Bloque III
Redes de Computadores

Sistemas Telemáticos
2010-2011

Rafael Sebastian
Departamento de Informática
Escuela Técnica Superior de Ingenierías
Universitat de València
Adaptado de Rogelio Montañana

Índice de contenido

Objetivos sección

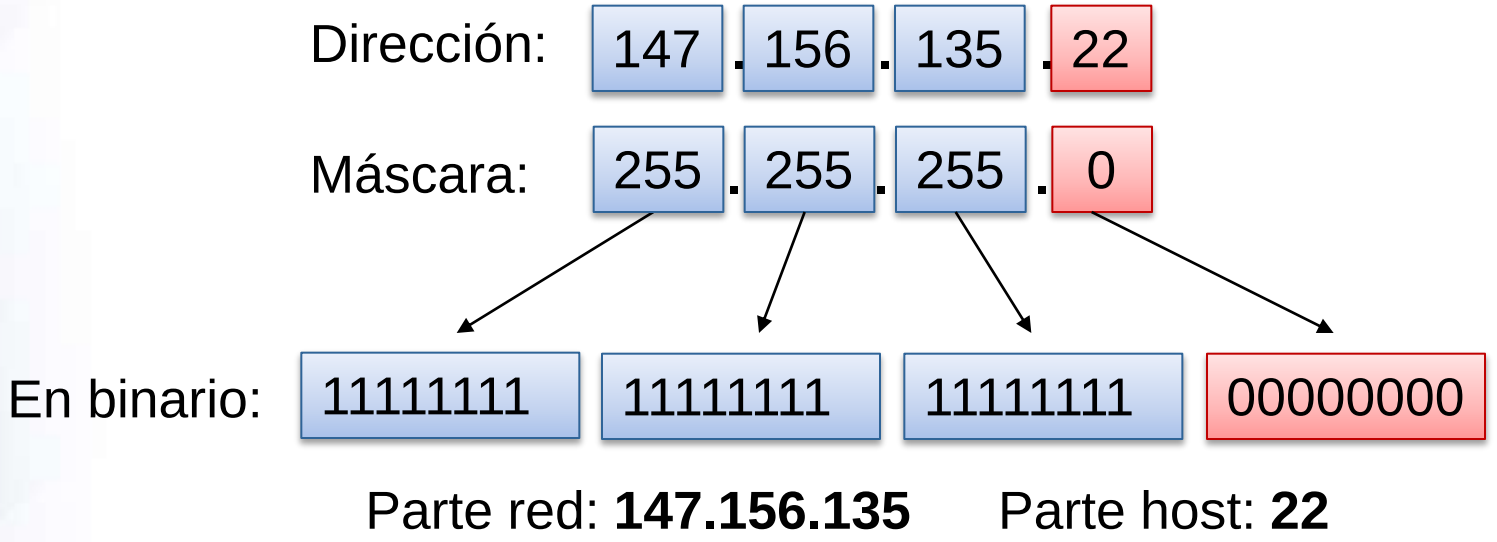
- ☑ Saber diseñar una red usando subredes y máscaras de tamaño variable
- ☑ Entender el funcionamiento de los mecanismos de TCP
- ☑ Asimilar el funcionamiento de protocolos de ventana deslizante
- ☑ Diferenciar entre TCP y UDP

Protocolo TCP/IP

- **Subredes**
- Protocolo UDP
- Protocolo TCP

Dirección IP y máscara

- Cuando asignamos la dirección IP a una tarjeta de red le tenemos que indicar la máscara que estamos utilizando. Ejemplo:



Direcciones IP con clase

Clase A	Red	Host		
Octeto	1	2	3	4

Clase B	Red		Host	
Octeto	1	2	3	4

Clase C	Red			Host
Octeto	1	2	3	4

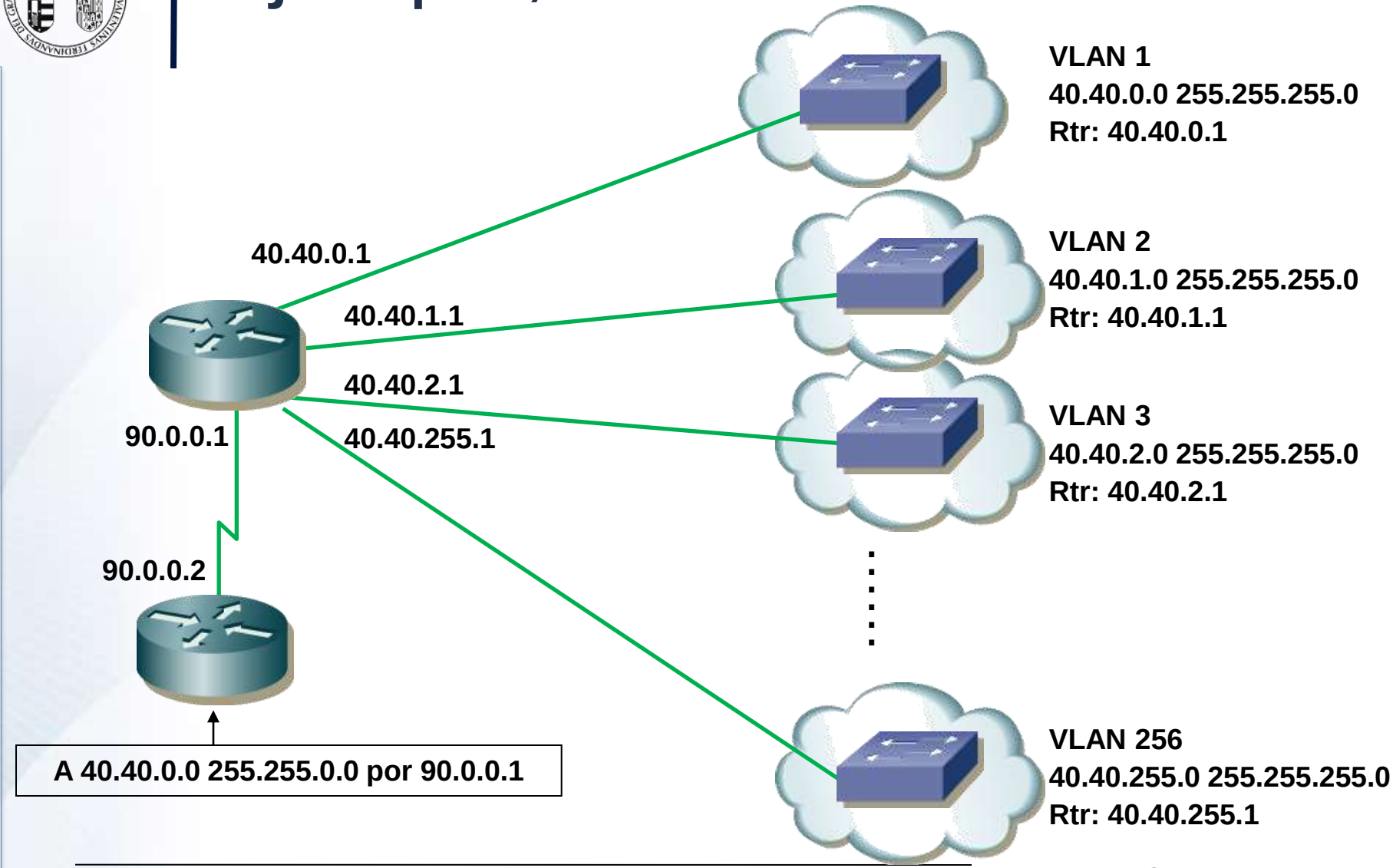
Clase D	Host			
Octeto	1	2	3	4

Subredes

- Red de organización formada por varias redes:
  - Conveniente partir de una red grande que dividimos en trozos más pequeños llamados ***subredes***
- Ejemplo
  - Empresa X utiliza la red 40.40.0.0 255.255.0.0 (desde 40.40.0.0 hasta 40.40.255.255) en una LAN enorme
  - Para reducir el tráfico broadcast decide dividirla formando VLANs, ninguna de las cuales tendrá más de 256 ordenadores. Las subredes podrían ser:

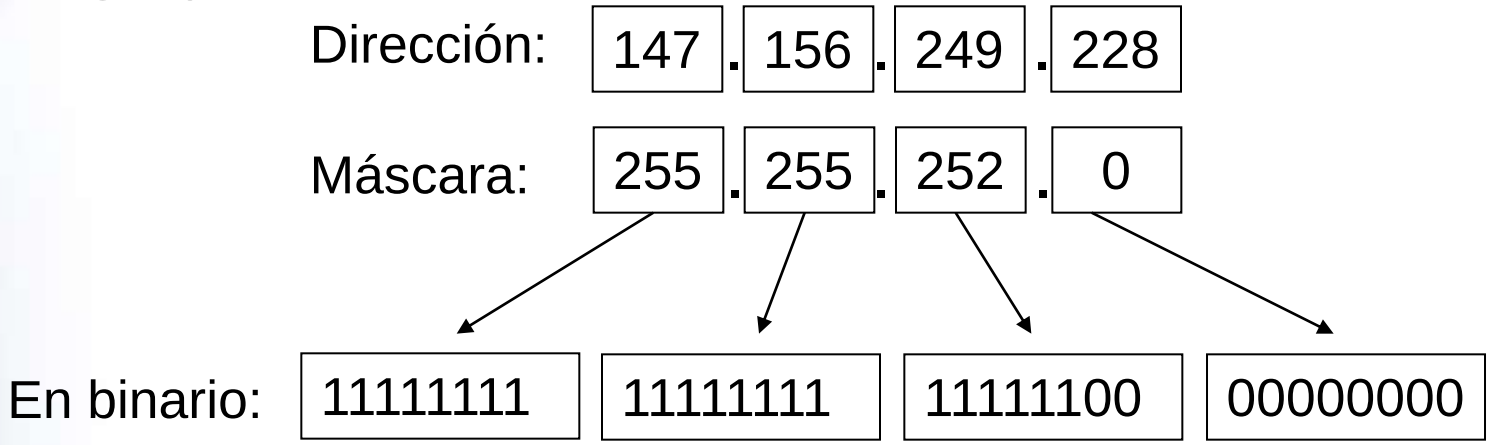
VLAN	Subred	Máscara	Rango
1	40.40.0.0	255.255.255.0	40.40.0.0 - 40.40.0.255
2	40.40.1.0	255.255.255.0	40.40.1.0 – 40.40.1.255
3	40.40.2.0	255.255.255.0	40.40.2.0 – 40.40.2.255
256	40.40.255.0	255.255.255.0	40.40.255.0 – 40.40.255.255

Ejemplo, uso subredes



Máscaras que no son múltiplo de 8

- Las máscaras no siempre son de 8, 16 o 24 bits. En estos casos la separación de la parte red y la parte host no es tan evidente, aunque el mecanismo es el mismo:



Parte red: **22 bits** Parte host: **10 bits**

Esta red tiene 1024 direcciones. Rango: 147.156.248.0 – 147.156.251.255
La primera y la última no son utilizables

Posibles valores de las máscaras

- En las máscaras los bits a 1 siempre han de estar contiguos empezando por la izquierda. No está permitida por ejemplo la máscara 255.255.0.255
- Los únicos valores que pueden aparecer en cualquier máscara

Bits de máscara (n)	Binario	Decimal
0	00000000	0
1	10000000	0 + 128 = 128
2	11000000	128 + 64 = 192
3	11100000	192 + 32 = 224
4	11110000	224 + 16 = 240
5	11111000	240 + 8 = 248
6	11111100	248 + 4 = 252
7	11111110	252 + 2 = 254
8	11111111	254 + 1 = 255

Máscara (n) = máscara (n-1) + $128/2^{n-1}$

Máscaras. Notación concisa

- La máscara se puede indicar en su longitud en bits (entre 0 y 32). Notación mucho más concisa al indicar direcciones de interfaces y rutas

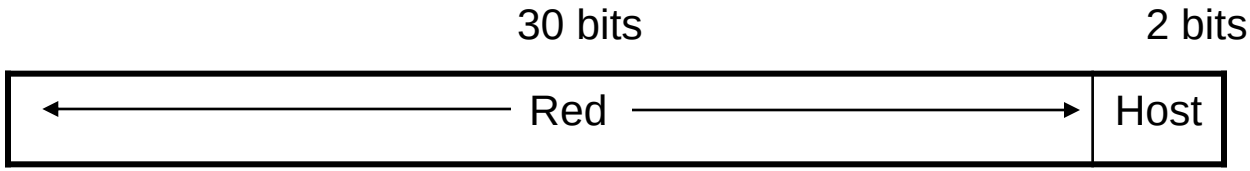
La interfaz “40.40.0.1 255.255.255.0” se convierte en “40.40.0.1/24”

La ruta “A 20.0.0.0 255.0.0.0 por 90.0.0.2” se convierte en “A 20.0.0.0/8 por 90.0.0.2”

Máscara	Bits	Máscara	Bits	Máscara	Bits	Máscara	Bits
0.0.0.0	0						
128.0.0.0	1	255.128.0.0	9	255.255.128.0	17	255.255.255.128	25
192.0.0.0	2	255.192.0.0	10	255.255.192.0	18	255.255.255.192	26
224.0.0.0	3	255.224.0.0	11	255.255.224.0	19	255.255.255.224	27
240.0.0.0	4	255.240.0.0	12	255.255.240.0	20	255.255.255.240	28
248.0.0.0	5	255.248.0.0	13	255.255.248.0	21	255.255.255.248	29
252.0.0.0	6	255.252.0.0	14	255.255.252.0	22	255.255.255.252	30
254.0.0.0	7	255.254.0.0	15	255.255.254.0	23	255.255.255.254	31
255.0.0.0	8	255.255.0.0	16	255.255.255.0	24	255.255.255.255	32

‘Mini-Redes’

La red más pequeña que podemos hacer es la de máscara de 30 bits:



Máscara: 11111111 . 11111111 . 11111111 . 111111 00

255 . 255 . 255 . 252

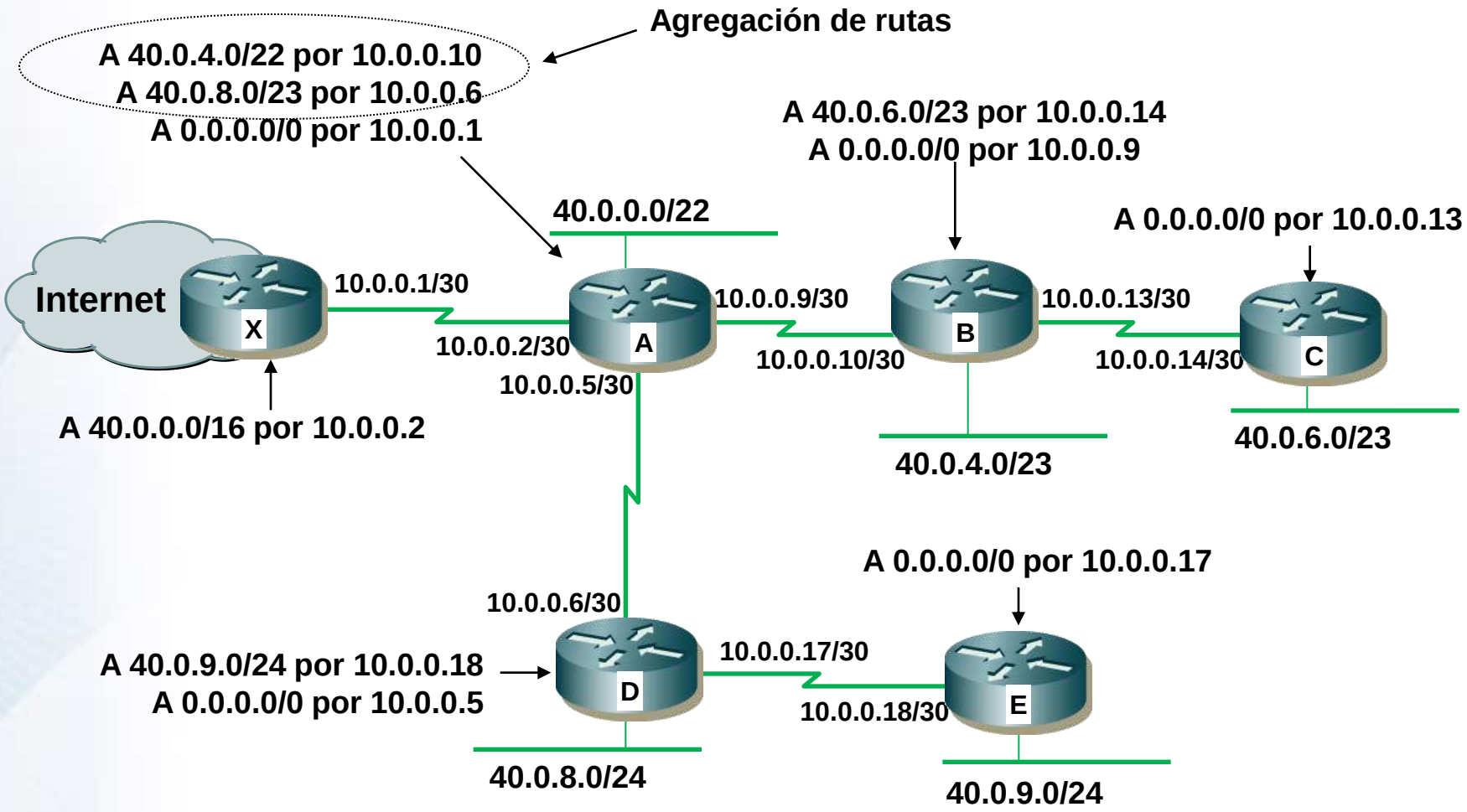
En este caso obtenemos cuatro direcciones, de las cuales solo podemos usar dos. Estas redes se suelen utilizar en enlaces punto a punto ya que en este caso solo se necesitan dos direcciones. Ejemplos:

Red	Rango	Broadcast	Direcciones utilizables
90.0.0.0/30	90.0.0.0 a 90.0.0.3	90.0.0.3	90.0.0.1 y 90.0.0.2
90.0.0.4/30	90.0.0.4 a 90.0.0.7	90.0.0.7	90.0.0.5 y 90.0.0.6
90.0.0.8/30	90.0.0.8 a 90.0.0.11	90.0.0.11	90.0.0.9 y 90.0.0.10

Máscaras de tamaño variable

- A menudo interesa dividir una red en subredes de diferentes tamaños
- Para esto se utilizan máscaras de tamaño variable, es decir la división red/host no es igual en todas las subredes
- Aunque las subredes pueden tener diferente tamaño no pueden solaparse (habría direcciones duplicadas)
- La visión que tenemos de las subredes puede variar. Por ejemplo lo que en un sitio de la red se ve como una subred /22 (1024 direcciones) puede dividirse en varias /24 (256 direcciones) cuando nos acercamos

Subredes con máscara de long. variable (VLSM)



División de una red clase C

Dirección de red 192.168.10.0 clase C

11000000.10101000.00001010.00000000
N . N . N . H

11000000.10101000.00001010.00000000
N . N . N . sN H

En este ejemplo se han asignado tres bits para designar la subred.

192.168.10.**1**	**00000001**
192.168.10.**2**	**00000010**
192.168.10.**253**	**11111101**
192.168.10.**254**	**11111110**

Sin uso de subredes
Disponibles 254 Direcciones

NOTA:

1 0 1 0 1 0 0 0

$2^7 + 0 + 2^5 + 0 + 2^3 + 0 + 0 + 0$

128+32+8 = 168

División de una red clase C

Dirección de red 192.168.10.0 clase C

11000000.10101000.00001010.00000000
N . N . N . H

11000000.10101000.00001010.00000000
N . N . N . sN H

En este ejemplo se han asignado tres bits para designar la subred.

Subred 1

192.168.10.1	00000001
192.168.10.30	00011110

Subred 2

192.168.10.33	00100001
192.168.10.62	00111110

Subred 7

192.168.10.193	11000001
192.168.10.222	11011110

Subred 8

192.168.10.225	11100001
192.168.10.254	11111110

Valor de la máscara de subred

192.168.10 .1 → **11000000.10101000.00001010.00000001**
255.255.255.0 **11111111.11111111.11111111.00000000** = /24

192.168.10 .1 → **11000000.10101000.00001010.00000001**
255.255.255.128 **11111111.11111111.11111111.10000000** = /25

192.168.10 .1 → **11000000.10101000.00001010.00000001**
255.255.255.192 **11111111.11111111.11111111.11000000** = /26

192.168.10 .1 → **11000000.10101000.00001010.00000001**
255.255.255.224 **11111111.11111111.11111111.11100000** = /27

Formato de barra diagonal	/25	/26	/27	/28	/29	/30	N/A	N/A
Máscara	128	192	224	240	248	252	254	255
Bits pedidos	1	2	3	4	5	6	7	8
Valor	128	64	32	16	8	4	2	1

Resumen subredes clase C

Formato de barra diagonal	/25	/26	/27	/28	/29	/30	No es aplicable	No es aplicable
Máscara	128	192	224	240	248	252	254	255
Bits pedidos	1	2	3	4	5	6	7	8
Valor	128	64	32	16	8	4	2	1
Subredes totales		4	8	16	32	64		
Subredes que se pueden utilizar		2	6	14	30	62		
Hosts totales		64	32	16	8	4		
Hosts que se pueden utilizar		62	30	14	6	2		

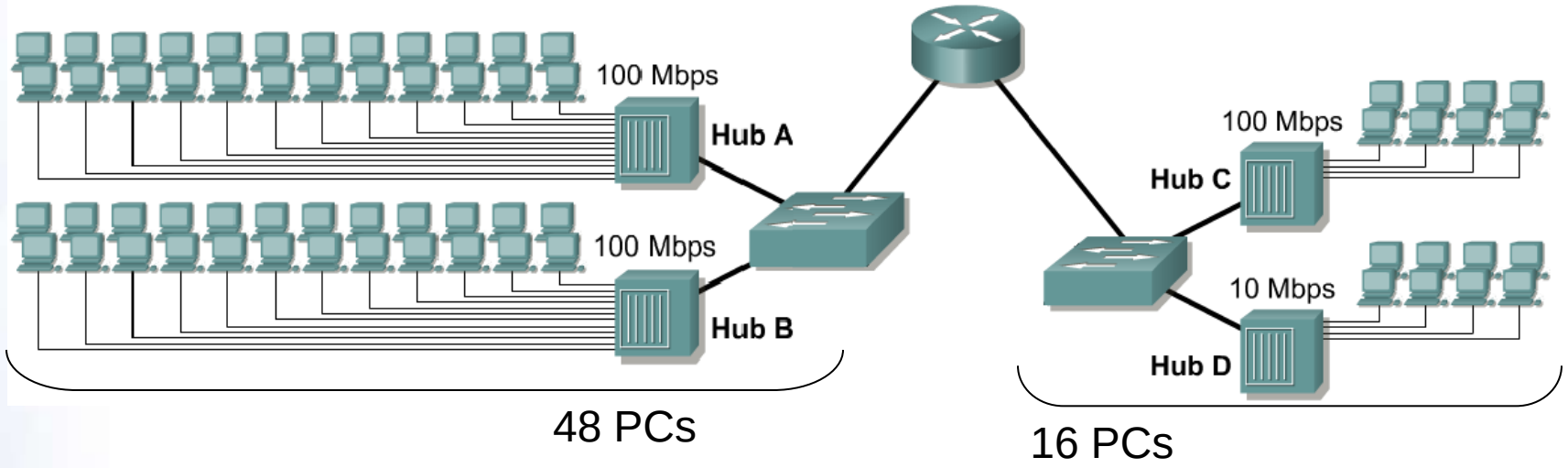
Relación subredes/Host en clase C

Subred N	ID de subred	Rango de hos	ID de broadcast
0	192.168.10.0	.1--.30	192.168.10.31
1	192.168.10.32	.33--.62	192.168.10.63
2	192.168.10.64	.65--.94	192.168.10.95
3	192.168.10.96	.97--.126	192.168.10.127
4	192.168.10.128	.129--.158	192.168.10.159
5	192.168.10.160	.161--.190	192.168.10.191
6	192.168.10.192	.193--.222	192.168.10.223
7	192.168.10.224	.225--.254	192.168.10.255

(Máscara de subred de 3 bits = .224)
(Campo hosts 5 bits = 32 direcciones-2 = 30)

Ejemplo creación de subredes

Red empresa: 192.168.10.0/24



Subred de al menos 48 PCs de tamaño:

Tamaño campo HOST: 6 bits

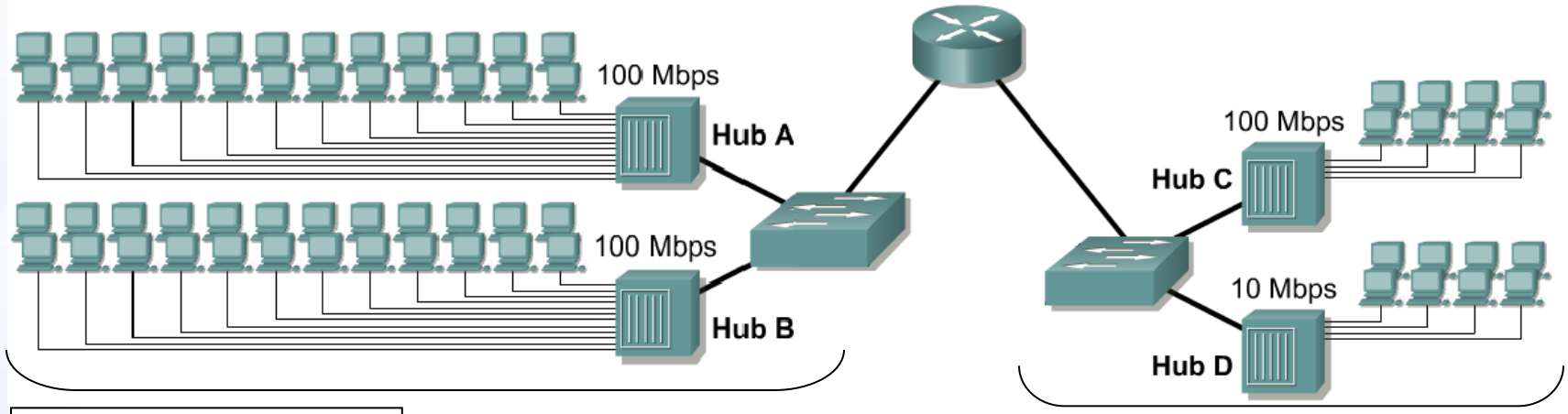
$2^0+2^1+2^2+2^3+2^4+2^5 = 63$ Pcs

Tamaño campo SUBRED: 2 bits

192.168.10.0/26
192.168.10.64/26
192.168.10.128/26
192.168.10.192/26

Ejemplo creación de subredes

Red empresa: 192.168.10.0/24



192.168.10.0/26
192.168.10.64/26
192.168.10.128/26
192.168.10.192/26

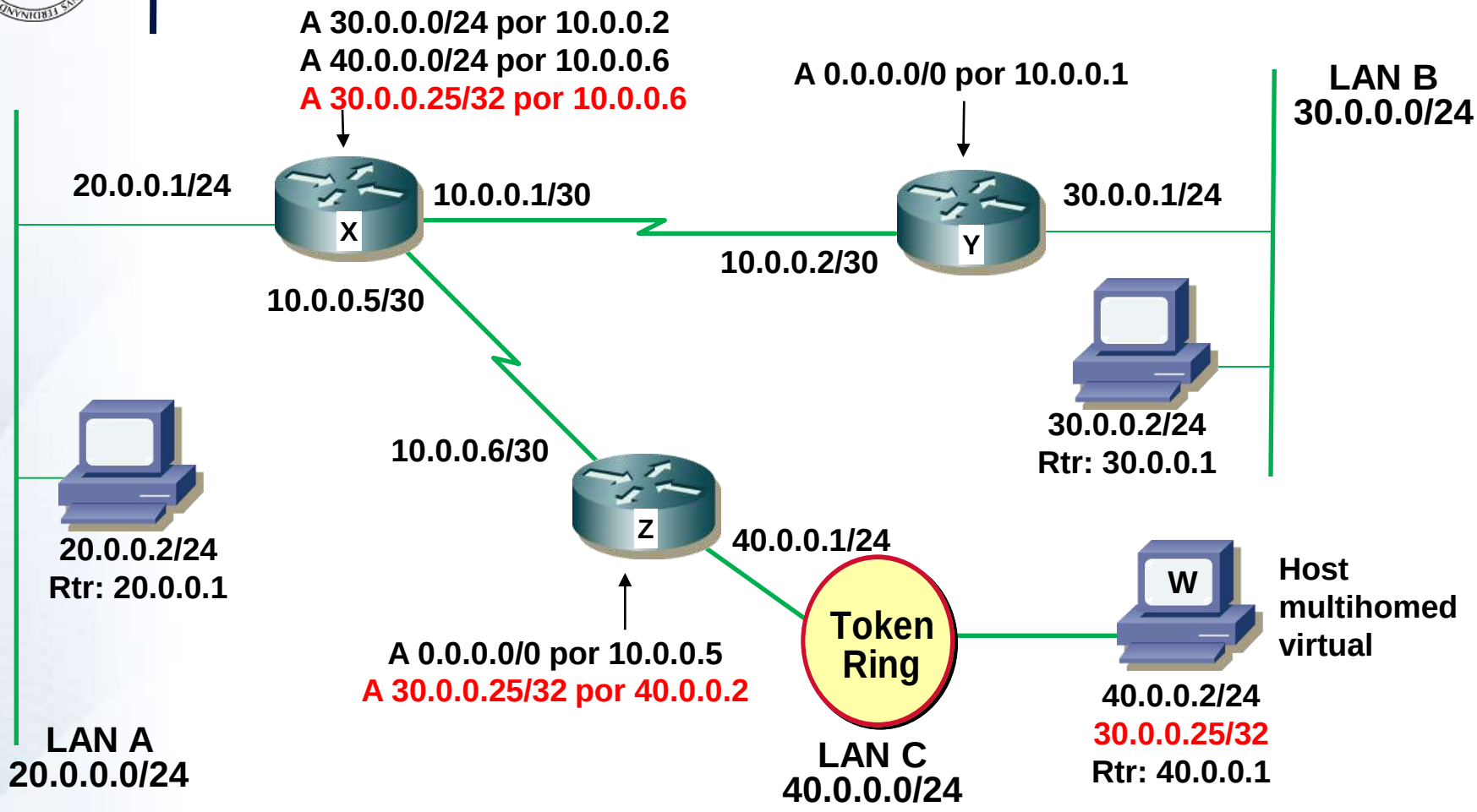
192.168.10.1/26
192.168.10.2/26
192.168.10.3/26
192.168.10.4/26
…
192.168.10.48/26

192.168.10.65/26
192.168.10.66/26
192.168.10.67/26
192.168.10.68/26
…
192.168.10.80/26

Rutas host

- La ruta por defecto (“A 0.0.0.0/0 por dir-IP”) es la ruta más general posible, pues la máscara de 0 bits abarca todas las direcciones. Esta ruta solo se aplica como último recurso, cuando la dirección de destino no encaja en ninguna de las rutas definidas
- El extremo opuesto son las rutas con máscara de 32 bits. Estas solo sirven para una dirección de destino concreta, por eso se les llama rutas host.
- Se suelen utilizar para marcar ‘excepciones’, por ejemplo cuando un host esta temporalmente fuera de su LAN habitual

Ejemplo, Rutas hosts



Este host tiene dos direcciones sobre la misma interfaz

IP sin clases o ‘classless’

- Hasta 1993 la asignación de direcciones se hacía en bloques de tamaño fijo de acuerdo con las clases A, B y C (redes /8, /16 y /24 respectivamente). Pero:
  - De la clase A solo hay 127 redes, hace mucho tiempo que no se asigna ninguna
  - La clase B es demasiado grande para la mayoría de organizaciones (65000 hosts)
  - La clase C es demasiado pequeña para la mayoría (256 hosts)
- Casi todas las organizaciones optaban por pedir redes clase B, aunque les sobraba mucho espacio.
- Consecuencias
  - Rápido agotamiento del espacio de direcciones.
- Solución
  - Ofrecer tallas intermedias asignando grupos de redes clase C

IP sin clases o ‘classless’

- Problema 2
  - Las tablas de rutas crecían mucho más deprisa que antes (había que enrutar por separado cada red asignada)
- Solución 2
  - Asignar los grupos de forma que sean agregables, es decir que puedan referenciarse por una máscara común, así solo se necesita declarar una ruta
- El tamaño de las redes puede ser ahora cualquier potencia entera de 2 (256, 512, 1024, etc.)
- Este mecanismo se aplica no solo al rango de clase C sino también al rango libre de clase A y B. En la práctica significa **abolir el sistema de clases (IP classless, sin clases)**

Protocolo TCP/IP

- Subredes
- **Protocolo UDP**
- Protocolo TCP

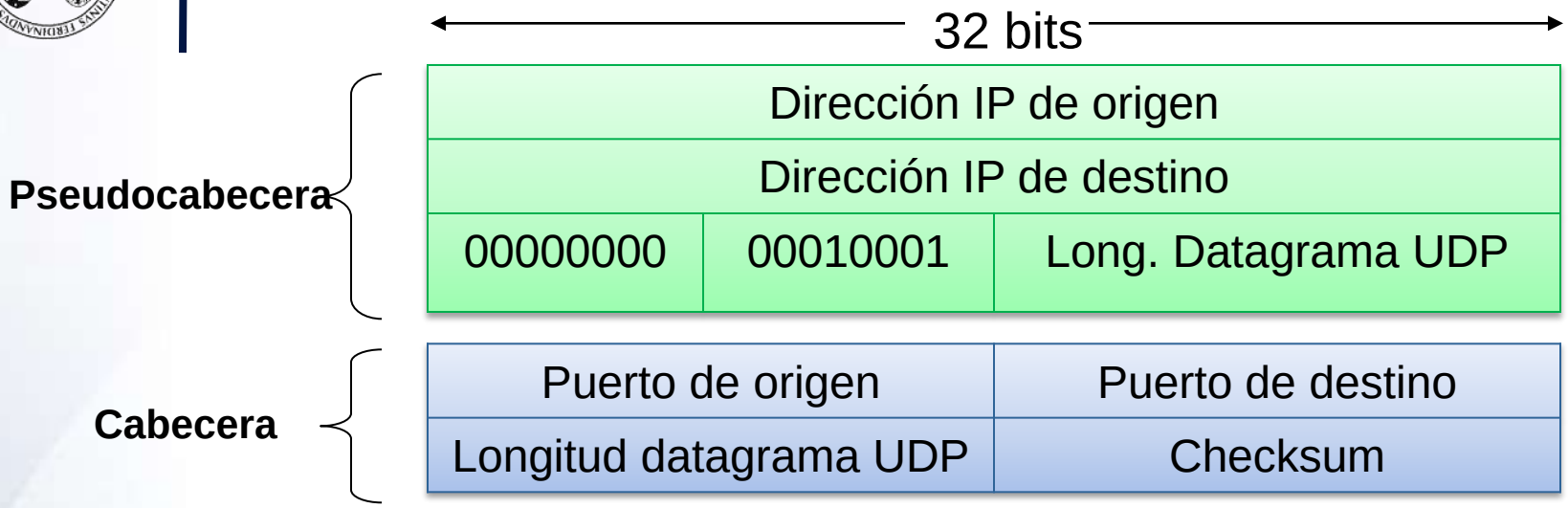
Funciones del Nivel de Transporte

- Se encarga del transporte de los datos extremo a extremo (host a host)
- Realiza la comunicación de forma transparente al medio físico. Usa los servicios del nivel de red
- Multiplexa tráfico de diversas instancias (procesos) del nivel de aplicación. El nivel de transporte (como el de red) tiene una sola instancia en el host
- El servicio que ofrece puede ser de dos tipos:
  - **Orientado a conexión:** garantiza la entrega de los datos, sin pérdidas ni duplicados. Ej.: TCP (Internet), TP4 (OSI)
  - **No orientado a conexión**: equivale al servicio que ofrece IP,pero a nivel de transporte. Ej.: UDP (Internet), TP0 (OSI)

Protocolo UDP

- Servicio sencillo, CLNS, no fiable
- Se utiliza en los siguientes entornos:
  - El intercambio de mensajes es muy escaso, ej.:consultas al DNS (servidor de nombres)
  - La aplicación es en tiempo real y no puede esperar confirmaciones. Ej.: videoconferencia, voz sobre IP
  - Los mensajes se producen regularmente y no importa si se pierde alguno. Ej: NTP, SNMP
  - El medio de transmisión es altamente fiable y sin congestión (LANs). Ej: NFS
  - Se envía tráfico broadcast/multicast

La cabecera UDP



La pseudocabecera se añade al principio del datagrama para el cálculo del checksum, pero no se envía. Permite a UDP comprobar que IP no se ha equivocado (ni le ha engañado) en la entrega del datagrama.

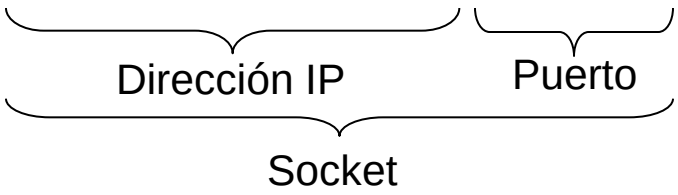
El valor $10001_2 = 17_{10}$ indica que el protocolo de transporte es UDP

Protocolo UDP

- Las TPDUs de UDP se denominan **mensajes** o **datagramas UDP**
- UDP multiplexa los datos de las aplicaciones y efectúa opcionalmente una comprobación de errores, pero **no realiza**:
  - Control de flujo
  - Control de congestión
  - Retransmisión de datos perdidos
  - Conexión/desconexión

Multiplexación

- La multiplexación se realiza mediante el puerto (origen o destino) que puede valer de 0 a 65535.
- Los puertos 0 a 1023 están reservados para servidores ‘bien conocidos’ (‘well known ports’)
- La combinación de una dirección IP y un puerto identifica un ‘socket’ (origen o destino de los datagramas UDP): 147.156.135.22.1038



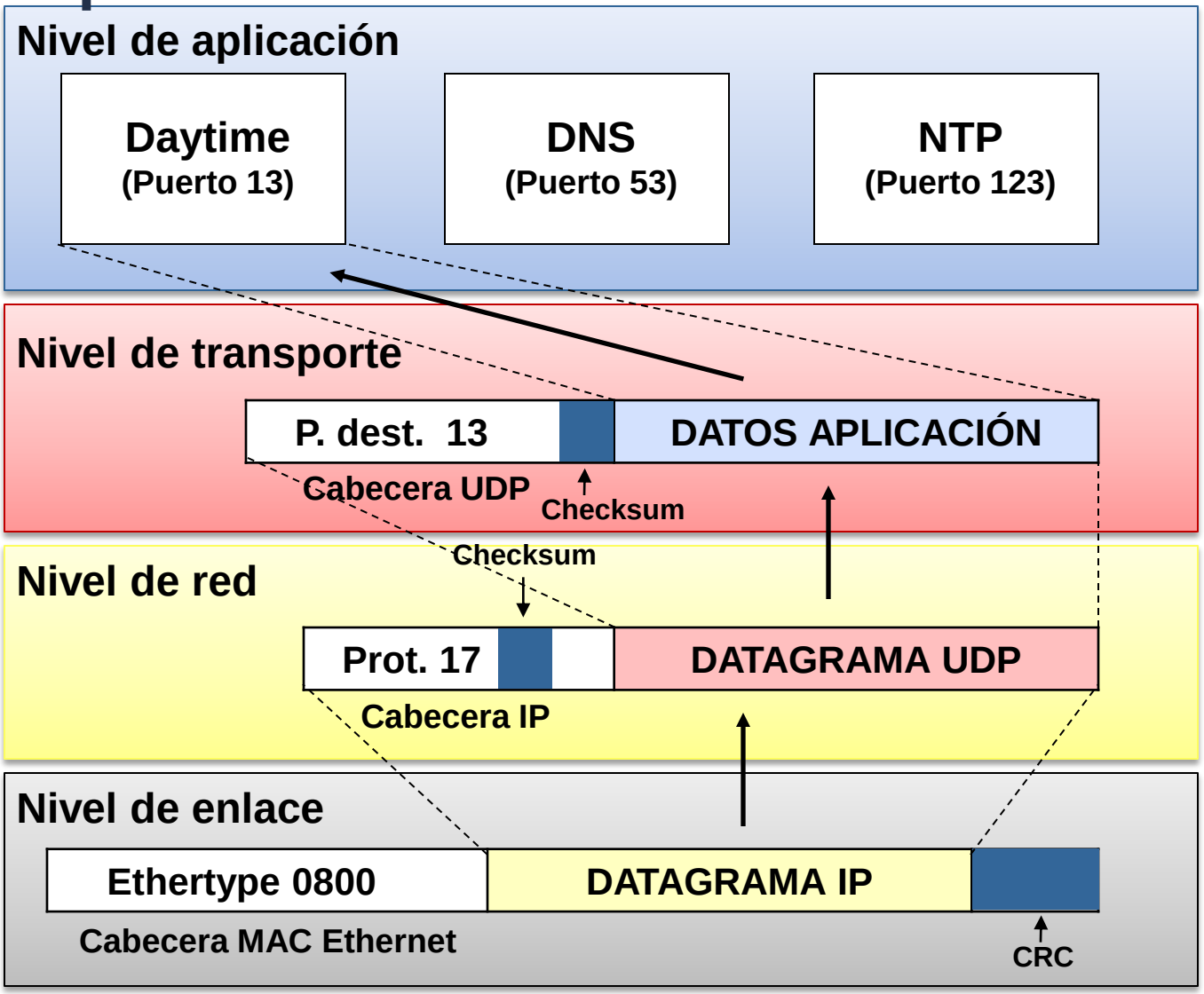
Multiplexación

Múltiples instancias (una o varias por protocolo)

Dos instancias (TCP y UDP)

Una instancia IP (puede haber otros protocolos)

Múltiples instancias (una por interfaz)



Cabeceras IP y UDP en una petición/respuesta SNMP

IP: ----- IP Header -----	IP: ----- IP Header -----
IP:	IP:
IP: Version=4, header length=20 bytes	IP: Version=4, header length=20 bytes
IP: DiffServ = 00	IP: DiffServ = 00
IP: Total length = 131 bytes	IP: Total length = 160 bytes
IP: Identification = 21066	IP: Identification = 2015
IP: DF = 0, MF = 0	IP: DF = 0, MF = 0
IP: Fragment offset = 0 bytes	IP: Fragment offset = 0 bytes
IP: Time to live = 60 seconds/hops	IP: Time to live = 64 seconds/hops
→ IP: Protocol = 17 (UDP)	IP: Protocol = 17 (UDP)
IP: Header checksum = 2A13 (correct)	IP: Header checksum = 7061 (correct)
IP: Source address = [128.1.1.1]	IP: Source address = [128.1.1.10]
→ IP: Destination address = [128.1.1.10]	IP: Destination address = [128.1.1.1]
IP: No options	IP: No options
IP:	IP:
UDP: ----- UDP Header -----	UDP: ----- UDP Header -----
UDP:	UDP:
→ UDP: Source Port = 1227	UDP: Source Port = 161 (SNMP)
UDP: Destination port = 161 (SNMP)	UDP: Destination port = 1227
UDP: Length = 111	UDP: Length = 140
UDP: No checksum	UDP: Checksum = 4D4F (correct)
UDP:	UDP:

Protocolo TCP/IP

- Subredes
- Protocolo UDP
- **Protocolo TCP**
  - **Dia Examen: 16 Mayo (Aula 7)**
  - **Presentacion 23 Mayo (15:30 Aula 7)**

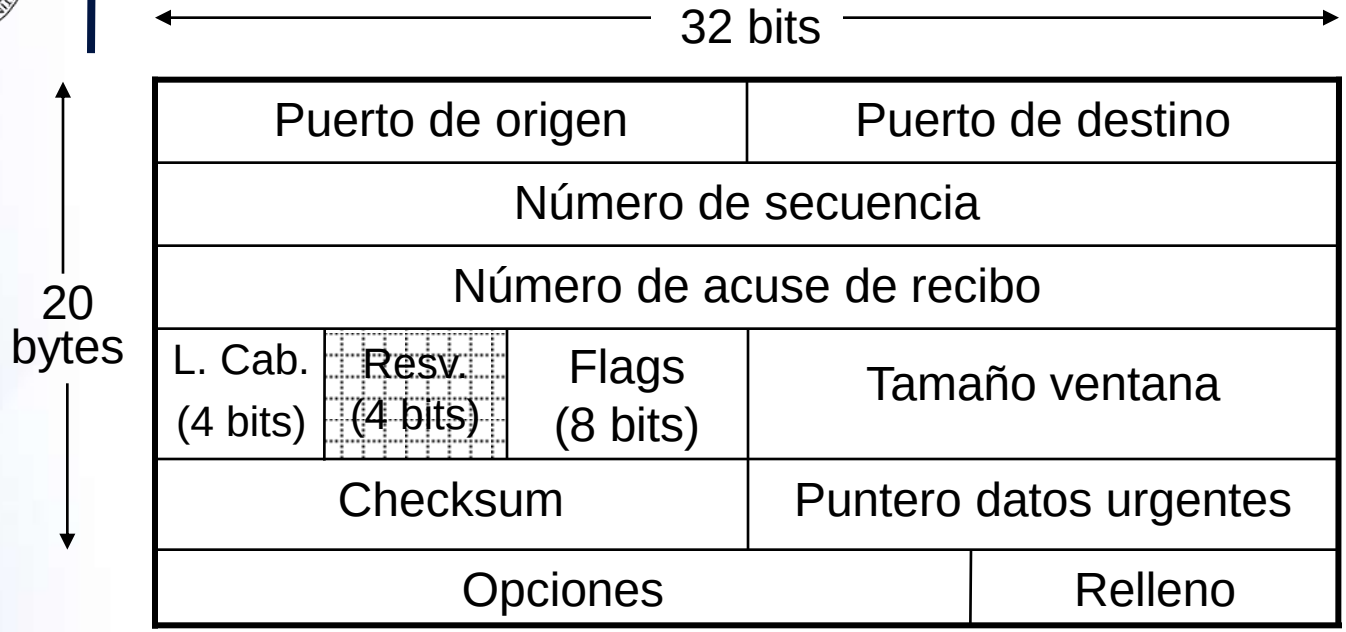
TCP
(Transmission Control Protocol)

- El protocolo TCP ofrece el servicio de transporte orientado a conexión (CONS) en Internet
- Está diseñado para ofrecer un transporte fiable sobre un servicio no fiable del nivel de red (el que le suministra IP)
- Las TPDUs de TCP se llaman **segmentos**
- El TCP actual se especificó en el RFC 793 en 1981 y sigue plenamente vigente

Funciones de TCP

- Multiplexar el nivel de aplicación (port)
- Establecer y terminar conexiones
- Controlar errores, retransmitiendo segmentos perdidos o erróneos. Eliminar duplicados
- Gestionar los buffers y ejercer control de flujo de forma eficiente
- Gestionar el intercambio de datos con las aplicaciones
- Efectuar control de congestión

La cabecera TCP



Flags:	CWR:	Congestion Window Reduced
	ECE:	ECN Echo (ECN=Explicit Congestion Notification)
	URG:	el segmento contiene datos urgentes
	ACK:	el campo número de acuse de recibo tiene sentido
	PSH:	el segmento contiene datos ‘Pushed’
	RST:	ha habido algún error y la conexión debe cerrarse
	SYN:	indica el inicio de una conexión
	FIN:	indica el final de una conexión

Multiplexación

- Se utiliza el número de puerto (origen o destino) como en UDP. Puede valer de 0 a 65535
- Los puertos 0 a 1023 están reservados para servidores ‘bien conocidos’
- La combinación de dirección IP y puerto identifica el ‘socket’
- Una conexión TCP queda especificada por los dos sockets que se comunican (IP origen-puerto origen, IP destino-puerto destino)

Algunos servicios 'bien conocidos'

Servicio	Puerto	TCP	UDP
DayTime	13	X	X
FTP	**21**	X	
SSH	**22**	X	
TelNet	23	X	
SMTP	**25**	X	
Domain (DNS)	53	X	X
BOOTP	67		X
TFTP	69		X
HTTP	**80**	X	
POP3	110	X	
NTP	123		X
SNMP	161		X
LDAP	389	X	
HTTPS	**443**	X	

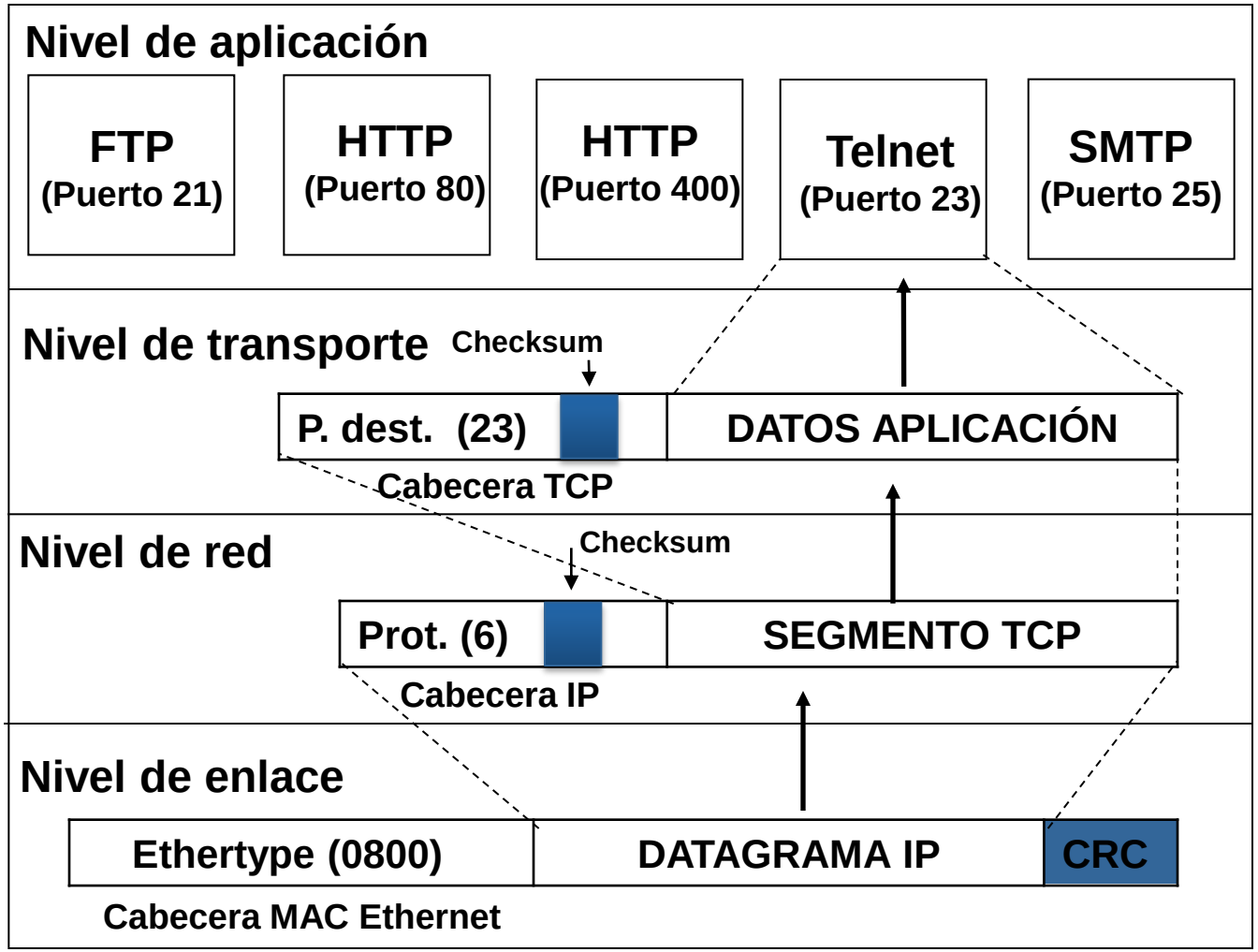
Multiplexación

Múltiples instancias (una o varias por protocolo)

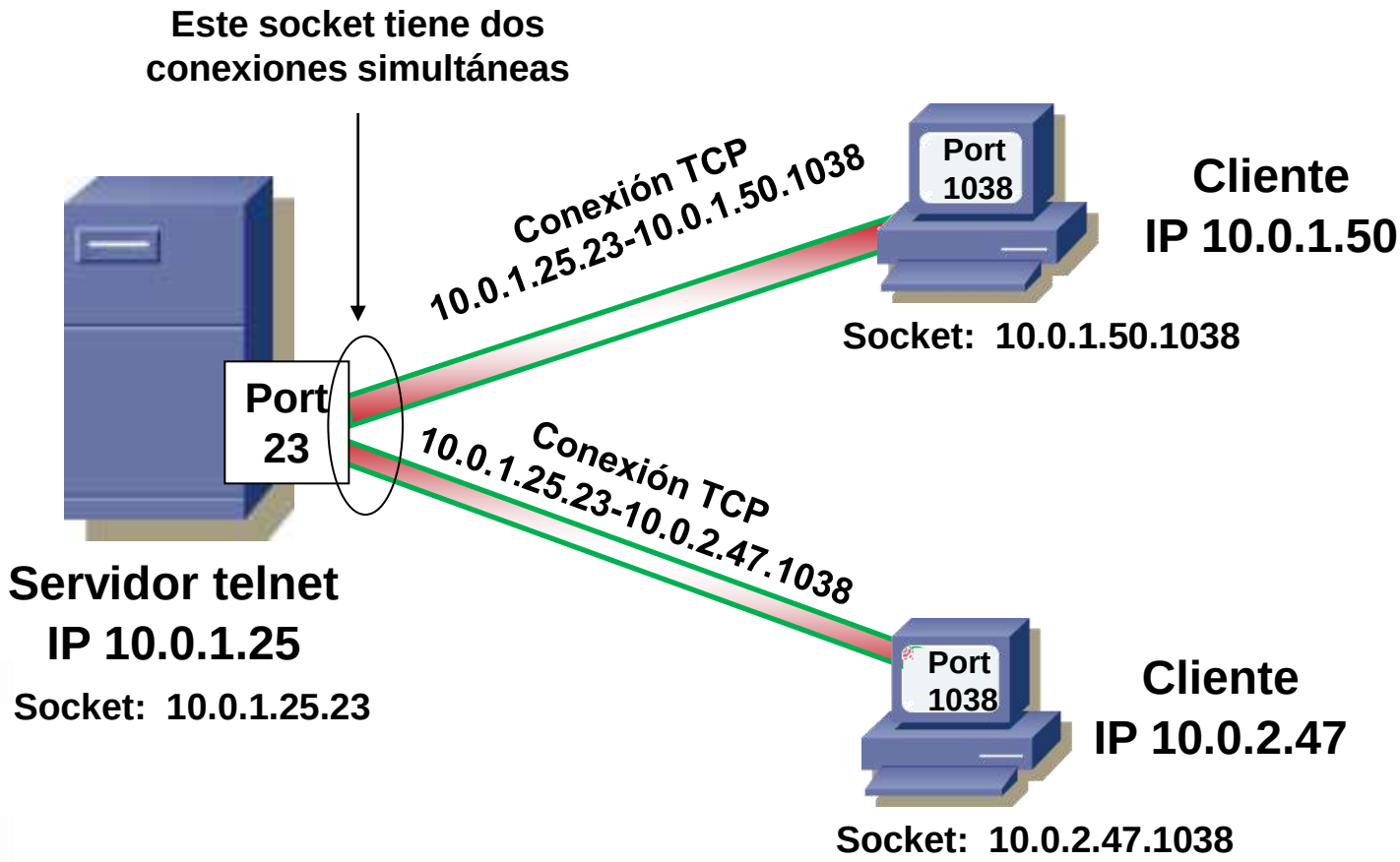
Dos instancias (TCP y UDP)

Una instancia IP (puede haber otros protocolos)

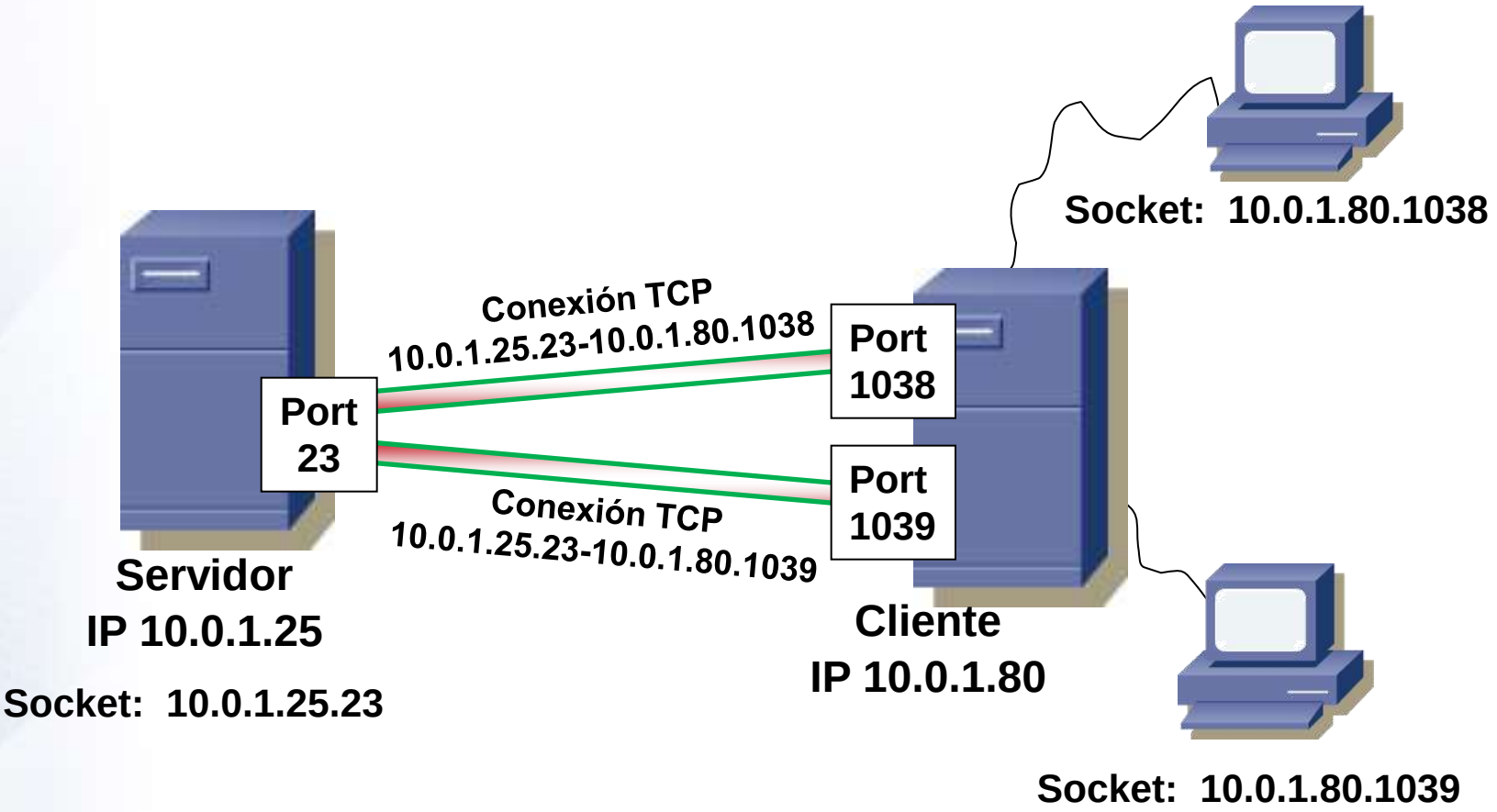
Múltiples instancias (una por interfaz)



Dos conexiones TCP a un mismo socket desde dos sockets con el mismo número de puerto



Dos conexiones TCP a un mismo socket desde dos sockets con la misma dirección IP



Conexiones TCP del host 147.156.1.25
(conectado por telnet desde 147.156.1.219)

```
Netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         (state)
tcp        0      0  147.156.1.25.2480      147.156.1.1.143         ESTABLISHED
tcp        0      0  147.156.1.25.23        147.156.96.8.1034       ESTABLISHED
tcp        0    240  147.156.1.25.23        147.156.1.219.1036      ESTABLISHED
tcp        0      0  147.156.1.25.513       147.156.1.3.1018        ESTABLISHED
tcp        0      0  147.156.1.25.513       147.156.1.3.1019        ESTABLISHED
tcp        0      0  147.156.1.25.2429      147.156.1.15.6000       ESTABLISHED
tCP        0      0  147.156.1.25.2428      147.156.1.15.6000       ESTABLISHED
tcp        0      0  147.156.1.25.1022      147.156.1.3.1002        ESTABLISHED
tcp        0      0  147.156.1.25.514       147.156.1.3.1004        CLOSE_WAIT
tcp        0      0  147.156.1.25.1023      147.156.1.3.1005        ESTABLISHED
tcp        0      0  147.156.1.25.514       147.156.1.3.1007        CLOSE_WAIT
tcp        0      0  147.156.1.25.139       147.156.1.219.1029      ESTABLISHED
tcp        0      0  *.143                  *.*                     LISTEN
tcp        0      0  *.144                  *.*                     LISTEN
tcp        0      0  147.156.1.25.23        147.156.3.12.1945       ESTABLISHED
tcp        0      0  *.139                  *.*                     LISTEN
tcp        0      0  *.5000                 *.*                     LISTEN
tcp        0      0  *.25                   *.*                     LISTEN
tcp        0      0  *.19                   *.*                     LISTEN
tcp        0      0  *.9                    *.*                     LISTEN
udp        0      0  *.16522                *.*
udp        0      0  *.16520                *.*
udp        0      0  147.156.1.25.123       *.*
udp        0      0  127.0.0.1.123          *.*
udp        0      0  *.123                  *.*
```

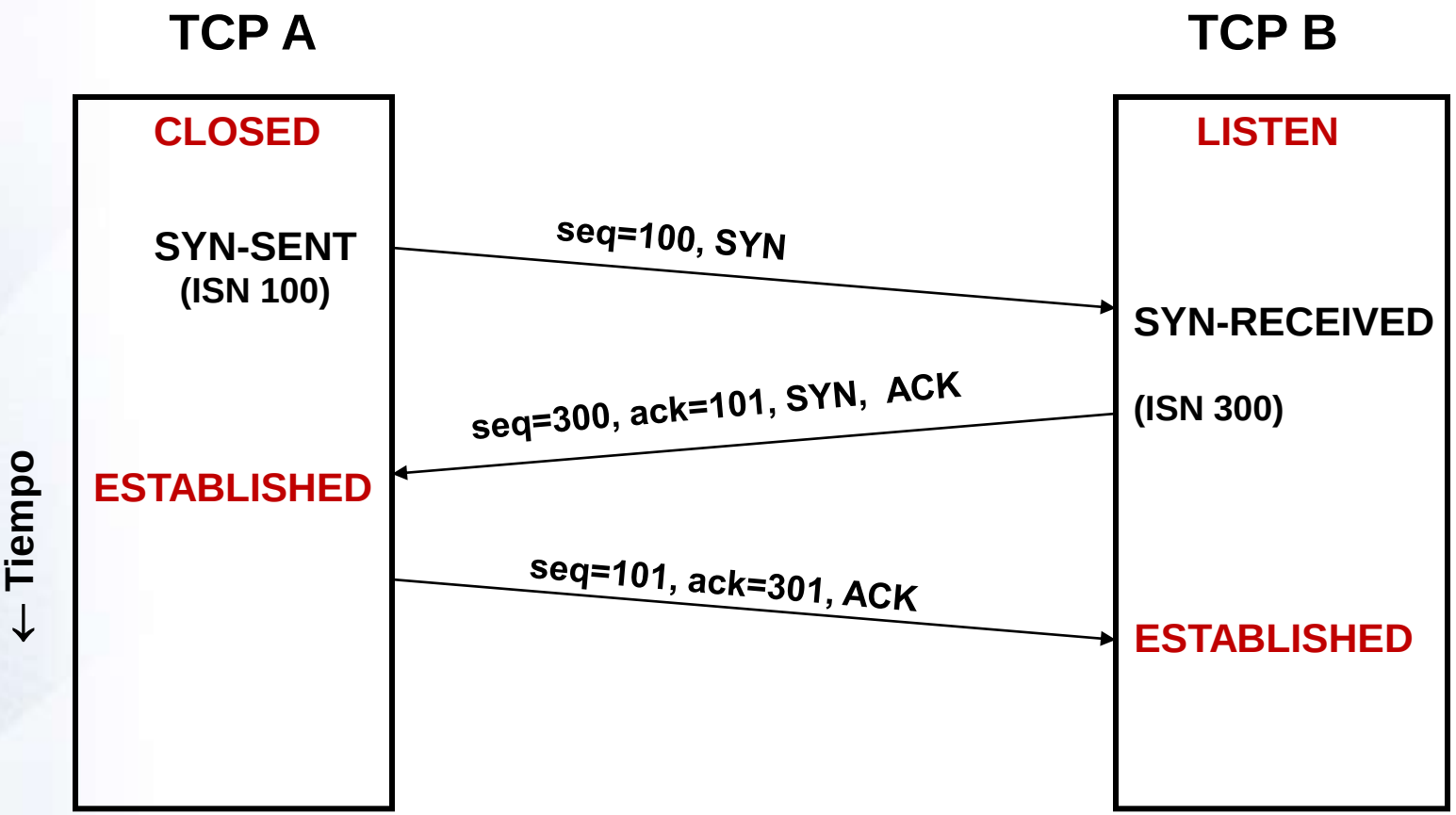
Conexión por ‘Saludo a tres vías’

- Los segmentos pueden llegar duplicados (p. ej. se pierde la confirmación de un segmento con lo que el emisor lo reenvía)
- Con un procedimiento de conexión simple los segmentos duplicados podrían causar problemas. Una sesión entera podría duplicarse.
- Para evitar los problemas debidos a duplicados se utiliza un procedimiento de conexión más elaborado denominado saludo a tres vías
- El saludo a tres vías se basa en la elección de un número que identifica de forma única cada intento de conexión y que actúa como PIN. De este modo se evita el riesgo de aceptar como válidos segmentos retrasados que pudieran aparecer fruto de conexiones anteriores

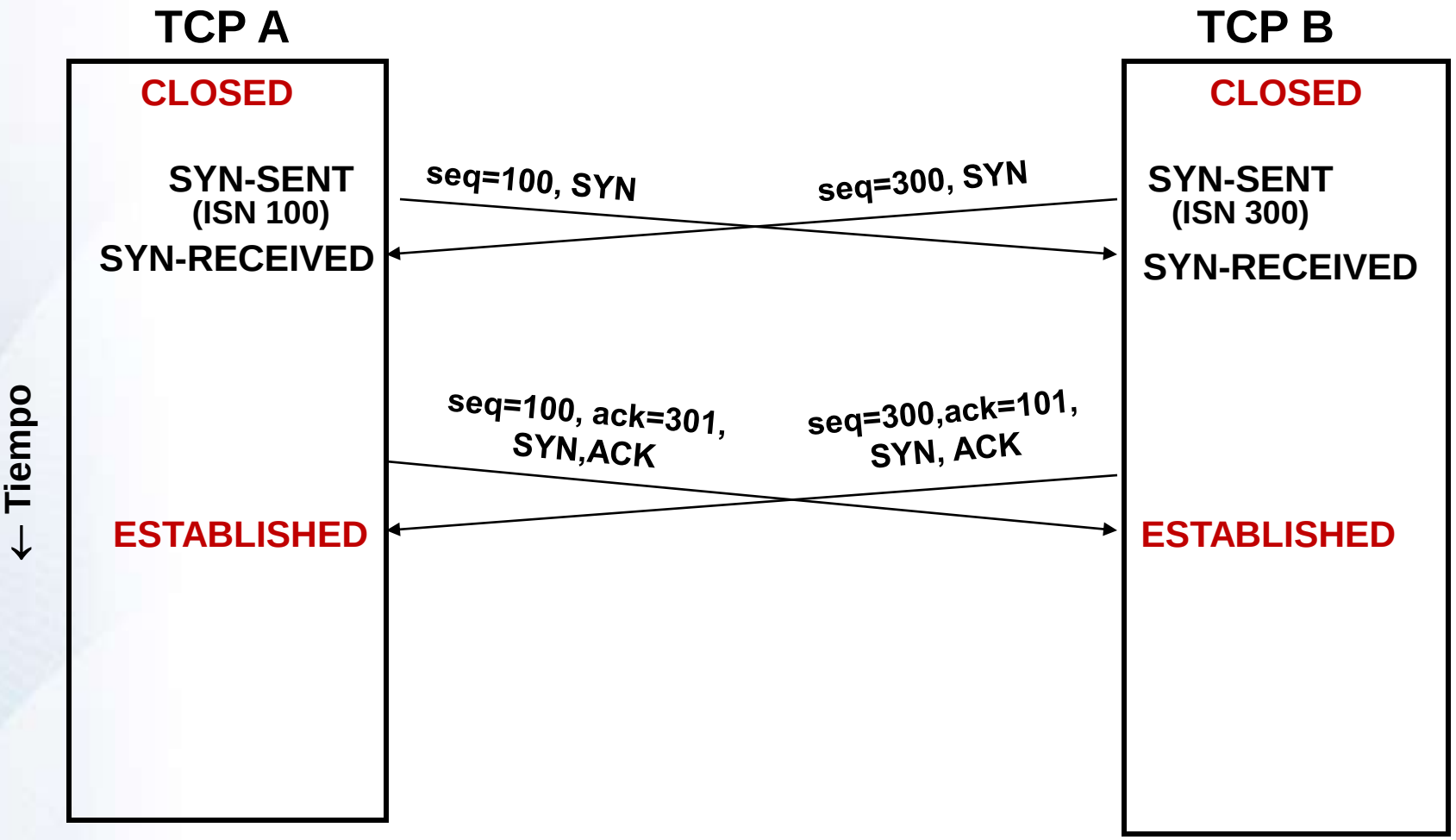
Procedimiento del saludo a tres vías

1. El cliente elige para cada intento de conexión un número único. El número elegido lo incluye en la petición de conexión que envía al servidor
2. El servidor, cuando recibe la petición, elige otro número único y envía una respuesta al cliente indicándoselo
3. El cliente al recibir la respuesta considera establecida la conexión. A continuación envía un tercer mensaje en el que acusa recibo del anterior. El servidor considera establecida la conexión cuando el recibe este tercer mensaje

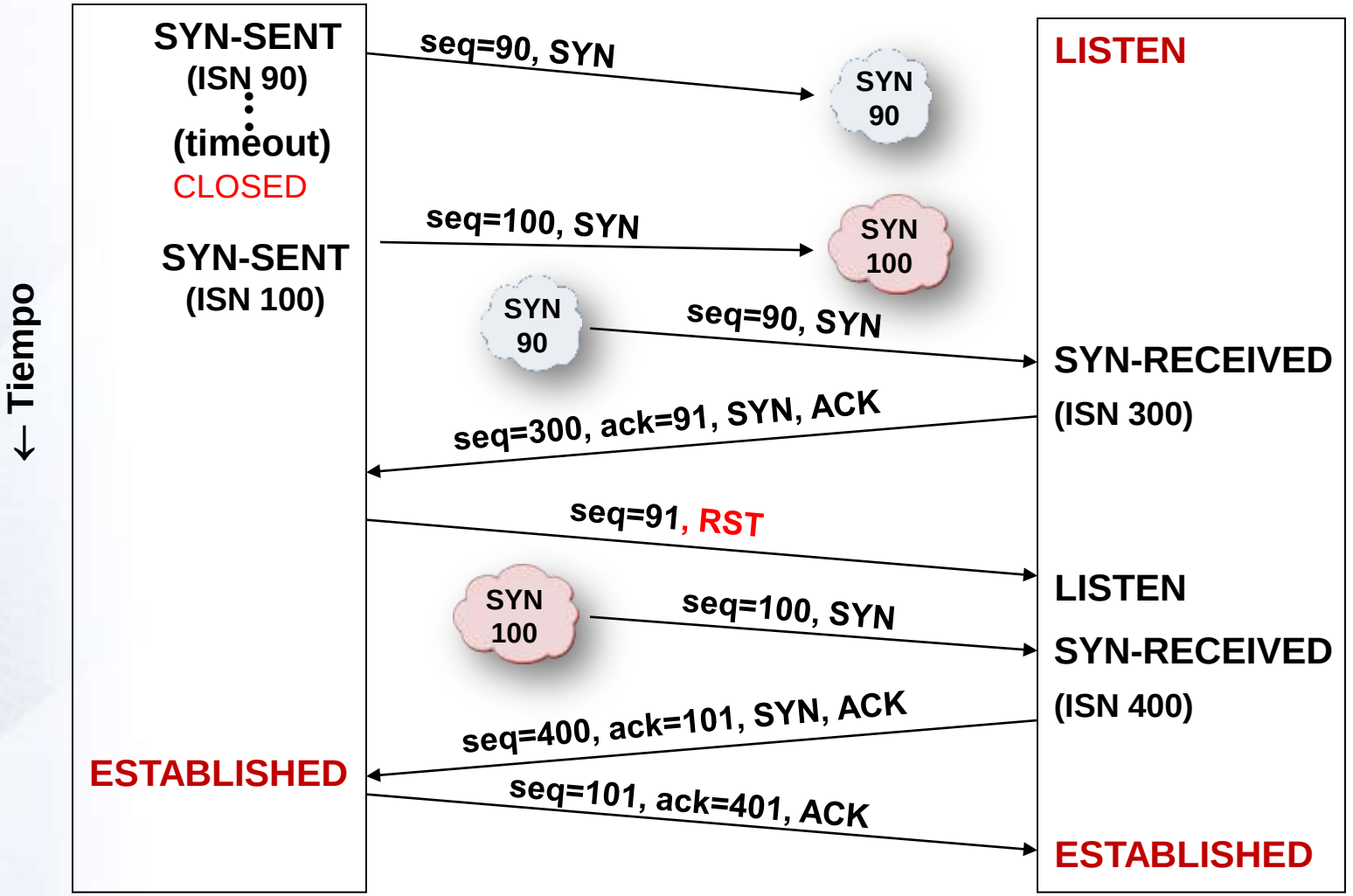
Establecimiento de una conexión TCP por saludo a tres vías



Saludo a tres vías, conexión simultánea



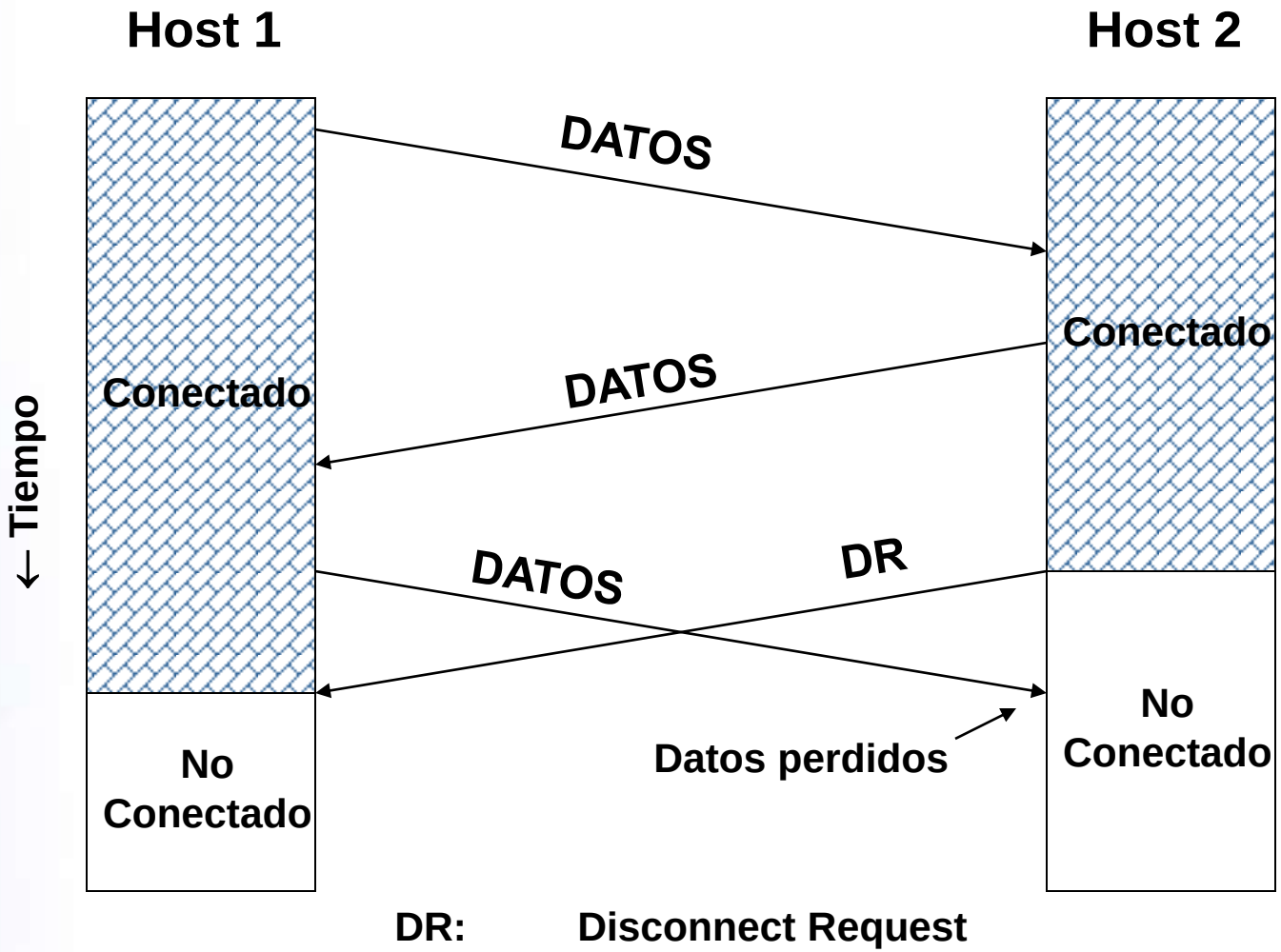
Conexión con SYN duplicado



Desconexión

- Puede ser de dos tipos:
  - *Simétrica*: la conexión se considera formada por dos circuitos simplex y cada host solo puede cortar uno (aquel en el que él emite datos). El cierre de un sentido se interpreta como una 'invitación' a cerrar el otro
  - *Asimétrica:* desconexión unilateral (un host la termina en ambos sentidos sin esperar a recibir confirmación del otro). Puede provocar pérdida de información

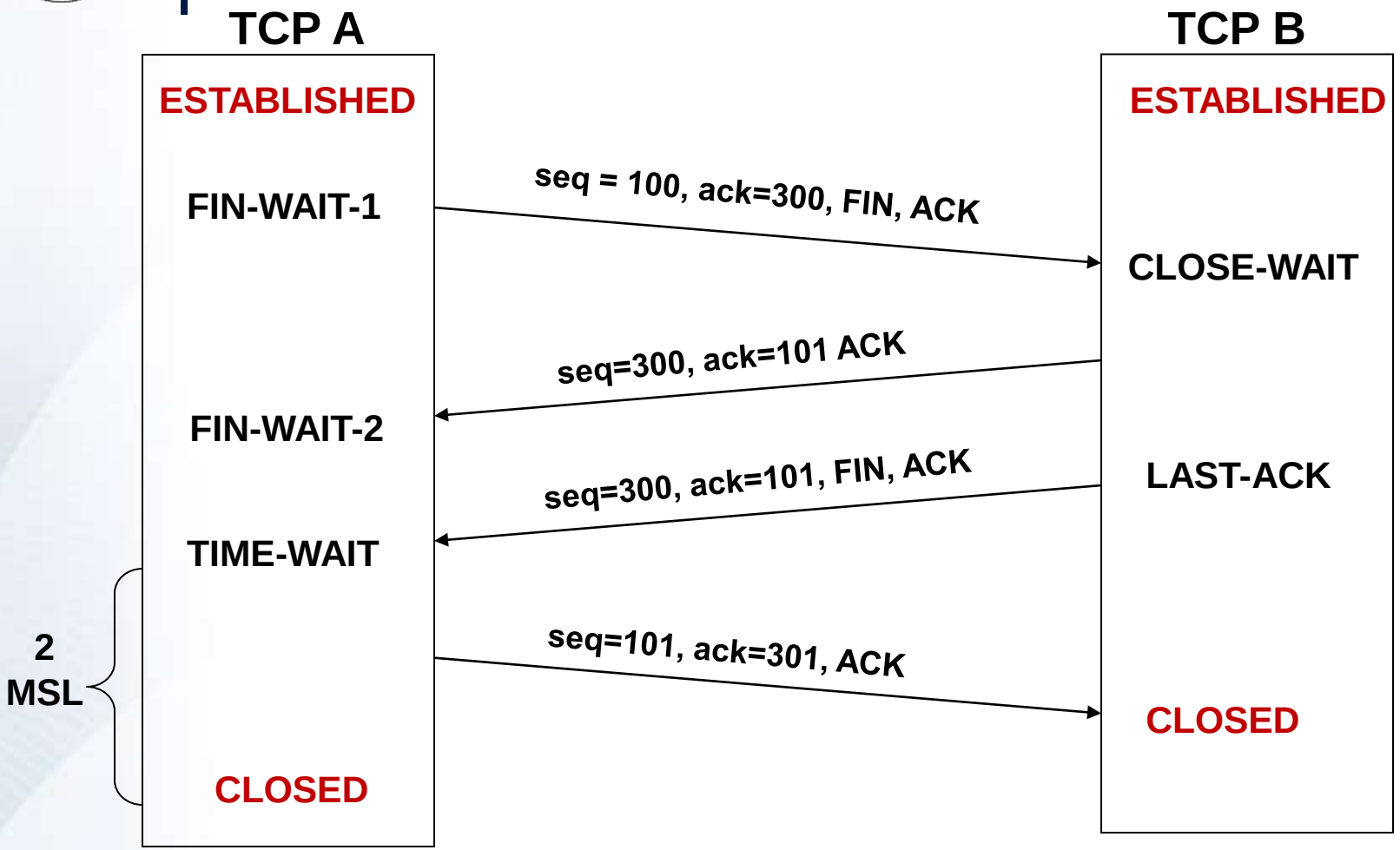
Desconexión asimétrica



Desconexión por saludo a tres vías

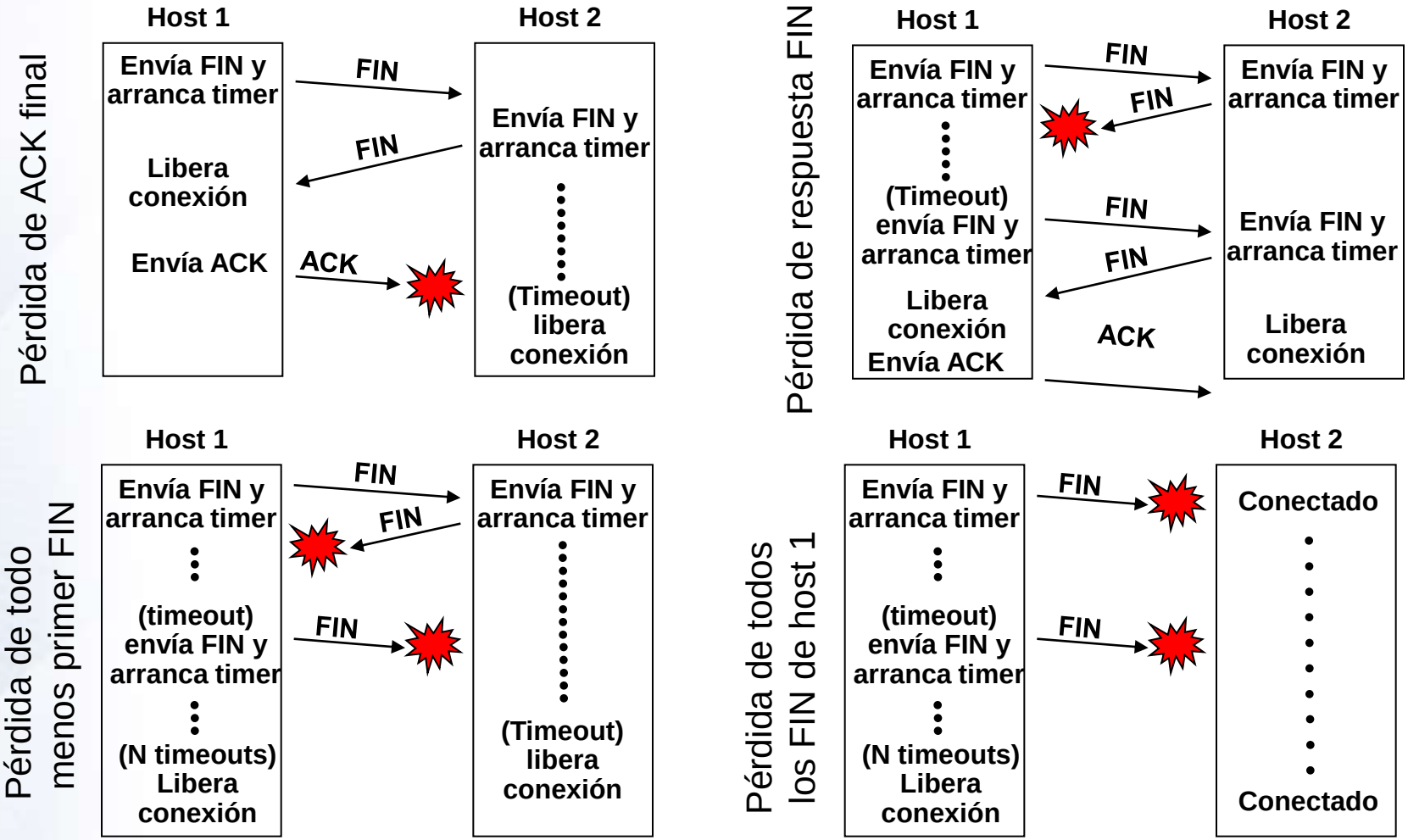
- Se trata de una desconexión simétrica en la que se tiene una seguridad razonable de que no se pierden datos
- Supone el intercambio de tres mensajes, de forma análoga a la conexión, de ahí su nombre
- En caso de que alguno de los mensajes de desconexión se pierda una vez iniciado el proceso la conexión se termina por timeout

Desconexión a tres vías



MSL: Maximum Segment Lifetime (normalmente 2 minutos)

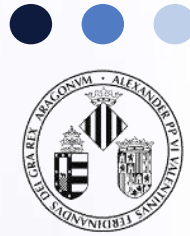
Desconexión a tres vías, casos anormales



Números de secuencia y flags

- El número de secuencia es el que corresponde al primer byte enviado en ese segmento
- TCP incrementa el número de secuencia de cada segmento según los bytes que tenía el segmento anterior, con una sola excepción:

  Los flags SYN y FIN, cuando están puestos, incrementan en 1 el número de secuencia
- Esto permite que se pueda acusar recibo de un segmento SYN o FIN sin ambigüedad
- Podemos considerar que los segmentos que tienen puesto el flag SYN o FIN llevan un byte de datos 'virtual'
- La presencia del flag ACK no incrementa el número de secuencia



Intercambio de datos
TCP ↔ aplicación

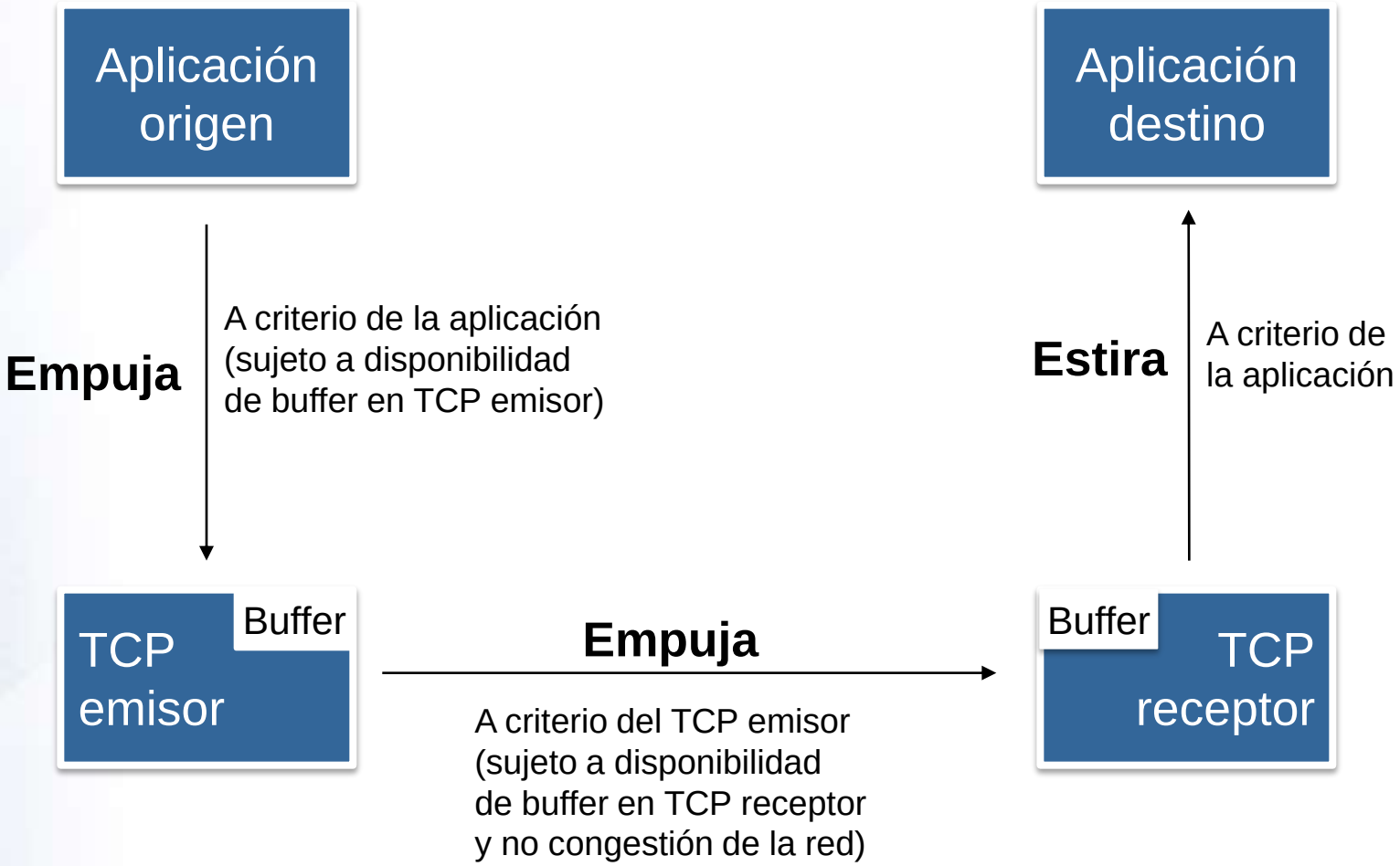
- **Aplicación → TCP**: la aplicación envía los datos a TCP cuando quiere (siempre y cuando TCP tenga espacio libre en el buffer de emisión)
- **TCP → Aplicación**: la aplicación lee del buffer de recepción de TCP cuando quiere y cuanto quiere. Excepción: datos urgentes
- Para TCP los datos de la aplicación son un flujo continuo de bytes, independientemente de la separación que pueda tener la aplicación (registros, etc.). Es responsabilidad de la aplicación asegurarse que esa separación (si existe) se mantendrá después de transmitir los datos

Intercambio de datos
TCP ↔ TCP

- El TCP emisor manda los datos cuando quiere.
- El TCP emisor decide el tamaño de segmento según sus preferencias. Al inicio de la conexión se negocia el MSS (Maximum Segment Size)
- Cada segmento ha de viajar en un datagrama
- Normalmente TCP intenta agrupar los datos para que los segmentos tengan la longitud máxima, reduciendo así el overhead debido a cabeceras y proceso de segmentos
- El TCP emisor puede aplicar la técnica de descubrimiento de la MTU del trayecto ('Path MTU Discovery', MTU = Maximum Transfer Unit) para ajustar el MSS al tamaño óptimo para esa comunicación

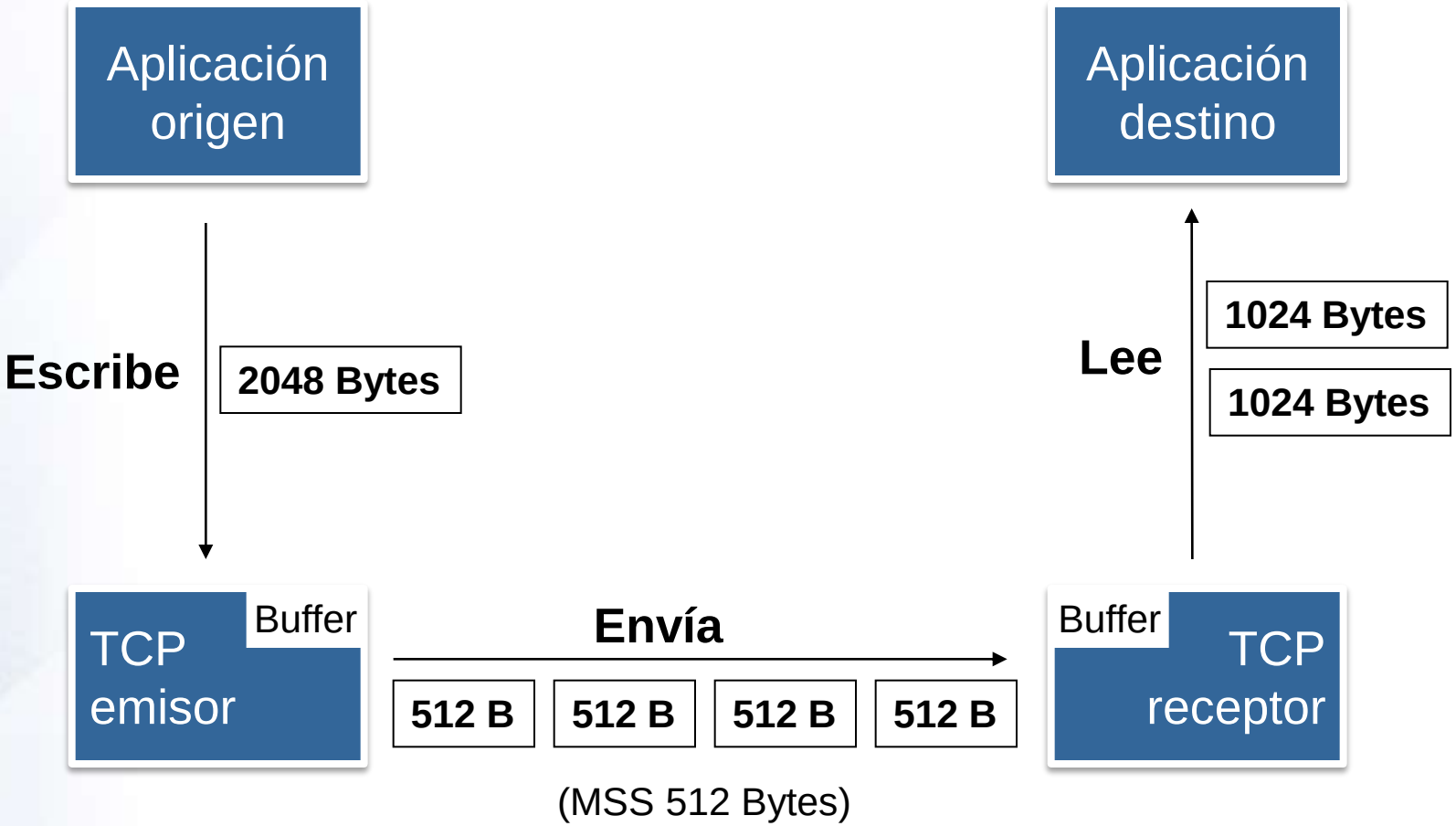
Intercambio de datos

TCP ↔ Aplicación
TCP ↔ TCP

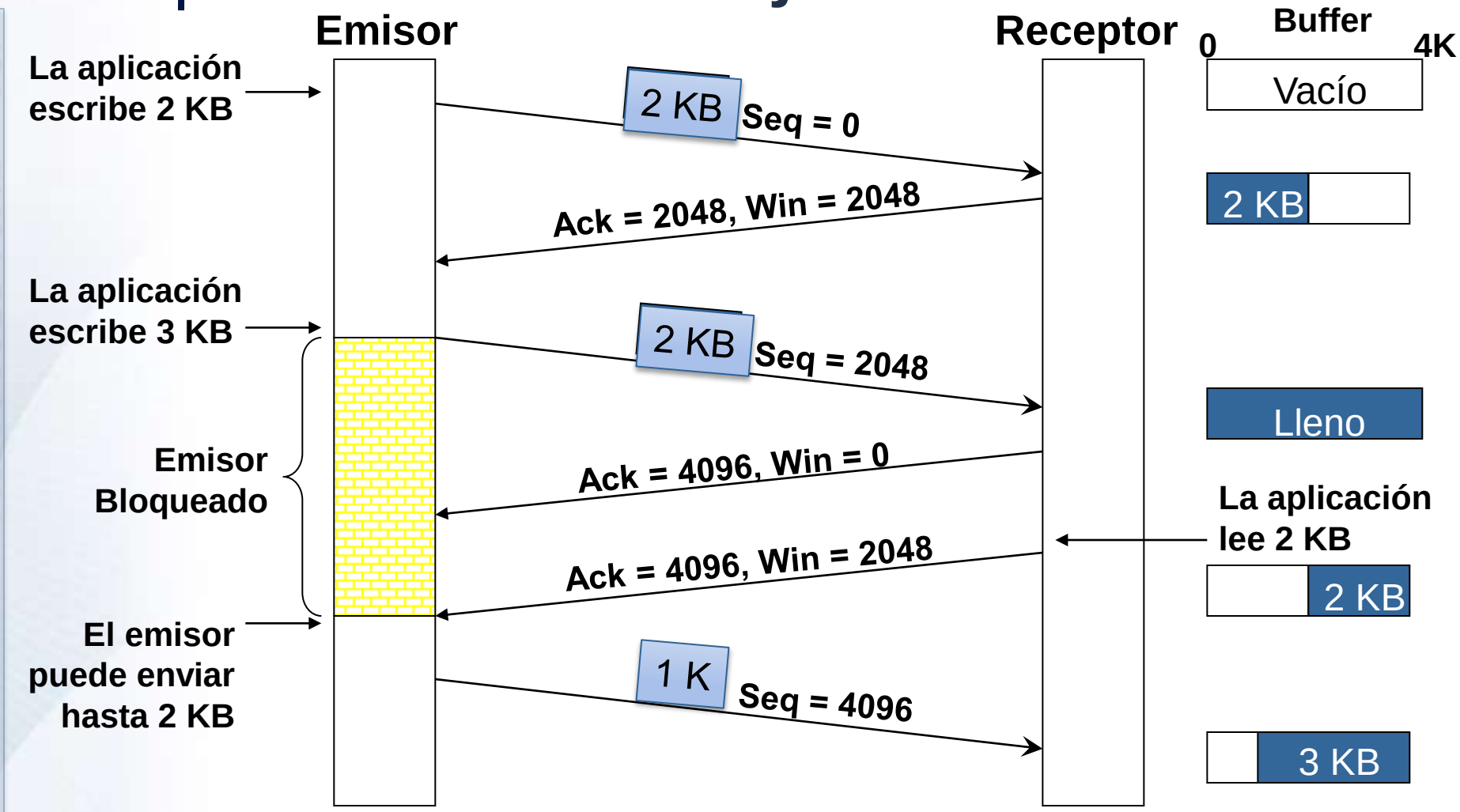


Intercambio de datos

TCP ↔ Aplicación
TCP ↔ TCP



Gestión de buffers y Control de flujo



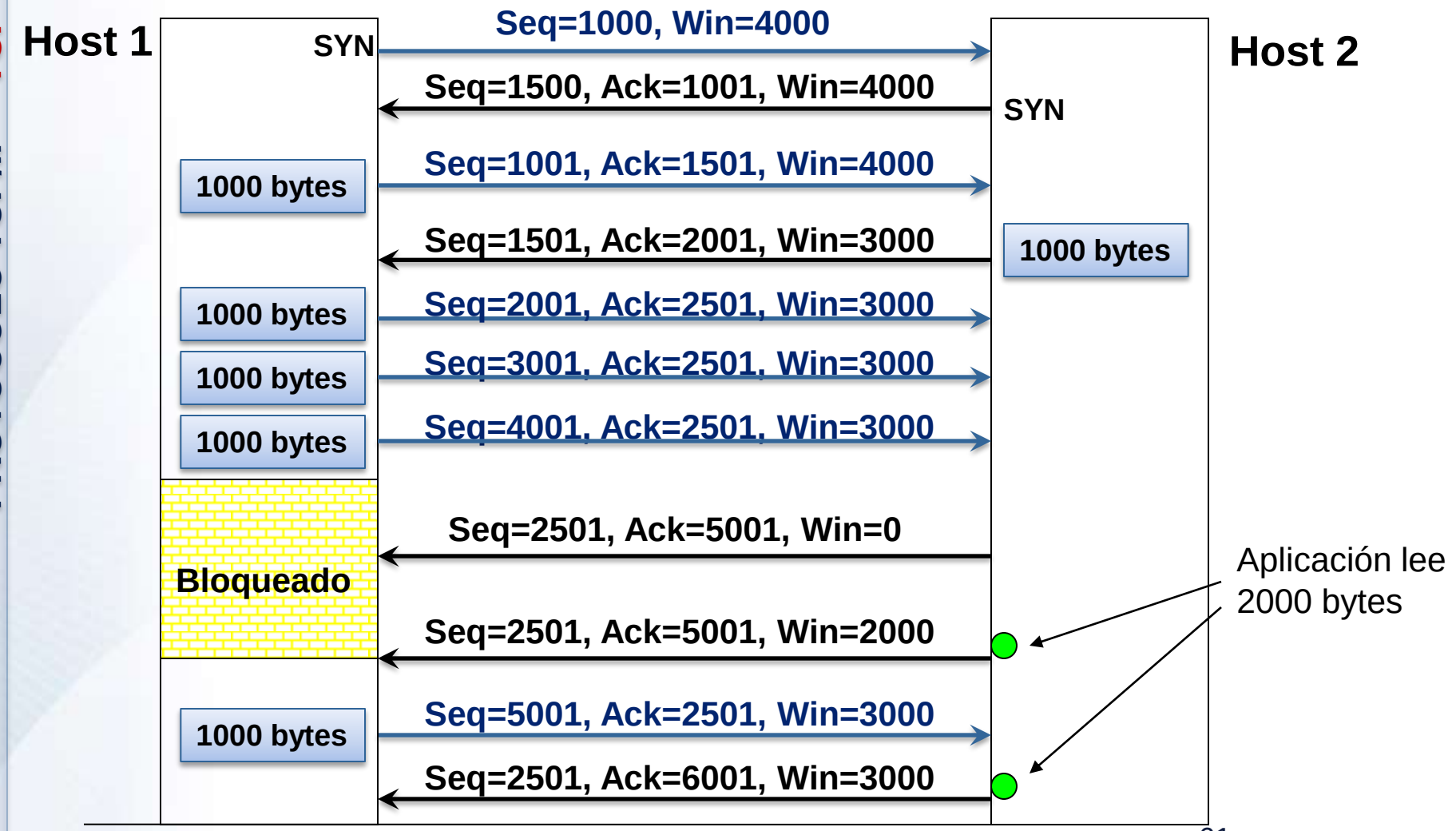
Gestión de buffers y control de flujo

- El TCP receptor nunca debería retirar el espacio en buffer que ya ha anunciado al emisor
- Sin embargo TCP debe estar preparado por si el del otro lado lo hace (esto se denomina ‘contraer la ventana’)

(Recordemos la Ley de Postel):

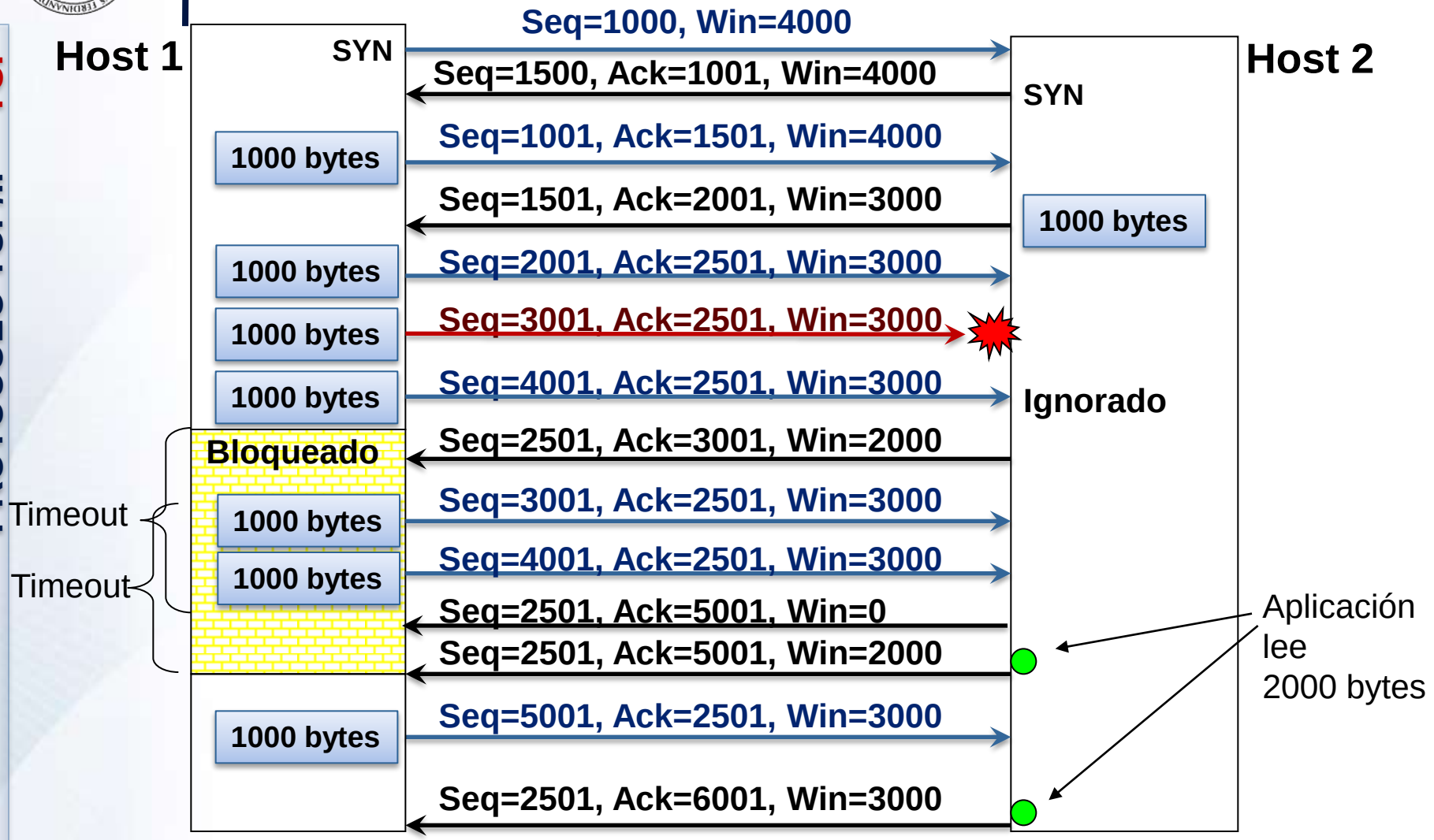
‘Sé estricto al enviar y tolerante al recibir’

Control de flujo y números de secuencia. Caso normal, sin pérdidas



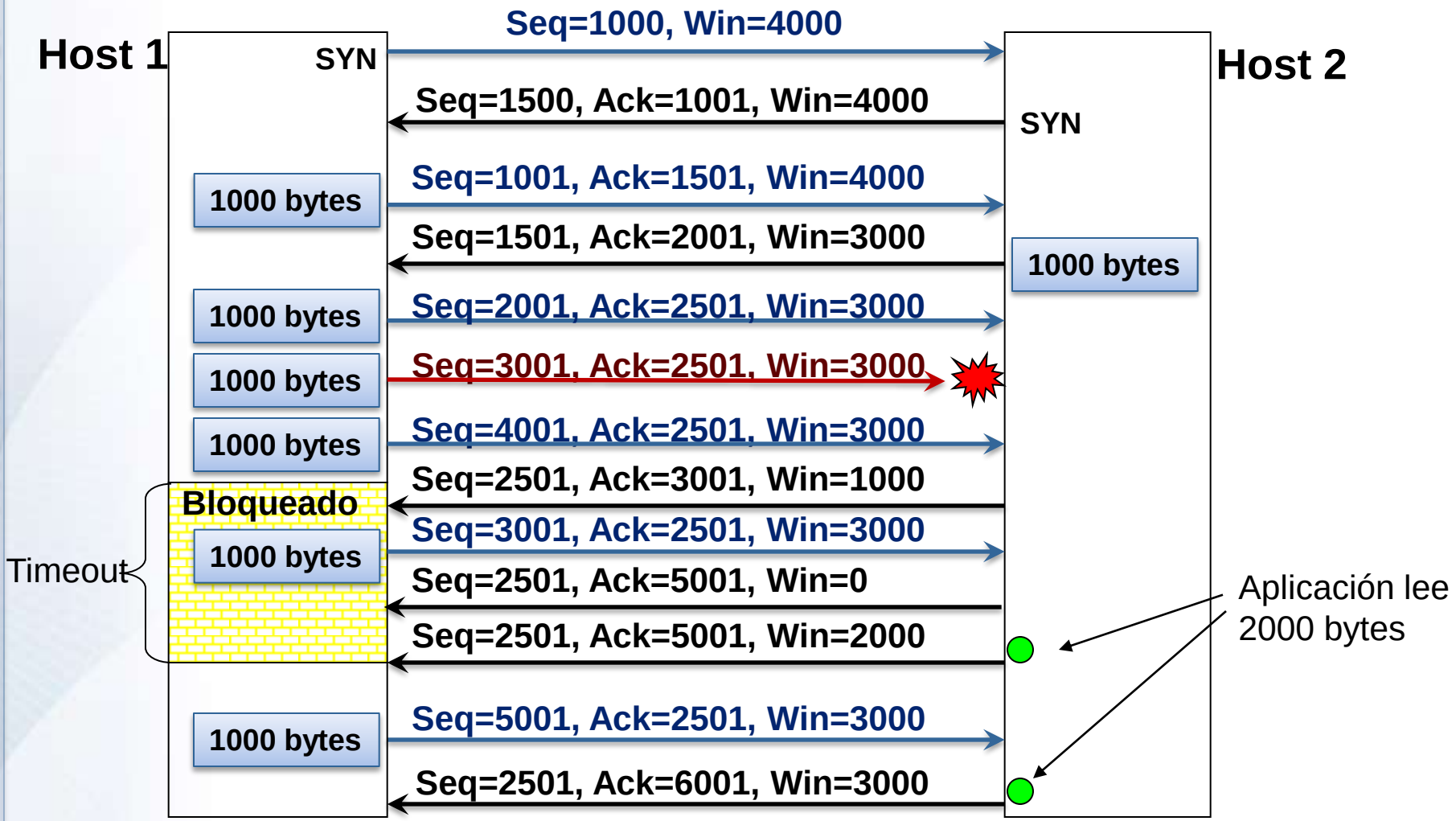
Pérdida de un paquete

Retransmisión con retroceso n



Pérdida de un paquete

Retransmisión con repetición selectiva



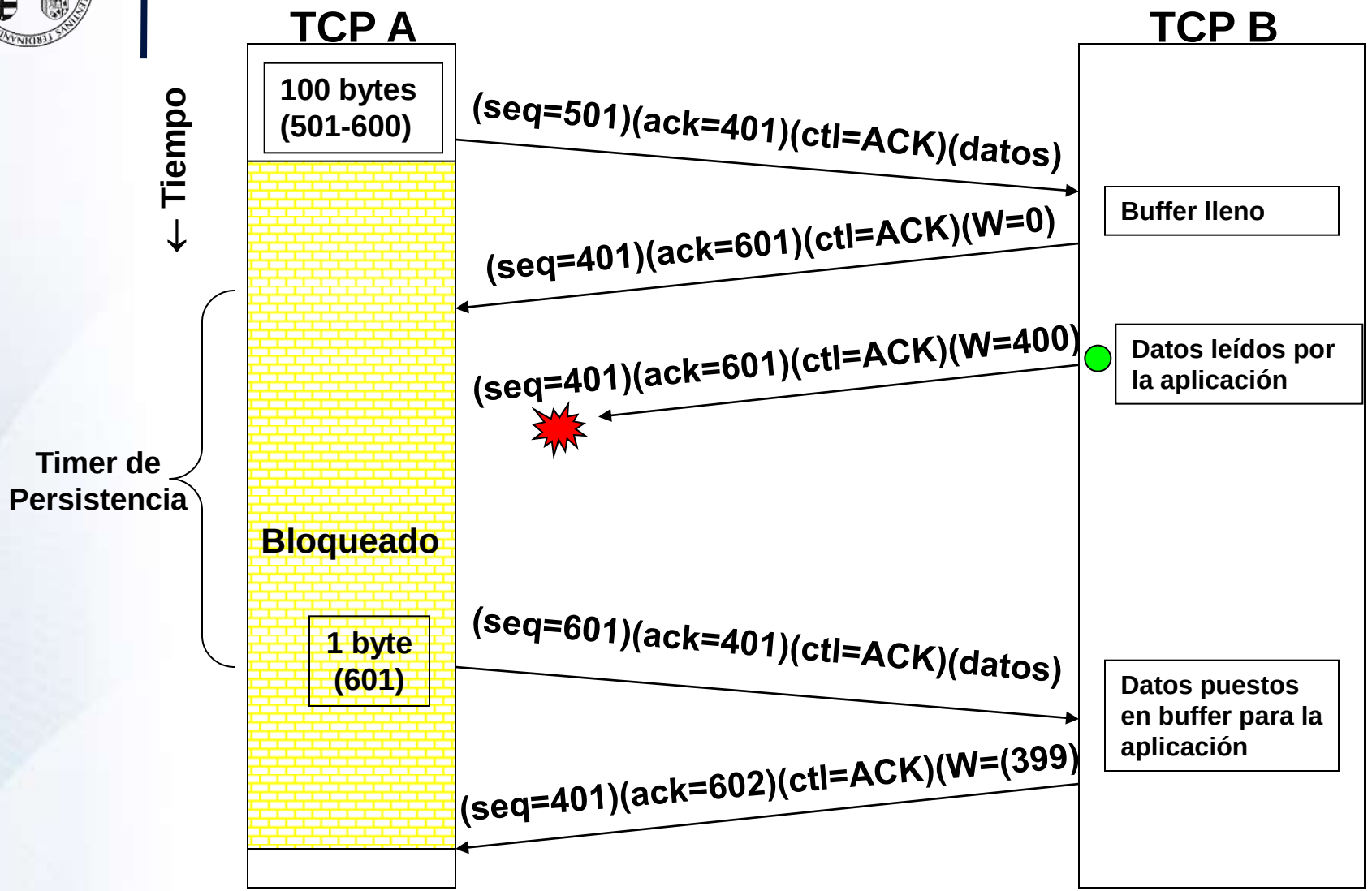
Intercambio de datos
casos excepcionales

- Datos ‘Pushed’ (bit PSH)
  - La aplicación pide al TCP emisor que envíe esos datos lo antes posible. El TCP receptor los pondrá a disposición de la aplicación de inmediato, para cuando ésta le pida datos. Ejemplo: telnet.
- Datos Urgentes (bit URG y Urgent Offset)
  - Los datos se quieren entregar a la aplicación remota sin esperar a que esta los pida. Ejemplo: abortar un programa con CTRL-C en una sesión telnet

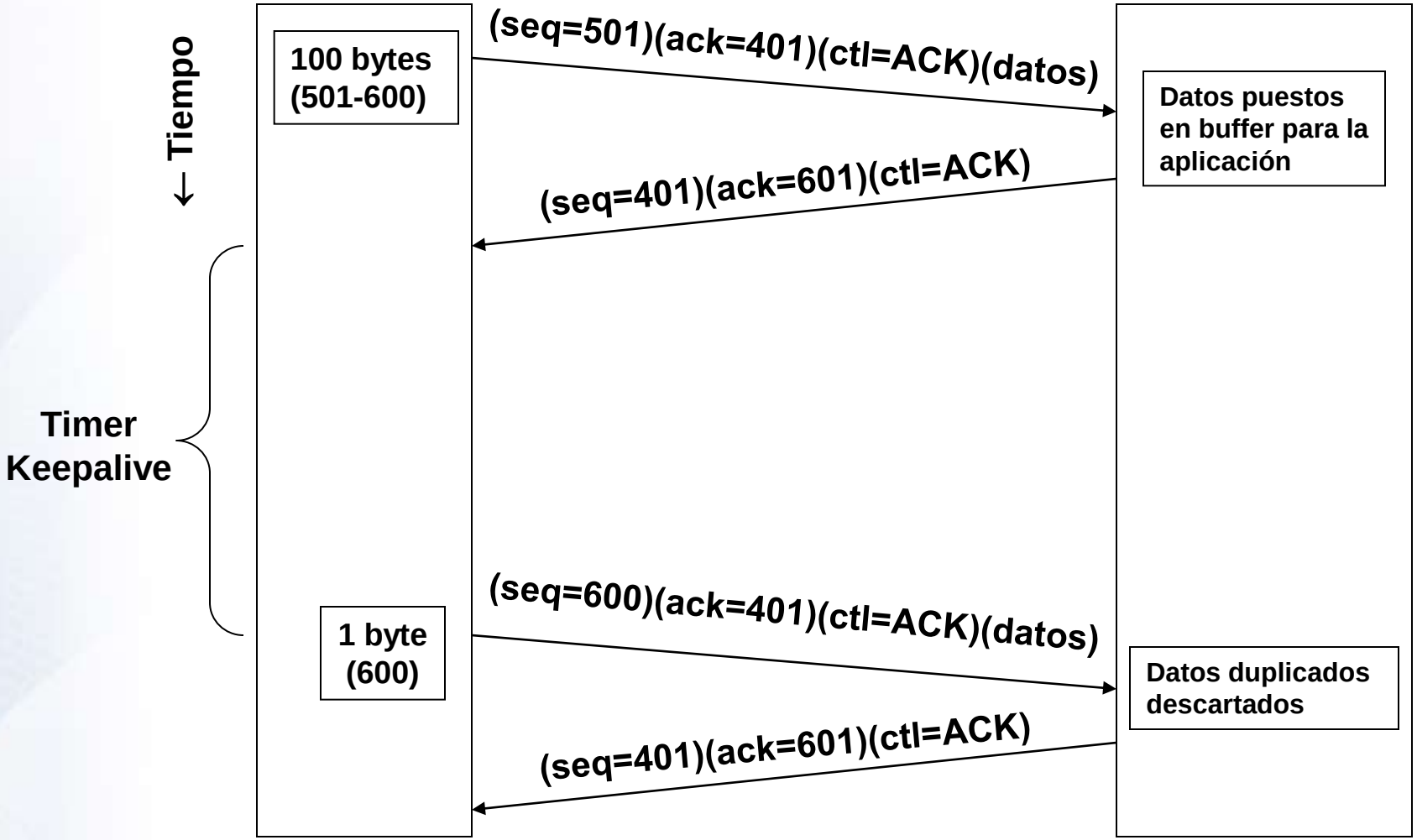
Timer de Persistencia

- Mientras la ventana está cerrada el TCP emisor puede enviar de vez en cuando un segmento con un byte de datos; esto provoca el envío de un ACK por parte del receptor y evita el bloqueo que se podría producir en caso de pérdida de un segmento anunciando una ventana mayor que cero
- La frecuencia con que el TCP emisor envía los reintentos se fija en el 'Timer de Persistencia'.

Timer de Persistencia



Mensajes de *Keepalive*



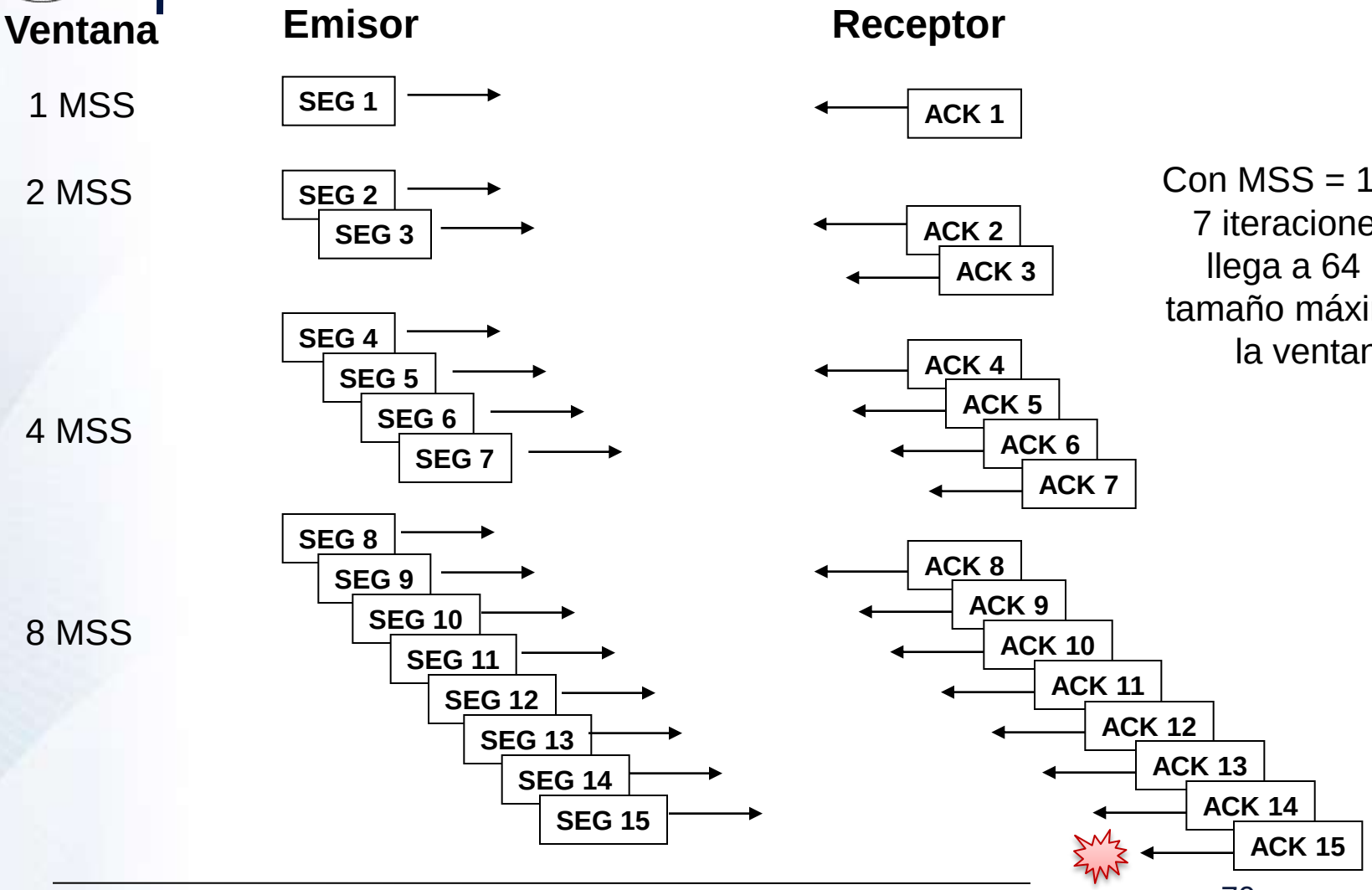
Control de congestión en TCP

- Cuando hay congestión TCP ha de reducir el flujo
- El mecanismo para detectarla es implícito, por la pérdida de segmentos. Cuando ocurre TCP baja el ritmo.
- Se presupone que la red es altamente fiable a nivel físico y que las pérdidas se deben a congestión únicamente. Cuando no es así (redes con errores) bajar el ritmo es contraproducente.
- Además de la ventana de control de flujo (dictada por el receptor y transmitida en la cabecera TCP) el emisor tiene una ventana de control de congestión, que ajusta a partir de los segmentos perdidos. En cada momento se usa la más pequeña de ambas.
- El mecanismo de control de congestión de TCP se denomina *slow-start* (arranque lento) y fue diseñado por Van Jacobson en los años 80.

Slow Start (primera fase)

- Inicialmente la ventana de congestión tiene el tamaño de un MSS (Maximum Segment Size)
- Por cada segmento enviado con éxito la ventana se amplía en un MSS
- En la práctica esto supone un crecimiento exponencial (en potencias de dos)
- Si la ventana de congestión supera a la de control de flujo se aplica ésta con lo cual aquella deja de crecer

Funcionamiento de slow start, fase 1

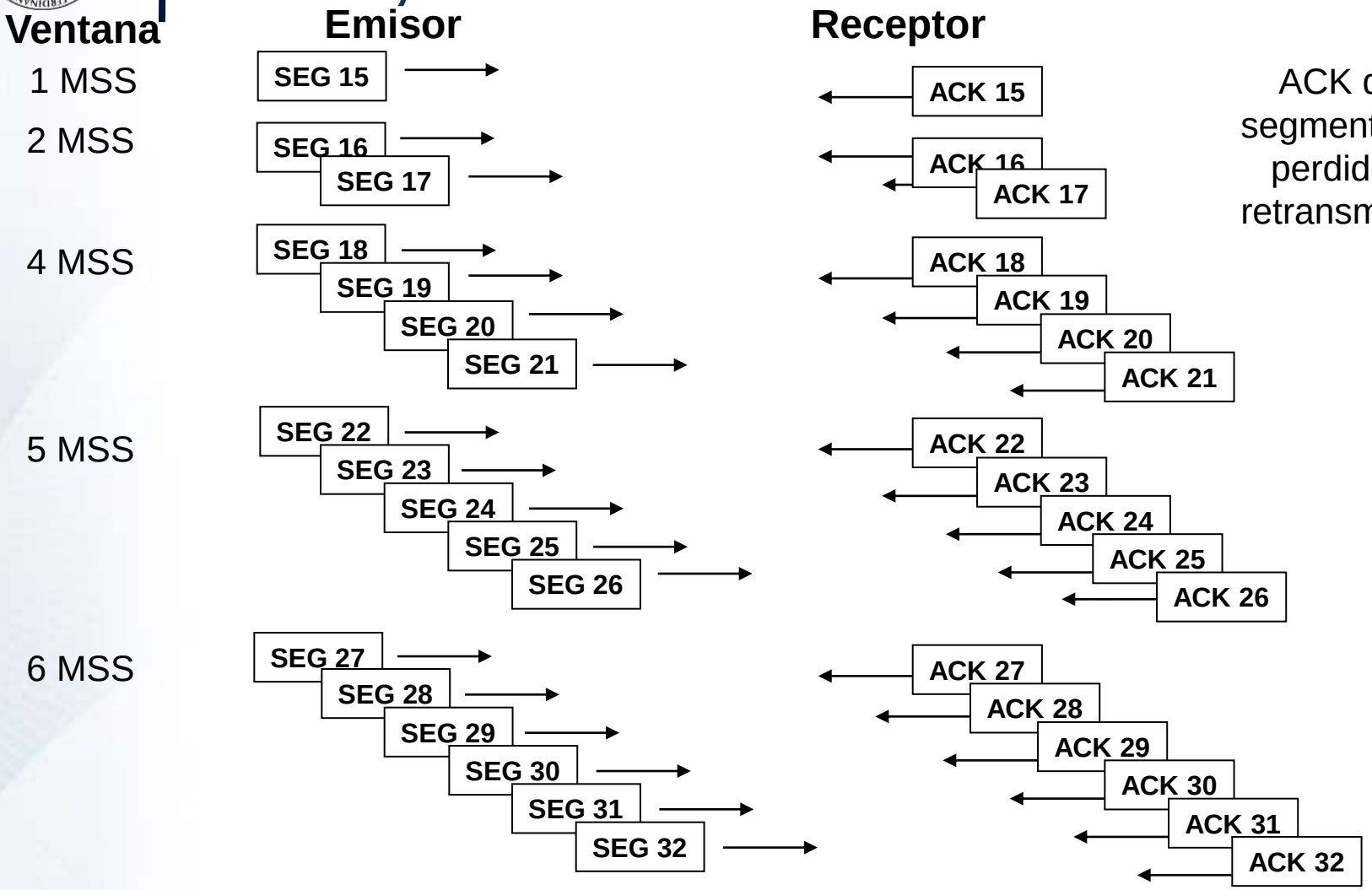


Con MSS = 1KB en 7 iteraciones se llega a 64 KB, tamaño máximo de la ventana

Slow start (segunda fase)

- Cuando se pierde un segmento:
  - La ventana de congestión vuelve a su valor inicial
  - Se fija un ‘umbral de peligro’ en un valor igual a la mitad de la ventana que había cuando se produjo la pérdida
  - La ventana de congestión crece como antes hasta el umbral de peligro; a partir de ahí crece en sólo un segmento cada vez

Funcionamiento de slow start, fase 2



ACK del segmento 15 perdido y retransmitido

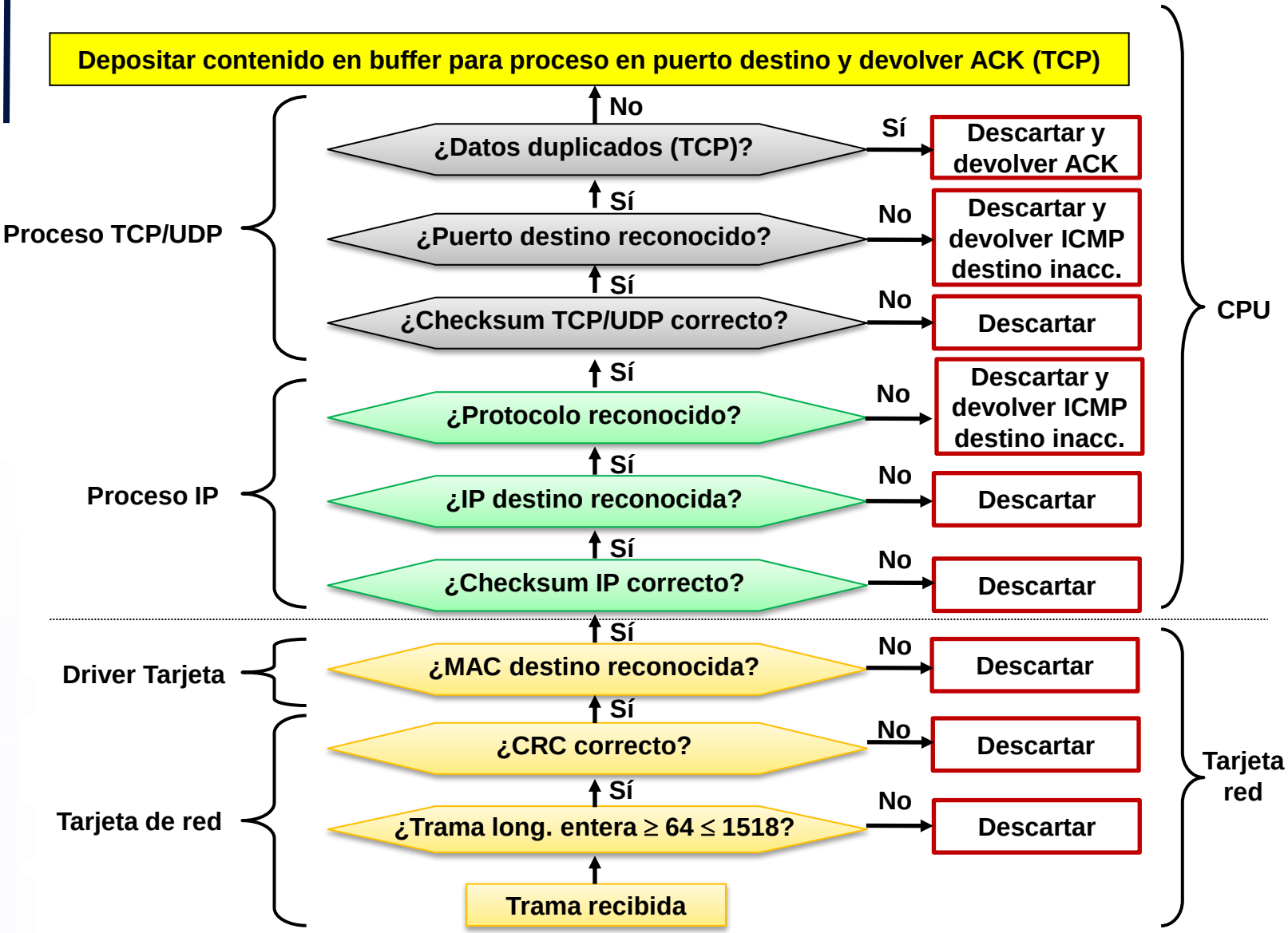
Timer de retransmisión

- Debe ser adecuado para la comunicación:
  - Si es excesivo se esperará innecesariamente
  - Si es muy corto se harán reenvíos innecesarios
- Como la fluctuación es muy grande se utilizan mecanismos autoadaptativos. A partir de los ACK se mide el tiempo de ida y vuelta o Round Trip Time (RTT)
- La estimación de este timer es crucial para el correcto funcionamiento del ‘slow-start’.

Comparación TCP - UDP

Función	**TCP**	**UDP**
Transporte	Sí	Sí
Multiplexación	Sí	Sí
Detección de errores	Sí	Opcional(*)
Corrección de errores	Sí	No
Control de flujo	Sí	No
Control de congestión	Sí	No
Establecimiento/ terminación de conexión	Sí	No

(*) Obligatorio en IPv6


Depositar contenido en buffer para proceso en puerto destino y devolver ACK (TCP)
Proceso TCP/UDP
¿Datos duplicados (TCP)?
Sí
Descartar y devolver ACK
No
¿Puerto destino reconocido?
Sí
Descartar y devolver ICMP destino inacc.
No
¿Checksum TCP/UDP correcto?
Sí
Descartar
No
Proceso IP
¿Protocolo reconocido?
Sí
Descartar y devolver ICMP destino inacc.
No
¿IP destino reconocida?
Sí
Descartar
No
¿Checksum IP correcto?
Sí
Descartar
No
CPU
Driver Tarjeta
¿MAC destino reconocida?
Sí
Descartar
No
Tarjeta de red
¿CRC correcto?
Sí
Descartar
No
¿Trama long. entera ≥ 64 ≤ 1518?
Sí
Descartar
No
Trama recibida
Tarjeta red