

Índice hora 4

Hora 1

- 1 Paradigmas de comunicaciones
 - 1.1 Paradigma cliente/servidor
 - 1.2 Paradigma Peer-to-Peer (P2P)
- 2 Multiplexación por puerto
- 3 UDP
 - 3.1 Cabecera UDP
 - 3.2 Ejemplo de servicio UDP
 - 3.3 Cuándo usar UDP

Hora 2

- 4 TCP
 - 4.1 Cabecera TCP
 - 4.2 Opciones cabecera TCP

Hora 3

- 4.3 Conexiones TCP
- 4.4 Diagrama de transición de estados de TCP
- 4.5 Transferencia interactiva

Hora 4

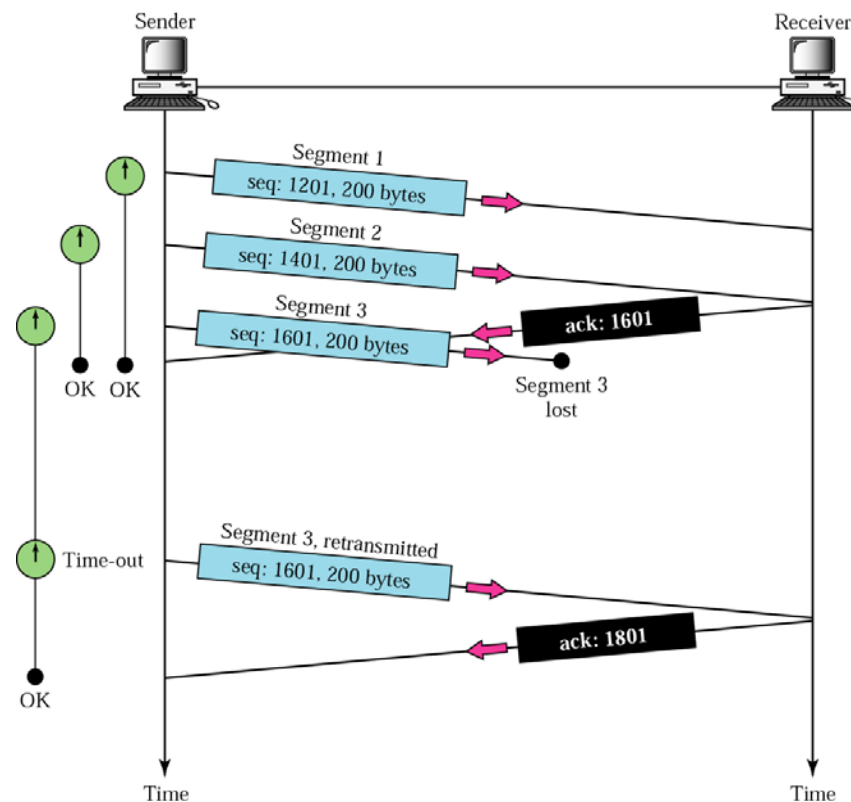
- 4.6 Fiabilidad en TCP
- 4.7 Transferencia masiva
 - 4.7.1 Transferencia normal
 - 4.7.2 Control de flujo
 - 4.7.3 Control de congestión
- 4.8 Producto $RTT \times BW$
- 4.9 Ejemplo de traza TCP

Objetivos

- Presentar los mecanismos de TCP para detección de pérdidas
- Identificar la problemática en transferencias de datos masivas
 - Congestionar al receptor: control de flujo
 - Congestionar la red: control de congestión
- Presentar los mecanismos de TCP para control de flujo:
 - Ventana anunciada por el receptor
 - Mecanismo de ventana deslizante
- Introducir los mecanismos de TCP para el control de congestión
 - Ventana de congestión
- Aprender a dimensionar el buffer de recepción de TCP para que no determine la velocidad de la conexión.
 - Producto $RTT \times BW$

4.6 Fiabilidad en TCP

- Los paquetes de datos o ACKs se pueden perder.
- Un temporizador de retransmisión (RTO, Retransmission Timeout) es el encargado de retransmitir aquellos paquetes que se estiman perdidos.

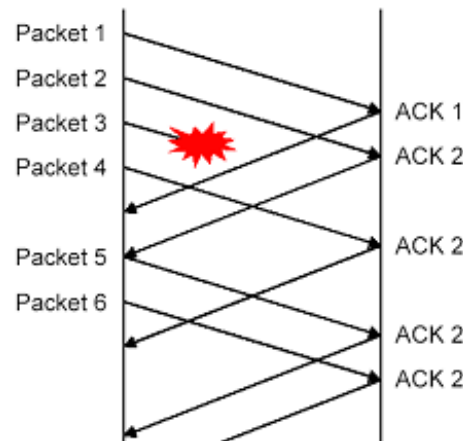


Fiabilidad en TCP

- Si el tiempo desde que se envió un segmento de datos hasta el momento actual supera el RTO, es indicador de que el paquete de datos o su ACK se ha perdido.
 - No hay forma de distinguir cual de los dos se ha perdido.
 - Si se pierde el ACK con la retransmisión se están mandando datos redundantes que ya posee el receptor.
 - Menor probabilidad de pérdida de paquetes de ACK al ser pequeños.
- El RTO se calcula dinámicamente en función del RTT medido en la conexión. Típicamente se usan valores $RTO=2*RTT$
- Vencido el RTO, se retransmite el paquete y se espera un nuevo RTO que sigue un backoff exponencial ($RTO_{nuevo}=2*RTO_{anterior}$); se retransmitió así varias veces hasta algunos minutos de espera según la implementación, tras lo cual se da la conexión por cerrada.
- Repaquetización: al retransmitir un segmento se hace uno nuevo que incluya los nuevos datos a mandar que han aparecido desde que se envió el segmento original (si caben).

Fiabilidad en TCP

- TCP incorpora otro mecanismo para detectar la pérdida de paquetes que es más rápido en la detección: la presencia de 3 ACKs duplicados.



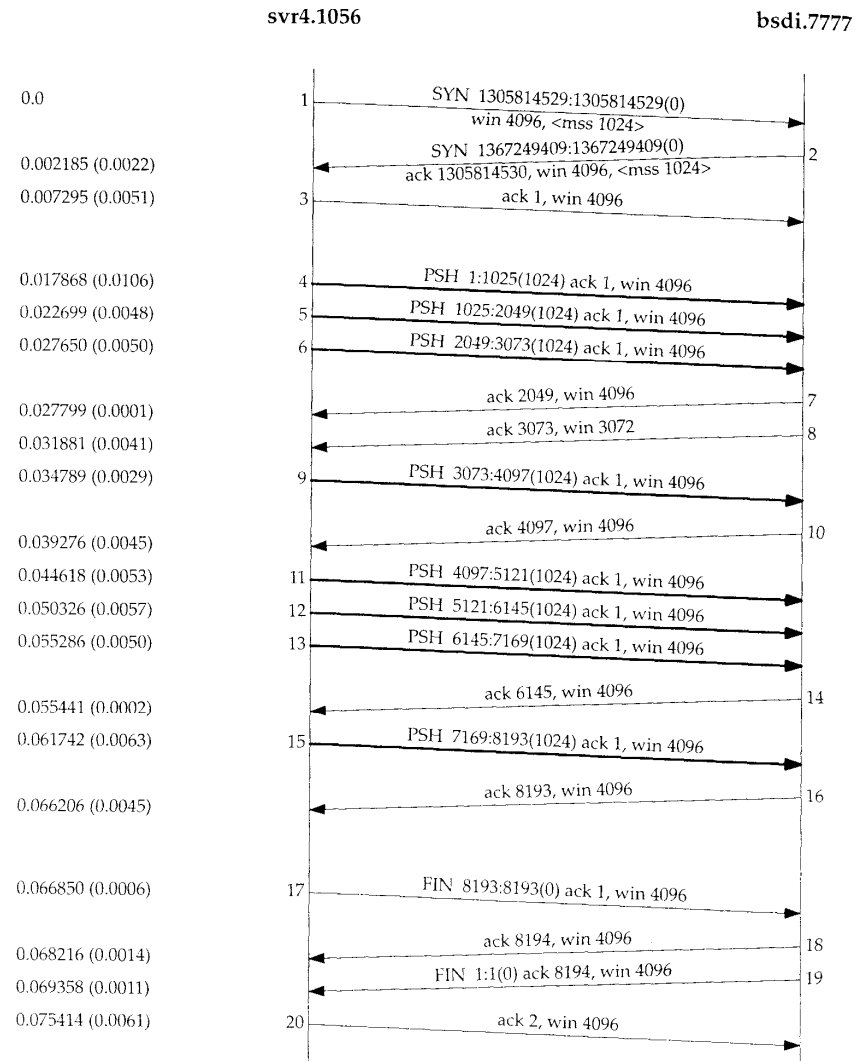
- 2 o menos ACKs duplicados pueden deberse a un retraso del paquete 3 (no su pérdida).
- Se considera pérdida a partir del 3er ACK duplicado.

4.7 Transferencia masiva

4.7.1 Transferencia normal

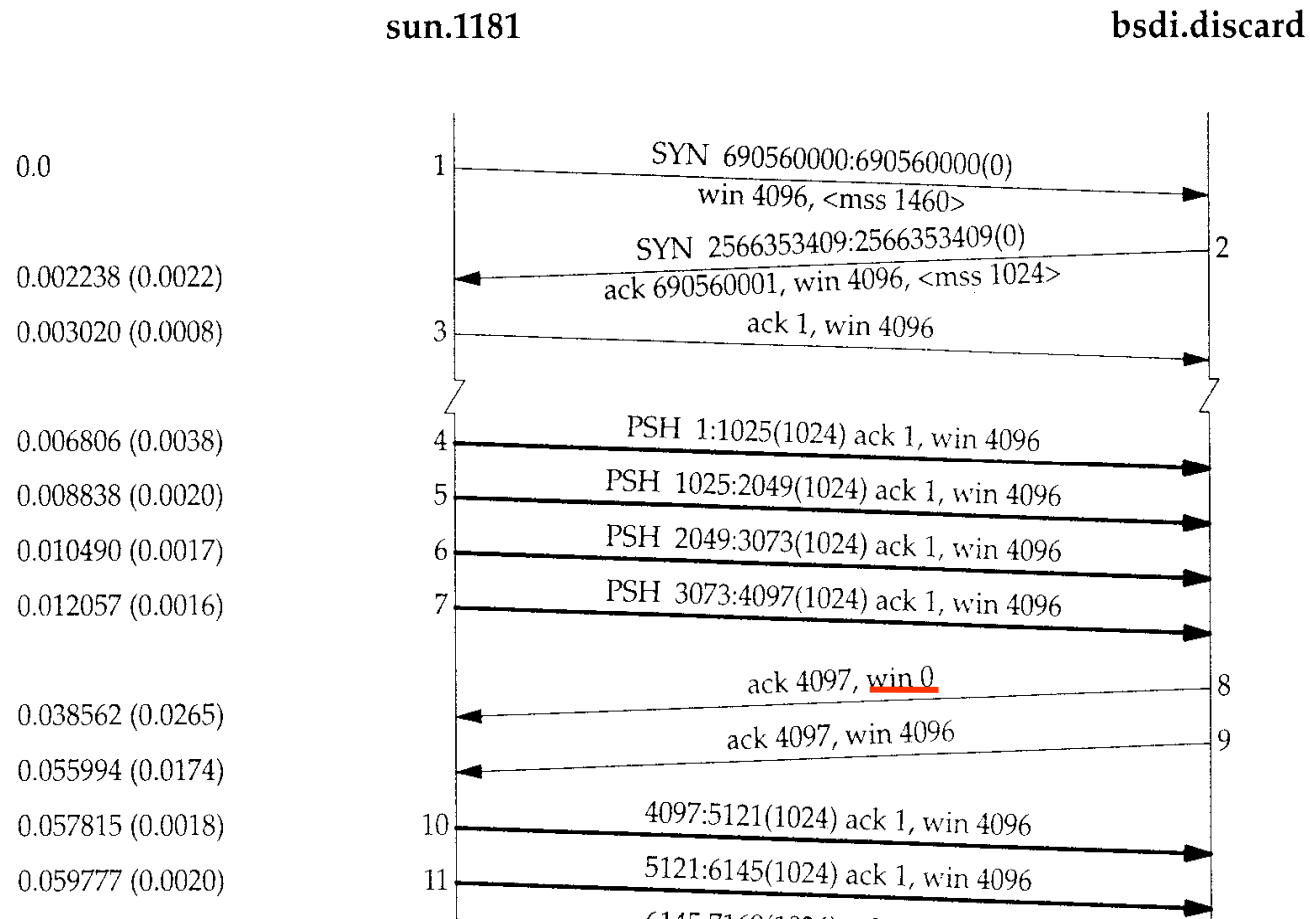
- El emisor envía más despacio que el ancho de banda de la red y que el límite de velocidad de lectura que tenga el receptor.

- SYN ISN
- win
- mss
- ack
- X:Y (Z)
- PSH
- FIN



4.7.2 Control de flujo

- Evitar saturar al receptor: caso emisor rápido/receptor lento



Control de flujo

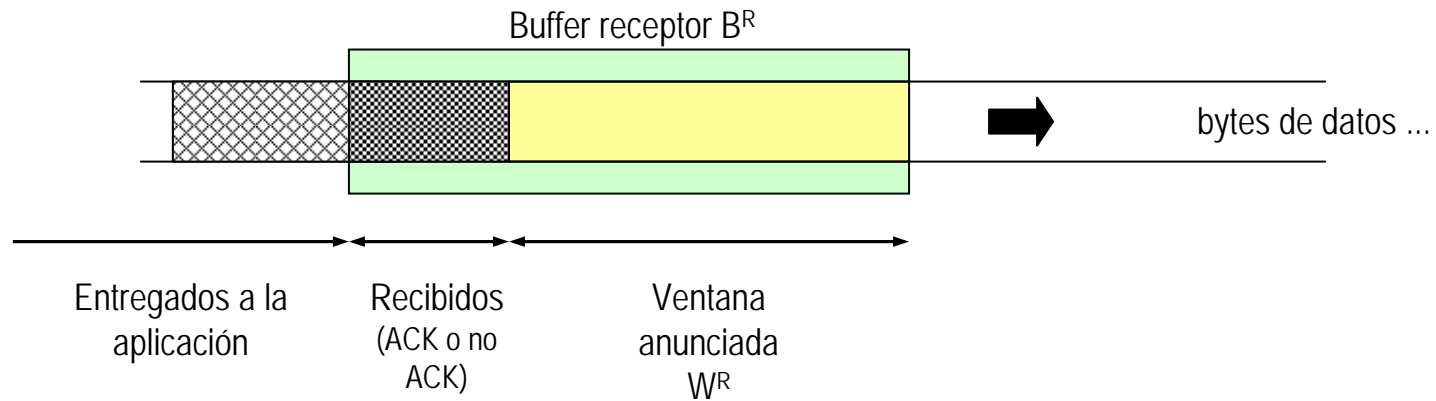
- La ventana anunciada por el receptor llegará a 0 con lo que el emisor no puede mandarle nada más.
- Mandará actualización de ventana (en la figura anterior segmento 9) sin confirmar datos cuando la ventana crezca a 2 MSS o al 50% de la original.
- Control de flujo: se implementa mediante el mecanismo de ventana deslizante.

4.7.2.1 Ventana deslizante

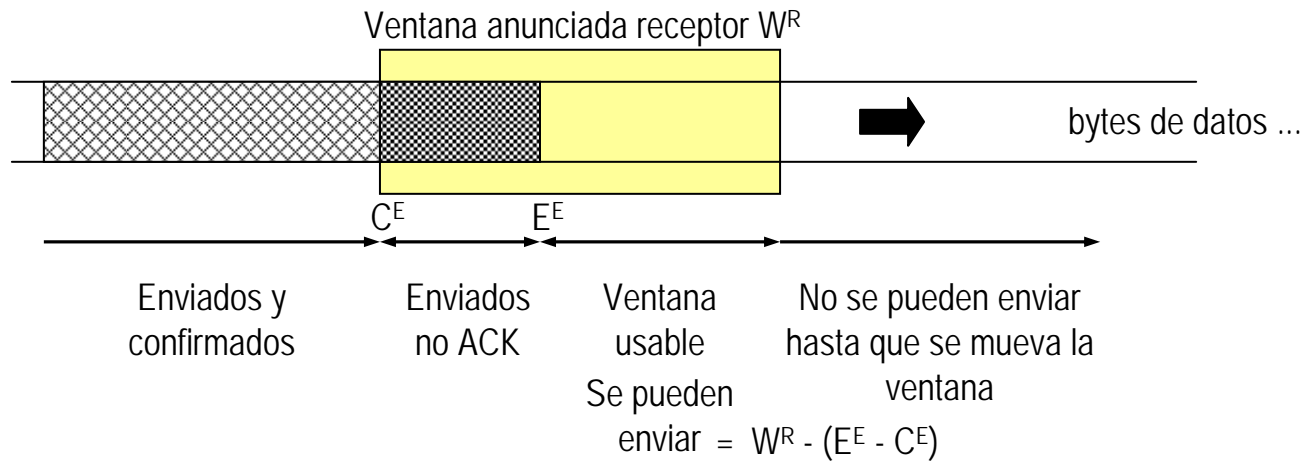
- Control de flujo para no saturar al receptor y nunca transmitir más que el buffer que tenga disponible.
- Permite al transmisor enviar varios paquetes antes de parar y esperar el ACK.
- En TCP los ACKs son acumulativos: confirman la recepción correcta de hasta el n^0 de bytes indicado por el n^0 de confirmación menos uno.
- Parámetros que intervienen:
 - B^R : buffer del receptor.
 - Tamaños típicos de buffer B^R : 8, 32, 64 Kbytes
 - W^R : ventana anunciada por el receptor (win, campo cabecera TCP).

Ventana deslizante

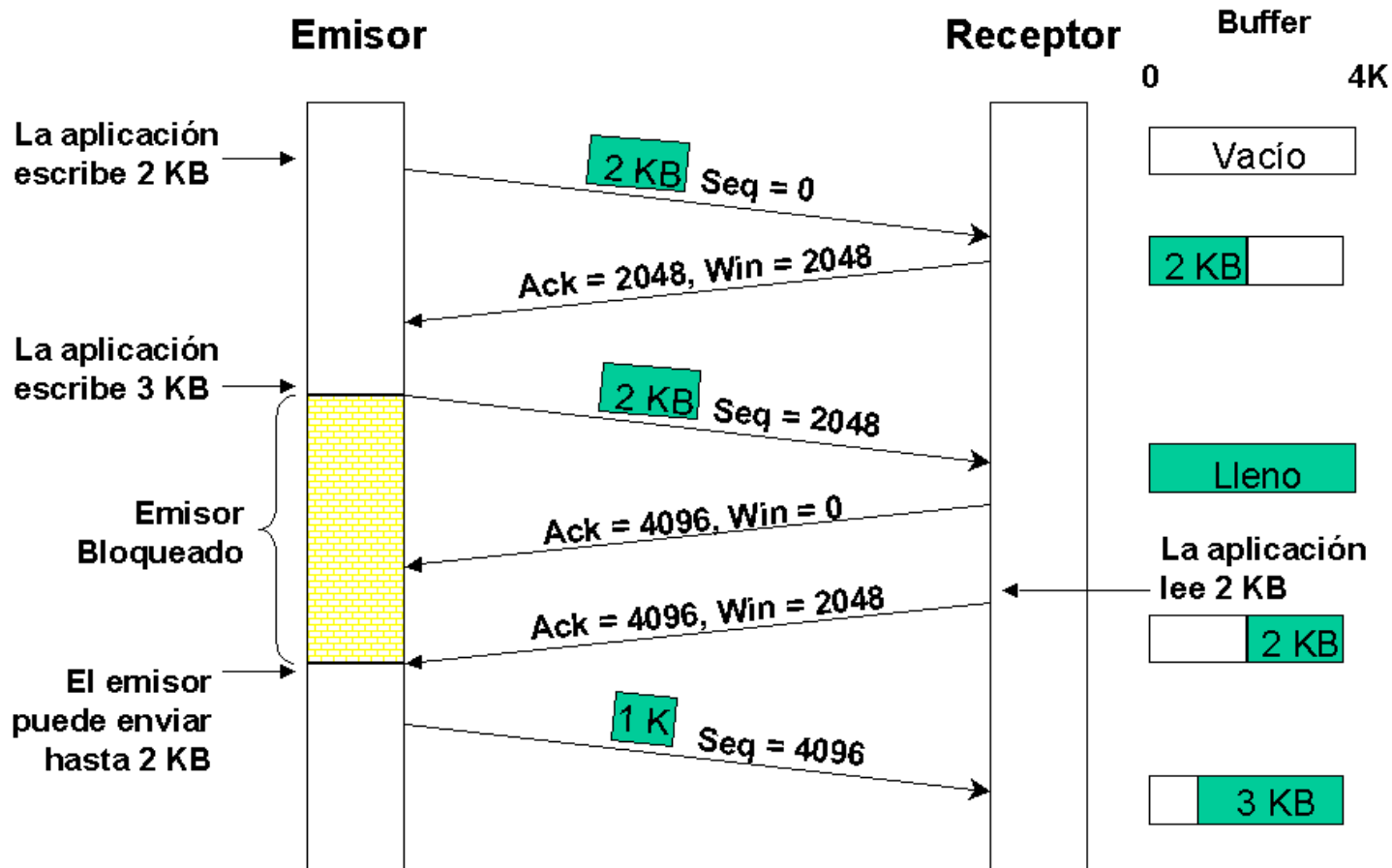
■ En receptor:



■ En emisor:

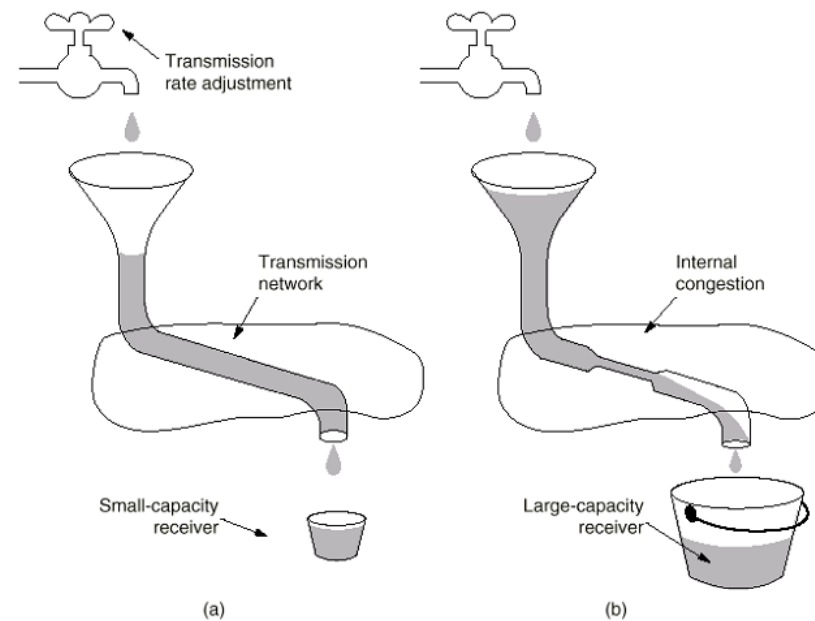


Ventana deslizante



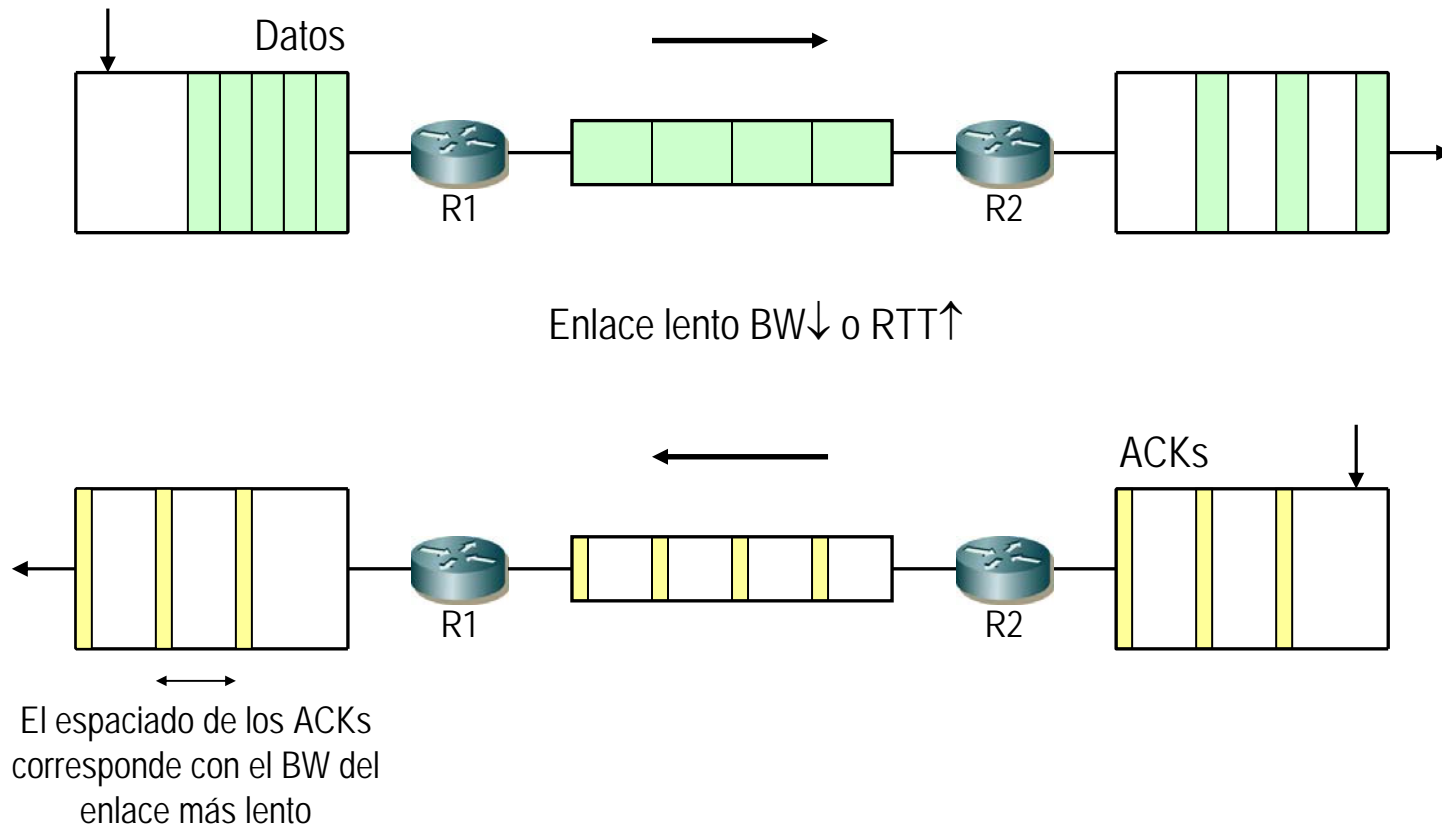
4.7.3 Control de congestión

- Una red está congestionada cuando las colas de los routers se llenan debido a la lentitud de los enlaces de salida.
- Ocurre si un emisor rápido alimenta un enlace lento o varios enlaces rápidos desembocan en uno lento.



Control de congestión

- ¿Cómo podemos estimar la velocidad de la red?
 - Según la velocidad de llegada de ACKs



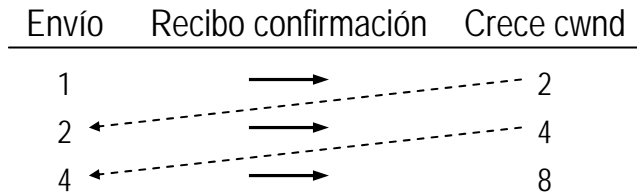
Control de congestión

- El emisor envía segmentos al receptor hasta llenar la ventana anunciada por este.
 - En LANs no tiene por qué suponer ningún problema.
 - En enlaces más lentos puede dar problemas: se pueden encolar segmentos en routers con enlaces lentos, pudiendo llenarse los buffers y tirar los paquetes.
 - Pérdida de paquetes → Congestión de red
- Control de congestión: se implementa mediante la ventana de congestión (cwnd).
 - Es un parámetro interno de cada extremo TCP.
- Junto con la ventana anunciada por el receptor (control de flujo) imponen el tamaño final de envío:
 - ventana de envío = $\min(\text{cwnd}, W^R)$
- La ventana de congestión se adapta a las condiciones de la red en 2 fases:
 - Slow Start
 - Congestion Avoidance

4.7.3.1 Slow start

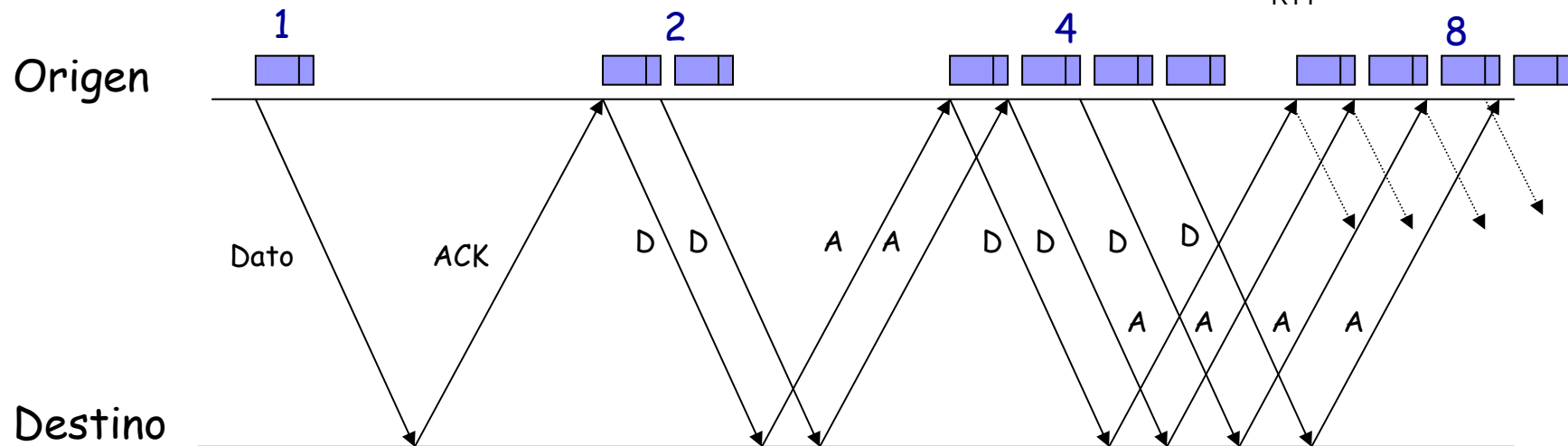
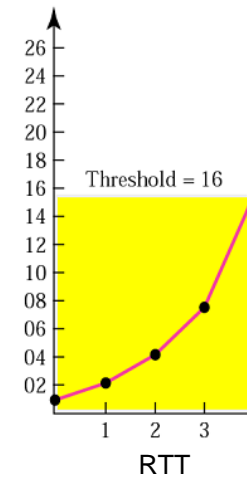
- Al mandar los primeros paquetes de una conexión TCP desconocemos las limitaciones de velocidad de la red.
- Se comienza enviando “despacio” para ir incrementando la velocidad de manera exponencial.
- *Slow start*, inyectar paquetes a la red a la velocidad a la que recibe ACKs retornados por el otro extremo.
- Pasos:
 1. Al iniciar la conexión: $cwnd = 1 \text{ MSS}$ (depende de la implementación)
 - El emisor podrá enviar hasta ventana = $\min(cwnd, W^R)$ bytes.
 2. Al confirmarse cada segmento enviado: $cwnd = cwnd + 1 \text{ MSS}$
 - Crecimiento exponencial.
 - Por cada ACK real recibido (no por ACK duplicados).

Slow start



cwnd: impuesto por el emisor
 WR: impuesto por el receptor

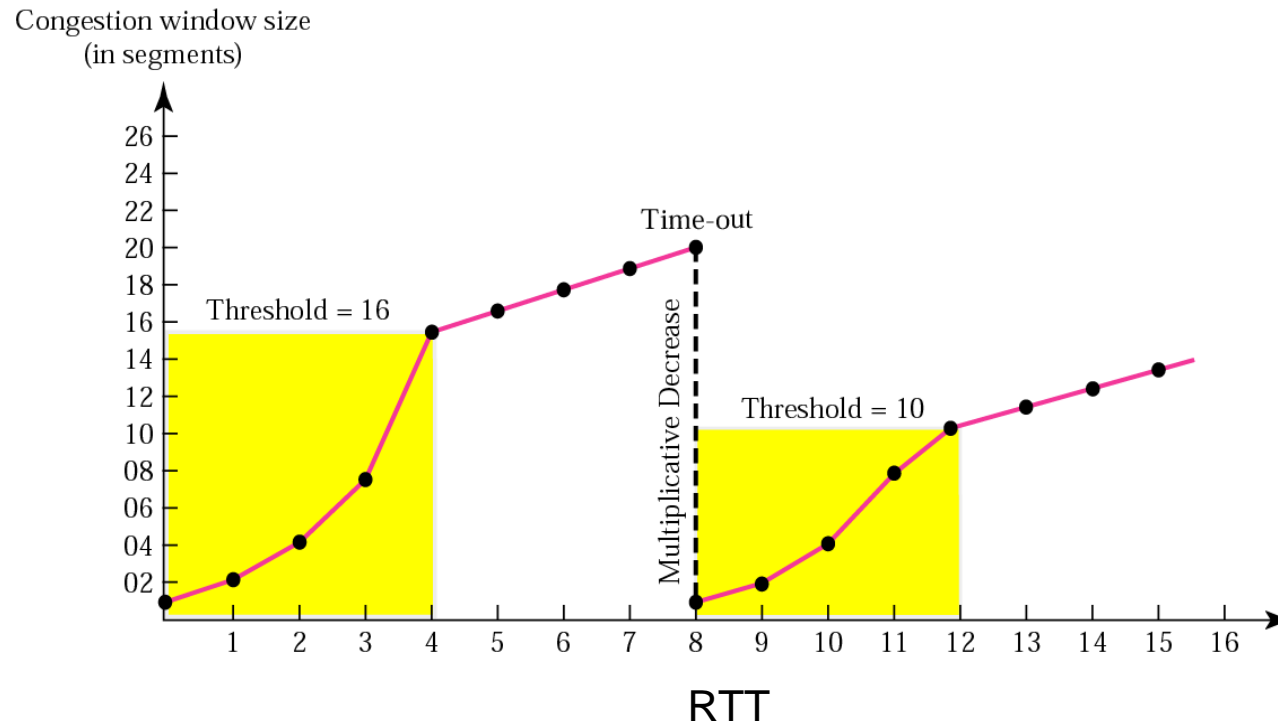
Congestion window size
 (in segments)



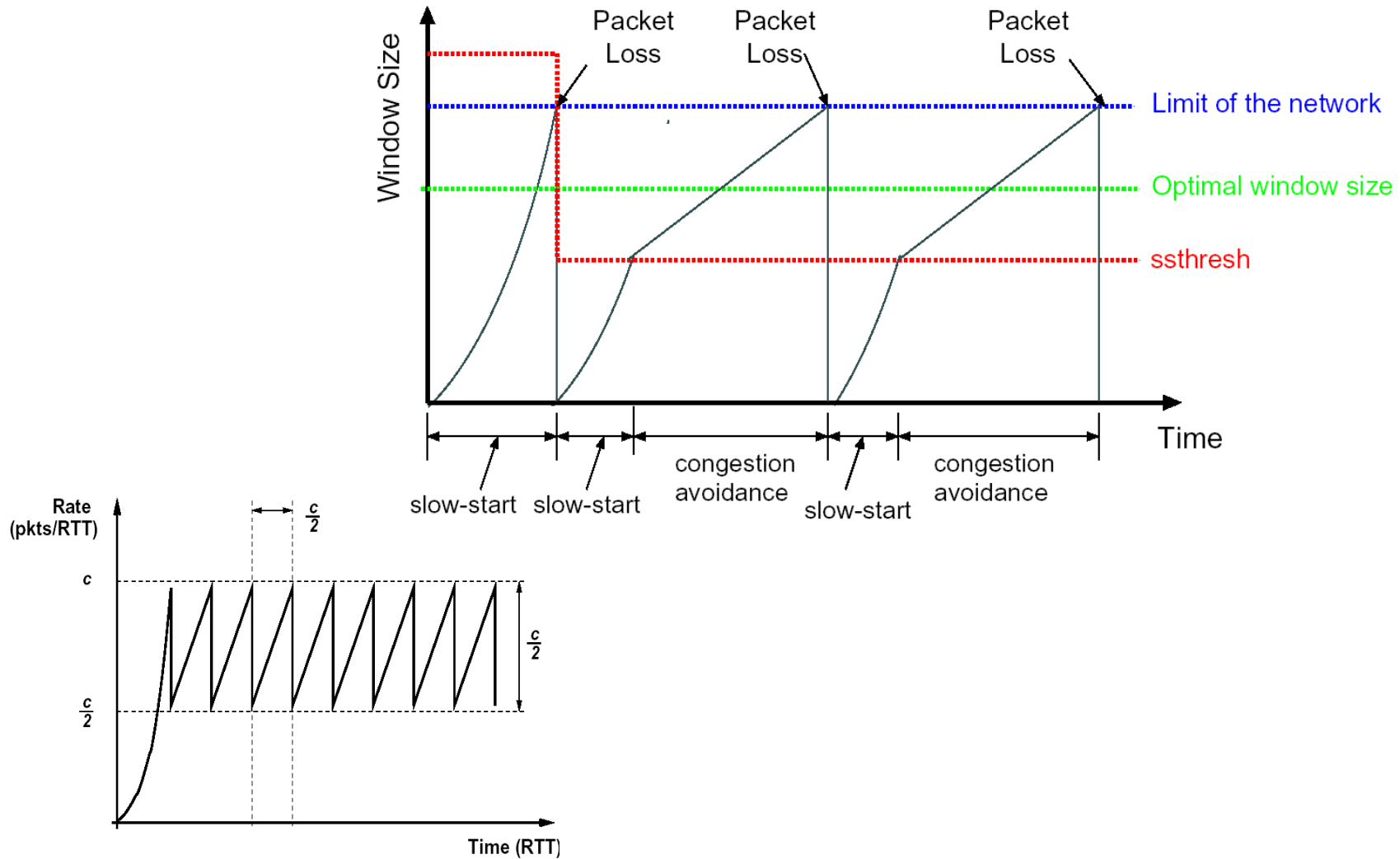
4.7.3.2 Congestion avoidance

- Si una ventana empieza a crecer con slow start llegará un momento en que congestione la red y se empiece a tirar paquetes.
 - Normalmente se asume que la pérdida de paquetes debido a errores físicos (fallos CRC) es un % muy pequeño en las redes actuales y despreciable.
 - Por tanto se puede tomar la pérdida de paquetes como indicador de congestión: se interpreta como que algún router los ha tirado porque ha llenado sus buffers.
- Se define el ssthresh, umbral del tamaño de ventana de congestión en el que finaliza el periodo slow start y comienza el periodo congestion avoidance. Su valor depende de la implementación (8-64 Kbytes)
- En este periodo la ventana de congestión crece más despacio: 1 MSS por RTT como máximo.
- En caso de pérdida detectada por RTO
 - cwnd baja a 1MSS y se continúa en slow start.
 - $ssthresh = \text{ventana_anterior} / 2 = \min(\text{cwnd}, W^R) / 2$

Congestion avoidance

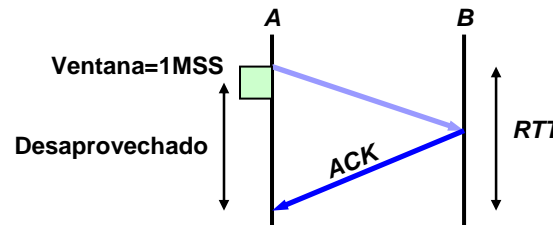


Ajuste a la capacidad del canal



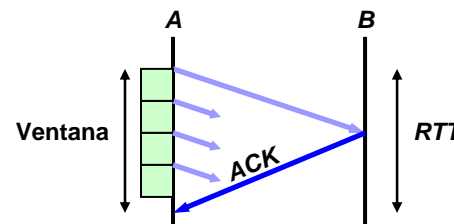
4.8 Producto RTTxBW

- Para maximizar la velocidad de transferencia TCP debemos analizar el tamaño de ventana a anunciar por el receptor (W^R).
- Si $W^R=1MSS$



- Mejor caso: Ventana = RTT x BW
 - (bits) = (s) x (bps)
 - ventana= min(cwnd, W^R)
 - W^R no es controlable, pero sí el buffer de recepción B^R .
 - Si la aplicación receptora lee rápido: $W^R \approx B^R$

Ej: Ethernet 10Mbps, RTT 10ms
 ventana = 12,5 Kbytes



4.9 Ejemplo de traza TCP

■ Captura en el extremo 130.206.158.141

```

18:35:35.522405 IP 130.206.158.141.55908 > 130.206.13.20.http: S 2664700947:2664700947(0) win 5840 <mss 1460,sackOK,timestamp 2618480914
0,nop,wscale 7>
18:35:35.547232 IP 130.206.13.20.http > 130.206.158.141.55908: S 3076916275:3076916275(0) ack 2664700948 win 4380 <mss 1460,nop,wscale
0,nop,nop,timestamp 393065200 2618480914,sackOK,eol>
18:35:35.547259 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 1 win 46 <nop,nop,timestamp 2618480938 393065200>
18:35:35.547410 IP 130.206.158.141.55908 > 130.206.13.20.http: P 1:120(119) ack 1 win 46 <nop,nop,timestamp 2618480939 393065200>
18:35:35.577493 IP 130.206.13.20.http > 130.206.158.141.55908: P 2897:4345(1448) ack 120 win 4499 <nop,nop,timestamp 393065229 2618480939>
18:35:35.577521 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 1 win 46 <nop,nop,timestamp 2618480969 393065200,nop,nop,sack 1
{2897:4345}>
18:35:35.577499 IP 130.206.13.20.http > 130.206.158.141.55908: P 1449:2897(1448) ack 120 win 4499 <nop,nop,timestamp 393065229 2618480939>
18:35:35.577535 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 1 win 46 <nop,nop,timestamp 2618480969 393065200,nop,nop,sack 1
{1449:4345}>
18:35:35.577554 IP 130.206.13.20.http > 130.206.158.141.55908: P 1:1449(1448) ack 120 win 4499 <nop,nop,timestamp 393065229 2618480939>
18:35:35.577572 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 4345 win 69 <nop,nop,timestamp 2618480969 393065229>
18:35:35.601650 IP 130.206.13.20.http > 130.206.158.141.55908: P 4345:5793(1448) ack 120 win 4499 <nop,nop,timestamp 393065254 2618480969>
18:35:35.601685 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 5793 win 91 <nop,nop,timestamp 2618480993 393065254>
18:35:35.601657 IP 130.206.13.20.http > 130.206.158.141.55908: . 7241:8689(1448) ack 120 win 4499 <nop,nop,timestamp 393065254 2618480969>
18:35:35.601697 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 5793 win 91 <nop,nop,timestamp 2618480993 393065254,nop,nop,sack 1
{7241:8689}>
18:35:35.601662 IP 130.206.13.20.http > 130.206.158.141.55908: . 5793:7241(1448) ack 120 win 4499 <nop,nop,timestamp 393065254 2618480969>
18:35:35.601706 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 8689 win 114 <nop,nop,timestamp 2618480993 393065254>
18:35:35.601668 IP 130.206.13.20.http > 130.206.158.141.55908: . 8689:10137(1448) ack 120 win 4499 <nop,nop,timestamp 393065254 2618480969>
18:35:35.601713 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 10137 win 137 <nop,nop,timestamp 2618480993 393065254>
18:35:35.625774 IP 130.206.13.20.http > 130.206.158.141.55908: . 10137:11585(1448) ack 120 win 4499 <nop,nop,timestamp 393065279 2618480993>
18:35:35.625803 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 11585 win 159 <nop,nop,timestamp 2618481017 393065279>
18:35:35.625781 IP 130.206.13.20.http > 130.206.158.141.55908: P 11585:12266(681) ack 120 win 4499 <nop,nop,timestamp 393065279 2618480993>
18:35:35.625814 IP 130.206.158.141.55908 > 130.206.13.20.http: . ack 12266 win 182 <nop,nop,timestamp 2618481017 393065279>
18:35:35.625786 IP 130.206.13.20.http > 130.206.158.141.55908: F 12266:12266(0) ack 120 win 4499 <nop,nop,timestamp 393065279 2618480993>
18:35:35.625991 IP 130.206.158.141.55908 > 130.206.13.20.http: F 120:120(0) ack 12267 win 182 <nop,nop,timestamp 2618481017 393065279>
18:35:35.650011 IP 130.206.13.20.http > 130.206.158.141.55908: . ack 121 win 4499 <nop,nop,timestamp 393065303 2618481017>

```

Resumen

- Las pérdidas de paquetes se asocian a presencia de congestión de la red. Se identifican por:
 - Vencimiento de RTO
 - Presencia de ACKs duplicados
- Control de flujo: ventana deslizante
- Control de flujo + control de congestión:
 - $\text{ventana} = \min(\text{cwnd}, W^R)$
- Si se quiere que la conexión esté limitada por el ancho de banda de la red y no por el buffer del receptor:
 - $B^R = \text{RTT} \times \text{BW}$

Referencias

- [Forouzan]
 - Capítulo 15, “Transmission Control Protocol (TCP)”, secciones 15.6-15.10
- [Stevens]
 - Capítulo 20, “TCP Bulk Data Flow”
 - Capítulo 21, “TCP Timeout and Retransmission”