TEMA 3
CAPA DE TRANSPORTE EN INTERNET

Fundamentos de Redes

2021/2022

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.

Objetivo de este capítulo:

Comprender las funcionalidades y servicios de la capa de transporte.

- Conocer el concepto de puerto
- Identificar y distinguir un servicio orientado a conexión frente a no orientado a conexión
- Comprender cómo conseguir una transferencia de datos fiable (sin errores)
- Comprender cómo proporcionar control de flujo
- Comprender cómo proporcionar control de congestión
- Comprender cómo se han implementado estas funcionalidades y servicios en Internet

1. **Introducción.**
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de la 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:

  Ofrece una comunicación extremo a extremo (*end-to-end)*.

  Realiza la multiplexación/demultiplexación de aplicaciones → *puerto*.

- Protocolo UDP:

  Realiza la multiplexación/demultiplexación de aplicaciones**.**

  Ofrece un servicio no orientado a conexión, no fiable (sin recuperación de errores).

- Protocolo TCP:

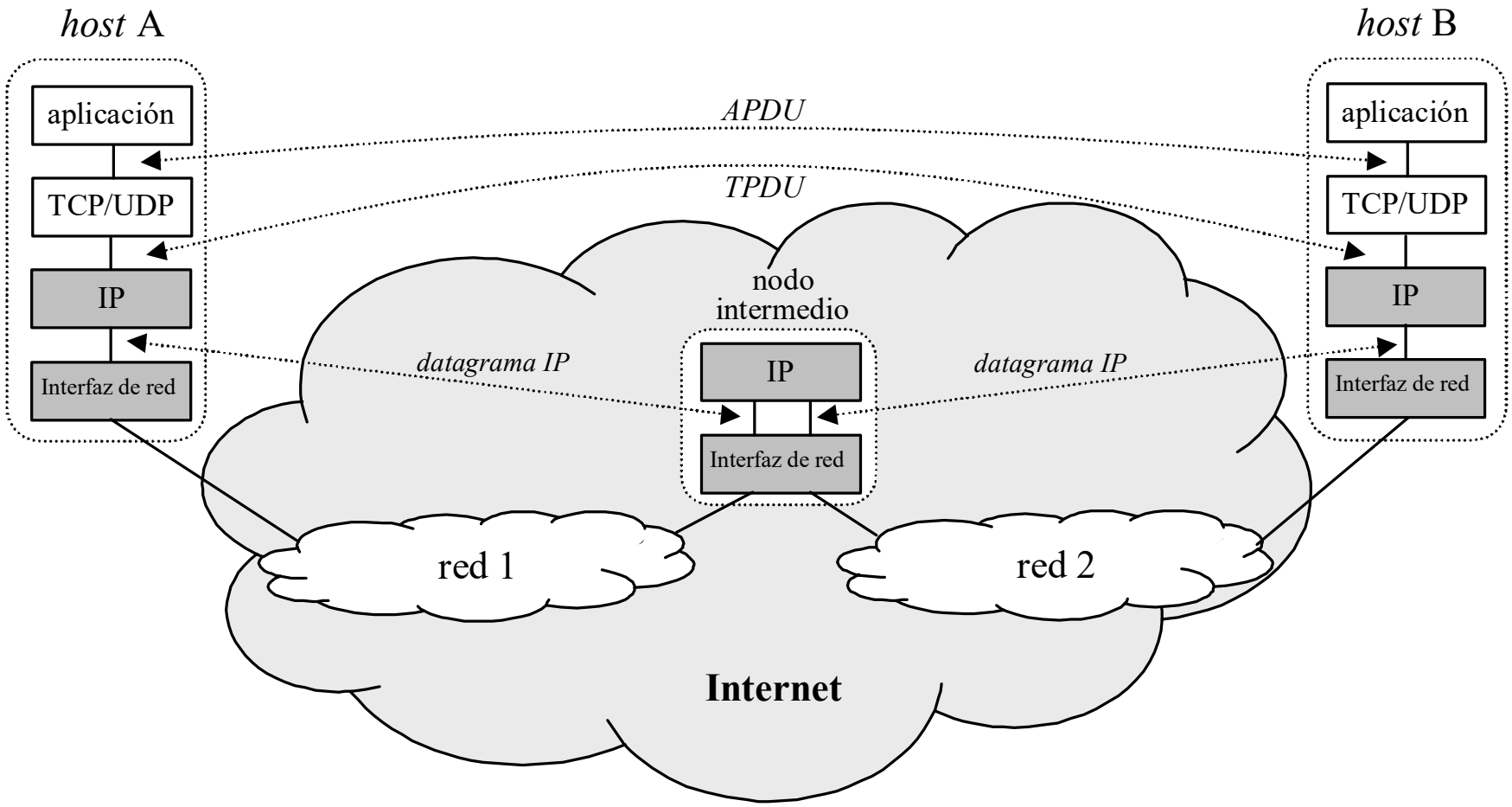
  Realiza la multiplexación/demultiplexación de aplicaciones**.**

  Ofrece un servicio orientado a conexión, fiable que incluye:

  Control de errores y de flujo.
  Control de la conexión.
  Control de congestión.

- Extensiones TCP

Capa de transporte: comunicación **extremo a extremo** (*end-to-end)*:



1. Introducción.

2. **Protocolo de datagrama de usuario (UDP).**

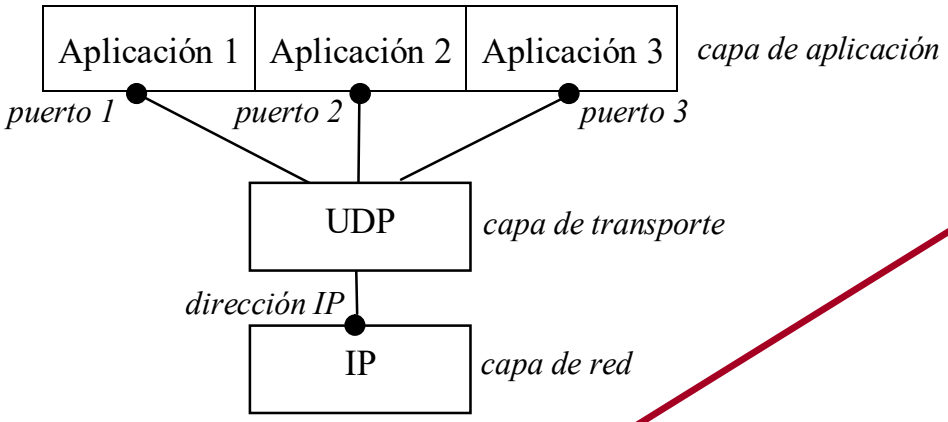
3. Protocolo de control de transmisión (TCP).
   1. Multiplexación/demultiplexación.
   2. Control de la conexión.
   3. Control de errores y de flujo.
   4. Control de congestión.

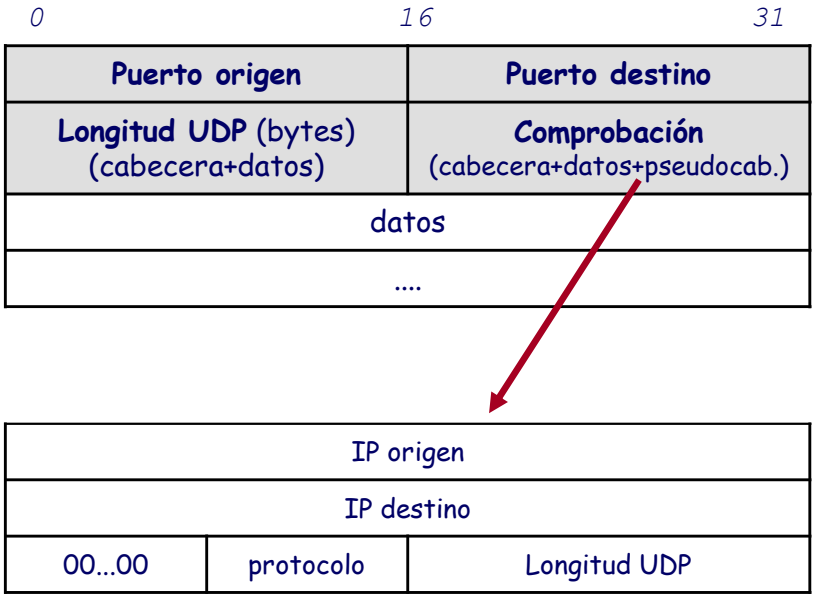
4. Extensiones TCP.

5. Ejercicios.

USER DATAGRAM PROTOCOL (UDP)

- "User Datagram Protocol": RFC 768.
- Ofrece una funcionalidad denominada como "*best-effort*" (*de buena voluntad*):
  - Es un servicio no orientado a conexión: no *hand-shaking*, no hay retardos de establecimiento, cada TPDU es independiente.
  - Resulta en un servicio no fiable: puede haber pérdidas.
  - No hay garantías de entrega ordenada.
  - No hay control de congestión, No hay control de flujo: entrega tan rápida como se pueda.
  - Realiza la multiplexación/demultiplexación: transportar las TPDU al proceso correcto.





Datagrama de usuario UDP.

USER DATAGRAM PROTOCOL (UDP)

- Multiplexación/demultiplexación: el objetivo es transportar las TPDU al proceso correcto. Para ello se usan los puertos → números enteros de 2 bytes que identifican al proceso origen y al proceso destino.
  - 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, que como sabemos son tolerantes a fallos y muy sensibles a los retardos (no admiten recuperaciones de errores)
- Cada segmento UDP se encapsula en un datagrama IP.

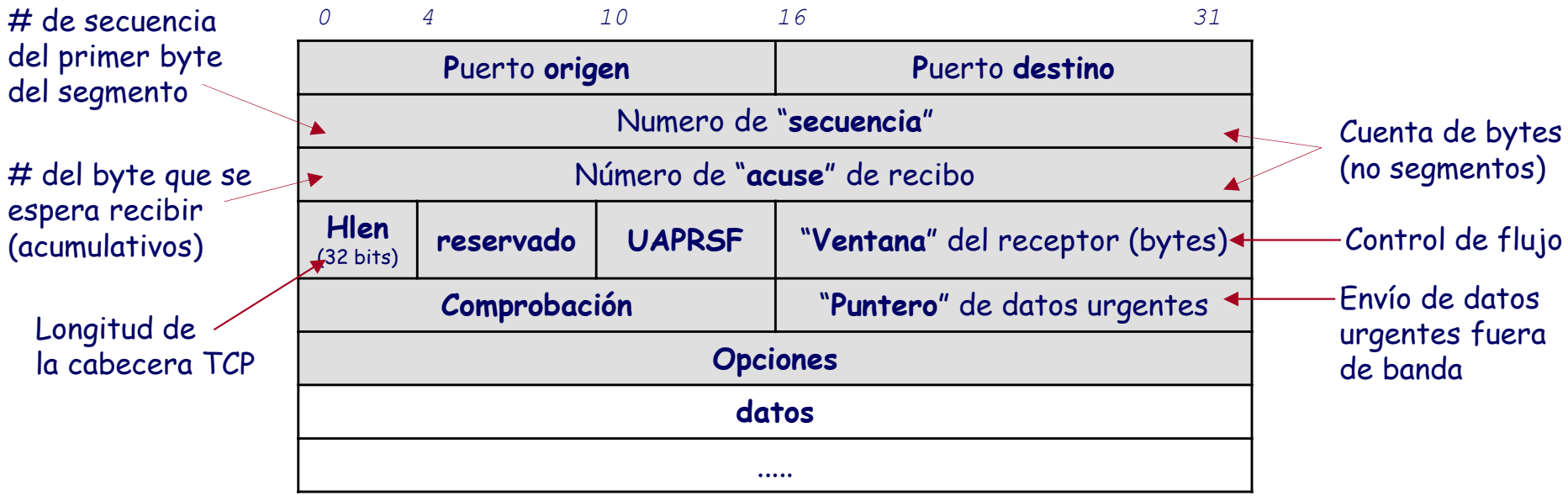
1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. **Protocolo de control de transmisión (TCP).**
    1. Multiplexación/demultiplexación.
    2. Control de la conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

Características principales del TCP: RFC 793 (1122, 3168, 6093, 6528).

- Ofrece un servicio punto a punto. No sirve para comunicaciones *multicast* (de uno a muchos).
- Implica un servicio orientado a conexión (exige un estado común entre el emisor y el receptor: "*hand-shaking*"). 3 fases: establecimiento, intercambio de datos y cierre
- Garantiza la entrega ordenada de las secuencias de bytes generadas por la aplicación ("*stream oriented*").
- Opera en transmisión *full-duplex*.
- Incluye mecanismos de detección y recuperación de errores (ARQ) con confirmaciones positivas ACKs (acumulativas) y "*timeouts*" adaptables.
- Ofrece un servicio fiable → control de congestión y control de flujo con ventanas deslizantes con tamaño máximo adaptable.
- Usa la técnica de Incorporación de confirmaciones ("*piggybacking*").
- Para mejorar su eficacia TCP se ADAPTA a las condiciones de la red DINÁMICAMENTE

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.
- Las TPDUs de TCP se denominan segmentos TCP:



Cada segmento TCP se encapsula en un paquete (denominado datagrama) IP.

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

TRANSMISSION CONTROL PROTOCOL (TCP)

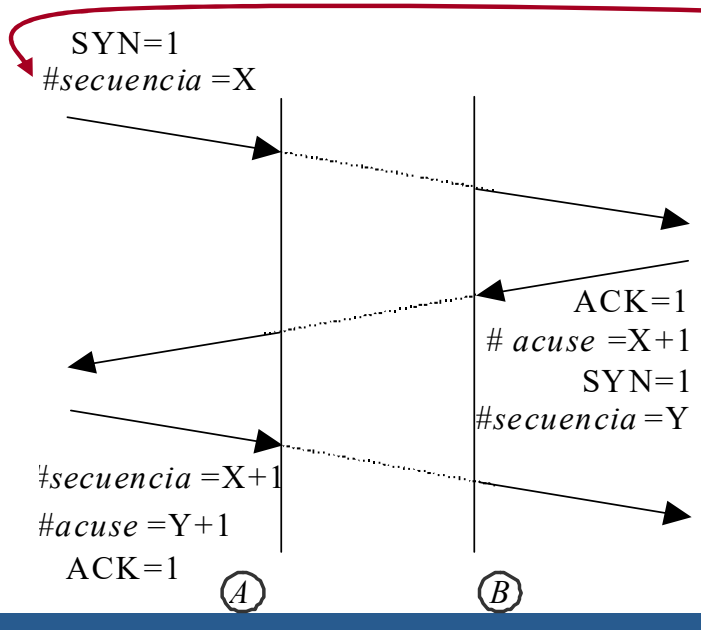
- Multiplexación/demultiplexación de aplicaciones: el objetivo es transportar las TPDUs (segmentos TCP) al proceso correcto. Para ello se usan los puertos → números enteros de 2 bytes que identifican al proceso origen y al proceso destino.
    - Existen puertos preasignados con servicios normalizados:

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

    - Otros puertos (>1024) están a libre disposición del desarrollador
- Cada "conexión TCP" se identifica por: puerto e IP origen y puerto e IP destino

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

- Control de la conexión:
  - TCP ofrece un servicio orientado a conexión.
  - El intercambio de información tiene tres fases:
    1. Establecimiento de la conexión (sincronizar # de secuencia y reservar recursos).
    2. Intercambio de datos (full-duplex).
    3. 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:

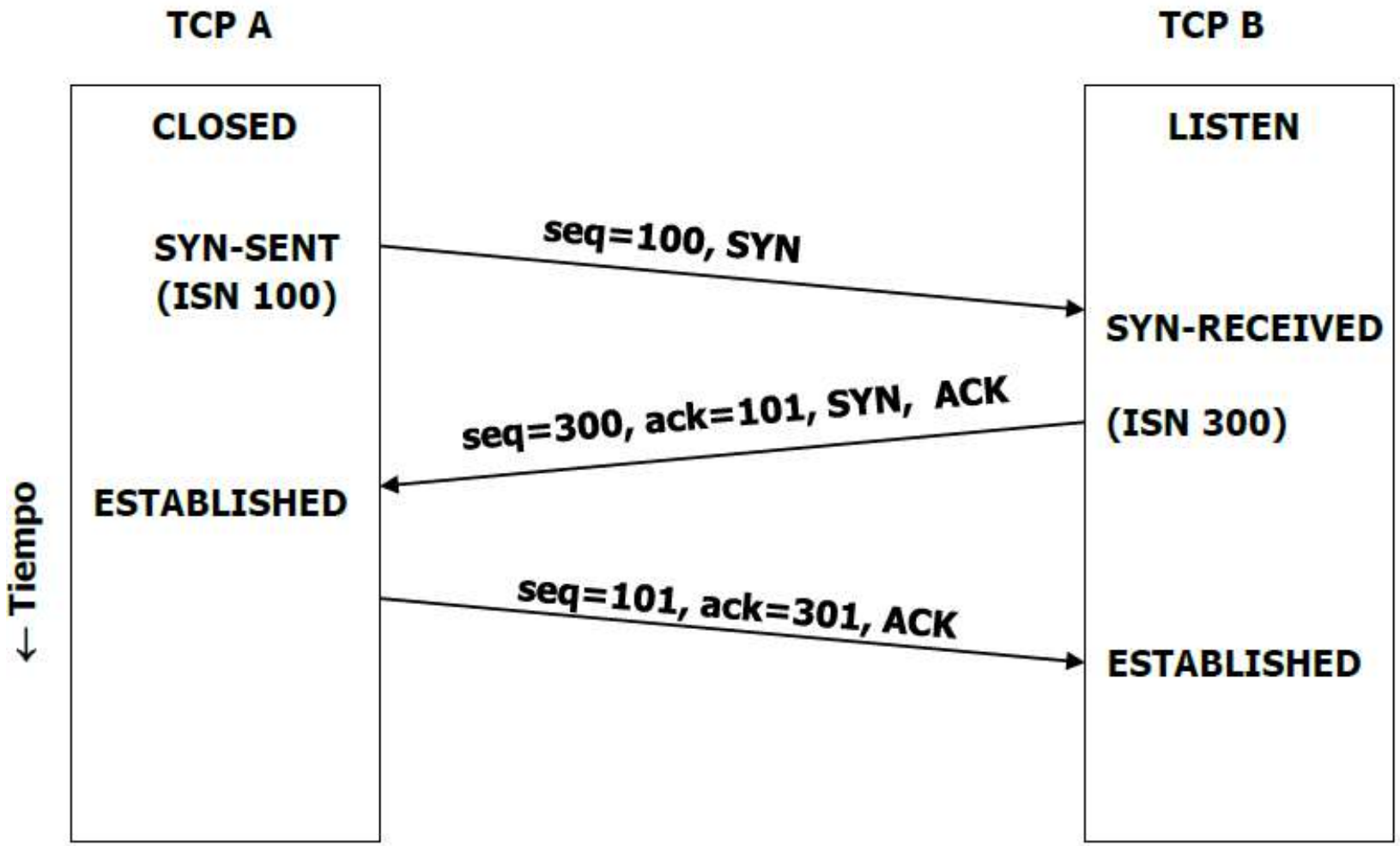
* *Bit S (SYN) del campo control*
* *Campo secuencia (valor aleatorio)*
* *Campo acuse*
* *Bit A (ACK) del campo control*

¿Qué debilidades tiene el establecimiento?

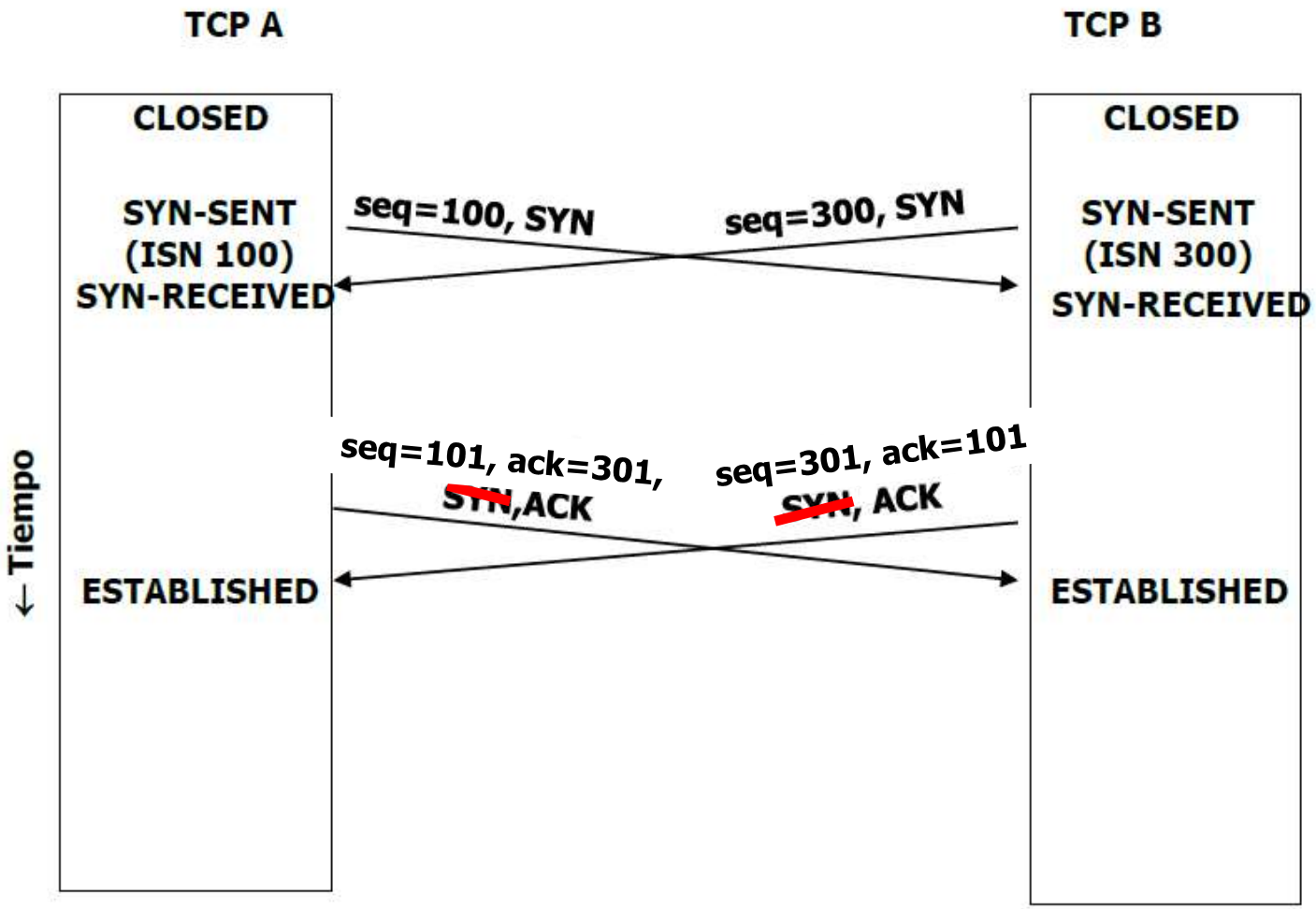
Establecimiento de la conexión. Números de secuencia.

- 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 incrementan en 1 el número de secuencia.
- Los segmentos ACK (sin datos) no incrementan el número de secuencia.

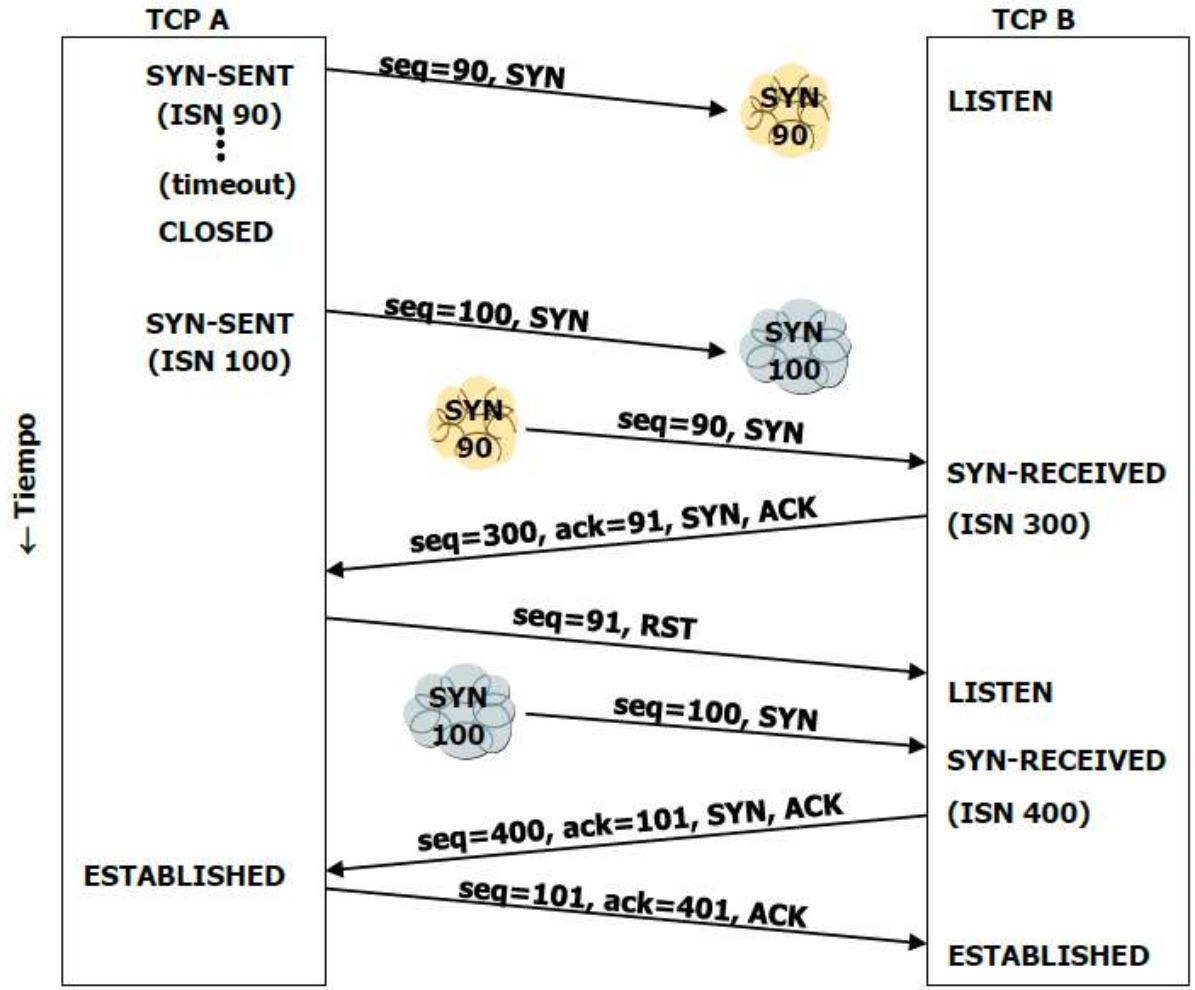
- Establecimiento de la conexión. Caso sin incidencias (normal):



- Establecimiento de la conexión. Caso de conexión simultánea:

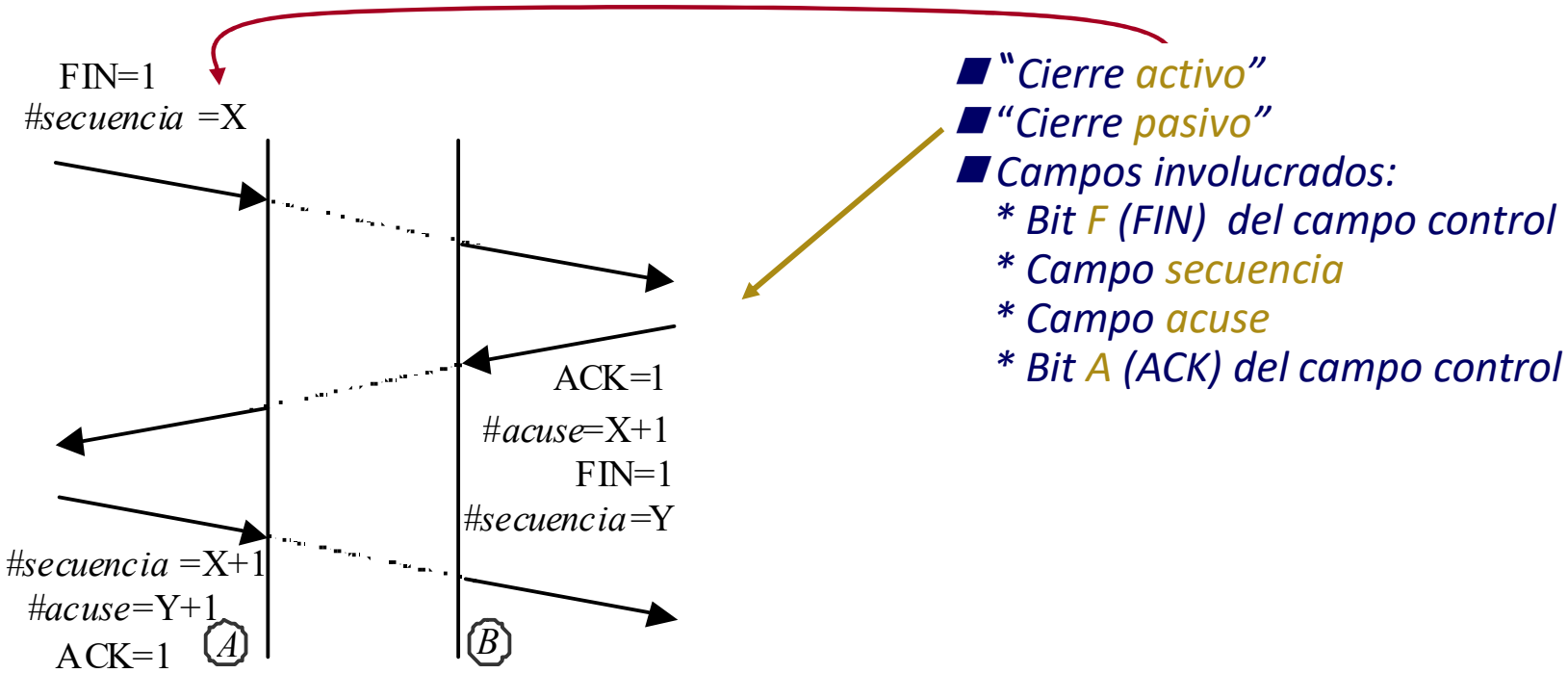


- Establecimiento de la conexión. Caso con SYN retrasados y duplicados:



- Control de la conexión:

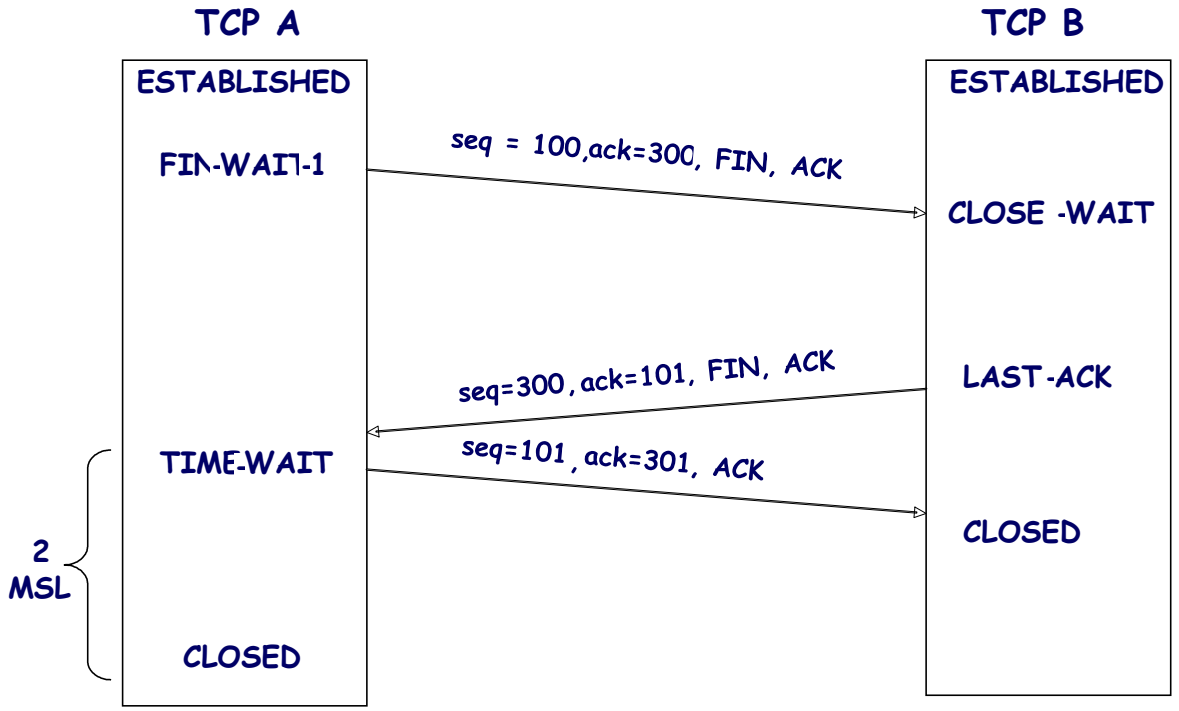
  Cierre de la conexión: liberación de recursos. Si no se hace ordenadamente puede provocar pérdidas de información



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

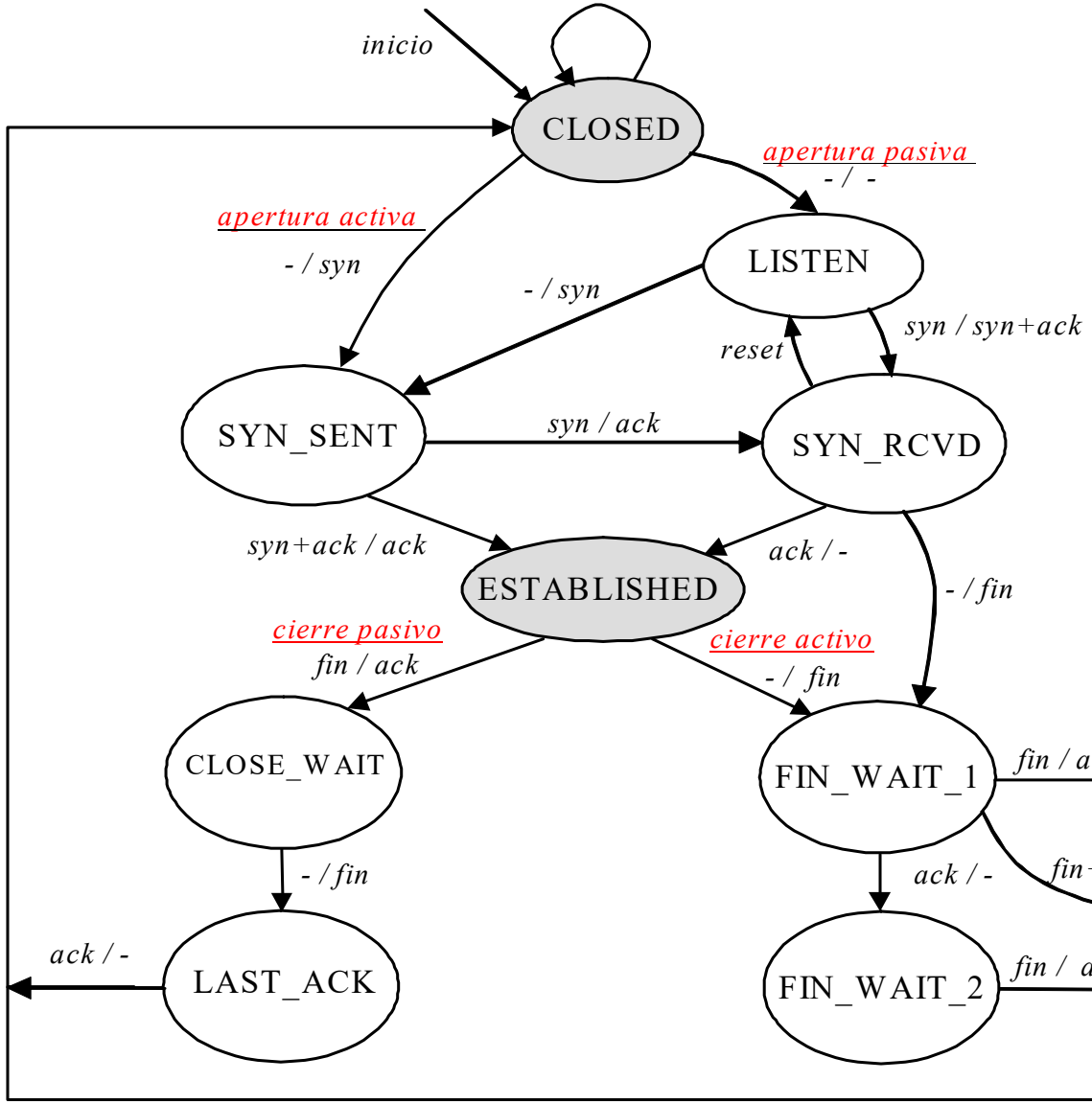
Control de la conexión:

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).



Autómata de estados finitos TCP

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

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

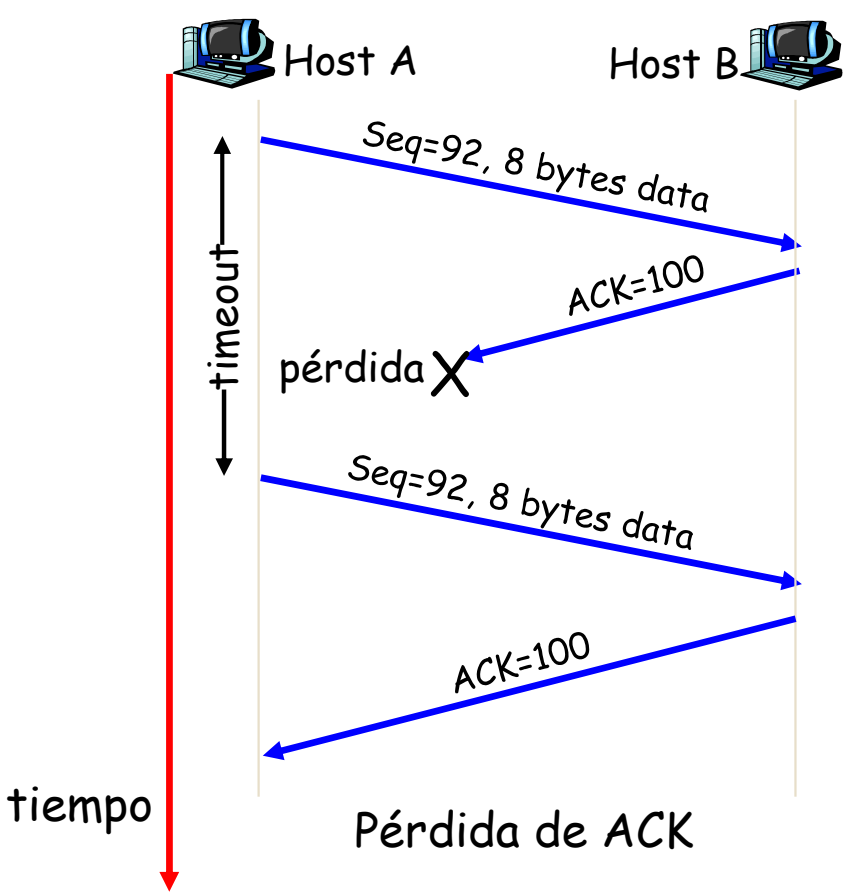
Control de errores y de flujo:

- 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:

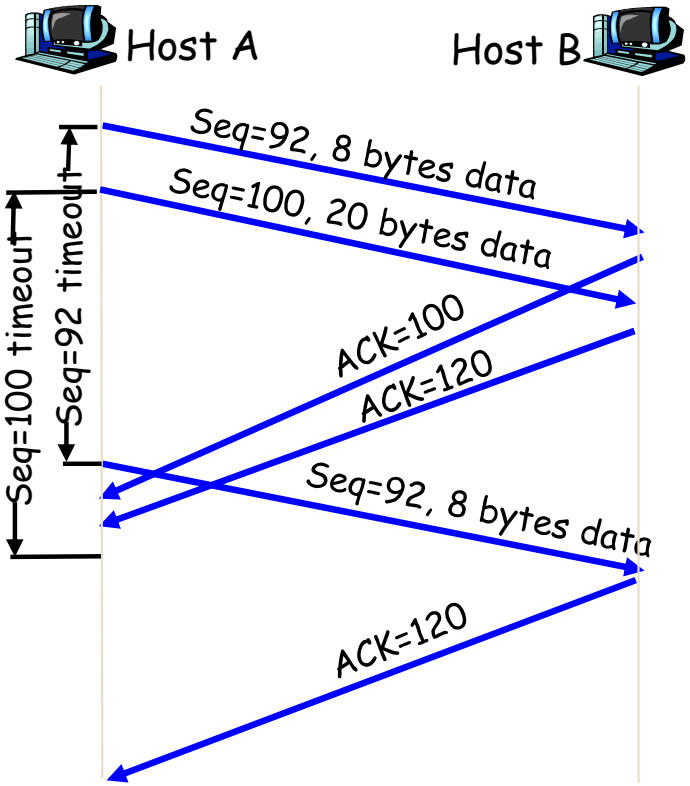
Iporigen		
IPdestino		
00...00	protocolo	longitudTCP

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.

Control de errores y de flujo:

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

Debe ser mayor que el tiempo de ida y vuelta (RTT, Round Trip Time), pero ¿cuánto?

Si es demasiado **pequeño**: ***timeouts* prematuros** → retransmisiones innecesarias

Si es demasiado **grande**: **reacción lenta** a pérdida de segmentos → baja eficacia

Para situaciones cambiantes …..la mejor solución es adaptarse dinámicamente.

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 duplicar el *timeout*:

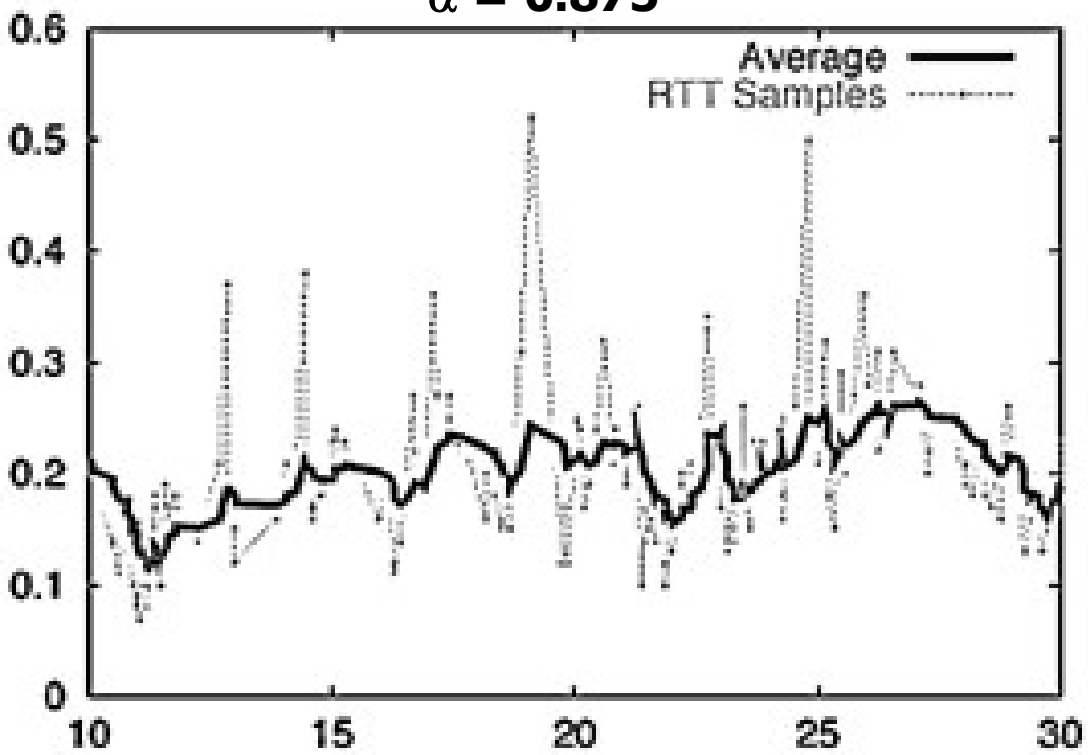
$$tout_{nuevo} = \gamma \,.\, tout_{viejo} \;,\; \gamma = 2.$$

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**

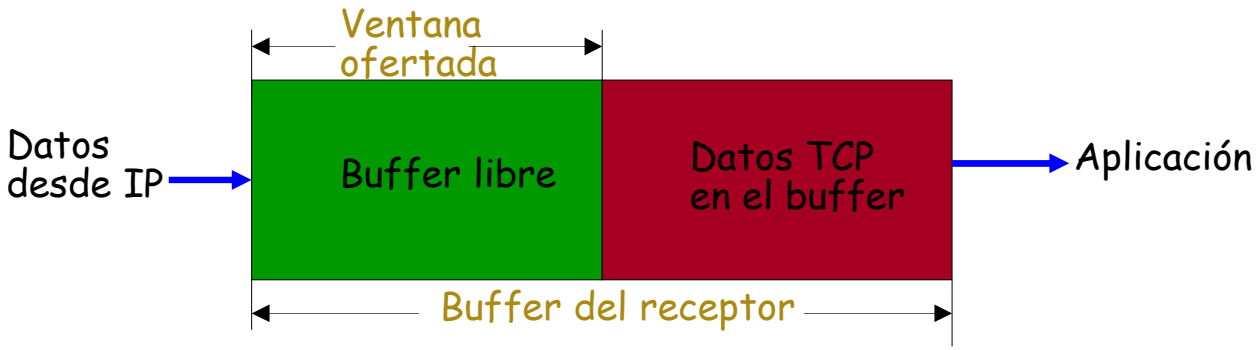


Control de errores y de flujo:

Control de flujo:

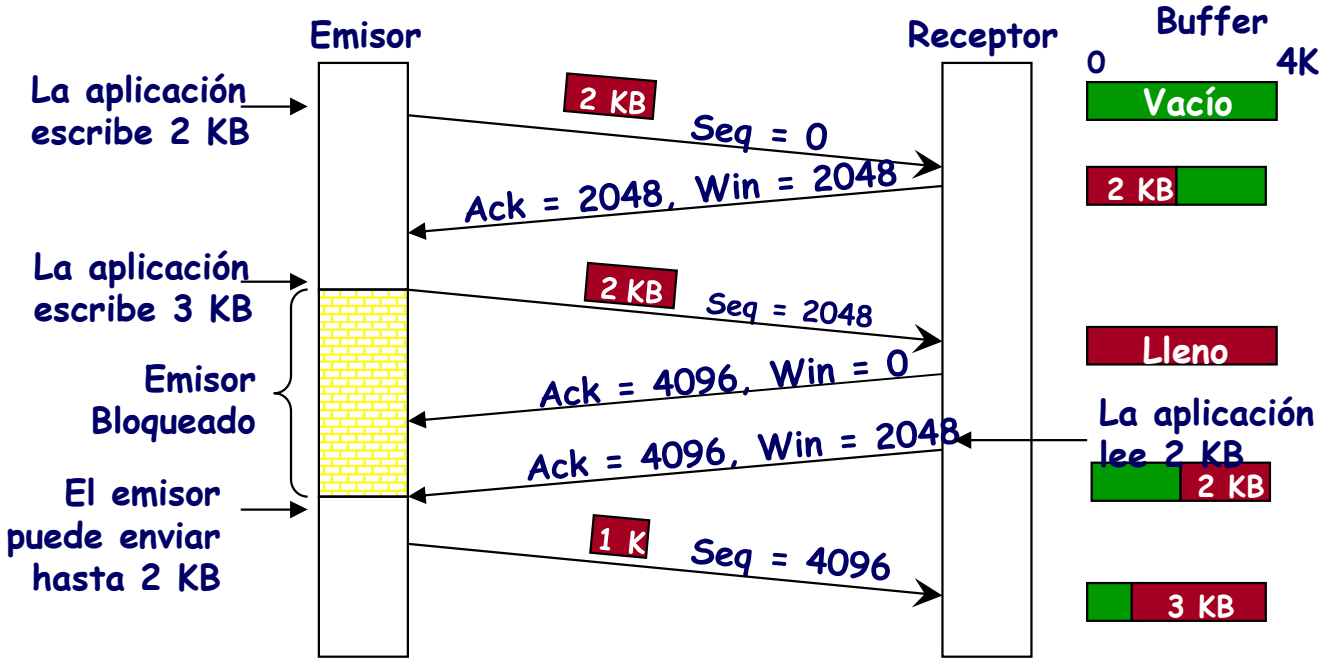
- Procedimiento para evitar que el emisor sature al receptor con el envío de demasiada información y/o demasiado rápido.
- Es un esquema crediticio: el receptor informa al emisor sobre los bytes autorizados a emitir sin esperar respuesta.
- Se utiliza el campo *ventana*:

ventana útil emisor = ventana ofertada receptor – bytes en tránsito



Control de errores y de flujo:

Control de flujo:

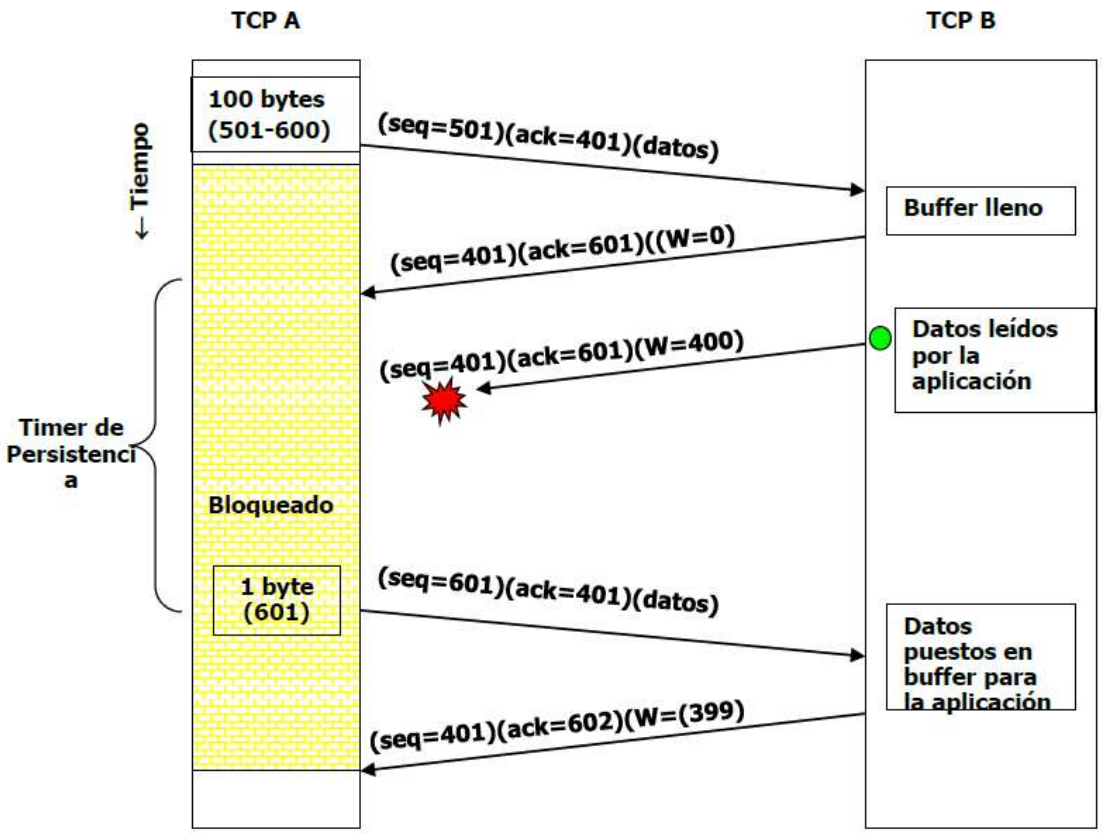


- ¿Alguna debilidad en el control de flujo?
- ¿Y si se pierde el anuncio de WIN = 2048? → !Bloqueo! ¿Cómo evitarlo? →

  Temporizador de persistencia

Control de errores y de flujo:

Control de flujo: temporizador de persistencia



- 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).

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

Control de congestión (RFC 2001):

- Es un problema debido a la insuficiencia de recursos (la capacidad o velocidad de transmisión de las líneas y el *buffer* en *routers* y *hosts* no son infinitos).
- Es un problema diferente al control del flujo: el control de congestión es para proteger a la red debido a sus limitaciones
- Puede tener naturaleza adelante-atrás, aunque en IP no
- Los episodios de congestión se manifiestan en retrasos en las ACKs y/o pérdidas de segmentos, dependiendo del nivel de severidad del episodio
- Solución extremo a extremo: en el emisor limitar de forma adaptable el tráfico generado para evitar pérdidas, pero siendo eficaz
- La limitación se hace se hace mediante una aproximación conservadora: limitando el tamaño de la ventana de emisión.
- ¿Qué es el producto *BandWidth-Delay* (RTT)? ¿Por qué es importante?

Control de congestión: procedimiento de prueba y error →

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 (ver pp. 12)

`VentanaCongestion`:

Inicialmente `VentanaCongestion = 1 · MSS`

Inicio lento

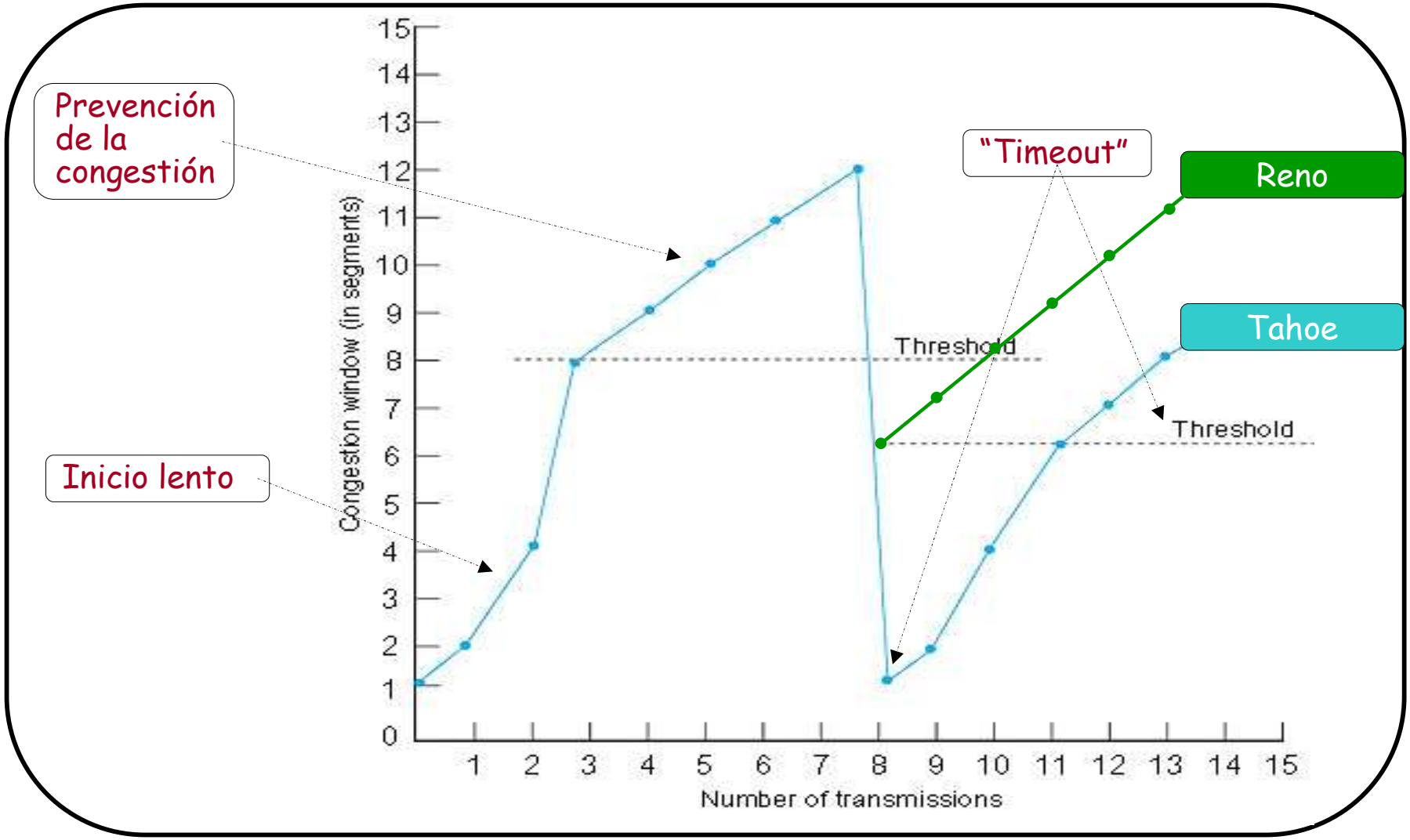
Si `VentanaCongestion < umbral`, por cada ACK recibido
`VentanaCongestion += MMS` **(crecimiento exponencial)**

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):



1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de la 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

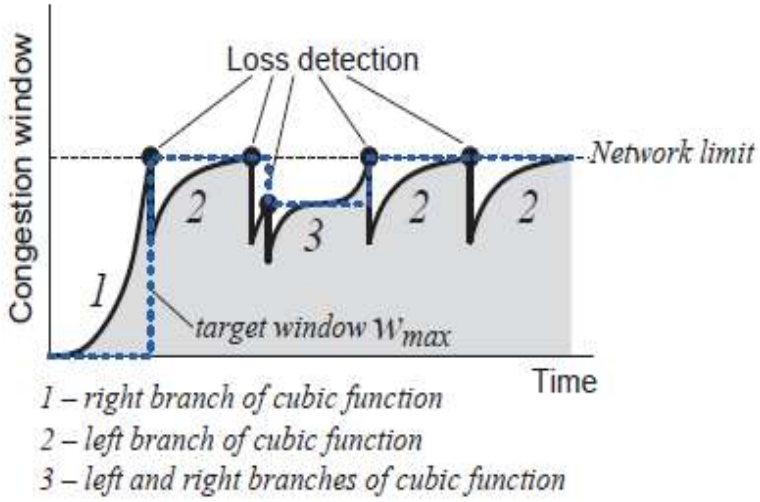
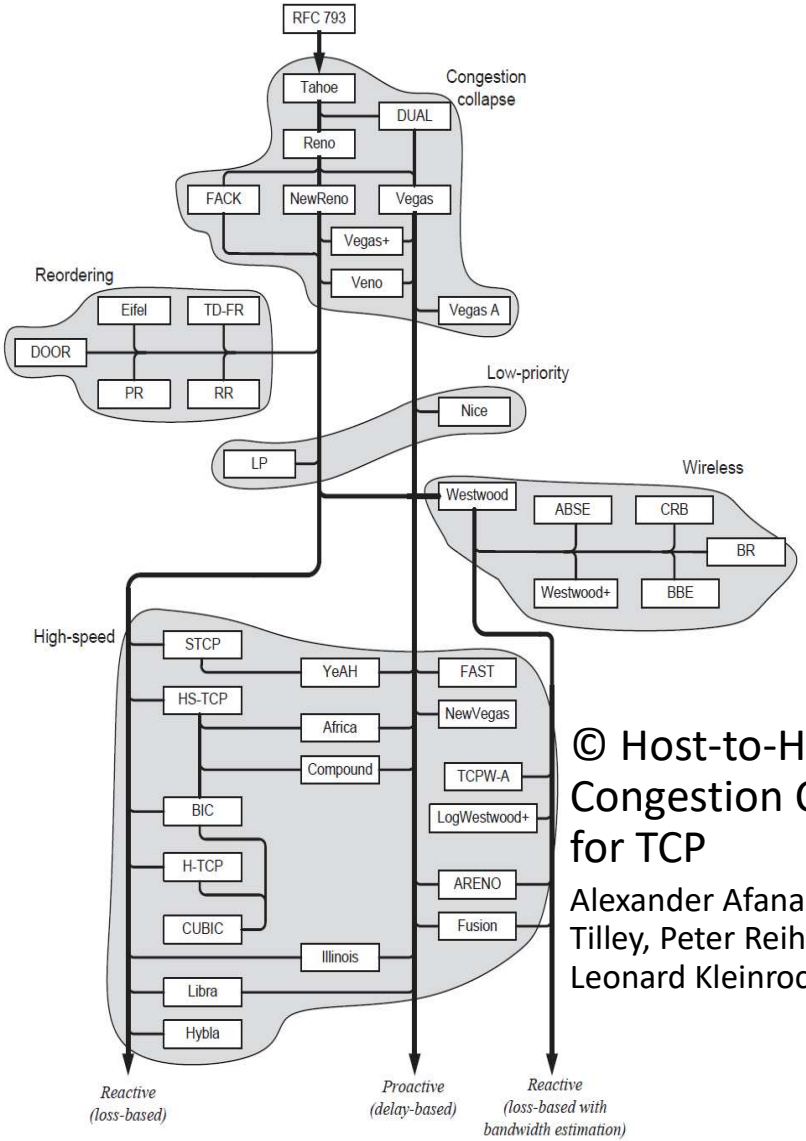


Fig. 50. Congestion window dynamics in CUBIC



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

EXTENSIONES TCP

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.

SACKS ("ACKs selectivos"). RFCs 2018 y 2883

Referencias:

- RFCs
- /usr/src/linux-2…/net/ipV4/tcp.c
- /usr/include/netinet/tcp.h
- Herramientas de análisis: wireshark y tcpdump

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

TEMA 3
CAPA DE TRANSPORTE EN INTERNET

Fundamentos de Redes

2021/2022