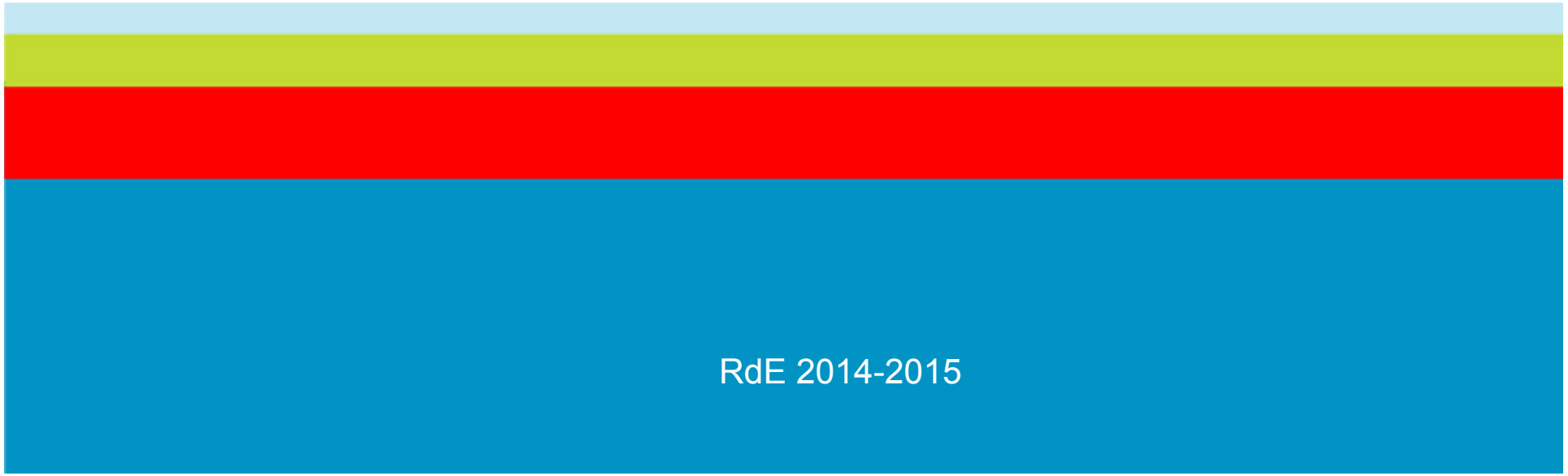
5 La capa de transporte

5.3 TCP



RdE 2014-2015

5 Guión del Tema 5

- 5. Capa de transporte:
  - 5.1 El servicio de transporte.
  - 5.2 UDP.
  - **5.3 TCP.**
  - 5.4 Problemas de rendimiento.

5 Guión del Tema 5.4

- TCP:
  - Modelo de servicio.
  - El protocolo TCP.
  - Control de flujo.
  - Control de la congestión.
  - Control de la temporización.

5.3 Introducción a TCP

- TCP **Transmission Control Protocol**.

- Diseñado para transportar datos de manera confiable sobre una red no confiable.
- Descrito en RFC 793, correcciones en RFC 1122, y ampliaciones en RFC 1323.
- Implementado como proceso, librería, parte del *kernel* o en *hardware*.
- Maneja flujos de datos y realiza el interfaz con el nivel IP. Acepta datos de la capa de aplicación, los trocea en partes inferiores a 64kB (1460 Bytes en *ethernet*), y envía cada trozo como un datagrama IP.
- Protocolo TCP (conjunto de reglas) versus entidad TCP (*software*).

5.3 Características funcionales

- **Protocolo *Unicast***. No se soporta ni *multicast*, ni *broadcast*.
- **Orientado a conexión**. Sincronización entre dos puntos finales.
- **Confiable**. Los datos llegan íntegros, no duplicados y en el mismo orden.
- ***Full-duplex***. Se envían datos en ambas direcciones en el mismo contexto.
- ***Streaming***. No se preserva la estructura del mensaje enviado desde la aplicación.
- **Velocidad adaptativa**. Se trata de conseguir la mayor velocidad de transferencia sin pérdida de datos.

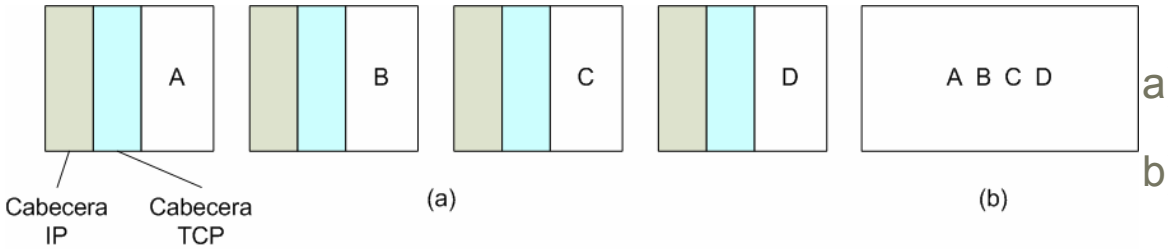
5.3 Modelo de servicio

- Puntos finales: sockets. Identificador: dirección IP local + puerto local. Multiplexación.

- Puerto: número 16 bit, TSAP. Puertos bien conocidos o *well-know*.
- Flags:
  - **PUSH**. Envío inmediato sin esperar a que haya más peticiones. TCP puede recolectar todos los datos PUShed en un solo datagrama IP. No se garantiza la separación.
  - **URGENT**. Envío inmediato sin esperar más datos.

5.3 Modelo de servicio

Puerto	Protocolo	Uso
21	FTP	Transferencia de ficheros
23	Telnet	Login remoto
25	SMTP	Envío de correo electrónico
69	TFTP	Protocolo de transferencia trivial de ficheros
79	Finger	Petición de información sobre usuarios
80	HTTP	WWW
110	POP-3	Acceso remoto a correo electrónico
119	NNTP	Noticias Usenet



a) 1 segmento TCP enviado en 4 paquetes IP

b) Se entrega en un solo segmento a la aplicación.

5 Guión del Tema 5.4

- TCP:
  - Modelo de servicio.
  - El protocolo TCP.
  - Control de flujo.
  - Control de la congestión.
  - Control de la temporización.

5.3 Segmento TCP

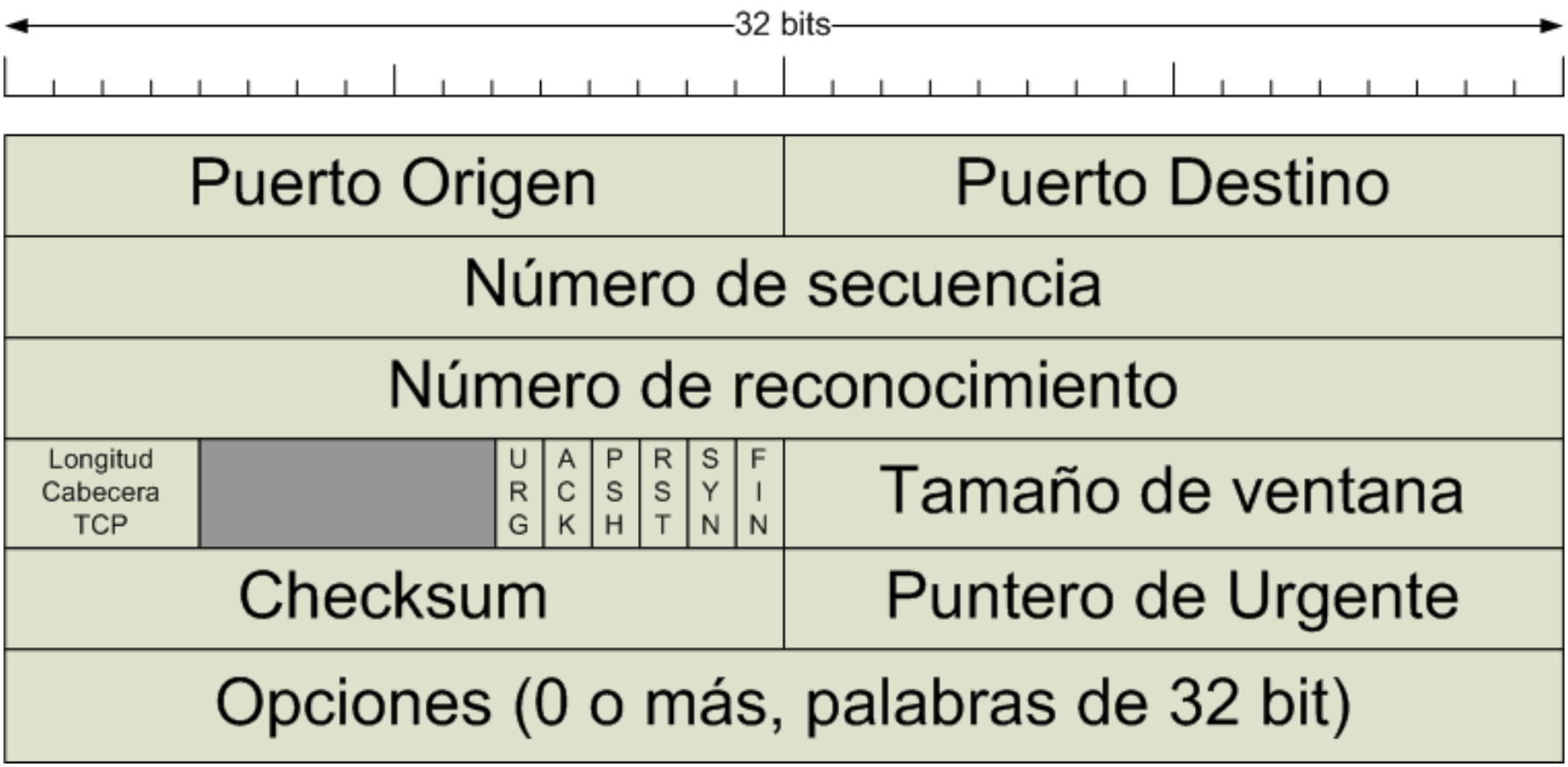
- Cabecera de 20 Bytes (sin opciones) seguidos de cero o más Bytes. El tamaño del segmento es decidido por la entidad TCP:
  - Acumular varias escrituras desde la aplicación en un solo segmento.
  - Romper una escritura en varios segmentos.
  - Limitaciones: IP *payload* 65536 Bytes, MTU generalmente 1500 Bytes.
- Cada Byte tiene su propio **número de secuencia** de 32 bit. Se usa para el mecanismo de ventana y para los reconocimientos.
- Protocolo de **ventana deslizante**. Dispone de un temporizador de retransmisión. En *Ack Number* se pone el siguiente número de secuencia que se espera recibir.

5.3 Problemas protocolo TCP

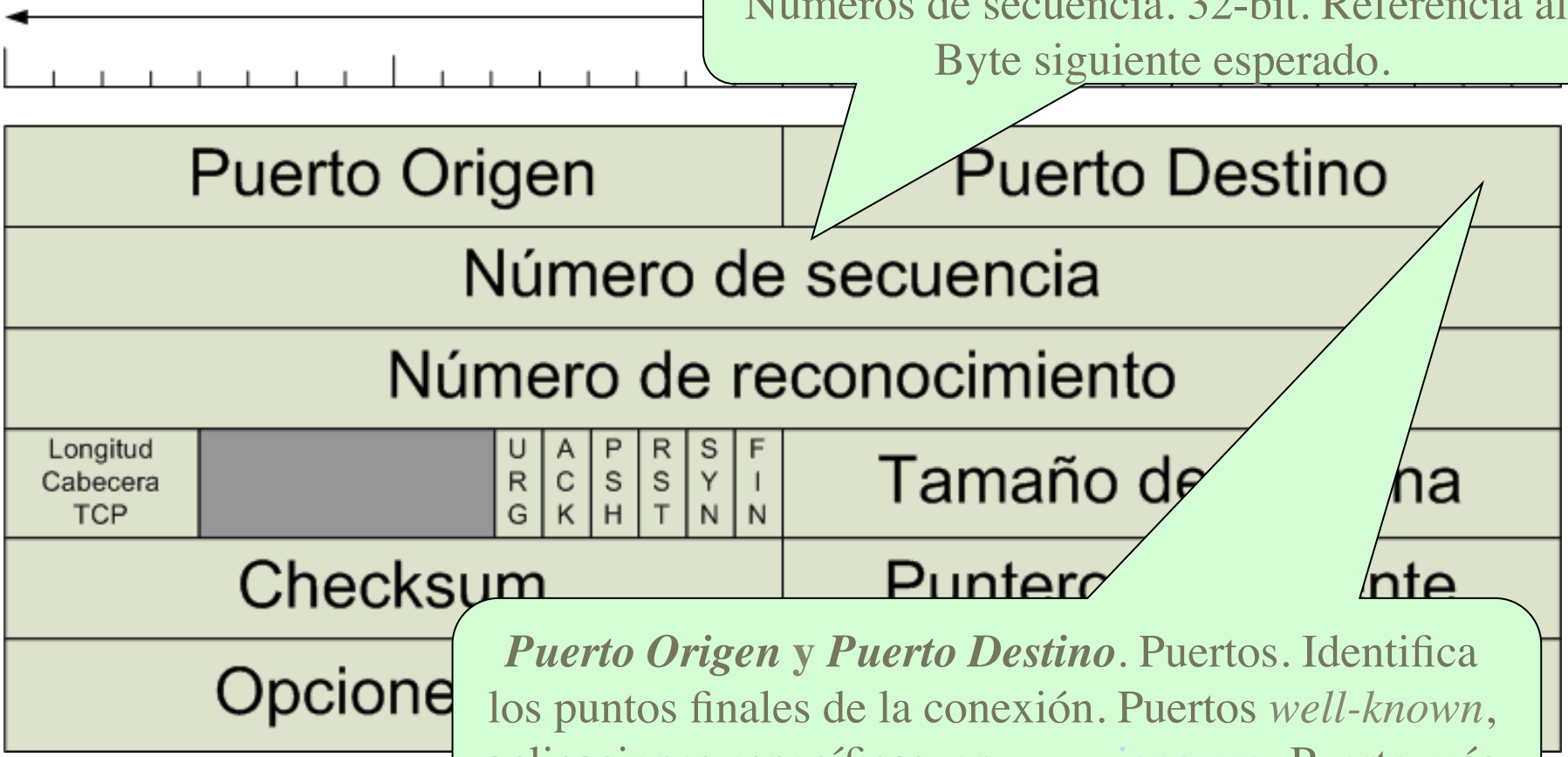
- Segmentos pueden llegar **fuera de orden**. No se puede enviar reconocimiento hasta que se reciban todos.

- Puede vencer el temporizador, y retransmitirse diferentes segmentos.

5.3 Cabecera TCP



5.3 Puertos y números de secuencia



Números de secuencia y reconocimiento. Números de secuencia. 32-bit. Referencia al Byte siguiente esperado.

Puerto Origen y ***Puerto Destino***. Puertos. Identifica los puntos finales de la conexión. Puertos *well-known*, aplicaciones específicas, en www.iana.org. Puerto más dirección IP identificador de 48-bit.

5.3 Longitud y flag

- **TCP *header length***. 4 bit. Número de palabras de 32-bit. Necesario por las opciones.
- 6 bit reservados + 6 bit ***flag***:
  - **URG=1**. Indica que se usa el puntero de urgencia, que indica donde pueden ser encontrados los datos. Interrupción.
  - **ACK=1**. Indica que *Acknowledgement number* es válido. Si ACK=0 el campo se ignora.
  - **PSH=1**. Indica que el receptor debe pasar los datos a la aplicación lo más pronto posible.
  - **SYN=1**. Establecimiento de conexión. Si ACK=0 no se usa *piggyback*. Respuesta SYN=1, ACK=1.
  - **FIN=1**. Finalización de la conexión. Entidad en proceso de cierre, sigue recibiendo datos indefinidamente.

5.3 Tamaño de ventana

- ***Window Size***. Tamaño de la ventana. Tamaño variable. Indica cuantos bytes pueden ser enviados comenzando por el Byte reconocido.

- Se admite tamaño 0, e indica que de momento no se desean recibir más datos.
- Cuando puede recibir más datos, el receptor envía el mismo número de reconocimiento con tamaño de ventana mayor que cero.

5.3 Pseudocabecera TCP

- ***Checksum***. Se calcula sobre pseudocabecera, cabecera TCP, y datos. Se añade un Byte 0x00 si el número es impar. Se suman las palabras de 16-bit en complemento a uno y se toma el complemento a uno de la suma. El receptor realiza el mismo cálculo con el *checksum* debe resultar cero.

Dirección Origen		
Dirección Destino		
00000000	Protocolo=6	Longitud Segmento TCP

5.3 Opciones

- ***Options***. Procura soporte a características no cubiertas por la cabecera.

- La más interesante es MSS, ***Maximum Segment Size***. Utilizada cuando se utilizan encapsulados tipo IP en IP, o IPSEC, y cuando los sistemas tienen un buffer limitado. No tiene porque ser el mismo en ambas direcciones.

5.3 Mejoras

- **Escalado de la ventana**. RFC 1323. En líneas de altas velocidad y latencia, por ejemplo Gigabit transoceánico, o satélite, una ventana de 64 KB es insuficiente. Negociación. Mover tamaño de ventana 14 bit a la izquierda, tamaño de ventana de 2^{30}

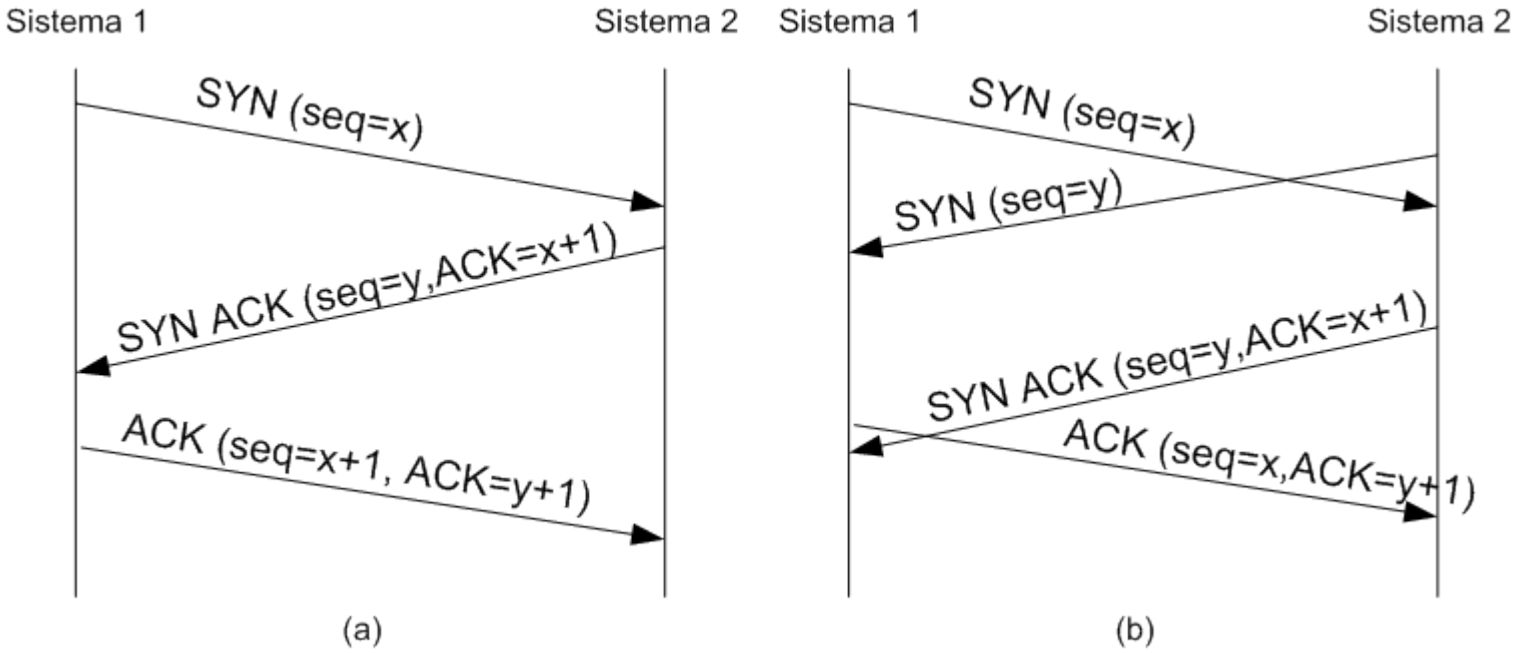
- **Reconocimiento selectivo o SACK *option***. RFC 1106. Permite el uso de NAK relativo a un número de secuencia determinado, y su retransmisión.

5.3 Conexión

a) Caso normal. SYN consume 1 byte de espacio de reconocimiento.

b) Caso de conexión simultánea. Una sola conexión.

- Número de secuencia diferente de cero. Se usa un esquema basado en reloj con un *tick* de 4 μs.



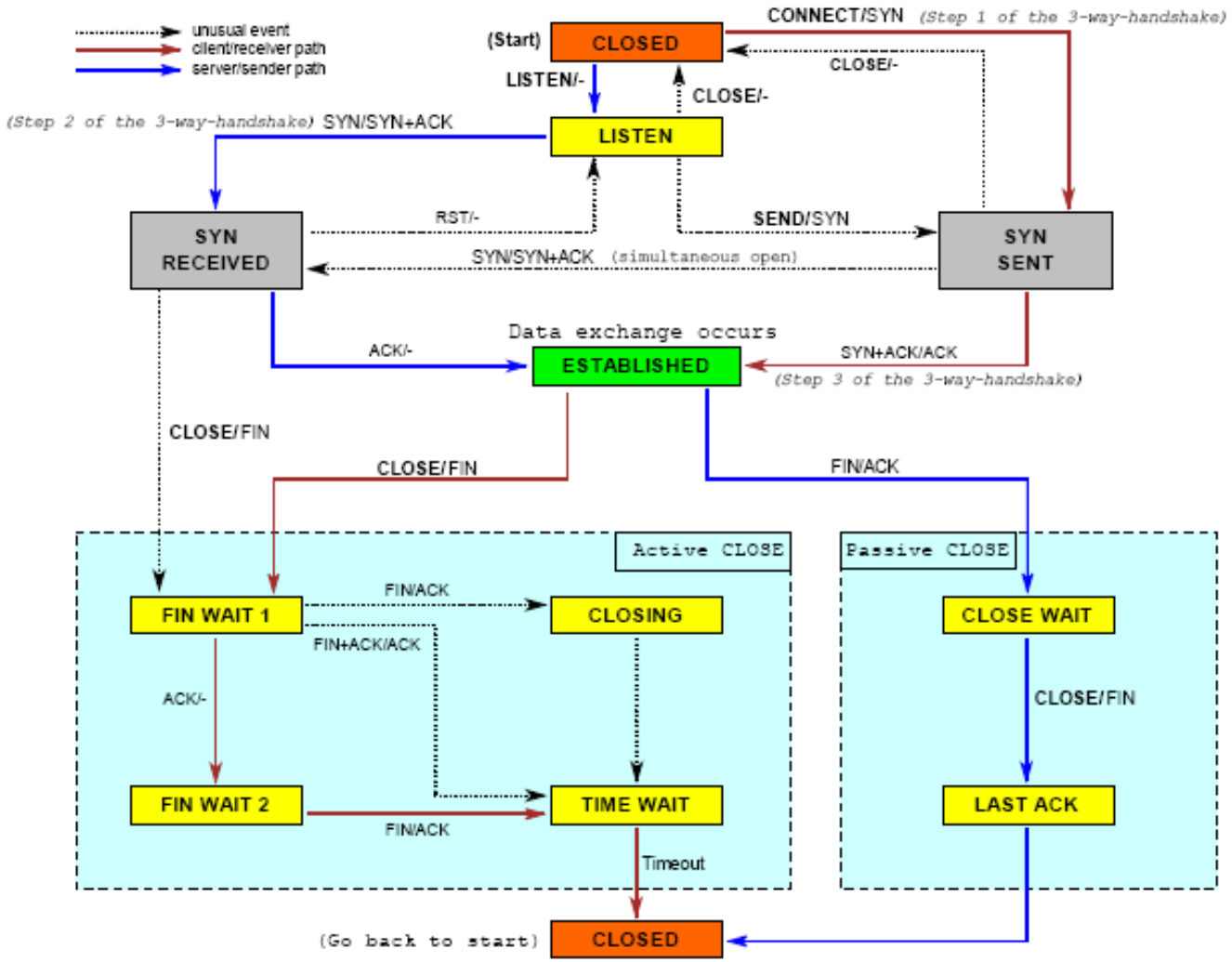
5.3 Desconexión

- La desconexión se produce como si fuesen conexiones *simplex*: se debe producir en ambos sentidos.
- Envío de FIN, con respuesta de ACK. Puede ser FIN-ACK en el mismo paquete.
- Temporizador si no existe envío de ACK. Problema de los dos ejércitos.

5.3 Estados

Estado	Descripción
CLOSED	Sin conexión activa o pendiente
LISTEN	El servidor está esperando una petición entrante
SYN RCVD	Una petición de conexión ha llegado esperando ACK
SYN SENT	La aplicación ha comenzado a abrir una conexión
ESTABLISHED	El estado normal de transferencia
FIN WAIT 1	La aplicación ha dicho que se termine
FIN WAIT 2	El otro lado está de acuerdo en terminar
TIMED WAIT	A la espera que los paquetes mueran
CLOSING	Ambos lados han intentado cerrar simultáneamente
CLOSE WAIT	El otro lado ha comenzado la desconexión
LAST ACK	A la espera que los paquetes mueran

5.3 Máquina de estados TCP



CEU

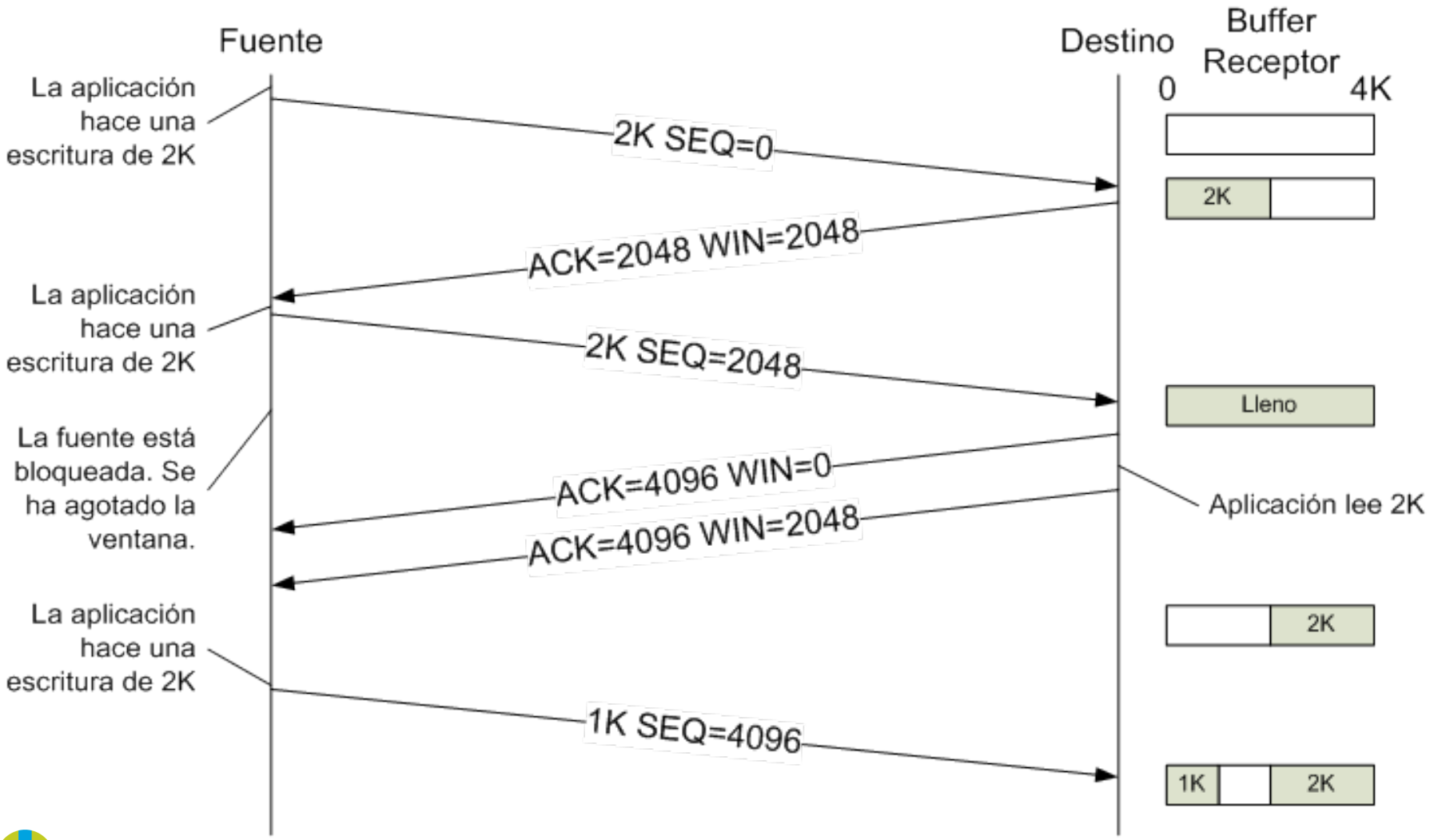
5 La capa de transporte

5.3 TCP (2/2)

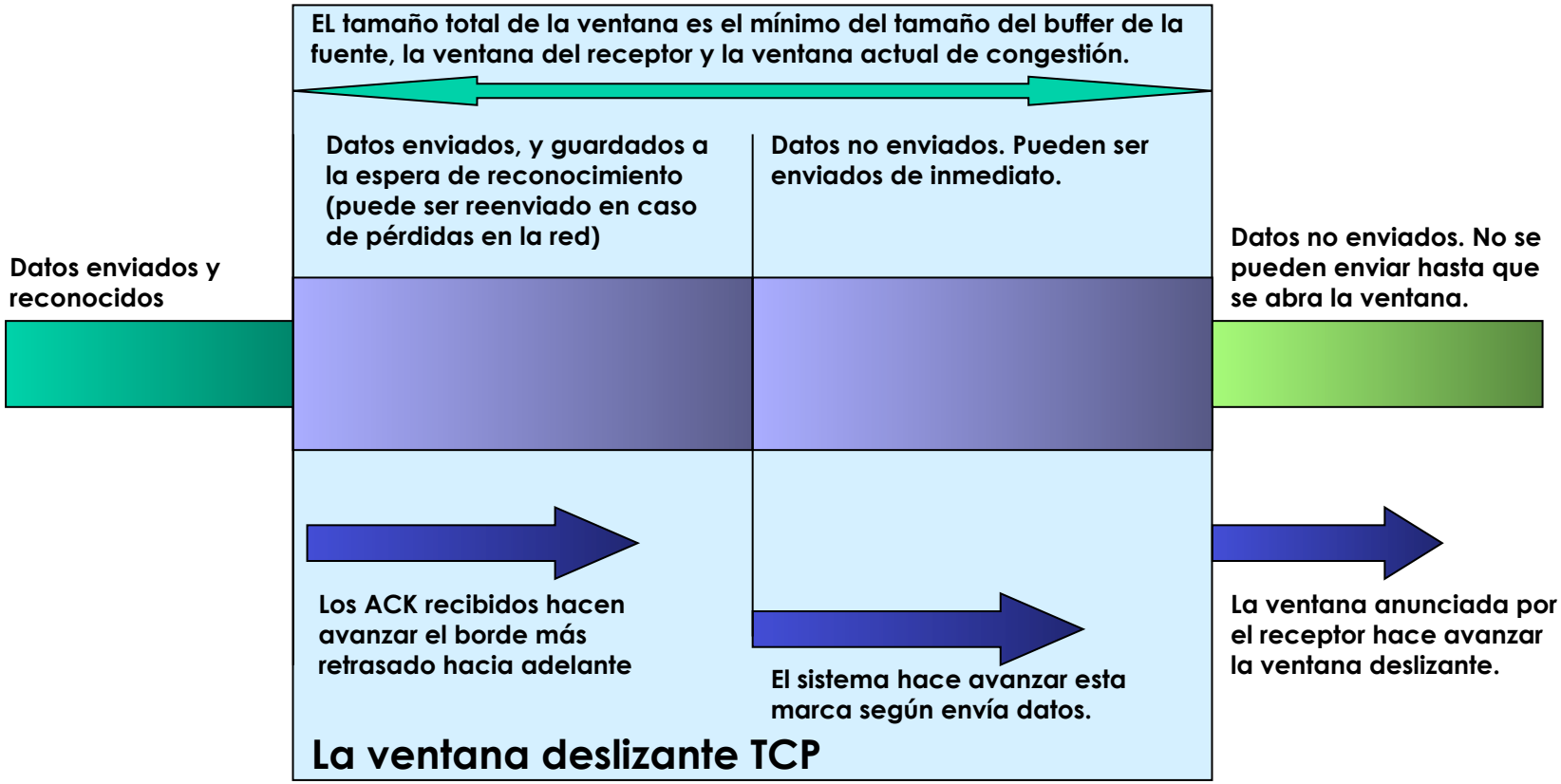
5 Guión del Tema 5.4

- TCP:
  - Modelo de servicio.
  - El protocolo TCP.
  - Control de flujo.
  - Control de la congestión.
  - Control de la temporización.

5.3 Gestión de ventana TCP



5.3 Gestión de la ventana



5.3 Tamaño de la ventana

- El tamaño de la ventana se elige para maximizar los datos enviados. Se debe alcanzar un punto de equilibrio en que la red no esté ociosa, pero no se pierdan datos por exceso de ocupación.

- **Tamaño ventana = BW x RTT**
- Se impone un límite al rendimiento de TCP.
- El tamaño de la ventana cambia dinámicamente para control de flujo, y control de la congestión.

5.3 Gestión de ventana TCP: excepciones

Cuando ventana enviada por el receptor es 0, la fuente no puede enviar datos, a no ser que:

1. Envío de datos urgentes, p.e. matar la aplicación en el servidor.
2. Solicitar reanuncio de la ventana, mediante el envío de 1 Byte. Evitar bloqueo debido a pérdida de paquete TCP.

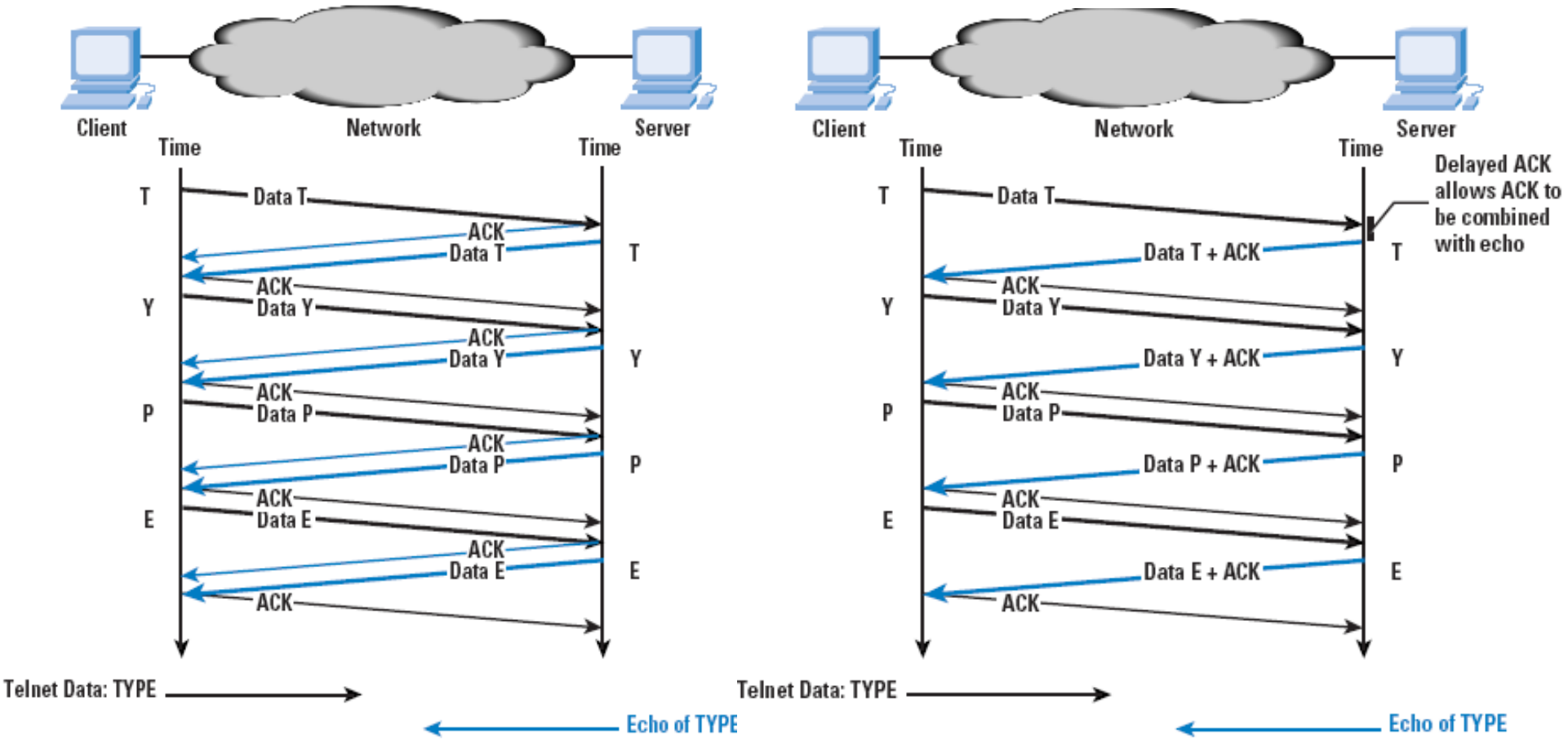
5.3 Mejoras

- Uso muy ineficiente del ancho de banda, p.e. telnet. Se retrasa el envío de reconocimiento 500 ms [1].

- **Mejoras:**

1. ***Piggybacking***. Se realiza el reconocimiento al realizar un envío de datos.
2. **Retraso del reconocimiento.** Se retrasa el reconocimiento 200 ms [3] para dar a la aplicación oportunidad de generar datos sobre los que hacer *piggyback*.

5.3 Mejoras


Client
Network
Server
Time
Time
T
Data T
ACK
Data T
T
Y
ACK
Data Y
ACK
Data Y
Y
P
ACK
Data P
ACK
Data P
P
E
ACK
Data E
ACK
Data E
E
ACK
Telnet Data: TYPE
Echo of TYPE
Client
Network
Server
Time
Time
Delayed ACK allows ACK to be combined with echo
T
Data T
Data T + ACK
T
Y
ACK
Data Y
Data Y + ACK
Y
P
ACK
Data P
Data P + ACK
P
E
ACK
Data E
Data E + ACK
E
ACK
Telnet Data: TYPE
Echo of TYPE

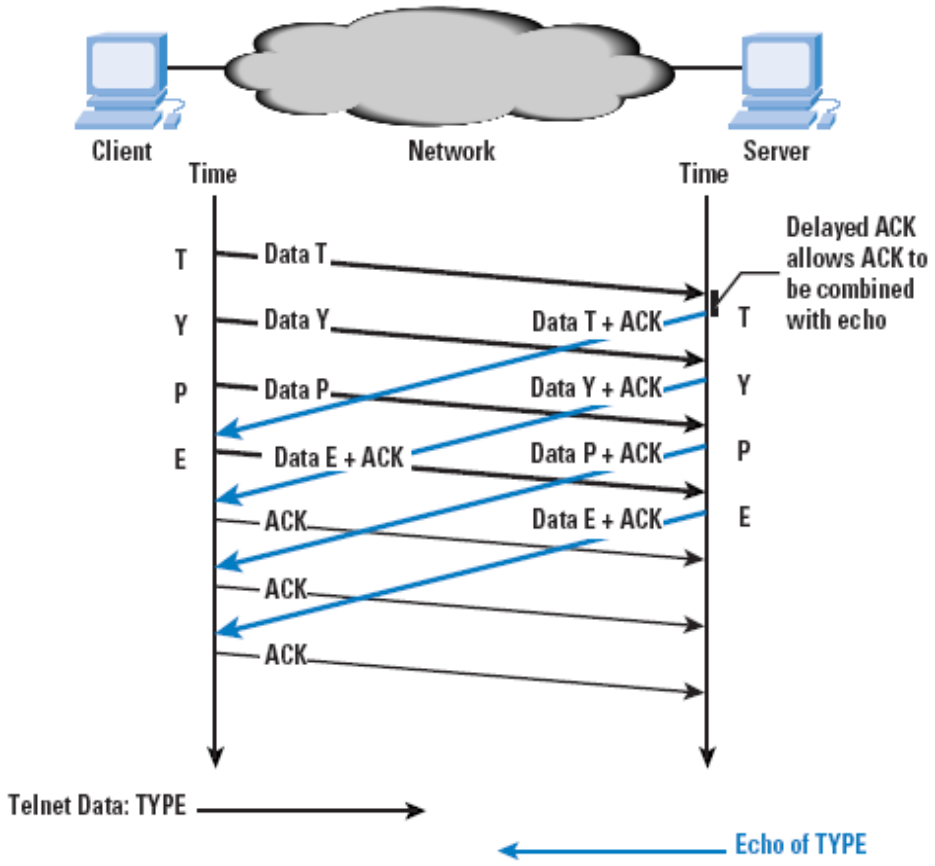
5.3 Algoritmo de Nagle

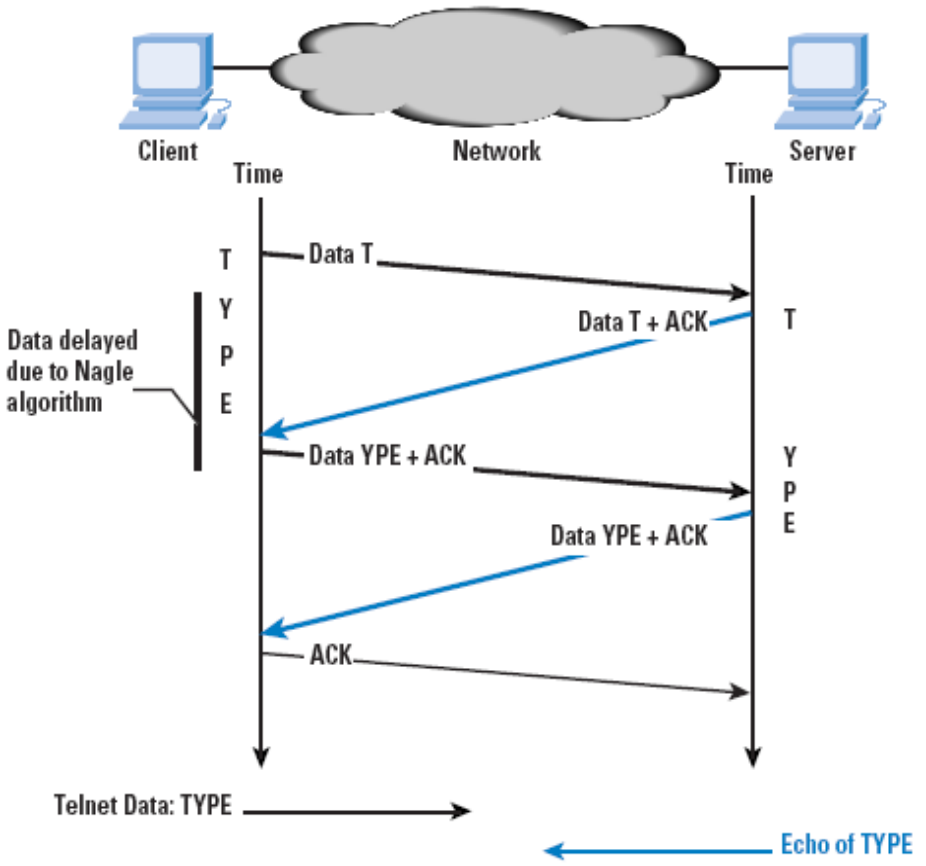
- **Algoritmo de Nagle.** RFC 896. El emisor envía primer byte, y se espera a enviar siguiente segmento, con bytes acumulados durante la espera, a recibir reconocimiento. Se permite el envío si el segmento alcanza la mitad de su tamaño.

- A veces no es deseable, por ejemplo en X-Windows. En general TCP no es un protocolo eficiente para tráfico interactivo.

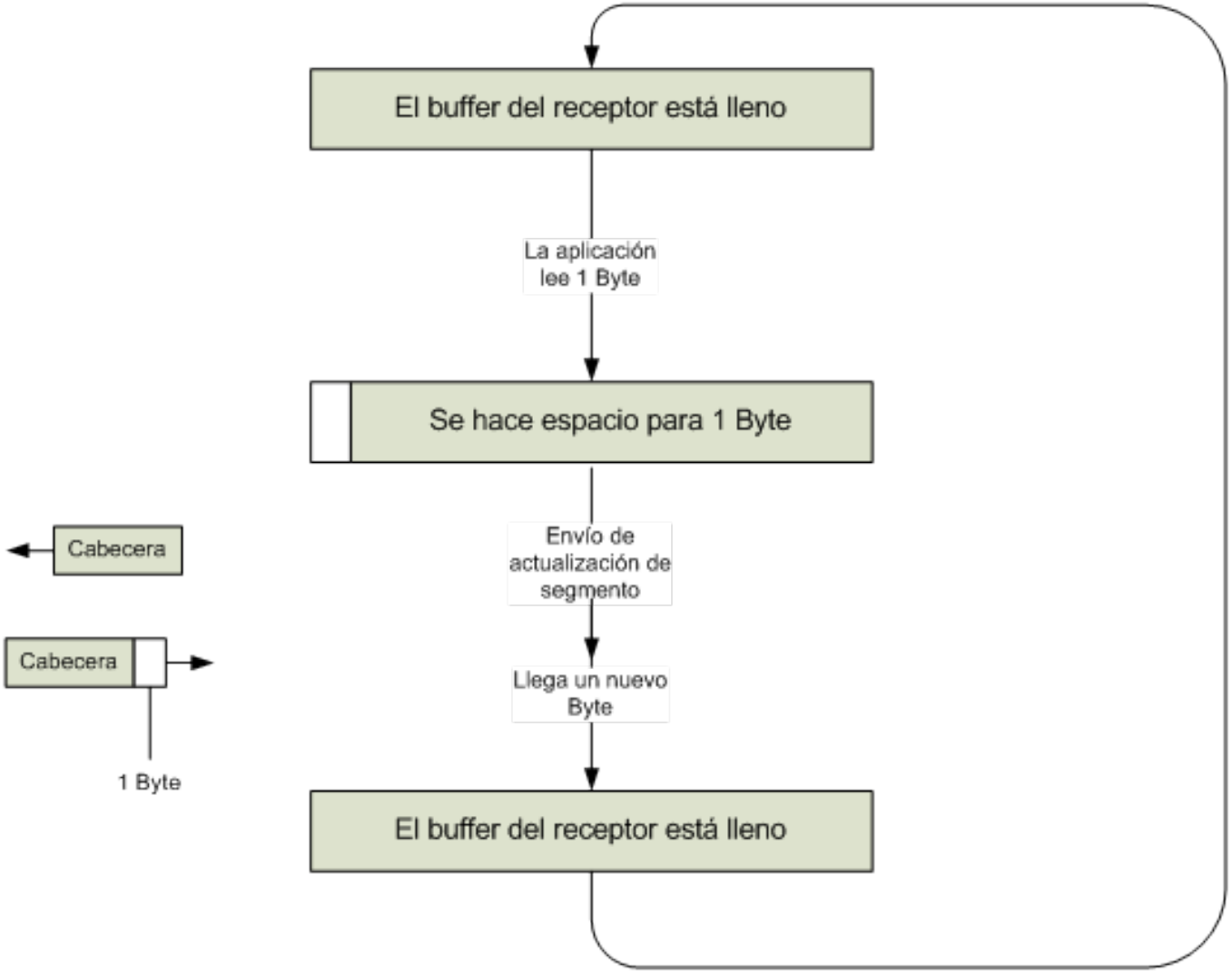
- Objetivo. Mejorar rendimiento. Menos cabeceras.

5.3 Algoritmo de Nagle





5.3 Síndrome de la ventana tonta SWS



5.3 Solución a SWS

- **Algoritmo de Clark** (1982). Receptor espera hasta tener suficiente espacio para manejar al menos la MSS. La fuente se espera hasta tener un segmento completo o la mitad del tamaño de buffer del receptor (estimado a partir de datos de ventana).

- Los algoritmos de Nagle y de Clark son complementarios.

5.3 Mejoras

- Bloqueo de las lecturas desde la aplicación, llamadas a TCP, hasta acumular muchos datos. No válido para aplicaciones interactivas.

5 Guión del Tema 5.4

- TCP:
  - Modelo de servicio.
  - El protocolo TCP.
  - Control de flujo.
  - Control de la congestión.
  - Control de la temporización.

5.3 Control de la congestión

- Detección de la congestión => pérdida de paquetes:
  - Errores en el viaje (raro?).
  - Congestión en equipos intermedios o destino.

- Algoritmos de control de la congestión:
  - Evitar la congestión.
  - *Slow start*.
  - *Fast retransmit.*
  - *Fast recovery.*

5.3 Evitar la congestión

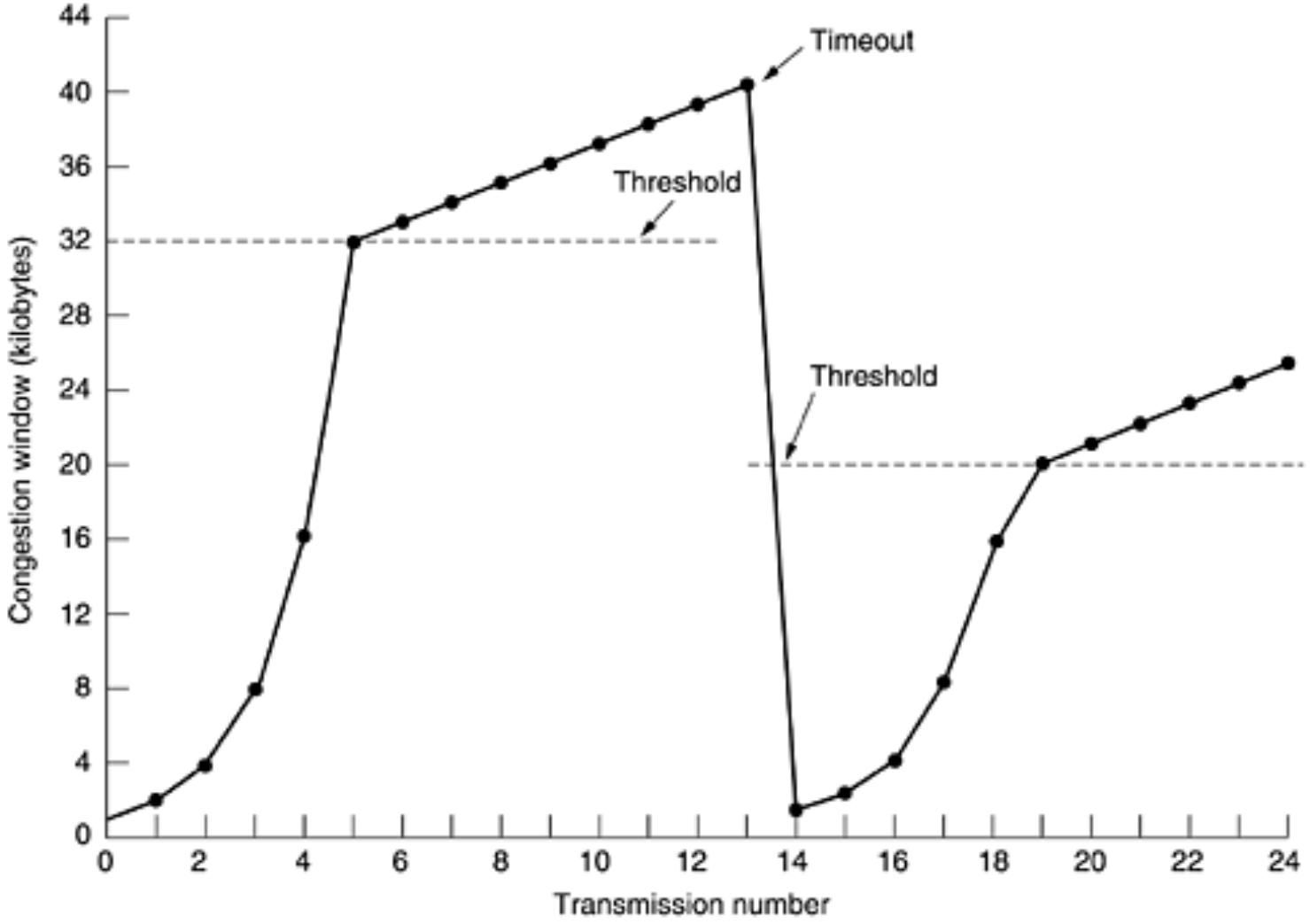
- Dos problemas: capacidad de la red y capacidad del receptor.
- Se mantienen dos ventanas: ventana aceptada por el receptor y la ventana de congestión, que refleja lo que la fuente puede enviar.

- Se utiliza la menor de ambas.

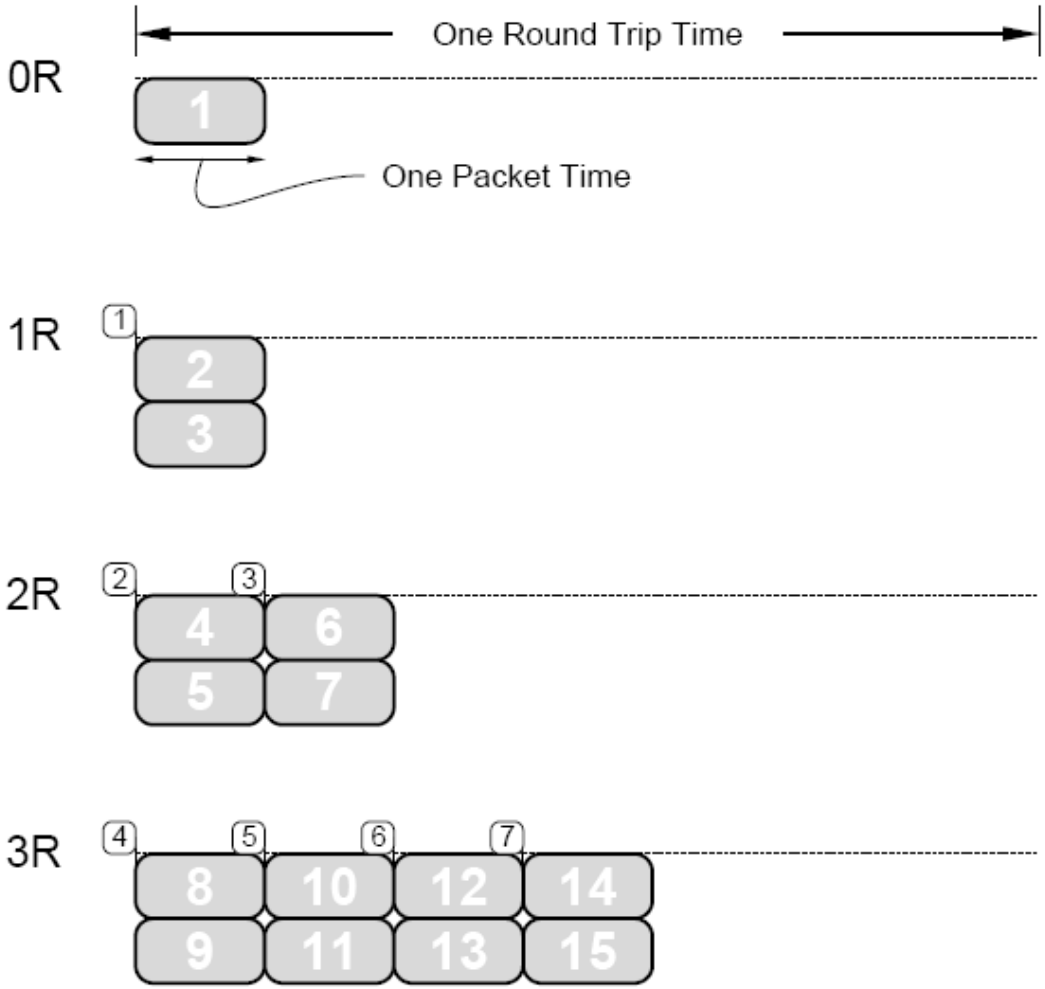
5.3 Algoritmo *Slow Start*

- La fuente mantiene dos ventanas, la que se usa es la mínima de ambas:
  - Ventana acordada con el receptor, ***rwnd***.
  - Ventana de congestión, ***cwnd***.
  - Umbral, ***ssthresh***, inicialmente 64 KB.
- Algoritmo (ver [6]):
  - La fuente fija la ventana de congestión como MSS. Se envía un segmento.
  - Si el primer envío es reconocido, se envía el doble de segmentos.
  - Mientras no haya *timeout* se duplica el envío, hasta llegar al *ssthresh (Slow Start)*, en que se comienza a aumentar la ventana en un segmento (*Congestion Avoidance*).
  - Cuando sucede un *timeout*, *ssthresh* se divide por la mitad, y se vuelve a comenzar por el principio.
- Es exponencial hasta *ssthresh* y después lineal.
- Es requerido en las implementaciones de TCP.
- El mensaje ICMP SOURCE QUENCH es tratado como *timeout*.

5.3 Ejemplo de *Slow Start*



5.3 Ejemplo de *Slow Start*



5.3 *Fast Retransmit* y *Fast Recovery*

- ***Fast Retransmit***
  - Se retransmite al recibir 3 ACK duplicados, en vez de esperar al *Timeout*.

- ***Fast Recovery***
  - Se pone la ventana de congestión igual al umbral cwnd = ssthresh

CEU

5 La capa de transporte

5.3 TCP (2/2)

5 Guión del Tema 5.4

- TCP:
  - Modelo de servicio.
  - El protocolo TCP.
  - Control de flujo.
  - Control de la congestión.
  - Control de la temporización.

5.3 TCP *Timer Management*

- **Temporizador de retransmisión. RTO.** Cuando se envía un segmento se arranca un temporizador que se detiene si se produce reconocimiento, o provoca retransmisión si vence.
- Problema: tiempo en transporte no es predecible.
- ¿Cómo debe ser? Si es muy pequeño se producen retransmisiones innecesarias. Si es muy grande la eficiencia del protocolo es pequeña.

5.3 Algoritmo para RTT y RTO Jacobson (1)

Algoritmo de Jacobson para RTT [5]:

- TCP mantiene RTT por conexión. Mejor estimación para el destino del que se trate.
- RTT da el valor para el temporizador de retransmisión.
- Se mide el tiempo hasta el reconocimiento en cada caso.
- A partir del RTT y su desviación típica se calcula el RTO.

5.3 Algoritmo para RTT y RTO Karn

- Si se una retransmisión, el algoritmo no está claro, y se produce una contaminación de la estimación.

- La propuesta de **Karn** es una corrección del algoritmo de Jacobson:
  - ❑ No se actualiza el RTT en segmentos retransmitidos.
  - ❑ El RTO se duplica en cada fallo, hasta que se reciben sin retransmisión.

5.3 Bibliografía (1/2)

[1] Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall Pub, 4ª Ed 2002, apartado 6.5.

[2] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Walrand, “Analysis and Comparison of TCP Reno and Vegas”, July 13, 1998.

[3] Huston G., “TCP Performance”, IPJ June 2000 vol 3, n 2.

[4] Huston G., “The Future for TCP”, IPJ September 2000 vol 3, n 3.

[5] Nagle, J., “Congestion Control in IP/TCP Internetworks”, RFC 896,January 1984.

[6] Jacobson V., “Congestion Avoidance and Control”, ACM Computer Communication Review, Vol. 18, No. 4, August 1988.

[7] L.S. Brakmo and L.L. Peterson, “TCP vegas: end to end congestion avoidance on a global internet”, IEEE Journal on Selected Areas in Communications, 13(8):1465{80, October 1995.

[8] Cheng Huang, ‘‘TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation’’, 02.07.2003

[9] V. Paxson, M. Allman, “RFC 2988 STD 1 Computing TCP's Retransmission Timer”, November 2000.

5.3 Bibliografía (2/2)

[10] M. Allman, V. Paxson, W. Stevens, “RFC 2581 STD 1 TCP Congestion Control”, April 1999.

[11] Lawrence Landweber,Jun Murai, CS 640 Introduction to computer networks, fall 1999.

[12] Wan Gang Zeng, Introduction to TCP in Wireless Networks, 27/10/03.