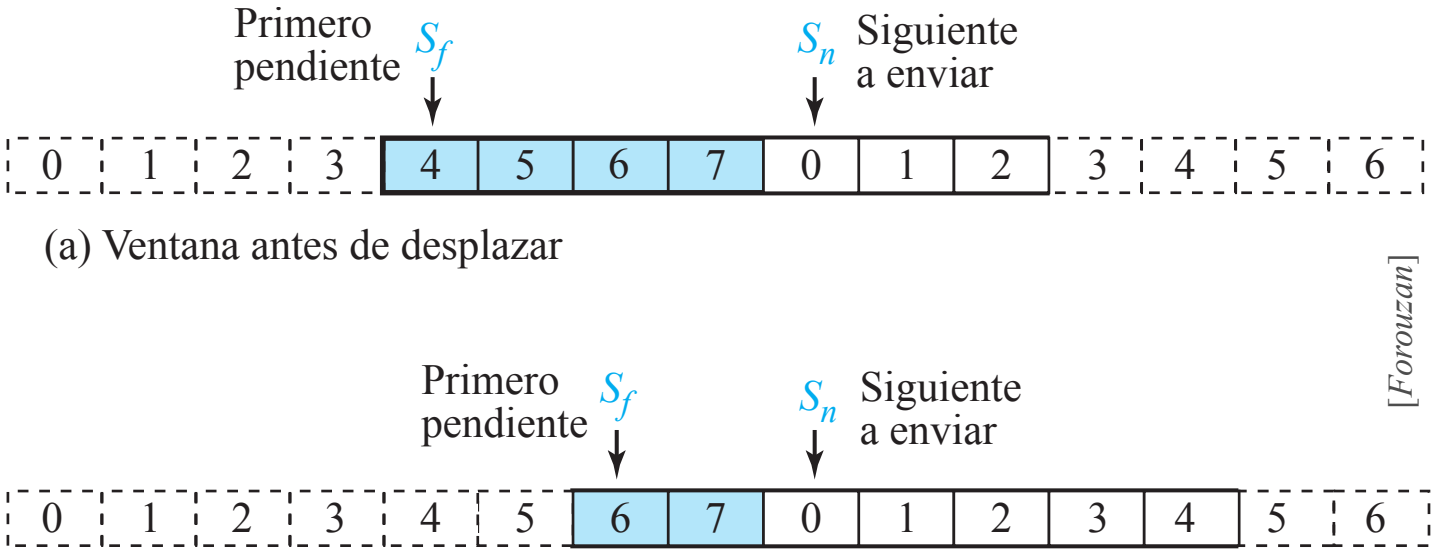
Protocolo *Retroceso-N*: ventana emisor (II)



Mediante las variables S_f, S_n y S_{size}, la ventana del emisor divide las posibles números de secuencia en cuatro regiones:

1. Números de secuencia de los **paquetes que ya han sido confirmados**. El emisor ya no debe preocuparse por estos paquetes, por tanto, no se mantienen en buffer.
2. Rango de números de secuencia que define los **paquetes que han sido enviados pero están sin confirmar**. Es necesario esperar para determinar si fueron entregados, o por el contrario, se perdieron. Se denominan paquetes pendientes.
3. Rango de números de secuencia de **paquetes podrán ser enviados** en caso de que el nivel aplicación así lo requiera.
4. Rango de números de secuencia que **no pueden ser utilizados hasta que se produzca el deslizamiento de la ventana**.

Protocolo *Retroceso-N*: desplazamiento ventana emisor



(a) Ventana antes de desplazar

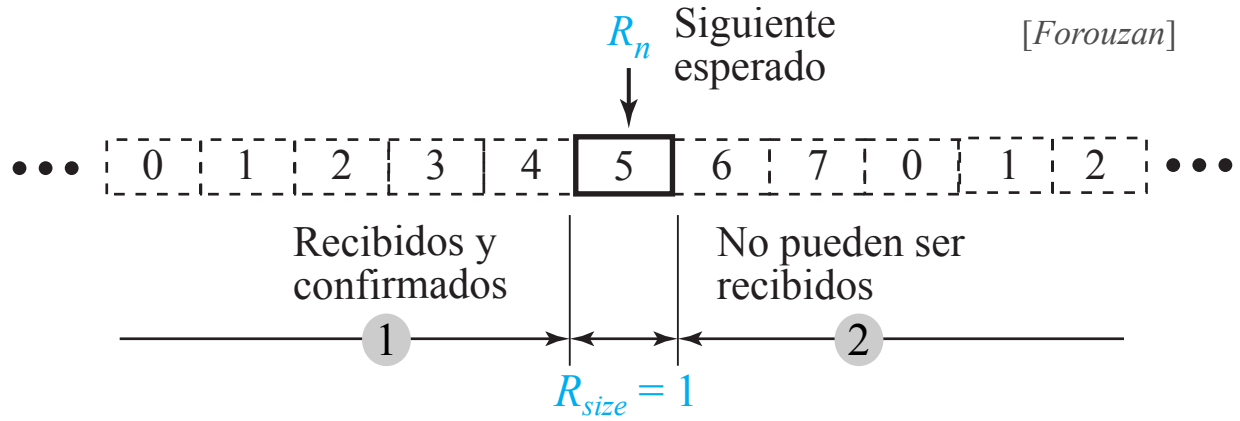
(b) Ventana después de desplazar (ha llegado una ACK con *ackNo=6*)

[*Forouzan*]

La ventana del emisor **se desplaza** uno o varios slots cuando llega una ACK libre de error.

El ACK tiene que ser con número de secuencia *ackNo* mayor que S_f y menor o igual que S_n (en aritmética modular).

Protocolo *Retroceso-N*: ventana receptor



La **ventana del receptor** asegura que los paquetes de datos se reciben correctamente y que se envían las confirmaciones correctas.

Tamaño de la ventana de recepción **siempre es 1**.

El receptor espera siempre la llegada de un paquete específico, cualquier otro fuera de secuencia se descarta, y por tanto, necesita ser reenviado.

Sólo es necesaria la variable R_n (*siguiente paquete esperado*), que define dos regiones:

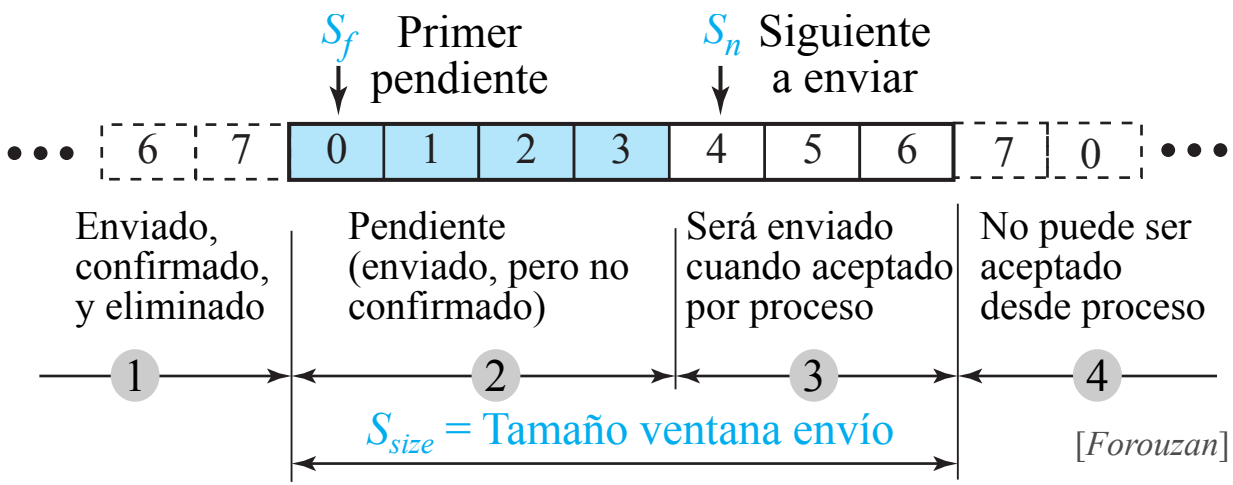
1. Números de secuencia de **paquetes que ya han sido recibidos y confirmados**.
2. Números de secuencia de **paquetes que no pueden ser recibidos**.

Cualquier paquete recibido con número de secuencia en (1) o (2), se descarta.

Sólo se acepta y confirma un paquete con número de secuencia que coincida con R_n.

Cuando se recibe un paquete correcto, se desliza la ventana $\rightarrow R_n = (R_n + 1)_{|2^m}$.

Protocolo *Retroceso-N*: temporizadores



Para cada paquete enviado se activa un temporizador.

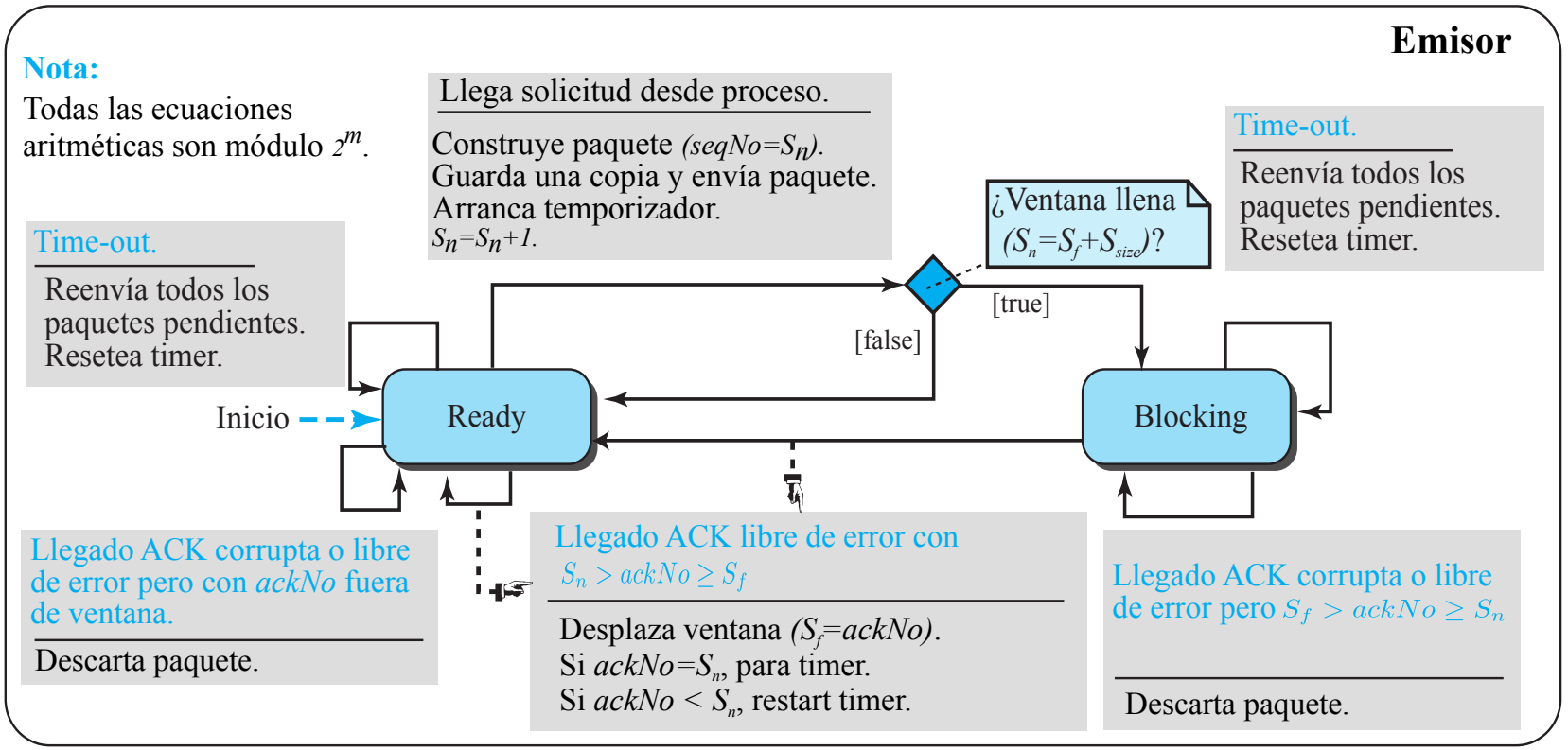
Si expira el temporizador antes de la llegada de alguna confirmación correcta, se envían **todos los paquetes pendientes**.

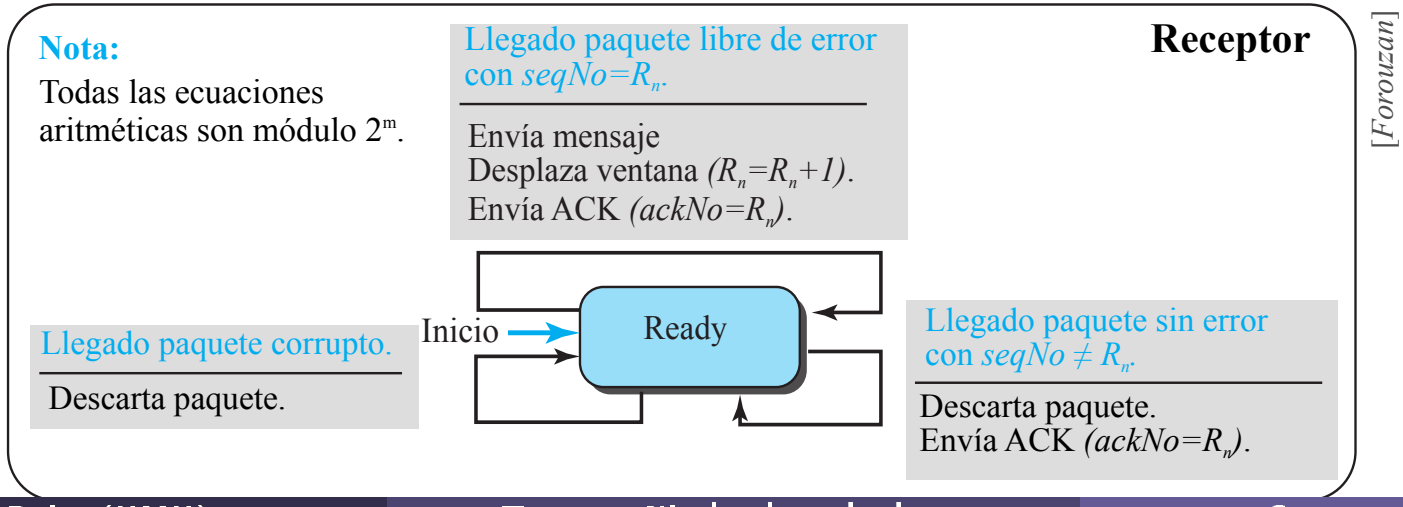
Por ejemplo:

- Supongamos que el emisor ha enviado el paquete 3, por lo tanto, $(S_n = 4)$.
- Un paquete llega erróneo (o se pierde) en el receptor, y por tanto, no devuelve confirmación.
- Expira el temporizador.
- Si $S_f = 0$, significa que los paquetes 0, 1, 2 y 3 no han sido confirmados (pendientes), entonces, el emisor reenvía otra vez los paquetes 0, 1, 2 y 3.

De aquí el nombre de Retroceso-N, ya que el emisor retrocede N posiciones y reenvía todos los paquetes.

Protocolo *Retroceso-N*: FSM



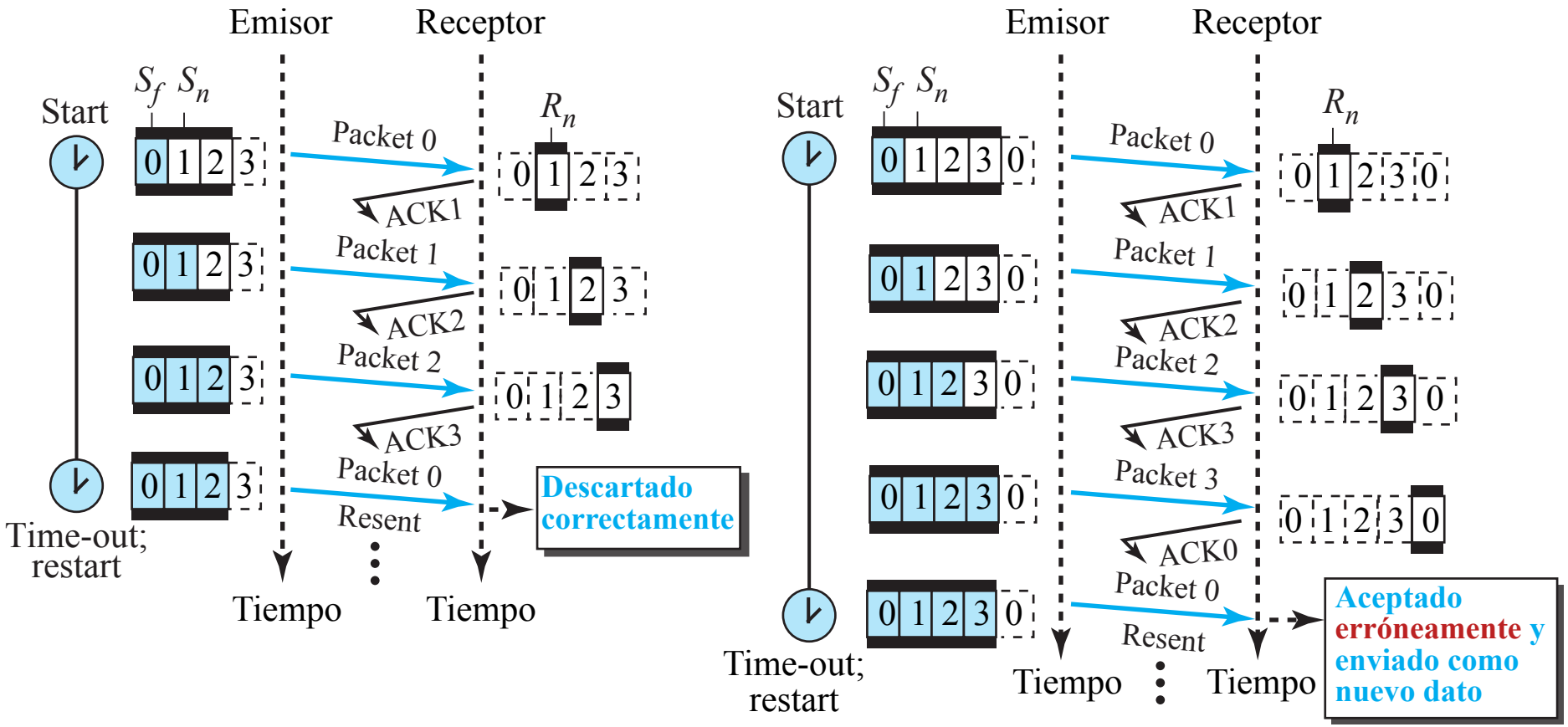


[Forouzan]

Protocolo *Retroceso-N*: justificación tamaño ventana envío

Supongamos:

$$m = 2$$

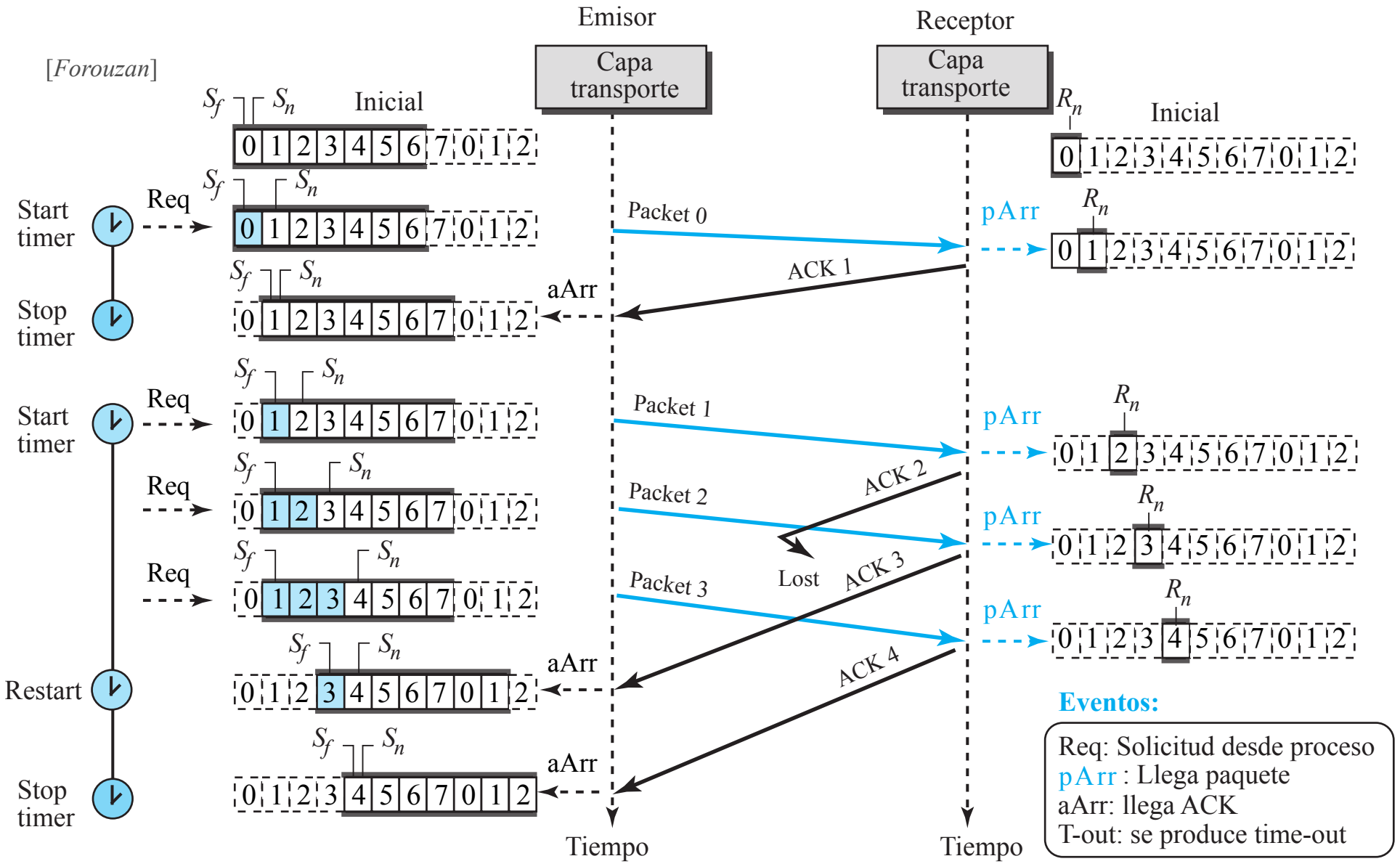


(a) Tamaño ventana de envío < 2^m (b) Tamaño ventana de envío = 2^m

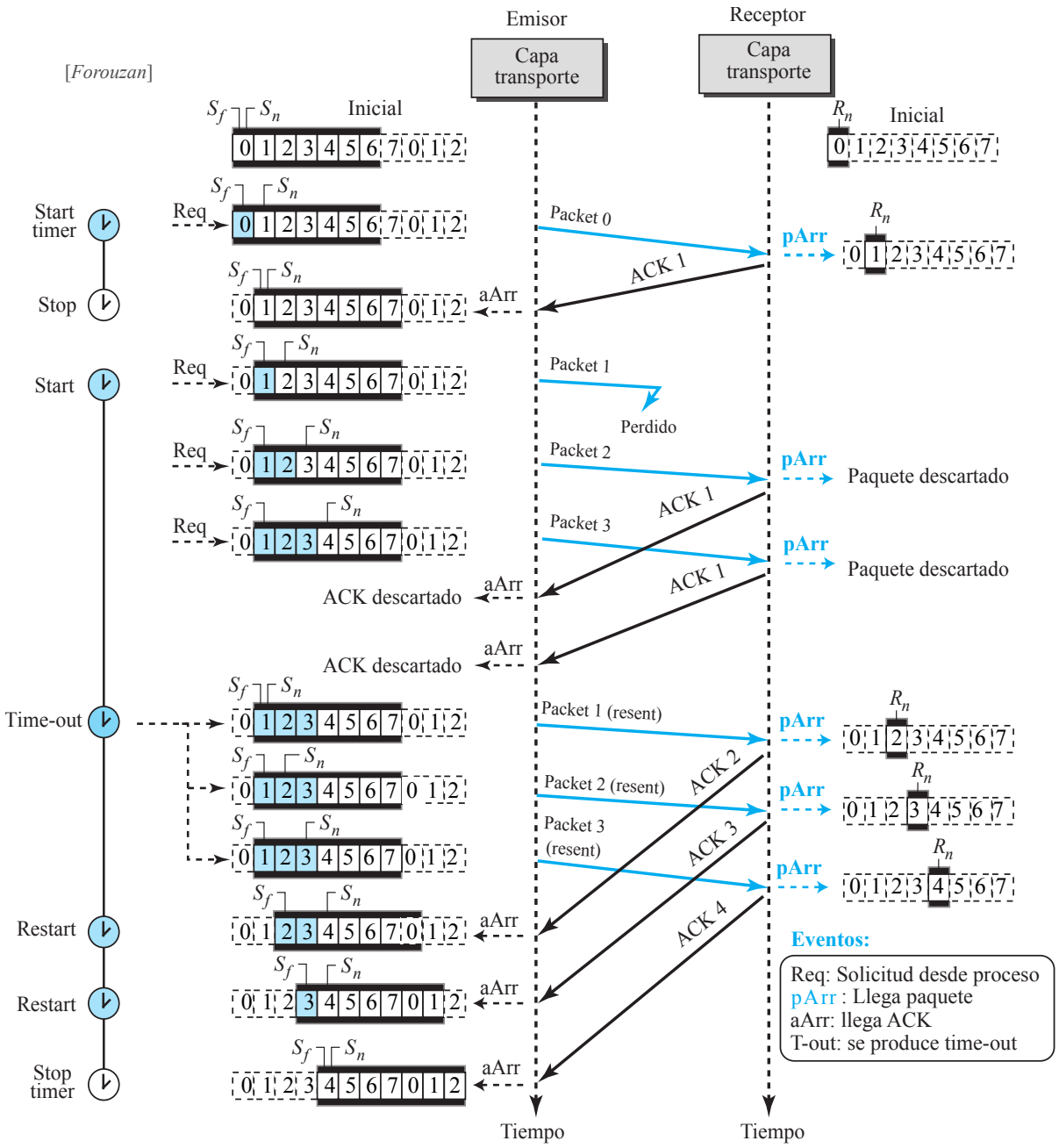
[*Forouzan*]

En Retroceso-N, el tamaño máximo de ventana de envío debe ser, como máximo, $2^m - 1$, mientras que el tamaño de ventana de recepción siempre es 1.

Protocolo *Retroceso-N*: confirmaciones acumulativas



Protocolo *Retroceso-N*: ejemplo



Retroceso-N vs Stop-and-wait

Podemos encontrar **similitudes** entre **Retroceso-N** vs **Stop-and-wait**:

- Stop-and-wait, realmente, es Retroceso-N en el que sólo hay dos números de secuencia y tamaño de ventana es 1.
- Por tanto, $m = 1$ y $2^m - 1 = 1$.
- En Retroceso-N, la aritmética es módulo 2^m, y Stop-and-wait es módulo 2, que es lo mismo que decir que, 2^m cuando $m = 1$.

Retroceso-N: ejemplos

HDLC

- Protocolo de enlace de datos ISO (1970)

TCP (*Transport Control protocol*)-Tahoe

- Protocolo de transporte en Internet

Rechazo selectivo

Protocolo rechazo selectivo

El protocolo Retroceso-N **simplifica** el proceso en el receptor:

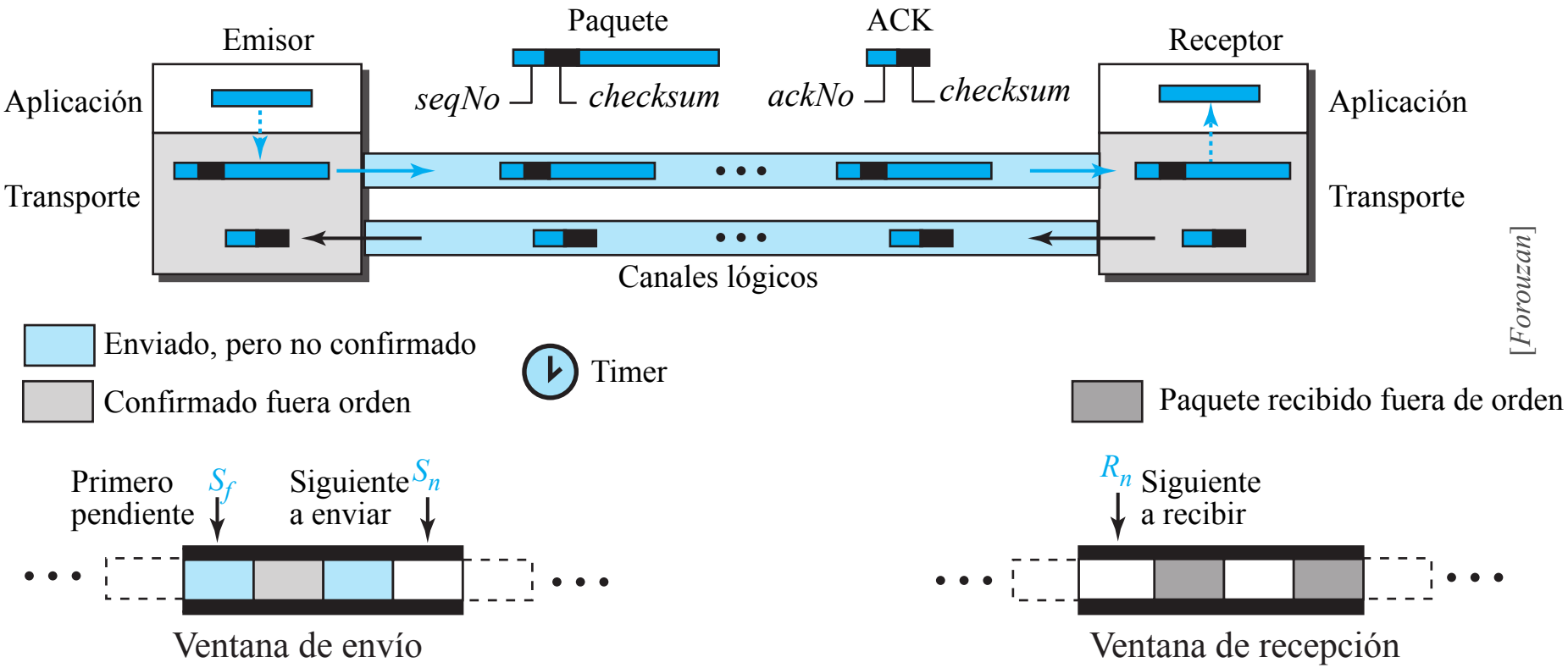
- Sólo mantiene una variable.
- No hay necesidad de almacenar paquetes fuera de secuencia, simplemente, los descarta.

Pero, Retroceso-N es **ineficiente** si el protocolo subyacente pierde muchos paquetes:

- Cada vez que un paquete se pierde o corrompe, el emisor reenvía todos los paquetes pendientes (retroceso).
- Incluso aquellos paquetes que habían llegado en buen estado pero fuera de secuencia.
- Si el nivel de red pierde muchos paquetes debido a congestión, el reenvío de los paquetes pendientes producirá un empeoramiento de la congestión, y en consecuencia, más paquetes se perderán.
- Se produce un efecto avalancha que puede producir el colapso total de la red.

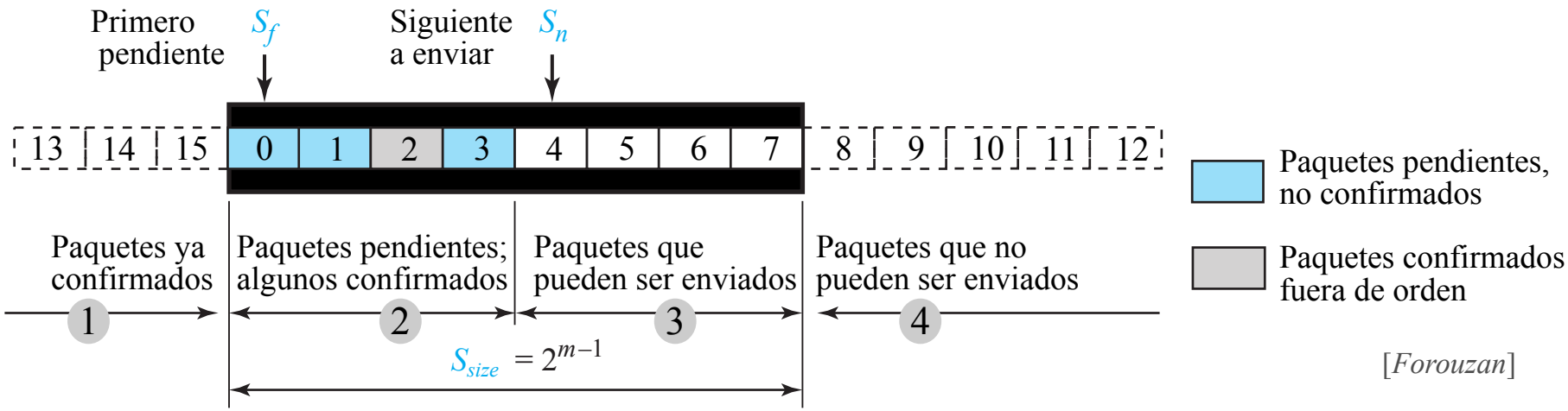
Para evitar los problemas anteriores, se plantea el protocolo de **Rechazo Selectivo** *(Selective Reject)*, como su nombre implica, los paquetes se reenvían de forma selectiva.

Rechazo selectivo: arquitectura



Rechazo selectivo también utiliza dos ventanas, pero con ciertas diferencias con respecto a Retroceso-N.

Rechazo selectivo: ventana emisor

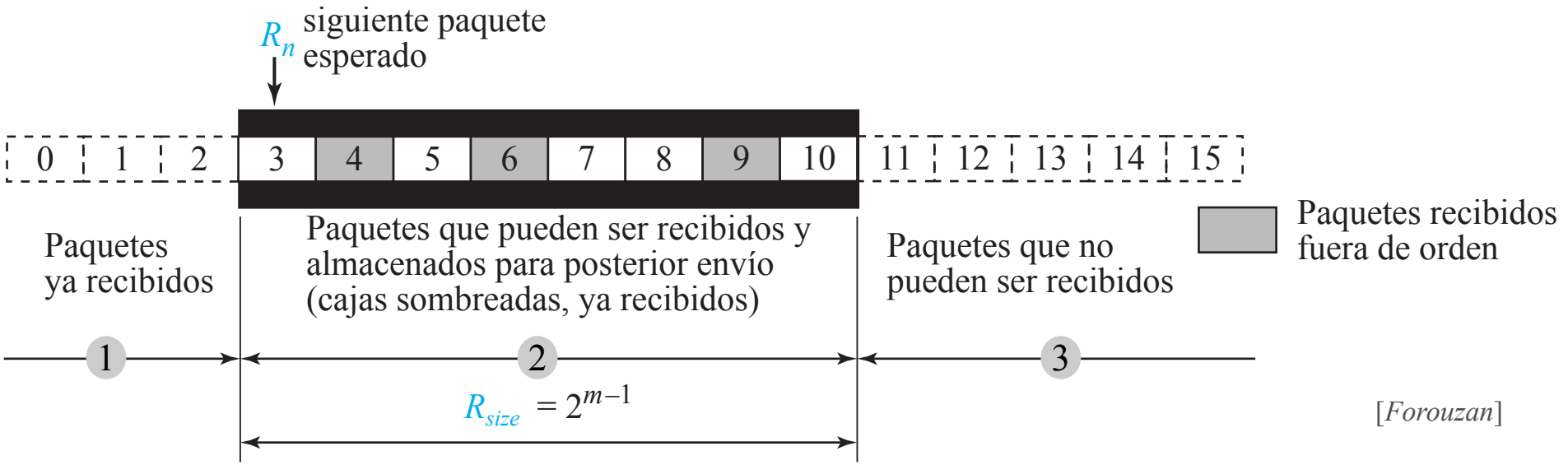


El tamaño máximo de la **ventana del emisor** es mucho más pequeño: 2^{m-1}.

Por ejemplo, si $m = 4$:

- La secuencia de números de secuencia: $0, 1, \cdots, 15$.
- Pero el tamaño máximo de ventana es 8 (recordemos que en Retroceso-N sería 15).

Rechazo selectivo: ventana receptor



La **ventana del receptor** es totalmente diferente a Retroceso-N:

- Mismo tamaño que ventana emisor: 2^{m-1}
- Permite que todos los paquetes de la ventana pueden llegar fuera de orden.
- Los paquetes se mantienen en el buffer hasta que se consigue un conjunto de paquetes consecutivos para enviar a nivel aplicación.
- Al tener el mismo tamaño de ventana emisor y receptor, realmente, **cualquier paquete puede llegar fuera de secuencia**.

Rechazo selectivo

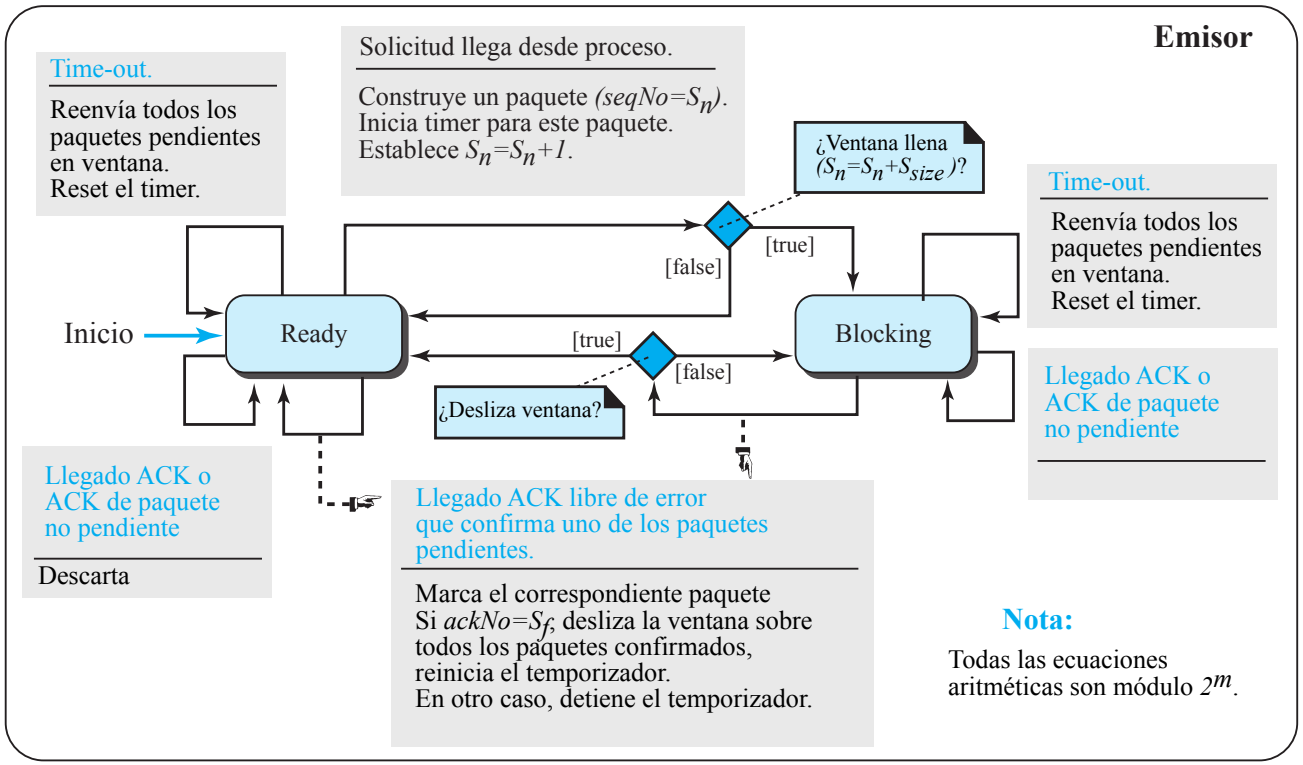
El uso de los **temporizadores** es diferente:

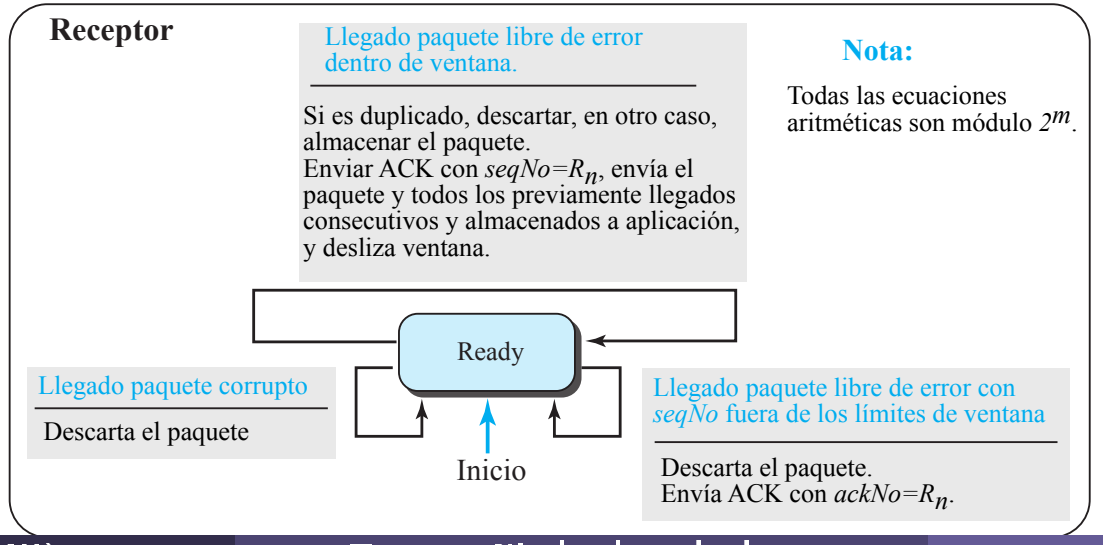
- Rechazo selectivo utiliza un temporizador para cada paquete pendiente.
- Cuando expira un temporizador, sólo se reenvía el correspondiente paquete.

Y también hay diferencias con respecto a las **confirmaciones**:

- En **Retroceso-N** una *ackNo* es **acumulativa**, define siguiente trama a recibir y confirmando todas las tramas que han sido recibidas.
- En **Rechazo selectivo**, la semántica de la confirmación es diferente: una *ackNo* define la confirmación de cada paquete de forma **individual**.

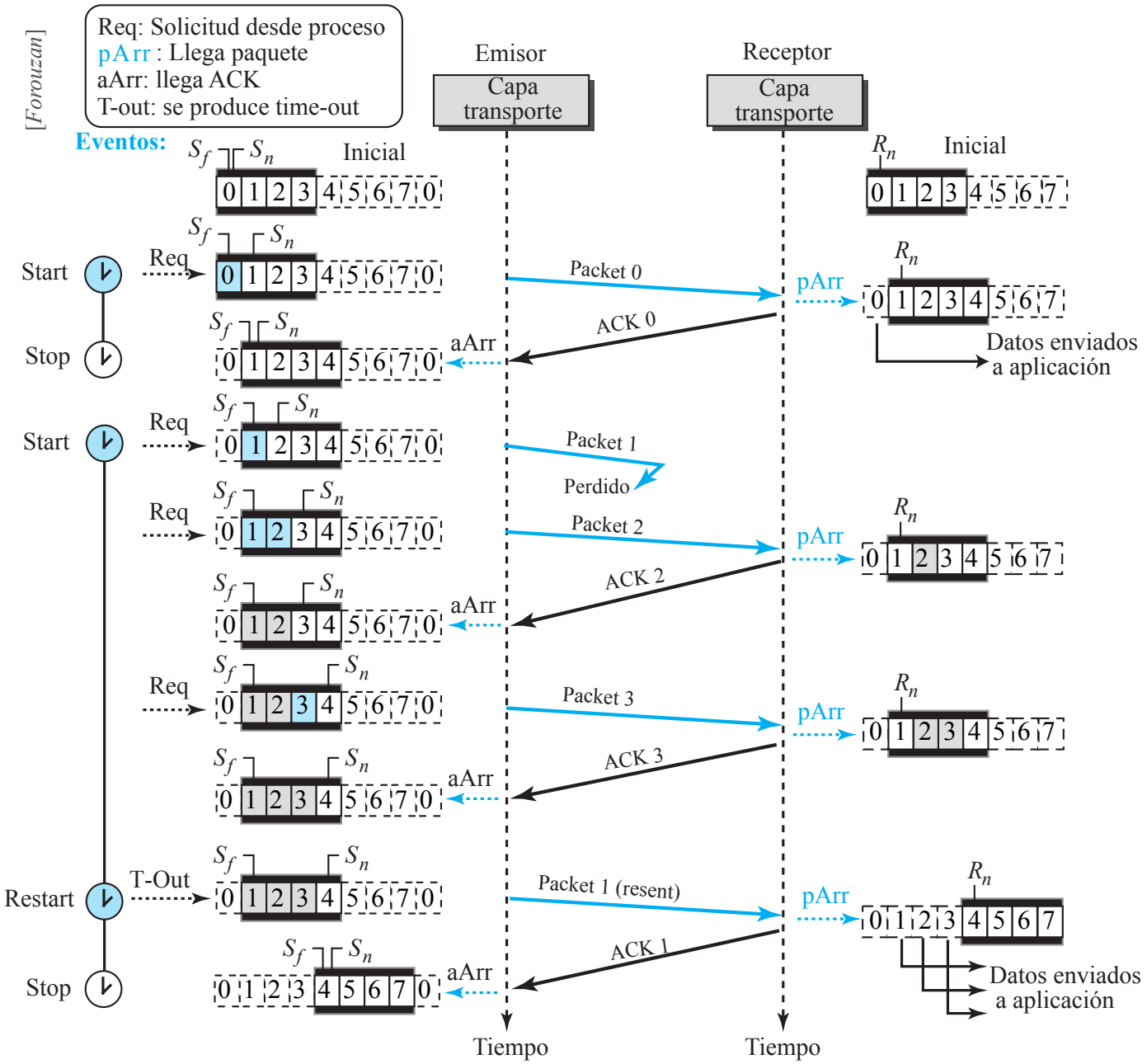
Rechazo selectivo: FSM



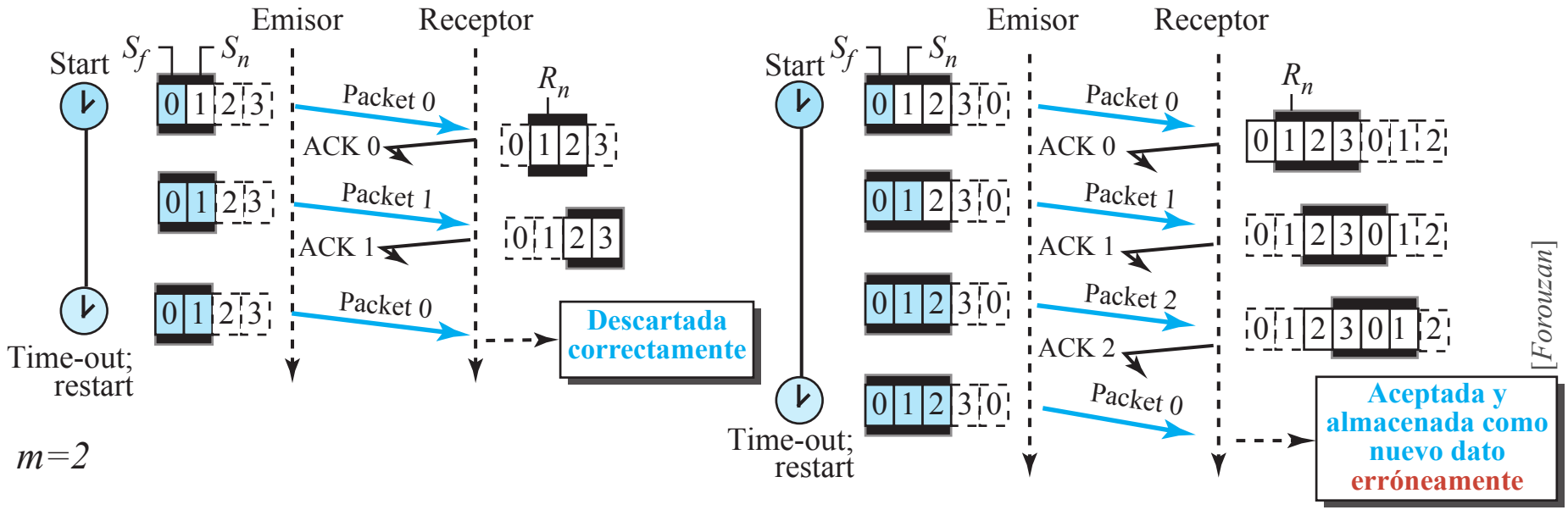


[Forouzan]

Rechazo selectivo: ejemplo



Rechazo selectivo: justificación tamaño ventana



(a) Ventana de envío y recepción de tamaño = 2^{m-1}. (b) Ventana de envío y recepción de tamaño > 2^{m-1}.

¿Por qué los tamaños de ventana emisor/receptor deben ser, como máximo, 2^{m-1}?

Supongamos $m = 2$:

- Si el **tamaño de ventana** es $2^{2-1} = 2$ y todas las confirmaciones se han perdido, el temporizador del paquete 0 expira, y por tanto, se reenvía. Además, la ventana del receptor está esperando el paquete 2, y no el paquete 0, por lo tanto, el paquete duplicado se descarta correctamente (la secuencia 0 no está en la ventana).
- Si el **tamaño de ventana** es $2^2 - 1 = 3$ y todas las confirmaciones se han perdido, el emisor reenvía un duplicado del paquete 0. En este momento, la ventana del receptor está esperando el paquete 0 (ya que 0 es parte de la ventana), por lo tanto, acepta el paquete 0, no como un duplicado, sino como un paquete del siguiente ciclo, lo que es, claramente, **erróneo**.

Rechazo selectivo: ejemplos

HDLC

- Protocolo de enlace de datos ISO (1970) - en modo Rechazo Selectivo.

TCP (*Transport Control protocol*)-Reno

- Protocolo de transporte en Internet (posterior a Tahoe)

Ventana deslizante: análisis de prestaciones

Ventana deslizante: medio perfecto

Definimos:

$$a = \frac{t_p}{t_t}$$

Tiempo de transmisión normalizado a la unidad $(t_t = 1)$, por tanto, $a = t_p$.

Se consideran despreciables:

- Tiempo de procesamiento de trama (t_{proc}).
- Tiempo transmisión de tramas ACK (t_{ack}).

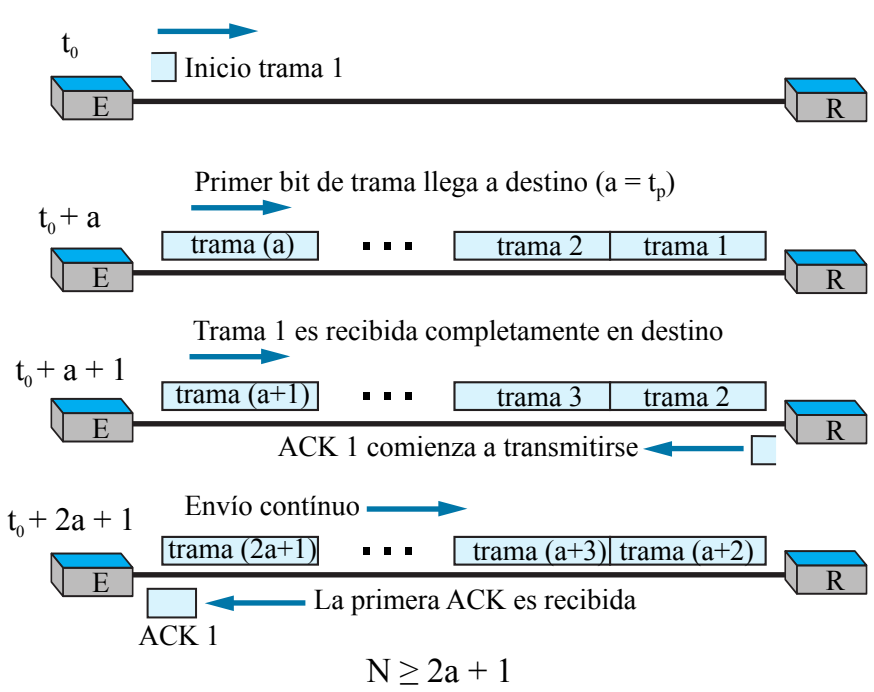
Ancho de ventana definido por N.

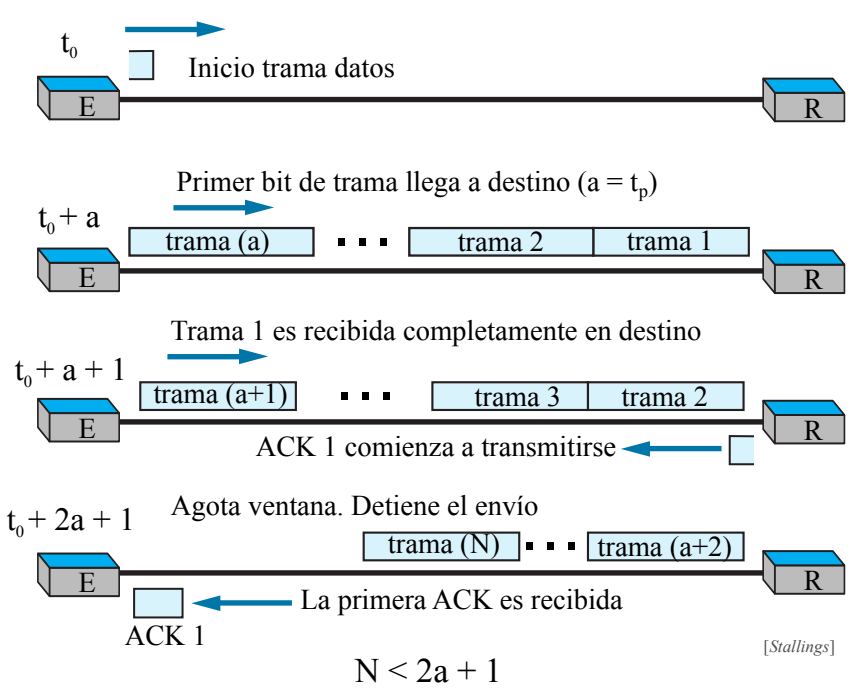
Suponemos **medio perfecto**, todas las tramas alcanzan el destino libres de error.

La eficiencia:

Ventana deslizante: $$U = \begin{cases} 1 & N \geq 2a + 1 \\ \frac{N}{2a+1} & N < 2a + 1 \end{cases}$$

Ventana deslizante: análisis de prestaciones (II)





La confirmación (`ACK`) de la `trama 1` alcanza `T` antes de agotar ventana.

Por tanto, se produce envío continuo al transmitirse $2a+1$ tramas en $2a+1$ unidades de tiempo.

$$U = \frac{2a+1}{2a+1} = 1$$

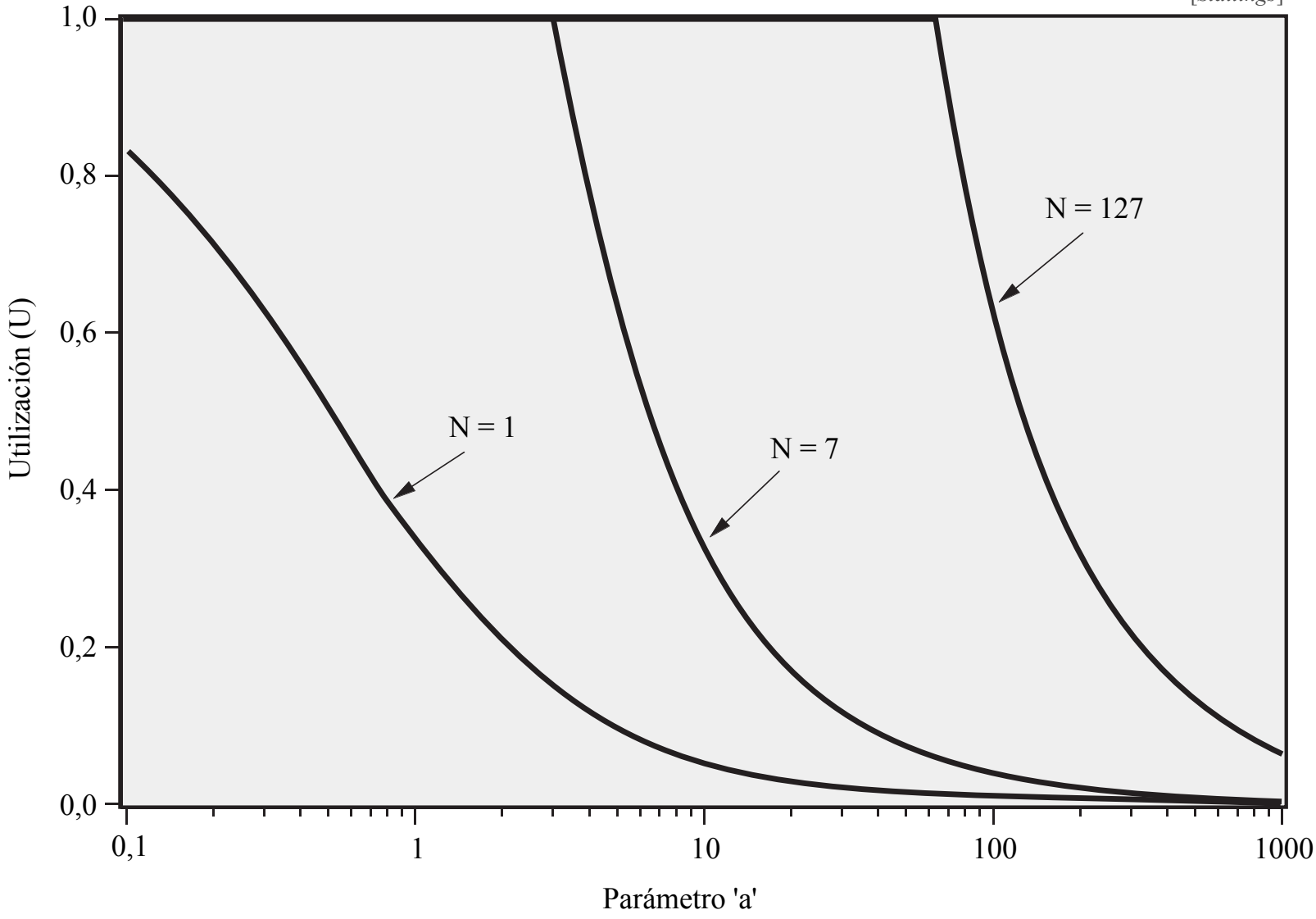
`T` agota su ventana en $t_0 + N$ y no podrá enviar tramas hasta que le llegue la primera `ACK` en $t_0 + 2a + 1$.

Por tanto, se han transmitido N tramas en $(2a+1)$ unidades de tiempo.

$$U = \frac{N}{2a+1}$$

Ventana deslizante: comparativa

[*Stallings*]



Rechazo selectivo

Ya vimos que el cálculo de la eficiencia en protocolos de ventana deslizante en el caso de escenarios sin error, se obtiene mediante:

$$U = \begin{cases} 1 & N \geq 2a+1 \\ \frac{N}{2a+1} & N < 2a+1 \end{cases}$$

Para el cálculo de la eficiencia en el Protocolo de Rechazo selectivo se puede utilizar el mismo razonamiento que en Stop-and-wait.

Las ecuaciones obtenidas para medios perfectos (no error), se deben modificar para incluir el parámetro relacionado con la probabilidad de trama errónea P.

Para ello, se dividen por el nº de retransmisiones, N_r, siendo:

$$N_r = \frac{1}{1-P}$$

por tanto:

Rechazo selectivo: $$U = \begin{cases} 1-P & N \geq 2a+1 \\ \frac{N(1-P)}{2a+1} & N < 2a+1 \end{cases}$$

Retroceso-N

El mismo razonamiento se puede aplicar al esquema Retroceso-N, pero en este caso, la aproximación de N_r es diferente, ya que por cada error es preciso retransmitir K tramas, en lugar de una sola, como se ha considerado hasta ahora. Por tanto:

$$N_r = E[\text{número de tramas para conseguir una correcta}] = \sum_{i=1}^{\infty} f(i) P^{i-1}(1-P)$$

donde $f(i)$ es el número total de tramas transmitidas si la trama original se debe transmitir i veces:

$$f(i) = 1 + (i-1)K = (1-K) + Ki$$

Sustituyendo y utilizando la siguiente propiedad:

$$\sum_{i=1}^{\infty} (X^{i-1}) = \frac{1}{1-X} \text{ para } (-1 < X < 1)$$

obtenemos:

$$N_r = (1-K)\sum_{i=1}^{\infty} P^{i-1}(1-P) + K\sum_{i=1}^{\infty} iP^{i-1}(1-P) = 1 - K + \frac{K}{1-P} = \frac{1-P+KP}{1-P}$$

Para la aproximación de K, observando la Figura (página 141), se puede concluir que:

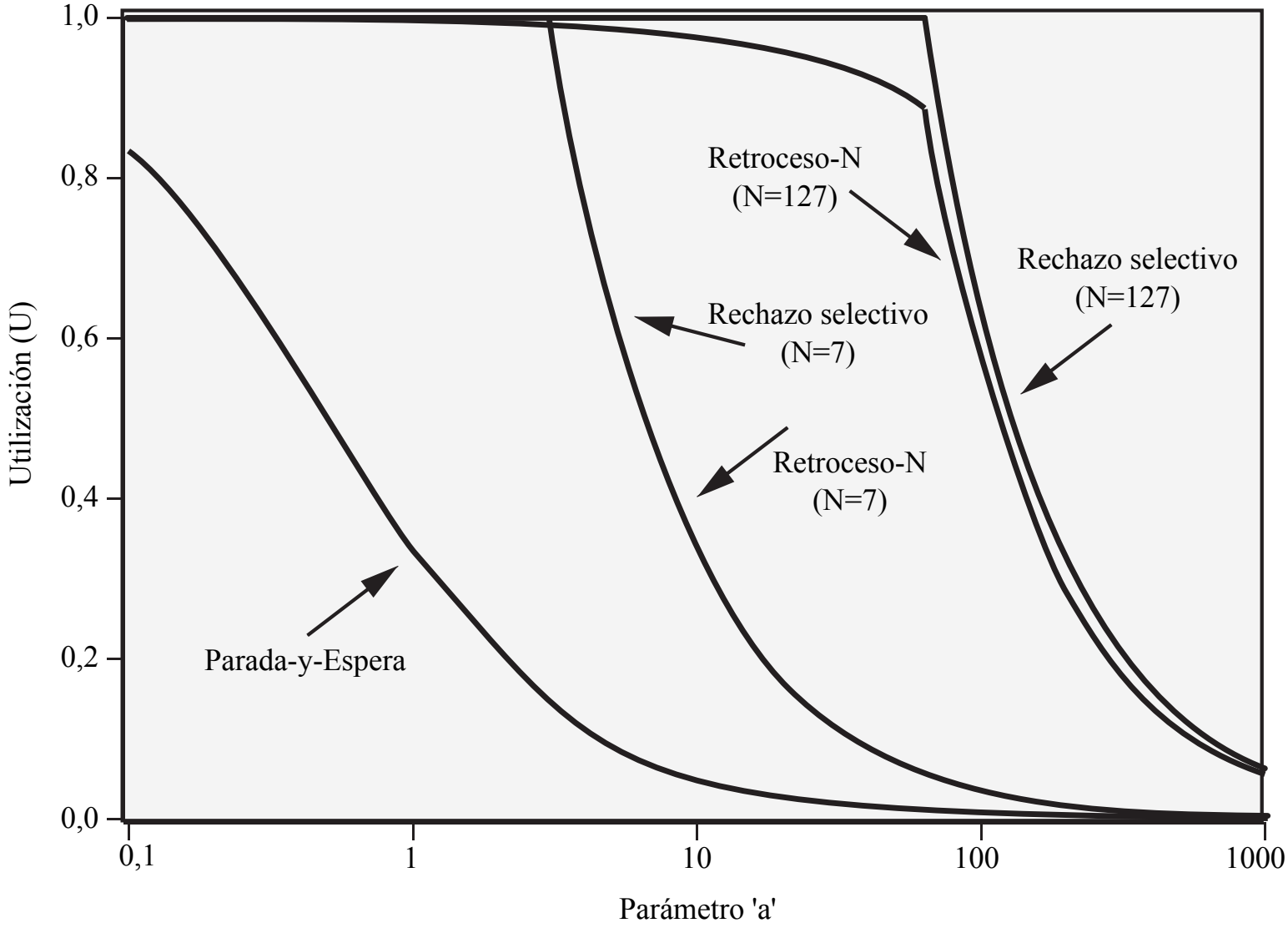
$$K \approx \begin{cases} 2a+1 & N \geq (2a+1) \\ N & N < (2a+1) \end{cases}$$

por lo tanto:

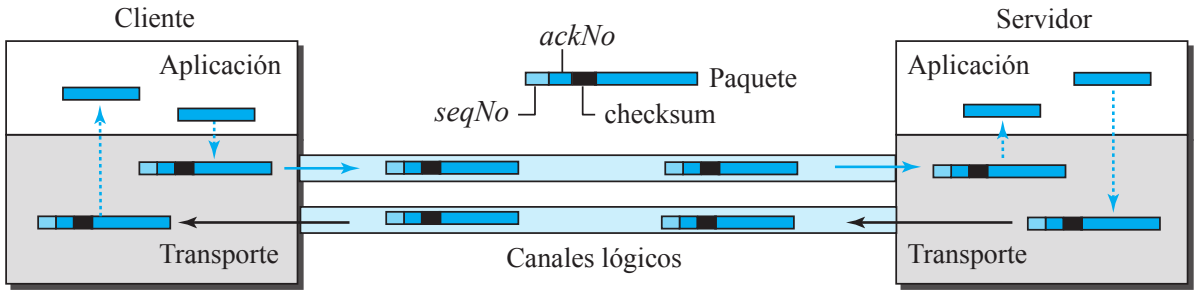
Retroceso-N: $$U = \begin{cases} \frac{1-P}{1+2aP} & N \geq 2a+1 \\ \frac{N(1-P)}{(2a+1)(1-P+NP)} & N < 2a+1 \end{cases}$$

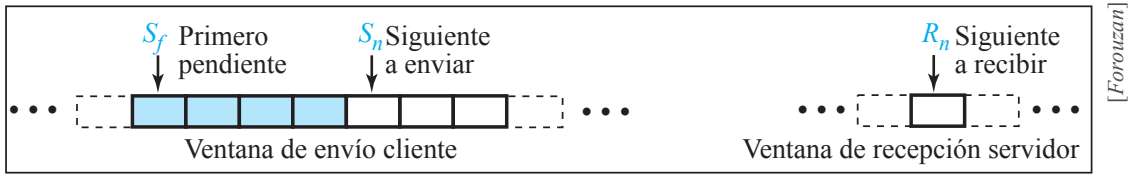
Análisis de prestaciones

[*Stallings*]

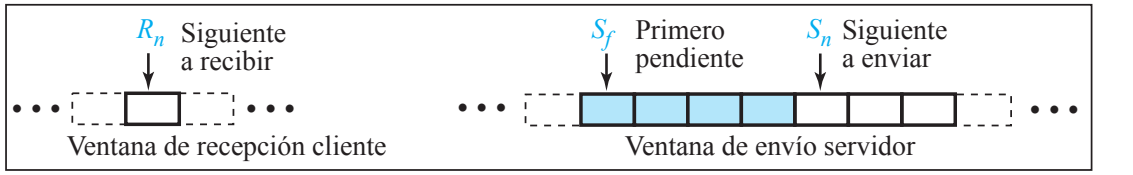


Protocolos bidireccionales: *piggybacking*





Ventana para comunicación de cliente → servidor



Ventana para comunicación de servidor → cliente

Los cuatro protocolos presentados son **unidireccionales**: el flujo de paquete de datos sólo va en una dirección y las confirmaciones viajan en la otra.

Normalmente el flujo de datos es **bidireccional**: del cliente al servidor y del servidor al cliente.

Por lo tanto, las confirmaciones también necesitan viajar en ambas direcciones.

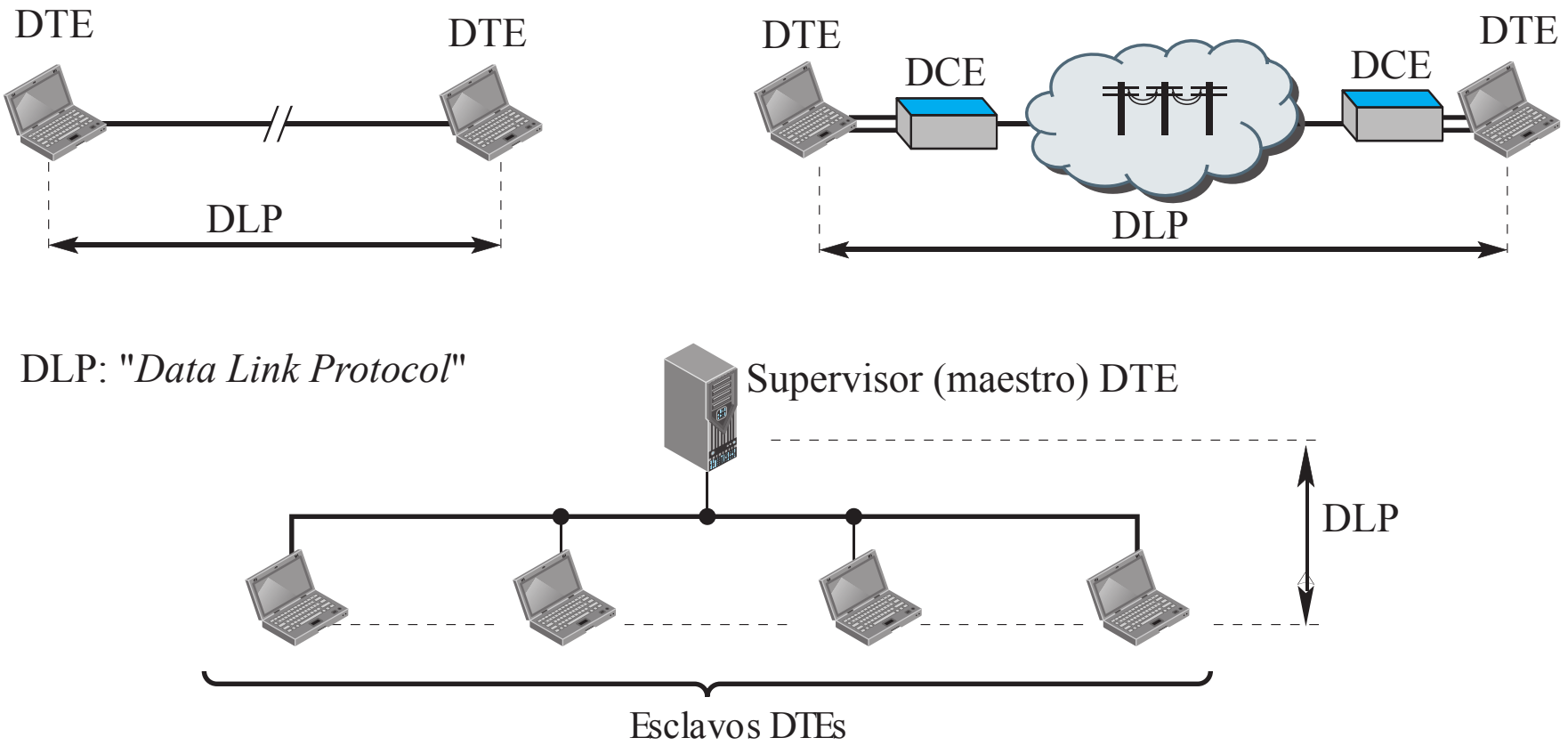
Se utiliza la técnica del **piggybacking** para mejorar la eficiencia de los protocolos unidireccionales.

Cuando un paquete de datos viaja $A \rightarrow B$, también transporta las confirmaciones de los paquetes $B \rightarrow A$.

Por lo tanto, es necesaria la implementación de dos mecanismos de ventana, **uno por cada sentido de la comunicación**.

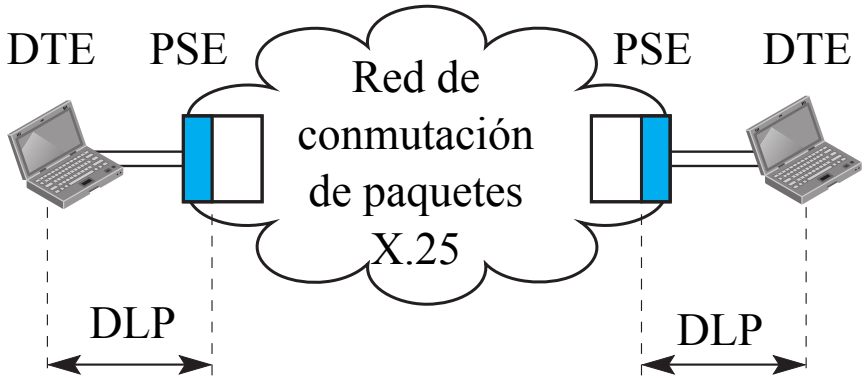
1. Funciones de nivel de enlace
2. Direccionamiento
3. Control de errores
4. Entramado
5. Control de flujo
6. ***HDLC***

Aplicación protocolos de enlace (I)

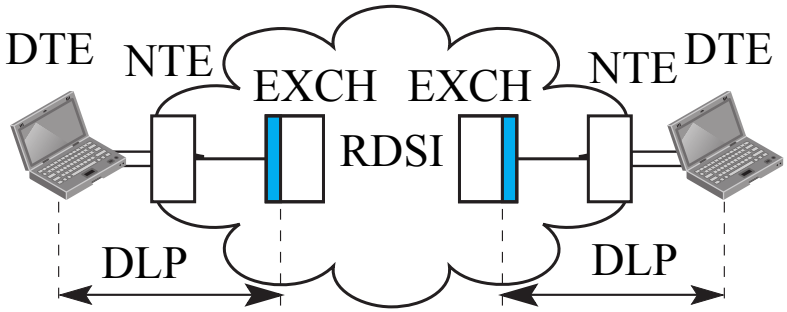


Los **protocolos a nivel de enlace** deben de ser capaces de solucionar las tareas de enlace de datos en cualquier escenario previsto: enlaces punto a punto, multipunto, paso de testigo, redes de conmutación de paquetes, etc.

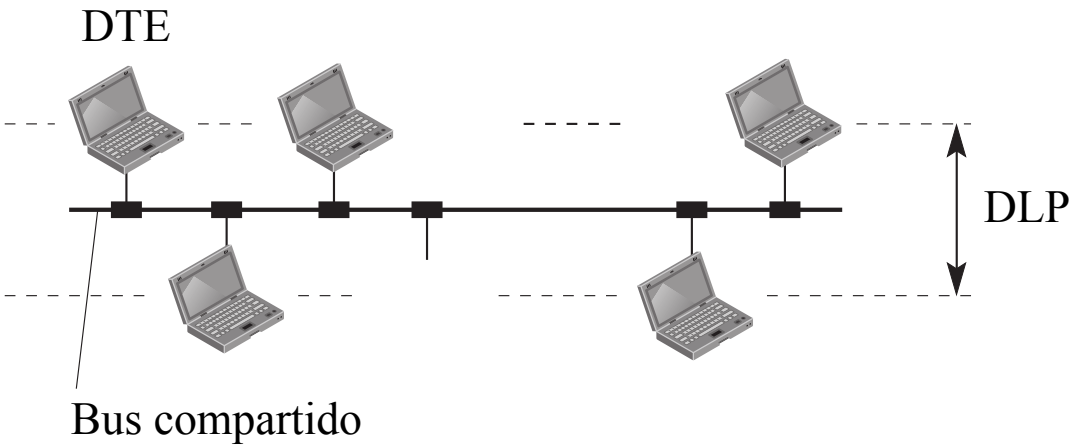
Aplicación protocolos de enlace (II)

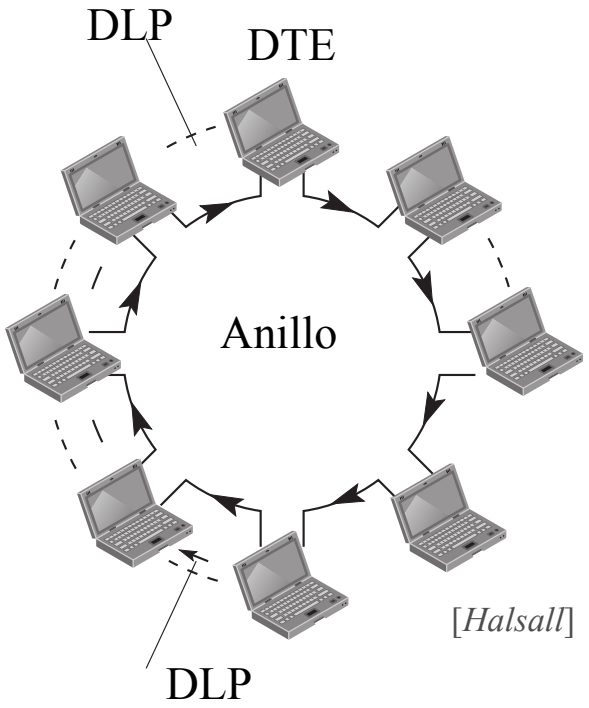


PSE: Central de conmutación de paquetes
DLP: Protocolo de enlace de datos



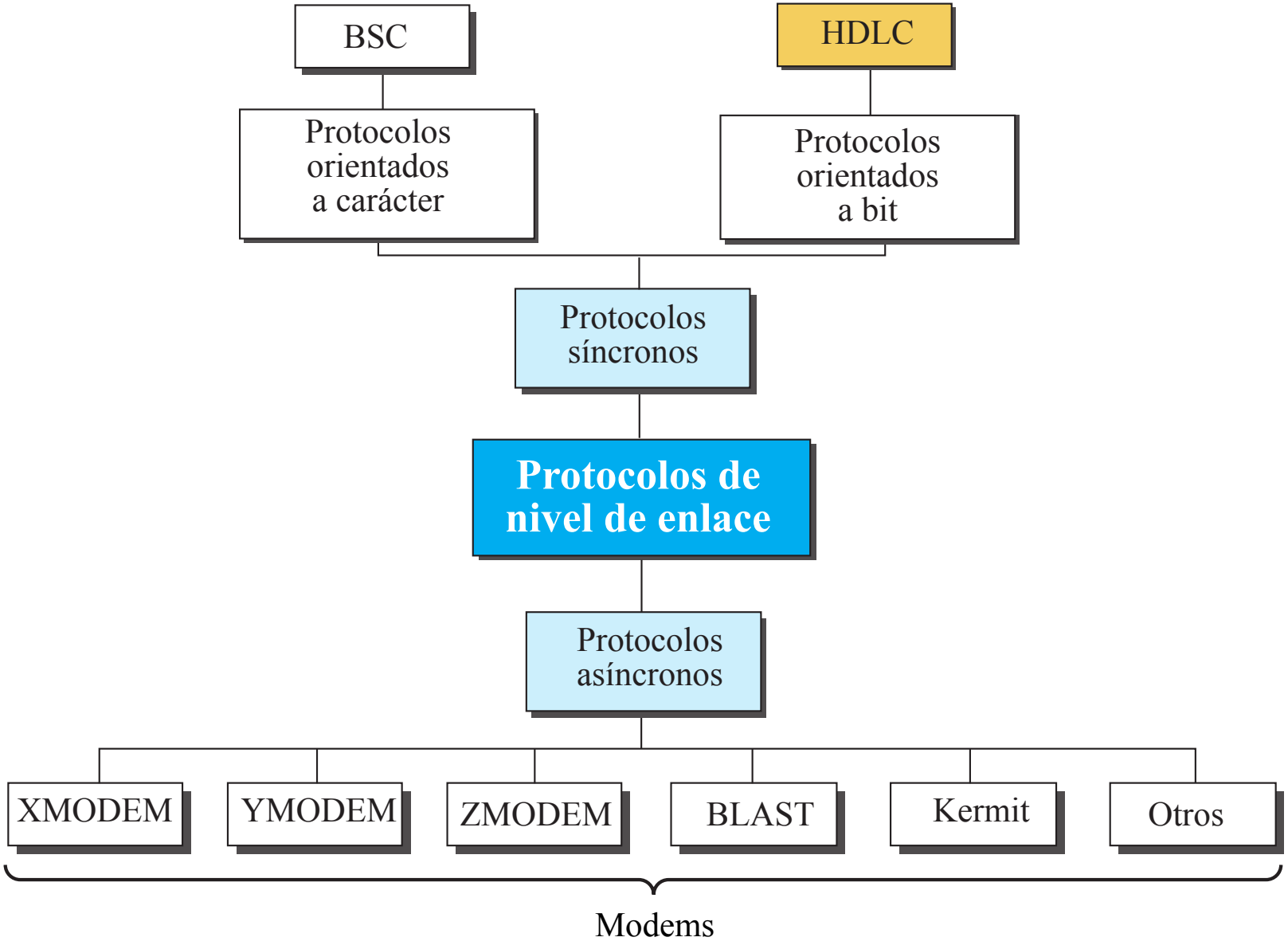
EXCH: Central de conmutación
NTE: Equipo terminal de red





[*Halsall*]

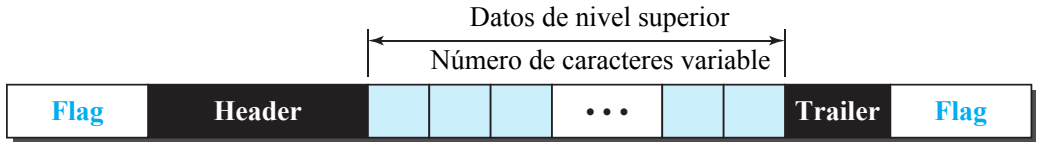
Protocolos de nivel de enlace



Protocolos síncronos

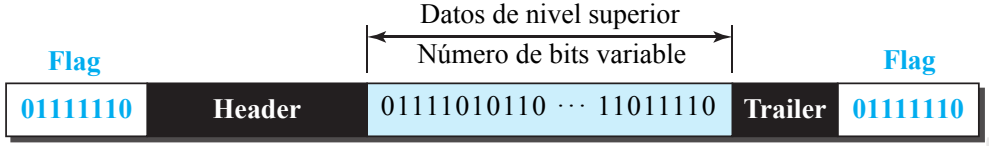
Protocolos orientados a carácter:

- Protocolo orientado a byte
- Sucesión de caracteres (ASCII/EBCDIC)
- Tramas de control y tramas de datos
- Actividad unidireccional
- Poco eficientes
- BSC (*Binary Synchronous Communication*), IBM
- Obsoletos



Protocolos orientados a bit:

- Sucesión de bits de forma individual
- Adaptable a diferentes configuraciones/aplicaciones
- Actividad bidireccional simultánea
- Gran eficiencia y fiabilidad
- Transparencia de datos
- HDLC (*High Level Data Link Control*), ISO
- Utilizado en multitud de protocolos: Ethernet, WIFI, etc.



HDLC (*High Level Data Link Control*)

HDLC es un protocolo de comunicaciones de propósito general punto a punto y multipunto, que opera a nivel de enlace de datos.

Aunque este protocolo es más teórico que práctico, muchos de los conceptos desarrollados en HDLC han sido aplicados a otros estándares tales como PPP, Ethernet o WLANs.

Proporciona recuperación de errores en caso de pérdida de paquetes de datos, fallos de secuencia y otros, por lo que ofrece una comunicación confiable entre emisor y receptor.

HDLC viene definido por:

- Tipos de estación
- Configuración de las estaciones
- Modos de operación

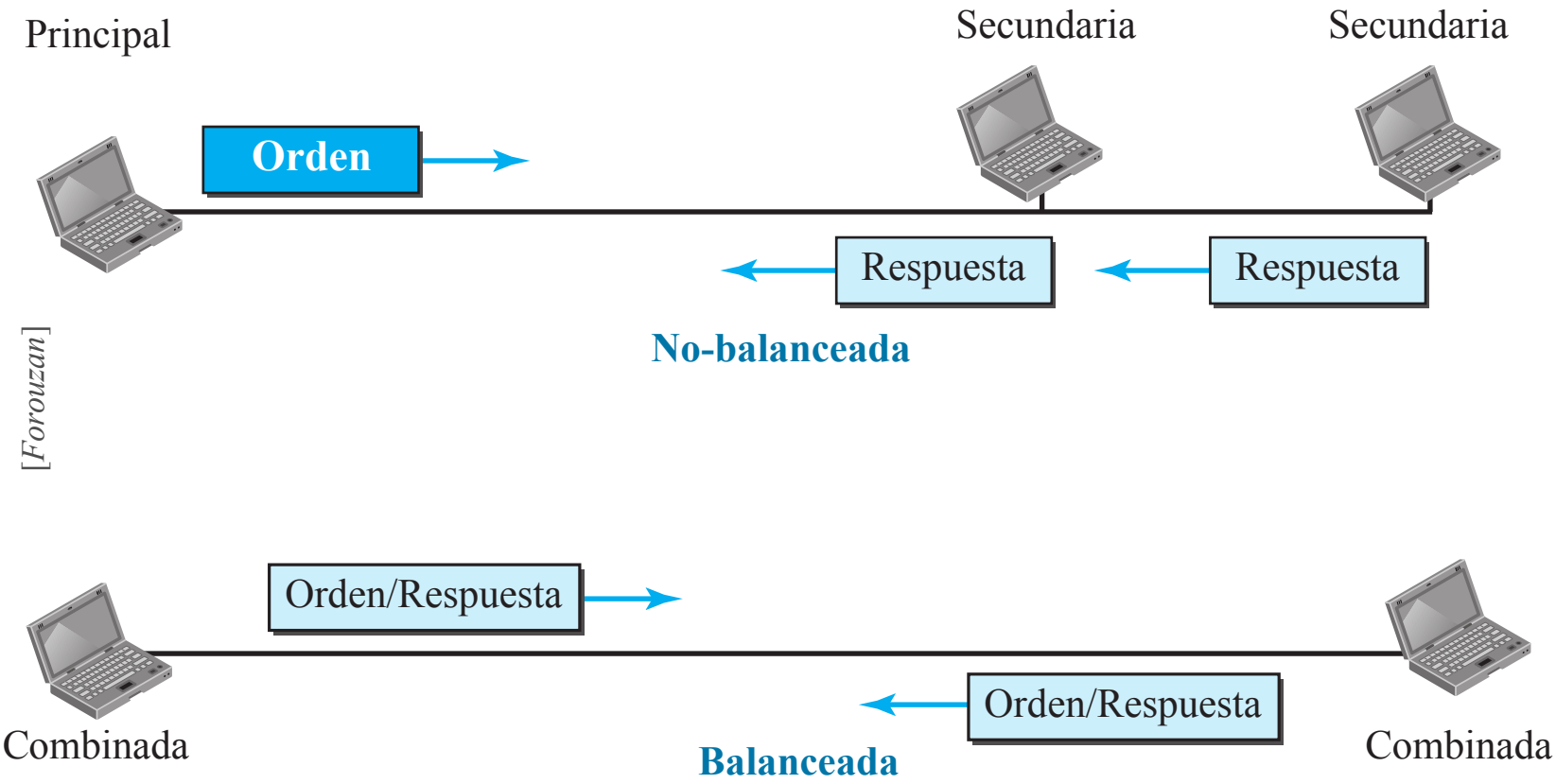
Tipos de estación

Primaria: se caracteriza porque tiene la responsabilidad de controlar el funcionamiento del enlace.
Envía **órdenes**.

Secundaria: funciona bajo el control de la estación primaria.
La primaria establece un enlace lógico independiente para cada una de las secundarias presentes en la línea.
Envía **respuestas**.

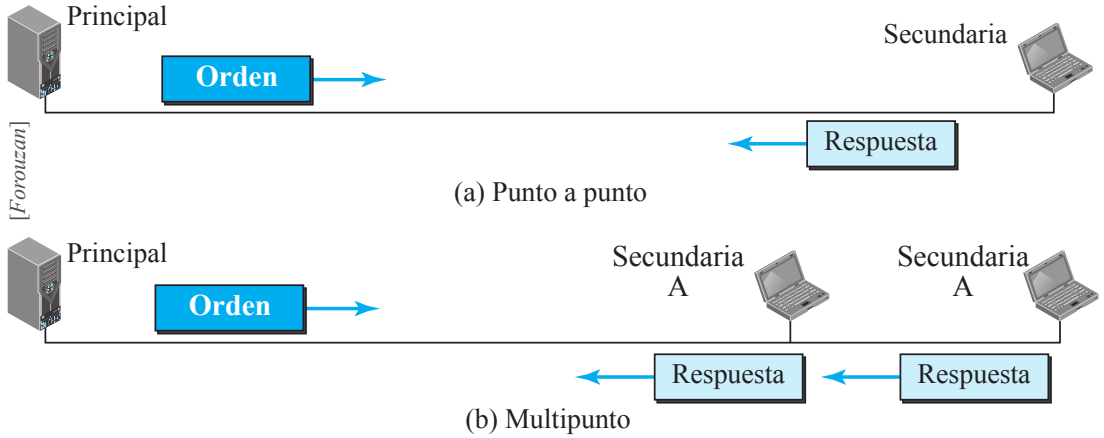
Combinada: es una mezcla entre las características de las primarias y las secundarias.
Una estación de este tipo puede generar tanto **órdenes** como **respuestas**.

Configuración



Modos de operación

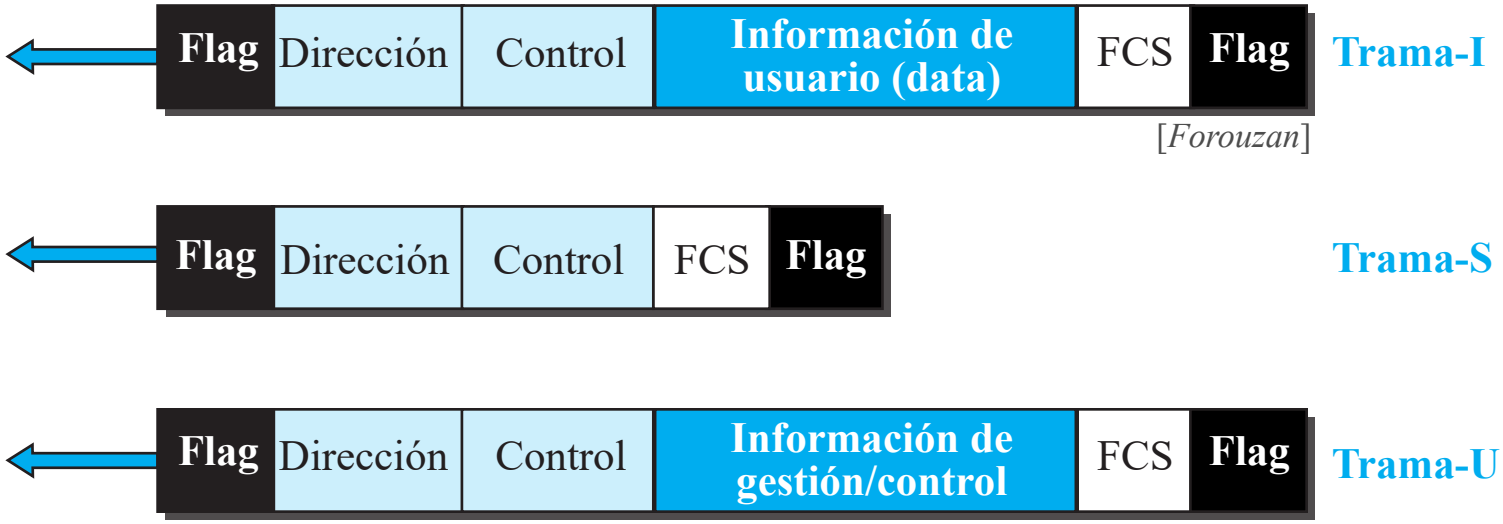
NRM, *Normal Response Mode*: utilizado en configuración no balanceada. La estación primaria puede iniciar la transferencia de datos a la secundaria, pero la secundaria solo puede transmitir datos mediante respuestas a las órdenes emitidas por la primaria.



ABM, *Asynchronous Balanced Mode*: utilizado en configuración balanceada. En este modo cualquier estación combinada podrá iniciar la transmisión sin necesidad de recibir permiso por parte de la otra estación combinada.

ARM, *Asynchronous Response Mode*: utilizado en configuración no balanceada. La estación secundaria puede iniciar la transmisión sin tener permiso explicito por parte de la primaria. La estación primaria sigue teniendo la responsabilidad del funcionamiento de la línea, incluyendo iniciación, recuperación de errores y desconexión lógica.

Tipos de tramas

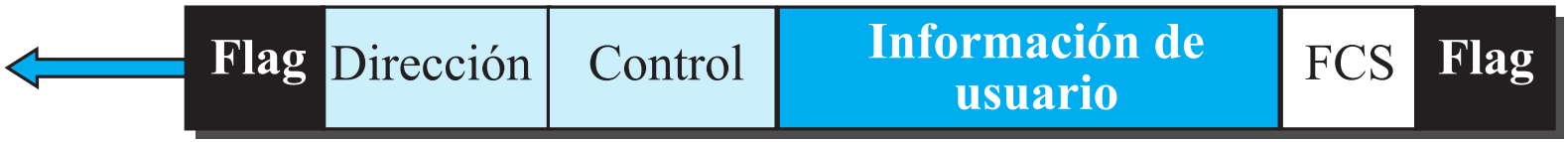


Campo *Flag*

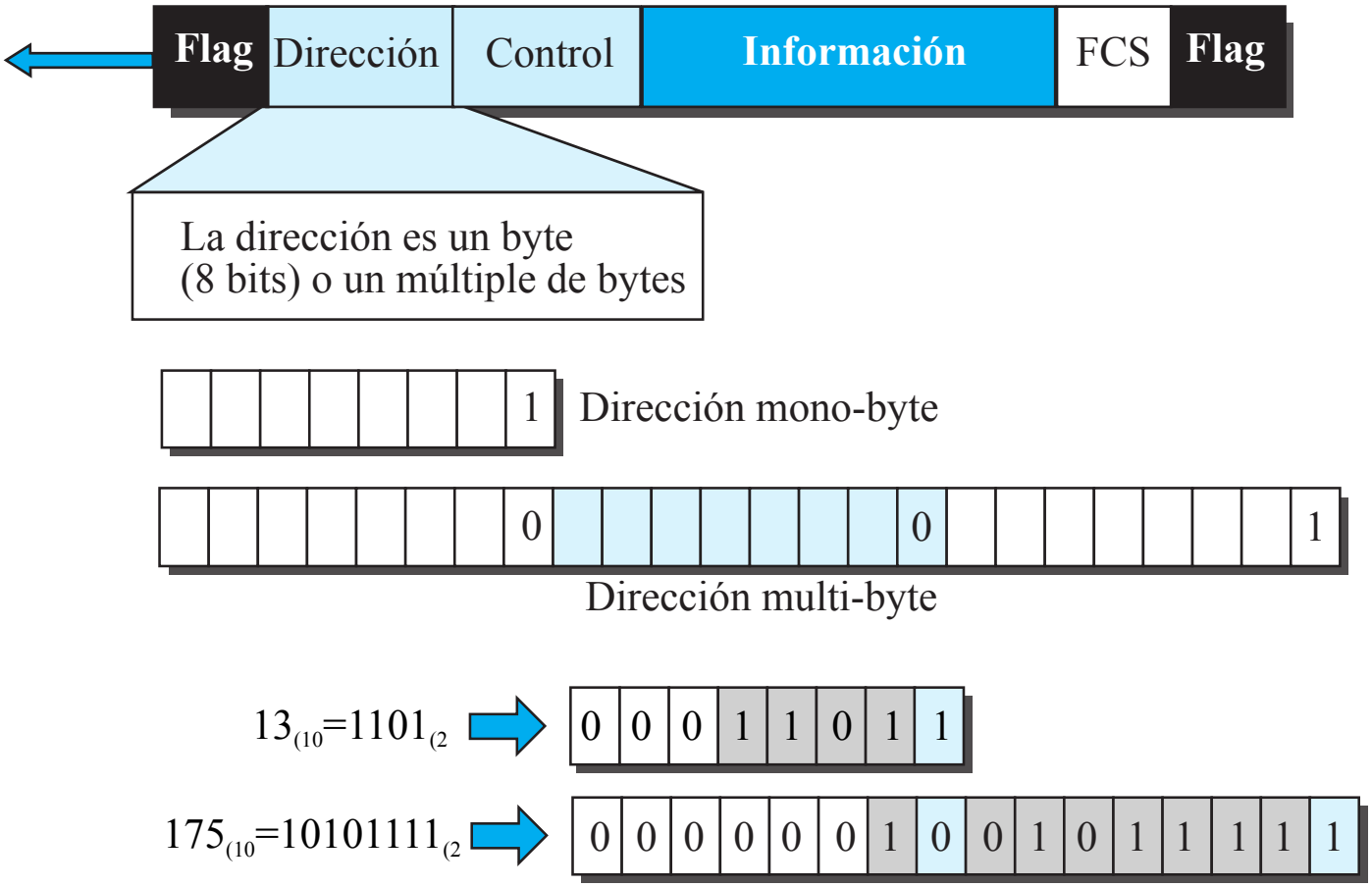
El flag es un patrón fijo de 8 bits. Formado por 6 unos entre 2 ceros. Hay un flag al inicio y al final de la trama.
El flag de finalización de una trama se puede utilizar como flag de inicio de la siguiente.

01111110

[*Forouzan*]

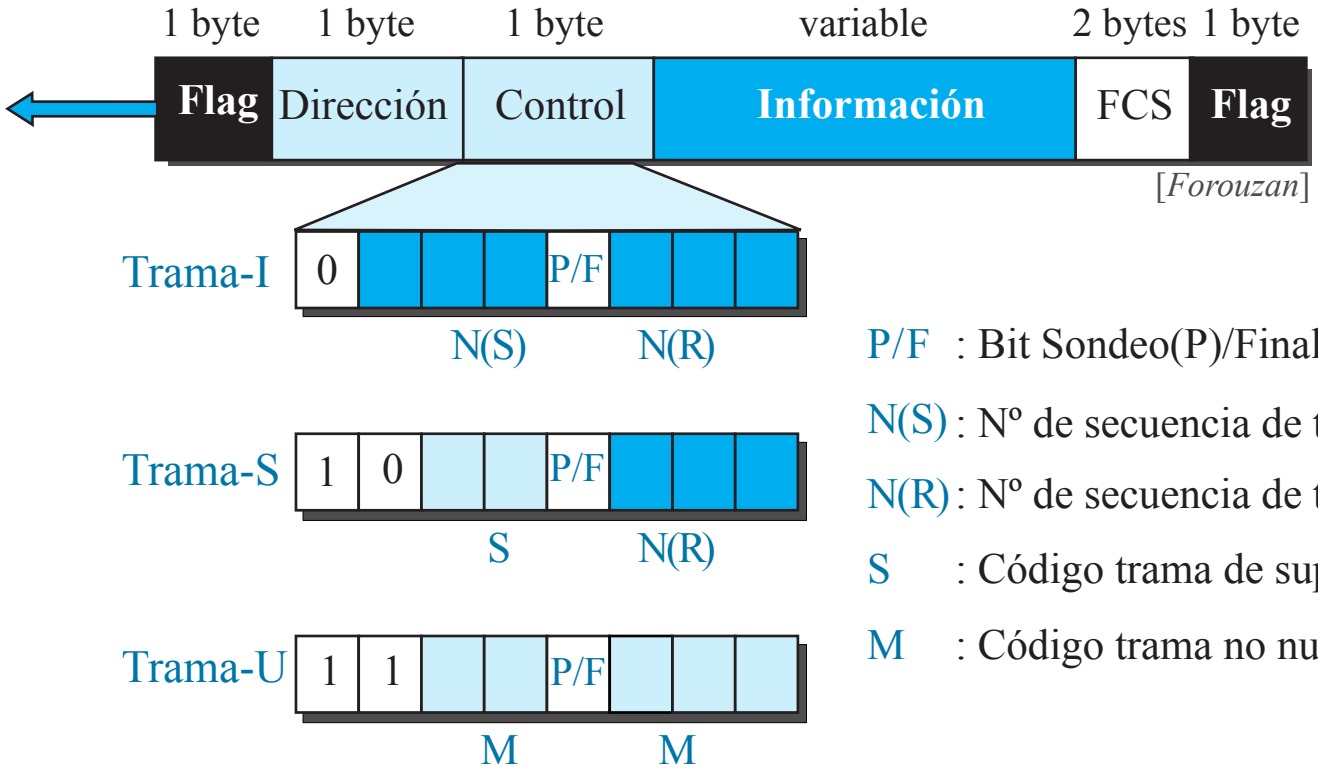


Campo *Dirección*



- HDLC permite la extensión del bit dirección para entornos multipunto con elevado número de dispositivos.
- El campo dirección siempre contiene la **dirección de la estación secundaria**.

Campo *Control*



P/F : Bit Sondeo(P)/Final(F)

N(S) : Nº de secuencia de trama enviada

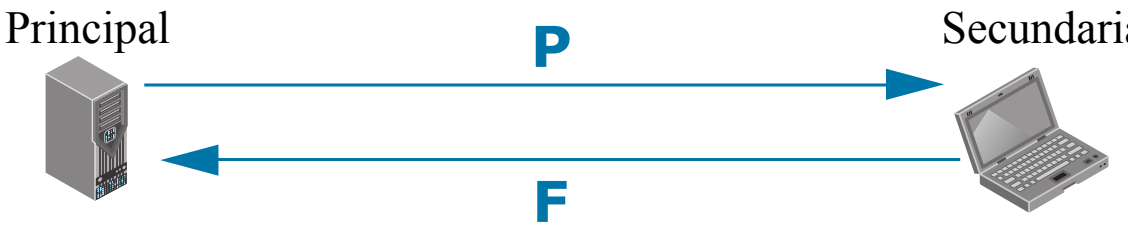
N(R) : Nº de secuencia de trama esperada

S : Código trama de supervisión

M : Código trama no numerada

El campo **control** define el tipo de trama mediante los bits correspondientes.

Campo *Control*: bit *P/F*



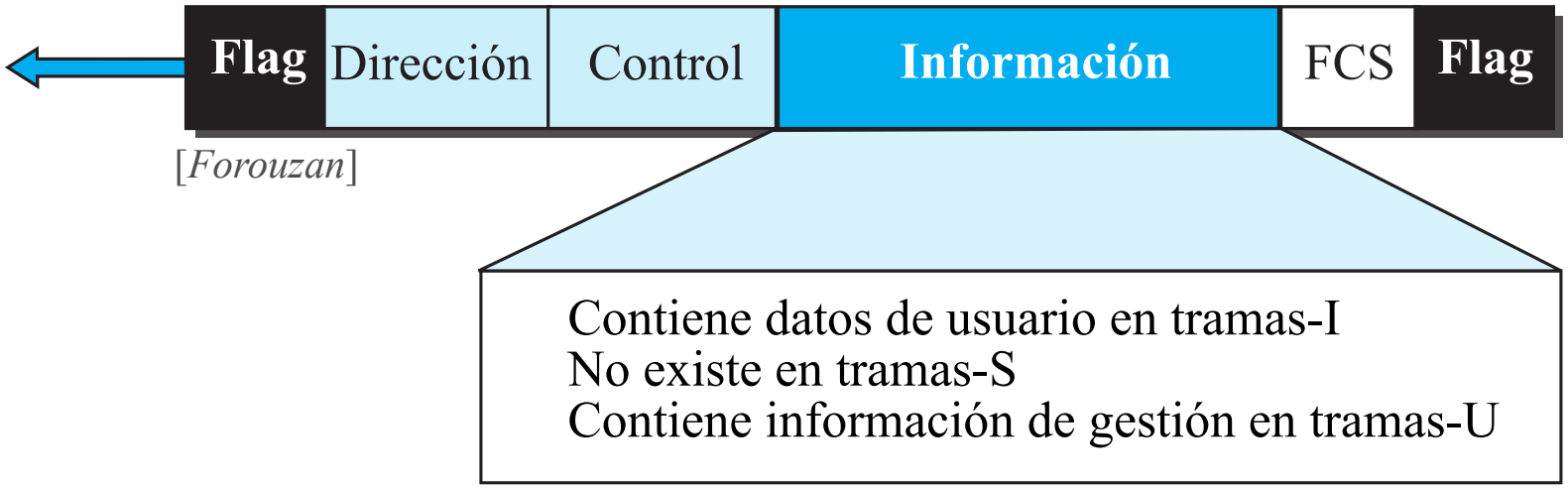
El **bit P/F** se define en el campo de *control* de la trama HDLC.

El bit activo en una **orden**, es decir, $\mathbf{P} = \mathbf{1}$, obliga a la secundaria a responder, dependiendo del tipo de trama y de la orden de supervisión.

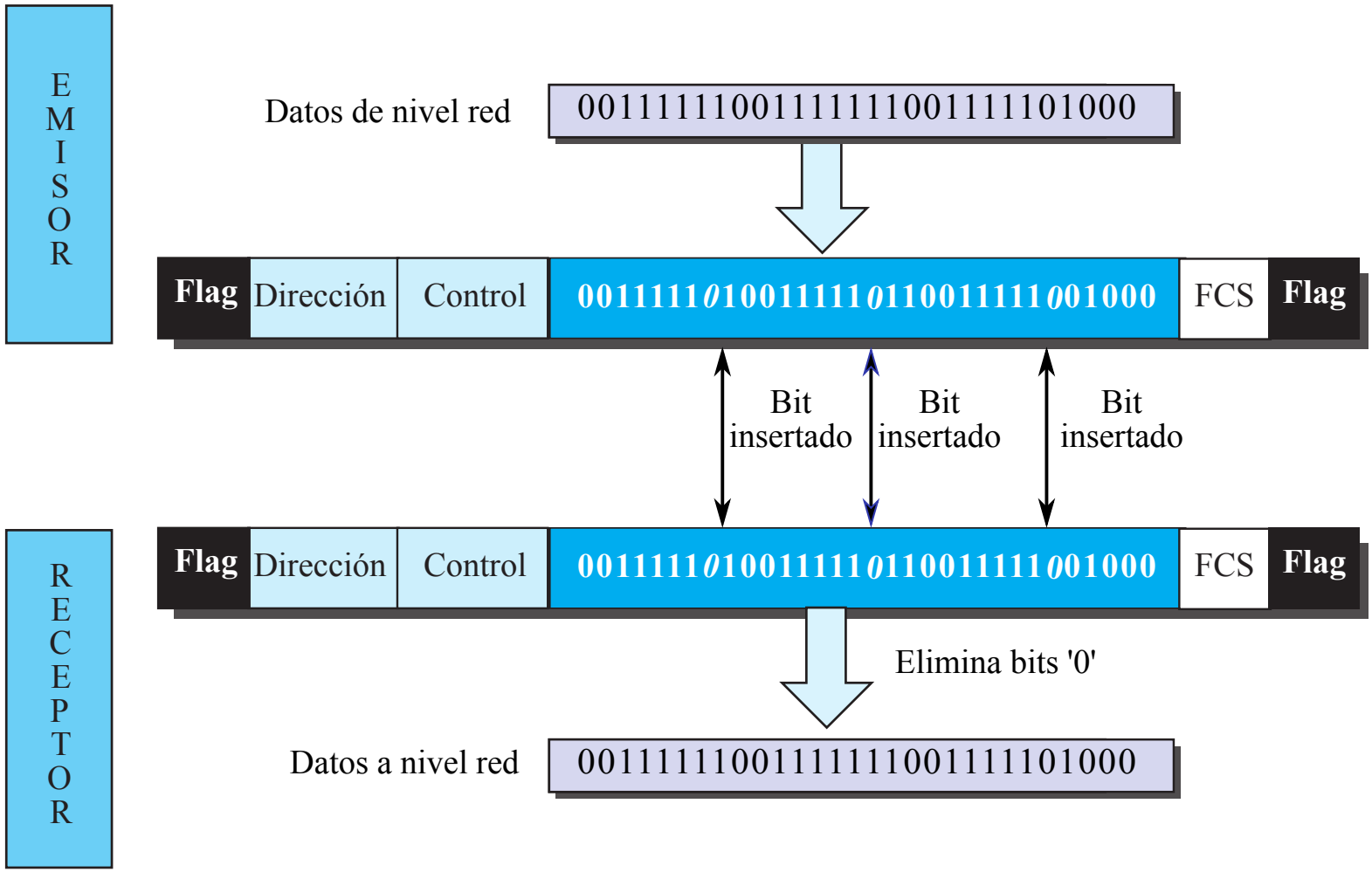
El bit activo en una **respuesta**, es decir, $\mathbf{F} = \mathbf{1}$, indica finalización del turno.

El bit *P/F* activo en una *orden* o *respuesta*, se conoce como P o F, respectivamente.

Campo *Información*

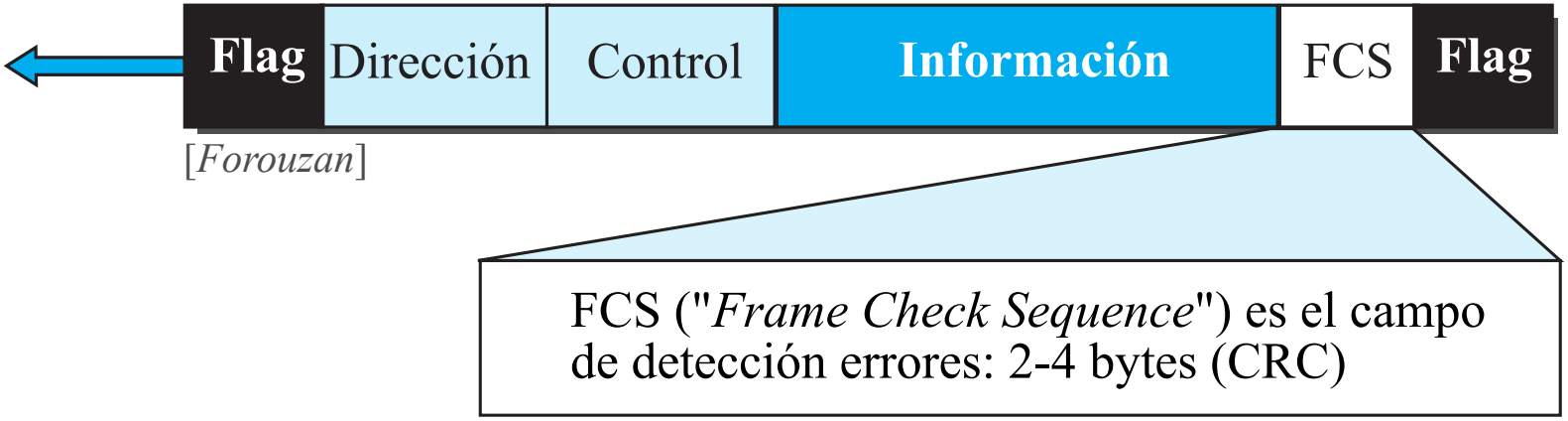


Inserción de bits

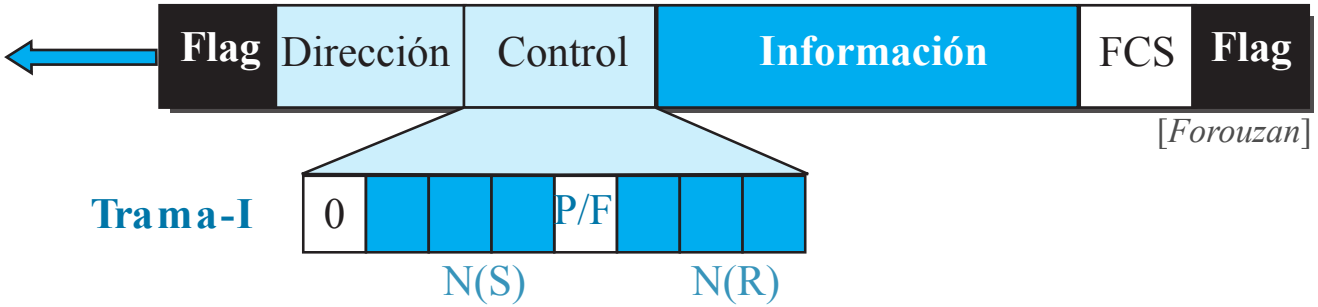


El mecanismo de relleno de bits es para evitar que la secuencia 01111110 no pueda producirse en el campo *información*.

Campo *FCS* (*Frame Check Sequence*)



Tramas de Información



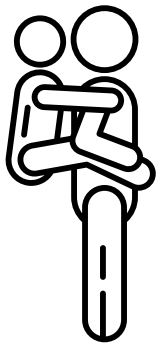
Las tramas-I (que pueden ser tanto **órdenes** como **respuestas**) se utilizan para transmitir información numerada secuencialmente.

Contiene los siguiente campos:

- `N(S)`: nº de secuencia de trama transmitida (*send*)
- `N(R)`: nº de secuencia de trama que espera recibir (*receive*).

El campo `N(R)` confirma al otro extremo la recepción de todas las tramas hasta `N(R)-1`.

En enlaces con actividad bidireccional simultánea, las tramas de información en un sentido contienen confirmaciones de la comunicación en sentido opuesto (**Piggybacking**), lo que permite una mejor utilización del circuito de datos.



V(S) y V(R)

Sea N el ancho de ventana y $m \in \mathbb{N}$, el nº de bits del campo secuencia, entonces:

- La numeración de secuencias es módulo 2^m, y
- se debe de cumplir:

$$N \leq 2^m - 1$$

Por ejemplo, para $m = 2$ se tiene que:

$$N \leq 3$$

y la numeración de secuencias es módulo 2^2, entonces:

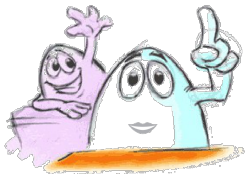
$$0, 1, 2, 3, 0, 1, 2, 3, \cdots$$

Además, cada estación mantiene las variables internas `V(S)` y `V(R)`.

Previo al envío de trama, los valores de `V(S)` y `V(R)` se copian, respectivamente, en los campos `N(S)` y `N(R)`, de las tramas I ó S, según proceda.

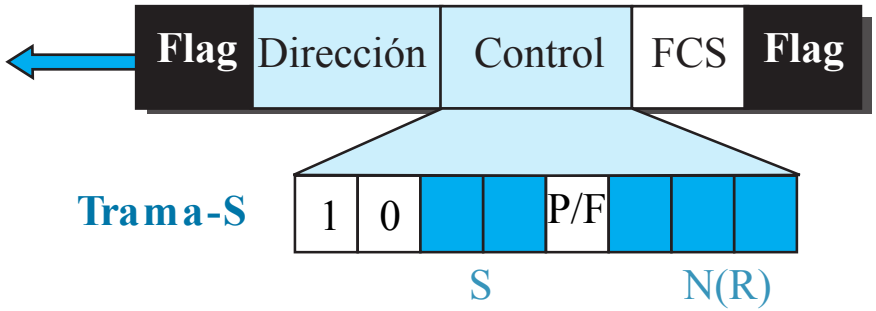
La actualización de las variables se realiza de la siguiente manera:

- Después de trama enviada $\rightarrow V(S) = (V(S) + 1)_{|2^m}$
- Después de trama recibida correctamente $\rightarrow V(R) = (V(R) + 1)_{|2^m}$



La terminología $|p + q|_m$ indica operación $p + q$ en aritmética modular m.

Tramas de supervisión (S) (I)



S	Orden/Respuesta	
00	RR	Receptor preparado
01	REJ	Rechazo
10	RNR	Receptor no preparado
11	SREJ	Rechazo selectivo

Se emplean para **control de flujo**.

Con los dos bits "S" podemos definir hasta cuatro tramas de supervisión.

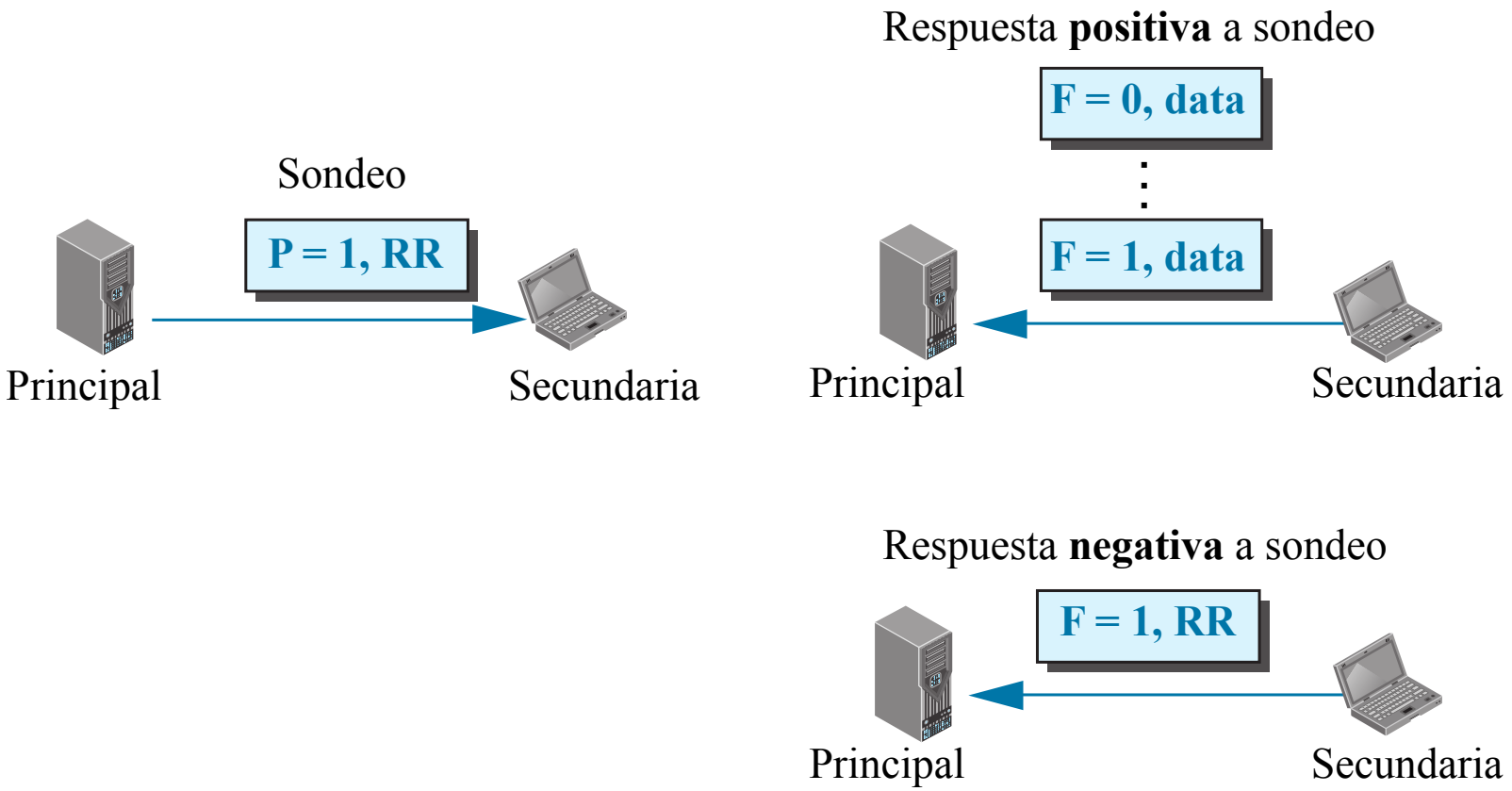
Se pueden utilizar tanto en **orden** como **respuesta**, dependiendo del tipo de estación que la emite.

Se utilizan junto con el bit P/F.

Tramas de supervisión (S) (II)

- RR (*Receive Ready. S=00*). Se utiliza para indicar la disponibilidad de recepción y confirmación de tramas junto con N(R). Una estación primaria puede usar el comando RR para sondear a una secundaria mediante el bit P.
- RNR (*Receive Not Ready. S=01*). Indica una indisponibilidad transitoria de recepción de tramas (control de flujo). También confirma tramas con el campo N(R). Cuando el receptor pueda aceptar tramas de nuevo enviará una trama RR.
- REJ (*Reject. S=10*). Utilizado para confirmar la recepción de tramas anteriores a la N(R) y solicitar la retransmisión de la trama N(R) y posteriores.
- SREJ (*Selective Reject. S=11*). Confirma la recepción de las tramas anteriores a la N(R) y solicita la retransmisión de la N(R), exclusivamente. Una trama SREJ debe ser transmitida por cada trama errónea, pero con la siguiente limitación: solamente puede haber una trama SREJ pendiente.

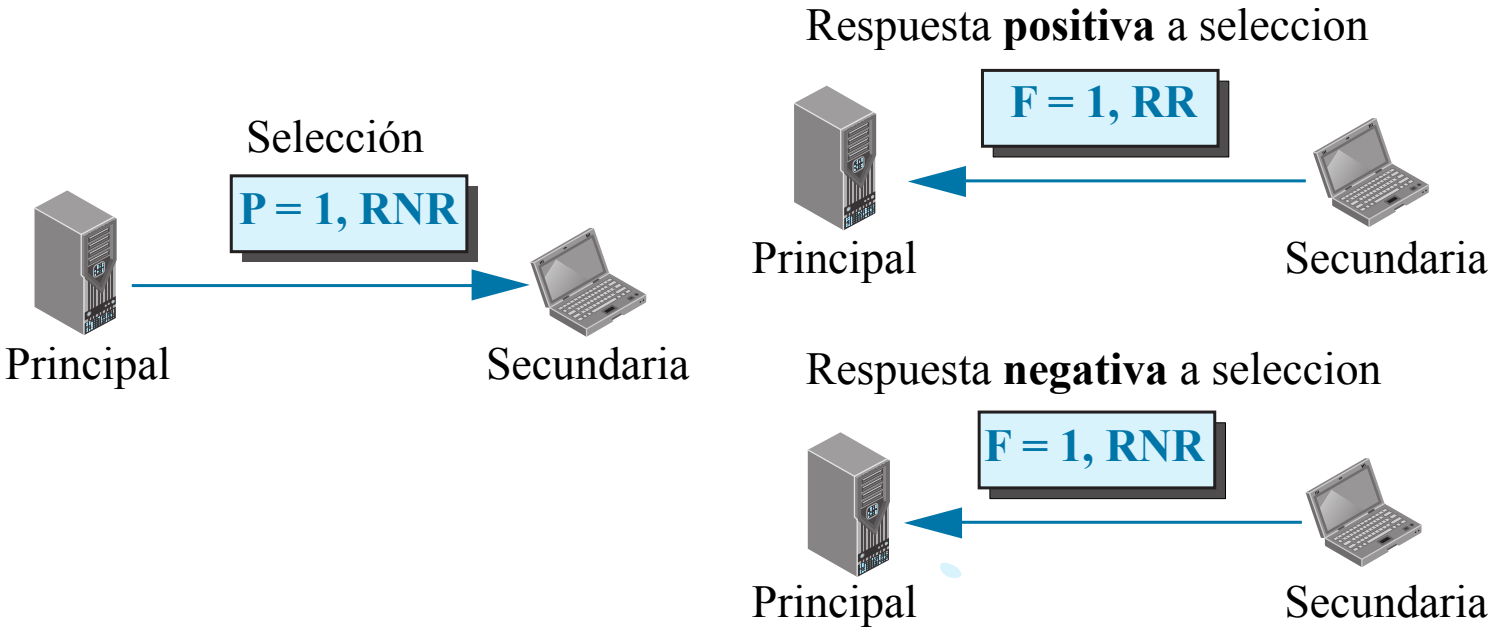
Tramas de supervisión (S): sondeo



La operación de **sondeo**, se realiza mediante `RR,P`:

- **Respuesta positiva a sondeo**: si la secundaria dispone de datos para enviar (tramas `I`), responde con el envío de tramas. Al finalizar el envío o agotar ventana, activa bit `F`.
- **Respuesta negativa a sondeo**: si la secundaria no dispone de datos para enviar, responde `RR,F`.

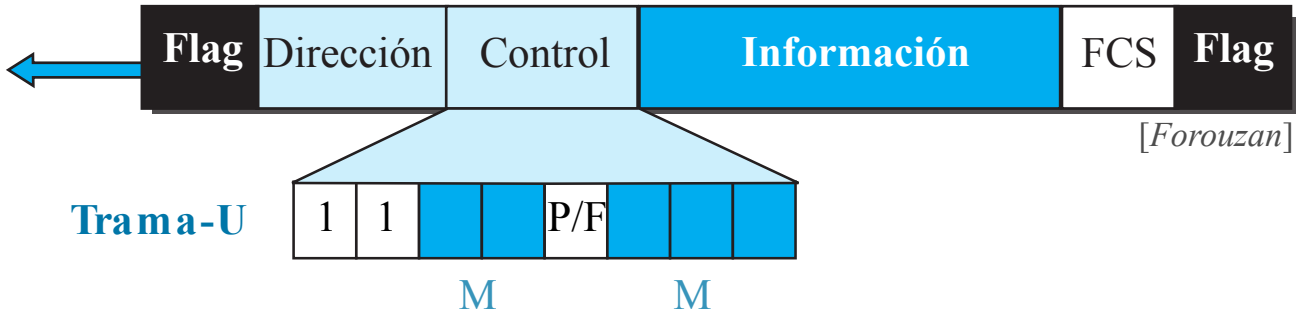
Tramas de supervisión (S): selección



La operación de **selección** se realiza mediante `RNR,P`.

- **Respuesta positiva a selección**: si la secundaria está disponible, responde `RR,F`, indicando que está preparada. A continuación, la principal enviará tramas `I`.
- **Respuesta negativa a selección**: si la secundaria no está disponible, responde `RNR,F`, por lo tanto, la principal no le deberá enviar tramas `I`.

Tramas no numeradas (M) (I)



Utilizadas para funciones de control: inicialización del enlace, selección del modo de transferencia de datos, variables de estado, etc.

También tenemos **tramas de información no numeradas**, esto es, sin número de secuencia que ni confirman ni su envío modifica el número de secuencia de las tramas numeradas.

Con los 5 bits "M"se codifican las diferentes tipos de tramas no numeradas.

M	Orden	Respuesta
00 001	SNRM	
11 011	SNRME	
11 000	SARM	DM
11 010	SARME	
11 100	SABM	
11 110	SABME	
00 000	UI	UI
00 110		UA
00 010	DISC	RD
10 000	SIM	RM
00 100	UP	
11 001	RSET	
11 101	XID	XID
10 001		FRMR

Tramas no numeradas (M) (II)

- `Sxxx` (*Set unextended numbering mode*) (Orden). Establece el modo de transferencia de datos. Se inicializan todas las variables y números de secuencia.
  - `SNRM` (*Set Normal Response Mode*). Inicializa enlace en modo de respuesta normal (NRM).
  - `SARM` (*Set Asynchronous Response Mode*). Inicializa enlace en modo de respuesta asíncrono (ARM).
  - `SABM` (*Set Asynchronous Balanced Mode*). Inicializa enlace en modo de respuesta asíncrono balanceado (ABM).
- `SxxxE` (*Set extended numbering Mode*). (Orden). Establece el modo de transferencia de datos con campo de control extendido. Permitirá tener siete bits para especificar `N(R)` y `N(S)`, por tanto, el tamaño de la ventana puede oscilar entre 1 y 127.

  `SNRME` (*Set Normal Response Mode Extended*).
  `SARME` (Set Asynchronous Response Mode Extended).
  `SABME` (Set Asynchronous Balanced Mode Extended).
- `DISC` (*Disconnect*). (Orden). Para abandonar el modo de operación en curso. Las estaciones entran en un modo de desconexión predeterminado por el sistema.
- `RSET` (*Reset*). (Orden). Reset de las variables `V(S)` y `V(R)`, utilizadas respectivamente para generar `N(S)` y `N(R)`. Las tramas previas no reconocidas permanecerán sin reconocerse.
- `UP` (*Unnumbered Poll*).(Orden). Solicita la transmisión de respuesta a una o más estaciones. Normalmente se establece la condición especial de que cada estación sólo responda una vez. No contiene campo de información ni asiente tramas anteriores.
- `UI` (*Unnumbered Information*). (Orden/Respuesta). Se utiliza para enviar información sin numerar secuencialmente y que no va a ser confirmada.
- `XID` (*eXchange Identification*). (Orden/Respuesta). Se utiliza para identificarse ante la otra estación, enviar información de algún parámetro específico, o especificar datos importantes.

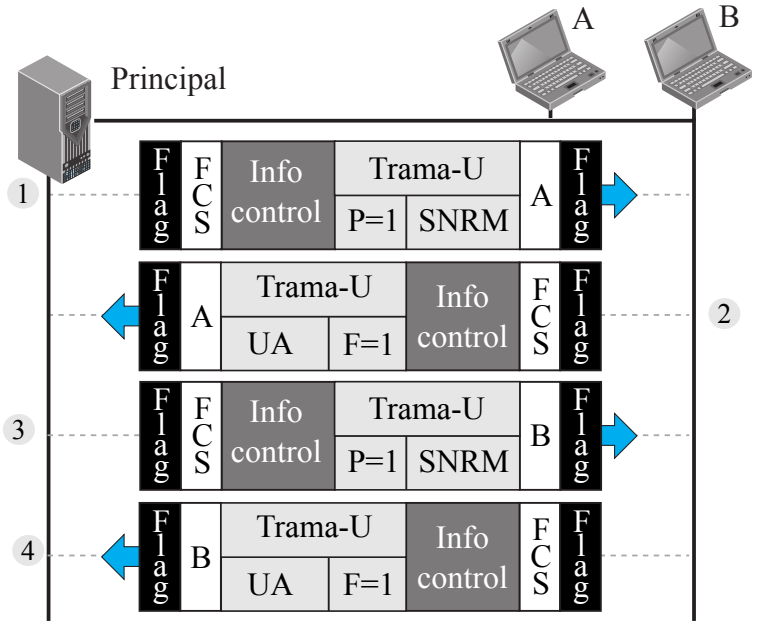
Tramas no numeradas (M) (III)

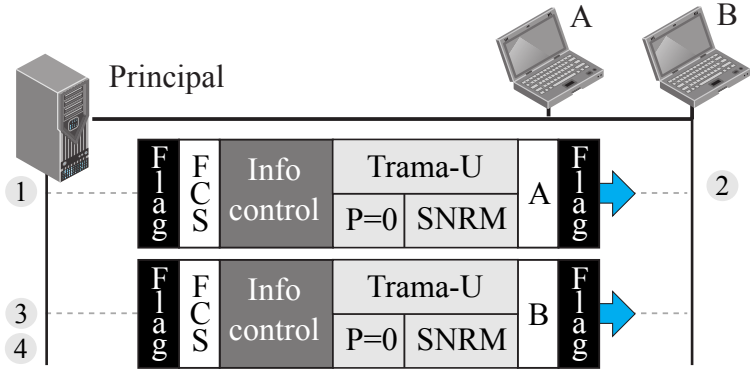
- SIM (*Set Initialization Mode*). (Orden). Orden de entrar en el modo de inicialización.
- RIM (*Request Initialization Mode*). (Respuesta). Solicita el envío de la orden de entrar en el modo de inicialización (SIM). Lo utiliza una secundaria para que la primaria envíe la orden SIM.
- UA (*Unnumbered Acknowledge*). (Respuesta). La respuesta de asentimiento no numerado se utiliza para confirmar la recepción y ejecución de una orden de elección de todos los modos, inicialización, desconexión o reposición (Sxxx, SxxxE, DISC, RSET, SIM).
- DM (*Disconnect Mode*). (Respuesta). Se utiliza para responder a todas las órdenes mientras la estación esté en modo desconectado. Como respuesta a una orden de elección de modo indica la imposibilidad de entrar en el modo solicitado. También se puede utilizar para solicitar la orden de elección de modo.
- TEST (Orden/Respuesta). Utilizada para solicitar respuestas de prueba a la otra estación y comprobar si las cosas funcionan bien. HDLC no establece cómo se deben usar las tramas TEST. Por ejemplo una implementación puede utilizar el campo de información para diagnosticar problemas.
- FRMR (*Frame Reject*). (Respuesta). Indica que una trama que hemos recibido tiene un error no recuperable con la retransmisión de la trama. La respuesta *FRMR* contiene un campo de información que contiene: el campo de control rechazado, el estado (variables V(R) y V(S)) de la estación que origina la respuesta FRMR y cuatro bits (WXYZ) que puestos a uno indican las anomalías detectadas:
  - W: el campo de control recibido (y devuelto) es inválido o no implementado.
  - X: la trama recibida y rechazada contenía un campo de información que no debía llevar.
  - Y: la trama recibida y rechazada tiene una longitud que excede la capacidad del receptor.
  - Z: el N(R) recibido asiente tramas que no han sido transmitidas.

  Si la causa del rechazo es diferente de esas, los cuatro bits se ponen a cero.

Escenarios HDLC

Ejemplo: *Conexión* modo NRM





Conexión con confirmación

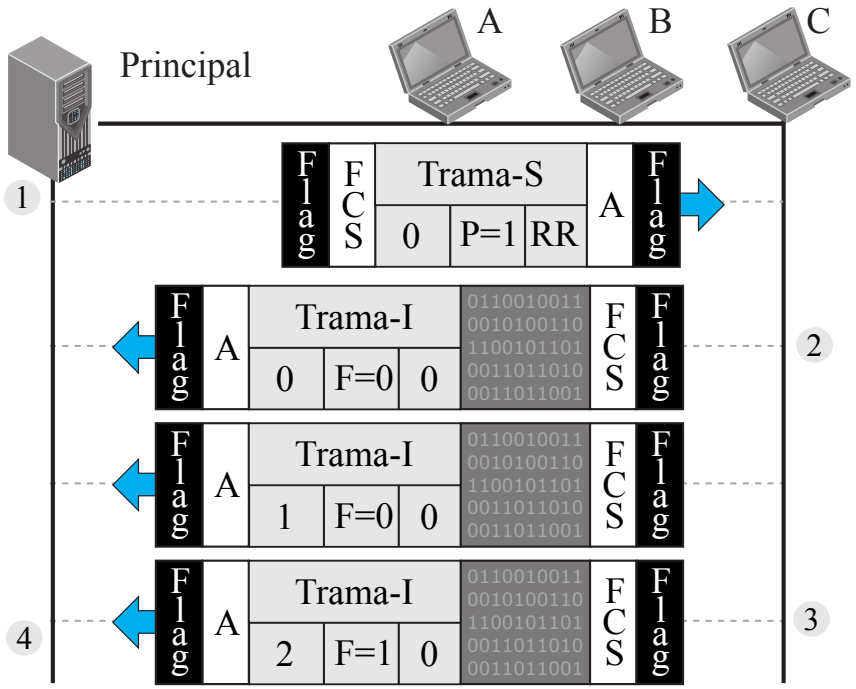
1. Orden de conexión con confirmación a modo *NRM* (*Normal Response Mode*), mediante `SNRM`. Al incluir el bit `P`, es un servicio con confirmación, obligando a la secundaria a emitir una respuesta.
2. Confirmación no numerada mediante `UA,F`.
3. Idem para estación *B*.
4. El enlace NRM entre las estaciones *A* y *B* y la estación *principal* está establecido. Además, es una conexión *confirmada*.

Conexión sin confirmación

1. Orden de conexión sin confirmación a modo de operación *NRM*.
2. Al no haber recibido el bit `P` por parte de la principal, la estación secundaria no responde.
3. Idem para estación *B*.
4. El enlace NRM entre las estaciones *A* y *B* y la estación *principal* está establecido (aunque no confirmado).

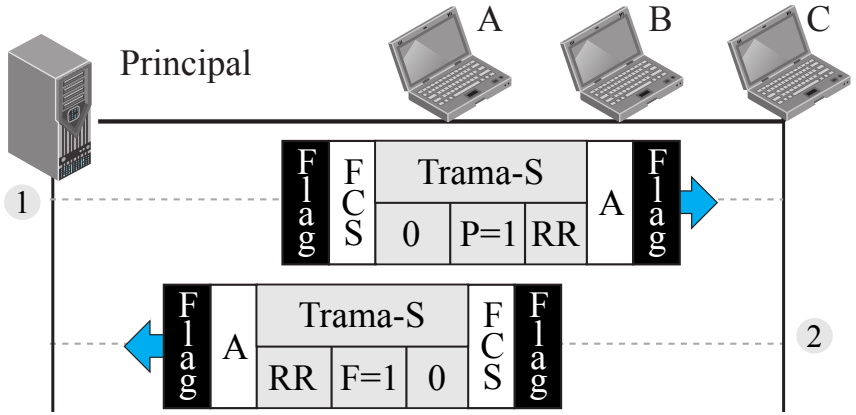
El campo *dirección* siempre contiene la dirección de la *secundaria*: si es una *orden*, la dirección de la destinataria, y si es una *respuesta*, la dirección de procedencia de la trama.

Ejemplo: *Sondeo*



Respuesta positiva a sondeo:

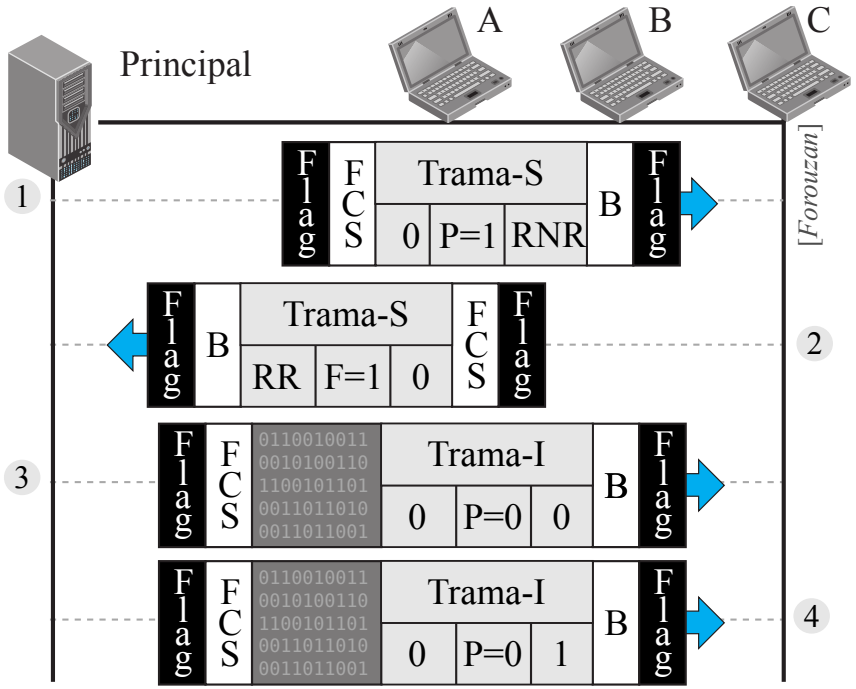
1. Orden de *sondeo* mediante la *trama-S* denominada `RR`, además incluye el bit `P`.
2. La estación sondeada (*A*) dispone *tramas-I* para enviar, procede a su envío, considerando las limitaciones de ancho de ventana y los campos *N(S)* y *N(R)*.
3. Finaliza su turno de participación activando el bit `F` en la última trama.
4. La estación principal sabe que *A* ha finalizado su turno, ya que ha recibido una trama con el bit `F`.

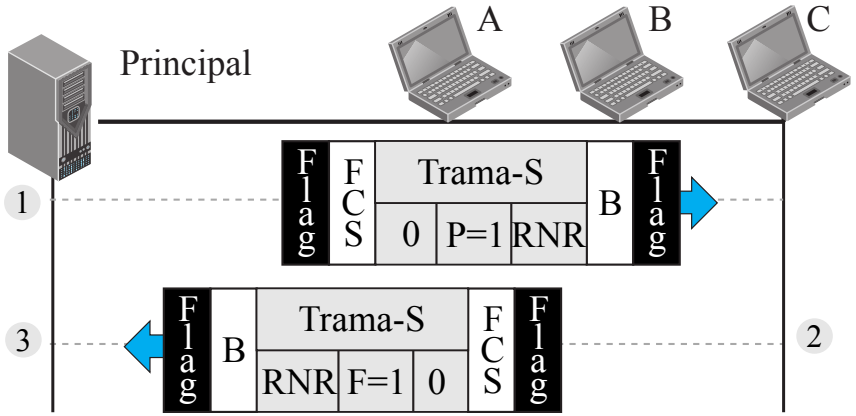


Respuesta negativa a sondeo:

1. Orden de *sondeo* mediante la *trama-S* denominada `RR`, además incluye el bit `P`.
2. La estación sondeada (*A*) no dispone *tramas-I* para enviar. Responde con `RR,F`, que informa que no tiene tramas I, y el campo *N(R)* se utiliza para confirmación de tramas pendientes.

Ejemplo: *Selección*





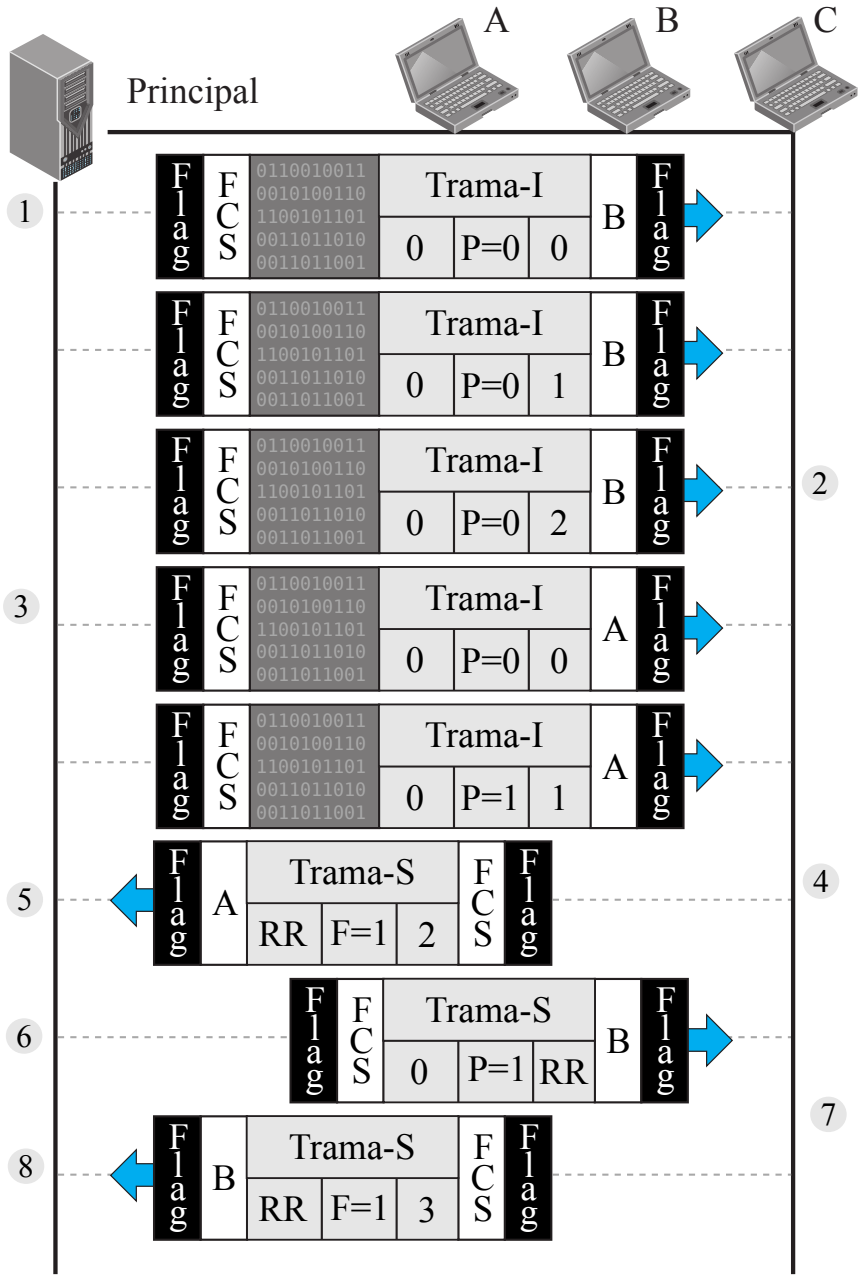
Respuesta positiva a selección

1. Orden de *selección* mediante la *trama-S* denominada `RNR`, además incluye el bit `P`.
2. La estación secundaria nunca responderá con tramas `I`. En este caso, realiza una respuesta positiva a selección (`RR,F`), indicando que está en disposición de recibir tramas `I`.
3. La estación *principal* envía tramas `I`.
4. Estación *B* no ha recibido bit `P`, por lo tanto, no responde.

Respuesta negativa a selección:

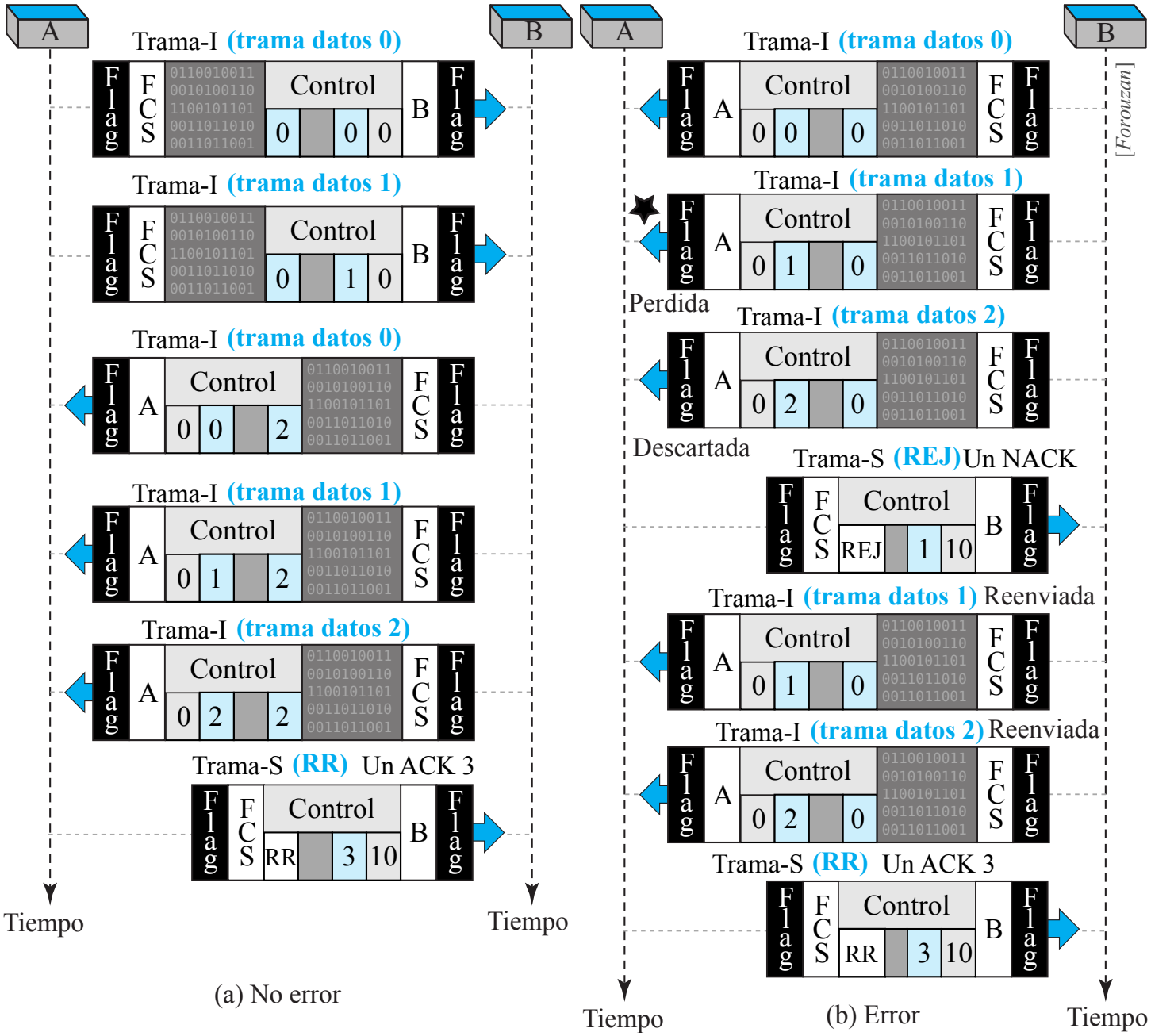
1. Orden de *selección* mediante la *trama-S* denominada `RNR`, además incluye el bit `P`.
2. La estación secundaria nunca responderá con tramas `I`. En este caso, realiza una respuesta negativa a selección (`RNR,F`), indicando que NO está en disposición de recibir tramas `I`.
3. La estación *principal* no envía tramas `I`.

Ejemplo: uso bit *P/F*

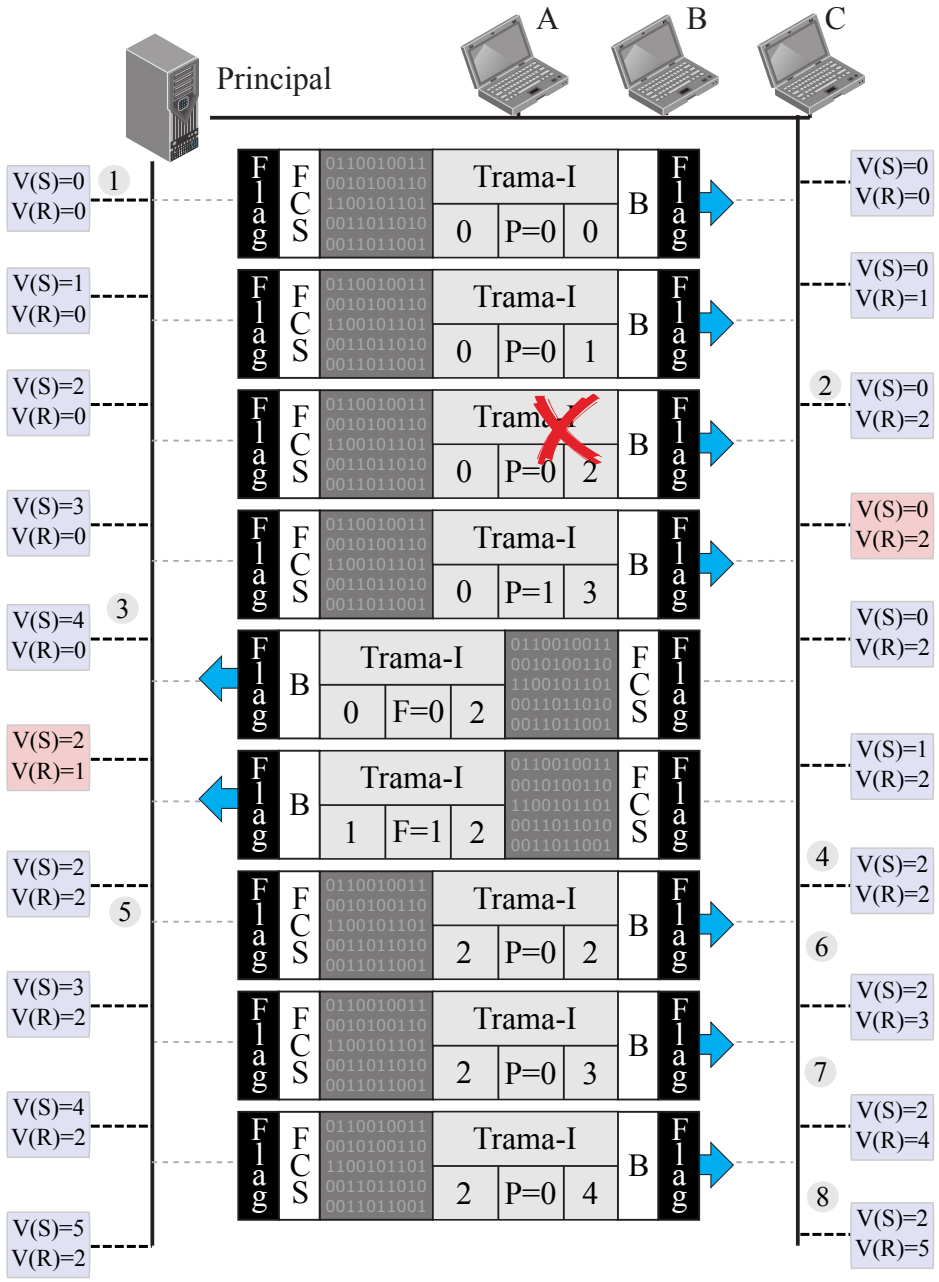


1. Envío de tramas $P \rightarrow B$.
2. Estación *B* no responde porque no ha recibido el bit `P`.
3. Envío de tramas $P \rightarrow A$.
4. Bit `P` en estación *A*, por lo tanto responde, al no tener tramas `I`, responde con `RR,F`.
5. $N(R) = 2$, por lo tanto, tramas 0 y 1 enviadas a *A*, confirmadas.
6. Envío *sondeo* a estación *B*.
7. Idem. que (4).
8. $N(R) = 3$, por lo tanto, tramas 0, 1 y 2 enviadas a *B*, confirmadas.

Ejemplo: piggybacking con/sin error (ABM)



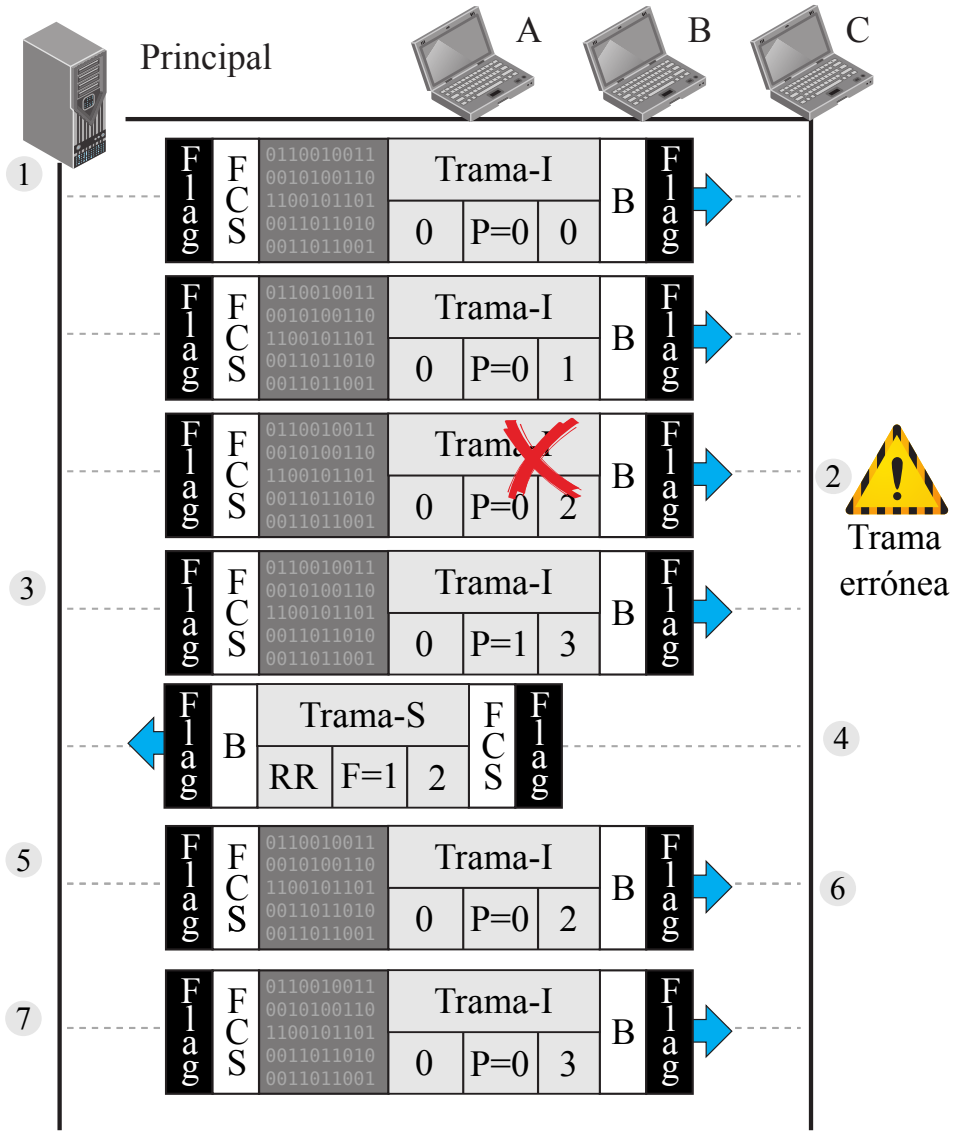
Ejemplo: *retroceso-N* (I)



1. Envío $P \rightarrow B$
2. Trama 2 detectada errónea (o no llega).
3. Enviadas tramas 0, 1, 2 y 3. Enviado bit P para sondeo.
4. Estación B tiene tramas 0 y 1 para enviar. $N(R) = 2$, indicando que 0 y 1 correctas, y que espera recibir la 2. Finaliza con F.
5. Envío de tramas 2, 3 y 4, con campo $N(R) = 2$, indicando que tramas 0 y 1 recibidas correctamente.
6. Trama 2 retransmitida y recibida correctamente ya que la anterior había sido errónea.
7. Trama 3 recibida duplicada.
8. Trama 4 recibida. Estación *B* no responde porque no ha recibido ninguna P.

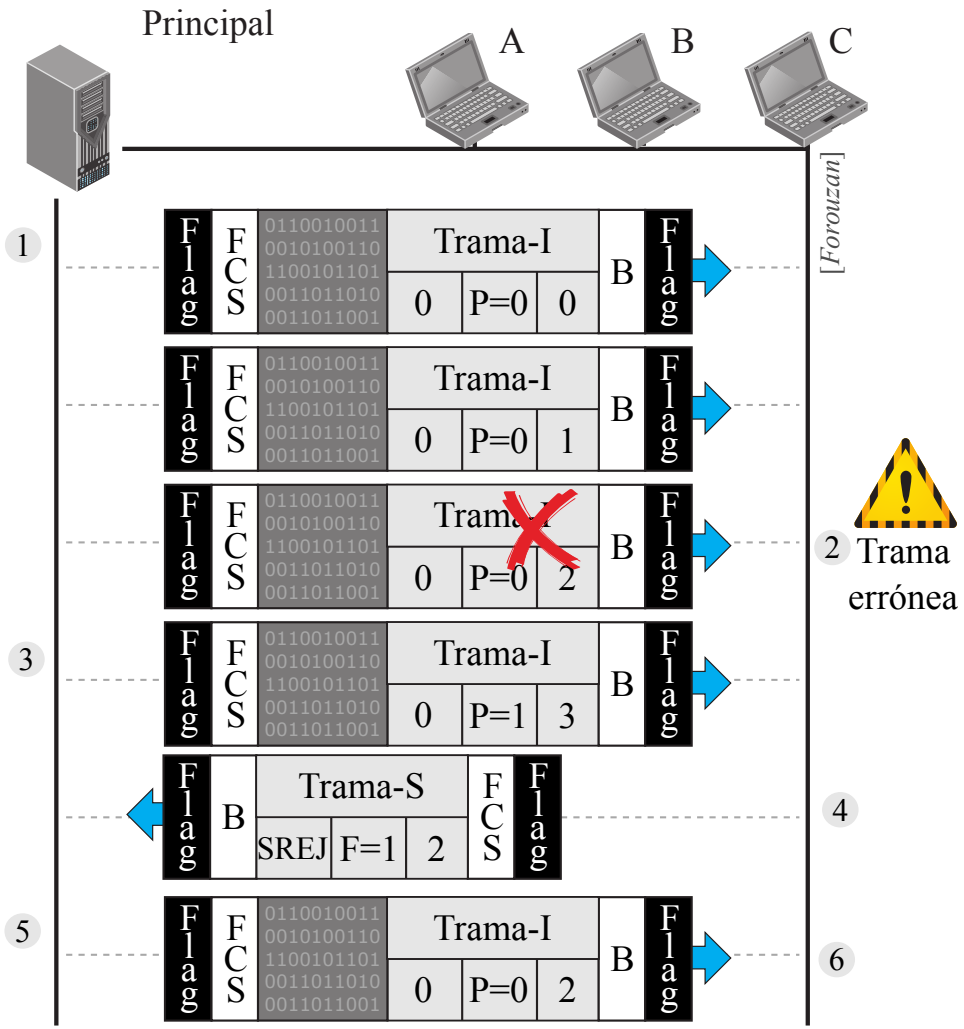
Cada estación mantiene las pareja de varibles $V(S)$ y $V(R)$ para cada enlace lógico, que se actualiza a medida que se envían y reciben tramas de datos y confirmaciones.

Ejemplo: *retroceso-N* (II)



1. Envío $P \rightarrow B$
2. Trama 2 detectada errónea (o no llega).
3. Enviadas tramas 0, 1, 2 y 3. Enviado bit P para sondeo.
4. Estación *B* no tiene tramas I, por lo tanto, responde con RR2,F indicando que 0 y 1 correctas, y próxima trama la 2.
5. Envío de tramas 2, 3 y 4.
6. Trama 2 retransmitida y recibida correctamente ya que la anterior había sido errónea.
7. Trama 3 recibida. Estación *B* no responde porque no ha recibido ninguna P.

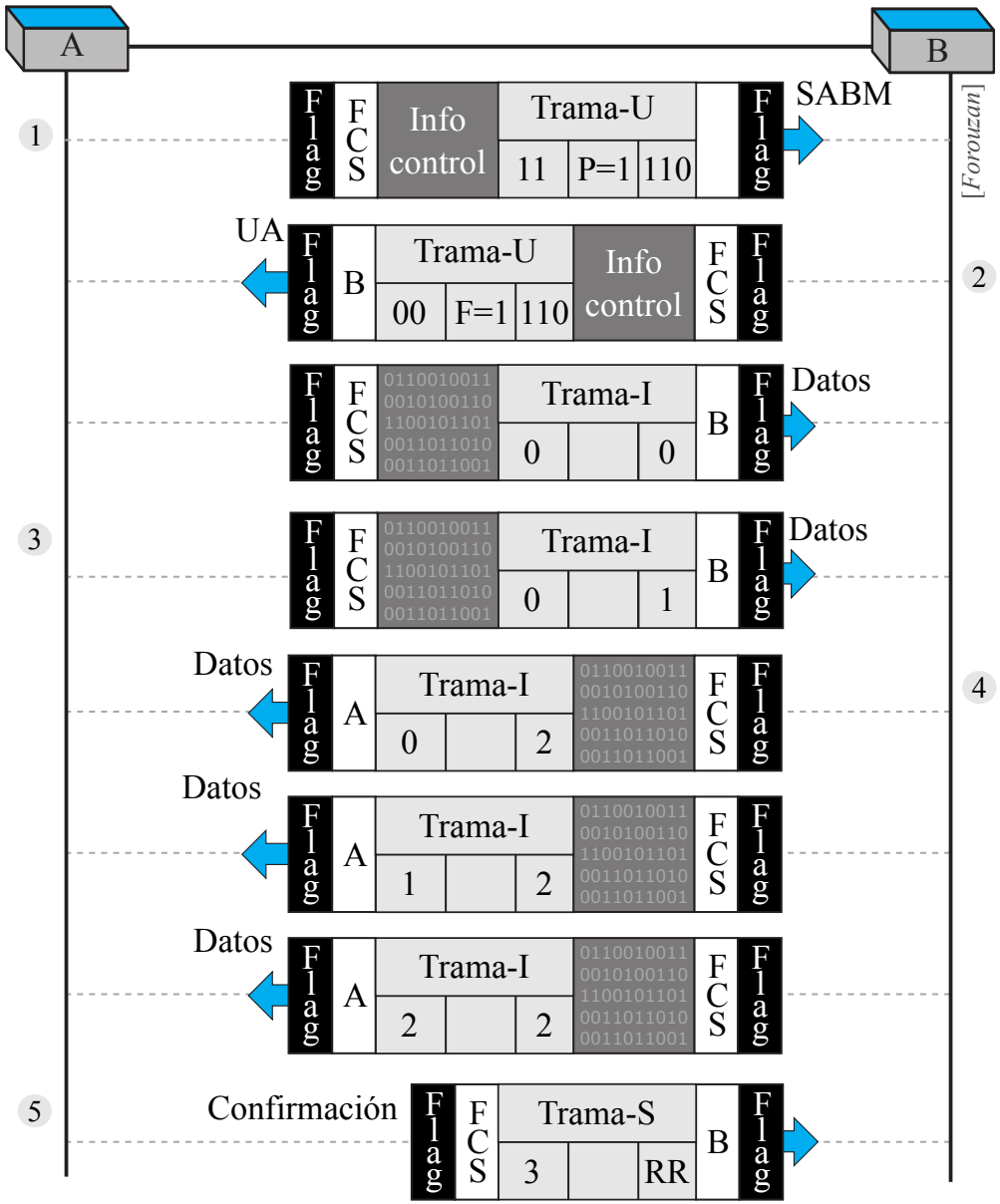
Ejemplo: *rechazo selectivo*



1. Envío $P \rightarrow B$
2. Trama 2 detectada errónea (o no llega).
3. Enviadas tramas 0, 1, 2 y 3. Enviado bit P para sondeo.
4. Utiliza SREJ,F rechazar selectivamente la trama *2*.
5. Envío sólo la trama 2.
6. Estación *B* no responde porque no ha recibido ninguna P.

En rechazo selectivo, sólamente se retransmite la trama errónea.

Ejemplo: dispositivos *peer*



1. Orden de conexión `SABM,P`.
2. Respuesta `UA,F`.
3. Envío tramas 0 y 1. Campo $N(R) = 0$.
4. Envío tramas 0, 1 y 2. Campo $N(R) = 2$, confirmando la 0 y 1.
5. Trama `RR` con $N(R) = 3$, confirmando la 0, 1 y 2.
6. ...

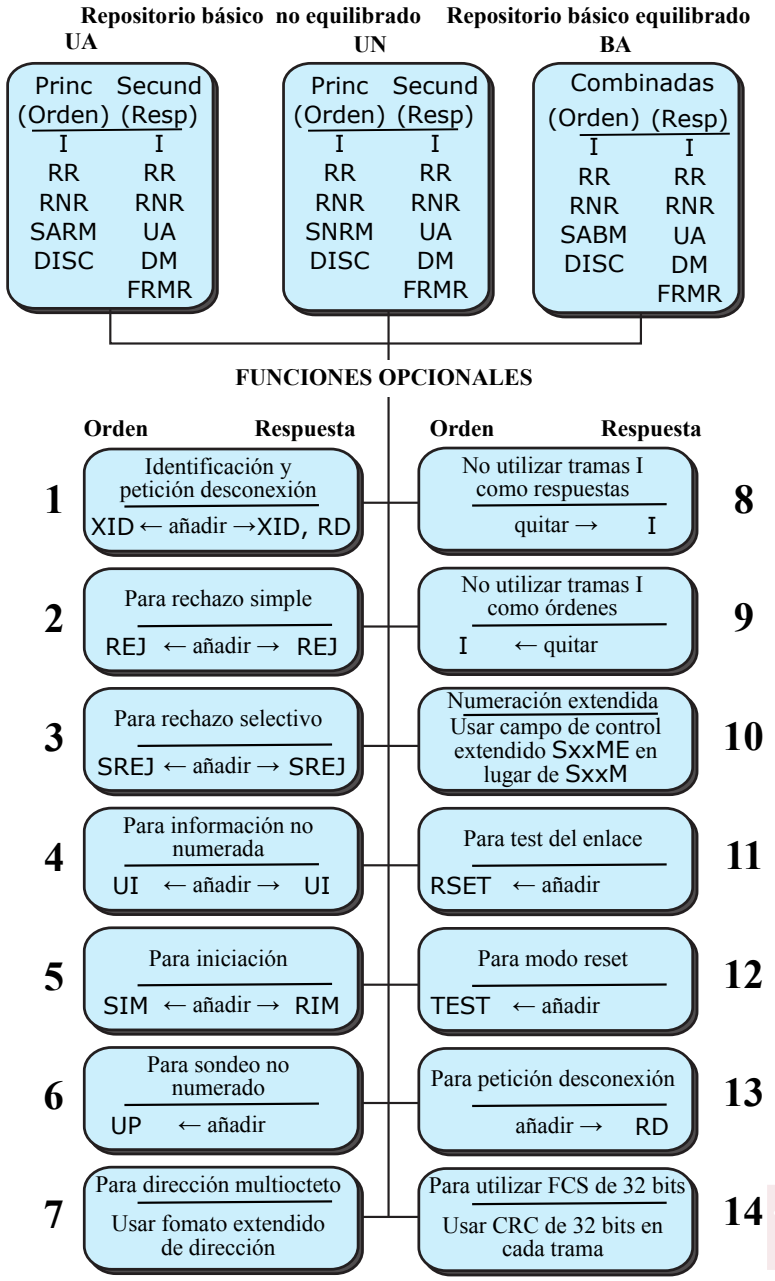
En el modo asíncrono balanceado (ABM), las estaciones participan de forma independiente, sin previo aviso de ninguna otra estación. Son dispositivos iguales.

Clases de procedimientos

Las **clases de procedimientos** consisten en la definición de un conjunto básico de órdenes y respuestas, permitidos en cada uno de los modos de transferencia de datos, que puede ser **ampliado** con la inclusión de ciertas **funciones opcionales**.

Los tres modos de operación (`NRM`, `ARM` y `ABM`) definen tres clases de procedimientos:

- `UN` (*UNC-Unbalanced Normal Class*)
- `UA` (*UAC-Unbalanced Asynchronous Class*)
- `BA` (*BAC-Balanced Asynchronous Class*)

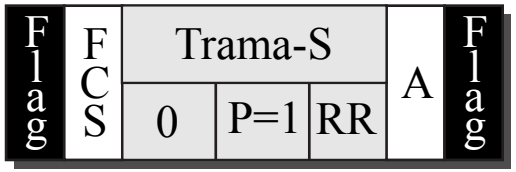


Ejemplos de procedimientos

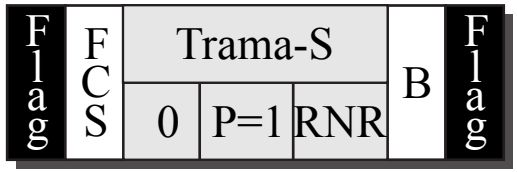
HDLC ha sido el punto de partida para difernetes protocolos de comunicaciones estándard:

- **UN 3,4**
  Modo de respuesta normal (`NRM`), rechazo selectivo (opción 3) y la posibilidad de enviar tramas de información no numerada (opción 4).
  Escenarios típicos son circuitos multipunto con varias estaciones secundarias y actividad bidireccional alternada (semiduplex).
- **BA 2,8**
  Modo de respuesta asíncrona balanceada (`ABM`), rechazo simple (opción 2) y solamente se pueden utilizar tramas I como órdenes (opción 8).
  El que no se pueda transmitir información en tramas de respuesta no plantea ningún problema puesto que las estaciones son combinadas y pueden transmitir tanto órdenes como respuestas al otro extremo.
  El escenario típico es un enlace punto a punto entre dos estaciones combinadas.
- **LAPB (*Link Access Procedure Balanced*)**
  Protocolo de capa enlace en redes X.25. LAPB está está definido `BA 2,8` ó `BA 2,8,10` en función de si se utiliza o no el campo de control extendido (opción 10).
- **LLC (*Logical Link Control*)**
  IEEE 802.2. Estándar para redes de área local clasificado como `BA 2,4`.
- **LAPD (*Link Access Procedure, D channel*)**
  Una extensión de LAPB.

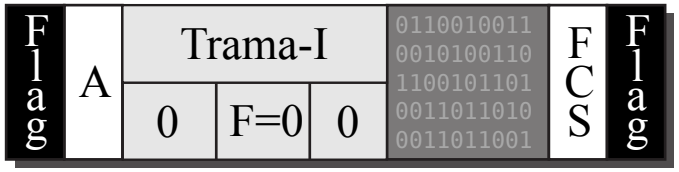
Terminología



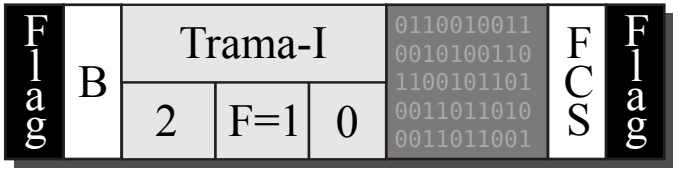
A,RR0,P



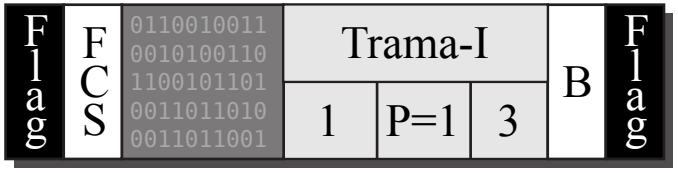
B,RNR0,P



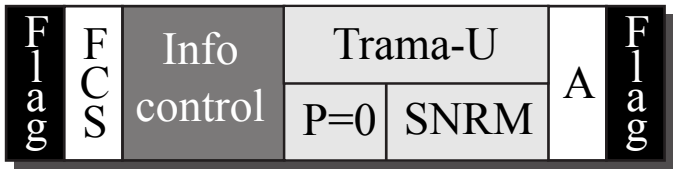
A,I00



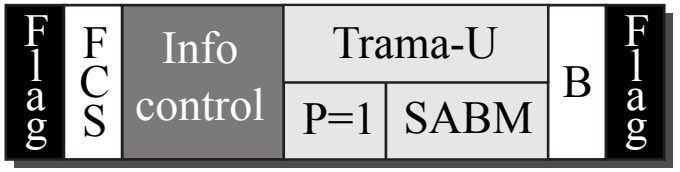
B,I20,F



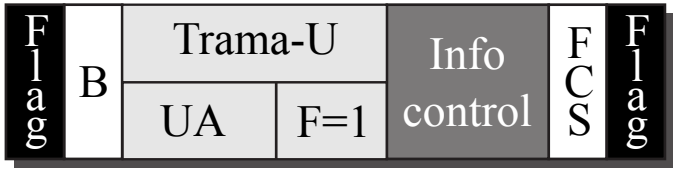
B,I31,P



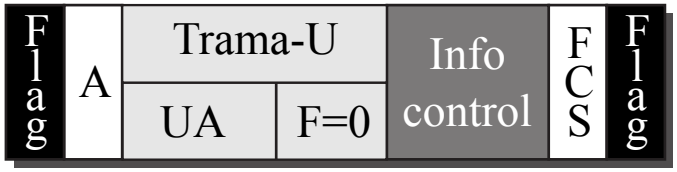
A,SNRM



B,SABM,P

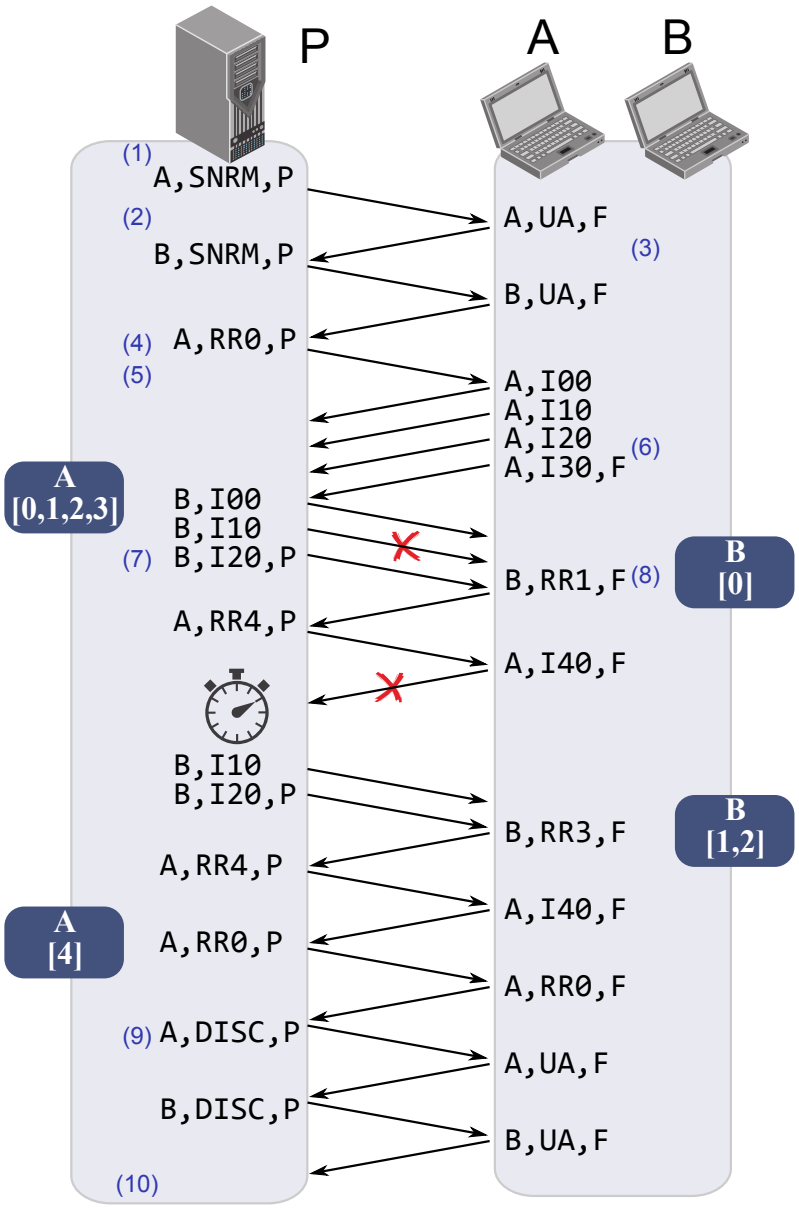


B,UA,F



A,UA

Ejemplo



1. Enlace desconectado.
2. Orden de conexión en `NRM` con confirmación.
3. Confirmación no numerada.
4. Enlace conectado.
5. Estación `P` no tiene datos para enviar, por lo que envía sondeo a `A`.
6. `A` recibe sondeo y envía tramas disponibles.
7. Envío a B y sondeo.
8. Respuesta negativa a sondeo.
9. Todo enviado y confirmado. Paso a desconexión.
10. Enlace desconectado.

Cuestiones de revisión

El estudiante debería ser capaz de responder a las siguientes cuestiones:

¿Qué diferencia hay entre direccionamiento físico y lógico?

¿Qué es un bit de paridad?

¿Qué es CRC?

¿Por qué esperas que un CRC detecte más errores que un bit de paridad?

Enumera tres formas diferentes con los que el algoritmo CRC se puede describir.

¿Qué representa el código de bloque $c(n, k)$?

¿Cómo se resuelve la transparencia de datos en protocolos orientados a byte?

¿Cómo se resuelve la transparencia de datos en protocolos orientados a bit?

Enumera y define, brevemente, algunos de los requisitos para la comunicación efectiva sobre un enlace de datos.

Define el control de flujo.

Define el control de flujo parada y espera.

¿Cuáles son las razones para segmentar una transmisión de datos larga en un cierto número de tramas?

Describe el control de flujo basado en ventana deslizante.

¿Cuál es la ventaja del control de flujo ventana deslizante comparado con el control de flujo parada y espera?

¿Por qué ARQ rechazo selectivo mejora el rendimiento sobre ARQ Retroceso-N?

UNIVERSITAS
Miguel Hernández

UNIVERSITAS
Miguel Hernández