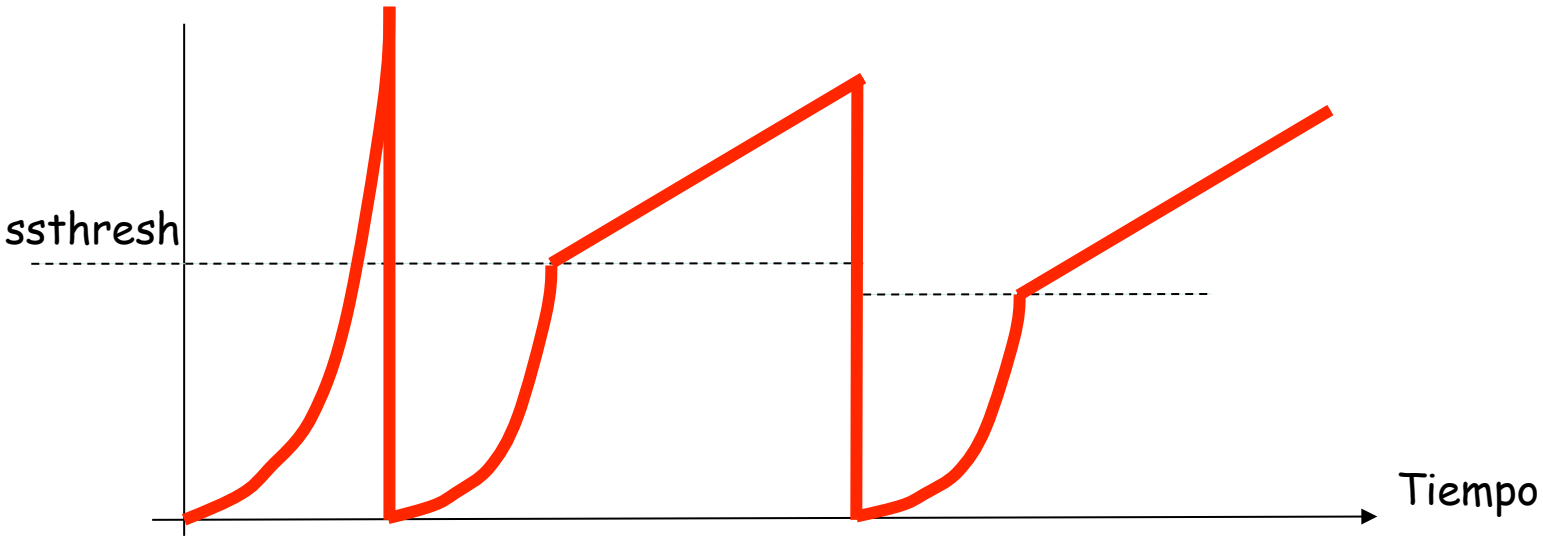
Congestión y rendimiento de TCP

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Grado en Ingeniería en Tecnologías de Telecomunicación, 4º

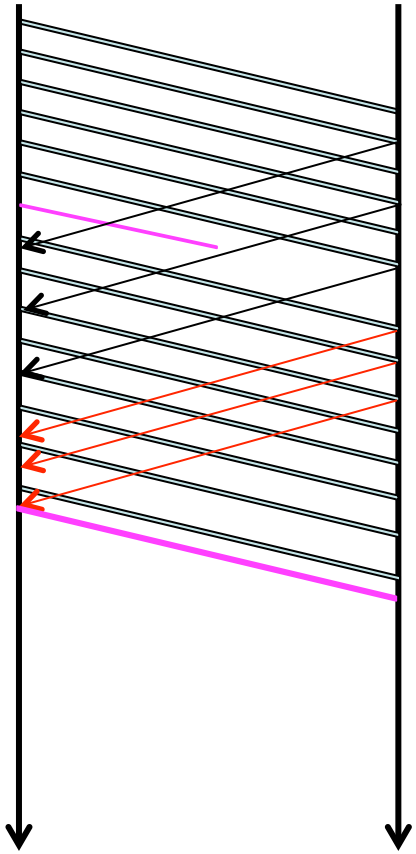
Hemos visto

- TCP Tahoe



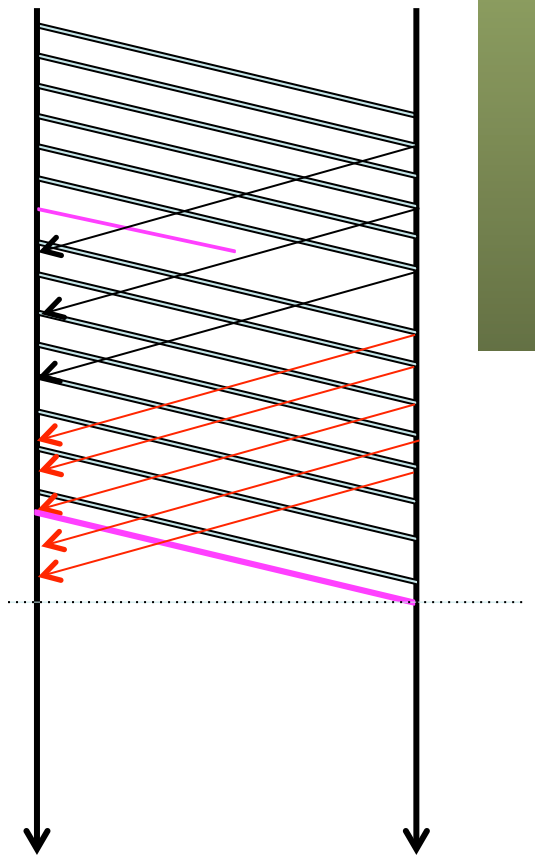
Fast retransmit

- Ante una pérdida, los paquetes siguientes que llegan generan ACKs
- Pero no confirman nuevos datos sino que repiten la confirmación anterior
- Se envía lo que se conoce como un “ack duplicado”
  - Hay datos enviados sin confirmar
  - El ACK no lleva datos, ni SYN ni FIN
  - Repite el número de ACK más grande recibido
  - Repite el tamaño de ventana anunciada del último ACK
- Cuando se completa el hueco debe enviar un nuevo ACK
- Ante 3 ACKs duplicados retransmite (*fast retransmit*)
- Entra en *fast recovery* hasta que dejen de llegar ACKs duplicados
- (...)



Fast recovery

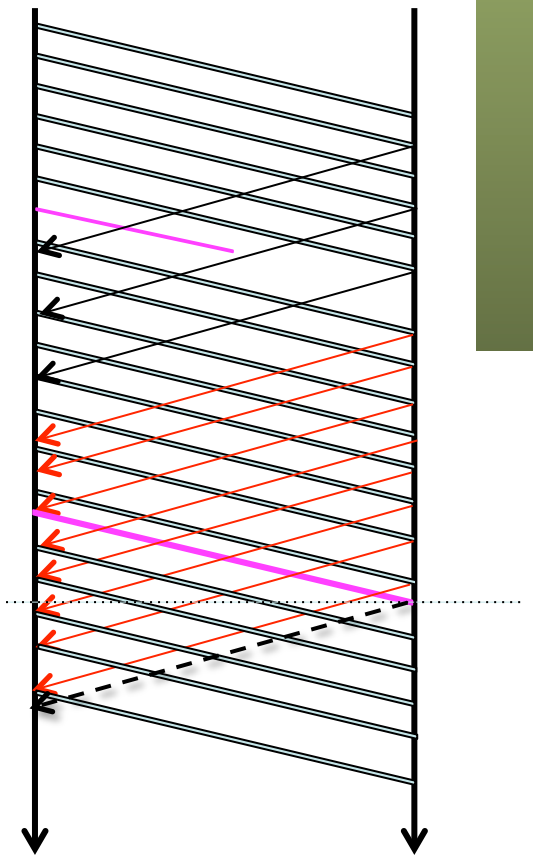
- Pueden seguir llegando ACKs duplicados según el *FlightSize*
- No pasa a slow start pues el que lleguen los duplicados implica que los paquetes están llegando al destino
- Seguimos teniendo el "reloj" que dan los ACKs, puede seguir transmitiendo
- Se reduce ssthresh a la mitad
- cwnd = ssthresh + 3xSMSS
- Esto añade a la ventana los 3 segmentos que han debido llegar (eso dicen los dup ACKs)
- Por cada nuevo ACK duplicado se incrementa cwnd += SMSS
- Esto permite un nuevo envío
- (...)



Fast recovery

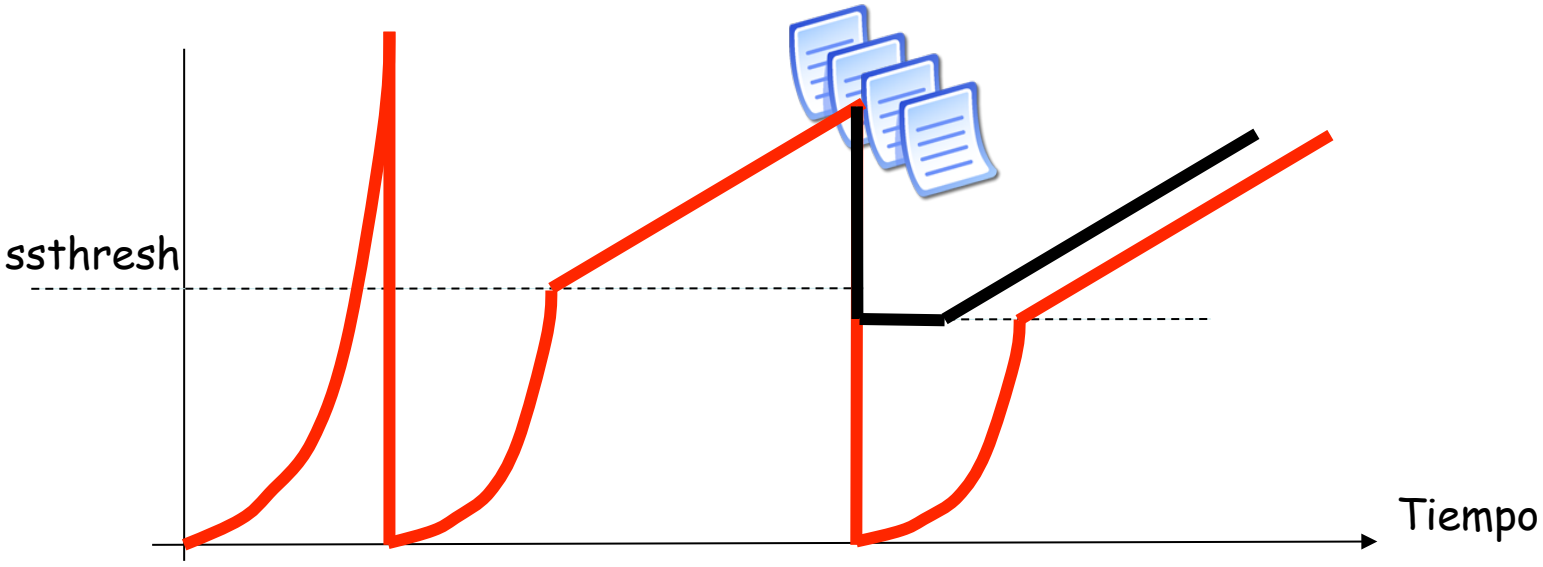
- Pueden seguir llegando ACKs duplicados según el *FlightSize*
- No pasa a slow start pues el que lleguen los duplicados implica que los paquetes están llegando al destino
- Seguimos teniendo el "reloj" que dan los ACKs, puede seguir transmitiendo
- Se reduce ssthresh a la mitad

  ssthresh = máx(FlightSize / 2, 2xSMSS)
- cwnd = ssthresh + 3xSMSS
- Esto añade a la ventana los 3 segmentos que han debido llegar (eso dicen los dup ACKs)
- Por cada nuevo ACK duplicado se incrementa cwnd += SMSS
- Esto puede permitir un nuevo envío
- Cuando llegue el segmento perdido se genera confirmación para todo lo recibido
- Finalmente llegará ACK para nuevos datos
- Pone cwnd = ssthresh (*deflating* la ventana)
- Debería confirmar todo lo recibido tras el hueco



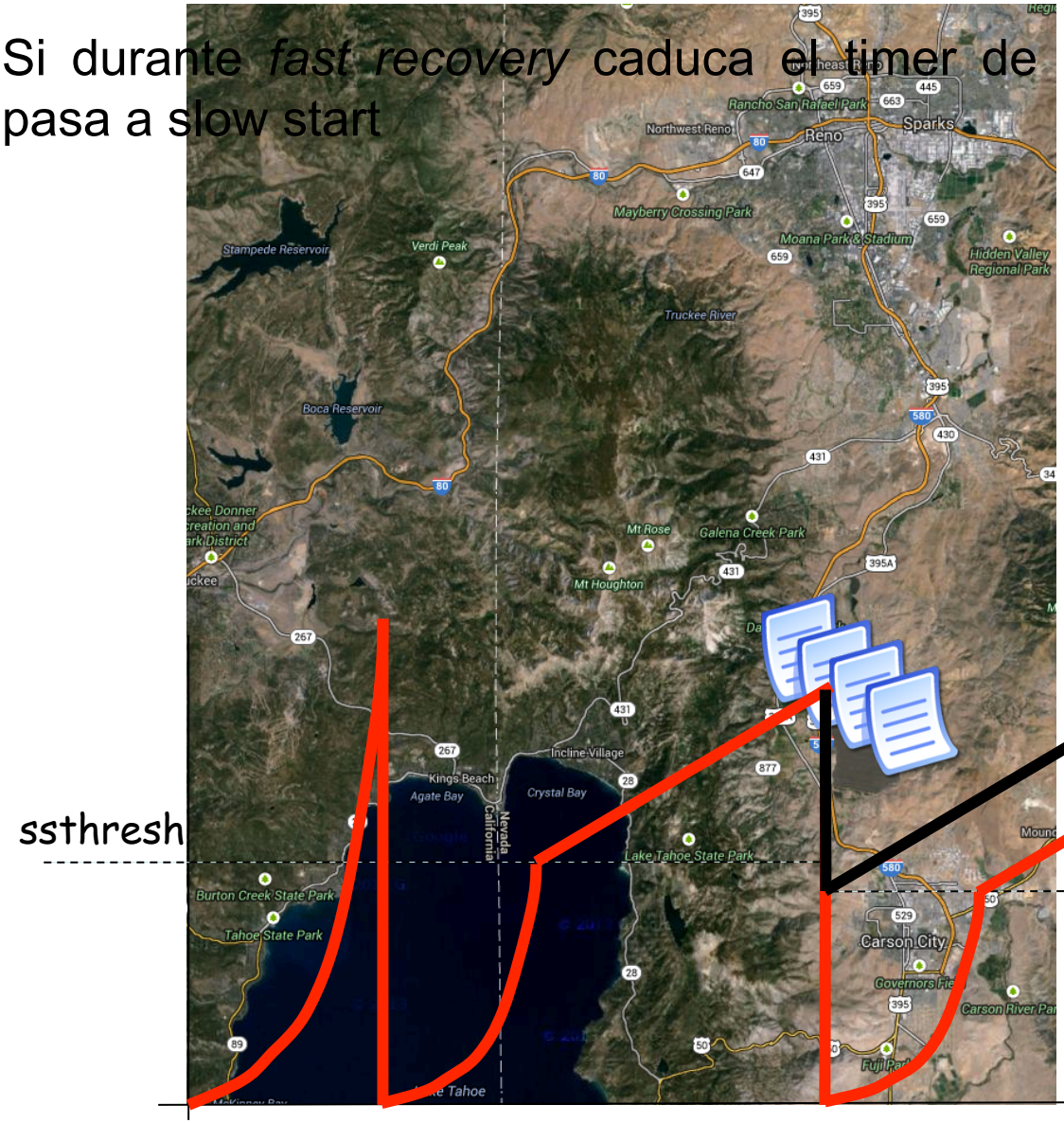
Fast retransmit/recovery

- Si durante *fast recovery* caduca el timer de retransmisión se pasa a slow start



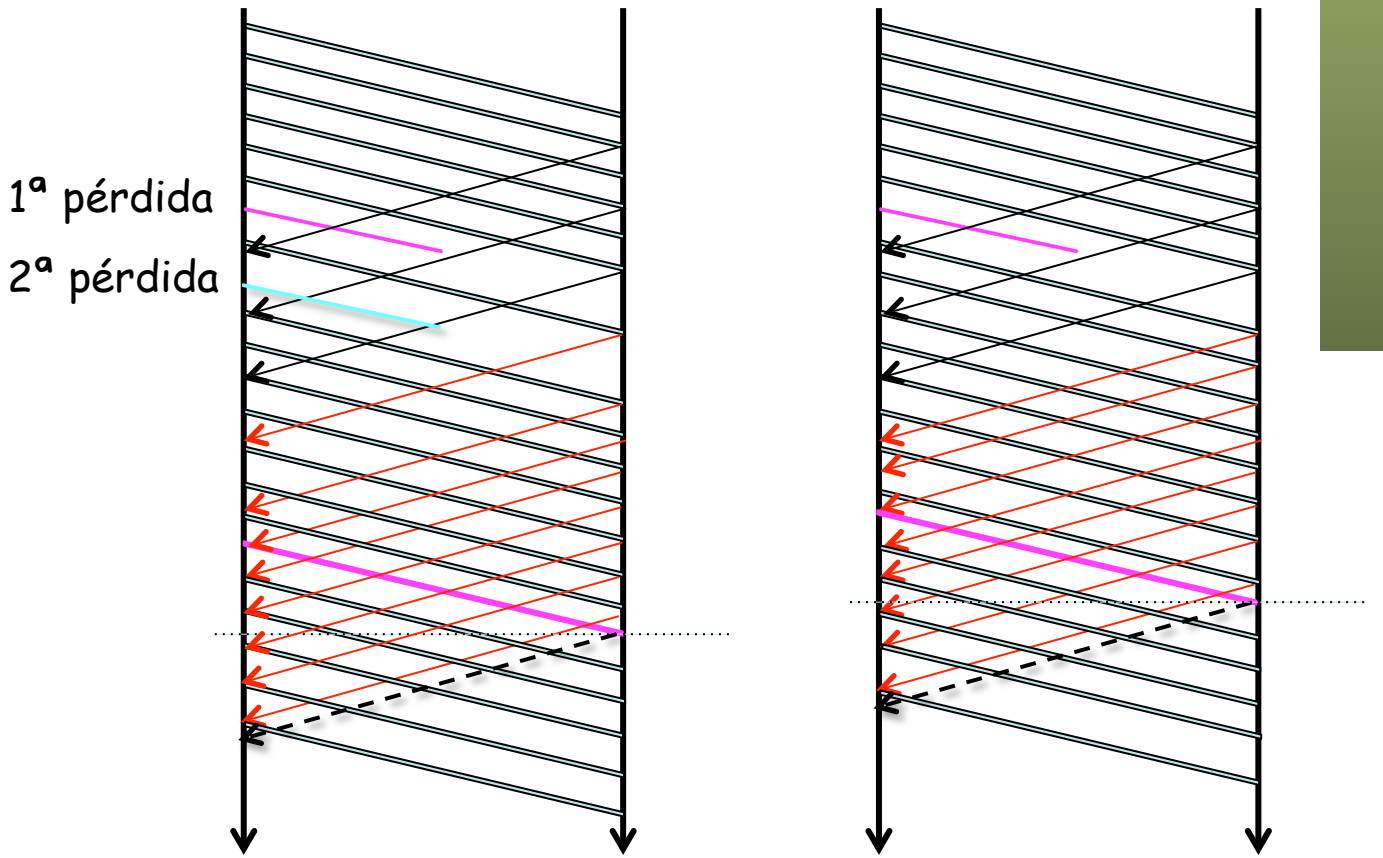
Fast retransmit/recovery

- Si durante *fast recovery* caduca el timer de retransmisión se pasa a slow start



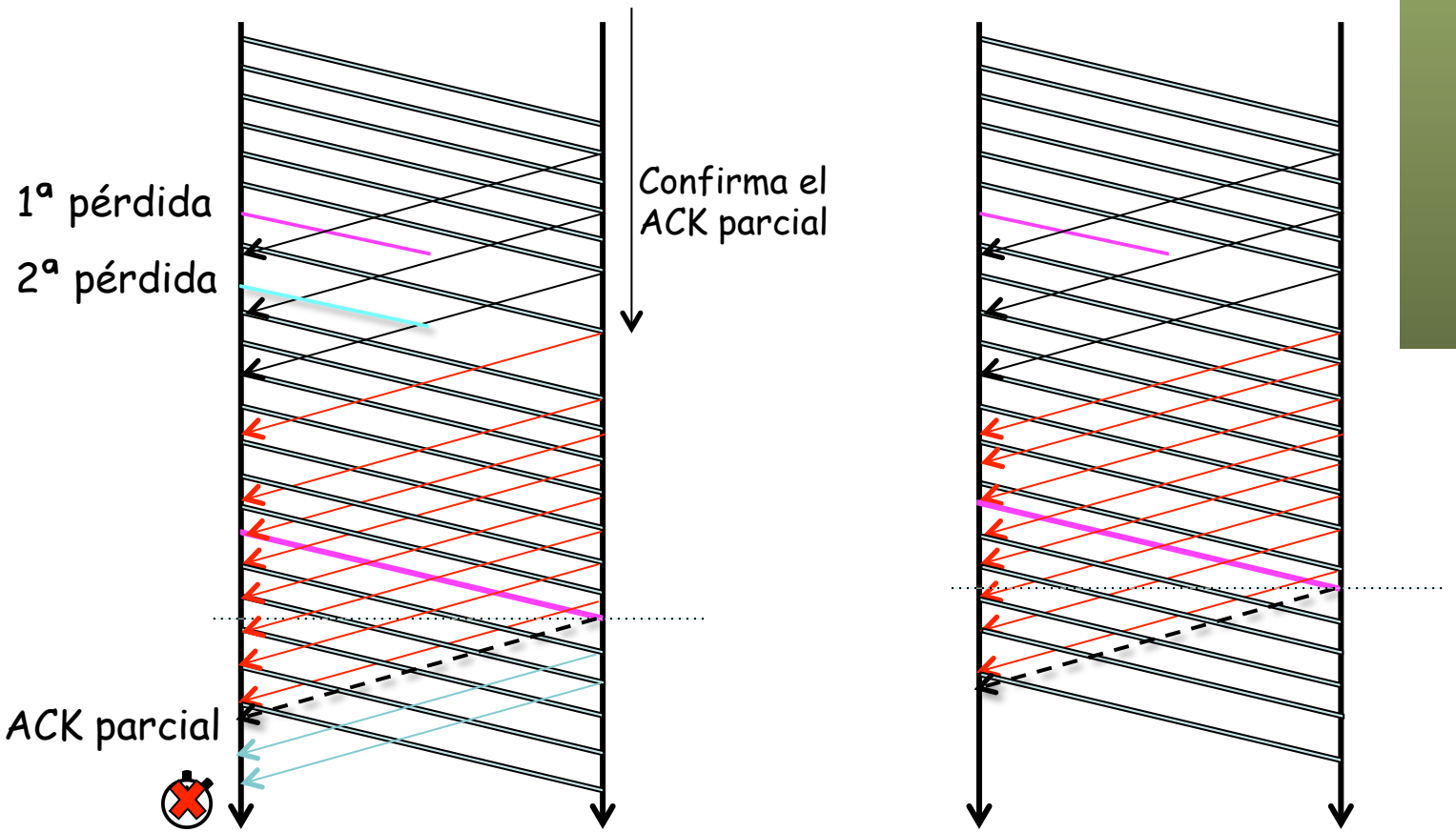
Pérdidas múltiples

- Fast retransmit/fast recovery no recupera tan bien pérdidas múltiples en el *FlightSize* dado que solo retransmite 1 segmento
- En este caso por ejemplo se pierden 2
- Ahora el tercer dup ACK es más tarde (falta el ACK del 2º perdido)
- (…)



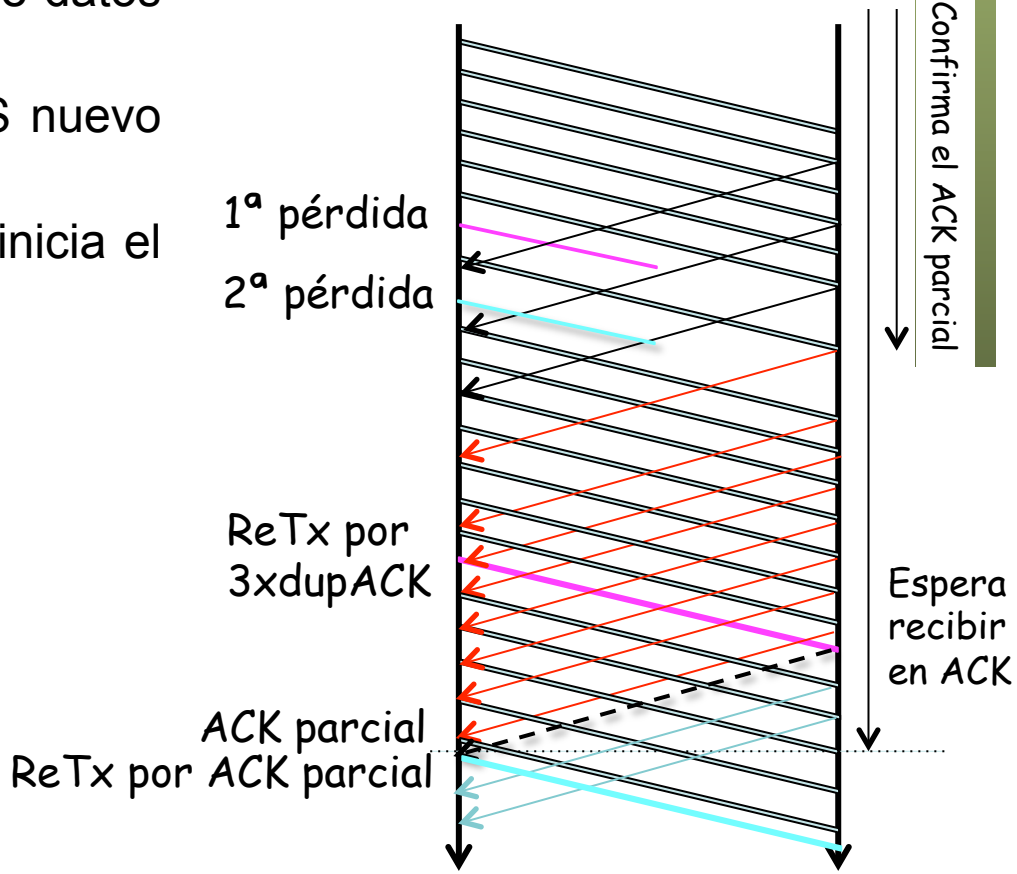
Pérdidas múltiples

- El ACK que llega al llenarse el hueco es un **ACK *parcial***
- La secuencia está interrumpida por la segunda pérdida
- Deberían llegar nuevos *dup* ACKs para reactivar *fast retransmit*
- Pero es improbable que haya tanto *FlightSize* y suele caducar el timer



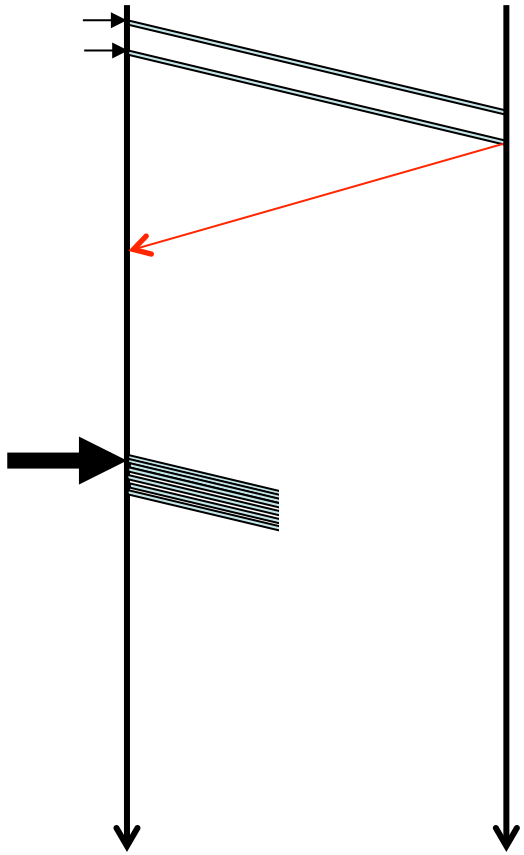
NewReno

- Modifica *Fast recovery* (RFC 6582)
- Con el 3º dup-ACK apunta número de secuencia máximo enviado (hasta ahí debería recibir confirmación)
- Al recibir un **ACK parcial** retransmite el primer segmento sin confirmar
- Reduce cwnd en la cantidad de datos confirmados
- Si confirma al menos 1 SMSS nuevo incrementa cwnd en 1 SMSS
- Si es el primer ACK parcial reinicia el timer de retransmisión
- Sigue en fast recovery



Restarting idle connections

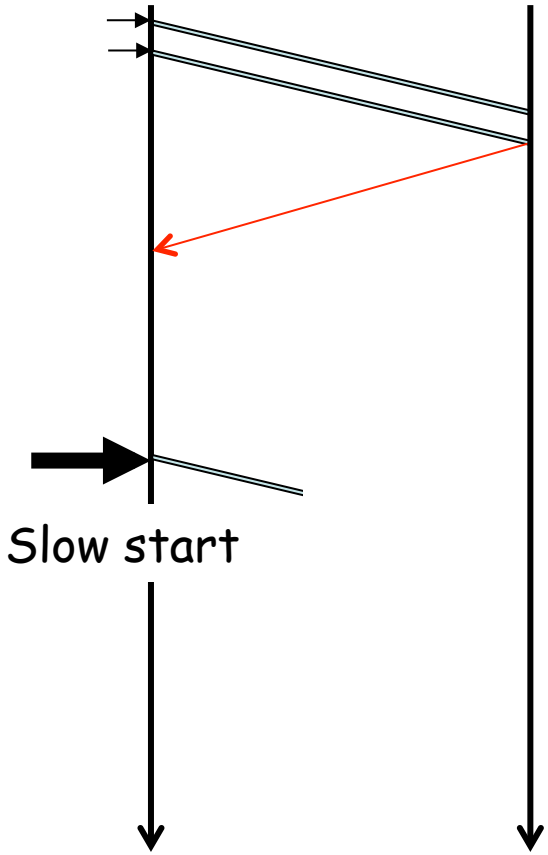
- ¿Qué sucede si el flujo queda inactivo un tiempo (no hay datos a enviar)?
- Ha perdido el reloj que dan los ACKs
- Puede tener una cwnd grande
- Si llega una gran cantidad de datos se pueden enviar en ráfaga
- Se supone que esos paquetes “caben” en la red (en buffers) pues lo dice cwnd
- Pero pueden haber cambiado las condiciones de la red
- Ahora cwnd puede ser una mala estimación
- Y de todos modos era una estimación de buffer para todo el trayecto
- Estamos soltando esos paquetes en ráfaga (*line rate*) al primer conmutador



Restarting idle connections

- La recomendación es volver a slow start tras un periodo largo de inactividad
- RFC 5681 sugiere hacerlo cuando TCP no ha recibido un segmento durante más de un RTO
- O cuando no ha tenido datos para transmitir durante al menos un RTO
- Se reduce cwnd a

  cwnd' = RW = mín(IW, cwnd)
- No es obligatorio implementarlo



Límites de la ventana

Escenario LAN

- Por ejemplo 6 conmutadores (2xAcceso + 2xAgregación + 2xCore)
- Retardo de conmutación en LAN (hardware) en unos 25 µs
- 6x25 = 150 µs de procesado
- Propagación < 5 µs (despreciable)
- Falta retardo de procesado en extremos
- Si todo es 10 Mbps
  - Paquete 1518bytes : 1.2x7 = 8.4 ms (tx)
  - ACK 64 bytes : 51.2x7 = 0.36 ms (tx)
  - RTT < 9.1 ms
  - Window = 10 Mbps x 9.1 ms = 11 KiB
- Si todo es 100 Mbps
  - Paquete 1518 bytes : 0.84 ms
  - ACK 64 bytes : 0.036 ms
  - Propagación + procesado = 0.31 ms
  - RTT < 1.2 ms (contando procesado)
  - Window = 100 Mbps x 1.2 ms = 15 KiB
  - Velocidad x10 ➔ retardos aprox. x0.1

Tamaño paquete	Velocidad transmisión	Retardo transmisión
64 Bytes	512 Kbps	1 ms
	2 Mbps	256 µs
	10 Mbps	51.2 µs
	100 Mbps	5.1 µs
	1 Gbps	0.51 µs
	10 Gbps	0.051 µs
1518 Bytes	512 Kbps	23.72 ms
	2 Mbps	6.1 ms
	10 Mbps	1.21 ms
	100 Mbps	121 µs
	1 Gbps	12 µs
	10 Gbps	1.2 µs

Límites de la ventana

Escenario LAN

- Por ejemplo 6 conmutadores (2xAcceso + 2xAgregación + 2xCore)
- Retardo de conmutación en LAN (hardware) en unos 25 µs
- 6x25 = 150 µs de procesado
- Propagación < 5 µs (despreciable)
- Falta retardo de procesado en extremos
- Si todo es **1 Gbps**
  - Paquete 1518 bytes : 12x7 = 85 µs (tx)
  - ACK 64 bytes : 0.51x7 = 3.6 µs (tx)
  - Propagación + procesado = 310 µs
  - RTT 400 µs
  - Procesado es ya muy apreciable y muy dependiente de los equipos
  - Window = 1 Gbps x 400 µs = 49 KiB
  - Esos retardos de procesado podrían incrementarse fácilmente
- Operaciones con enlaces desocupados

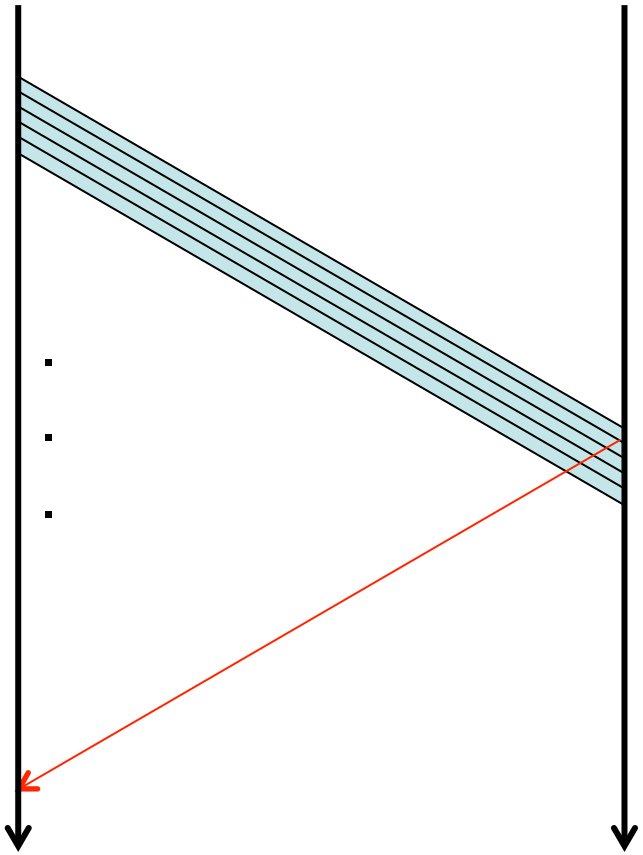
Tamaño paquete	Velocidad transmisión	Retardo transmisión
64 Bytes	512 Kbps	1 ms
	2 Mbps	256 µs
	10 Mbps	51.2 µs
	100 Mbps	5.1 µs
	1 Gbps	0.51 µs
	10 Gbps	0.051 µs
1518 Bytes	512 Kbps	23.72 ms
	2 Mbps	6.1 ms
	10 Mbps	1.21 ms
	100 Mbps	121 µs
	1 Gbps	12 µs
	10 Gbps	1.2 µs

Límites de la ventana

Escenario WAN

- América – Europa
- Retardo transatlántico : 70-90 ms (según cable)
- Retardo continental : 30 ms (según destino)
- Altas velocidads (si acceso no limita)
- Tiempos de transmisión despreciables
- RTT en torno a 200-300 ms
- Ventana máxima TCP de 64 KiB
- 65535 x 8 / 0.2 = 2.6 Mbps máximo
- Para alcanzar 10 Mbps con esa ventana el RTT debe ser menos de:

  65535 x 8 / 10^7 = 52.4 ms
- Para alcanzar 1 Gbps, < 0.52 ms
- Para mayores retardos necesitamos mayor ventana

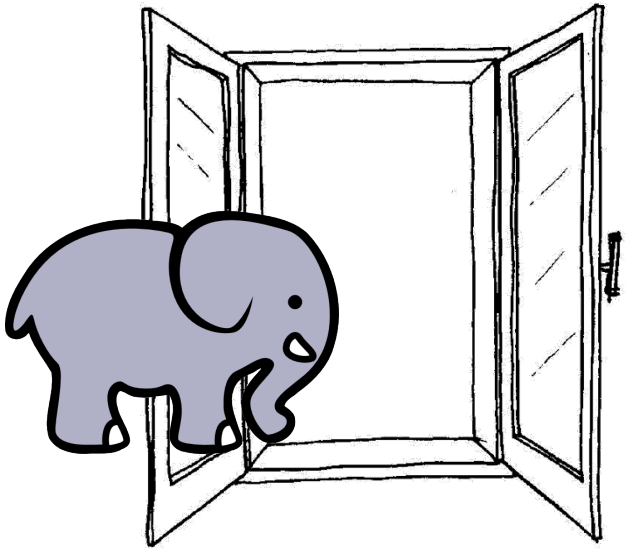


"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Ejemplos de LFP:
  - E3 por satélite BWxRTT = 2 MBytes o más de 1.300 paquetes grandes
  - STM-16 por fibra transatlántica BWxRTT = 19MBytes o más de 12.000 pkts (vale, sí, eso es para que una sola conexión saque provecho a 622 Mbps)
- (…)

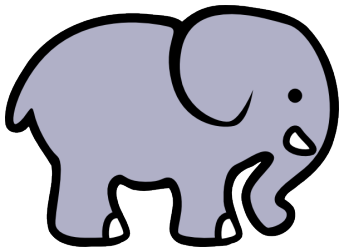
"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- **Tamaño de la ventana de control de flujo**
  - Es de 16 bits así que llega a 65.535 bytes
  - Se define una opción en la cabecera TCP: "Window Scale"
  - Se envía solo en el SYN y fija la escala
  - Deben enviar la opción ambos para que tenga efecto (puede ser diferente)
  - Solo se permiten potencias de 2 y van en escala logarítmica
  - Es decir, el valor anunciado de ventana se desplaza hacia la derecha en binario (al transmitir) tantas posiciones como dice la escala
  - La máxima escala es 14
  - Eso permite ventana de 2^{30} bytes
  - Eso es 1 GByte de *FlightSize*
  - Con un RTT de 200ms daría para 42 Gbps



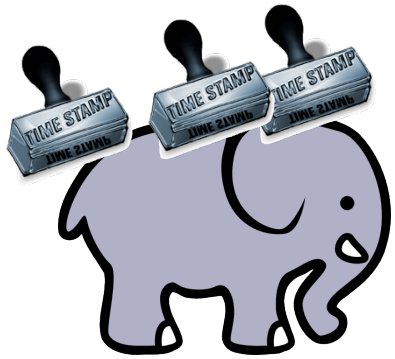
"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Tamaño de la ventana de control de flujo
- **Mejores medidas de RTT**
  - Es fundamental para un buen ajuste del RTO
  - Las técnicas habituales dan 1 medida del RTT por cada ventana
  - Con ventanas grandes es un muestreo muy escaso
  - (...)



"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Tamaño de la ventana de control de flujo
- **Mejores medidas de RTT**
  - Es fundamental para un buen ajuste del RTO
  - Las técnicas habituales dan 1 medida del RTT por cada ventana
  - Con ventanas grandes es un muestreo muy escaso
  - Se define una nueva opción de **Timestamp**
  - Datos llevan timestamp y ACK lo repite
  - Ambos deben soportalo (negociado en SYNs)
  - En realidad todos los segmentos llevan timestamp de ida y de respuesta
  - Solo se calcula un RTT con el timestamp de confirmación a nuevos datos
  - Hay reglas para decidir qué timestamp devolver en los casos con pérdidas y huecos

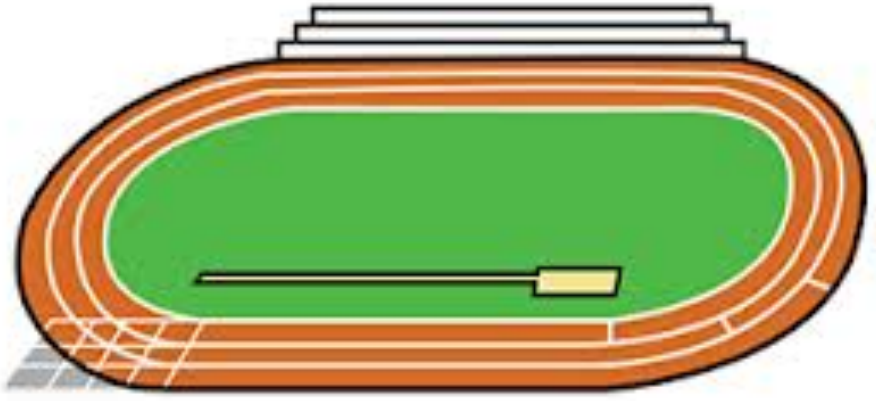


"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Tamaño de la ventana de control de flujo
- Mejores medidas de RTT
- **Problemas de fiabilidad**
  - Número de secuencia se emplea con aritmética módulo
  - ¿Se puede repetir? Tiene un rango de 2^{32} bytes
  - 2^{32} bytes en una Gigabit Ethernet se transmiten en (...)

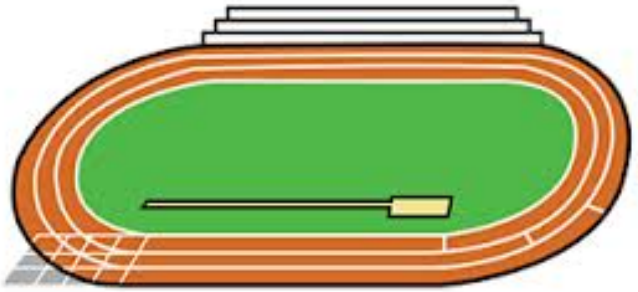
"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Tamaño de la ventana de control de flujo
- Mejores medidas de RTT
- **Problemas de fiabilidad**
  - Número de secuencia se emplea con aritmética módulo
  - ¿Se puede repetir? Tiene un rango de 2^{32} bytes
  - 2^{32} bytes en una Gigabit Ethernet se transmiten en ... ¡ 34 segundos !
  - ¿Qué sucede si se repite por haber dado toda la vuelta?
  - Paquetes en la red retrasados ese tiempo llevan a corrupción
  - (...)



"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Tamaño de la ventana de control de flujo
- Mejores medidas de RTT
- **Problemas de fiabilidad**
  - **PAWS** = *Protection Against Wrapped Sequence numbers*
  - Emplea el valor de la opción timestamp y descarta segmentos "anteriores a los ya recibidos" (referencia el que devuelve para cálculo de RTT)
  - Que haga "tick" suficientemente rápido para que cambie antes de que se dé la vuelta el número de secuencia
  - Que no haga "tick" tan deprisa que se dé la vuelta el reloj antes del tiempo de vida de los paquetes en la red
  - Cuándo hace "tick" el reloj depende de la implementación (1ms-1s)

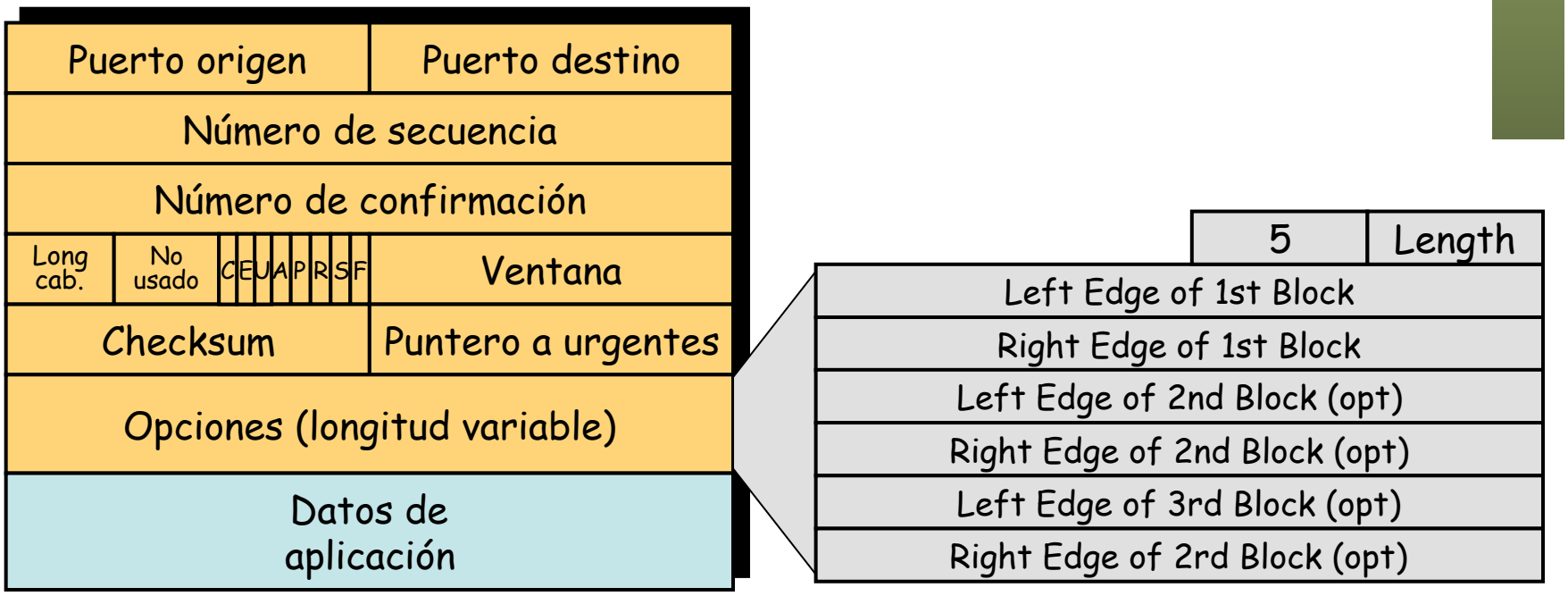


"High Performance TCP"

- RFC 1323 TCP Extensions for High Performance (1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Tamaño de la ventana de control de flujo
- Mejores medidas de RTT
- Problemas de fiabilidad
- **Recuperación ante pérdidas**
  - Con Tahoe entra en slow start
  - Con Reno en fast retransmit
  - En general funcionan mal ante varias pérdidas en la ventana
  - Pero ahora la ventana es muy grande
  - Un mecanismo como RED también puede introducir esas pérdidas
  - Se puede mejorar el rendimiento con SACK

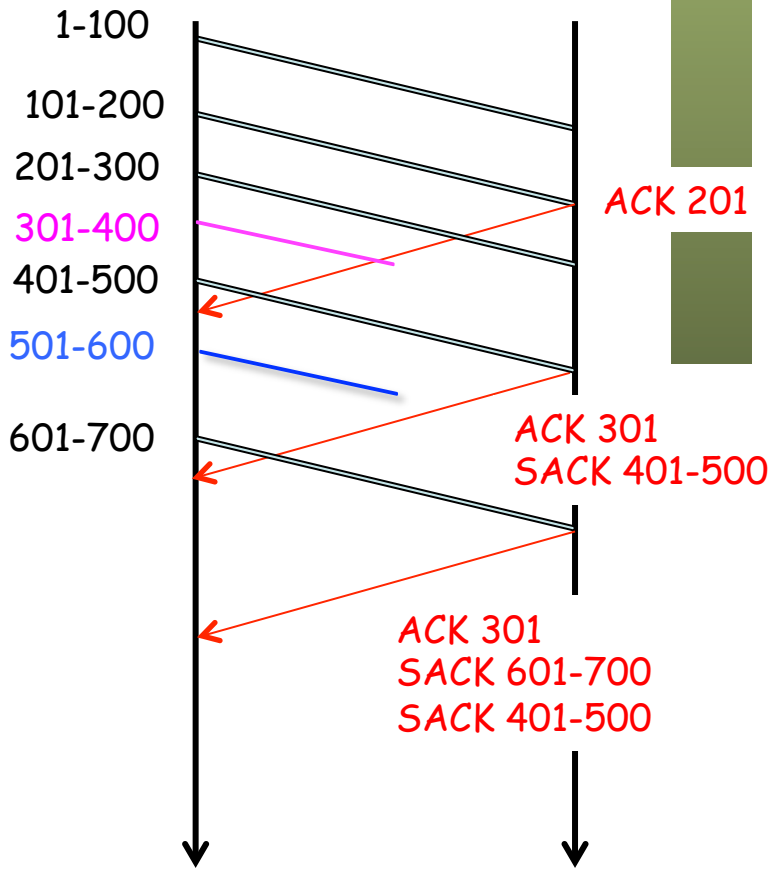
SACK

- RFC 2018 TCP Selective Acknowledgement Options
- Mejora el rendimiento en casos de múltiples pérdidas
- Mediante una opción en el SYN se indica que se soporta
- Mediante otra opción se indican los números de secuencia (rangos) que se confirman (un rango ocupa 8 bytes en la cabecera)
- Se pueden especificar hasta 4 rangos (alcanza máxima cabecera)
- Si se emplea la opción Timestamp baja a 3 rangos



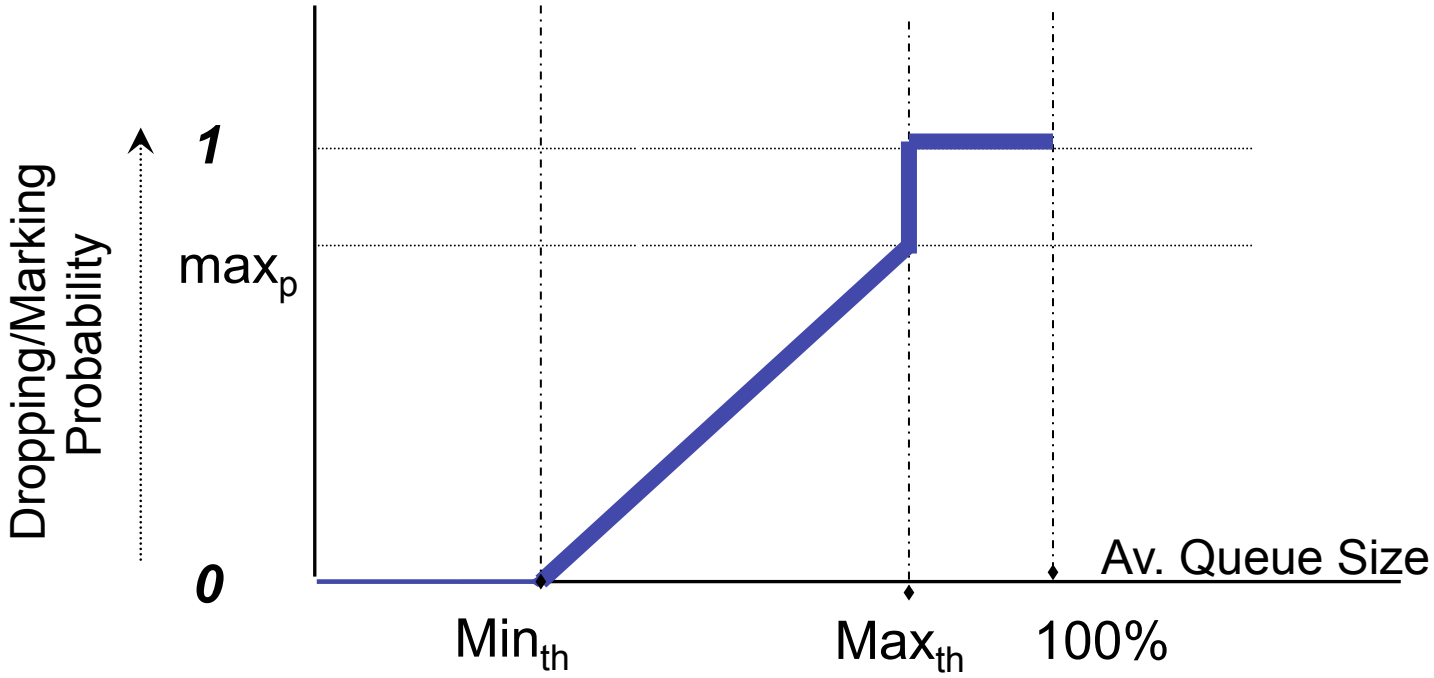
SACK

- No cambia el significado del campo de número de ACK
- El receptor podría descartar más tarde esos datos para los que ha confirmado con SACK (si le falta buffer)
- No puede eliminar los segmentos aunque se hayan confirmado por SACK hasta que lo sean por el nº de ACK
- Si caduca el timer de retransmisión se olvida de lo retransmitido por SACK
- Algoritmo para decidir cuándo y qué retransmitir en RFC 6675



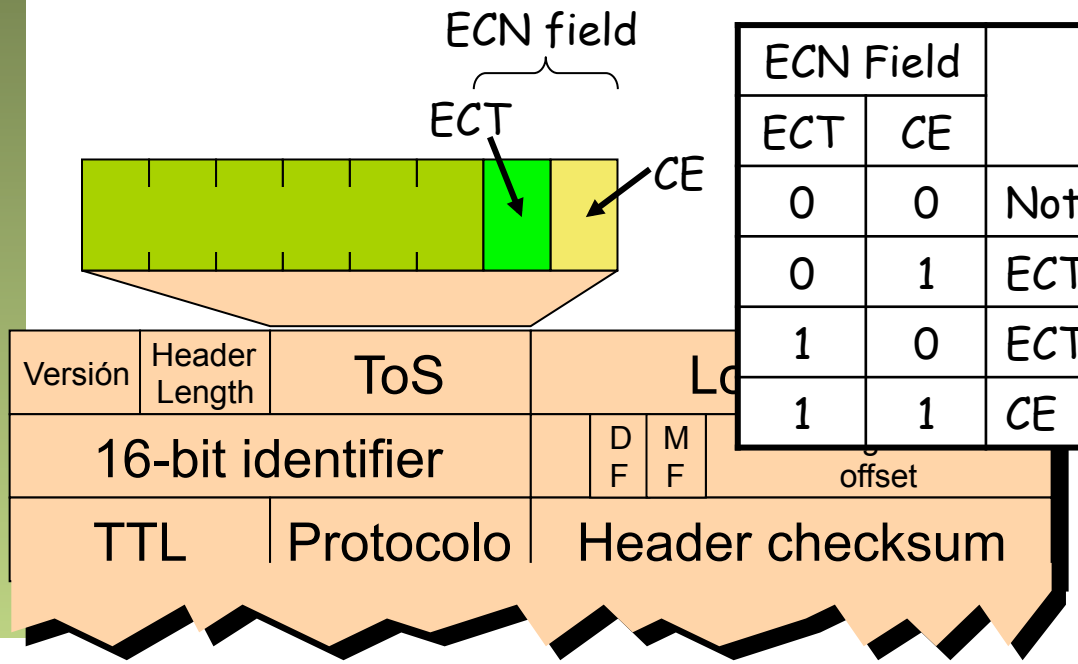
ECN

- Incorpora realimentación explícita
- ECN requiere AQM en los routers (normalmente RED)
- (...)



ECN

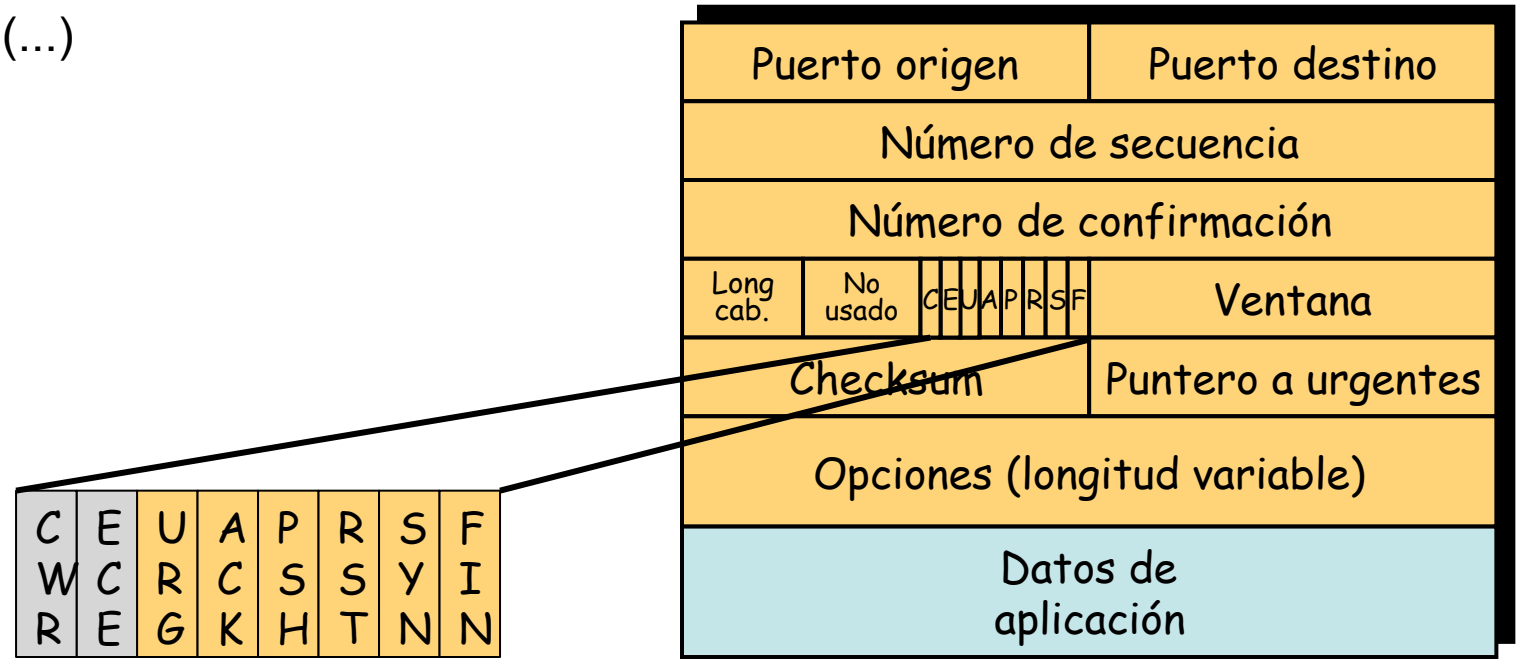
- Incorpora realimentación explícita
- ECN requiere AQM en los routers (normalmente RED)
- Marca en el TOS con dos bits (ECT y CE)
- Tiene mala fama porque algunos firewalls descartaban los paquetes con ECN activo
- La marca se pone al paquete que sufre congestión pero a quien se debe notificar es a la fuente
- (...)



ECN Field			
ECT	CE		
0	0	Not-ECT	Paquete no emplea ECN
0	1	ECT(1)	Extremos "ECN-capable"
1	0	ECT(0)	idem (recomendado)
1	1	CE	Congestion experienced

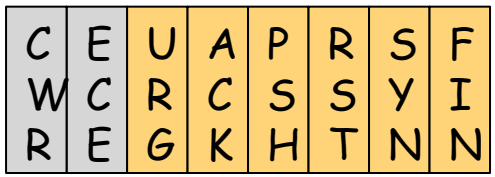
ECN

- Incorpora realimentación explícita
- ECN requiere AQM en los routers (normalmente RED)
- Marca en el TOS con dos bits (ECT y CE)
- Tiene mala fama porque algunos firewalls descartaban los paquetes con ECN activo
- La marca se pone al paquete que sufre congestión pero a quien se debe notificar es a la fuente
- Bit ECE (ECN-Echo) en TCP
- (...)



ECN

- Bit ECE (ECN-Echo) en TCP
- En los ACKs a los datos
- Se pueden perder, así que se activa en varios
- Pero no se quiere que por cada uno que llegue a la fuente se haga una reducción de cwnd
- Cuando la fuente reduce cwnd envía en el siguiente segmento (con datos nuevos) el bit CWR (Congestion Window Reduced) activo
- Es decir, se activa por cualquier causa que lleve a reducir cwnd
- Y solo en el primer paquete nuevo de datos (no en una retransmisión)
- El receptor, al recibir paquete con CWR=1 deja de enviar con ECE=1
- La fuente debería reaccionar a congestión solo 1 vez por cada ventana de ratos (o cada RTT)

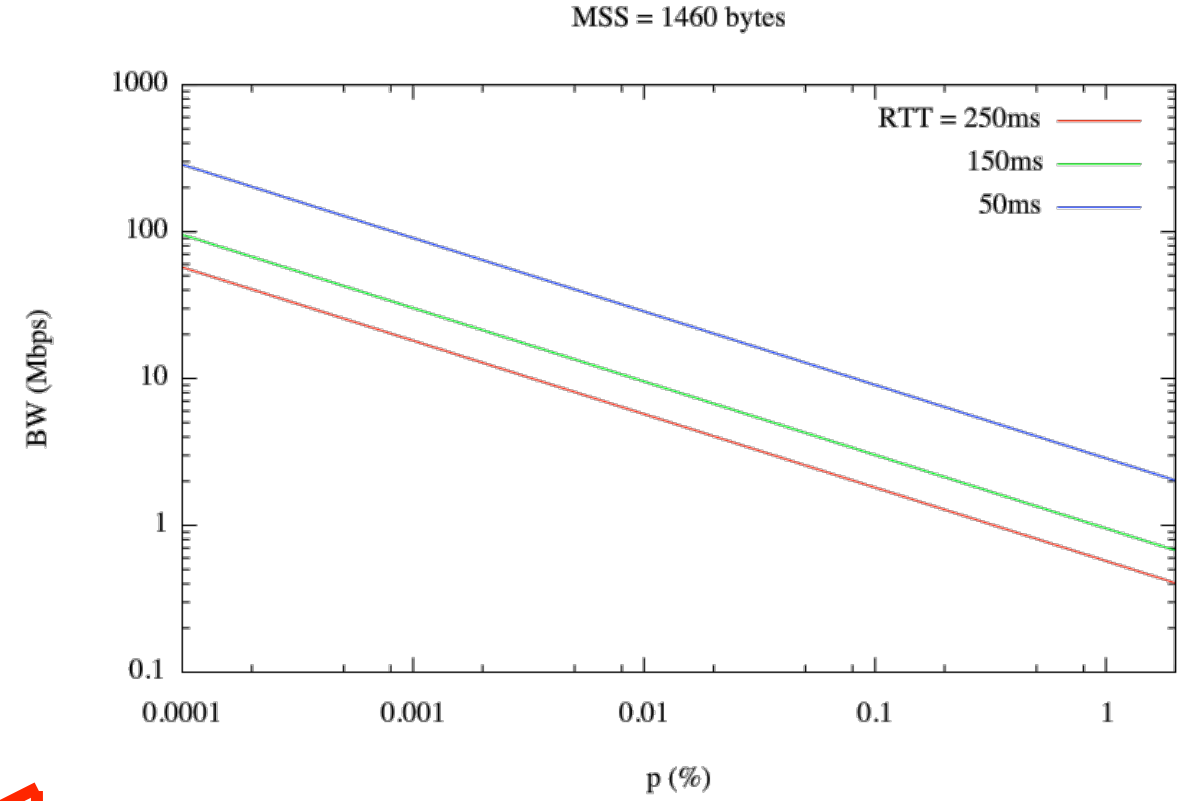


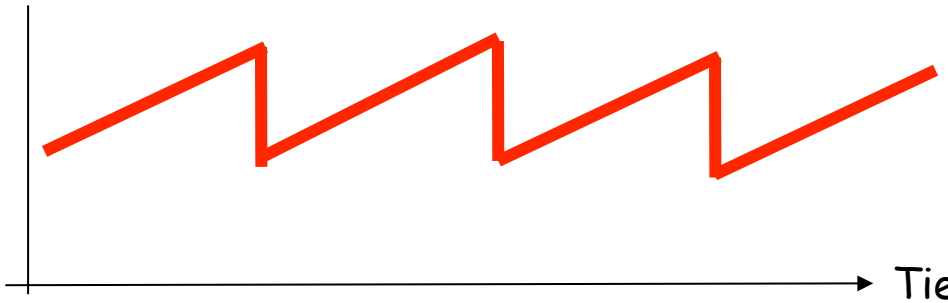
Rendimiento con pérdidas

- Aproximación al throughput promedio para conexiones largas (TCP Reno) con una tasa *p* constante y pequeña de pérdidas

$$BW = \frac{MSS}{RTT}\sqrt{\frac{3}{2p}}$$

M. Mathis
(ACM SIGCOMM'97)





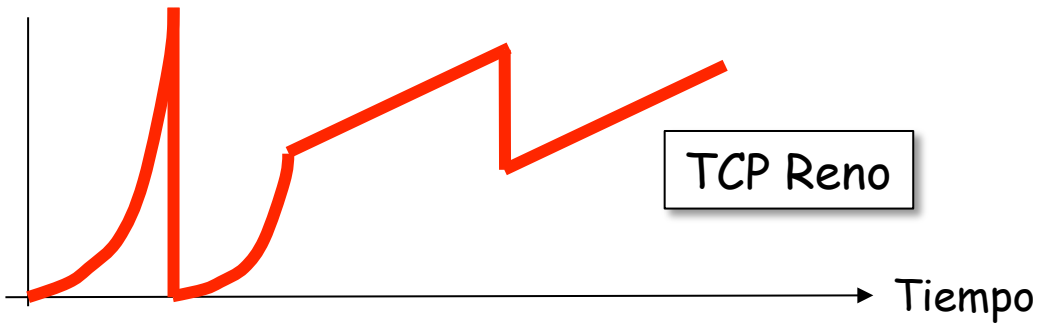
TCP Reno en
congestion avoidance

Rendimiento con pérdidas

- Aproximación al throughput promedio para conexiones largas (TCP Reno) con una tasa *p* constante y pequeña de pérdidas
- Incluye slow start

$$BW = \frac{MSS}{RTT\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

J. Padhye
(ACM SIGCOMM'98)



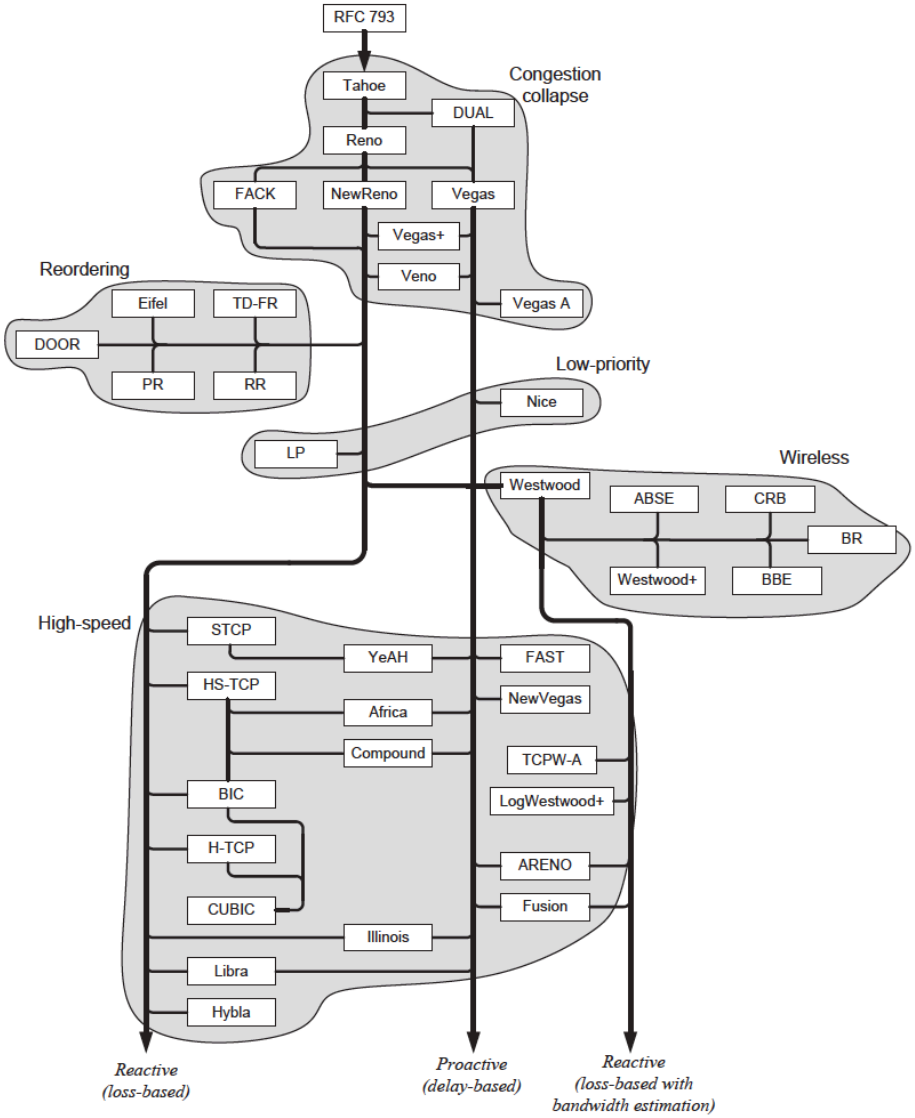
Mecanismos comunes en TCP

- Control de conexiones
- Control de flujo
- Retransmisiones con timer
- Delayed ACK
- Nagle y SWS
- Slow Start
- Congestion avoidance
- Fast retransmit
- Fast recovery
- Retransmisiones ante ACKs duplicados
- SACK
- ECN

Otras propuestas

- TCP-DUAL
- TCP FACK
- TCP Vegas, TCP-Vegas+, TCP-Vegas A
- TCP-Veno
- Eifel detection algorithm (RFC 3522)
- TCP DOOR, TCP PR
- D(uplicate)-SACK (RFC 2883), RR-TCP
- TCP Nice
- TCP LP
- TCP Westwood, TCP Westwood+, TCPW CRB, TCPW ABSE, TCPW BR, TCPW BBE
- HighSpeed TCP (RFC 3649), Scalable TCP, H-TCP- TCP Hybla
- BIC-TCP, CUBIC-TCP (por defecto en Linux kernels 2.6.19+)
- TCPW-A, LogWestwood+
- FAST TCP
- TCP Libra
- TCP NewVegas
- TCP AR, TCP Fusion
- TCP Africa, Compound TCP
- TCP Illinois
- YeAH TCP
- Señalización ECN robusta (RFC 3540)
- Multi-level ECN
- TCP-FR
- Forward RTO-Recovery (F-RTO) (RFC 5682)
- etc.

Otras propuestas



A.Afanasyev, N.Tilley, P.Reiher y L.Kleinrock. “Host-to-Host Congestion Control for TCP”. IEEE Communications Surveys & Tutorials, vol.12 nº 3 (2010)

Resumen

- TCP Reno
  - Fast retransmit
  - Fast recovery
- Mejoras para LFPs
  - Aumento de ventana
  - Timestamp para medir RTT
  - SACK