# 3. LA CAPA DE ENLACE DE DATOS

# Contenido

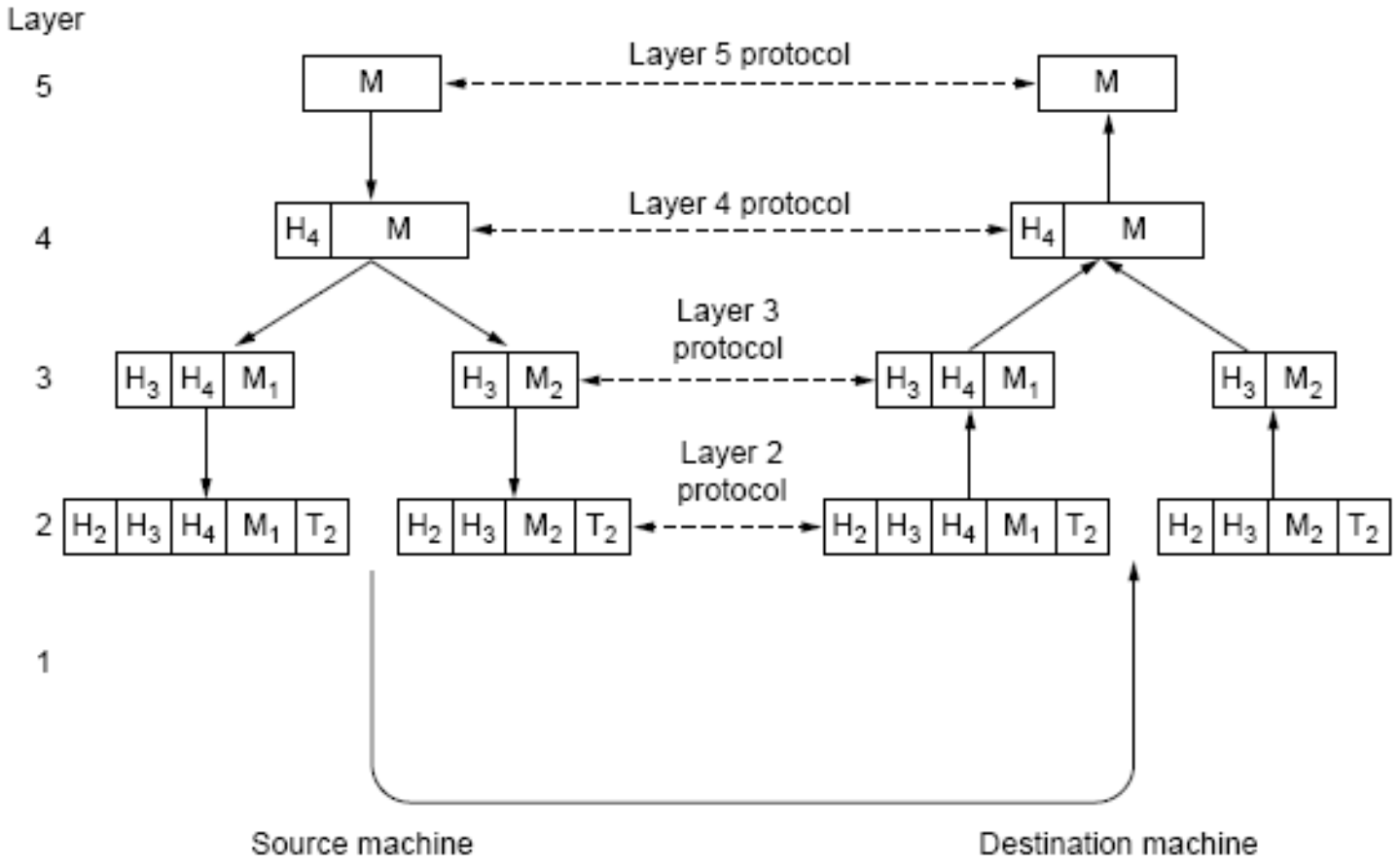
- 3.1 Cuestiones de diseño de la capa de enlace de datos
- 3.2 Detección y corrección de errores
- 3.3 Protocolos elementales de enlace de datos
- 3.4 Protocolos de ventana deslizante

- Limitaciones de los medios de comunicación:
  - Tienen una limitada tasa de datos
  - Retardan de propagación de los bits
  - Les afecta el ruido interno y externo
  - Se producen **errores** en la transmisión de los datos

# 3.1 Cuestiones de diseño de la capa de enlace de datos

- Las funciones de la capa 2 son dos:
  1. Maneja los errores de transmisión
  2. Regula el flujo de datos: emisores rápidos y receptores lentos

- Para realizar sus funciones, la capa 2 encapsula los paquetes de la capa 3 en tramas (frame)
- Trama:
  - Encabezado (head)
  - Carga útil: paquete de la capa 3 (packet)
  - Cola (tail)
- Los controles de flujo y errores los realizan las capas 2 y 4


Layer
5
4
3
2
1
M
Layer 5 protocol
M
H4 M
Layer 4 protocol
H4 M
Layer 3
protocol
H3 H4 M1
H3 M2
H3 H4 M1
H3 M2
Layer 2
protocol
H2 H3 H4 M1 T2
H2 H3 M2 T2
H2 H3 H4 M1 T2
H2 H3 M2 T2
Source machine
Destination machine

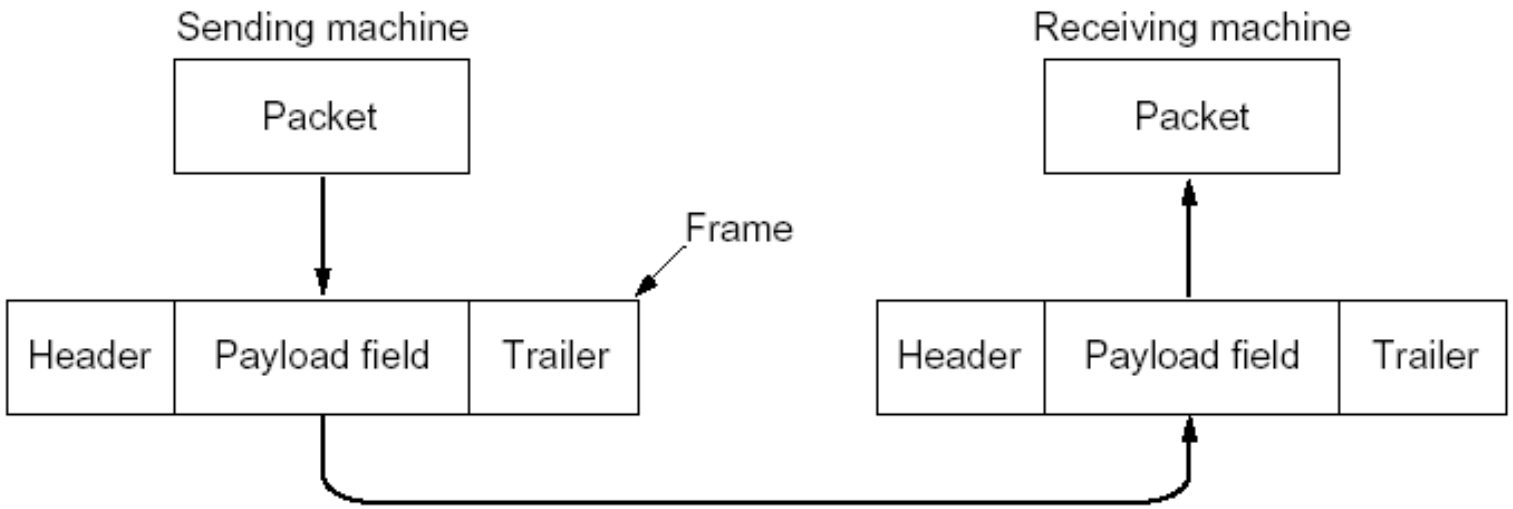



Fig. 3-1. Relationship between packets and frames.

# 3.1.1 Servicios a la capa de red

- Servicio principal: transferir datos de la capa 3 en el origen a la capa 3 en el destino

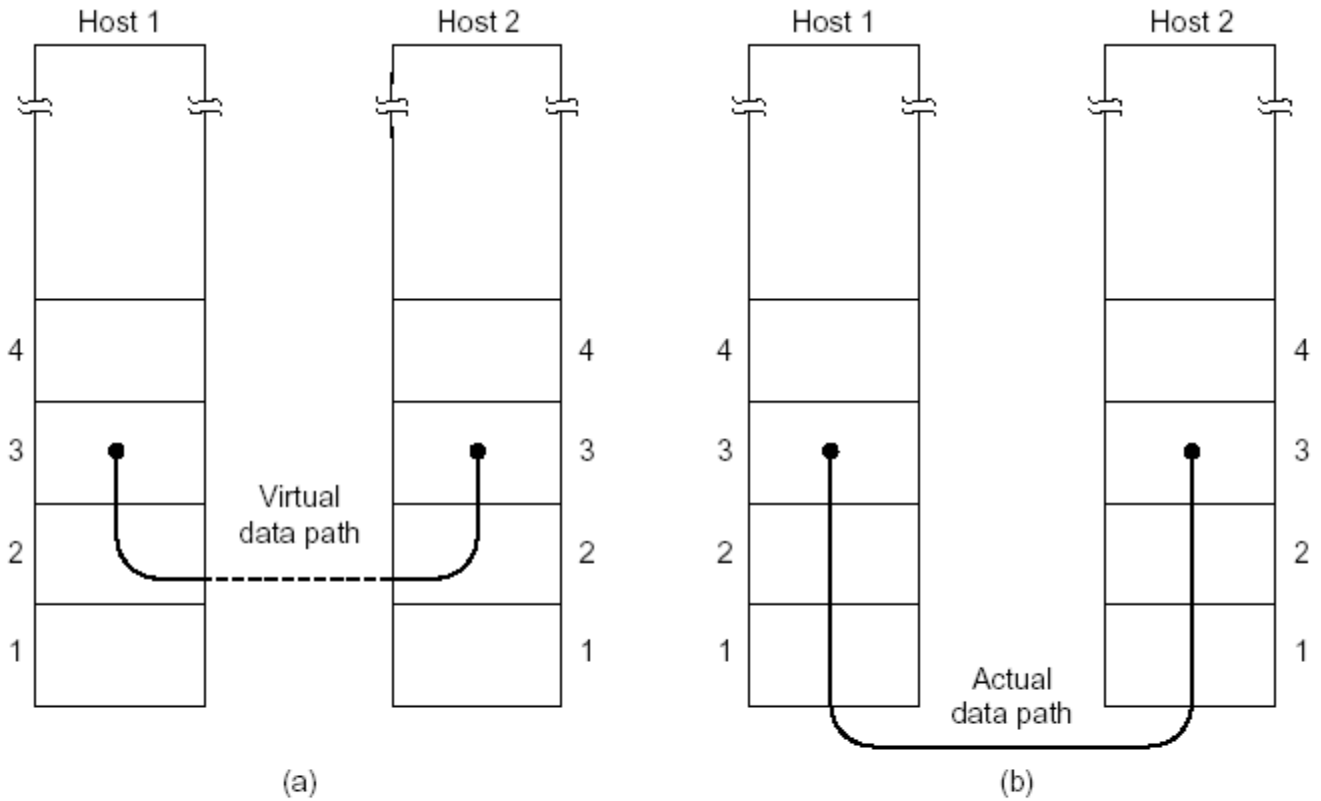



Fig. 3-2. (a) Virtual communication. (b) Actual communication.

# Formas de dar servicio de la capa de enlace

- La capa 2 tiene tres maneras de dar el servicio a la capa 3
  - Sin conexión sin ACK.
  - Sin conexión con ACK.
  - Orientado a la conexión con ACK.

# Servicio sin conexión **sin** acuso de recibo acknowledment

- El origen envía tramas independientes al destino sin pedir confirmación ACK
- No hay control de daño o pérdida de tramas debido al ruido en el medio
- En el destino, los datos se aceptan con o sin errores
- Servicio apropiado con medios de transmisión con baja tasa de errores
- Adecuado en:
  - redes LAN como Ethernet
  - aplicaciones de tiempo real: VoIP, videoconferencia

# Servicio sin conexión **con** ACK

- El receptor avisa al emisor de qué manera llegó la trama
- Si la trama no ha llegado o llega dañada, el emisor lo reenvía
- ¿Cómo sabe el emisor que la trama no ha llegado a su destino?
- Usado en canales inestables: inalámbricos
- Con FO este servicio es innecesario, ¿por qué?
- Porque los datos llega y sin errores
- Utilizado en mensajería de texto de telefonía móvil se envía un mensaje y se recibe su confirmación

# Servicio orientado a la conexión con ACK

- Es un servicio confiable
- Se simula una conexión numerando las tramas
- La numeración permite garantizar:
  - Recibir todas las tramas
  - En forma ordenada
  - Una sola vez
  - Sin errores
- Se simula una conexión porque las tramas llegan en orden y una sola vez
- El algoritmo podría implementarse en hardware

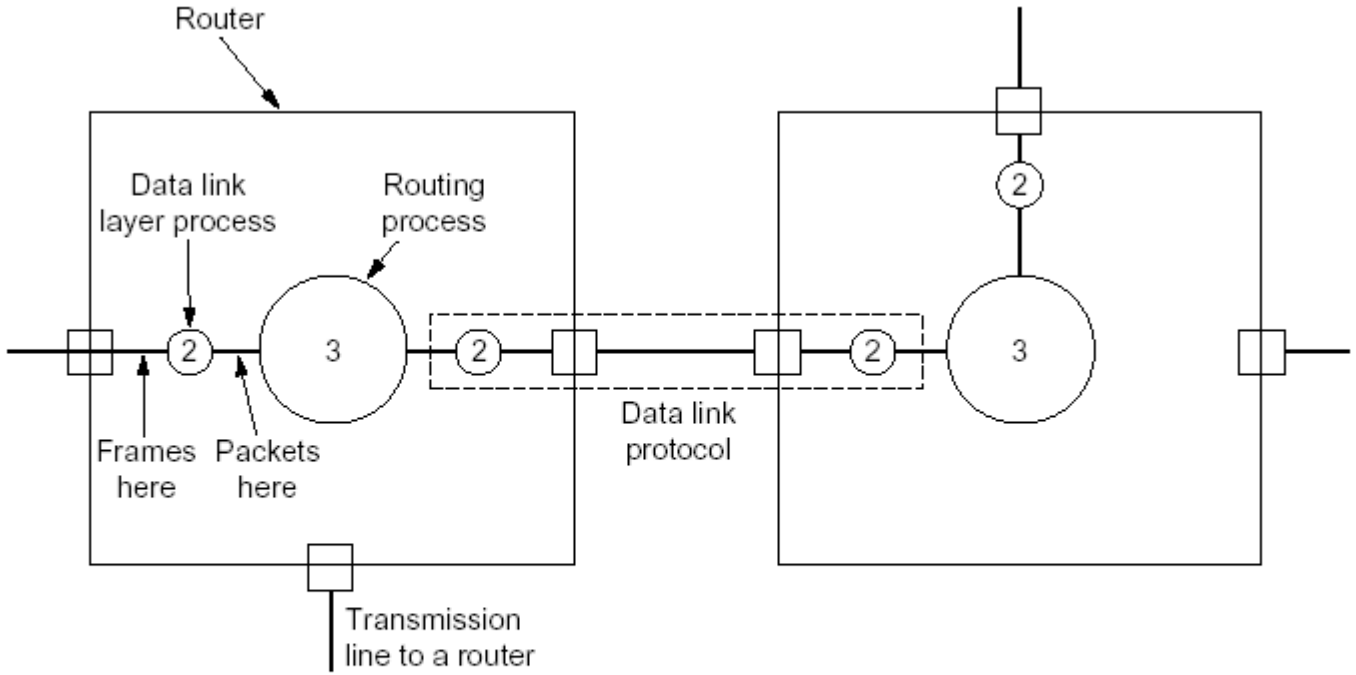



Fig. 3-3. Placement of the data link protocol.

# 3.1.2 Entramado

- Métodos para indicar el inicio y fin de una trama:
  - Inserción de intervalos de tiempo entre las tramas
  - Conteo de caracteres
  - Banderas de inicio y fin, con relleno de caracteres
  - Banderas de inicio y fin, con relleno de bits
  - Violación de codificación de la capa física

# Inserción de intervalos de tiempo

- Pausas entre tramas
  - Similar a los espacios de tiempo entre palabras
- Riesgo:
  - Los intervalos podrían ser eliminados
  - Por error podrían insertarse intervalos dentro de una trama

# Conteo de caracteres

- Longitud de la trama: campo en la cabecera para indicar el número de caracteres de la trama

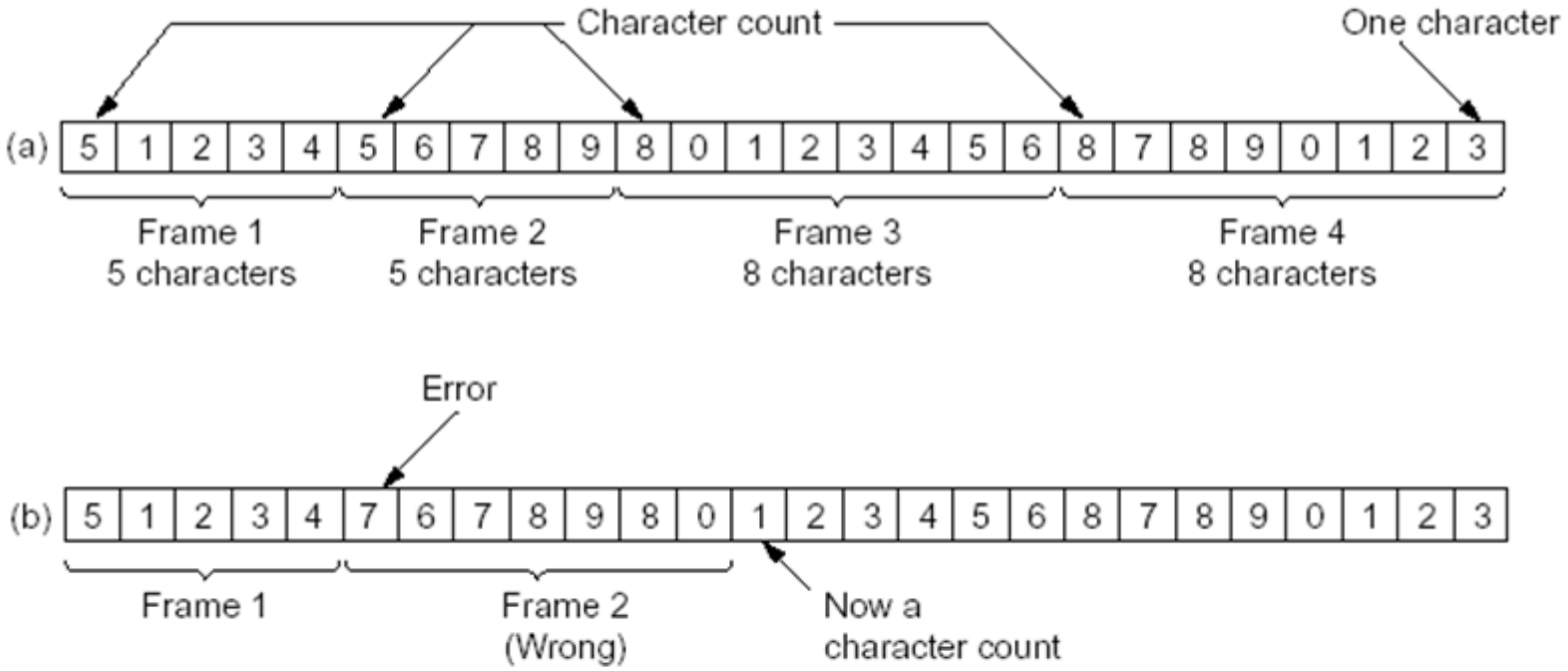
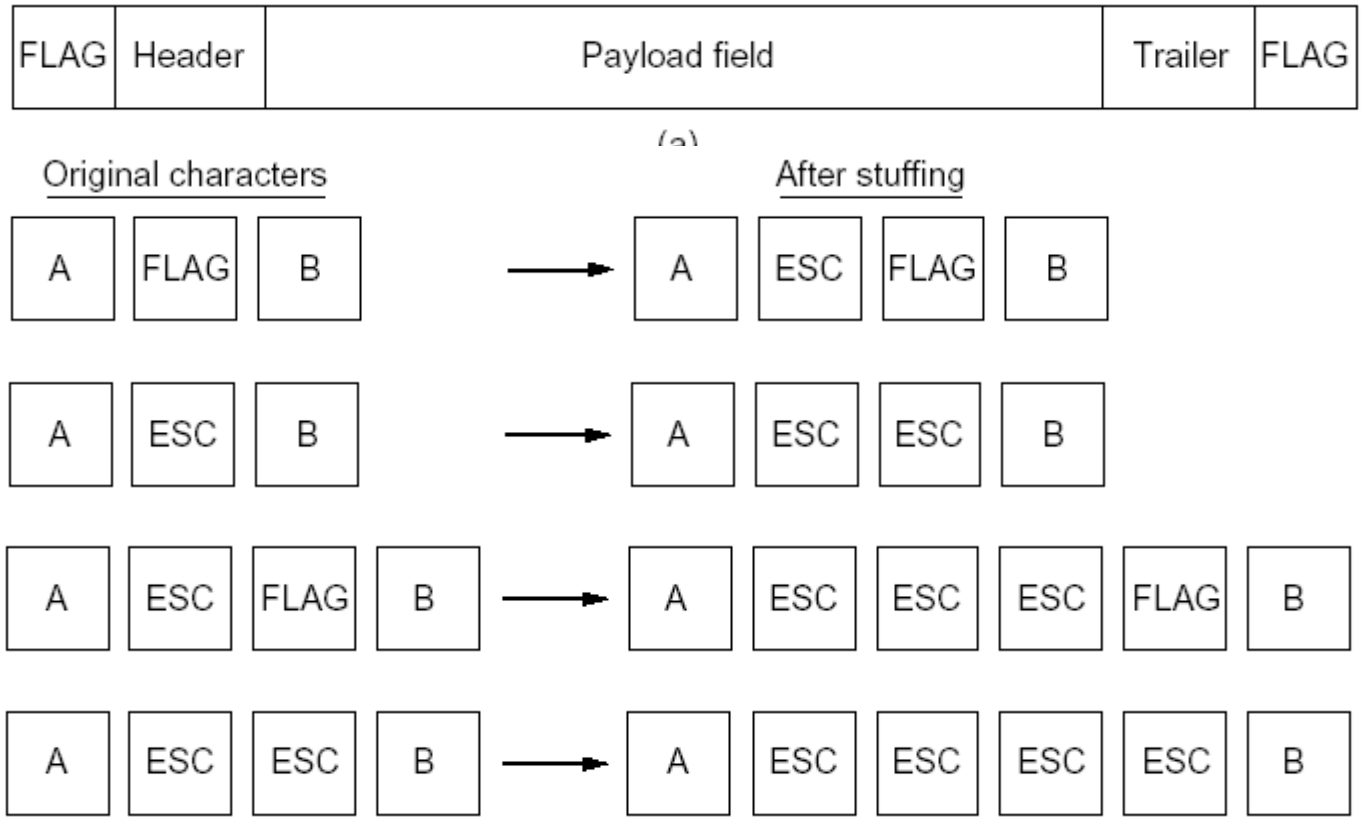



Fig. 3-4. A character stream. (a) Without errors. (b) With one error.

# Banderas con relleno de caracteres

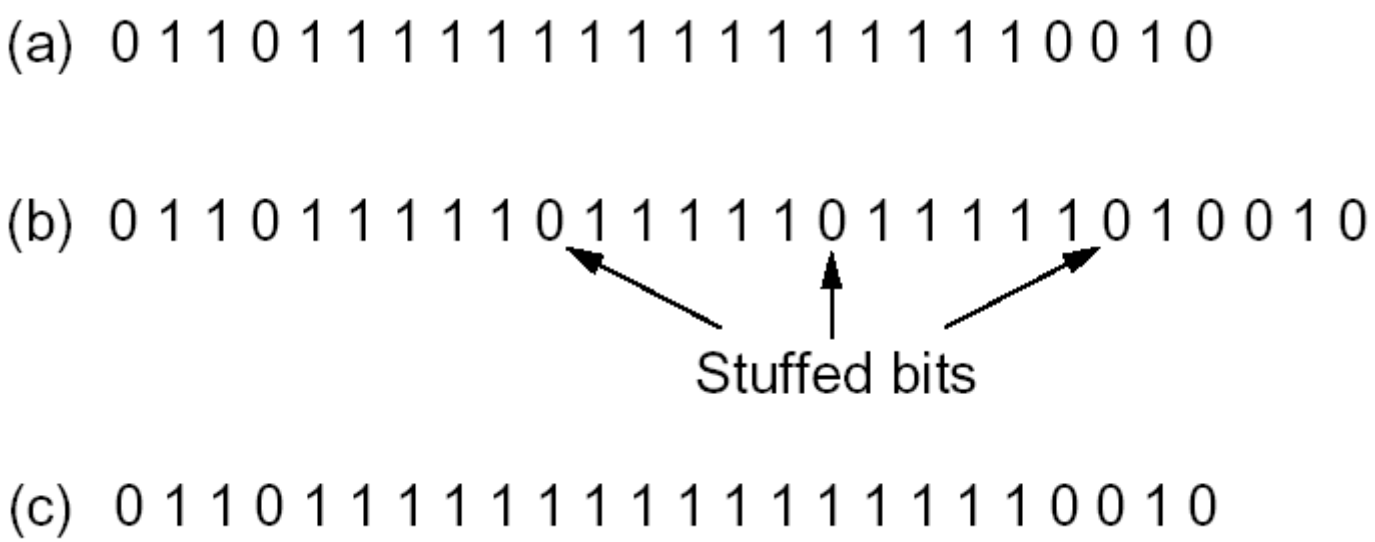
- Inicio y fin de una trama con un byte especial
- Carácter de relleno: 01111110
- La bandera sincroniza al receptor con el emisor
- Si el receptor pierde sincronía, puede esperar la siguiente bandera
- Dos banderas consecutivas señalan el fin de una trama e inicio de la siguiente
- Podría ocurrir que el patrón de bits de la bandera aparezca entre los datos
- Para solucionar se usa el byte ESC


FLAG
Header
Payload field
Trailer
FLAG
(a)
Original characters
After stuffing
A
FLAG
B
A
ESC
FLAG
B
A
ESC
B
A
ESC
ESC
B
A
ESC
FLAG
B
A
ESC
ESC
ESC
FLAG
B
A
ESC
ESC
B
A
ESC
ESC
ESC
ESC
B

# Banderas con relleno de bits

- Cada trama inicia y termina con 01111110 (seis 1s)
- Si la capa 2 encuentra cinco 1s consecutivos en los datos, inserta un bit 0
- El relleno es el bit 0
- Si el receptor ve cinco 1s de entrada seguido de un 0, extrae el 0 de relleno

# Bits de relleno en los datos



# Violaciones de codificación de la capa física

- Algunas LANs representan los bits de datos usando codificación Manchester
  - Alto – bajo: bit 1
  - Bajo – alto: bit 0
- alto-alto y bajo-bajo no se usa para datos
- Algunos protocolos usan alto-alto y bajo-bajo para delimitar tramas

# 3.1.3 Formas de controlar los errores

- Primero es necesario implementar un sistema de detección los errores
- Luego se retroalimenta al emisor con tramas de control ACK – NACK
- Son tramas de confirmación positiva o negativa
- ACK *Acknowledgement*
- ACK: envíe la siguiente trama
- NACK: Retransmita la trama dañada
- Si la trama de datos no llega al receptor, no hay retroalimentación:
  - Inicio de temporizador en el emisor
  - Expira el temporizador – retransmite
  - Tramas numeradas para no duplicar si el receptor si recibió la trama pero ACK se perdió

# 3.1.4 Control de flujo

- Si existen emisores rápidos y/o receptores lentos
- Hay saturación, sobrecarga o desbordamiento del receptor
- Para evitarlo se controla el flujo de tramas del emisor hacia el receptor:
- Capa de Enlace de Datos. Basado en retroalimentación al emisor

# 3.2 Detección y corrección de errores

- Fundamentalmente los errores se producen en:
  - Última milla, si es de cobre (*local loop,* bucle local, bucle de abonado, o planta externa del proveedor del servicio)
  - Comunicación inalámbrica
- Debido a ráfagas de ruido, los errores aparecen en ráfagas de bits, no individualmente
- Desventaja de errores en ráfaga: más difíciles de detectar y corregir
- Ventaja: dañan una o máximo dos tramas

# 3.2.1 Códigos de corrección de errores

- Hay dos estrategias:
- Códigos de corrección de errores
  - Incluir información redundante en la trama para que el receptor corrija el error
- Códigos de detección de errores
  - Incluir suficiente información para que el receptor sepa que ha ocurrido un error

- Fibra óptica: tasa baja de error. El medio es inmune al ruido
  - Es suficiente usar códigos de detección de errores
- Wireless: tasa alta de errores debido a ruido ambiental, descargas eléctricas atmosféricas, interferencia con otras señales
  - Es mejor usar códigos de corrección de errores

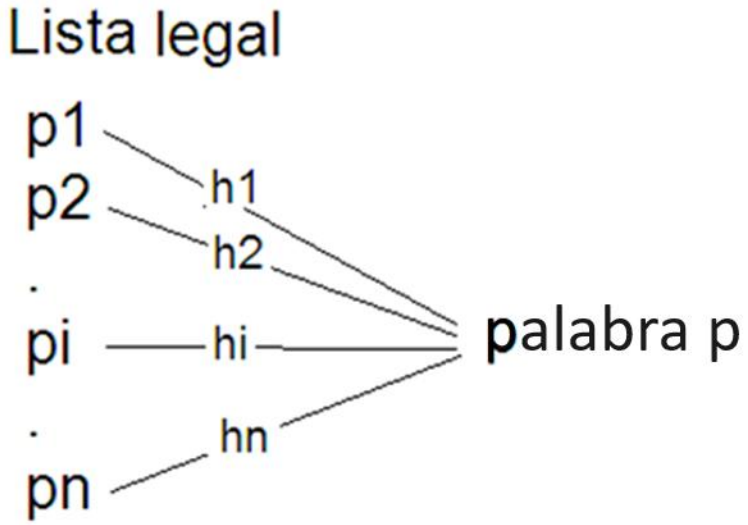
- Dos palabras codificadas deben estar suficientemente separadas
- En dos palabras codificadas, el número de bits en los que difieren es h

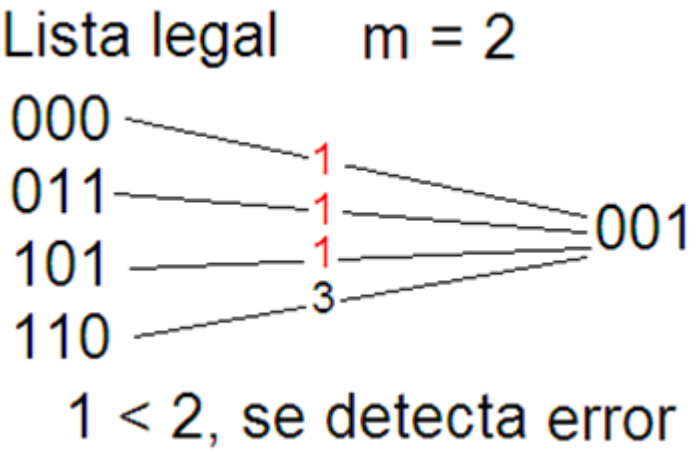
```
        10001001
        10110001
XOR     00111000
```

- En el ejemplo, h = 3
- h es la distancia de Hamming

- Se puede construir una lista completa de palabras válidas
- Las palabras deben ser lo suficientemente distintas entre sí
- Para ello, h entre dos palabras no será menor a una distancia mínima **m**
- La facilidad para detectar y corregir errores depende de h
- Si una palabra tiene un h < **m** con alguna de las palabras legales, se detectará el error
- Si h < m, esta cantidad h de bits no es posible convertir una palabra válida en otra palabra válida

La distancia de Hamming entre palabras legales es de m = 2



# Corrección de errores. 1era. forma

- Para corregir e bits errados o menos se necesita un código de distancia m = 2e + 1
- Las palabras legales están tan separadas que alterando hasta e bits todavía esta más cerca de la palabra original

# Ejemplo de corrección de errores

- En vez de: 00 Se tiene: 0000000000
  01 0000011111
  10 1111100000
  11 1111111111

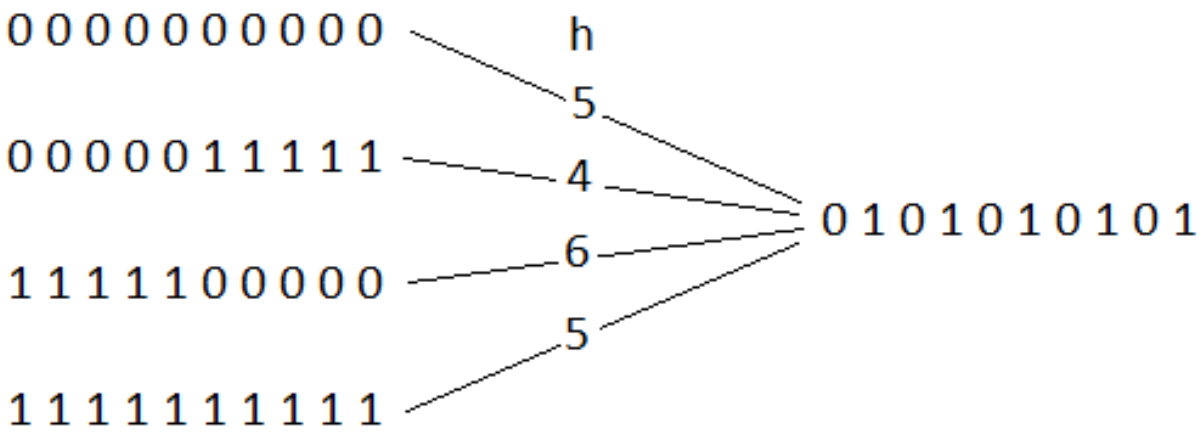
- Distancia mínima entre palabras legales: m = 5
- m = 2e + 1
- Se puede corregir hasta e bits errados
- e = (m – 1)/2 = (5 - 1)/2 = 2

Se pretende corregir hasta e = 2 errores.

m = 2e + 1

m = 5

Códigos legales




Las distancias h son mayores a e = 2

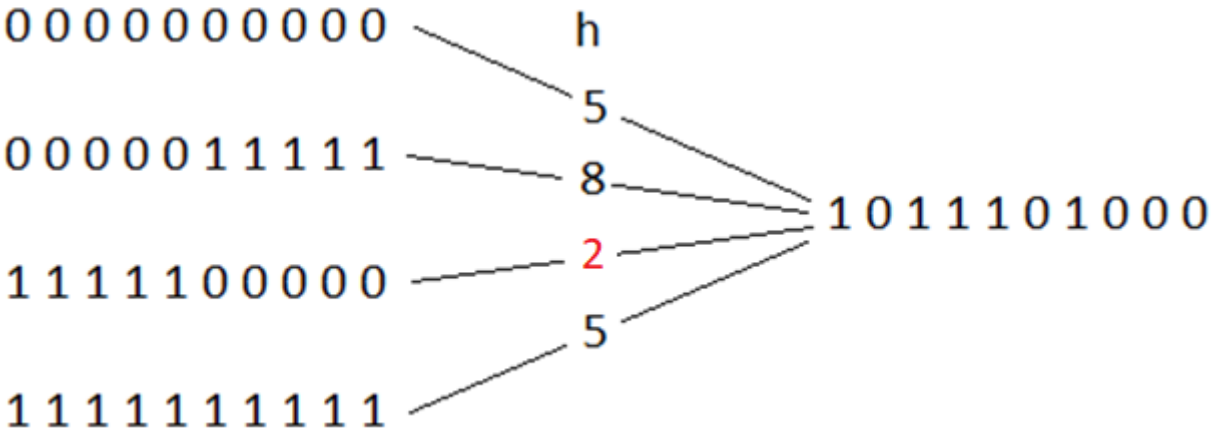
**No** se pueden corregir los errores

Se pretende corregir hasta e = 2 errores.

m = 2e + 1

m = 5

Códigos legales



# **Corrección** de errores. Códigos de Hamming. 2da. forma

- Se desea corregir errores individuales
- Se debe determinar el número de bits redundantes ***r***
- El mensajes tiene *m* bits
- n longitud de la trama
- n = m + r

# ¿Cuál es el número mínimo de bits redundantes *r* ?

- m = longitud del mensaje
- Hay $2^m$ mensajes legales
- Si m = 2 hay 4 mensajes legales: 00, 01, 10, 11
- Haremos que por cada mensaje legal haya $n$ palabras ilegales a una distancia 1 del mensaje
- Entonces, el total de mensajes ilegales tiene que ser $n2^m$
- Palabras legales + ilegales: $2^m + n2^m = (n + 1)2^m$
- Debe hacerse cumplir que: legales + ilegales ≤ $2^n$
- $(n + 1)\ 2^m \leq 2^n$
- n = m + r; $(m + r + 1)\ 2^m \leq 2^{m+r}$
- $m + r + 1 \leq 2^r$
- Por tanteo, si m = 2 entonces r = 3

# Ejemplo

m = 2; **r** = 3; n = m + **r** = 5

legales $2^m = 4$ ilegales $n2^m = 20$ con errores de 1 bit

Los dígitos redundantes **r** se calcularon con los códigos de Hamming

12345

- 00000 10000 01000 00100 00010 00001
- 10011 00011 11011 10111 10001 10010
- 11100 01100 10100 11000 11110 11101
- 01111 11111 00111 01011 01101 01110
- Legales + ilegales = $2^m + n\,2^m = (n + 1)* 2^m = 24$
- Legales + ilegales <= $2^n$
- 24 <= 32

Los 5 códigos de la derecha están a la distancia de 1 bit de su correspondiente código de la izquierda

# Ejemplo de Códigos de Hamming

bits m = 7

$m + r + 1 \leq 2^r$; r = 4

Posición k

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |

| $2^n$ |
|---|
| 1 |
| 2 |
| 4 |
| 8 |

$$3 = 1 + 2$$

Posición k

| | 1 | 2 | **3** | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | | | 1 | | | | | | | | |
| 2 | | | 1 | | | | | | | | |
| 4 | | | | | | | | | | | |
| 8 | | | | | | | | | | | |

$$5 = 1 + 4$$

Posición k

| | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | | | 1 | | 0 | | | | | | |
| 2 | | | 1 | | | | | | | | |
| 4 | | | | | 0 | | | | | | |
| 8 | | | | | | | | | | | |

$$6 = 4 + 2$$

Posición k

| | 1 | 2 | 3 | 4 | 5 | **6** | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | | | 1 | | 0 | | | | | | |
| 2 | | | 1 | | | 0 | | | | | |
| 4 | | | | | 0 | 0 | | | | | |
| 8 | | | | | | | | | | | |

| | Posición k | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | | | 1 | | 0 | | 1 | | 0 | | 0 |
| 2 | | | 1 | | | 0 | 1 | | | 0 | 0 |
| 4 | | | | | 0 | 0 | 1 | | | | |
| 8 | | | | | | | | | 0 | 0 | 0 |

# Paridad par de 1s

| | Posición k | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | 0 | | 1 | | 0 | | 1 | | 0 | | 0 |
| 2 | | | 1 | | | 0 | 1 | | | 0 | 0 |
| 4 | | | | | 0 | 0 | 1 | | | | |
| 8 | | | | | | | | | 0 | 0 | 0 |

# Paridad par de 1s

| | Posición k | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | 0 | | 1 | | 0 | | 1 | | 0 | | 0 |
| 2 | | 0 | 1 | | | 0 | 1 | | | 0 | 0 |
| 4 | | | | | 0 | 0 | 1 | | | | |
| 8 | | | | | | | | | 0 | 0 | 0 |

# Paridad par de 1s

| | Posición k | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| ASCII | | | 1 | | 0 | 0 | 1 | | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | 0 | | 1 | | 0 | | 1 | | 0 | | 0 |
| 2 | | 0 | 1 | | | 0 | 1 | | | 0 | 0 |
| 4 | | | | 1 | 0 | 0 | 1 | | | | |
| 8 | | | | | | | | 0 | 0 | 0 | 0 |

# Código de Hamming

| | Posición k | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $2^n$ | | | | | | | | | | | |
| 1 | 0 | | 1 | | 0 | | 1 | | 0 | | 0 |
| 2 | | 0 | 1 | | | 0 | 1 | | | 0 | 0 |
| 4 | | | | 1 | 0 | 0 | 1 | | | | |
| 8 | | | | | | | | 0 | 0 | 0 | 0 |

Simulación de error en los datos:

Dato enviado: 1001000

Dato recibido con error: 1001001

Posición k

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| $2^n$ | | | | | | | | | | | |
| 1 | 1 | | 1 | | 0 | | 1 | | 0 | | 1 |
| 2 | | 1 | 1 | | | 0 | 1 | | | 0 | 1 |
| 4 | | | | 1 | 0 | 0 | 1 | | | | |
| 8 | | | | | | | | 1 | 0 | 0 | 1 |

| Posición de los bits de paridad: | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Bits recibidos: | 0 | 0 | 1 | 0 |
| Bits calculados en el receptor: | 1 | 1 | 1 | 1 |
| | error | error | | error |

Sumar las potencias de 2 en donde hubo error:
1 + 2 + 8 = 11

El bit 11 está errado. Hay que invertir su valor

Error: 1001001
Dato corregido: 1001000

# Códigos de Hamming. Ejemplo 2

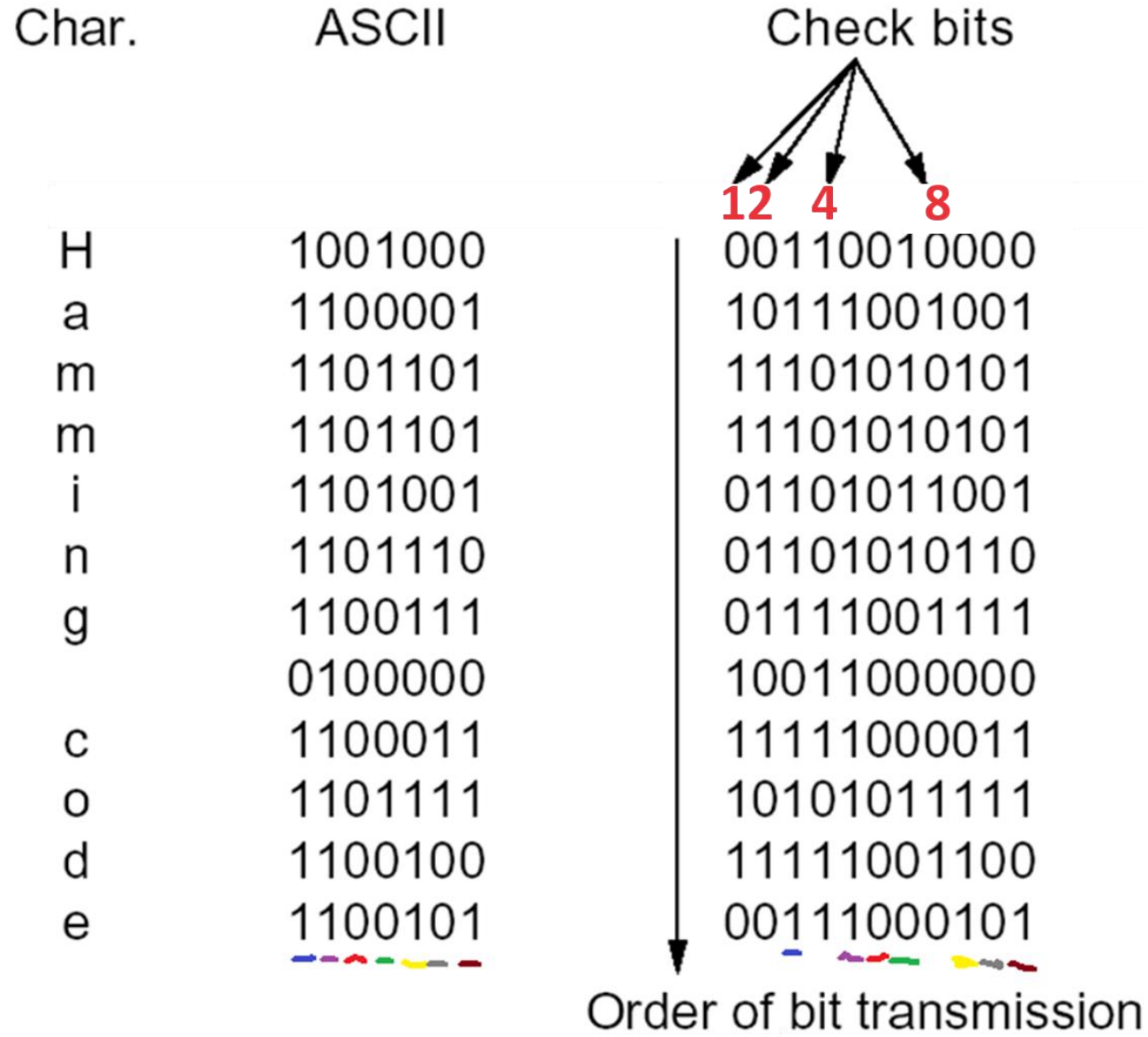
Mensaje: 1 0 1 0 1 0 1 0 1 0

bits m = 10

m + r + 1 ≤ $2^r$;  r = 4

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| $2^n$ | | | | | | | | | | | | | | |
| 1 | 0 | | 1 | | 0 | | 0 | | 1 | | 1 | | 1 | |
| 2 | | 1 | 1 | | | 1 | 0 | | | 0 | 1 | | | 0 |
| 4 | | | | 0 | 0 | 1 | 0 | | | | | 0 | 1 | 0 |
| 8 | | | | | | | | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

- Los Códigos de Hamming sólo corrigen errores individuales
- Hay un truco para corregir códigos de ráfaga
- k palabras se disponen en matriz
- No se transmite por filas, sino por columnas
- Un error de ráfaga de longitud k, dañará a lo mucho 1 bit de cada una de las k palabras


Char.
ASCII
Check bits
12
4
8
H 1001000 00110010000
a 1100001 10111001001
m 1101101 11101010101
m 1101101 11101010101
i 1101001 01101011001
n 1101110 01101010110
g 1100111 01111001111
0100000 10011000000
c 1100011 11111000011
o 1101111 10101011111
d 1100100 11111001100
e 1100101 00111000101
Order of bit transmission

# 3.2.2 Códigos de detección de errores

- Bit de paridad
  - Paridad par: cantidad par de **1**s
  - Paridad impar: cantidad impar de **1**s
- Códigos con bit de paridad tienen distancia mínima m = 2

10100011

11100010

XOR 01000001 h = 2

- Cualquier error en uno o tres bits será detectado
- Es un sistema para detectar un número impar de bits errados

# Código de redundancia cíclica **CRC**

- Cyclic Redundancy Code
- Una cadena de bits se trata como un polinomio, con coeficientes 0 y 1
- Una trama de k bits se considera un polinomio de k términos de grado k - 1
- $b_{k-1}x^{k-1} + b_{k-2}x^{k-2} + \ldots + b_1x + b_0$
- $101011 = 1x^5 + 0x^4 + 1x^3 + 0x^2 + 1x + 1$

- La suma y la resta se reducen a un XOR

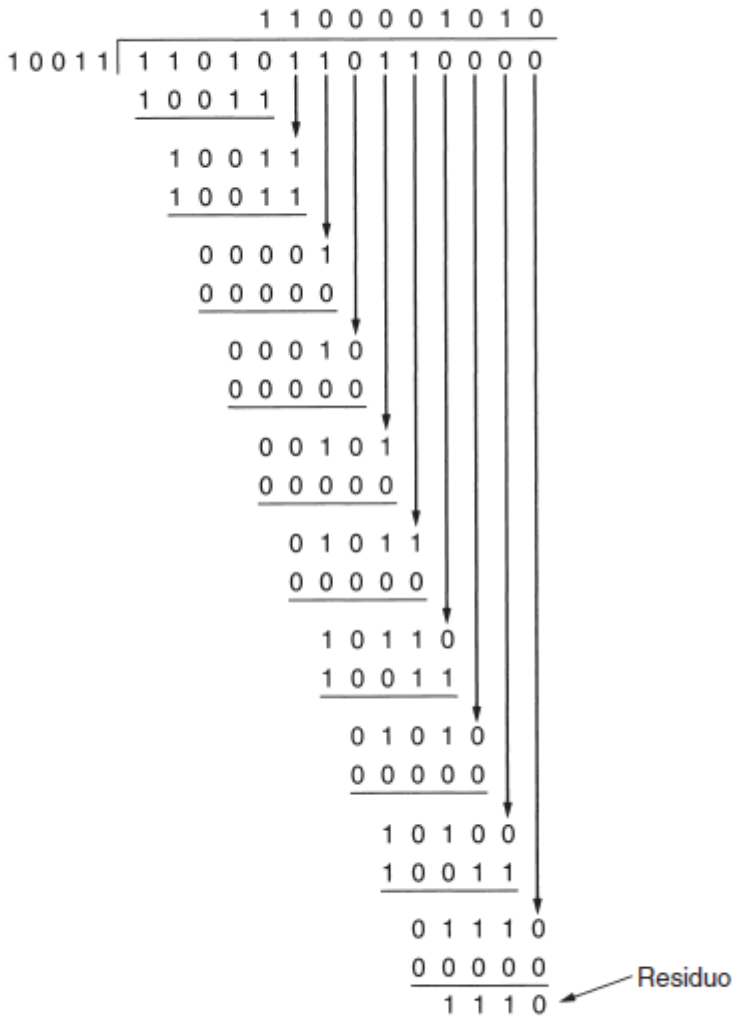
$$\begin{array}{rr} & 10011011 \\ \pm & 11001010 \\ \hline & 01010001 \end{array}$$

- Emisor y receptor acuerdan el generador G(x)
- El mensaje M(x) de m bits, debe ser más grande que G(x)
- Se divide M(x) para G(x)
- El residuo es CRC
- Se añade el CRC al final de M(x)
- T(x) = M(x) + CRC
- T(x) trama que se transmite
- T(x) debe ser divisible para G(x)
- Si hay residuo, hay un error de transmisión

**M(X)** Trama : 1 1 0 1 0 1 1 0 1 1

**G(X)** Generador: 1 0 0 1 1

Mensaje tras anexar 4 bits cero: 1 1 0 1 0 1 1 0 1 1 0 0 0 0




**T(X)** Trama transmitida: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

# 3.3 Protocolos elementales de enlace de datos

# Archivo *protocol.h*

```
#define MAX_PKT 1024                                /* determina el tamaño del paquete
                                                       en bytes */

typedef enum {false, true} boolean;                 /* tipo booleano */
typedef unsigned int seq_nr;                        /* números de secuencia o
                                                       confirmación */
typedef struct {unsigned char data[MAX_PKT];} packet;/*  definición de paquete */
typedef enum {data, ack, nak} frame_kind;           /* definición de frame_kind */

typedef struct {                                    /* las tramas se transportan en
                                                       esta capa */
 frame_kind kind;                                   /* ¿qué clase de trama es? */
 seq_nr seq;                                        /* número de secuencia */
 seq_nr ack;                                        /* número de confirmación de
                                                       recepción */
 packet info;                                       /* paquete de la capa de red */
} frame;

/* Espera que ocurra un evento; devuelve el tipo en la variable event. */
void wait_for_event(event_type *event);

/* Obtiene un paquete de la capa de red para transmitirlo por el canal. */
void from_network_layer(packet *p);

/* Entrega información de una trama entrante a la capa de red. */
void to_network_layer(packet *p);

/* Obtiene una trama entrante de la capa física y la copia en r. */
void from_physical_layer(frame *r);
```

```
/* Pasa la trama a la capa física para transmitirla. */
void to_physical_layer(frame *s);

/* Arranca el reloj y habilita el evento de expiración de temporizador. */
void start_timer(seq_nr k);

/* Detiene el reloj e inhabilita el evento de expiración de temporizador. */
void stop_timer(seq_nr k);

/* Inicia un temporizador auxiliar y habilita el evento ack_timeout. */
void start_ack_timer(void);

/* Detiene el temporizador auxiliar e inhabilita el evento ack_timeout. */
void stop_ack_timer(void);

/* Permite que la capa de red cause un evento network_layer_ready. */
void enable_network_layer(void);

/* Evita que la capa de red cause un evento network_layer_ready. */
void disable_network_layer(void);

/* La macro inc se expande en línea: incrementa circularmente k. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

- Tres casos con niveles de complejidad crecientes
  - Protocolo símplex sin restricciones
  - Protocolo símplex de parada y espera
  - Protocolo simplex para un canal ruidoso

# 3.3.1 Protocolo símplex sin restricciones

- Es sencillo
- Es irreal
- Los datos se transmiten sólo en una dirección
- Las capas de red del emisor y receptor siempre están listas
- Se ignora el tiempo de procesamiento
- Hay un espacio infinito de buffer
- El canal está libre de errores y pérdidas

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                          /* búfer para una trama de salida */
  packet buffer;                    /* búfer para un paquete de salida */

  while (true) {
     from_network_layer(&buffer); /* consigue algo que enviar */
     s.info = buffer;             /* lo copia en s para transmisión */
     to_physical_layer(&s);       /* lo envía a su destino */
  }                                /* Mañana, y mañana, y mañana,
                                      Se arrastra a este mísero paso de día
                                        a día
                                      Hasta la última sílaba del tiempo
                                        recordado
                                          -Macbeth, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event;                 /* ocupado por wait, pero no se usa aquí */

  while (true) {
     wait_for_event(&event);      /* la única posibilidad es frame_arrival */
     from_physical_layer(&r);     /* obtiene la trama entrante */
     to_network_layer(&r.info);   /* pasa los datos a la capa de red */
  }
}
```

- No se usan números de secuencia ni confirmaciones de recepción
- Único evento posible: llega tramas sin daños

# 3.3.2 Protocolo simplex de parada y espera

- El receptor requiere un tiempo para recibir y procesar la trama que llega
- Se evita que el emisor sature al receptor
- El canal está libre de errores y pérdidas
- El tráfico de datos es símplex
- Solución: receptor retroalimenta al emisor
- Luego de enviar una trama, el emisor espera que llegue una trama ACK
- La trama ACK autoriza al emisor para que envíe la siguiente trama
- Por eso el protocolo se llama de «parada y espera»

```
void sender2(void)
{
  frame s;                          /* búfer para una trama de salida */
  packet buffer;                    /* búfer para un paquete de salida */
  event_type event;                 /* frame_arrival es la única posibilidad */

  while (true) {
     from_network_layer(&buffer);   /* consigue algo que enviar */
     s.info = buffer;               /* lo copia en s para transmisión */
     to_physical_layer(&s);         /* adiós a la pequeña trama */
     wait_for_event(&event);        /* no procede hasta que recibe la señal de
                                        continuación */
  }
}

void receiver2(void)
{
  frame r, s;                       /* búferes para las tramas */
  event_type event;                 /* frame_arrival es la única posibilidad */
  while (true) {
     wait_for_event(&event);        /* la única posibilidad es frame_arrival */
     from_physical_layer(&r);       /* obtiene la trama entrante */
     to_network_layer(&r.info);     /* pasa los datos a la capa de red */
     to_physical_layer(&s);         /* envía una trama ficticia para informar
                                       al emisor */
  }
}
```

- En realidad hay flujo de información en ambas direcciones en una alternancia estricta

# 3.3.3 Protocolo símplex para un canal ruidoso

- El canal comete errores o pierde tramas
- Puede ser que el ACK enviado por el receptor se pierda
  - El emisor lo vuelve a enviar duplicándose la trama
  - Para tratar pérdidas y duplicaciones se numera las tramas

- El número de secuencia es de 1 bit (0 ó 1)
- Éste se incrementa módulo 2
- Se transmite también en una sola dirección
- Los protocolos en los que el emisor espera confirmación se llaman ARQ (Automatic Repeat reQuest)
- Los errores se detectan en hardware

```
#define MAX_SEQ 1                                   /* debe ser 1 para el protocolo 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                        /* número de secuencia de la siguiente
                                                       trama de salida */
  frame s;                                          /* variable de trabajo */
  packet buffer;                                    /* búfer para un paquete de salida */
  event_type event;

  next_frame_to_send = 0;                           /* inicializa números de secuencia de
                                                       salida */
  from_network_layer(&buffer);                      /* obtiene el primer paquete */
  while (true){
     s.info = buffer;                               /* construye una trama para transmisión */
     s.seq = next_frame_to_send;                    /* inserta un número de secuencia en la
                                                       trama */
     to_physical_layer(&s);                         /* la envía a su destino */
     start_timer(s.seq);                            /* si la respuesta tarda mucho, expira el
                                                       temporizador */
     wait_for_event(&event);                        /* frame_arrival, cksum_err, timeout */
     if (event == frame_arrival){
           from_physical_layer(&s);                 /* obtiene la confirmación de recepción */
           if (s.ack == next_frame_to_send){
                 stop_timer(s.ack);                 /* desactiva el temporizador */
                 from_network_layer(&buffer);       /* obtiene siguiente a enviar */
                 inc(next_frame_to_send);           /* invierte next_frame_to_send */
           }
     }
  }
}
```

```
void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true){
     wait_for_event(&event);                  /* posibilidades: frame_arrival, cksum_err */
     if (event == frame_arrival){             /* ha llegado una trama válida. */
          from_physical_layer(&r);            /* obtiene la trama recién llegada */
          if (r.seq == frame_expected){       /* esto es lo que hemos estado esperando. */
                  to_network_layer(&r.info);  /* pasa los datos a la capa de red */
                  inc(frame_expected);        /* para la próxima se espera el otro número
                                                 de secuencia */
          }
          s.ack = 1 - frame_expected;         /* indica la trama cuya recepción se está
                                                 confirmando */
          to_physical_layer(&s);              /* envía confirmación de recepción */
     }
  }
}
```

# 3.4 Protocolos de ventana deslizante

- Una forma de transmitir datos dúplex es con 2 circuitos símplex
- Otra forma es usar un circuito para datos en ambas direcciones
- El campo *kind* de la cabecera de la trama indica si es de datos o ACK

# Técnica de superposición o *piggybacking*

- Se añade en el encabezado un ACK a una trama de datos de retorno
- ACK viaja gratis
- Se aprovecha mejor el ancho de banda del canal
- Hay menos interrupciones de "*ha llegado una trama*"
- Problema: Podría expirar el temporizador del emisor porque el receptor no tiene datos que enviar y retransmitir la trama
- Solución: El receptor espera un tiempo. Si el receptor no tiene datos que enviar, envía solo un ACK

# Tres protocolos de ventana deslizante

1. Protocolo de ventana corrediza de un bit
2. Protocolo de retroceso N
3. Protocolo de repetición selectiva

- Son bidireccionales
- Cada trama tiene un número de secuencia
- El emisor registra las tramas enviadas aun no confirmadas
- El receptor tiene una ventana de tramas a aceptar

- Estos protocolos entregan en orden los paquetes a la capa 3
- En el emisor:
  - Si llega un paquete nuevo de la capa 3, el extremo superior de la ventana avanza 1
  - Al llegar un ACK, el extremo inferior de la ventana avanza 1

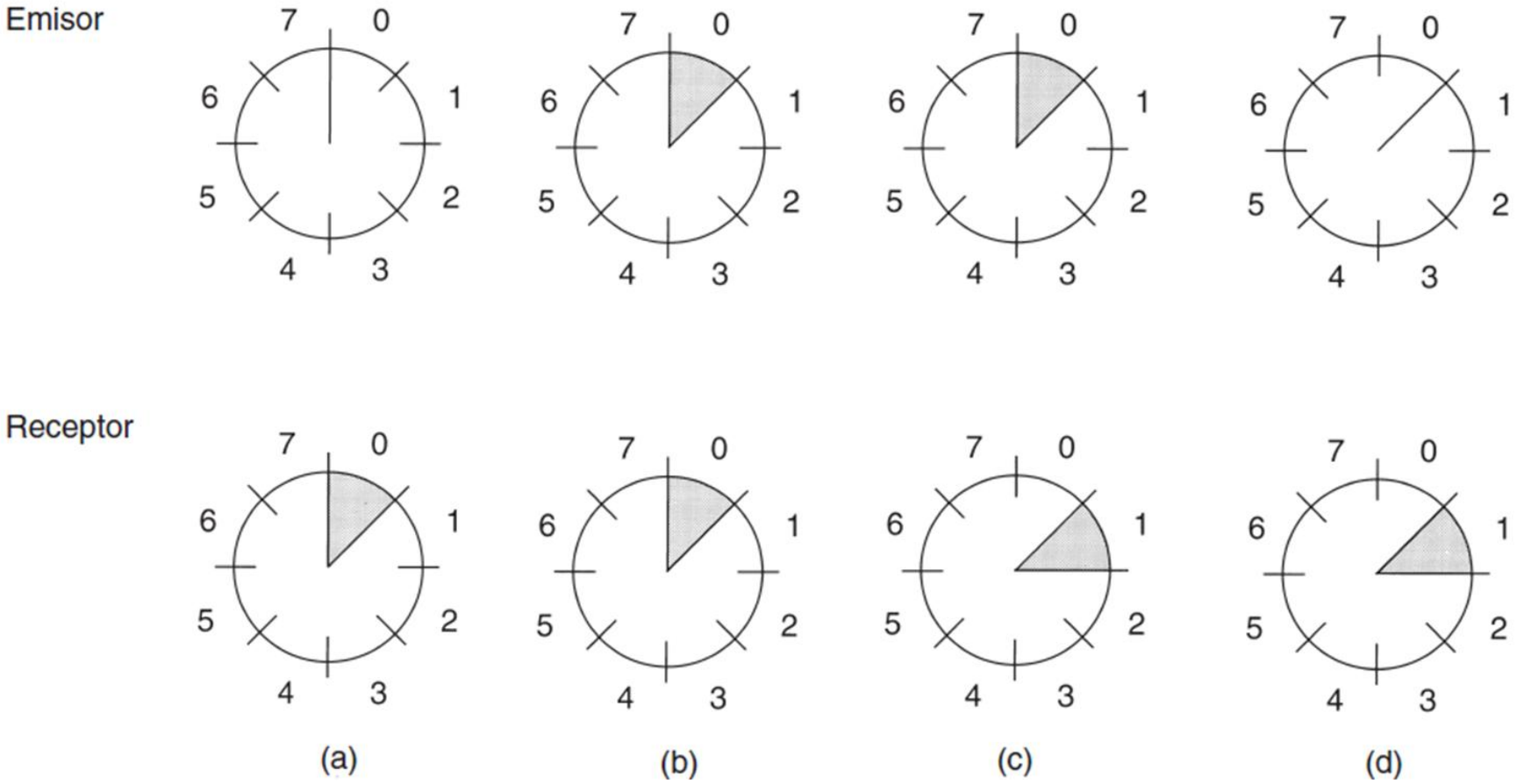



**Figura 3-13.** Ventana corrediza de tamaño 1, con un número de secuencia de 3 bits. (a) Al inicio. (b) Tras la transmisión de la primera trama (c) Tras la recepción de la primera trama. (d) Tras recibir la primera confirmación de recepción.

# 3.4.1 Protocolo de ventana corrediza de 1 bit

- Utiliza parada y espera
- Las tramas se numeran con 0, 1, 0, 1, …
- La transmisión es en los dos sentidos: semiduplex
- Sólo una de las máquinas puede iniciar la transmisión
- Si ambas inician al mismo tiempo hay funcionamiento anormal

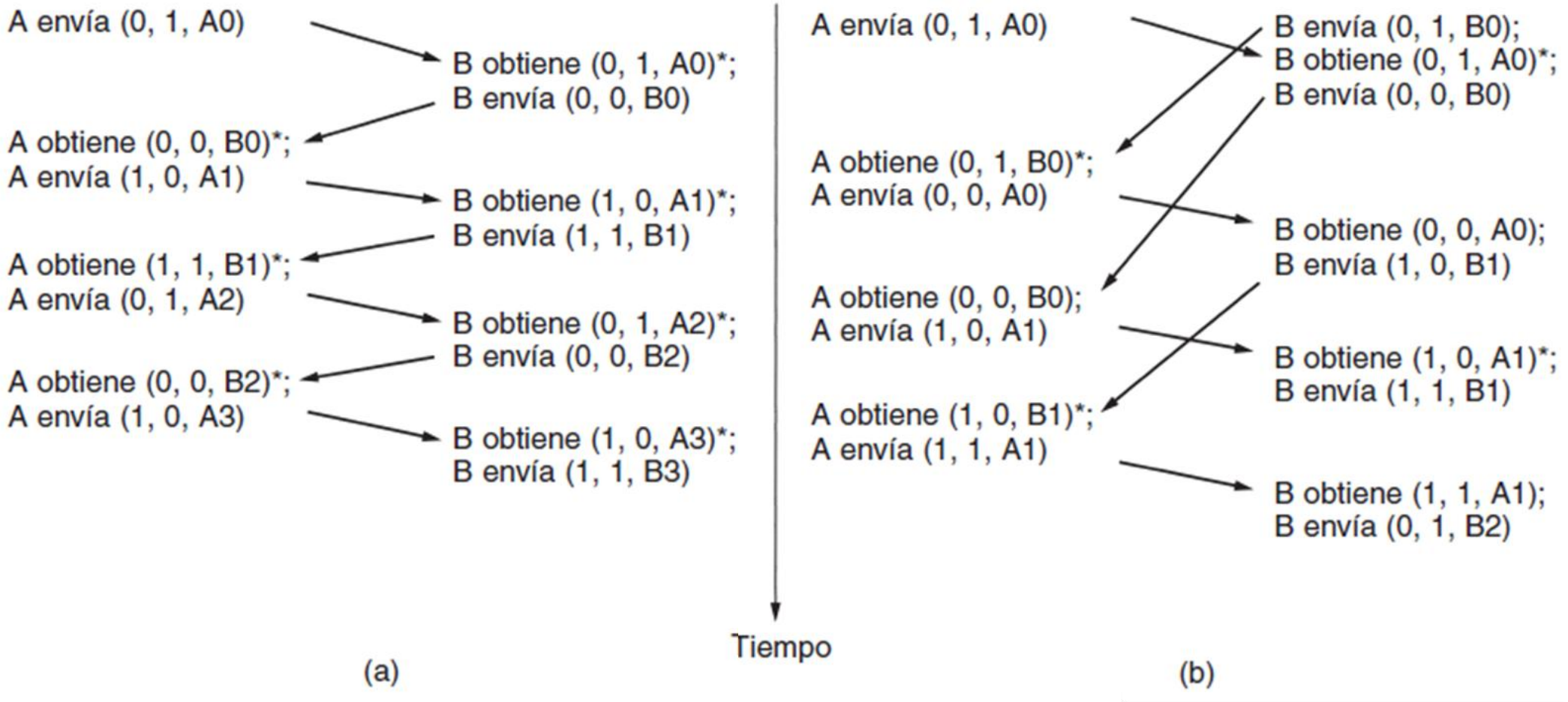
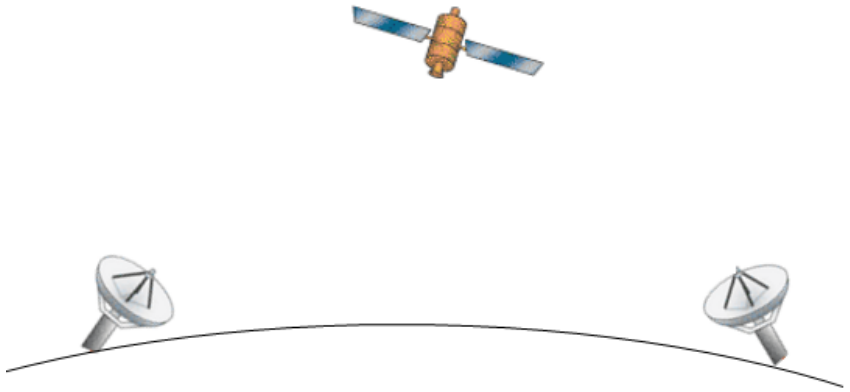



**Figura 3-15.** Dos escenarios para el protocolo 4. (a) Caso normal. (b) Caso anormal. La notación es (secuencia, confirmación de recepción, número de paquete). Un asterisco indica el lugar en que una capa de red acepta un paquete.

# 3.4.2 Protocolo que usa retroceso N: Canalización



- Luego de enviar una trama, el emisor debe esperar el ACK de la trama enviada para poder enviar la siguiente trama
- Enlace satelital de **C** = 50 kbps
- Satélite geoestacionario 35.786 km
- Asumimos que el tiempo de propagación de ida y vuelta es: $\boldsymbol{P} = 500\ ms$
- Longitud de la trama **L** = 1000 bits
- Tiempo de transmisión de la trama T
- $T = \frac{\boldsymbol{L}}{\boldsymbol{C}} = \frac{1000\ bits}{50\ kbps} = 20\ ms;$

- Porcentaje de uso del canal: $\mathrm{U1} = \frac{tiempo\ de\ transmisión\ de\ la\ trama}{tiempo\ de\ propagación\ de\ ida\ y\ vuelta} = \frac{T}{P}$
- $Rendimiento = \frac{20\ ms}{500\ ms} \times 100 = 4\%$
- La eficiencia tan baja del uso del canal se debe a que el emisor debe esperar el ACK antes de enviar la siguiente trama
- El tiempo de espera del emisor luego de que termina de enviar la trama es:
- Tiempo de propagación de ida: 250 ms
- Tiempo para recibir la trama: 20 ms
- Tiempo de propagación del ACK 250 ms
- Tiempo para transmitir una trama y recibir el ACK respectivo T1T = 520 ms
- Se busca aprovechar mejor el uso canal

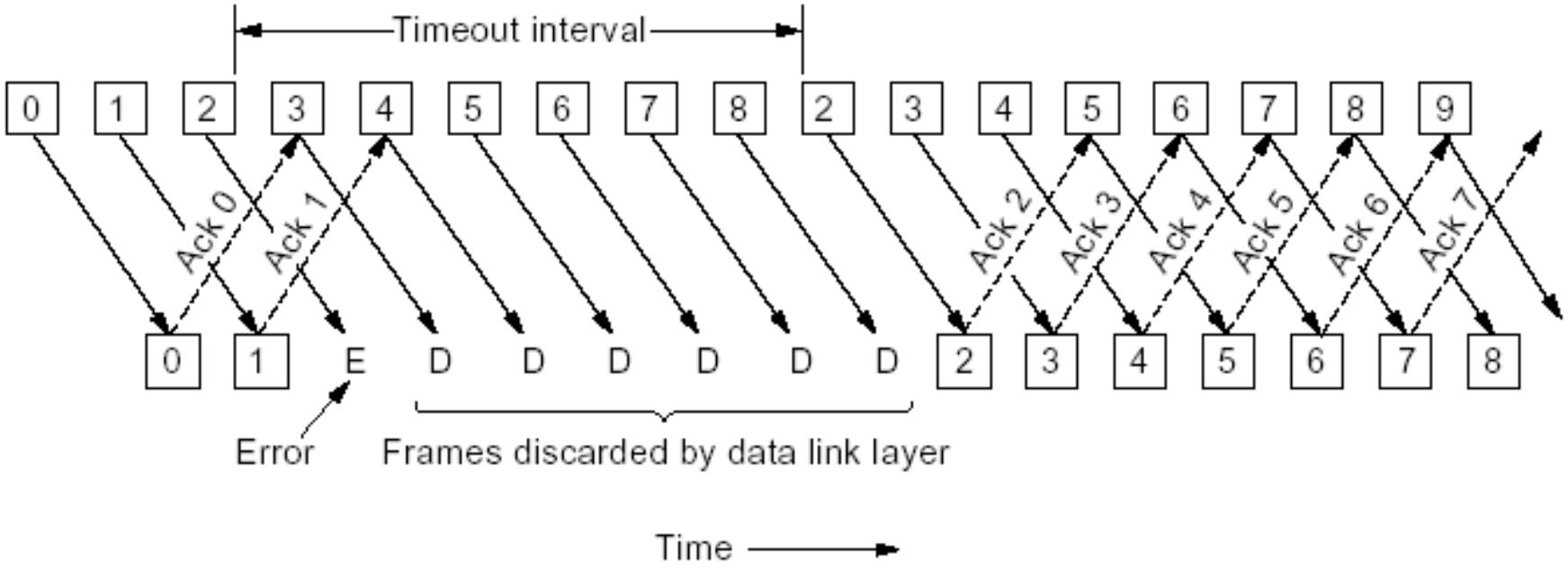
- Solución:
- La idea es que el emisor envíe no una sino un grupo de w tramas antes de que empiecen a llegar los ACK
- En el tiempo de espera del emisor se pueden enviar $w$ tramas
- $w = \frac{Tiempo\ de\ una\ trama+ACK}{Tiempo\ de\ transmisión\ de\ una\ trama} = \frac{T1T}{T} = \frac{520\ ms}{20\ ms} = 26$
- luego de lo cual comienzan a llegar los 25 ACKs
- En resumen 26 tramas que transmite el emisor + 25 ACKs que transmite el receptor
- El tiempo que espera el emisor para transmitir las siguientes 26 tramas es:
- $TT = T1T + 25 \times (tiempo\ para\ que\ el\ receptor\ reciba\ una\ trama)$
- $TT = 520\ ms + 25 \times 20\ ms = 1020\ ms$
- $Rendimiento = \frac{T1T}{TT} \times 100 = \frac{520\ ms}{1020\ ms} = 51\%$

- Al terminar de enviar la trama 26 en t = 520 ms, llega la confirmación de recepción de la trama 1
- Entonces las confirmaciones llegan cada 20 ms

# Métodos para manejar **errores** en canalización

1. Retroceso n
2. Repetición selectiva

# Retroceso n


Timeout interval
0
1
2
3
4
5
6
7
8
2
3
4
5
6
7
8
9
Ack 0
Ack 1
Ack 2
Ack 3
Ack 4
Ack 5
Ack 6
Ack 7
0
1
E
D
D
D
D
D
D
2
3
4
5
6
7
8
Error
Frames discarded by data link layer
Time

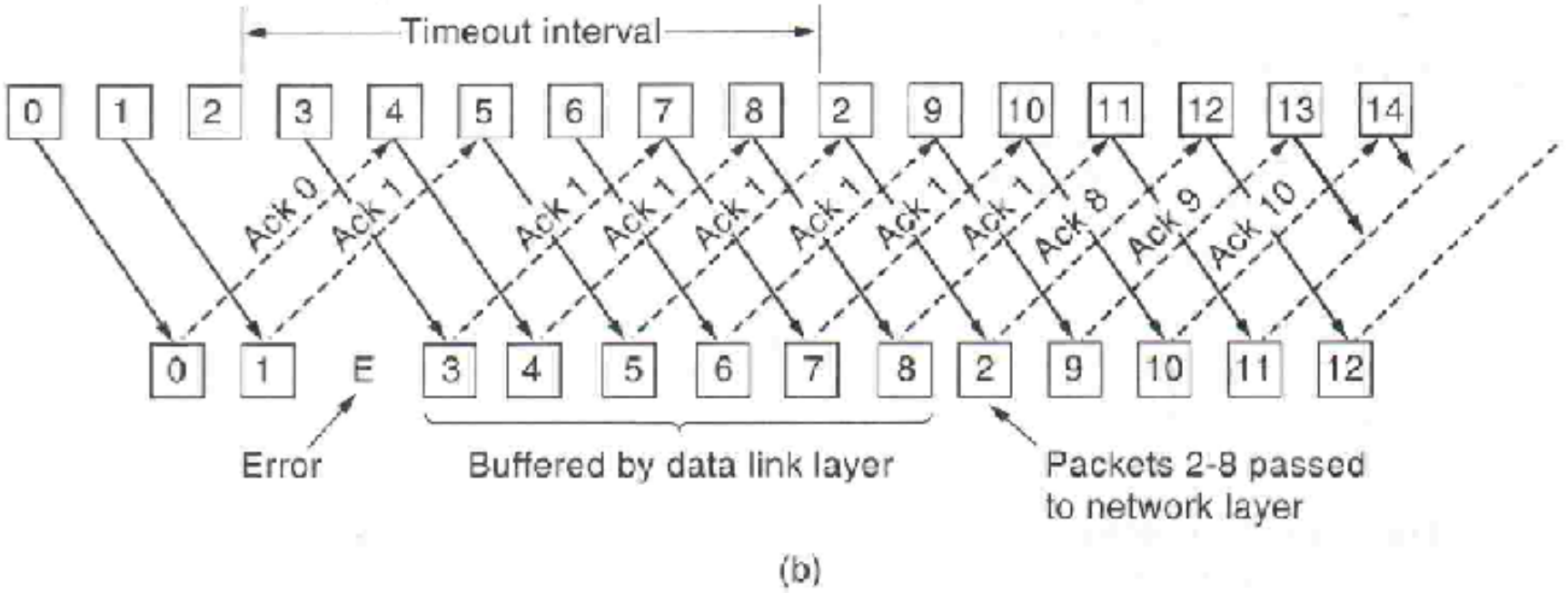
# Retroceso n continuación

- Tamaño de la ventana del receptor = 1
- El receptor descarta las tramas subsiguientes
- El canal de comunicación no se aprovecha si la tasa de errores es alta

# Repetición selectiva

- Ventana del receptor > 1
- Se descarta la trama dañada recibida
- Siguientes tramas recibidas correctamente se almacenan en buffer
- Emisor transmite sólo la última trama sin ACK

# Repetición selectiva




(b)

# Repetición selectiva

- El receptor podría enviar un NACK al detectar error para estimular la retransmisión antes de que expire el temporizador del emisor
- El emisor y receptor mantienen una ventana

# 3.5 EJEMPLOS DE PROTOCOLOS DE ENLACE DE DATOS

# Transmisión serie y paralelo

- Transmisión serie. Los bits se transmiten a través de un único camino
- Transmisión paralelo. Los bits se transmiten en forma simultánea a través de varios caminos
- En telemática se estudia solamente la transmisión serie

# Muestreo de la línea de transmisión

- En la transmisión de datos es importante la sincronización de los relojes del emisor y receptor
- El receptor debe saber la velocidad a la que se están transmitiendo los datos y en qué momentos inicia un bit
- Esto le permite muestrear la línea a intervalos constantes de tiempo para leer los bits
- Para ellos se utilizan dos técnicas de transmisión:
  - asíncrona y
  - síncrona

# Transmisión asíncrona

- La transmisión se realiza de carácter en carácter
- Cada carácter o byte se trata independientemente
- El primer bit de cada carácter marca su inicio y alerta al receptor de la llegada del carácter
- El último bit es el de paridad (par o impar) y sirve para validar cada carácter
- El % de estos bits suplementarios es alto

# Transmisión síncrona

- La transmisión se realiza en bloques de tramas
- El % de bits suplementarios para sincronizar y validar tramas es mucho menor
- Por tal razón la transmisión es más eficiente

# Líneas síncronas y asíncronas

- En otro contexto
- Línea síncrona. Línea telefónica arrendada o dedicada. Siempre está a disposición. Actualmente se usa FO
- Línea asíncrona. Línea telefónica básica o dial-up. La línea podría estar ocupada. Estas líneas hoy son obsoletas

# Protocolos de Capa 2

- Las redes pueden utilizar líneas punto a punto o líneas punto-multipunto
- WAN utiliza líneas punto a punto
- Un computador con terminales tontas utilizan líneas punto-multipunto
- HDLC High-level Data Link Control está diseñado para manejar líneas punto a punto y líneas punto-muptipunto
- PPP Point-to-Point Protocol se diseñó para manejar solamente líneas punto a punto, es decir, líneas que conectan dos computadoras directamente

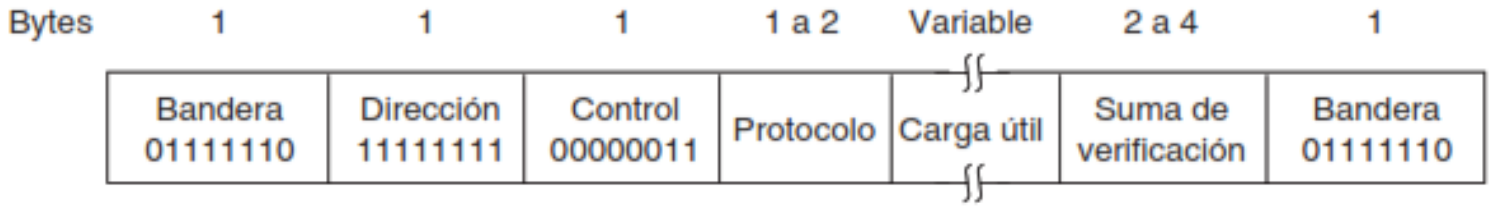
# HDLC High-level Data Link Control

- Es un estándar de OSI
- Es la base de otros protocolos muy utilizados
- Protocolo orientado a bit
- Permite tramas con número fraccionario de bytes
- Dentro de la carga útil usa relleno de bits (Sección 3.1.2)
- Se inserta en los datos un 0 si hay 5 1s seguidos
- Esto ayuda a la sincronización en la capa física
- Cada trama empieza y termina con la bandera o señalizador 01111110
- La bandera permite además sincronizar los relojes de emisor y receptor
- Usa transmisión con ventana deslizante y ACKs

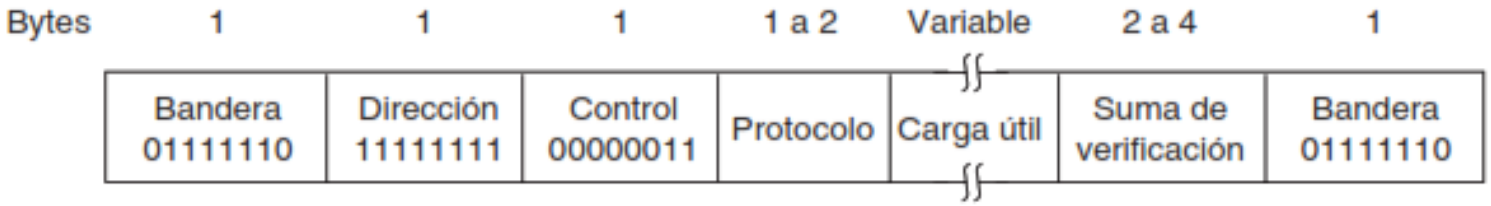

- Dirección: solo se necesita en transmisiones punto-multipunto (computador-terminales tontas)
- Se refiere al campo de dirección de la estación destino
- 11111111 es la dirección de broadcast
- En redes LAN el protocolo que se usa es Ethernet

# PPP Point-to-Point Protocol

- Pertenece al conjunto de protocolos TCP/IP y es una mejora de SLIP
- Está orientado a bytes
- Las tramas tienen un número entero de bytes
- Dentro de la carga útil utiliza relleno de caracteres




- Solo se necesita una bandera entre tramas, no dos
- Si no hay tramas para enviar se envían una secuencia de banderas




- Al ser un protocolo punto a punto no se requiere del campo Dirección
- Este campo está siempre en 11111111
- Control siempre tiene la secuencia 00000011
- Protocolo: IPv4, IPv6, IPX (Netware Novell), AppleTalk
- Carga útil de longitud variable. Longitud predeterminada 1500 bytes
- Usa Checksum para verificar la integridad de los datos, no usa CRC
- PPP es usado con muchos tipos de capas físicas

- Checksum es una técnica de verificación de errores que suma los valores de los datos transmitidos y luego envía el resultado junto con los datos
- CRC es un método de verificación más sofisticado que utiliza operaciones matemáticas de división en un campo finito (álgebra binaria) para generar un valor de verificación