El Nivel de Transporte en Internet

Sumario

- **Funciones del nivel de transporte**
- Protocolo UDP
- Protocolo TCP
  - Multiplexación
  - Conexión/Desconexión
  - Intercambio de datos y control de flujo
  - Casos de baja eficiencia en TCP
  - Control de congestión

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)

Sumario

- Funciones del nivel de transporte
- **Protocolo UDP**
- Protocolo TCP
  - Multiplexación
  - Conexión/Desconexión
  - Intercambio de datos y control de flujo
  - Casos de baja eficiencia en TCP
  - Control de congestión

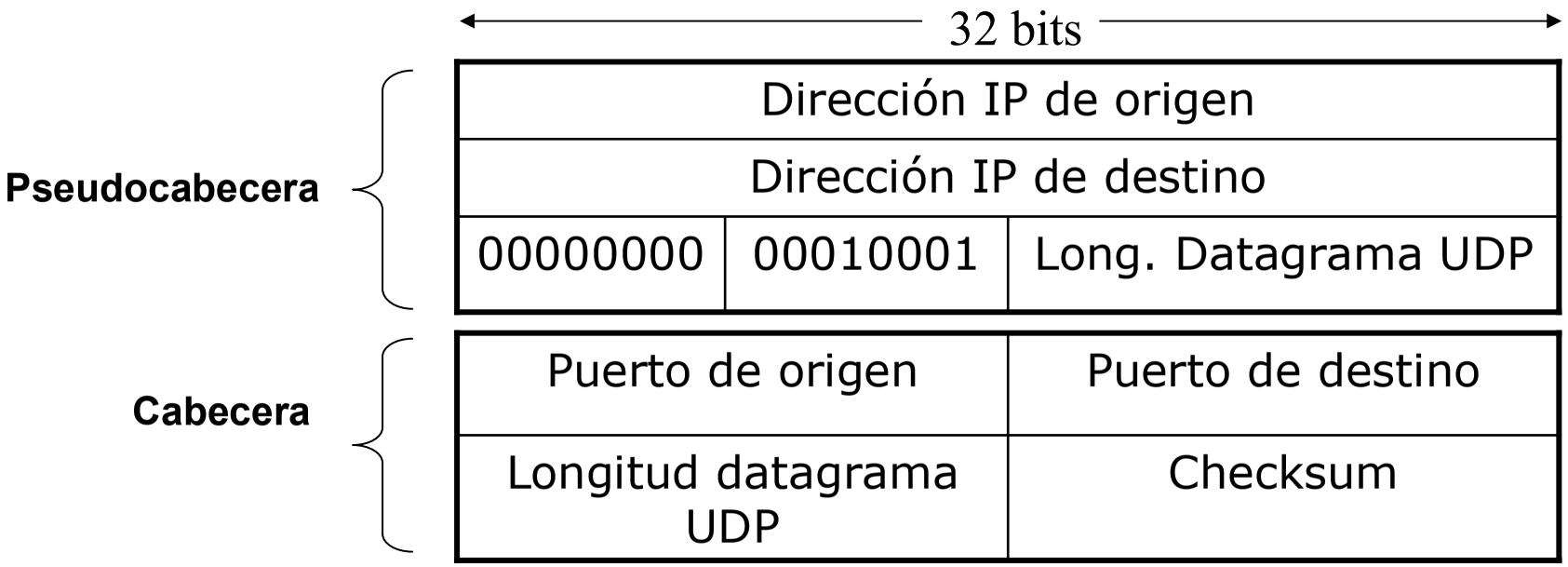
Protocolo UDP

- 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

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

La cabecera UDP

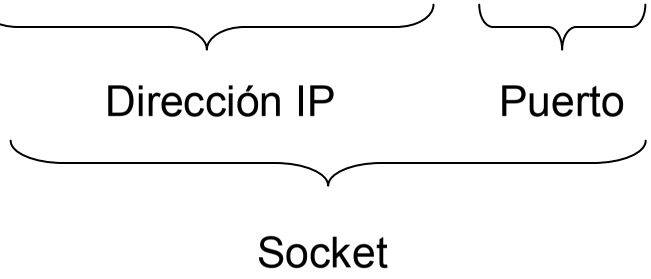


La pseudocabecera se añade al principio del datagrama para el cálculo del checksum, pero no se envía. Permite a UDP comprobar que IP no se ha equivocado (ni le ha engañado) en la entrega del datagrama.

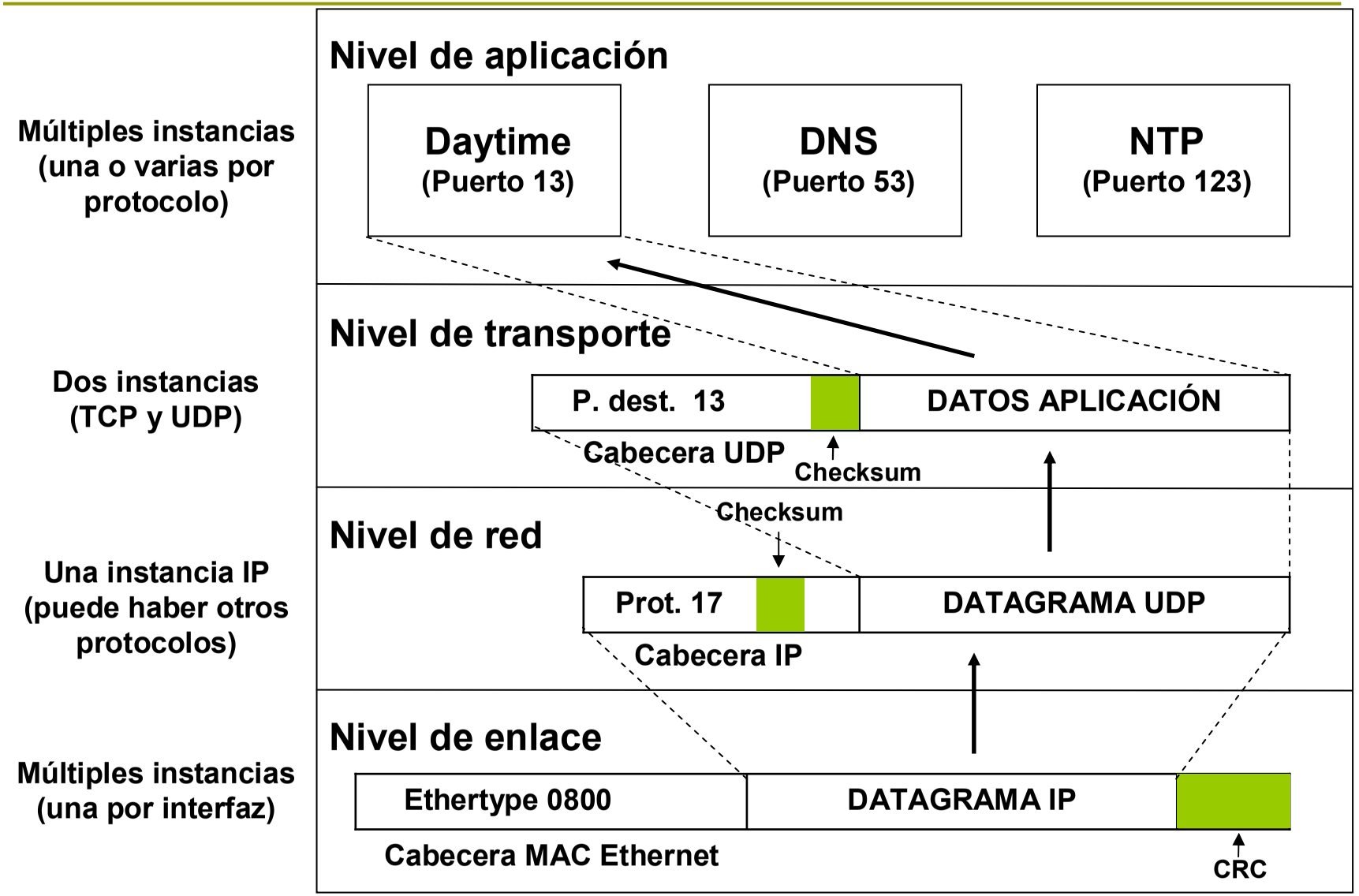
El valor $10001_2 = 17_{10}$ indica que el protocolo de transporte es UDP

Multiplexación

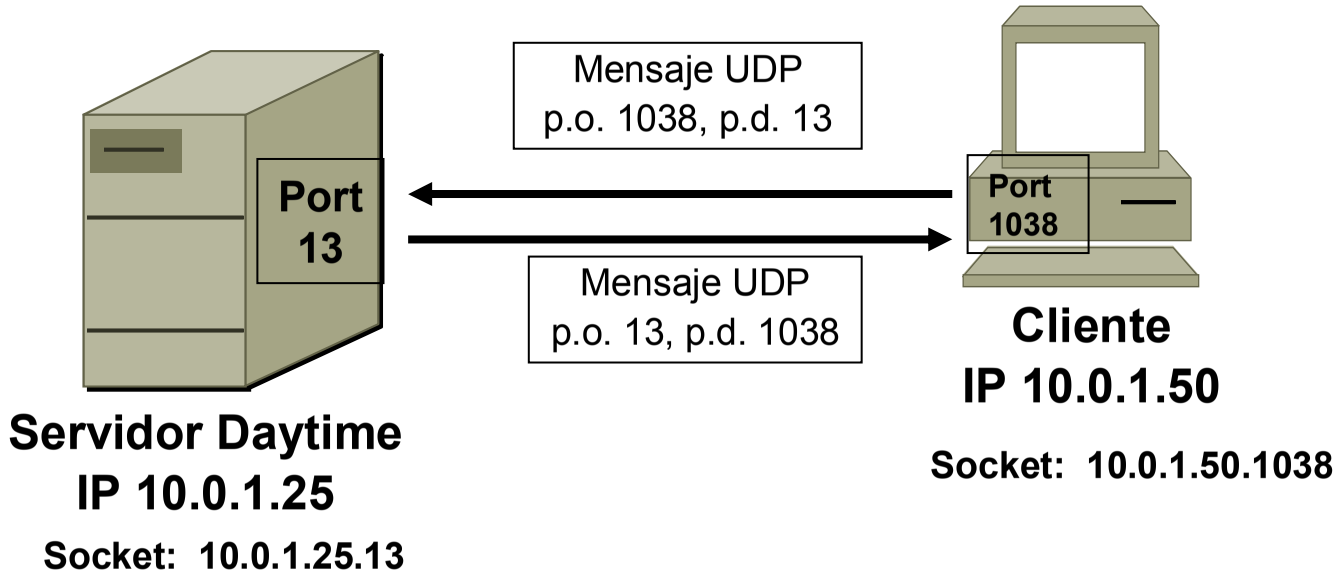
- La multiplexación se realiza mediante el puerto (origen o destino) que puede valer de 0 a 65535.
- Los puertos 0 a 1023 están reservados para servidores 'bien conocidos' ('well known ports')
- La combinación de una dirección IP y un puerto identifica un 'socket' (origen o destino de los datagramas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



Multiplexación



Conexión UDP cliente-servidor



Cabeceras IP y UDP Petición/Respuesta SNMP

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

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

Sumario

- Funciones del nivel de transporte
- Protocolo UDP
- **Protocolo TCP**
  - Multiplexación
  - Conexión/Desconexión
  - Intercambio de datos y control de flujo
  - Casos de baja eficiencia en TCP
  - Control de congestión

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.

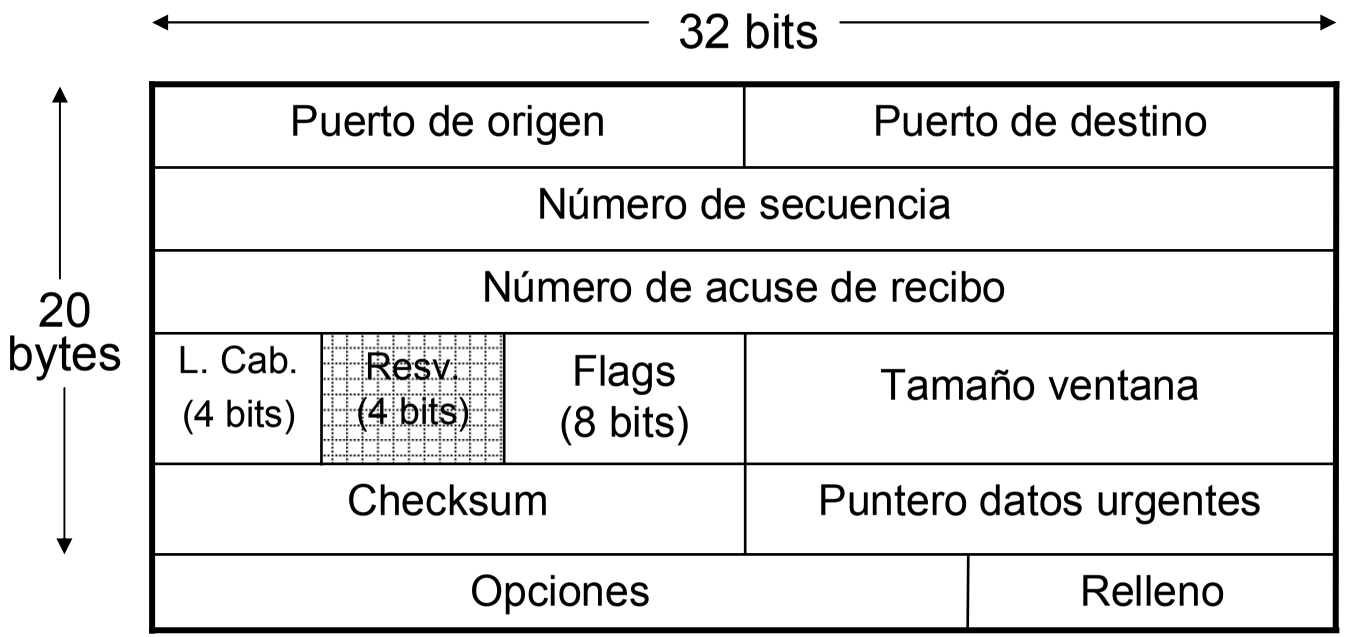
Servicio orientado a conexión

- Los servicios orientados a conexión requieren un procedimiento explícito de establecimiento y terminación de la comunicación.
- Durante la conexión las entidades participantes mantienen en memoria una información relativa a dicha conexión (contadores de bytes, espacio libre en buffers, etc.). Dicha información se conoce como información de estado.
- Para describir los servicios orientados a conexión se suele utilizar un modelo basado en dos protagonistas:
  - **Cliente**: el que inicia la conexión
  - **Servidor**: el que está a la espera de recibir peticiones de conexión
- Una conexión puede terminarse tanto por iniciativa del cliente como del servidor.
- También hay aplicaciones que utilizan el modelo igual a igual (peer-to-peer) como Emule, Edonkey, etc.

Funciones de TCP

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

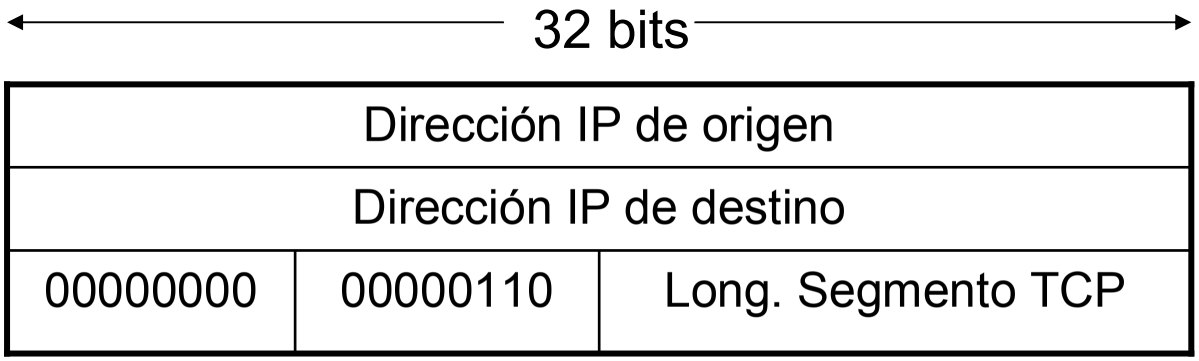
La cabecera TCP



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

La pseudocabecera TCP



Se añade al principio del segmento solo para el cálculo del Checksum, **no se envía**. Permite a TCP comprobar que IP no se ha equivocado (ni le ha engañado) en la entrega del segmento.

El valor $110_2 = 6_{10}$ indica que el protocolo de transporte es TCP

Sumario

- Aspectos generales del nivel de transporte
- Protocolo UDP
- Protocolo TCP
  - **Multiplexación**
  - Conexión/Desconexión
  - Intercambio de datos y control de flujo
  - Casos de baja eficiencia en TCP
  - Control de congestión

Multiplexación

- Se utiliza el número de puerto (origen o destino) como en UDP. Puede valer de 0 a 65535.
- Como en UDP los puertos 0 a 1023 están reservados para servidores 'bien conocidos'
- Como en UDP 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	

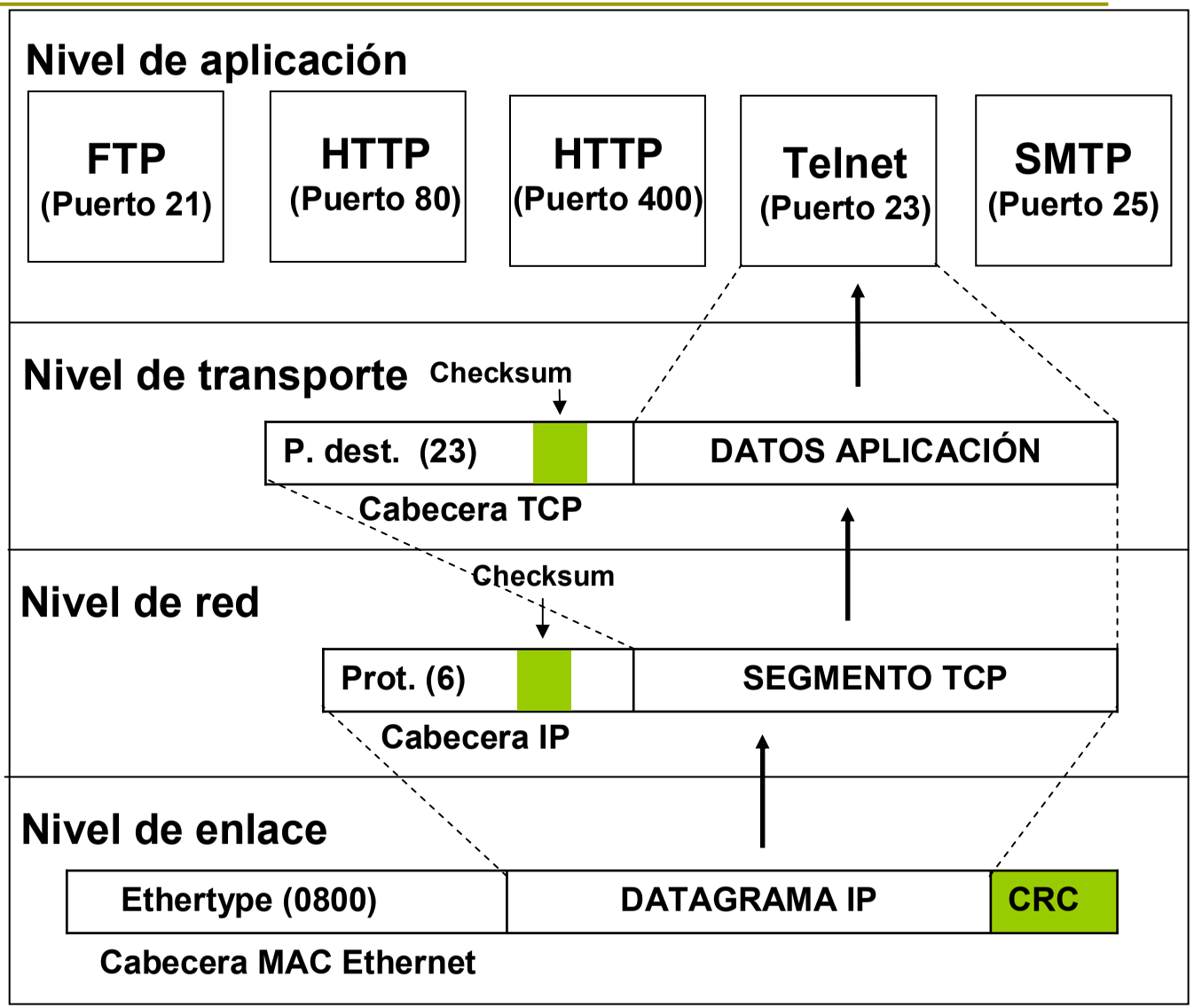
Multiplexación

Múltiples instancias (una o varias por protocolo)

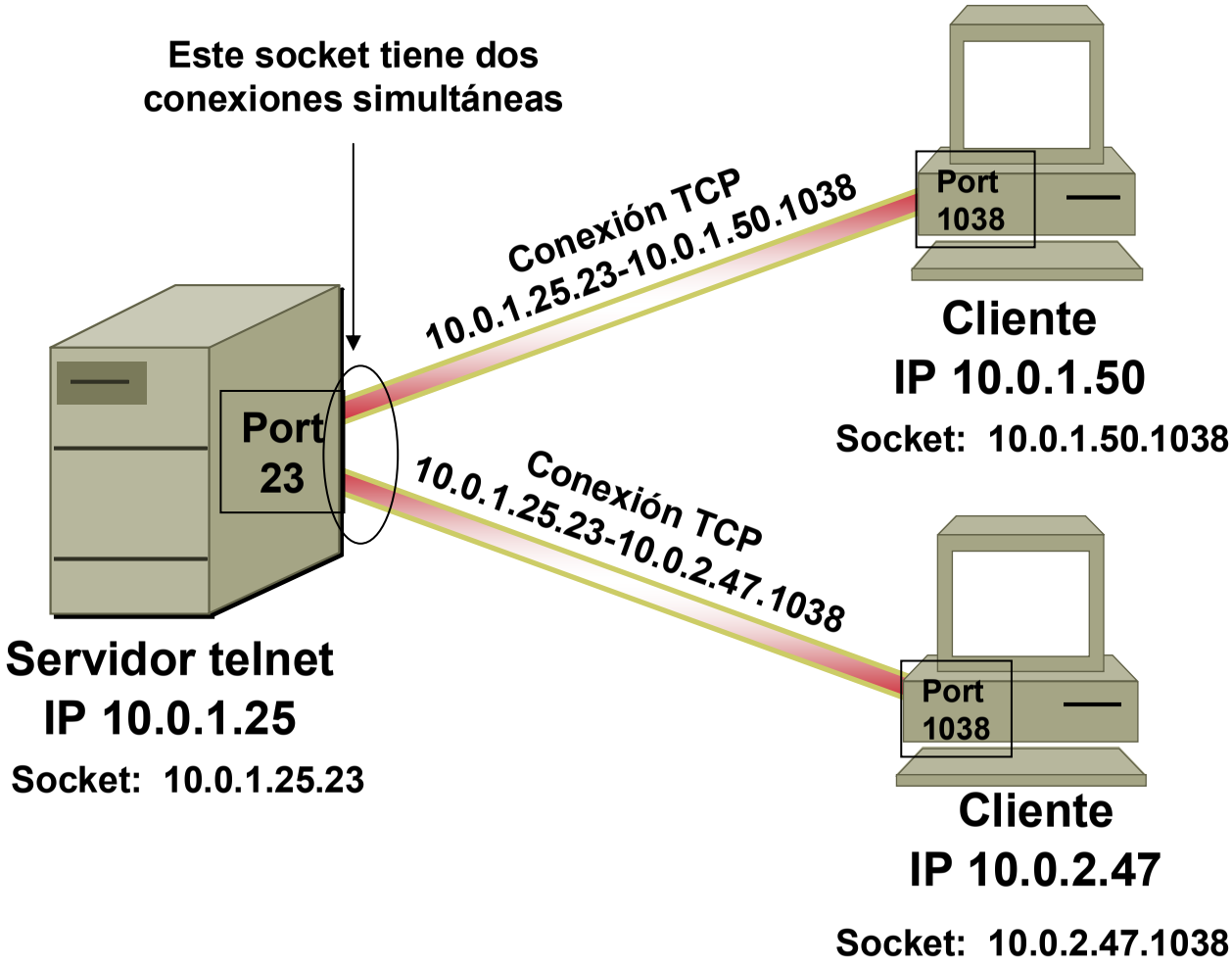
Dos instancias (TCP y UDP)

Una instancia IP (puede haber otros protocolos)

Múltiples instancias (una por interfaz)

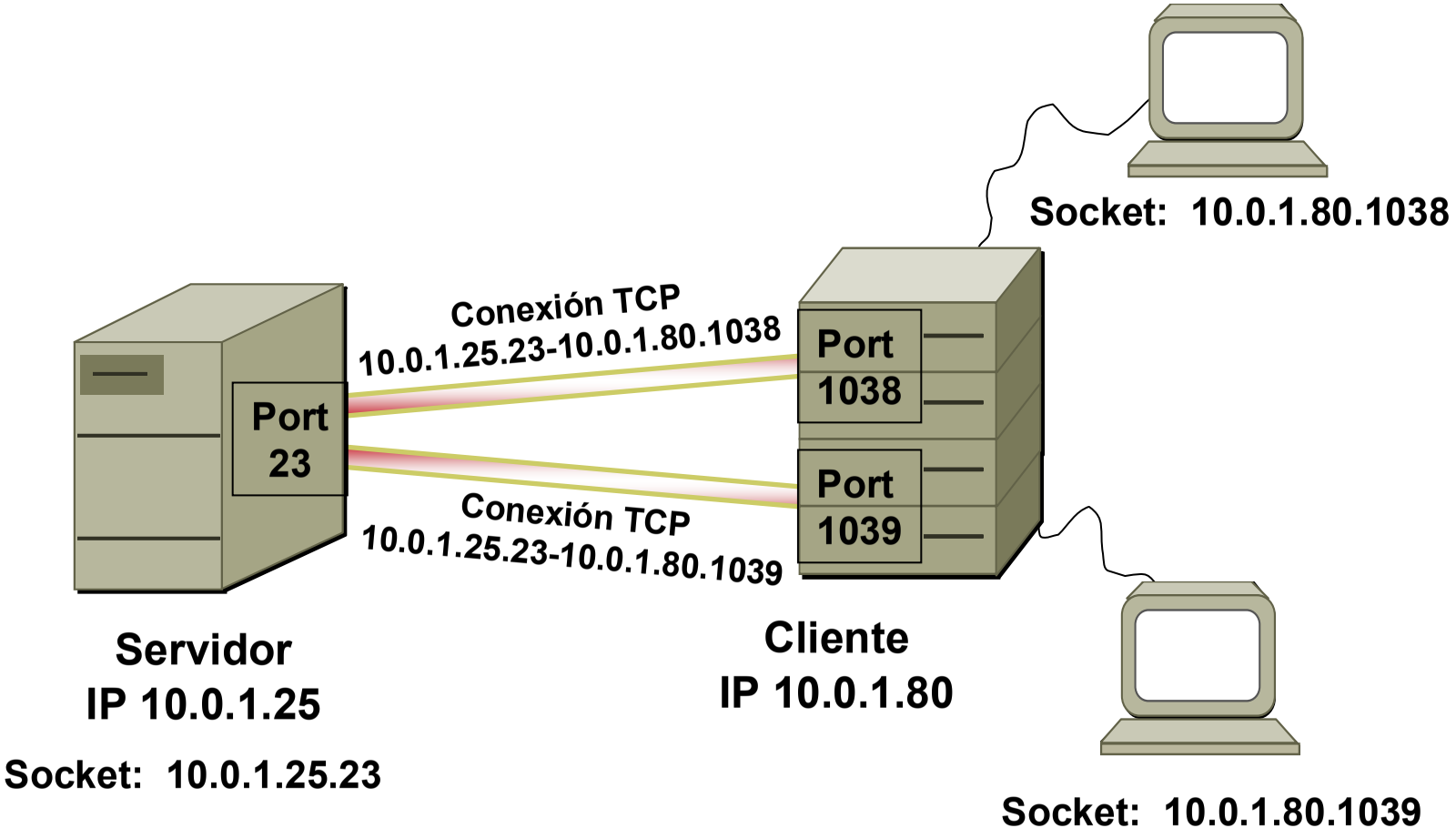


Dos conexiones TCP mismo Port



A un mismo socket desde dos sockets con el mismo número de puerto

Dos conexiones TCP mismo IP



A un mismo socket desde dos sockets con la misma dirección IP

Conexiones TCP

(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                 *.*
```

Sumario

- Aspectos generales del nivel de transporte
- Protocolo UDP
- Protocolo TCP
  - Multiplexación
  - **Conexión/Desconexión**
  - Intercambio de datos y control de flujo
  - Casos de baja eficiencia en TCP
  - Control de congestión
  - Opciones de TCP

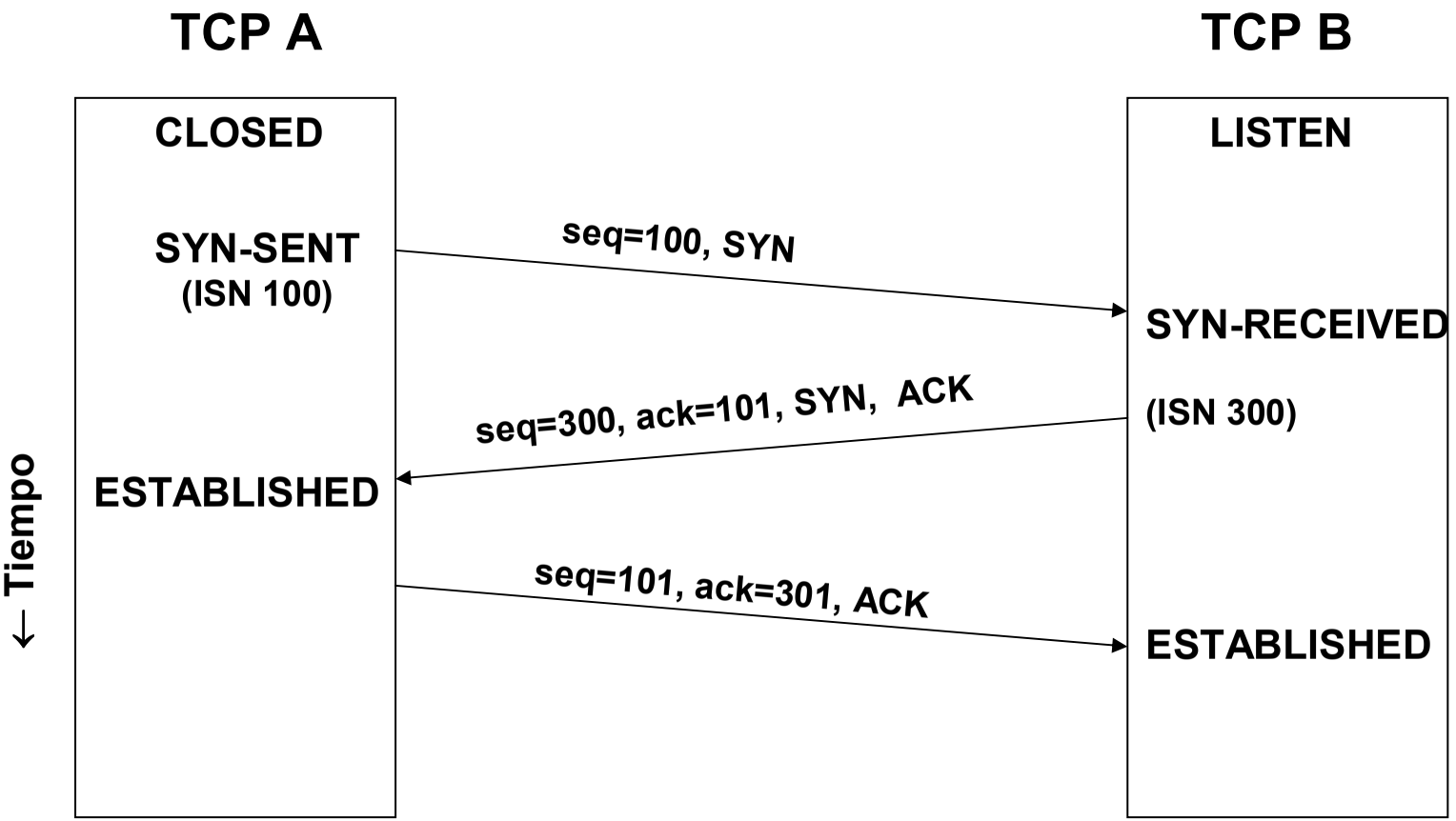
Conexión por 'Saludo a tres vías'

- 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 lo 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.

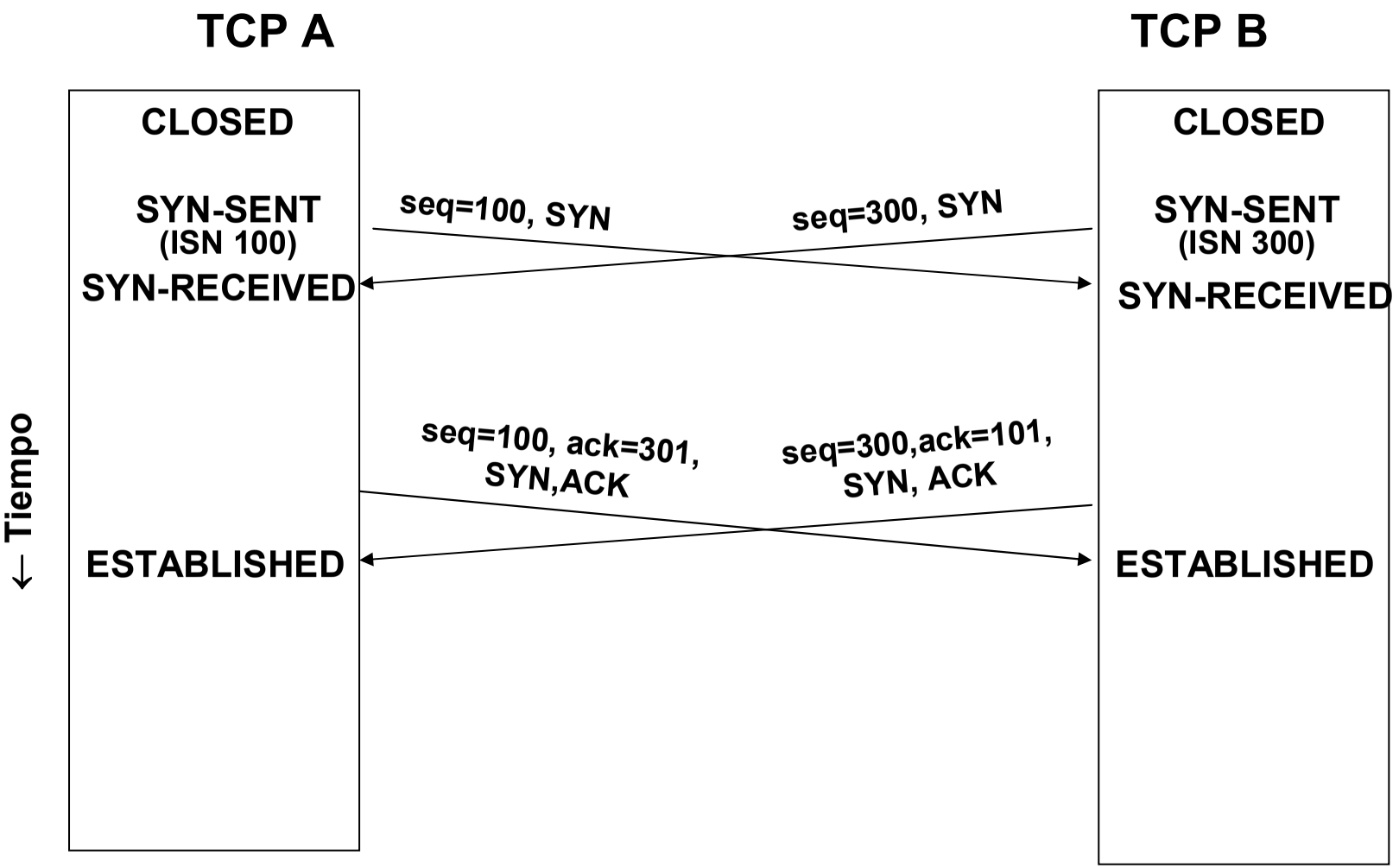
Procedimiento del saludo a tres vías

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.

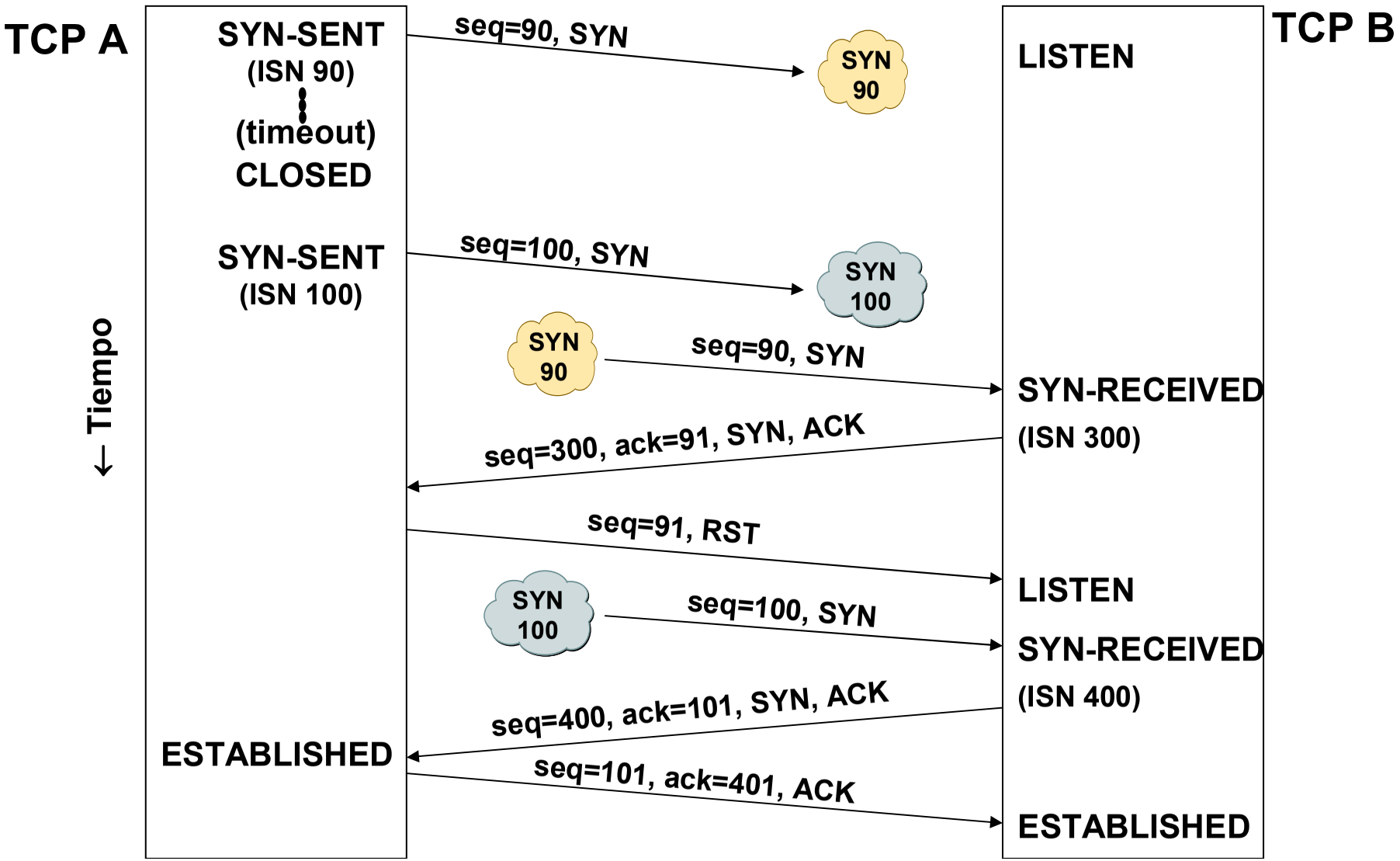
Establecimiento Conexión TCP
Saludo a tres vías



Saludo a tres vías
Conexión simultánea



Conexión con SYN duplicado



Conexión en TCP

- Los dos primeros segmentos de la conexión se identifican con el flag SYN.
- El número de secuencia es un campo de 32 bits que cuenta bytes en módulo 2^{32} (el contador se da la vuelta cuando llega al valor máximo).
- El número de secuencia no empieza normalmente en 0, sino en un valor denominado ISN (Initial Sequence Number) elegido al azar; el ISN sirve de 'PIN' en el saludo a tres vías para asegurar la autenticidad de la comunicación.
- Una vez establecida la comunicación el 'seq' y el 'ack' sirven para contar los bytes transmitidos y recibidos.

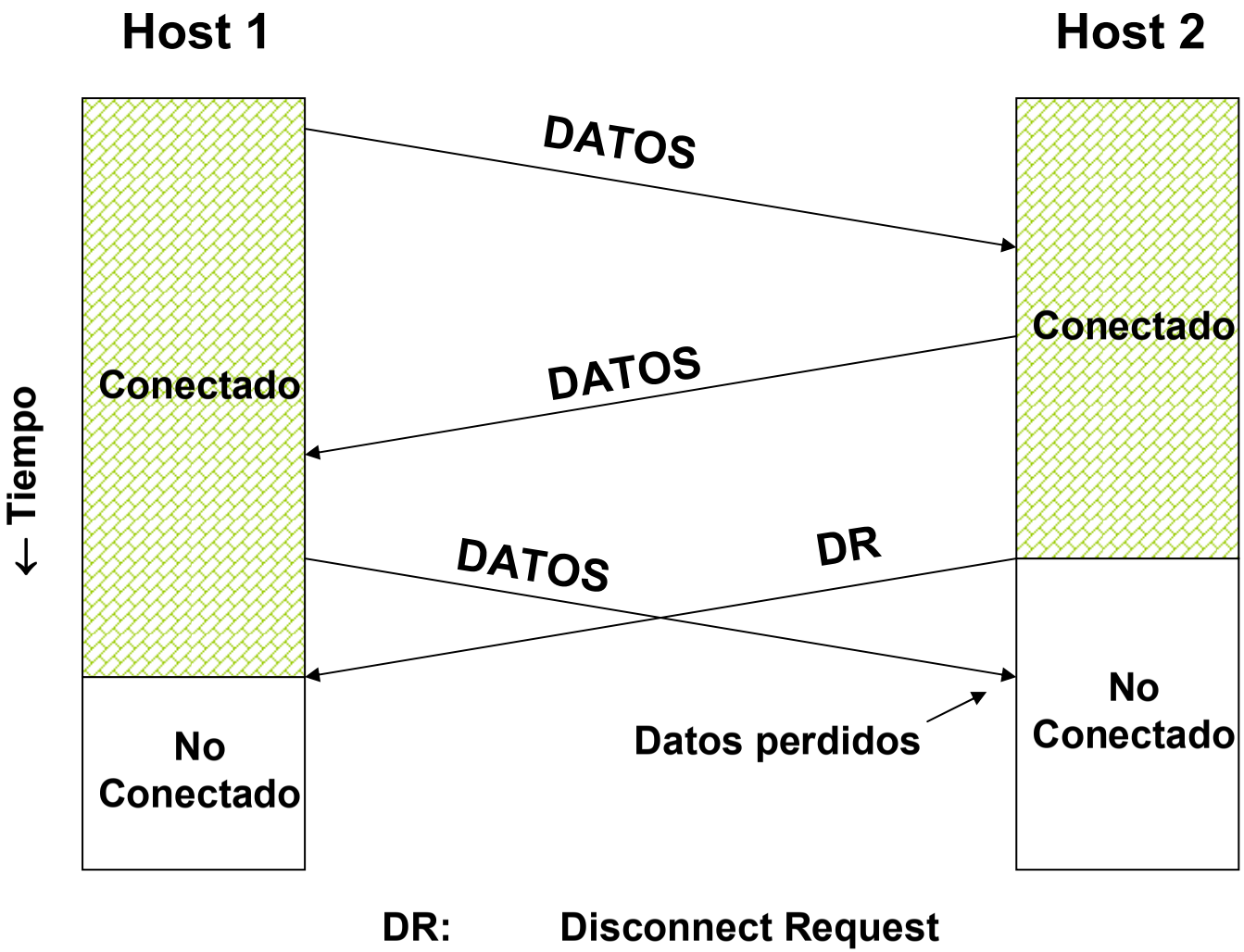
Conexión en TCP

- El ISN es elegido por el sistema (cliente o servidor). El estándar sugiere utilizar un contador entero incrementado en 1 cada 4 μs aproximadamente. En este caso el contador se da la vuelta (y el ISN reaparece) al cabo de 4 horas 46 min.
- El MSL (Maximum Segment Lifetime) típico es de unos 2 minutos, con lo que la probabilidad de que dos ISN coincidan es despreciable.
- El mecanismo de selección de los ISN es suficientemente fiable para proteger de coincidencias debidas al azar, pero no es un mecanismo de protección frente a sabotajes. Es muy fácil averiguar el ISN de una conexión e interceptarla suplantando a alguno de los dos participantes.

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



Mensaje de Desconexión

- EL mensaje solicitando la desconexión se puede perder. Por eso se pide una confirmación (ACK).
- Pero la confirmación también podría perderse, por lo que habría que enviar una reconfirmacion, y así sucesivamente.
- Este problema no tiene solución infalible, pues estamos usando un canal no fiable para asegurar un envío de información. Es lo que se conoce como el problema de los dos ejércitos.

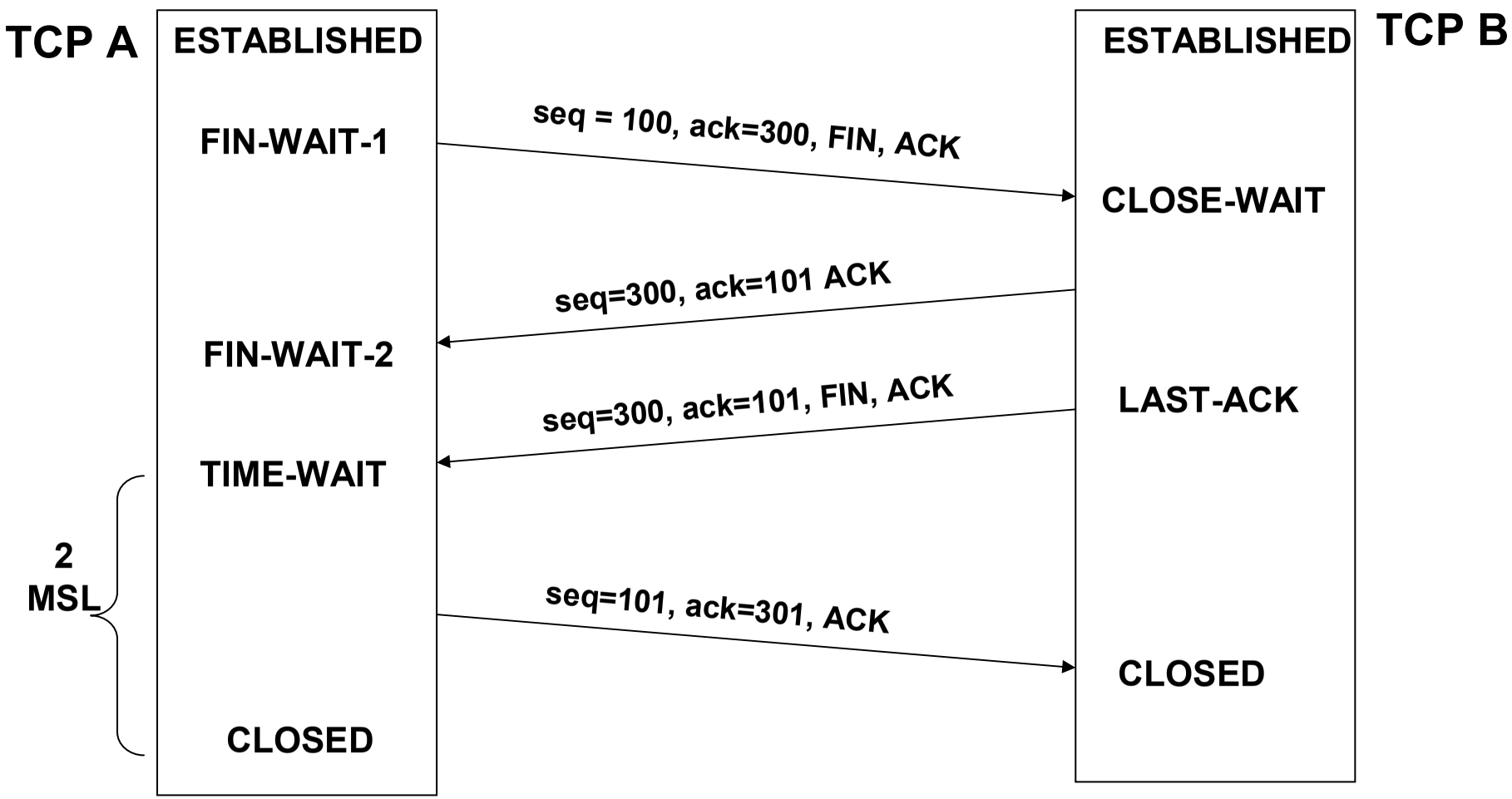
Desconexión por saludo a tres vías

- 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 en TCP

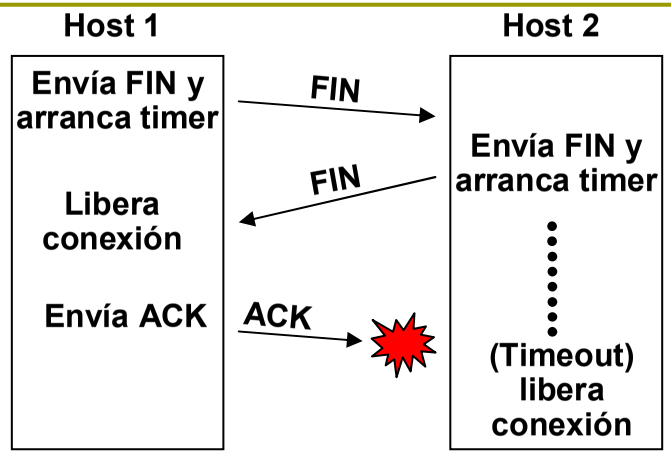
- Se utiliza el ‘saludo a tres vías’ invitando a la otra parte a cerrar.
- Para indicar el cierre se utiliza el flag FIN
- La desconexión puede iniciarla cualquiera de los dos TCP (el cliente o el servidor).
- Una vez efectuada la desconexión el host que inició el proceso está un cierto tiempo a la espera por si aparecen segmentos retrasados

Desconexión a tres vías, caso normal

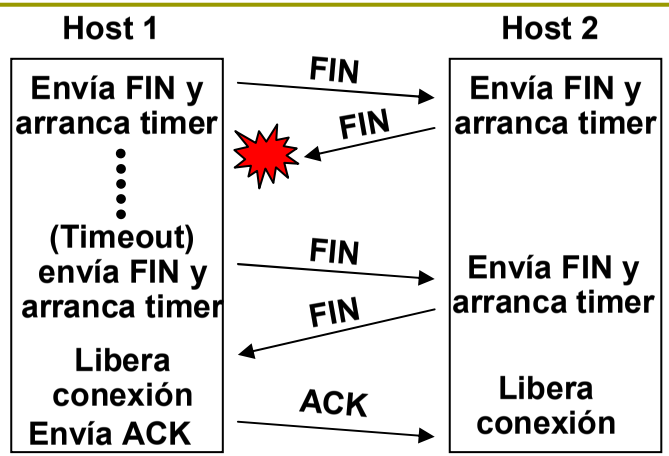


MSL: Maximum Segment Lifetime (normalmente 2 minutos)

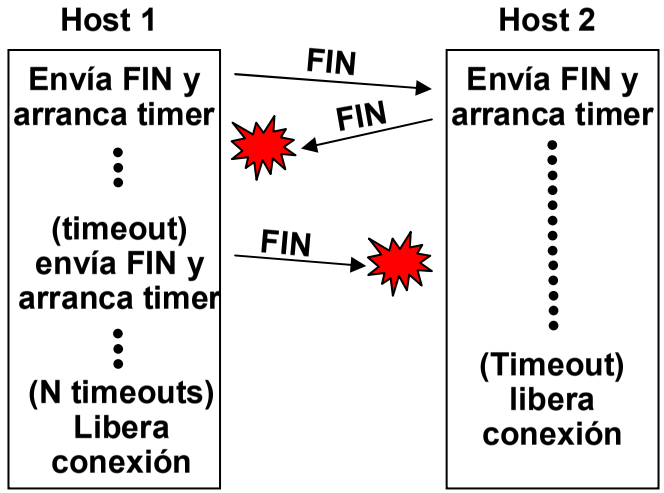
Desconexión a tres vías, casos anormales



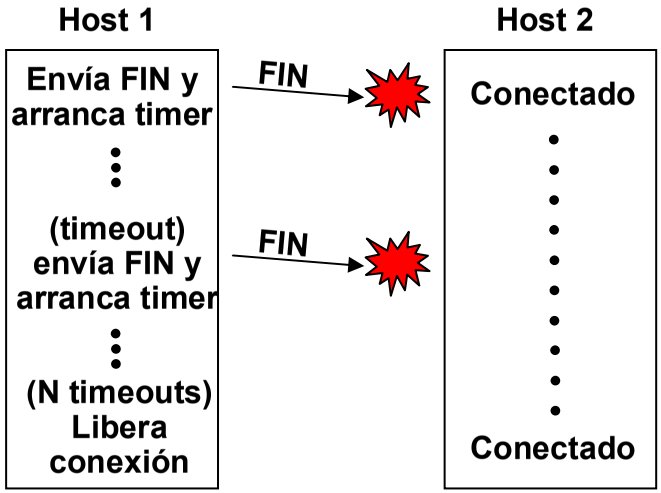
Pérdida de ACK final



Pérdida de respuesta FIN



Pérdida de todo menos primer FIN



Pérdida de todos los FIN de host 1

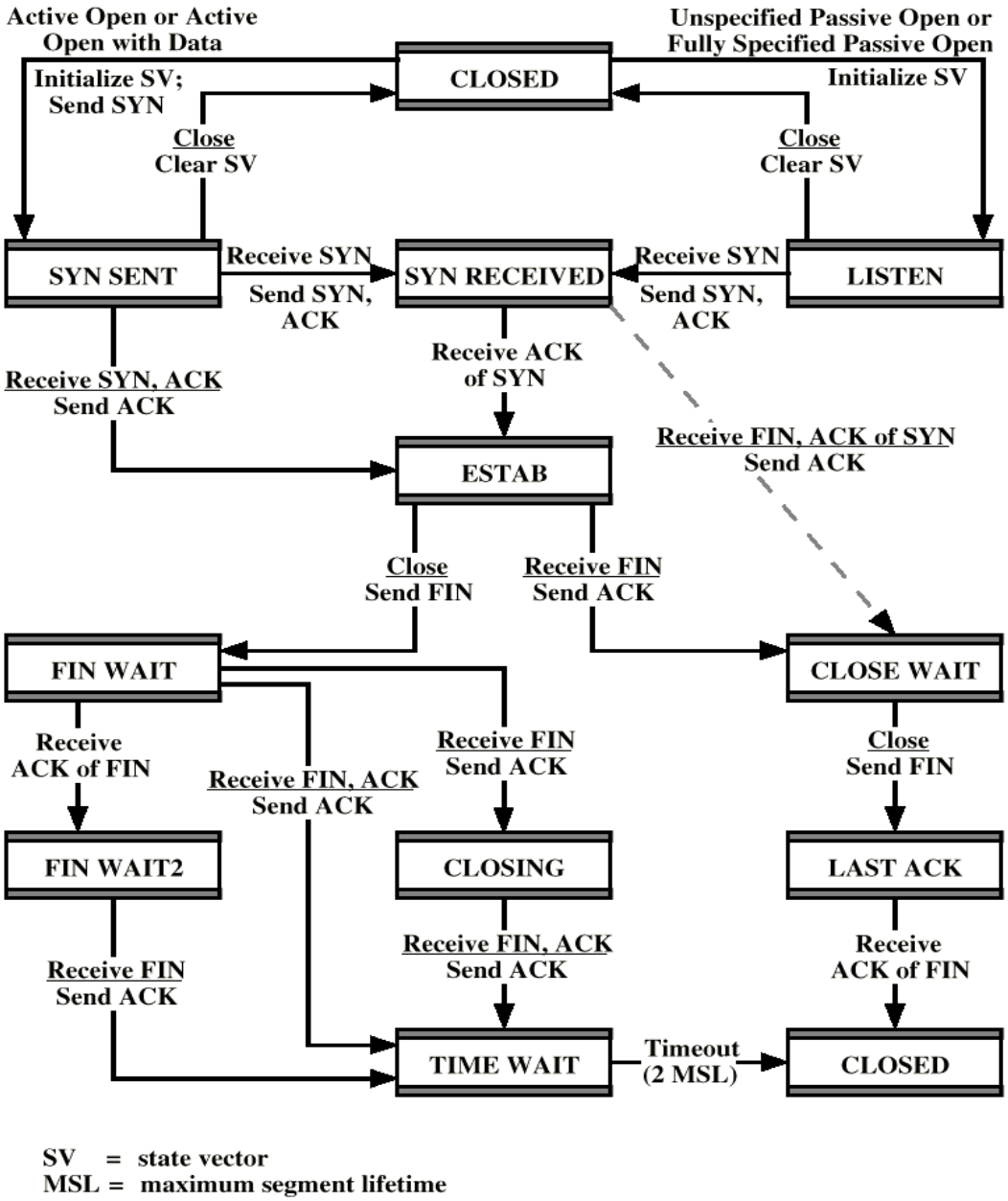


Figure 17.9 TCP Entity State Diagram

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 lleva un byte de datos 'virtual'
- La presencia del flag ACK no incrementa el número de secuencia

Sumario

- Funciones del nivel de transporte
- Protocolo UDP
- Protocolo TCP
  - Multiplexación
  - Conexión/Desconexión
  - **Intercambio de datos y control de flujo**
  - Casos de baja eficiencia en TCP
  - Control de congestión

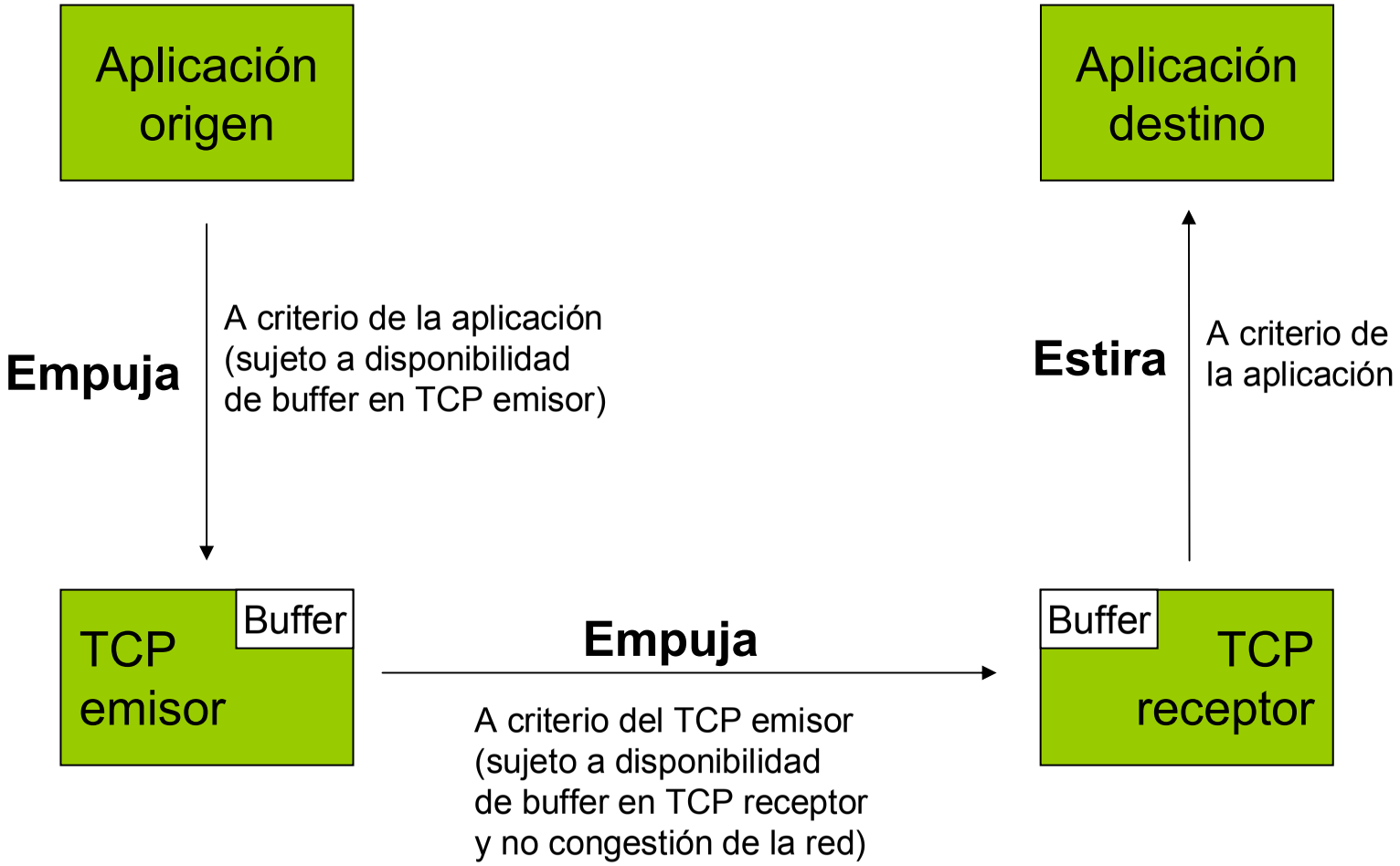
Intercambio de datos TCP ↔ aplicación

- **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.

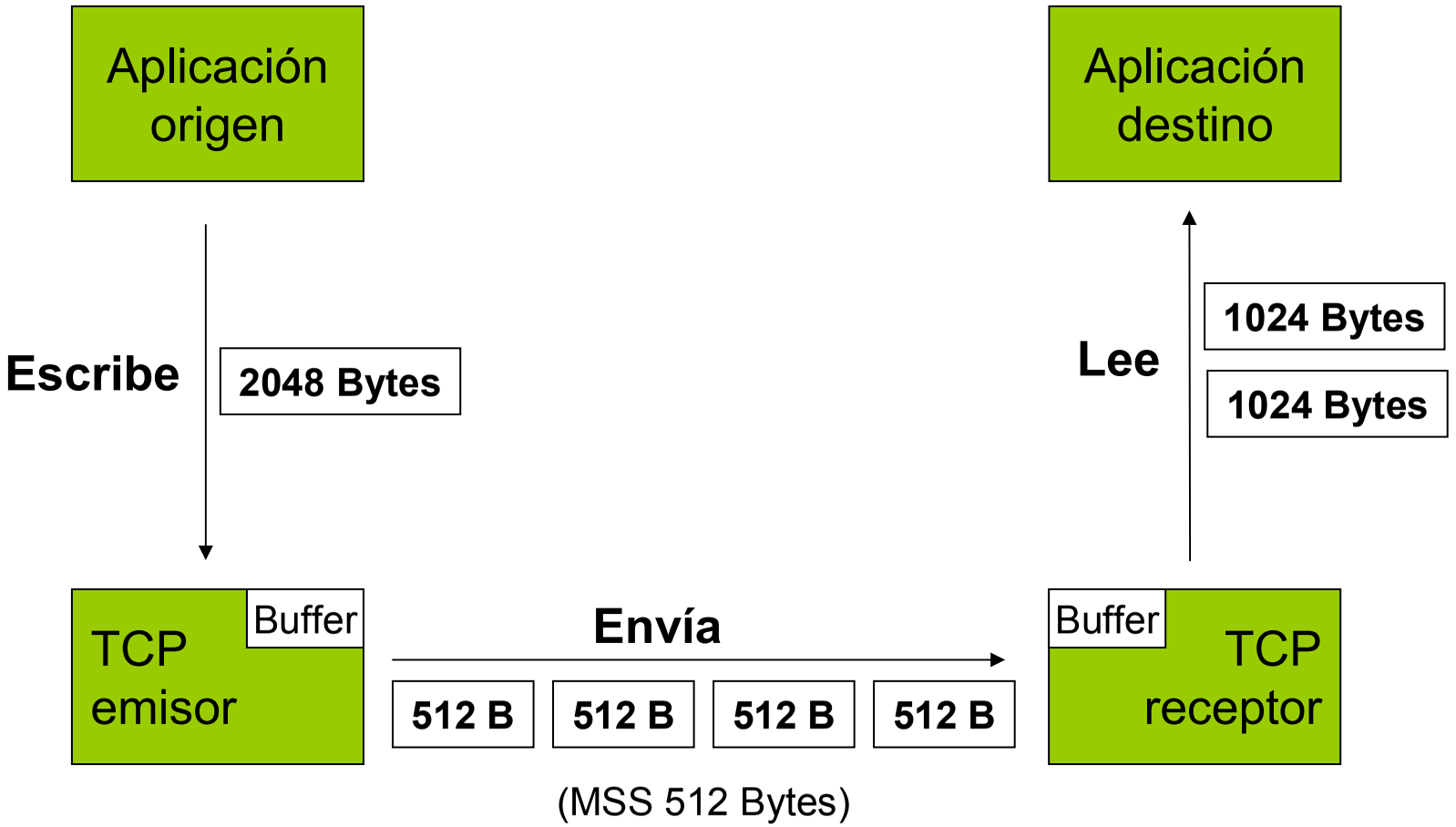
Intercambio de datos TCP ↔ TCP

- El TCP emisor manda los datos cuando quiere. Excepción: datos 'Pushed'
- 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 y TCP ↔ TCP



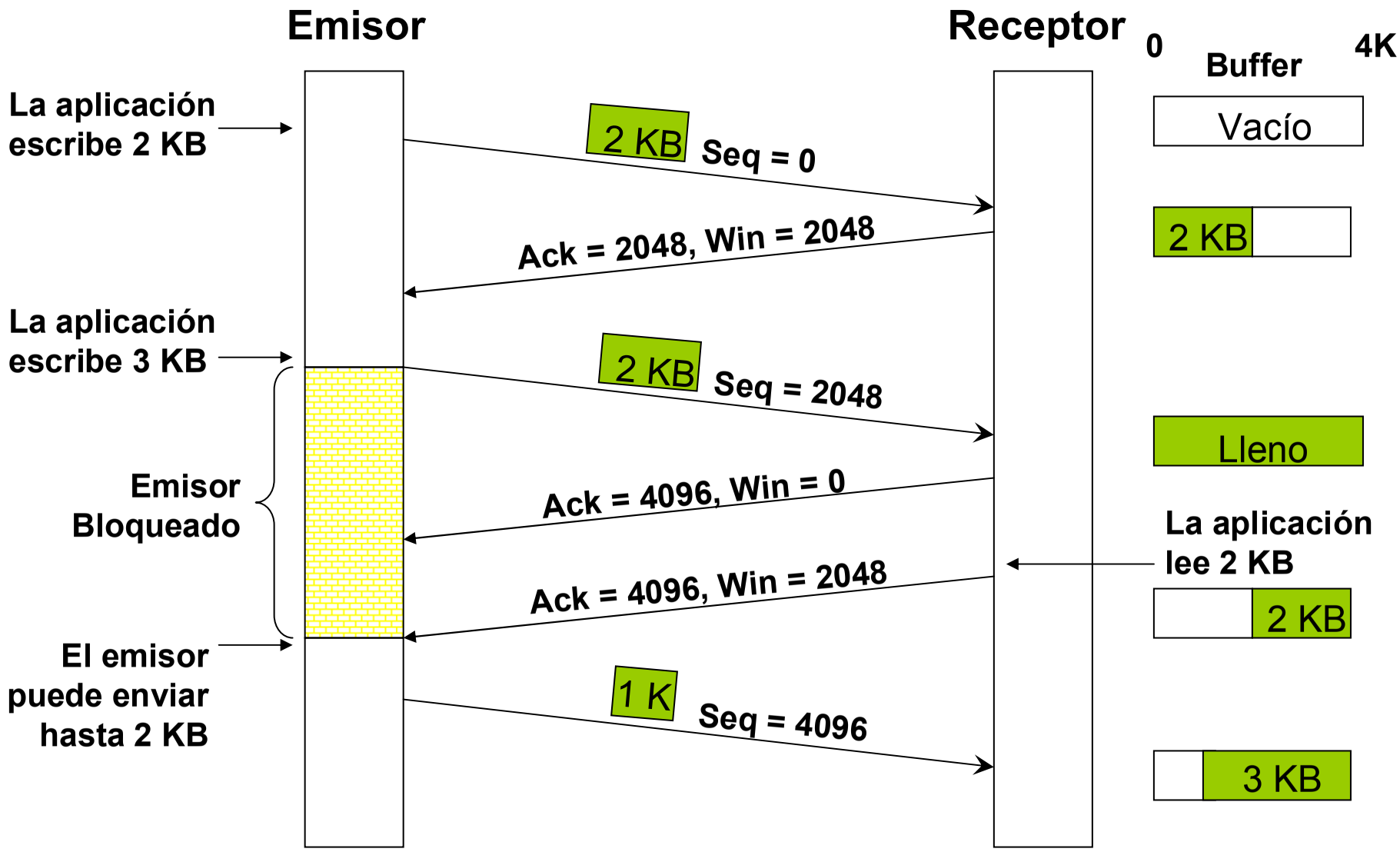
Intercambio de datos
TCP ↔ Aplicación y TCP ↔ TCP



Gestión de buffers y Control de Flujo

- El TCP receptor informa en cada segmento al emisor del espacio que le queda libre en el buffer para esa comunicación. Para ello usa el campo tamaño de ventana.
- Anunciando una ventana cero el receptor puede bloquear al emisor, y ejercer así control de flujo.
- La ventana anunciada es un espacio que el TCP receptor reserva para esa comunicación en su buffer.
- Tanto los números de secuencia como los tamaños de ventana cuentan bytes.

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').

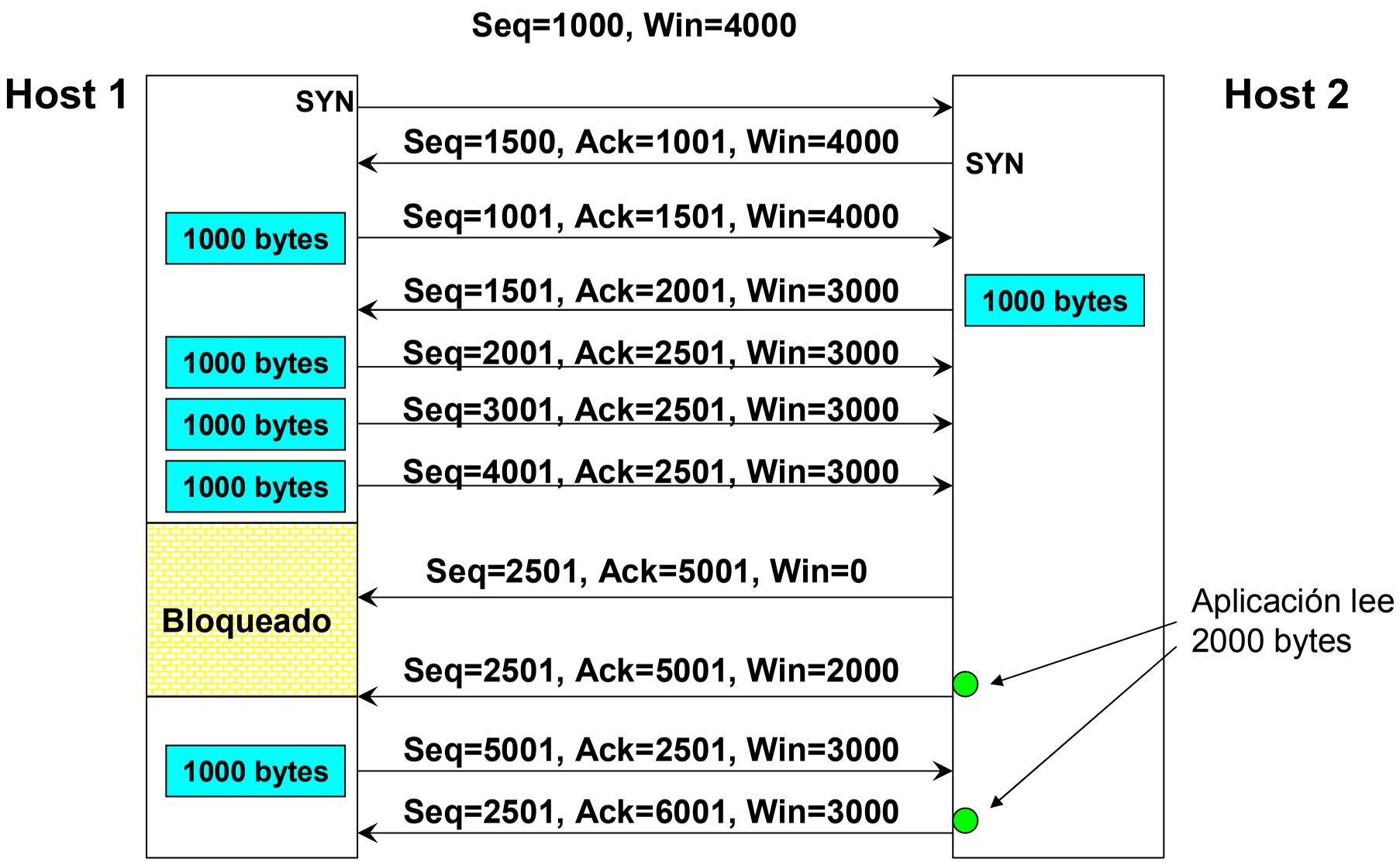
(Recordemos la Ley de Postel):

'Sé estricto al enviar y tolerante al recibir'

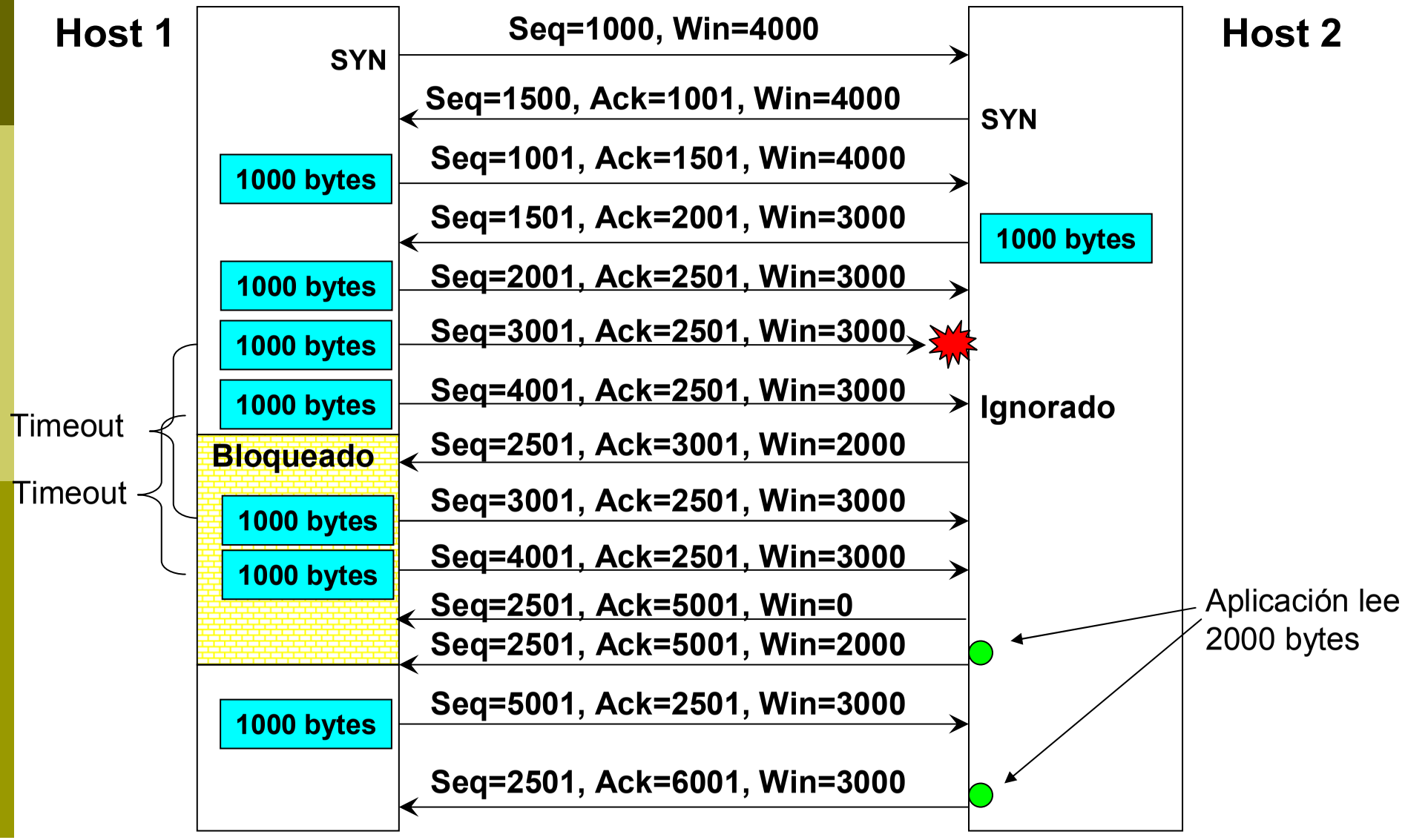
Reenvío de segmentos

- En caso de pérdida de un paquete en la red el segmento TCP no llegará a su destino
- Cada TCP cuando envía un segmento espera recibir el ACK; si este no llega dentro de un tiempo razonable reenvía el segmento.
- Si se enviaron varios segmentos y se pierde uno se puede hacer dos cosas:
  - Enviar solo ese segmento (repetición selectiva)
  - Enviar todos los segmentos a partir de ese (retroceso n)
- Lo normal es utilizar retroceso n

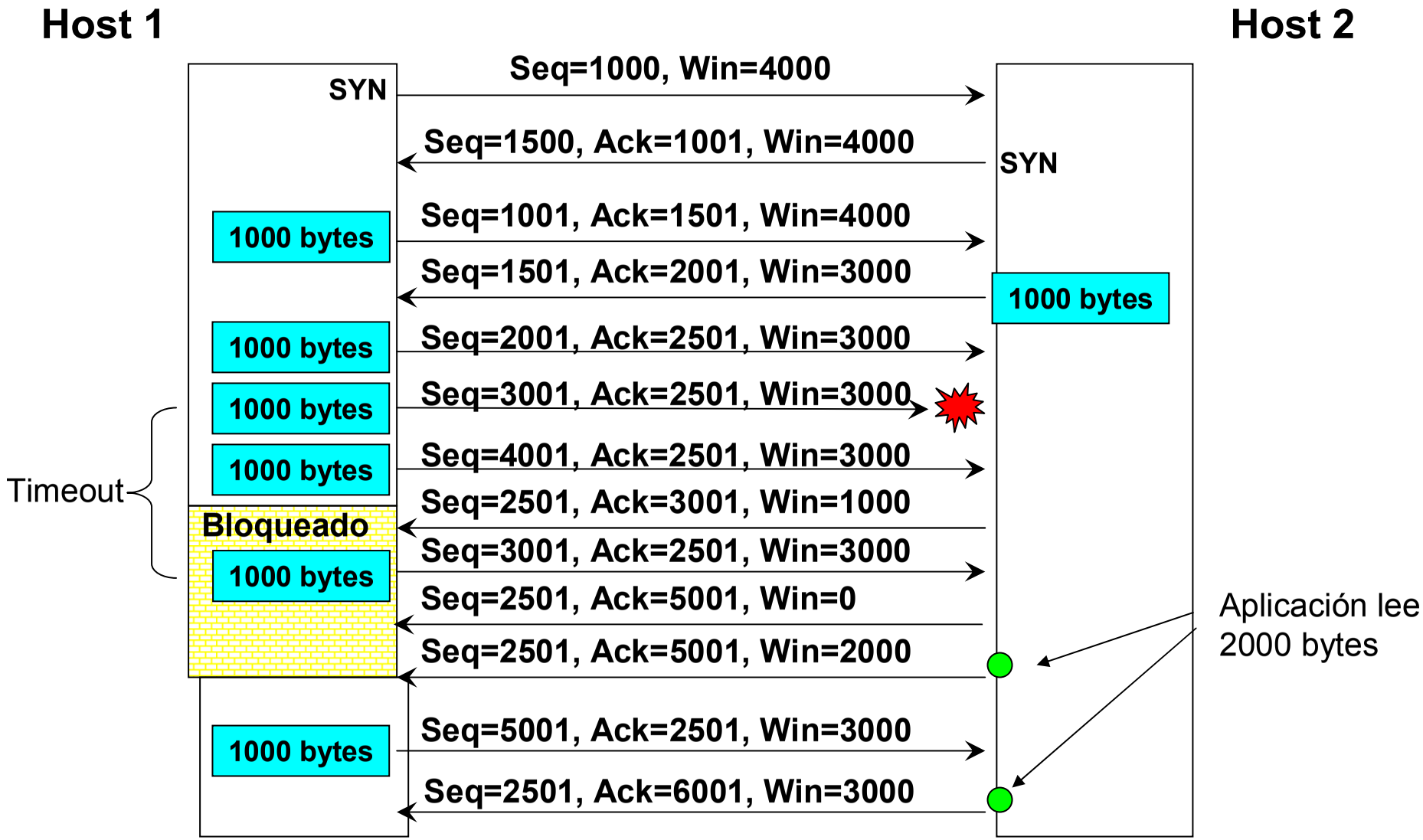
Control de flujo y números de secuencia
Caso normal, sin pédidas



Pérdida de un paquete
Retransmisión con retroceso n



Pérdida de un paquete
Retransmisión con repetición selectiva



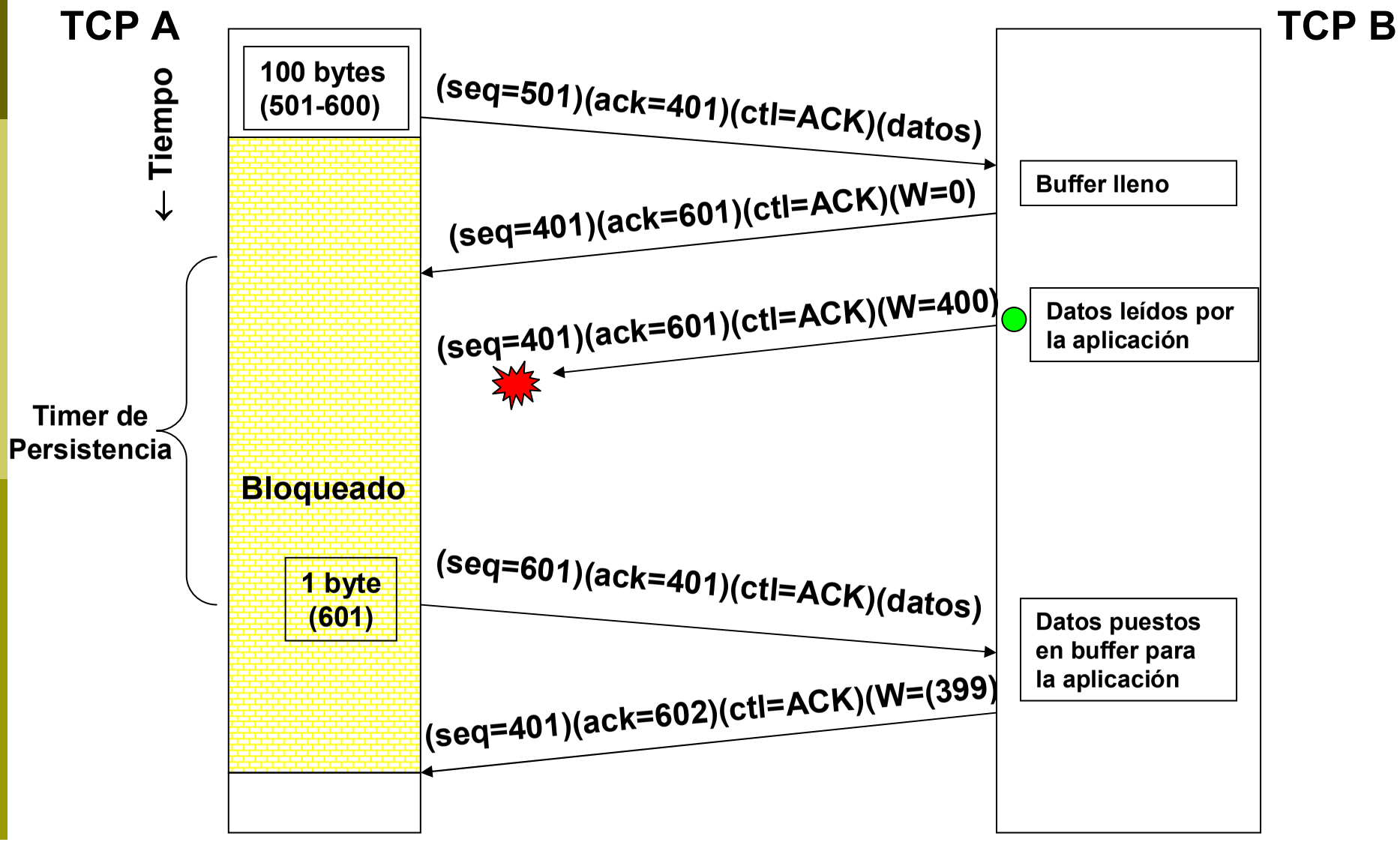
Intercambio de datos: casos excepcionales

- 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'.

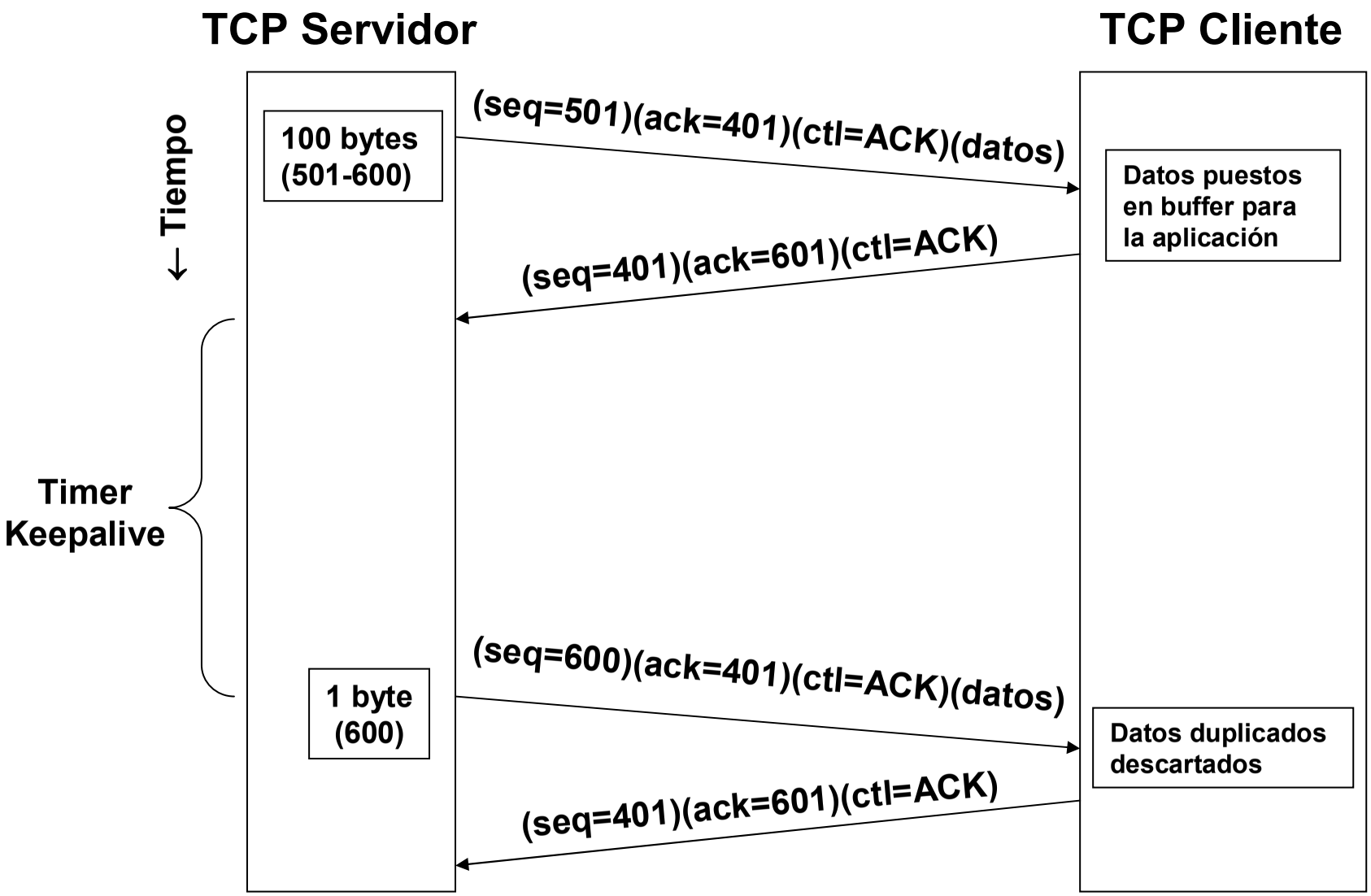
Timer de persistencia



Mensaje y timer de keepalive

- Evita que se queden conexiones 'medio abiertas'
- Se implementa reenviando el último byte transmitido en un segmento; el receptor descarta el dato pero devuelve un ACK
- Si se envían varios mensajes de keepalive sin respuesta se considera que se trata de una conexión medio abierta y se cierra.
- Para declarar una conexión medio abierta se espera a veces hasta 2 horas.
- El tiempo de envío de los mensajes se regula con el timer de keepalive.
- El keepalive no requiere modificaciones en el TCP receptor

Mensajes de keepalive



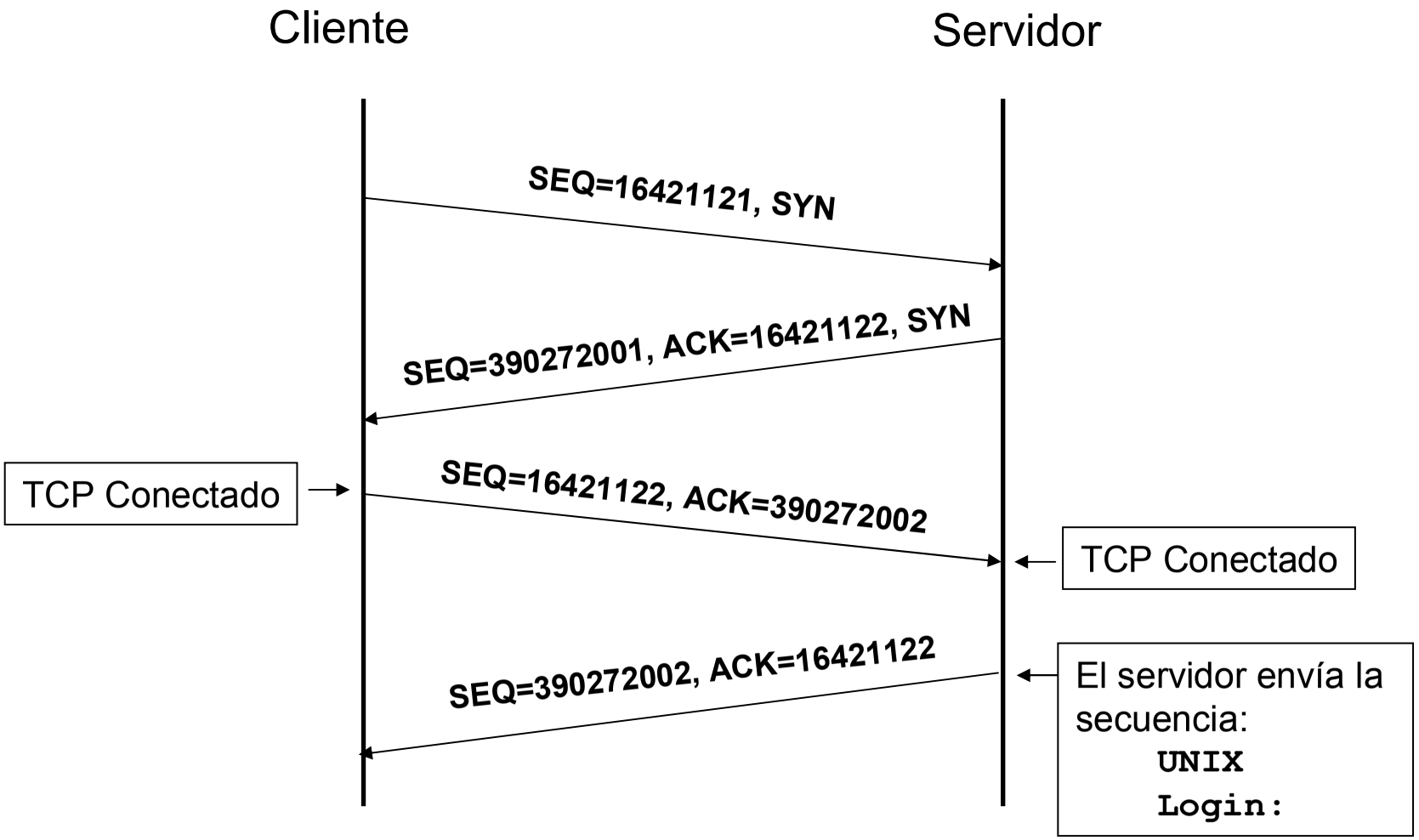
Cabeceras TCP

Inicio de una conexión Telnet

1. SYN TCP: --- TCP header --- TCP: TCP: Source port = 2345 TCP: Dest port = 23 (Telnet) TCP: Initial seq. Number = 16421121 TCP: Data offset = 24 bytes TCP: Flags = 02 (SYN) TCP: Window = 2048 TCP: Checksum = F2DA (correct) TCP: TCP: Options follow TCP: Max segment size = 1460	2. SYN TCP: --- TCP header -- TCP: TCP: Source port = 23 (Telnet) TCP: Dest port = 2345 TCP: Initial seq. Number = 390272001 TCP: Acknowledgment Number = 16421122 TCP: Data offset = 24 bytes TCP: Flags = 12 (ACK,SYN) TCP: Window = 4096 TCP: Checksum = C13A (correct) TCP: TCP: Options follow TCP: Max segment size = 1024
3. ACK TCP: --- TCP header --- TCP: TCP: Source port = 2345 TCP: Dest port = 23 (Telnet) TCP: Seq. Number = 16421122 TCP: Acknowledgment Number = 390272002 TCP: Data offset = 20 bytes TCP: Flags = 10 (ACK) TCP: Window = 2048 TCP: Checksum = DF43 (correct) TCP: No TCP options	4. DATA TCP: --- TCP header --- TCP: TCP: Source port = 23 (Telnet) TCP: Dest port = 2345 TCP: Seq. Number = 390272002 TCP: Acknowledgment Number = 16421122 TCP: Data offset = 20 bytes TCP: Flags = 18 (ACK,PSH) TCP: Window = 4096 TCP: Checksum = 9FEF (correct) TCP: No TCP options TCP: [12 byte(s) of data]

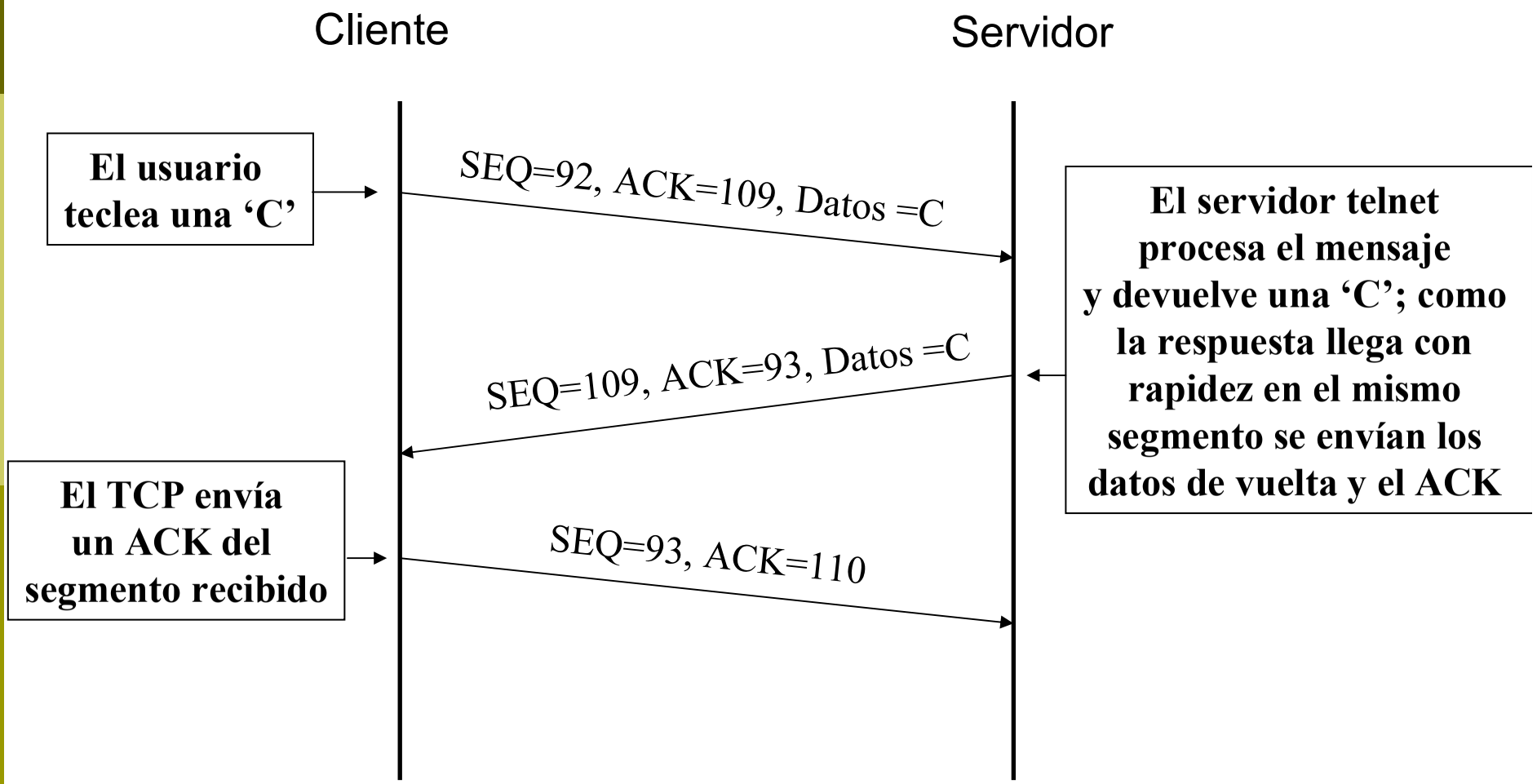
Intercambio de segmentos

caso anterior



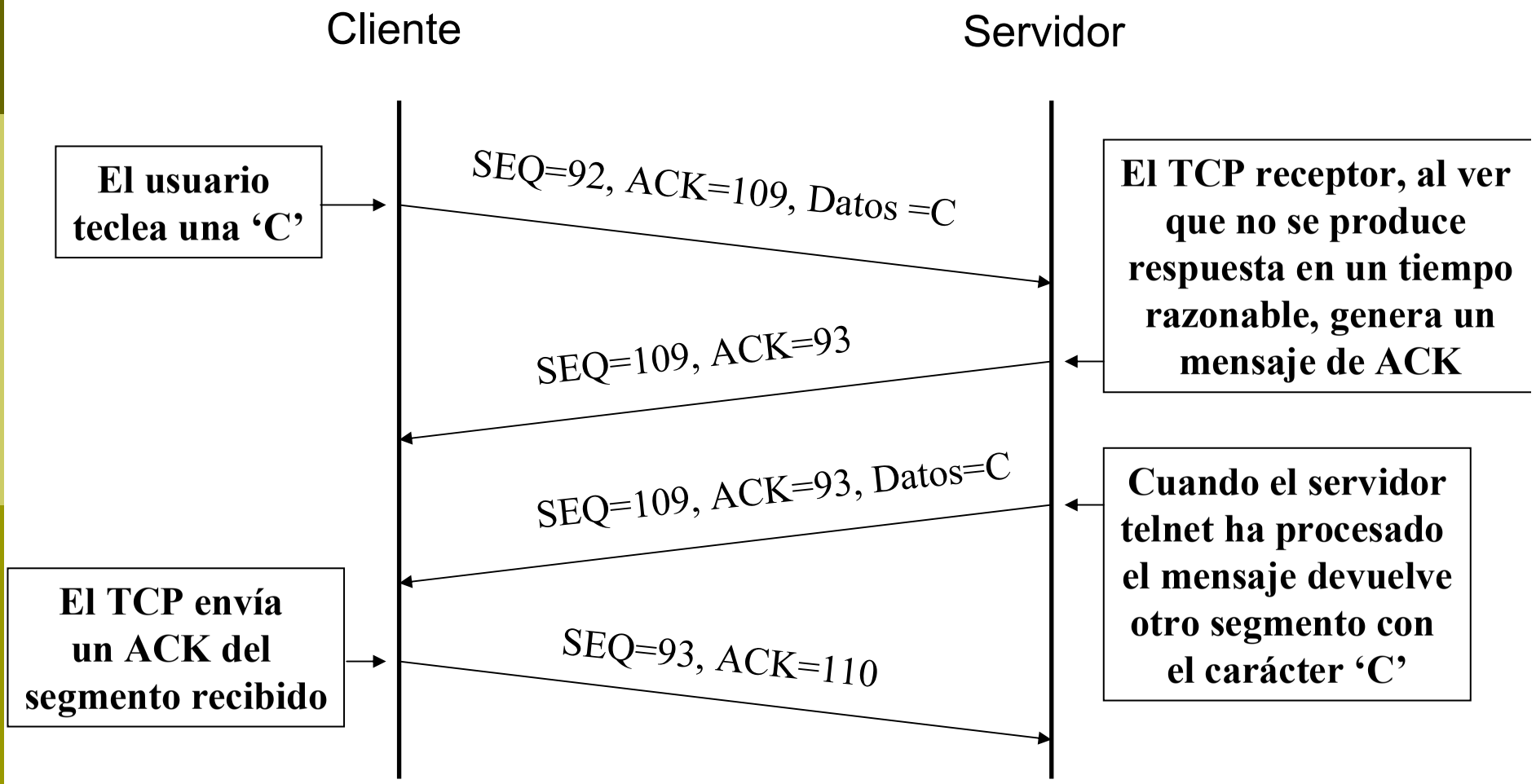
Funcionamiento de TCP en Telnet

con eco remoto, respuesta rapida de mensajes



Funcionamiento de TCP en Telnet

con eco remoto, respuesta lenta de mensajes



Sumario

- Funciones del nivel de transporte
- Protocolo UDP
- Protocolo TCP
  - Multiplexación
  - Conexión/Desconexión
  - Intercambio de datos y control de flujo
  - **Casos de baja eficiencia en TCP**
  - Control de congestión

Baja eficiencia en TCP

- El funcionamiento eficiente de TCP aconseja enviar segmentos del tamaño máximo permitido
- Cuando la aplicación emisora genera los datos en pequeñas dosis (telnet con eco remoto por ejemplo) se da un problema de eficiencia que se resuelve con el **algoritmo de Nagle**.
- Si la aplicación receptora los recoge byte a byte también se puede dar una baja eficiencia; esto se conoce como síndrome de la ventana tonta y se resuelve con la **Solución de Clark**.

Solución de Clark (RFC 813)

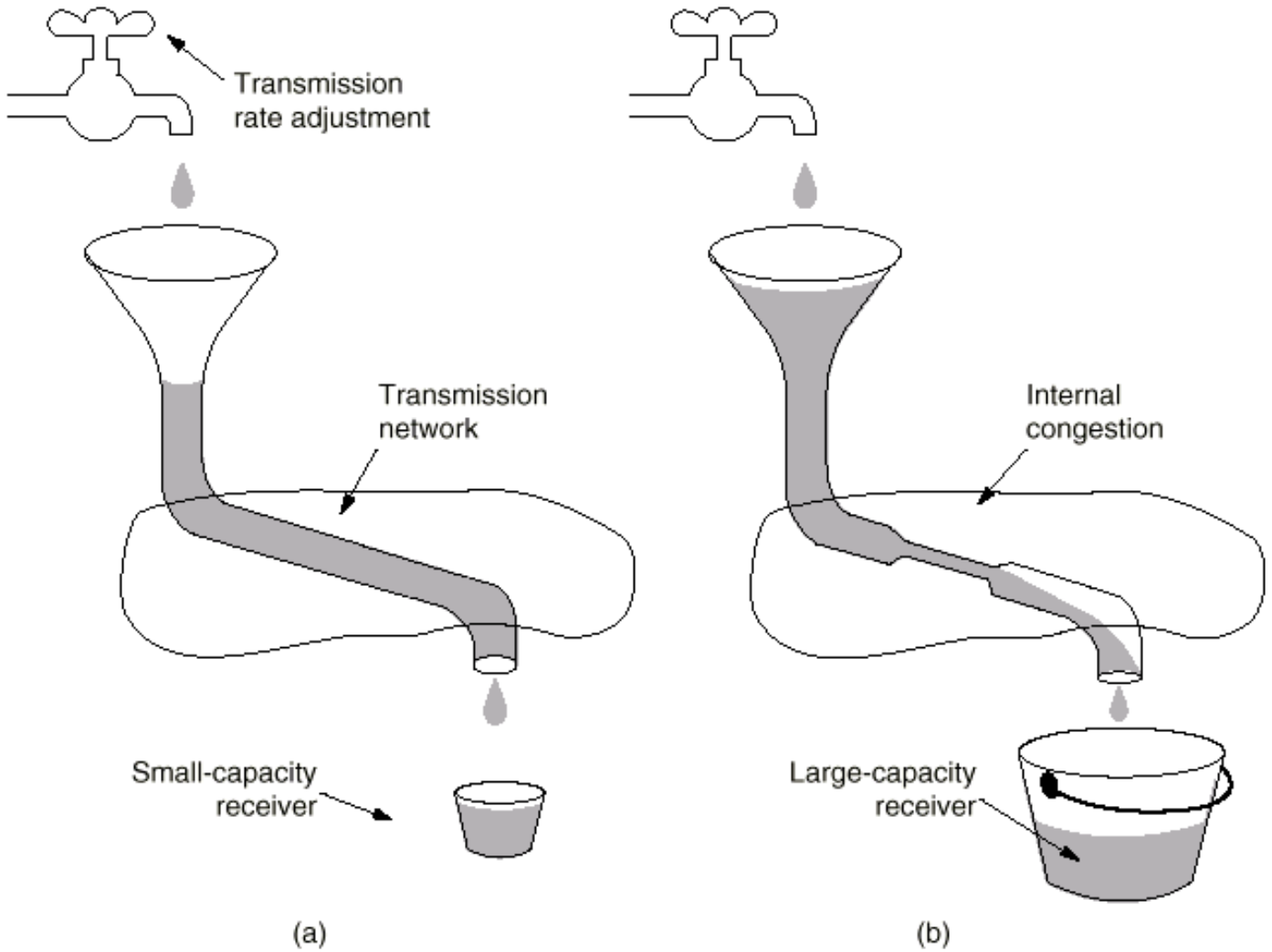
- El TCP receptor solo debe notificar una nueva ventana cuando tenga una cantidad razonable de espacio libre. Razonable significa:
  - Un MSS (segmento del tamaño máximo), o la mitad del espacio disponible en el buffer

Sumario

- Aspectos generales del nivel de transporte
- Protocolo UDP
- Protocolo TCP
  - Multiplexación
  - Conexión/Desconexión
  - Intercambio de datos y control de flujo
  - Casos de baja eficiencia en TCP
  - **Control de congestión**

Control de congestión

- Por medio del tamaño de ventana el receptor puede dosificar al emisor en función del buffer disponible. Esto es control de flujo.
- Pero puede que el receptor tenga espacio de sobra pero la red esté congestionada. En este caso el TCP debe regularse para no inyectar demasiado tráfico, a pesar de que la ventana disponible sea muy grande. Esto es control de congestión.
- Normalmente en TCP se utiliza control de congestión implícito. Ahora se está empezando a experimentar en Internet con el control de congestión explícito.



Control de flujo

Control de congestión

Fig. 6-31. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

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.

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

Proceso de una trama Ethernet TCP/IP recibida por un host