TEMA 3
CAPA DE TRANSPORTE EN INTERNET

Fundamentos de Redes

2017/2018

Bibliografía Básica:

Capítulo 10, Pedro García Teodoro, Jesús Díaz Verdejo y Juan Manuel López Soler. ***TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES***, Ed. Pearson, 2017, ISBN: 978-0-273-76896-8

Para saber más…

Capítulo 3 James F. Kurose y Keith W. Ross. ***COMPUTER NETWORKING. A TOP-DOWN APPROACH***, 5ª Edición, Addison-Wesley, 2010, ISBN: 9780136079675.

Agradecimientos:

Transparencias originales de **Juan Manuel López Soler, Pedro García Teodoro, Jorge Navarro Ortiz**, Departamento TSTC, UGR.

1. **Introducción.**
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

INTRODUCCIÓN

Funciones y servicios de la capa de transporte:

- Comunicación extremo a extremo (*end-to-end)*.
- Multiplexación/demultiplexación de aplicaciones → *puerto.*

Protocolo UDP:

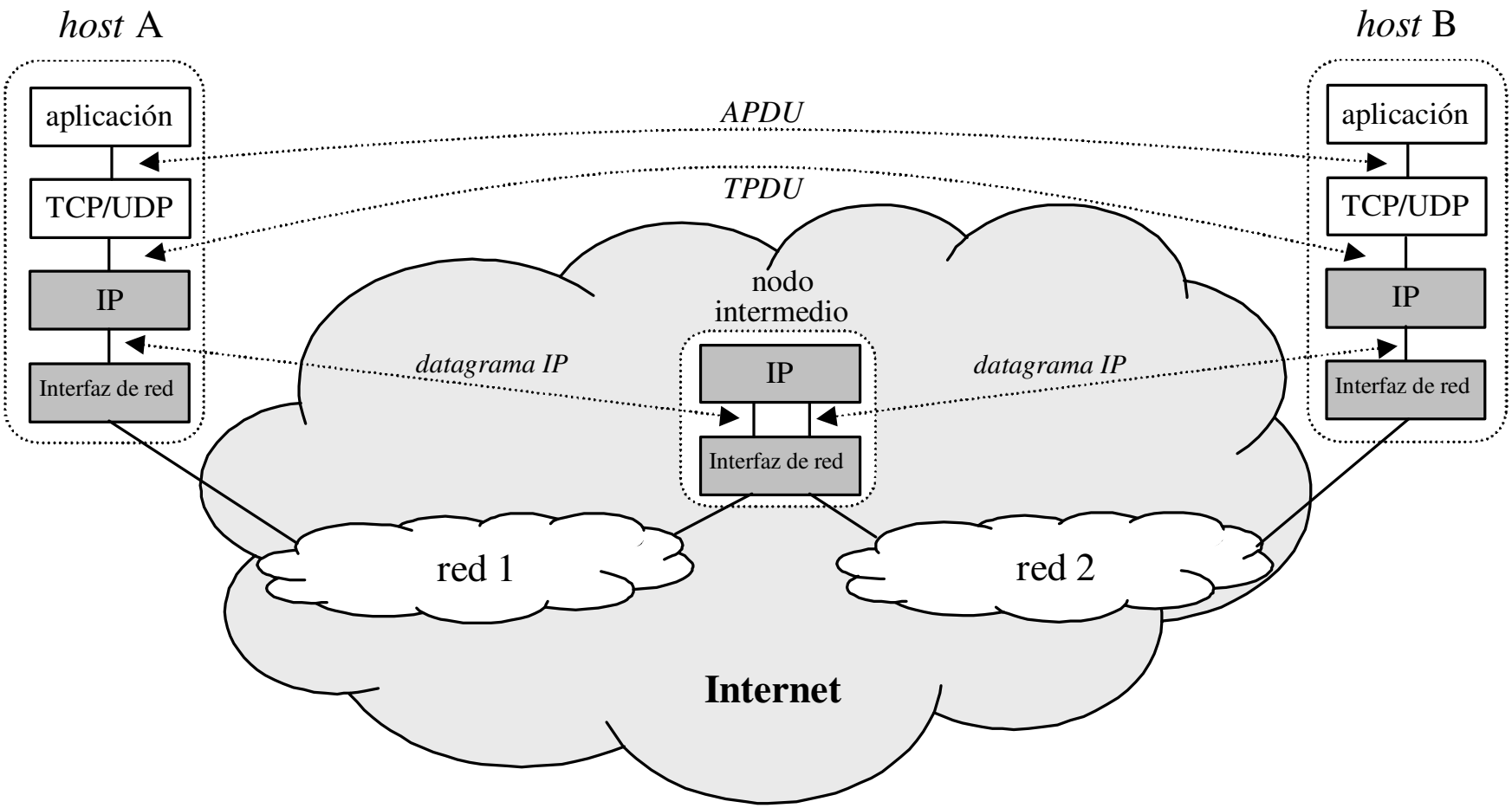
- Multiplexación/demultiplexación de aplicaciones**.**
- Servicio no orientado a conexión, no fiable**.**

Protocolo TCP:

- Multiplexación/demultiplexación de aplicaciones**.**
- Servicio orientado a conexión, fiable:
  - Control de errores y de flujo.
  - Control de la conexión.
  - Control de congestión.

Extensiones TCP

Comunicación extremo a extremo (*end-to-end)*:



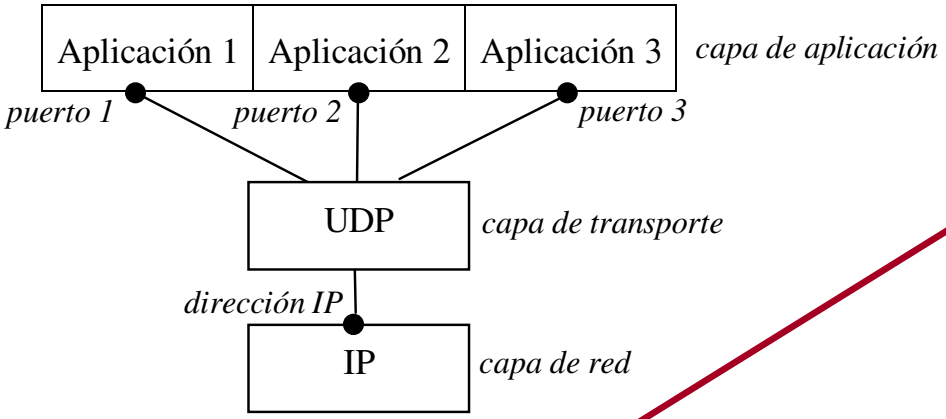
1. Introducción.
2. **Protocolo de datagrama de usuario (UDP).**
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

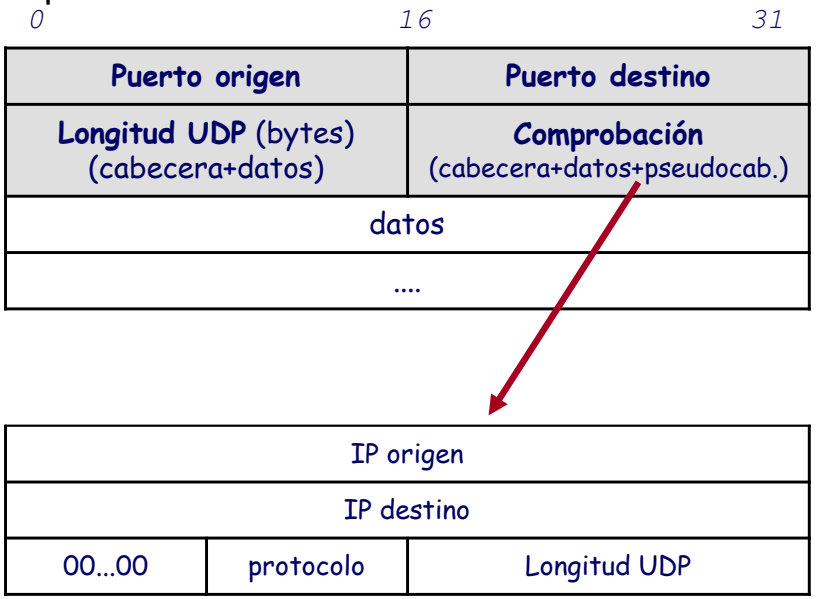
USER DATAGRAM PROTOCOL (UDP)

"User Datagram Protocol": RFC 768.

Funcionalidad "*best-effort*":

- Servicio no orientado a conexión: no hand-shaking, no hay retardos de establecimiento, cada TPDU es independiente.
- Servicio no fiable: puede haber pérdidas.
- No hay garantías de entrega ordenada.
- No hay control de congestión: entrega tan rápida como se pueda.
- Multiplexación/demultiplexación: transportar las TPDU al proceso correcto.





Datagrama de usuario UDP.

5. Calcule la suma de comprobación en UDP y TCP de las siguientes palabras de 8 bits (observe que aunque UDP y TCP utilicen palabras de 16 bits, en este ejercicio se pide el mismo cálculo sobre palabras de 8 bits): 01010011, 01010100, 01110100.

a) ¿Por qué UDP/TCP utilizan el complemento a uno de la suma complemento a uno, en lugar de directamente la suma en complemento a uno?

b) ¿cómo detecta el receptor los errores?

c) ¿se detectan todos los errores de 1 bit?

d) ¿se detectan todos los errores que afectan simultáneamente a 2 bits?

USER DATAGRAM PROTOCOL (UDP)

Multiplexación/demultiplexación: transportar las TPDU al proceso correcto.

Existen puertos preasignados con servicios normalizados:

Ejemplos de puertos UDP preasignados

Puerto	Aplicación/Servicio	Descripción
53	DNS	Servicio de nombres de domino
69	TFTP	Transferencia simple de ficheros
123	NTP	Protocolo de tiempo de red
161	SNMP	Protocolo simple de administración de red
520	RIP	Protocolo de información de encaminamiento

Otros puertos (>1024) están a libre disposición del desarrollador.

UDP se usa frecuentemente para aplicaciones multimedia: tolerantes a fallos y sensibles a retardos.

Cada segmento UDP se encapsula en un datagrama IP.

1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. **Protocolo de control de transmisión (TCP).**
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

TRANSMISSION CONTROL PROTOCOL (TCP)

Características del "Transmission Control Protocol": RFC 793 (1122, 1323, 2018, 2581).

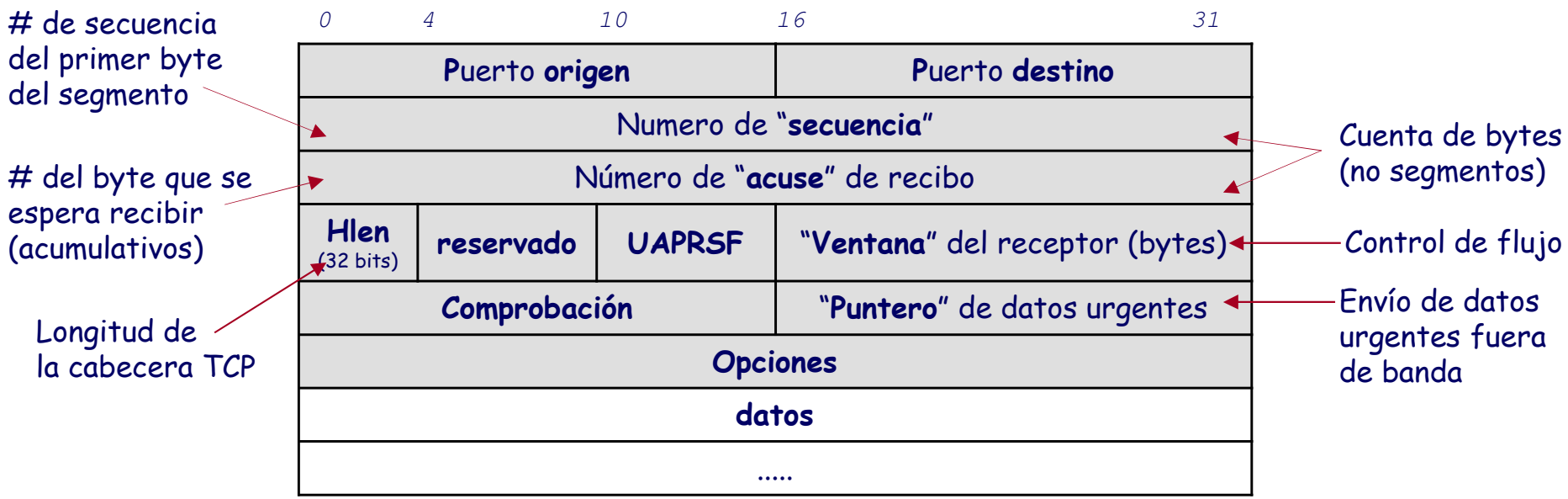
- Servicio punto a punto.
- Servicio orientado a conexión (exige un estado común entre el emisor y el receptor: "*hand-shaking*").
- Entrega ordenada de las secuencias de bytes generadas por la aplicación ("stream oriented").
- Transmisión *full-duplex*.
- Mecanismo de detección y recuperación de errores (ARQ) con confirmaciones positivas ACKs (acumulativas) y "timeouts" adaptables.
- Servicio fiable → control de congestión y control de flujo con ventanas deslizantes con tamaño máximo adaptable.
- Incorporación de confirmaciones ("piggybacking").

TRANSMISSION CONTROL PROTOCOL (TCP)

Funcionalidades de TCP:

- Multiplexación/demultiplexación de aplicaciones.
- Control de la conexión (establecimiento y cierre).
- Control de errores y de flujo.
- Control de congestión.

TPDU TCP = **Segmento TCP**:



Cada segmento TCP se encapsula en un datagrama IP.

Multiplexación/demultiplexación de aplicaciones:

Transportar las TPDU al proceso correcto.

Existen puertos preasignados con servicios normalizados:

Puerto	Aplicación/Servicio	Descripción
20	FTP-DATA	Transferencia de ficheros: datos
21	FTP	Transferencia de ficheros: control
22	SSH	Terminal Seguro
23	TELNET	Acceso remoto
25	SMTP	Correo electrónico
53	DNS	Servicio de nombres de domino
80	HTTP	Acceso hipertexto (web)
110	POP3	Descarga de correo

La "conexión TCP" se identifica por: puerto e IP origen y puerto e IP destino (y opcionalmente protocolo)

Control de la conexión:

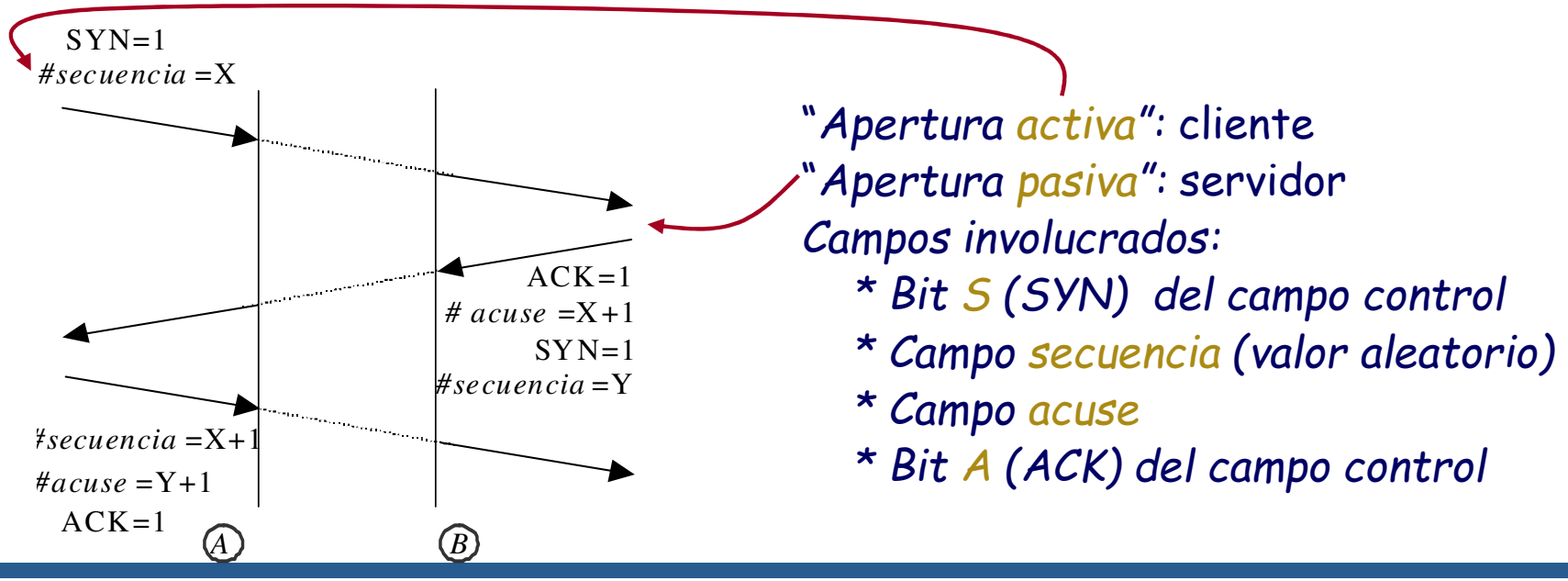
TCP ofrece un servicio orientado a conexión.

El intercambio de información tiene tres fases:

- Establecimiento de la conexión (sincronizar # de secuencia y reservar recursos).
- Intercambio de datos (full-duplex).
- Cierre de la conexión (liberar recursos).

¿Es posible garantizar un establecimiento/cierre fiable de la conexión sobre un servicio (IP) no fiable? NO.

Establecimiento de la conexión: *three-way handshake.*



"Apertura activa": cliente
"Apertura pasiva": servidor
Campos involucrados:
* Bit S (SYN) del campo control
* Campo secuencia (valor aleatorio)
* Campo acuse
* Bit A (ACK) del campo control

Control de la conexión. Números de secuencia.

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 "teóricamente" al azar; para evitar confusiones con solicitudes anteriores.

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 mecanismo de selección de los ISN es suficientemente fiable para proteger de coincidencias, 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.

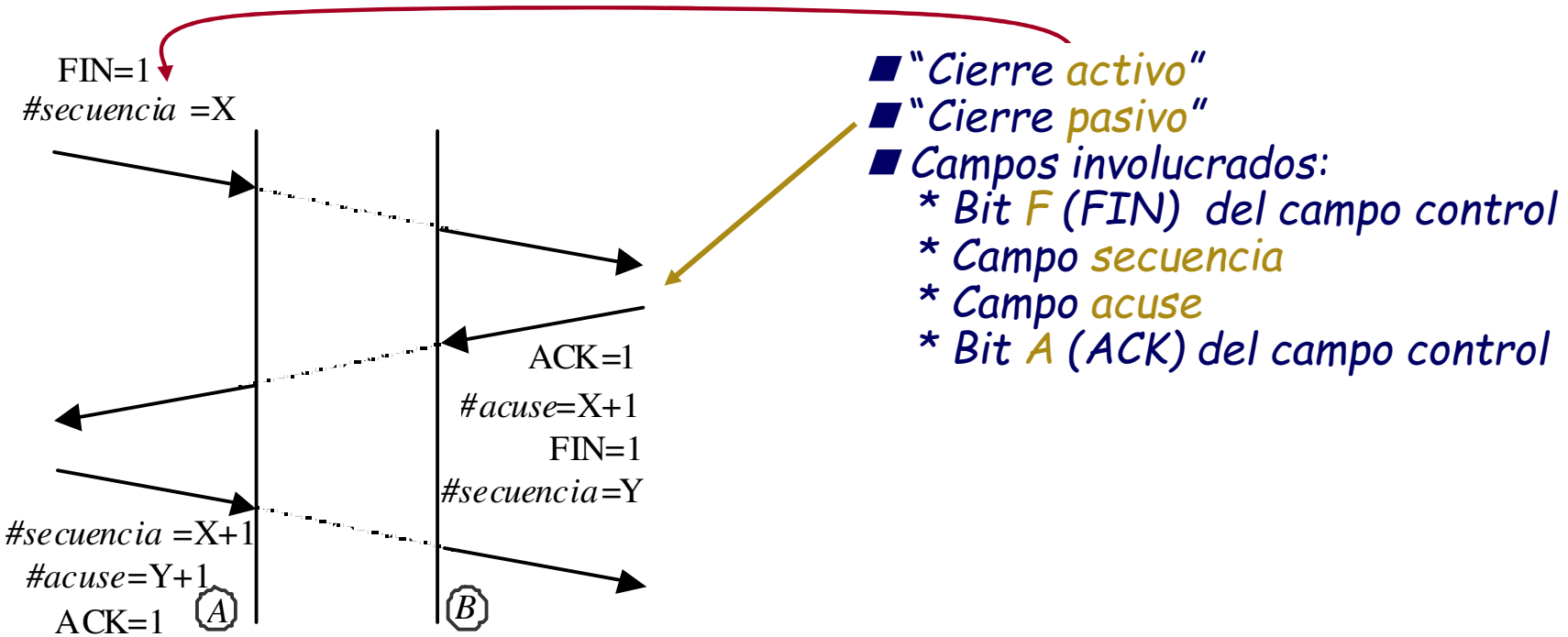
TCP incrementa el número de secuencia de cada segmento según los bytes que tenía el segmento anterior, con una sola excepción:

Los flags SYN y FIN, cuando están puestos, incrementan en 1 el número de secuencia.

La presencia del flag ACK no incrementa el número de secuencia.

Control de la conexión:

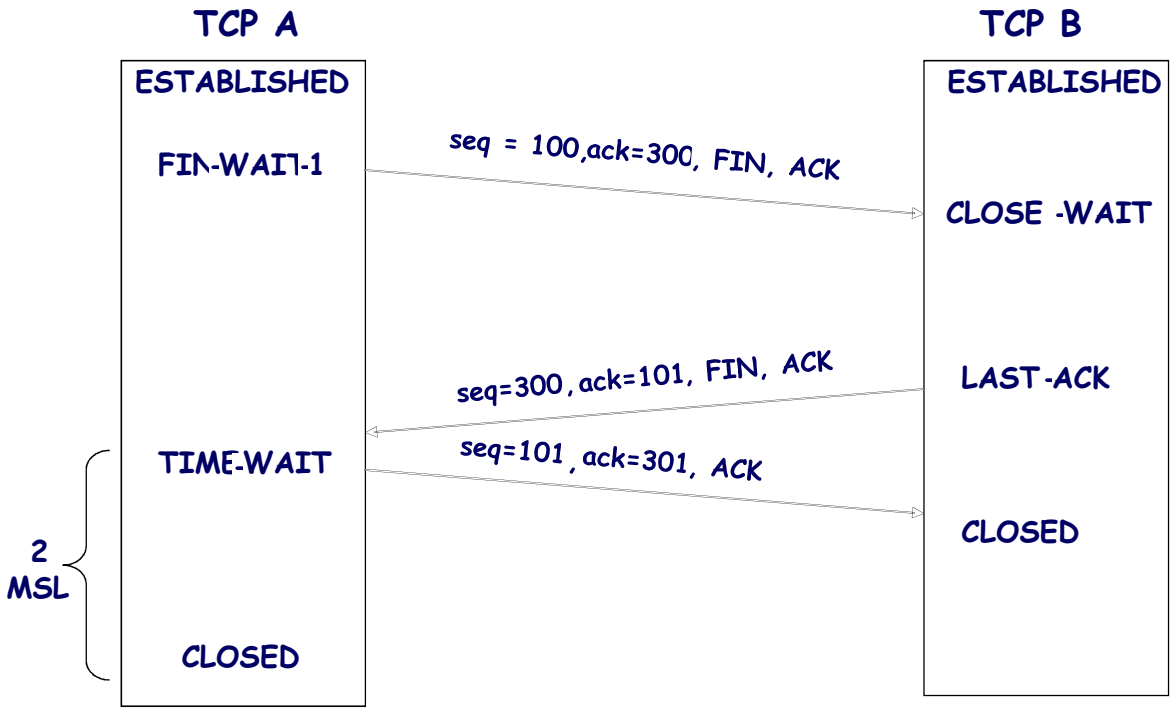
Cierre de la conexión: liberación de recursos.



Para evitar bloqueos por pérdidas, una vez comenzado el procedimiento de CIERRE se usan *timeouts* (ver diagrama de estados: *Maximum Segment LifeTime* = 2 min).

Control de la conexión:

Cierre de la conexión: caso normal.



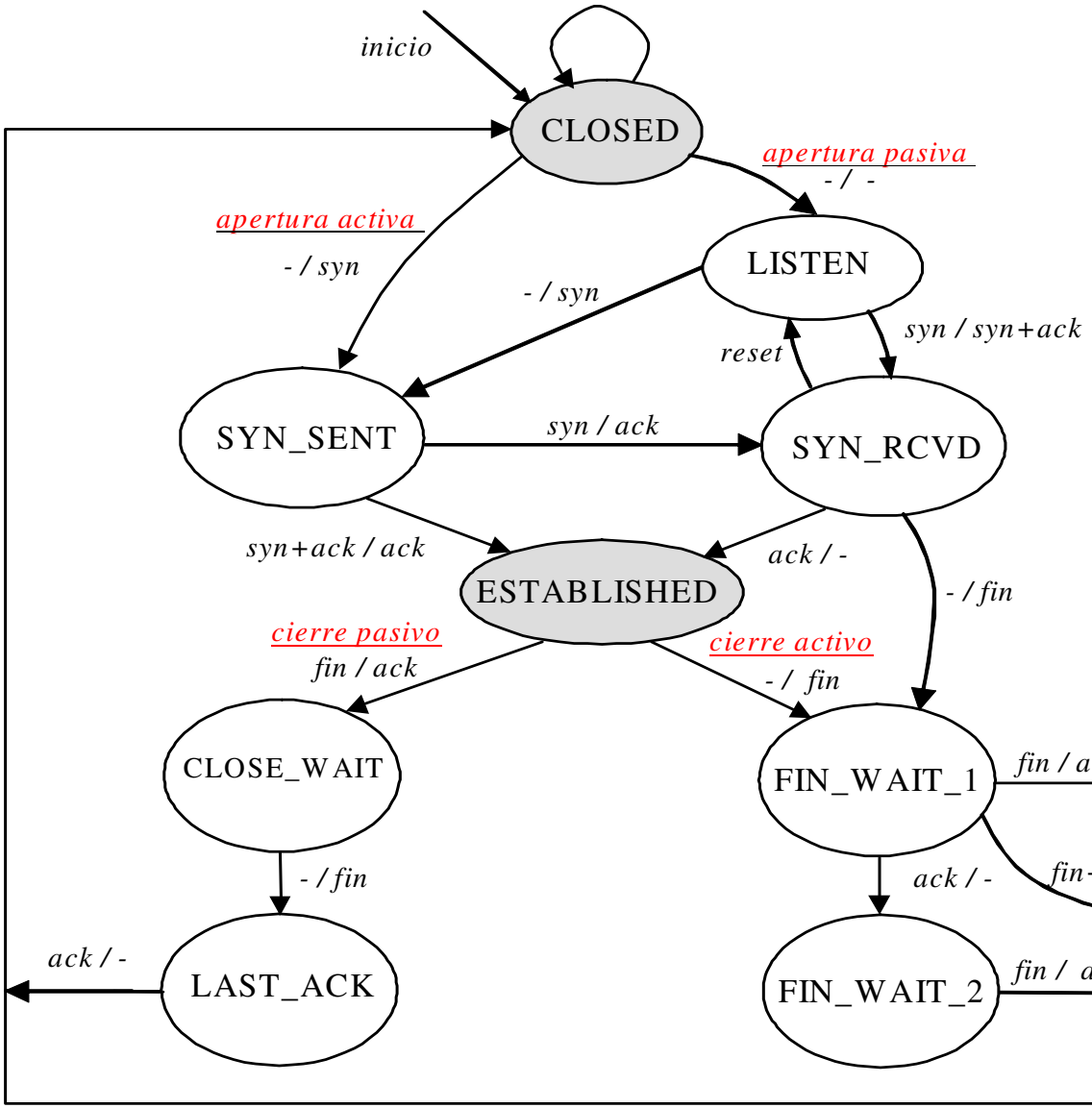
MSL: Maximum Segment Lifetime (normalmente 2 minutos)

Hay otras posibilidades de cierre de la conexión (ver el diagrama de estados siguiente).

9. Se desea transferir con protocolo TCP un archivo de L bytes usando un MSS de 536.

a) **¿Cuál es el valor máximo de L tal que los números de secuencia de TCP no se agoten?**

b) **Considerando una velocidad de transmisión de 155 Mbps y un total de 66 bytes para las cabeceras de las capas de transporte, red y enlace de datos, e ignorando limitaciones debidas al control de flujo y congestión, calcule el tiempo que se tarda en transmitir el archivo en A.**



Autómata de estados finitos TCP

Leyenda: *a/b*
Segmento a recibido, segmento b transmitido.

Control de errores y de flujo:

Mejorar rendimiento ⇒ ventana deslizante.

Control de errores: esquema ARQ con confirmaciones positivas y acumulativas.

Campos involucrados:

Campo *secuencia*: *offset* (en bytes) dentro del mensaje.

Campo *acuse*: número de byte esperado en el receptor.

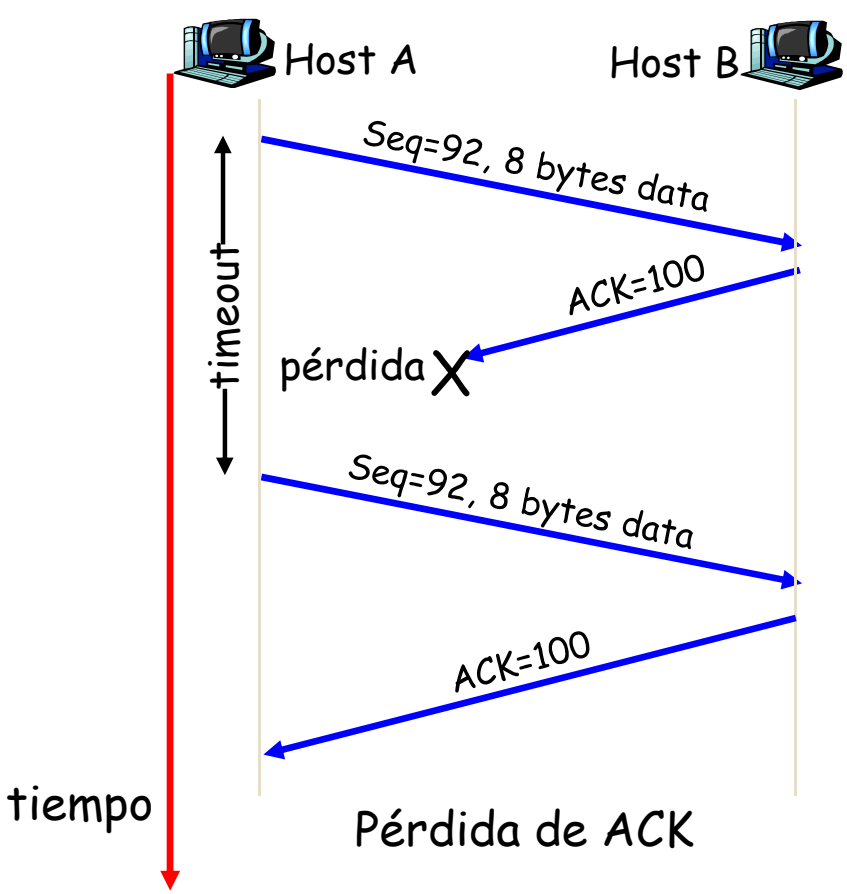
Bit *A* (ACK) del campo de *control.*

Campo *comprobación*: *checksum* de todo el segmento y uso de pseudo-cabecera TCP:

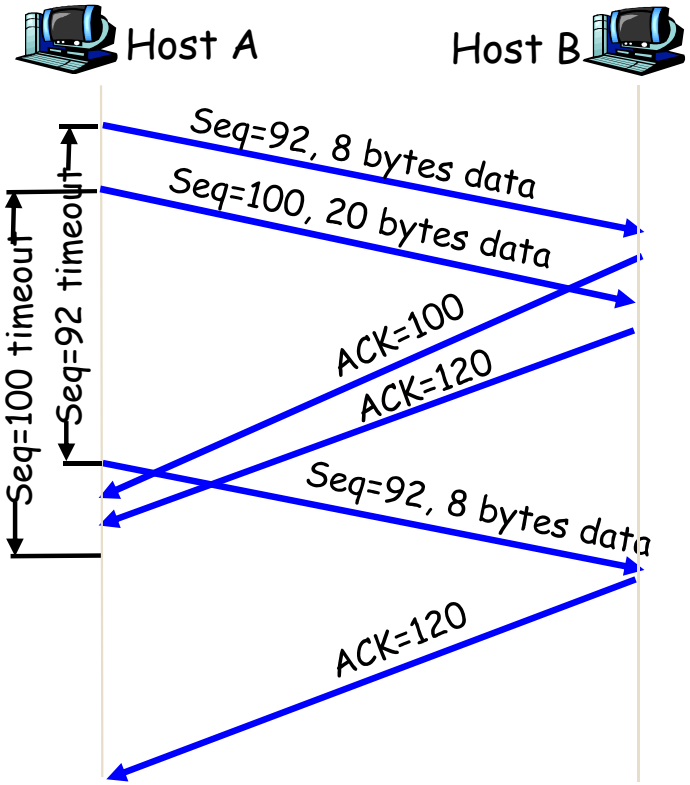
Iporigen		
IPdestino		
00...00	protocolo	longitudTCP

Control de errores y de flujo:

Control de errores: escenarios de retransmisión (gráficas © James F. Kurose).



Pérdida de ACK



timeout prematuro y
ACK acumulativo

Control de errores y de flujo:

Control de errores: generación de ACKs (RFC 1122, 2581).

Evento	Acción del TCP receptor
Llegada ordenada de segmento, sin discontinuidad, todo lo anterior ya confirmado.	Retrasar ACK. Esperar recibir al siguiente segmento hasta 500 mseg. Si no llega, enviar ACK.
Llegada ordenada de segmento, sin discontinuidad, hay pendiente un ACK retrasado.	Inmediatamente enviar un único ACK acumulativo.
Llegada desordenada de segmento con # de sec. mayor que el esperado, discontinuidad detectada.	Enviar un ACK duplicado, indicando el # de sec. del siguiente byte esperado.
Llegada de un segmento que completa una discontinuidad parcial o totalmente.	Confirmar ACK inmediatamente si el segmento comienza en el extremo inferior de la discontinuidad.

11. Los hosts A y B se están comunicando a través de una conexión TCP y B ya ha recibido y confirmado todos los bytes hasta el byte 126. Suponga que a continuación el host A envía dos segmentos seguidos a B que contienen, respectivamente, 70 y 50 bytes de datos. El envío de A es ordenado, el número de puerto origen en dichos segmentos es 302 y el de destino el 80. El host B envía una confirmación inmediata a la recepción de cada segmento de A, sin esperar el retardo de 500 ms del estándar.

a) **Especifique los números de secuencia de ambos segmentos.**

b) **Si el primer segmento llega antes que el segundo ¿cuál es el número de acuse y los puertos origen y destino en el primer ACK que se envía?**

c) **Si el segundo segmento llega antes que el primero ¿cuál es el número de acuse y los puertos origen y destino en el primer ACK que envía?**

d) **Imagine que los segmentos llegan en orden pero se pierde el primer ACK.**

Control de errores y de flujo:

Control de errores: ¿cómo estimar los "**timeouts**"?

Mayor que el tiempo de ida y vuelta (RTT).

Si es demasiado **pequeño**: **timeouts prematuros**.

Si es demasiado **grande**: **reacción lenta** a pérdida de segmentos.

Para situaciones cambiantes la mejor solución es la adaptable.

Control de errores y de flujo:

Control de errores: ¿cómo estimar los "**timeouts**"?

Kurose & Ross

***RTTmedido**: tiempo desde la emisión de un segmento hasta la recepción del ACK.*

$$RTTnuevo = (1-\alpha) \times RTTviejo + \alpha \times RTTmedido \ , \ \alpha \ \& \ \beta \in [0,1]$$

$$Desviacion_{nueva} = (1-\beta) \times Desviacion_{vieja} + \beta \times | RTTmedido - RTTnuevo |$$

$$Timeout = RTTnuevo + 4 * Desviacion$$

Problema con ACKs repetidos: ambigüedad en la interpretación.

Solución: Algoritmo de Karn, *actualizar el RTT sólo para los no ambiguos,* pero si hay que repetir un segmento incrementar el timeout eventualmente:

$$tout_{nuevo} = \gamma . \ tout_{viejo} \ , \ \gamma = 2.$$

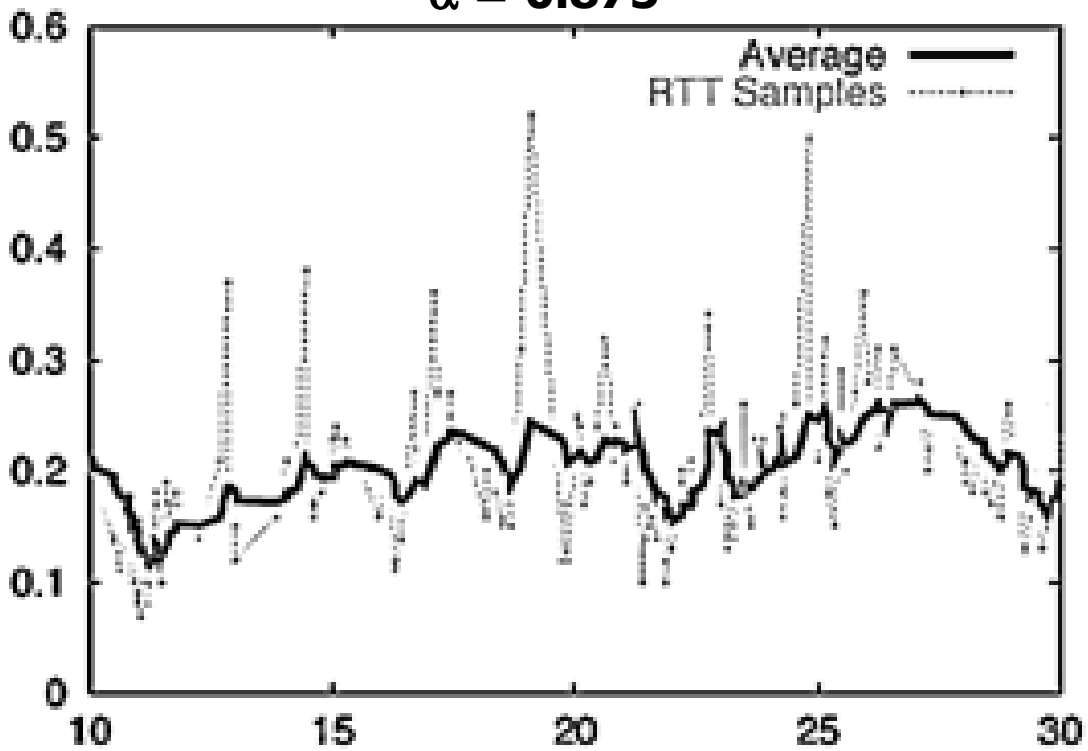
15. Si el RTT es 30 ms, la Desviación es 2 ms y se reciben ACKs tras 26, 32 y 24 ms, ¿Cuál será el nuevo RTT, Desviación y *timeout*? Usar α=0,125 y β=0,25. ¿Y si los dos primeros ACKs tienen el mismo número de acuse y se usa el algoritmo de Karn?

Control de errores y de flujo:

Control de errores: ¿cómo estimar los "timeouts"?

Ejemplo de RTT medidos y estimados entre Amherst, Massachusetts y St. Louis, Missouri.

α = 0.875



Control de errores y de flujo:

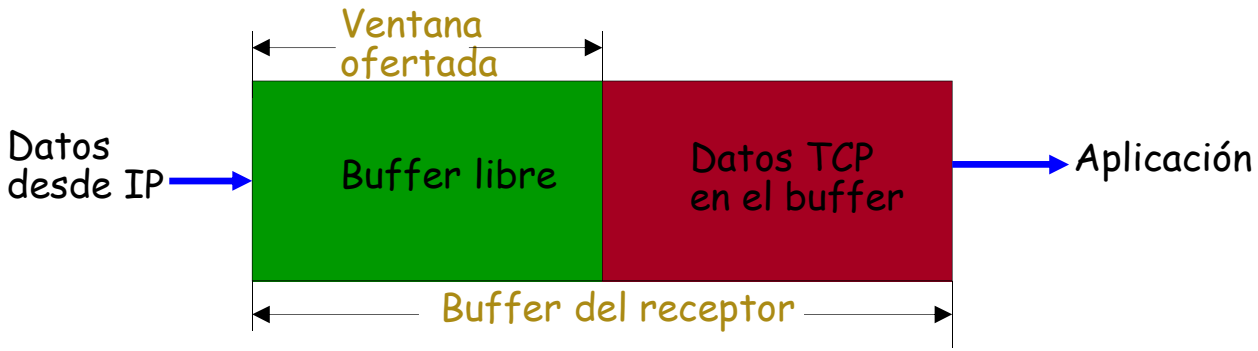
Control de flujo:

Procedimiento para evitar que el emisor sature al receptor con el envío de demasiada información y/o demasiado rápido.

Es un esquema crediticio: el receptor informa al emisor sobre los bytes autorizados a emitir sin esperar respuesta.

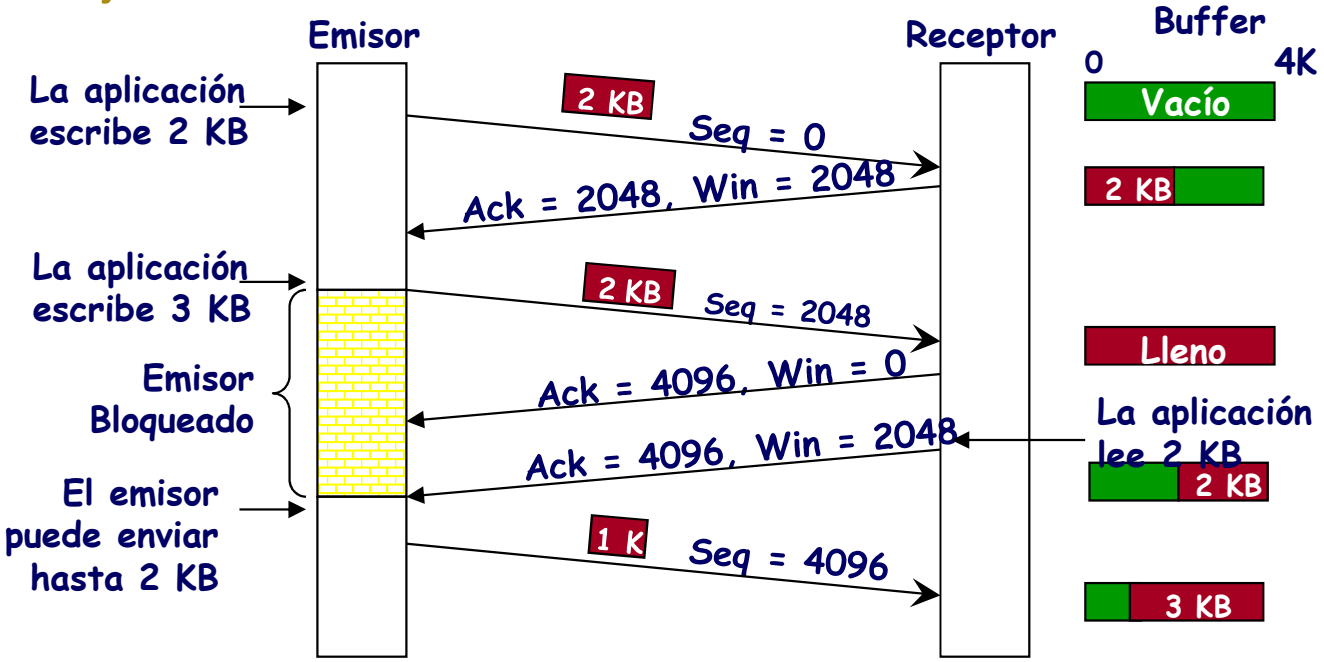
Se utiliza el campo *ventana*:

ventana útil emisor = ventana ofertada receptor – bytes en tránsito



Control de errores y de flujo:

Control de flujo:



Posible problema: *síndrome de la ventana tonta (RFC 813) si se utilizan segmentos muy pequeños.*

Posible mejora: *la ventana optimista* (*RFC 813*).

Es posible hacer entregas "no ordenadas": Bit *U* (URG), campo *puntero.*

Solicitar una entrega inmediata a la aplicación: bit *P* (PSH).

Control de congestión (RFC 2001):

Es un problema debido a la insuficiencia de recursos (ancho de banda de las líneas como buffer en *routers* y sistemas finales).

Es un problema diferente al control del flujo: involucra a la red y a los sistemas finales.

Puede tener naturaleza adelante-atrás.

Se manifiesta en pérdidas y/o retrasos en las ACKs.

Solución: en la fuente limitar de forma adaptable el tráfico generado.

La limitación se hace se hace mediante una aproximación conservadora:

por el tamaño de la ventana de emisión.

Importancia del producto Bandwidth-delay (RTT).

Control de congestión:

En el emisor se utilizan dos **`ventanas`** y un **`umbral`**.

`Bytes_permitidos_enviar` =
min{`VentanaCongestion`, `VentanaDelReceptor`}

`VentanaDelReceptor`: utilizada para el control de flujo (de tamaño variable) según el campo "`ventana`" recibido.

`VentanaCongestion`:
Inicialmente `VentanaCongestion = 1· MMS`

Inicio lento

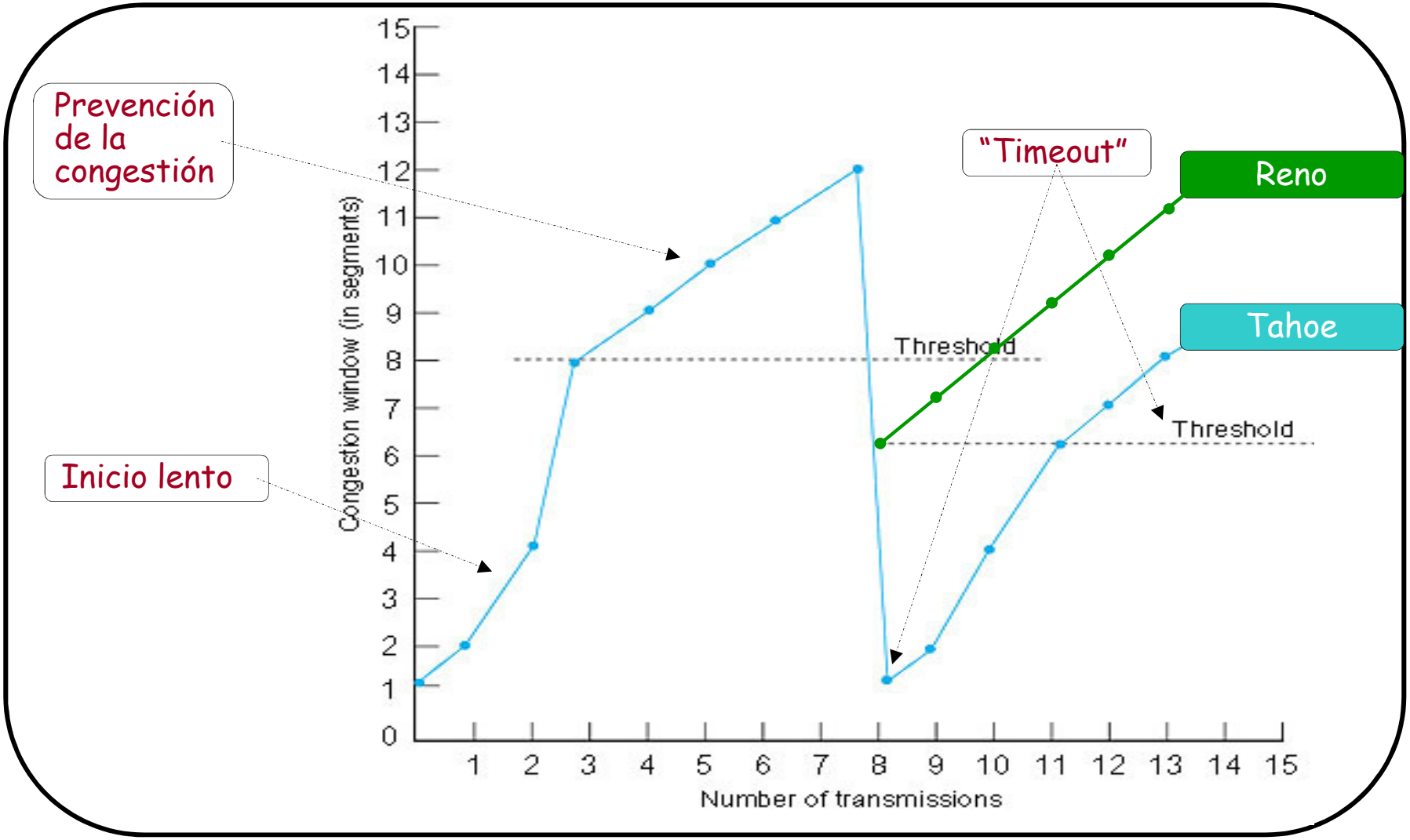
Si `VentanaCongestion < umbral`, por cada ACK recibido
`VentanaCongestion += MMS` **(crecimiento exponencial)**

Prevención de la congestión

Si `VentanaCongestion` > umbral, cada vez que se recibe **todos los ACKs pendientes**
VentanaCongestion += MMS (**crecimiento lineal**)

Si hay `timeout` entonces
`umbral=VentanaCongestion/2` y `VentanaCongestion = MMS`

Control de congestión (gráfica © James F. Kurose):



16. Teniendo en cuenta el efecto del inicio lento, en una línea sin congestión con 10 ms de tiempo de propagación, 1 Mbps de velocidad de transmisión y un MSS de 2KB, ¿cuánto tiempo se emplea en enviar 24 KB?

1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. **Extensiones TCP.**
5. Ejercicios.

- TCP se define con mútiples "Sabores"
- Los diferentes sabores no afectan a la interoperabilidad entre los extremos
- Desde cualquier versión de Linux con kernel mayor que la 2.6.19 se usa por defecto TCP CuBIC

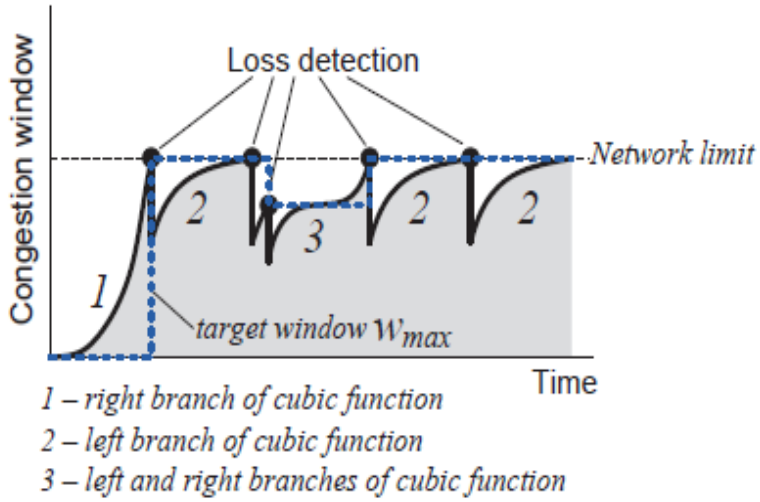
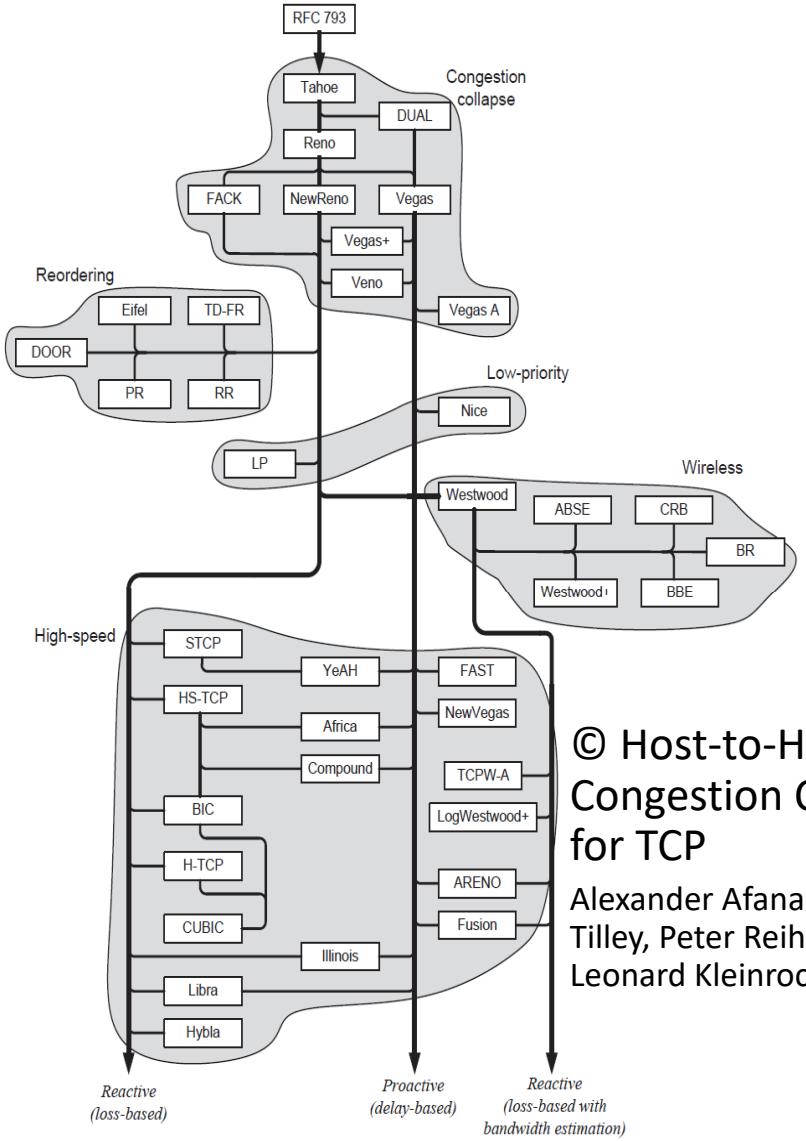


Fig. 50. Congestion window dynamics in CUBIC



57. Evolutionary graph of variants of TCP congestion control.

© Host-to-Host Congestion Control for TCP

Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock

Adaptación de TCP a redes actuales (RFC 1323).

Ventana escalada:

Opción TCP en segmentos SYN:

Hasta 2^{14}x2^{16} bytes (=2^{30} bytes=1GB) autorizados.

Estimación RTT:

Opción TCP de *sello de tiempo*, en todos los segmentos.

PAWS ("Protect Against Wrapped Sequence numbers"):

Sello de tiempo y rechazo de segmentos duplicados.

1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. Extensiones TCP.
5. **Ejercicios.**

10. Considere un enlace con una velocidad de transmisión de 1 Mbps, un tiempo de ida y vuelta (RTT) de 30 ms y segmentos fijos de 1500 bytes, incluyendo cabeceras y datos, ¿cuál tiene que ser el tamaño de la ventana para que la eficiencia en la transmisión (ratio entre el tiempo de transmisión y el tiempo total en el emisor) sea de al menos un 95%?

17. Suponiendo que la ventana de congestión es 18 KB y que se dispara un *timeout* ¿Cuánto será la ventana de congestión si las 4 siguientes ráfagas de transmisiones, donde se envía la ventana completa, son exitosas? Suponer que el MSS es 1 KB.

TEMA 3
CAPA DE TRANSPORTE EN INTERNET

Fundamentos de Redes

2017/2018