

# **Redes de Computadores**

## **Tema 3 – Capa de enlace de datos**

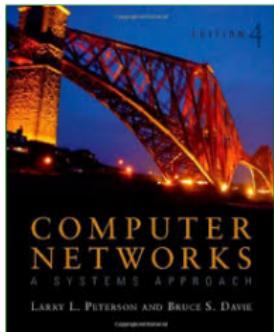
**Natalia Ayuso, Juan Segarra y Jesús Alastruey**



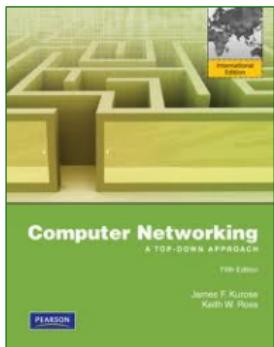
Departamento de  
Informática e Ingeniería  
de Sistemas

**Universidad** Zaragoza

1. Introducción
2. Definiciones y métricas
3. Control de acceso al medio (MAC)
4. Protocolos de particionado de canal
5. Protocolos de acceso aleatorio
6. Protocolos “por turnos”
7. Comutación en Ethernet
8. Control del enlace de datos
9. Control de errores
10. Secuenciación de datos



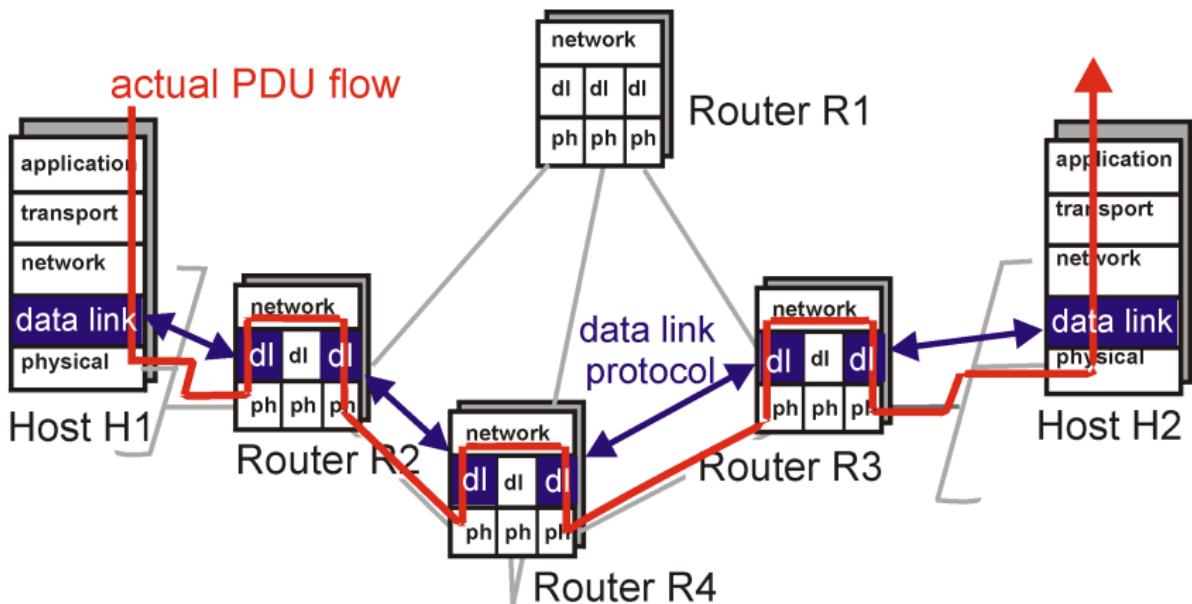
Capítulo 2



Capítulo 5

# 1 Introducción

## ► Contexto:



Fuente: The Data Link Layer: Introduction, Services

# 1 Introducción (II)



Funciones típicas de la capa de enlace:

- Reglas para que varias entidades compartan un mismo canal de transmisión: *control de acceso al medio (MAC)*
- *Comunicación fiable* entre dos entidades:
  - Control de errores
  - Secuenciación de datos



- Unidad básica de nivel de enlace: *trama (frame)*

## 2 Definiciones y métricas

---



Longitud trama ( $L$ ): longitud de la trama (bits, b)

Velocidad de transmisión ( $V_t$ ): “User data rate” velocidad nominal de la capa MAC (bits/segundo,bps)

Tiempo de transmisión ( $T_t$ ): tiempo necesario para injectar una trama en el medio de transmisión (segundos, s)  $T_t = \frac{L}{V_t}$

Velocidad de capa física ( $V_{PHY}$ ): tasa de bits que se inyecta al medio de transmisión (bits/segundo, bps). Incluye los datos de usuario y el overhead requerido en la capa física. Eg.

4B/5B

- Ethernet 100BASE-TX,  $V_t = 100$  Mbps (capa MAC)  
 $V_{PHY} = \frac{5}{4} \times V_t = 125$  Mbps (capa Física)

## 2 Definiciones y métricas (II)



- Ethernet anuncia la capacidad del usuario
- Wi-Fi anuncia la máxima posible. E.g. 802.11a/g de 54 Mbps → 20-25 Mbps; 802.11n de 600 Mbps → 150 -200 Mbps

Ethernet Standard	Common Name	Data Rate (Bit Rate)
10BASE-T	Standard Ethernet	10 Mbps
100BASE-TX	Fast Ethernet	100 Mbps (125 Mbps 4B/5B)
1000BASE-T	Gigabit Ethernet	1 Gbps (1.25 Gbps 8B/10B)
10GBASE-T	10 Gigabit Ethernet	10 Gbps
25G, 40G, 100G, 400G	Data Center Standards	25 Gbps up to 400 Gbps

Standard	Wi-Fi Generation	Frequency Band(s)	Data Rate (Bit Rate)
802.11b	Wi-Fi 2	2.4 GHz	11 Mbps
802.11a/g	Wi-Fi 3	5 GHz (a), 2.4 GHz (g)	54 Mbps
802.11n	Wi-Fi 4 (HT)	2.4 GHz and 5 GHz	600 Mbps
802.11ac	Wi-Fi 5 (VHT)	5 GHz	6.9 Gbps
802.11ax	Wi-Fi 6 (HE)	2.4 GHz, 5 GHz, and 6 GHz	9.6 Gbps
802.11be	Wi-Fi 7 (EHT)	2.4 GHz, 5 GHz, and 6 GHz	46 Gbps

### ➤ Velocidad en la LAN:

- Real: iperf3, transferencia de fichero (1 GB)...
- La que se negocia en la red (máxima): ~\$ iw wlp3s0 link  
~\$ cat /sys/class/net/eth0/speed

## 2 Definiciones y métricas (III)

---



Distancia ( $D$ ): longitud del enlace (metros, m)

Velocidad de propagación ( $V_p$ ): velocidad a la que la onda EM viaja por el enlace (metros/segundo, m/s) [Tema 2]

Tiempo de propagación ( $T_p$ ): tiempo necesario para que la onda EM viaje de emisor a receptor (segundos, s)

E.g. en ethernet, un segmento de red está limitado a 100 m.

Del dispositivo al router pasando por 3 switches, 300 m. Los retardos son del orden de los  $\mu$  s

Round Trip Time (RTT): es el tiempo de ida y vuelta de la señal ( $RTT = 2 \cdot T_p$ ).

## 2 Definiciones y métricas (IV)

Tiempo de acceso al canal ( $T_a$ ): desde que un nodo quiere transmitir hasta que empieza a transmitir la trama «definitiva» (esperas por colisiones, por turnos, etc.)

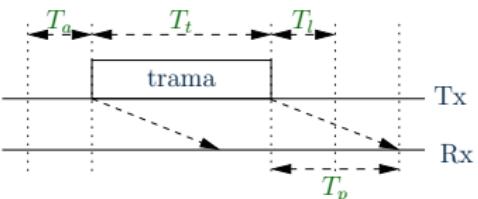
Tiempo liberación del canal ( $T_l$ ): desde finalización de transmisión hasta intento de una nueva transmisión

Tiempo de procesamiento: espera / toma de decisiones

Velocidad efectiva ( $V_e$ ): tasa media de bits de datos enviados

Utilización efectiva del canal ( $U_e$ ): porcentaje de tiempo en el que se transmiten datos por el canal

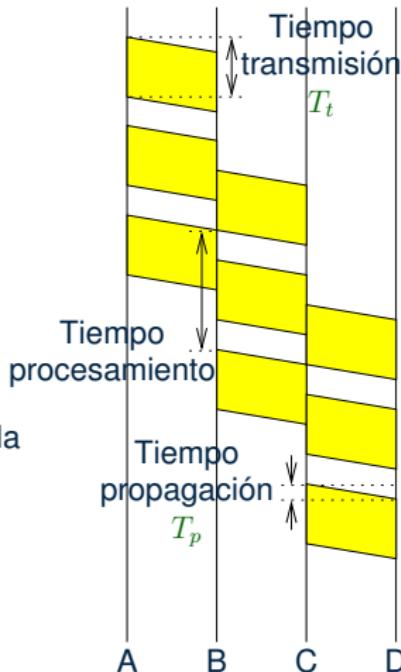
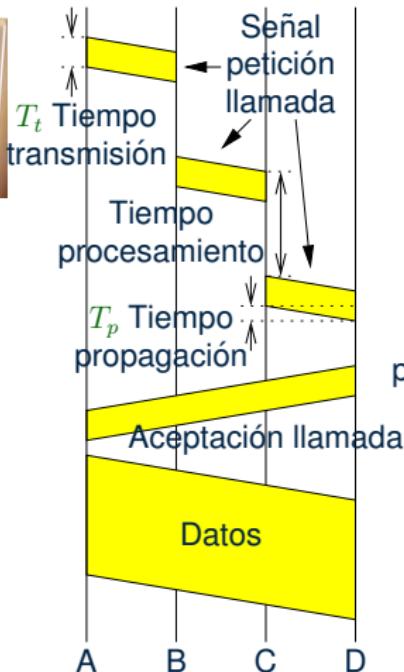
$$V_e = \frac{\text{datos}}{T_a + T_t + T_l} \quad U_e = \frac{V_e}{V_t} \cdot 100$$



 Reproduce los tiempos en un diagrama vertical

## 2 Definiciones y métricas (V)

### ► Comutación de circuitos vs. paquetes



## 2 Definiciones y métricas (VI)

---



### ► Comutación de circuitos:

- Se establece *un circuito* antes de iniciar la transmisión
- El establecimiento va asociado con una reserva de recursos en los nodos
- Una vez establecido el circuito, todos los datos viajan por él
- Si se corta el circuito, la conexión se cierra y hay que reiniciarla
- *Calidad de servicio* (QoS) implícita en el circuito establecido
- *Control de admisión*: cuando se agotan los recursos, no se permiten más circuitos y no se admiten más conexiones
- *Comutación por circuito virtual*: uso de circuitos lógicos sobre comutación de paquetes [Tema 4]
- Ejemplos: **Spectrum monitoring. Standards**

## 2 Definiciones y métricas (VII)

### ➤ Comutación de paquetes:

- Comm. paquetes es más *tolerante a fallos* que comm. circuitos: ante fallos, los paquetes pueden ir por rutas alternativas [Tema 4]
- Suele implicar *más sobrecarga*: información de control por paquete y no por canal como en commutación de circuitos
- Ocupación del canal bajo demanda
  - Si no hay datos a transmitir, no se ocupa el canal
  - Funciona muy bien para tráfico en ráfagas (usual en Internet)

### 3 Control de acceso al medio (MAC)

- Un medio de transmisión suele ser compartido
- Si varios emisores transmiten a la vez, sus señales pueden colisionar (superponerse) y no ser recibidas correctamente
- El control de acceso al medio establece reglas de uso del canal para evitar/minimizar colisiones
- Tipos básicos de MAC:
  - *Protocolos de particionado del canal:* TDMA, FDMA y CDMA
  - *Protocolos de acceso aleatorio:* Aloha y CSMA
  - *Protocolos “por turnos”:* consulta y paso de testigo

## 4 Protocolos de particionado de canal

---

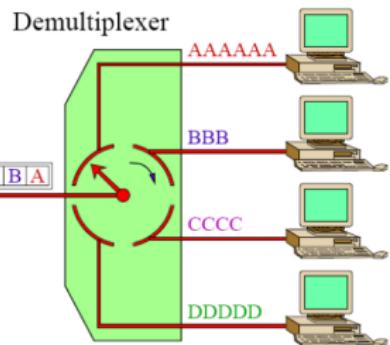
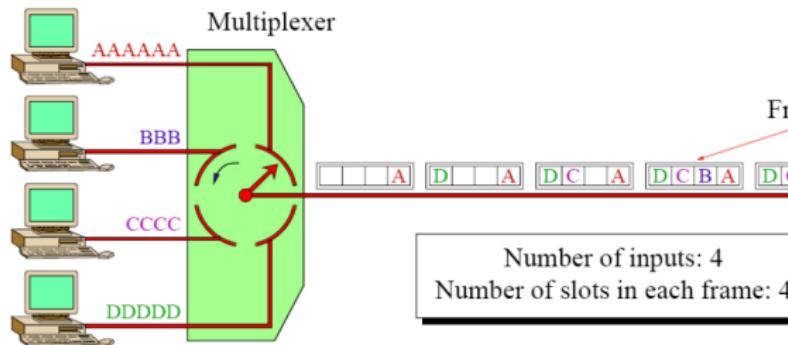
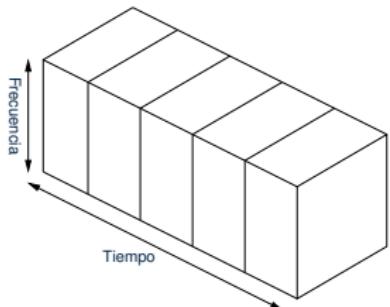


- Dividen el medio compartido en “trozos” más pequeños sin solapamientos
- Cada “trozo” se reserva para un usuario específico o un par de usuarios, eliminando las colisiones por completo

# 4.1 TDMA

*Time Division Multiple Access: acceso múltiple por división en tiempo (TD)*

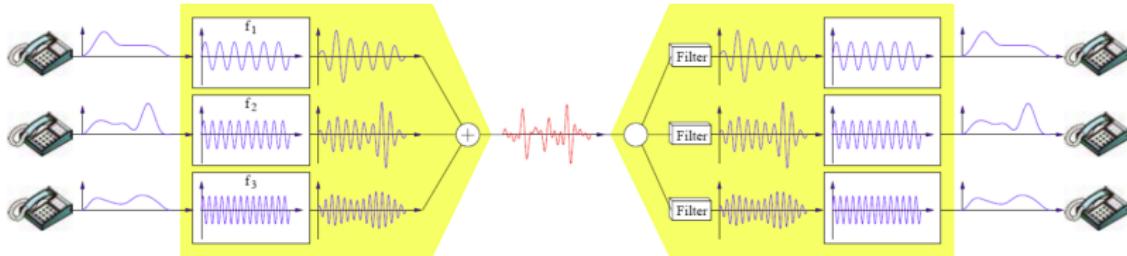
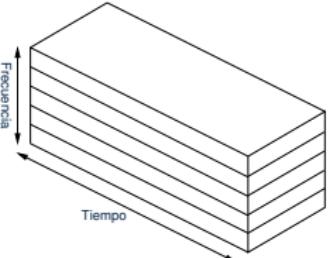
- Necesita *buffers*
- Necesita sincronización
- Fácil jerarquización de canales
- E.g. GSM (Global System for Mobile communications) y Bluetooth



## 4.2 FDMA

*Frequency Division Multiple Access: acceso múltiple por división en frecuencia (FD)*

- Señales moduladas con portadoras a distinta frecuencia (sin solapar)
- E.g. Bluetooth: 79 canales 1 MHz, 2.402-2.480 GHz, ADSL (Asymmetric Digital Subscriber Line): 1 canal voz,  $n$  canales uplink,  $m(> n)$  downlink



## 4.2.1 WDMA

*Wavelength Division Multiple Access:* acceso múltiple por división en longitudes de onda (WD)

- División en frecuencia en comunicaciones ópticas

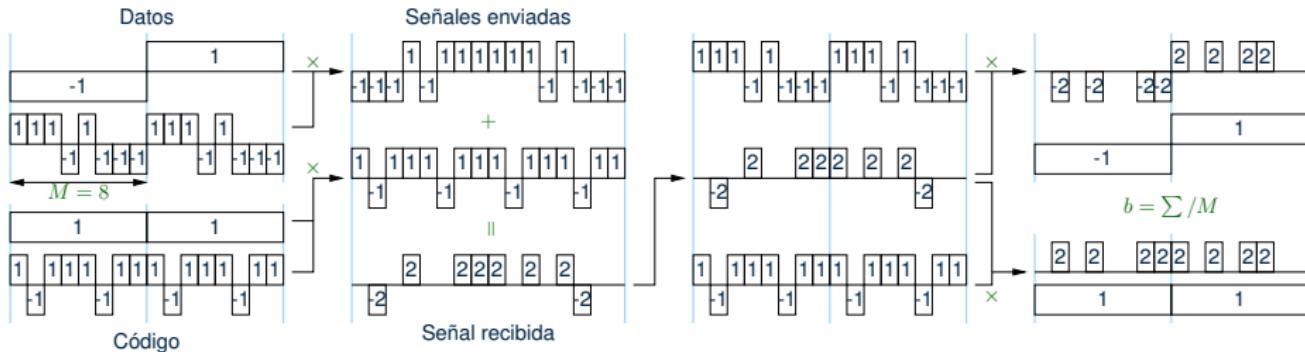
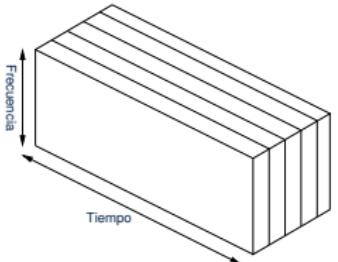


Fuente: Commscope. Data Center Best Practices

# 4.3 CDMA

*Code Division Multiple Access: acceso múltiple por división de código (CD).*

- ▶ Código de espectro ensanchado
- ▶ Usado en comunicaciones inalámbricas
- ▶ Mayor tolerancia ante interferencias
- ▶ E.g. GPS (Global Positioning System),  
UMTS (Universal Mobile Telecommunication System)



## 5 Protocolos de acceso aleatorio

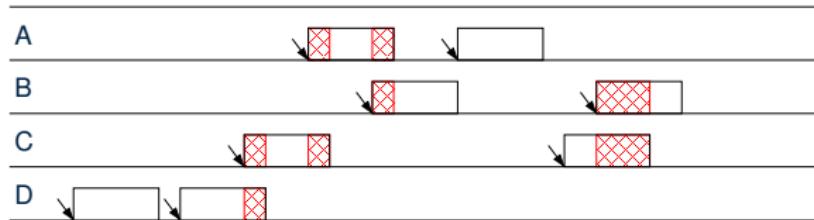
---

- Permiten que una estación transmita cuando quiera
- Las colisiones son posibles!
- Se basa en protocolos que detectan y se recuperan de las colisiones

# 5.1 Aloha

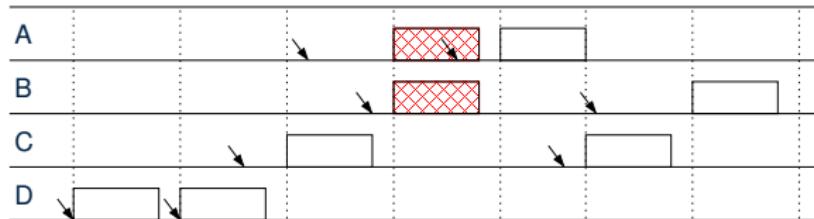
Pensado para comunicaciones de radio entre islas

- Aloha puro: pueden iniciarse transmisiones en cualquier momento. Si no hay colisión:  $T_a = 0$



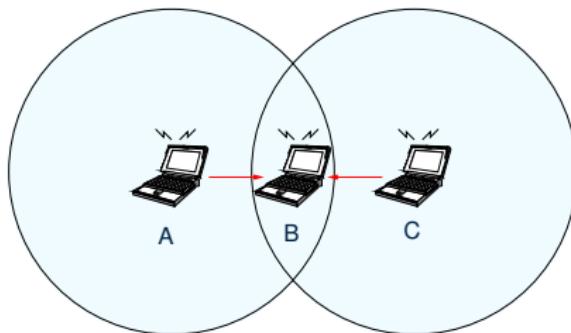
- Aloha ranurado: sólo pueden iniciarse transmisiones al inicio de ranuras/slots periódicas. Si no hay colisión:

$$T_a = [0, T_{slot}]$$



## 5.2 CSMA/CA

*Carrier Sense Multiple Access / Collision Avoidance*



Procedimiento emisor

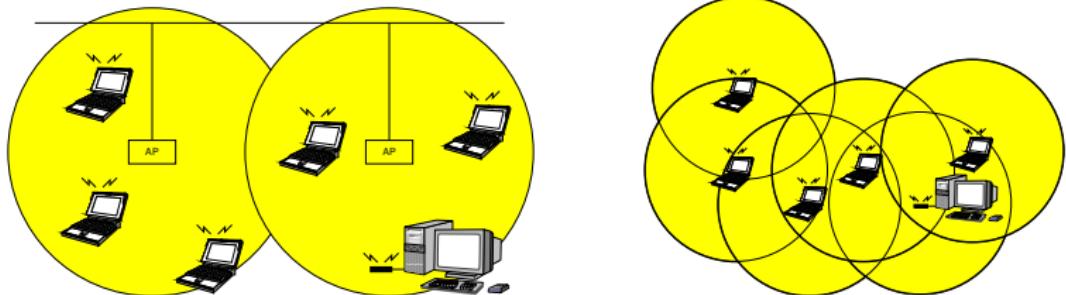
1. Escucha el canal hasta que esté libre +
2. Espera un tiempo breve (*interframe gap*) y transmite
3. Espera confirmación de recepción (trama ACK)
4. Si no recibe ACK (colisión en el receptor por nodo oculto), espera un tiempo aleatorio y vuelve al paso 1

## 5.2 CSMA/CA: *backoff* exponencial

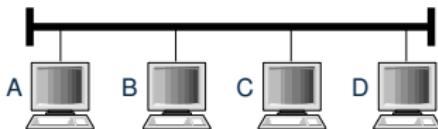
- Si el *tiempo de espera* fuera determinista → nueva colisión
- Siendo aleatorio se reduce la probabilidad de colisión
- Se adapta aleatoriedad a probabilidad de colisión
  - Empezar asumiendo poca probabilidad
  - Ante colisiones consecutivas, aumentar exponencialmente el intervalo de tiempo máximo
- $TiempoEspera = backoff \cdot T_{slot}$ . E.g.  $T_{slot} = 9 \mu s$  en 802.11n
- Ejemplo:
  - 1 col:  $backoff \in \{0, 1\}$
  - 2 col:  $backoff \in \{0, 1, 2, 3\}$
  - 3 col:  $backoff \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
  - 4 col:  $backoff \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
  - ...
- Inconveniente: efecto LIFO (*last-in-first-out*)

## 5.2 CSMA/CA: uso en 802.11

- Puntos de acceso (access point, AP) 
  - Los APs conforman una red ya desplegada
  - Los nodos se conectan a uno de los APs
  - Los APs proporcionan configuración dinámica de red a los nodos (DHCP)
  - Los nodos envían/reciben siempre a través del AP
- Redes ad-hoc
  - No hay una red desplegada a la que conectarse
  - Los nodos se autoorganizan funcionando como retransmisores



## 5.3 CSMA/CD



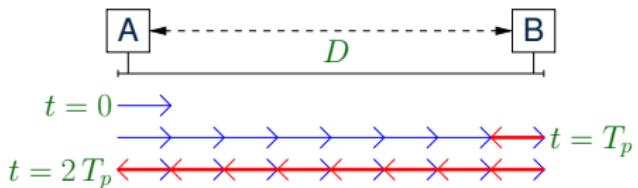
*Carrier Sense Multiple Access with Collision Detection*

Procedimiento emisor

1. Escucha el canal hasta que esté libre
  2. Espera un tiempo breve (*interframe gap*) y transmite
  3. Si detecta colisión, sustituye la transmisión por una señal corta (32 bits) de alerta (*jam*) para enfatizar la colisión
  4. Después de enviar la señal de alerta, espera un tiempo aleatorio y vuelve al paso 1
- No hay ACK → *el emisor debe detectar colisiones*
  - *Backoff* igual que en CSMA/CA, con  $T_{slot} \approx T_t$  (*trama min*)
  - Usado en IEEE 802.3 Ethernet

## 5.3.1 Detección de colisiones

El emisor debe detectar *siempre* toda colisión en sus envíos



- Cuando el emisor transmite, comprueba que la señal en el canal coincide con la que envía
- Hay que transmitir al menos  $T_t = 2 \cdot T_p$  para detectar colisión en el punto más lejano, con  $T_p = D/V_p$  y  $T_t = L/V_t$
- Por tanto:  $T_t = L/V_t \geq 2 \cdot T_p = 2 \cdot D/V_p$ 
  - Si se fija  $T_{p_{max}}$ , entonces  $L_{min} = 2 \cdot T_{p_{max}} \cdot V_t$
  - Por ejemplo, en Ethernet sobre par trenzado:  
 $T_{p_{max}} = 25.6 \mu s, V_t = 10 Mbps, V_p = 177000 km/s$   
 $\rightarrow L_{min} = 512 \text{ bits} = 64 \text{ bytes}$  y  $D_{max} = 4531 \text{ m}$

## 5.3.1 Detección de colisiones: ejercicio



✍ Se desea aumentar la velocidad de transmisión Ethernet a  $V_t = 100 \text{ Mbps}$ .

1. Si se mantiene el tiempo máximo de propagación,  $T_{p_{max}} = 25.6 \mu\text{s}$ , ¿qué ocurre con el tamaño mínimo de trama Ethernet  $L_{min}$ ?
2. Si se mantiene el tamaño mínimo de trama,  $L_{min} = 64 \text{ bytes}$ , ¿qué ocurre con el tiempo máximo de propagación  $T_{p_{max}}$ ? ¿Y con la distancia máxima  $D_{max}$ ?

## 5.3.2 Ejemplo trama

Ejemplo: Trama Ethernet II (64–1518 bytes + preámbulo)

7	1	6	6	2	46–1500	4
Preámbulo	SFD	Dir. destino	Dir. origen	Tipo	Datos	FCS

Preámbulo: 7 x 10101010

SFD: Start-of-Frame-Delimiter: 10101011

Direcciones: identificadores MAC de las tarjetas de red origen y destino. Por ejemplo, para lab000: 50:65:f3:42:43:37<sup>i</sup>

Tipo: identificador<sup>i</sup> del protocolo encapsulado en el campo «Datos» (demultiplexor)

FCS: Frame Check Sequence, CRC de 32 bits [p. 54]



# 6 Procolos “por turnos”

---

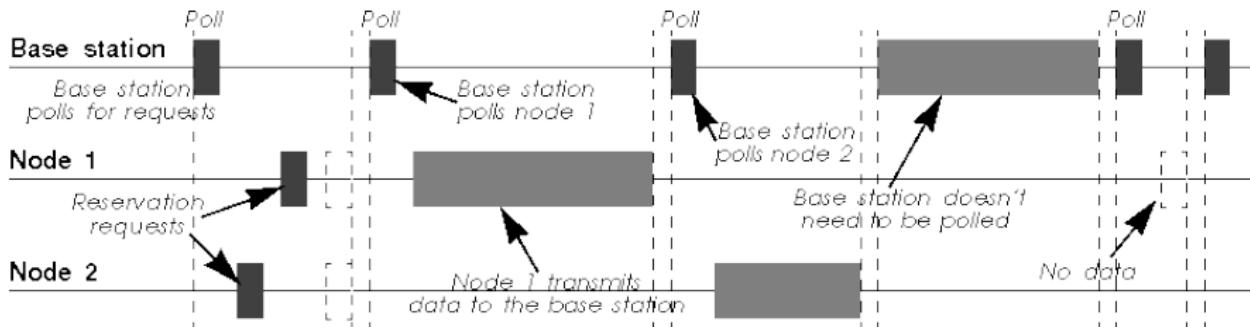


- Pensados con las ventajas de los dos anteriores: evitar la colisión sin la ineficiencia de las reservas cuando los usuarios están *idle*

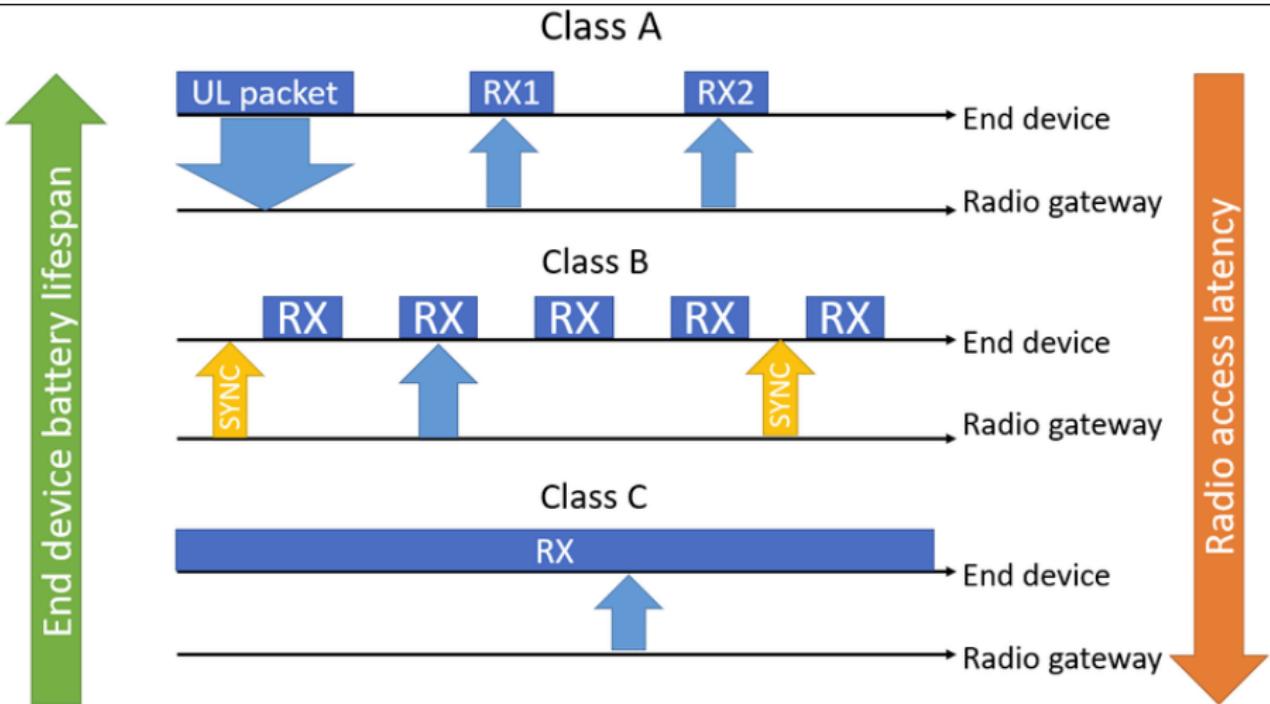
# 6.1 Polling



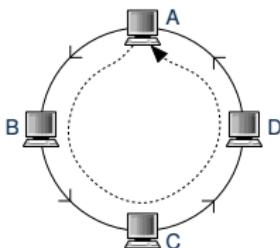
- Se basa en una estación que actúa como controladora y el resto responden
- Funcionamiento: la controladora consulta/invita a cada estación. Si tiene datos los transmite y si no, lo indica
- Es muy eficiente cuando hay mucha carga
- Es vulnerable si falla la controladora



## 6.1.1 Ejemplo LoRa



## 6.2 Token Ring (IEEE 802.5)



- Estaciones conectadas en anillo unidireccional
- Cada estación propaga las tramas que recibe (con 1 bit de retardo para procesar bits particulares)
- Una trama especial (*testigo/token*) circula por el anillo
- Cuando se recibe el token, se puede capturar o propagar
- Para transmitir hay que estar en posesión del *token*:
  - Esperar a que llegue el *token* y capturarlo
  - Transmitir la trama de datos en un sentido del anillo
  - Quitar la trama de datos cuando llegue por el otro lado
  - Transmitir el *token*

## 6.2 Token Ring (IEEE 802.5) (II)



- Transmisiones por turnos → no hay colisiones
- No hay tamaño máximo de trama,  
sino tiempo máximo de posesión del *token*
- El *token* no se puede transmitir antes de que el primer bit  
de la trama de datos haya dado la vuelta
- Permite usar prioridades
  - Mensajes más prioritarios se transmiten antes
- Control distribuido del anillo. Todas las estaciones deben  
acordar qué estación monitoriza el funcionamiento:
  - que el turno vaya pasando
  - que ninguna trama de datos se quede dando vueltas  
indefinidamente
  - que no se «atasque» la prioridad
  - que quien monitoriza el funcionamiento no se bloquee

## 6.2.1 Ejemplo tramas

Tramas de token y datos en Token Ring (IEEE 802.5)

1	1	1								
SD	AC	ED								
1	1	1	2-6	2-6	n	4	1	1		
SD	AC	FC	Dir. destino	Dir. origen	Datos	FCS	ED	FS		

SD: Starting Delimiter

AC: Access Control, PPPTMRRR

(P: prioridad, T: token, M: monitor, R: reserva)

FC: Frame Control, tipo trama (demultiplexor) y bits de control

Direcciones: identificadores MAC origen y destino

FCS: Frame Check Sequence, CRC de 32 bits [p. 54]

ED: Ending Delimiter

FS: Frame Status, ACrrACrr

(A: dir. reconocida, C: trama copiada, r: reservado)

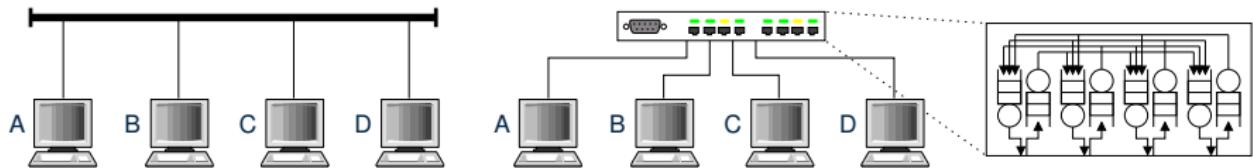
## 7. Comutación en Ethernet

- 7.1. Conmutadores aprendices
- 7.2. LANs virtuales (VLANs)
- 7.3. Protocolo Spanning Tree
- 7.4. Ejercicio resumen commutación
- 7.5. Estándares Ethernet

# 7 Conmutación en Ethernet



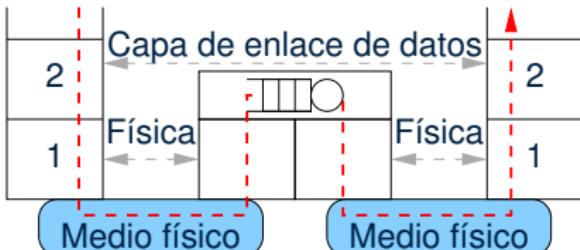
- LANs Ethernet (CSMA/CD) fueron ganando terreno
  - Fácil mantenimiento, buenas prestaciones con baja carga
- Más dispositivos en medio compartido → rendimiento *no escala* → menos prestaciones
- Solución: conmutador (*switch*)  que implementa conjunto de enlaces no compartidos



- Cada nodo está conectado al conmutador con un *cable dedicado* → todos pueden transmitir a la vez → *n* veces más capacidad pico

# 7 Comutación en Ethernet (II)

- Cambio de comunicación directa por medio compartido a indirecta a través de conmutador



- El conmutador es *transparente* para los dispositivos
- El conmutador almacena y reexpide las tramas (*store & forward*) o las reenvía al vuelo (*cut-through*)
- ✗ *Congestión*: pérdida de tramas por falta de memoria en conmutador [Tema 6]
- ✗ *Contención*: latencias adicionales y variables en colas de conmutador
- ✗ El conmutador sólo une *redes del mismo tipo o muy similares*, e.g. AP conecta ethernet y WiFi

# 7 Ejemplo: switch L1.02

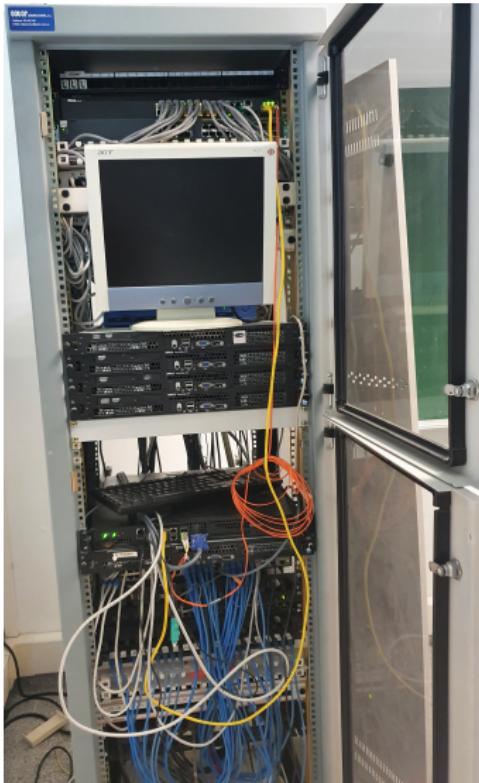


Imagen: rack que aloja equipos de comunicaciones y servidores del L1.02.



## Dell Networking N1524

- 24 puertos RJ-45 10/100/1000Mb con detección automática
- 4 puertos 10GbE SFP+ integrados
- 1 fuente de alimentación integrada (40 W de CA)

Figura: vista frontal y características básicas del switch Dell N1524



Imagen: detalle del cableado del switch Dell N1524

# 7 Ejemplo: switch L1.02 (II)

- ▶ Puerto RJ-45: conector para pares trenzados
- ▶ Puerto SFP+: conector para fibra óptica
- ▶ RJ45 (8P8C)

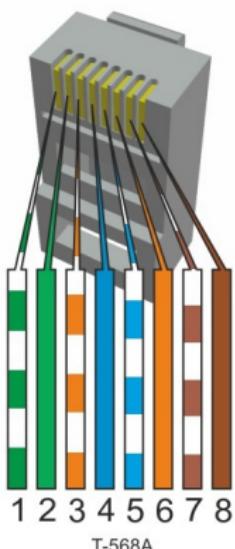


Figura: Conector TIA T568A.

Pin	Description	10base-T	100Base-T	1000Base-T
1	Transmit Data+ or BiDirectional	TX+	TX+	BI_DA+
2	Transmit Data- or BiDirectional	TX-	TX-	BI_DA-
3	Receive Data+ or BiDirectional	RX+	RX+	BI_DB+
4	Not connected or BiDirectional	n/c	n/c	BI_DC+
5	Not connected or BiDirectional	n/c	n/c	BI_DC-
6	Receive Data- or BiDirectional	RX-	RX-	BI_DB-
7	Not connected or BiDirectional	n/c	n/c	BI_DD+
8	Not connected or BiDirectional	n/c	n/c	BI_DD-

 PinoutsGuide.com

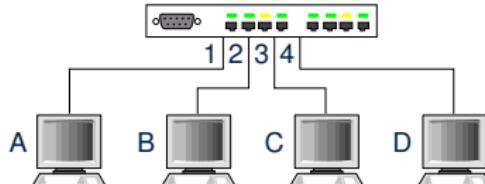
- ▶ SFP+: *small form-factor pluggable*
  - ▶ transceptor: emisor + receptor



Imagen: Transceptor, SFP+, 10GbE, LR, 1310 nm, 10 km. Fuente: [www.dell.com](http://www.dell.com).

## 7.1 Conmutadores aprendices

- Sin conmutadores aprendices, aunque no se comparta el canal, el tráfico es el mismo → *no escala*
- Los conmutadores aprendices ven tramas con identificador MAC de origen *x* que llegan por puerto *y*
- Mantienen tabla con parejas  $\langle x, y \rangle$  que expiran con el tiempo
- Al recibir tramas dirigidas a *x*, se reexpedirán sólo por *y*
  - Se evita tráfico innecesario
  - Se dificulta la monitorización del tráfico de otros
- Tramas con destinos no conocidos o de difusión total (*broadcast*) se reexpiden a todos



Estación	Puerto HW
A	1
B	2
C	3
D	4

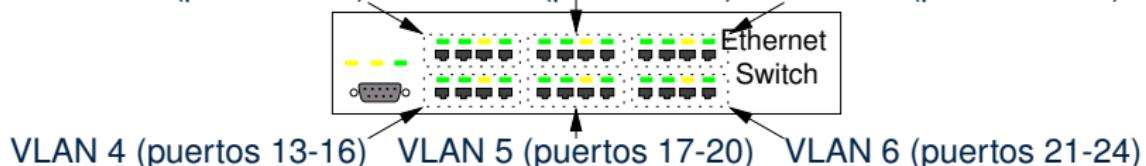
## 7.2 LANs virtuales (VLANs)



- Las tramas de difusión (*broadcast*) han de llegar a todos → no escala para redes muy grandes
- División de LAN en dominios de difusión (VLANs)
  - Tramas de difusión no se reexpiden a otras VLANs
  - Id. VLAN asociado a puertos o id. MAC
  - IEEE 802.1Q: modifica la cabecera ethernet (0x8100) para incluir campo id. VLAN (VLAN trunking)

...	6	6	4	2	...
	Dir. destino	Dir. origen	802.1Q	Tipo	

VLAN 1 (puertos 1-4)      VLAN 2 (puertos 5-8)      VLAN 3 (puertos 9-12)



## 7.3 Protocolo *spanning tree*



- Tolerancia a fallos → conmutadores y enlaces redundantes
- Bucles dan problemas (ejemplo: 1 2 3 2 3 2 3 ...)
  - Tramas duplicadas y dando vueltas indefinidamente:  
Tormenta de difusión (*broadcast storm*)
  - Conmutadores no pueden asociar dir. origen con puerto



- El objetivo del protocolo *spanning tree* (STP) es generar un conjunto de rutas libre de bucles
- Para ello, este protocolo distribuido crea dinámicamente una topología de árbol mediante el bloqueo de ciertos puertos
- Inconveniente: *algunas tramas no siguen el camino óptimo*

## 7.3 Protocolo *spanning tree* (II)



1474

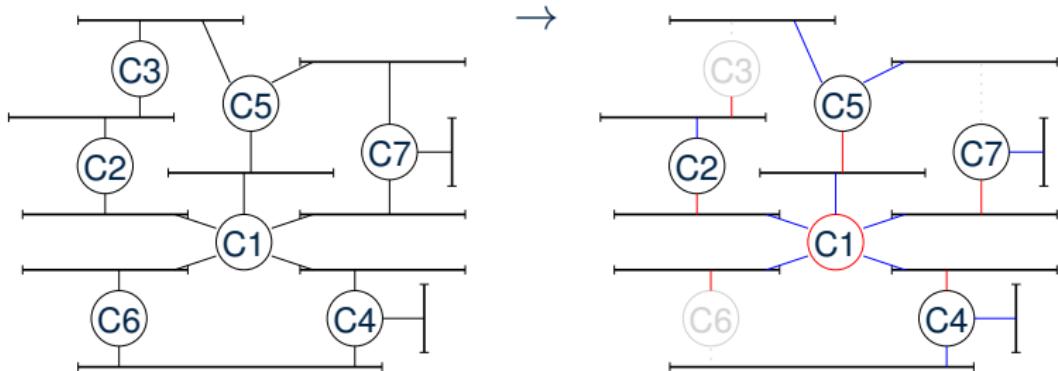
- Basado en algoritmo inventado por *Radia Perlman*<sup>i</sup>
  - 1985 - An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN<sup>i</sup>
  - 1990 - Protocolo estandarizado como 802.1D
  - Actualizaciones en 1998 y 2004
- Fases:
  1. Selección conmutador raíz
  2. Selección camino a conmutador raíz
- Resultado: todos los conmutadores aprenden
  - Cuál es su puerto más cercano al raíz
  - Si para cierto segmento están en el camino más corto hacia raíz

## 7.3 Protocolo *spanning tree* (III)



- Cada commutador tiene un id. obtenido concatenando un valor de prioridad (configurable) y su dir. MAC (única)
- Inicialmente, todos los commutadores creen ser raíz
- Quien cree ser raíz genera mensajes de configuración  $\langle id\_propio, id\_raiz, distancia\_a\_raiz \rangle$
- Quien aprende que no es raíz (recibido  $id\_raiz < id\_propio$ ) no crea más mensajes, pero sí reexpide ( $distancia + 1$ ) los que llegan desde raíz
- Sólo el raíz genera mensajes de configuración. Si no se reciben durante cierto tiempo, se reinicia el proceso

## 7.3 Protocolo *spanning tree* (IV)



- Raíz: conmutador con el menor id. (único)
- Puerto raíz de cada conmut. (RP): el más cercano al conmut. raíz (empates → menor id.)
- Puerto designado de cada segmento de red (DP): el más cercano al conmut. raíz (empates → menor id.)
- ... Comunicadores bloquean sus puertos no-RP y no-DP

## 7.4 Ejercicio resumen conmutación



1474

Asocia cada concepto con su principal aportación:

Concepto	Aportación
Conmutador	a) Reduce el tráfico unicast
Conmut. aprendiz	b) Reduce el tráfico broadcast
VLANs	c) Reduce colisiones
Spanning tree	d) Permite añadir redundancia

## 7.5 Estándares Ethernet

---



- Ethernet está estandarizado por el *Institute for Electrical and Electronics Engineers* (IEEE)
  - IEEE 802.3 es el estándar oficial de Ethernet
  - 1985 - IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications
- Identificadores cortos sistemas: velocidad, señalización, medio físico. Ejemplos:
  - 10BASE-T: 10 Mbps, banda base, dos pares de cable trenzado
  - 100BASE-FX: 100 Mbps, banda base, fibra óptica multimodo
  - 1000BASE-T: 1000 Mbps, banda base, pares trenzados
  - 10GBASE-SR: 10 Gbps, banda base, fibra multimodo de corto alcance

## 7.5 Ejemplo tarjeta red Ethernet

```
lab000:~/ ethtool eno1
Settings for eno1:
Supported ports: [ TP ]
Supported link modes:  10baseT/Half 10baseT/Full
                       100baseT/Half 100baseT/Full
                           1000baseT/Full
Supported pause frame use: No
Supports auto-negotiation: Yes
Supported FEC modes: Not reported
Advertised link modes:   10baseT/Half 10baseT/Full
                        100baseT/Half 100baseT/Full
                           1000baseT/Full
Advertised pause frame use: No
Advertised auto-negotiation: Yes
Advertised FEC modes: Not reported
Speed: 1000Mb/s
Duplex: Full
Port: Twisted Pair
PHYAD: 1
Transceiver: internal
Auto-negotiation: on
MDI-X: off (auto)
Cannot get wake-on-lan settings: Operation not permitted
Current message level: 0x00000007 (7)
                         drv probe link
Link detected: yes
```

## 7.5 Power over Ethernet (PoE)



- Además de enviar datos, el cable Ethernet alimenta al equipo conectado al conmutador
- Usos: alimentación de puntos de acceso, sensores

Pin	Function
1	Tx+
2	Tx-
3	Rx+
4	DC+ PoE
5	DC+ PoE
6	Rx-
7	DC- PoE
8	DC- PoE

# 8 Subcapa control de enlace



- Función: establecer *comunicación fiable*



- *Fiabilidad*: los datos se reciben correctamente, ordenados, sin pérdidas y sin duplicados
- Las tramas recibidas pueden contener errores
  - *Control de errores*
- Se pueden perder tramas
  - *Secuenciación de datos*

Un protocolo ofrece un servicio fiable si detecta tramas:

- Erróneas o perdidas y las retransmite
- Duplicadas y las descarta
- Desordenadas y las ordena

## 9. Control de errores

### 9.1. Detección de errores

Códigos de paridad

Checksum

CRC

### 9.2. Corrección de errores

# 9 Control de errores

---



- Valores típicos de *bit-error-ratio* (BER) (tramas  $\approx 10^4$  bits):
  - Fibra óptica:  $\sim 10^{-12}$
  - Cable de cobre:  $\sim 10^{-9}$  (muy pocos errores)
  - Aire (medio inalámbrico):  $\sim 10^{-5}$  (1/10 tramas erróneas)
- Detección: paridad, checksum, CRC, SHA1, etc.
  - Añade redundancia: alteraciones  $\leftrightarrow$  inconsistencias
  - Sobrecarga «baja»
  - Tasa código: relación entre bits de datos y bits totales
  - *Ninguna detección es 100 % fiable*
- Ante error detectado:
  - *Descartar* trama  $\rightarrow$  no hay fiabilidad (e.g. Ethernet, IP, UDP)
  - *Retransmitir* trama  $\rightarrow$  necesita secuenciación (e.g. TCP)
  - *Corregir* error  $\rightarrow$  sobrecarga elevada (e.g. Hamming) pero interesante ante coste de retransmisión muy alto

## 9.1.1 Códigos de paridad simple

### Bit de paridad

- Cada bloque de  $k$  bits se envía con un bit de paridad:  
 $1010100\ P$
- Par ( $P = \oplus$ ): número par de unos:  $1010100\ 1$
- Impar ( $P = \overline{\oplus}$ ): número impar de unos:  $1010100\ 0$
- Detecta cualquier número impar de bits erróneos
- No corrige errores
- Tasa código:  $\frac{k}{k+1}$
- Tasa redundancia:  $\frac{1}{k+1}$

3 bit data			Message with even parity		Message with odd parity	
A	B	C	Message	Parity	Message	Parity
0	0	0	000	0	000	1
0	0	1	001	1	001	0
0	1	0	010	1	010	0
0	1	1	011	0	011	1
1	0	0	100	1	100	0
1	0	1	101	0	101	1
1	1	0	110	0	110	1
1	1	1	111	1	111	0

## 9.1.1 Códigos de paridad por bloques



- Toma el mensaje como una matriz de  $m$  bits de ancho
- Calcula una nueva fila de  $m$  bits de paridad vertical
- Detecta errores en ráfaga de longitud  $\leq m$ 
  - Longitud de ráfaga de error: número de bits (erróneos o no) entre dos bits erróneos (incluidos en la longitud)
- Código rectangular o código producto: paridad en 2 dimensiones. Caso particular donde cada fila de la matriz tiene a su vez un bit de paridad

1	1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	1	1
0	1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1	1
0	1	0	1	0	1	0	1	1

a. Design of row and column parities

Código rectangular de detección de errores.

Fuente: <http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/801-simple-parity-check-code.html>

## 9.1.2 Checksum

---

- Redundancia: bits obtenidos al sumar palabras de datos
- El receptor verifica si la suma es correcta
  - E.g. emisor:  $3 + 7 + 5 + 1 + 3 = 19$  (checksum)
  - Comprobación:  $3 + 7 + 5 + 1 + 3 - 19 = 0?$  ( $\neq 0$ : error)
- Toma el mensaje como secuencia de números de  $m$  bits
- Calcula una nueva fila de  $m$  bits como la suma de la secuencia de números
  - Cualquier tipo de suma sirve, aunque distintos tipos de suma tendrán distintas propiedades de detección
  - Paridad por bloques: caso particular de checksum con  $\oplus$  bit a bit como operación de suma
  - TCP/IP: suma de enteros de 16 bits en complemento a 1 con inversión de bits en el resultado
- Detecta errores en ráfaga de longitud  $\leq m$

## 9.1.3 Cyclic Redundancy Check (CRC)



- Suma tiene limitaciones para detectar errores:

	mensaje			checksum
Original	6	23	4	33
Recibido	8	20	5	33

- Bits de redundancia obtenidos mediante división permiten detectar más errores
- Ejemplo: letra DNI = número DNI módulo 23  
(T R W A G M Y F P D X B N J Z S Q V H L C K E)
- CRC: resto de la división entera entre mensaje y un divisor
- Aritmética polinomial
  - Operaciones módulo 2:  $+ \equiv - \equiv \oplus$
  - Un divisor cabe en un dividendo si éste tiene tantos bits como el divisor:  $10 \div 11 = 1, \text{resto } 1$
  - Un polinomio es de grado  $n$  si tiene  $n + 1$  bits y su bit de más peso es 1

## 9.1.3 Cyclic Redundancy Check (CRC) (II)



- Emisor: dado un mensaje  $M$  y un polinomio generador  $G$  de grado  $n$ , genera un polinomio  $T$  divisible por  $G$ 
  - $CRC = (M \times 2^n) \text{ mód } G$  (añade  $n$  0s, divide y coge resto)
  - $T = M \times 2^n + CRC$  (concatena  $M$  y  $CRC$ )
- Receptor:  $R = T + E$ ,  
asume mensaje correcto si:  $R \text{ mód } G = 0$
- Detecta:
  - Errores en ráfaga de longitud  $\leq n$
  - Número impar de errores si  $G$  es múltiplo de  $x + 1$   
(ningún polinomio  $E$  con un número impar de términos es múltiplo de  $x + 1$ )
- Algunos  $G$  populares:
  - 16 bits: X25: (16,12,5,0), CRC-16: (16,15,2,0)
  - 32 bits: Ethernet: (32,26,23,22,16,12,11,10,8,7,5,4,2,1,0)
- Más información en Tanembaum y en este [enlace](#).

### 9.1.3 Cyclic Redundancy Check (CRC) (III)



*M*: 1101011011

$$(x^9 + x^8 + x^6 + x^4 + x^3 + x + 1)$$

$$G: 10011 \ (x^4 + x + 1)$$

*n*: 4

*CRC:* 1110 ( $x^3 + x^2 + x$ )

*T*: 11010110111110

$$(x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + x)$$



## 9.2 Corrección de errores

---



**Forward Error Correction (FEC):** envío de información con suficiente redundancia para que el receptor pueda corregir errores de transmisión

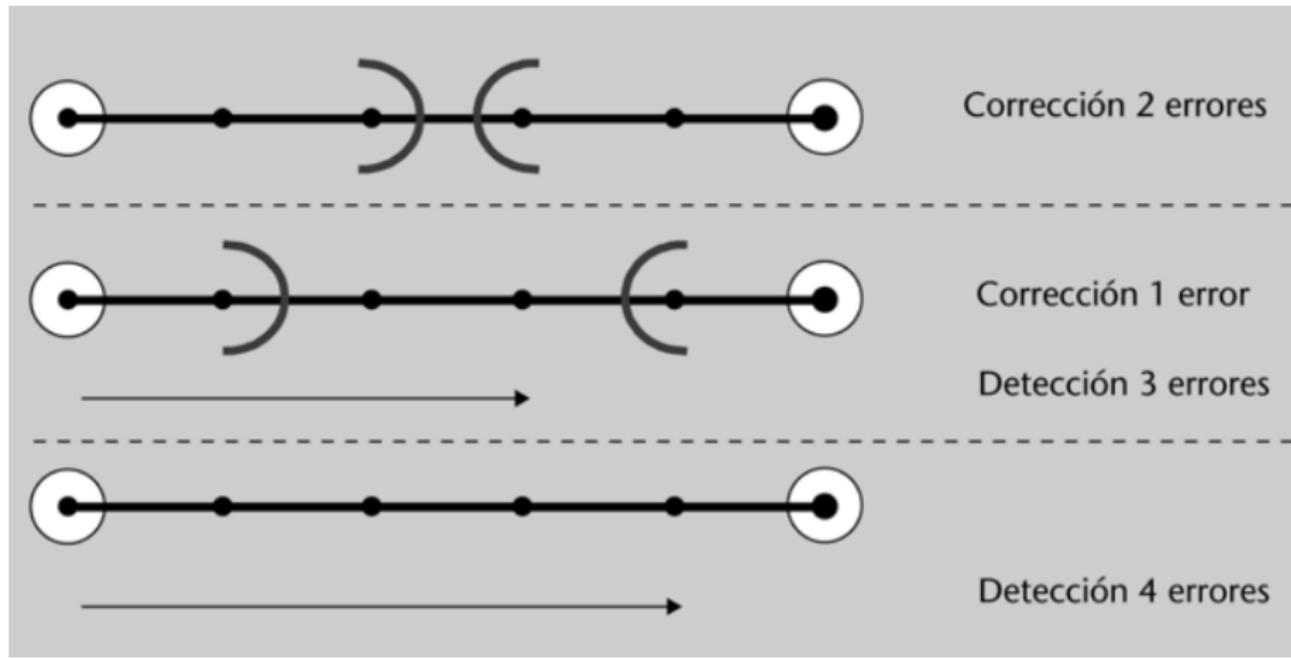
**Código de bloque:** asigna una palabra o vector de  $n$  bits a cada uno de los  $2^k$  posibles mensajes de  $k$  bits

Mensaje ( $k=2$ bits)	Palabra código ( $n=10$ bits)
00	0000000000
01	0000011111
10	1111100000
11	1111111111

**Distancia de Hamming:** número mínimo de bits que deben cambiar en una palabra código para convertirse en otra

## 9.2 Corrección de errores (II)

- Posibilidades de corrección/detección de errores con un código de distancia mínima 5



Fuente: Codificación de canal I: introducción y códigos de bloque. Francesc Tarrés y Margarita Cabrera.

## 9.2 Corrección de errores (III)

- La corrección de errores se basa en establecer palabras código lo suficientemente distantes como para que:
  1. Si hay error, se reciba una palabra código no válida
  2. Al recibir una palabra código no válida, la más cercana en distancia Hamming se considera la transmitida
- Con distancia  $d$  se pueden corregir  $\lfloor(d - 1)/2\rfloor$  bits
- Ejemplo: 0000000000, 0000011111, 1111100000, 1111111111
  - Distancia de Hamming: 5
  - Puede corregir 2 bits, e.g. 0000010011 → 0000011111
- Ejemplos: códigos Hamming, códigos Golay (NASA Voyager 1 y 2), etc.



## 10. Secuenciación de datos

10.1. Stop & wait

10.2. Ventana deslizante

Go-Back-N

Selective Repeat

Números de secuencia y tamaños de ventana

10.3. Ejemplo: HDLC

10.4. Ejercicios resumen de secuenciación

# 10 Secuenciación de datos

Contexto: protocolos que ofrecen servicio fiable deben recuperar tramas erróneas o perdidas.

Generalmente, esta funcionalidad se implementa mediante algoritmos *ARQ* (*Automatic Repeat reQuest*), que se basan en:

- *Acuse de recibo (ACK)*: trama de control enviada por el receptor de una trama para confirmar su recepción
  - *Piggyback*: ACK en trama de datos
- *Temporizador*: tiempo de espera antes de retransmitir una trama sin ACK

Algoritmos ARQ se usan en protocolos de

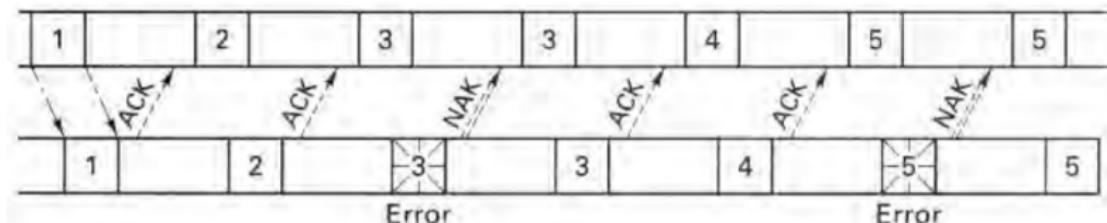
- Nivel de enlace: enlace lógico punto-a-punto
- Nivel de transporte: enlace lógico extremo-a-extremo

# 10 Secuenciación de datos (II)

Algoritmos ARQ:

- *Parada y espera (stop & wait)*: sencillo
- *Ventana deslizante*
  - *Vuelta atrás (go-back-n)*
  - *Repetición selectiva (selective repeat)*: más eficiente

## 10.1 Stop & wait



Fuente: Sklar. Digital Comunications

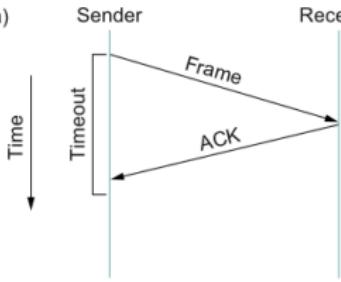
- Emisor envía trama  $i$ , inicia temporizador y espera ACK
    - Si recibe ACK de trama  $i \rightarrow$  parar temporizador e iniciar transmisión de trama  $i + 1$
    - Si vence el temporizador o recibe NAK  $\rightarrow$  reenviar trama
  - Receptor espera trama con identificador  $i$ 
    - Trama  $i$  correcta  $\rightarrow$  enviar ACK de  $i$  y esperar trama  $i + 1$
    - Trama  $i$  errónea  $\rightarrow$  nada o enviar NAK de  $i$  (negative ACK)
  - Puede funcionar con half-duplex

✍ Para transmitir un número ilimitado de tramas, ¿cuántos identificadores de trama se necesitan? ¿Cuántos bits?

# 10.1 Stop & wait (II)

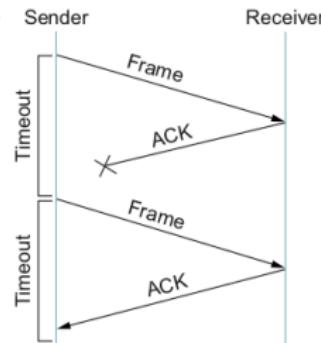
► 4 escenarios:

(a) Sender



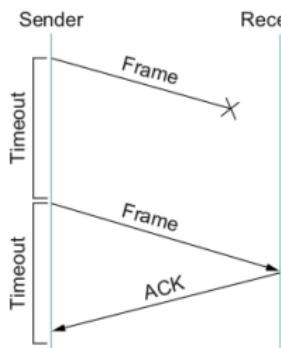
Receiver

(c) Sender

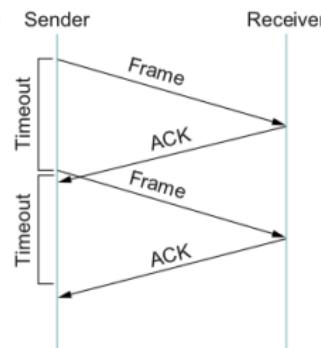


Receiver

(b) Sender



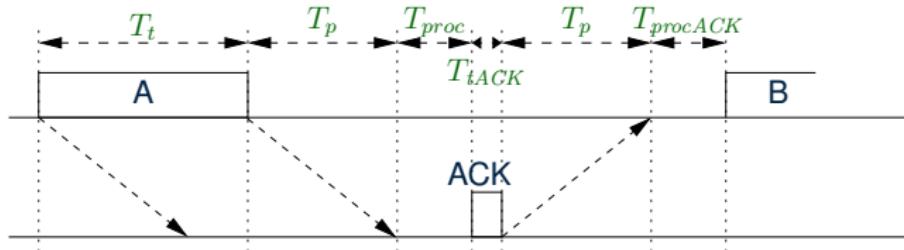
(d) Sender



# 10.1 Stop & wait (III)

Tiempo de ida y vuelta / *Round Trip Time* (RTT): tiempo necesario para enviar una trama y recibir su confirmación (segundos, s)

- RTT sin errores: transmisión  $T_t$  + propagación  $T_p$  + procesamiento  $T_{proc}$  de trama y ACK
- $RTT = T_t + T_p + T_{proc} + T_{tACK} + T_p + T_{procACK} = L_t/V_t + D/V_p + T_{proc} + L_{ACK}/V_t + D/V_p + T_{procACK}$



- Habitualmente:  $T_{proc} \approx T_{procACK} \approx 0$ ,  $T_t \gg T_{tACK} \approx 0$ :  
 $RTT = T_t + 2 \cdot T_p$

## 10.1 Stop & wait (IV)

- Utilización del enlace:

$$U = 100 \cdot \frac{T_t}{RTT} = 100 \cdot \frac{1}{1 + 2 \cdot T_p/T_t} \leq 100\%$$

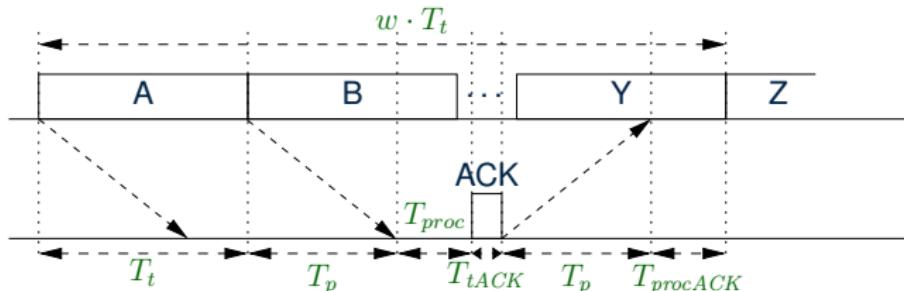
- ¿Y si viajan tramas de datos y ACK en ambos sentidos?

☞ Calcula la utilización de los siguientes enlaces:

- $T_t = 1 \text{ ms}, T_p = 5 \mu\text{s}$
- $T_t = 1 \text{ ms}, T_p = 50 \mu\text{s}$
- $T_t = 1 \text{ ms}, T_p = 500 \mu\text{s}$
- $T_t = 1 \text{ ms}, T_p = 5 \text{ ms}$
- $T_t = 1 \text{ ms}, T_p = 50 \text{ ms}$

## 10.2 Ventana deslizante

- Stop & wait es poco eficiente cuando no se cumple  $T_t \gg T_p$
- Ventana deslizante envía  $w$  tramas sin esperar el primer ACK:



- Utilización del enlace:

$$U(\%) = \begin{cases} 100 & \text{si } w \cdot T_t \geq RTT \rightarrow w \geq 1 + 2 \cdot \frac{T_p}{T_t} \\ 100 \cdot \frac{w \cdot T_t}{RTT} & \text{en caso contrario} \end{cases}$$

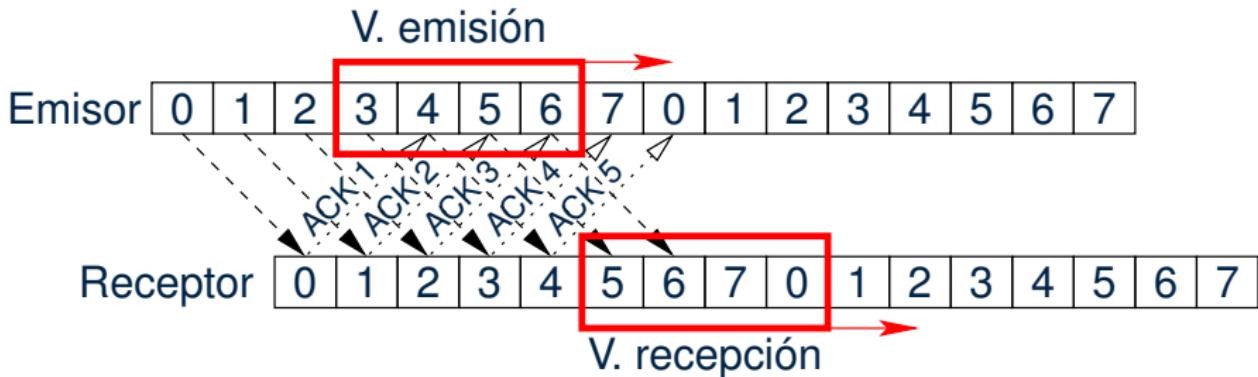
- Requiere canal full-duplex

## 10.2 Ventana deslizante (II)

✍ Calcula el mínimo tamaño de ventana deslizante que permite utilizar el 100 % de los siguientes enlaces:

- ▶  $T_t = 1 \text{ ms}, T_p = 5 \mu\text{s}$
- ▶  $T_t = 1 \text{ ms}, T_p = 50 \mu\text{s}$
- ▶  $T_t = 1 \text{ ms}, T_p = 500 \mu\text{s}$
- ▶  $T_t = 1 \text{ ms}, T_p = 5 \text{ ms}$
- ▶  $T_t = 1 \text{ ms}, T_p = 50 \text{ ms}$

## 10.2 Ventana deslizante (III)

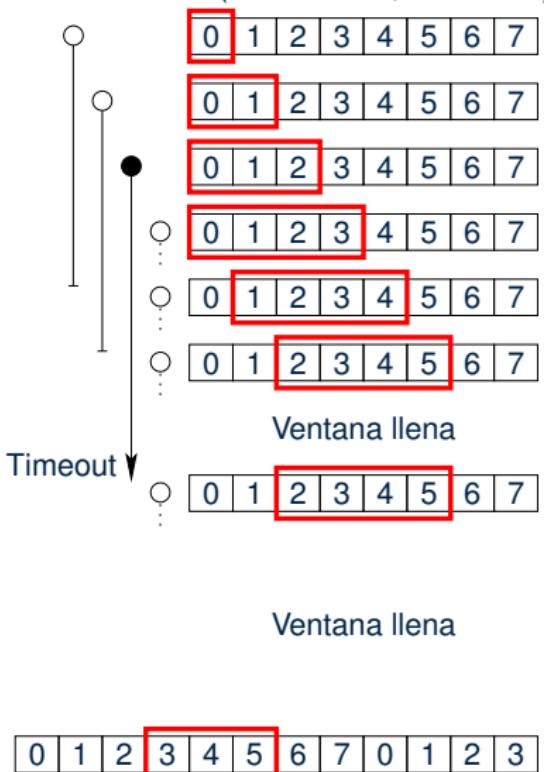


- El emisor asigna a cada trama un *número de secuencia*
- El emisor mantiene una *ventana de emisión* con las  $w$  tramas pendientes de confirmación
- El receptor mantiene una *ventana de recepción* para las  $w_r$  tramas que está dispuesto a aceptar
- $ACK_i$  indica que se ha recibido la trama  $i - 1$  y se espera la trama con número de secuencia  $i$

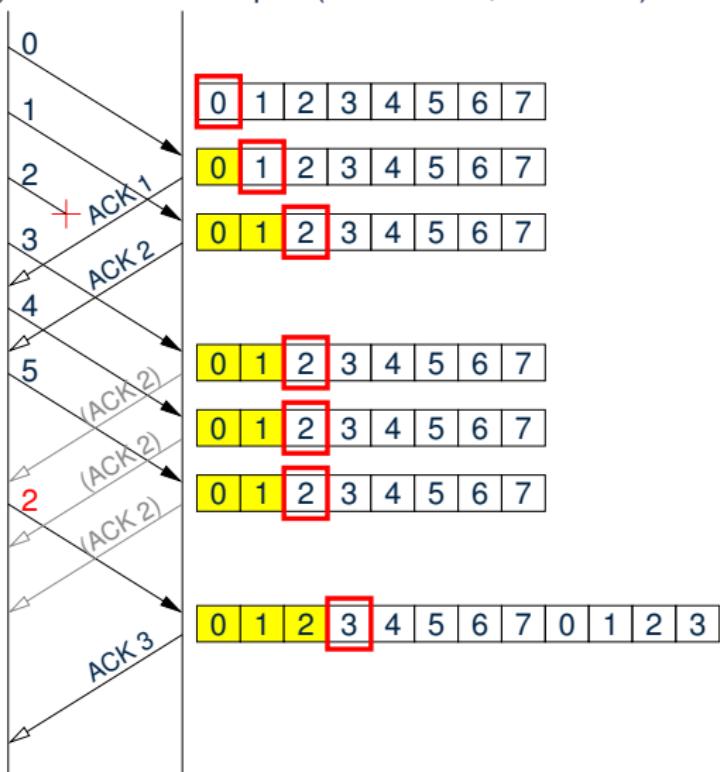
### 10.2.1 Go-Back-N



Emisor (Núm. sec. 8, ventana 4)



### Receptor (Núm. sec. 8, ventana 1)



## 10.2.2 Selective Repeat

Emisor (Núm. sec. 8, ventana 4)

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Ventana llena

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Timeout

0

1

2

3

4

5

6

7

+ ACK1

ACK2

ACK2

ACK2

ACK2

ACK2

ACK6

Receptor (Núm. sec. 8, ventana 4)

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

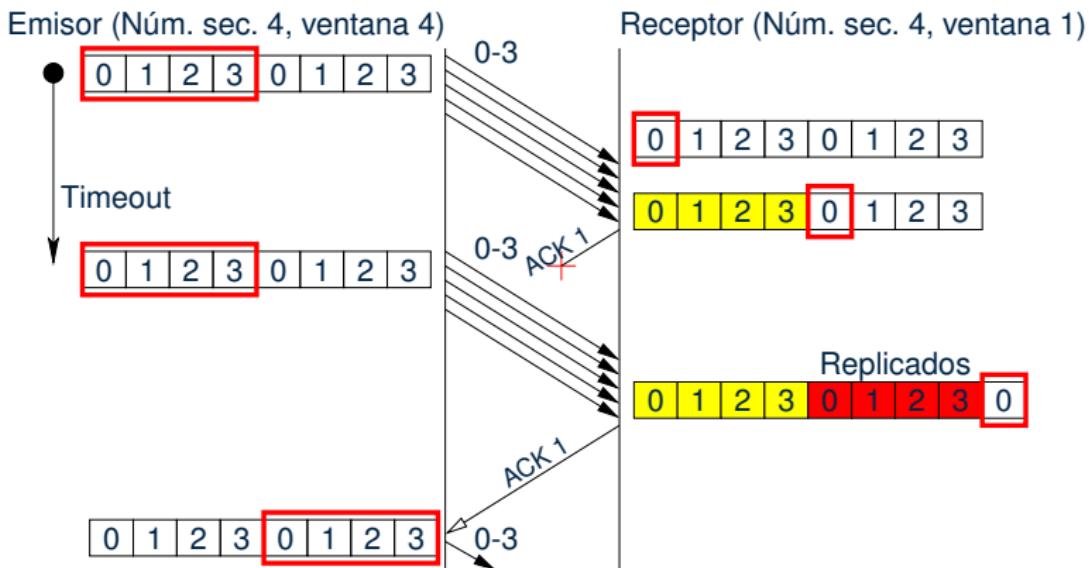
0	1	2	3	4	5	6	7	0	1	2	3
---	---	---	---	---	---	---	---	---	---	---	---

6-1

0	1	2	3	4	5	6	7	0	1	2	3
---	---	---	---	---	---	---	---	---	---	---	---

## 10.2.3 Núm. secuencia y tamaño vent.

- Se necesitan  $NS$  números de secuencia:  $NS \geq w + w_r$
- Go-back-n:  $NS = w + 1$ ,  $n = \lceil \log_2(w + 1) \rceil$  bits
- Contraejemplo:



## 10.2.3 Núm. secuencia y tamaño vent. (II)

- Se necesitan  $NS$  números de secuencia:  $NS \geq w + w_r$
- *Selective Repeat*:  $NS = w + w_r$ ,  $n = \lceil \log_2(w + w_r) \rceil$  bits
- Contraejemplo:

Emisor (Núm. sec. 8, ventana 5)



Timeout



Receptor (Núm. sec. 8, ventana 5)



Sustituidos



## 10.3 Ejemplo: HDLC

ISO High-Level Data Link Control (6 ó 7 bytes + datos)

1	1	1 ó 2	n	2	1
01111110	Dirección	Control	Datos	FCS	01111110

**Dirección:** id. secundaria (no usada en punto-a-punto)

**Control (8 bits):** número secuencia (3 bits), núm. sec. esperada (3 bits), 2 bits control

**Control (16 bits):** número secuencia (7 bits), núm. sec. esperada (7 bits), 2 bits control

**FCS:** CRC de 16 bits

## 10.4 Ejercicio resumen de secuenciación



☞ Completa la siguiente tabla de tamaños de ventana.

	V. Emisión	V. Recepción
Stop&Wait		
Go-Back-n		1
Selective Repeat		$w_r$

☞ Queremos implementar un protocolo para la secuenciación de datos en capa de enlace con 3 bits para el número de secuencia. ¿Es posible un protocolo libre de fallos en los siguientes casos?

- ▶ Ventana de emisión 2 y ventana de recepción 6
- ▶ Ventana de emisión 1 y de recepción 8
- ▶ Ventana de emisión 5 y de recepción 5