

Congestión y rendimiento de TCP

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

Grado en Ingeniería en Tecnologías de
Telecomunicación, 4º

Límites de la ventana

Límites de la ventana

Escenario LAN

- Por ejemplo 6 conmutadores (2xAcceso + 2xAgregación + 2xCore)
- Retardo de conmutación en LAN (hardware) en unos 25 μ s
- $6 \times 25 = 150 \mu$ s de procesado
- Propagación < 5 μ s (despreciable)
- Falta retardo de procesado en extremos
- Si todo es 10 Mbps
 - Paquete 1518bytes : $1.2 \times 7 = 8.4$ ms (tx)
 - ACK 64 bytes : $51.2 \times 7 = 0.36$ 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 \rightarrow retardos aprox. x0.1

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

Límites de la ventana

Escenario LAN

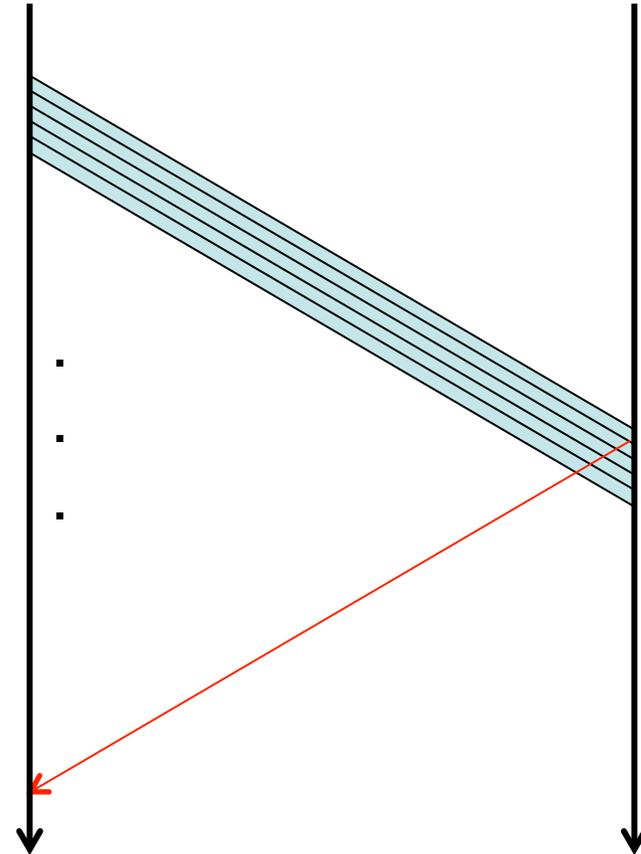
- Por ejemplo 6 conmutadores (2xAcceso + 2xAgregación + 2xCore)
- Retardo de conmutación en LAN (hardware) en unos 25 μ s
- $6 \times 25 = 150 \mu$ s de procesado
- Propagación < 5 μ s (despreciable)
- Falta retardo de procesado en extremos
- Si todo es **1 Gbps**
 - Paquete 1518 bytes : $12 \times 7 = 85 \mu$ s (tx)
 - ACK 64 bytes : $0.51 \times 7 = 3.6 \mu$ s (tx)
 - Propagación + procesado = 310 μ s
 - RTT 400 μ s
 - Procesado es ya muy apreciable y muy dependiente de los equipos
 - Window = 1 Gbps x 400 μ s = 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 μ s
	10 Mbps	51.2 μ s
	100 Mbps	5.1 μ s
	1 Gbps	0.51 μ s
	10 Gbps	0.051 μ s
1518 Bytes	512 Kbps	23.72 ms
	2 Mbps	6.1 ms
	10 Mbps	1.21 ms
	100 Mbps	121 μ s
	1 Gbps	12 μ s
	10 Gbps	1.2 μ s

Límites de la ventana

Escenario WAN

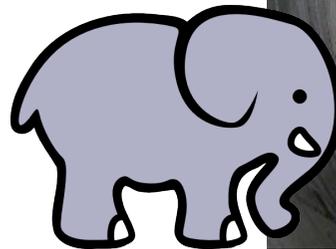
- América – Europa
- Retardo (RTT) transatlántico : 70-90 ms (según cable)
- Retardo (RTT) continental : 30 ms (según destino)
- Altas velocidades (si acceso no limita)
- Tiempos de transmisión despreciables
- RTT en torno a 200-300 ms
- Ventana máxima TCP de 64 KiB
- $65535 \times 8 / 0.2 =$ ¡ 2.6 Mbps máximo !
- Para alcanzar 10 Mbps con esa ventana el RTT debe ser menos de:
 $65535 \times 8 / 10^7 = 52.4$ ms
- Para alcanzar 1 Gbps, < 0.52 ms
- Para mayores retardos necesitamos mayor ventana



High performance TCP

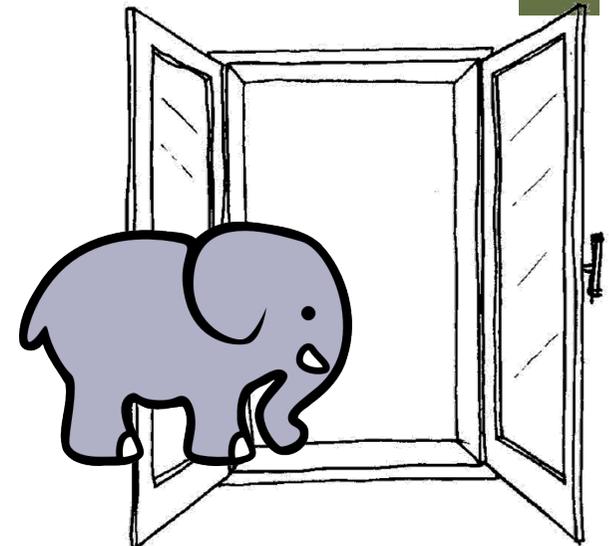
“High Performance TCP”

- RFC 1323 TCP Extensions for High Performance (año 1992)
- Entornos de alta velocidad o alto BWxRTT (*Long, Fat Pipe*)
- Actualizado en RFC 7323 (año 2014)
- Ejemplos de LFP:
 - E3 por satélite BWxRTT = 34Mbps x 0.48s = 2MBytes o más de 1.300 paquetes grandes
 - STM-16 por fibra transatlántica BWxRTT = 2.4Gbps x 0.07s = 21MBytes o más de 14.000 pkts (vale, sí, eso es para que una sola conexión saque provecho a 2.4 Gbps, pero siguen siendo muchos paquetes en vuelo)
- (...)



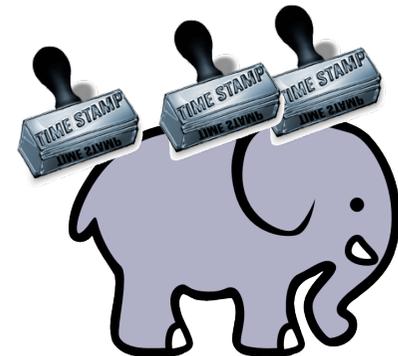
“High Performance TCP”

- **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 (no afecta la escala a la ventana en un SYN o SYN+ACK)
 - 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”

- **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 soportarlo (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
 - Se desactiva el algoritmo de Karn



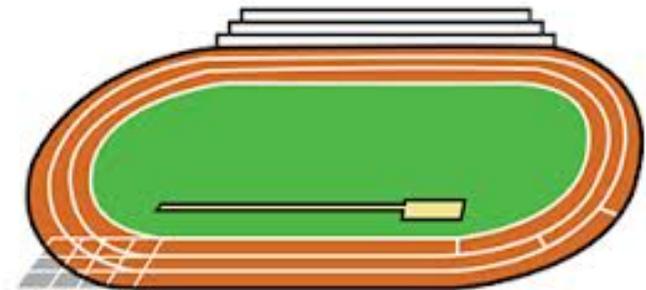
“High Performance TCP”

- **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 (...)



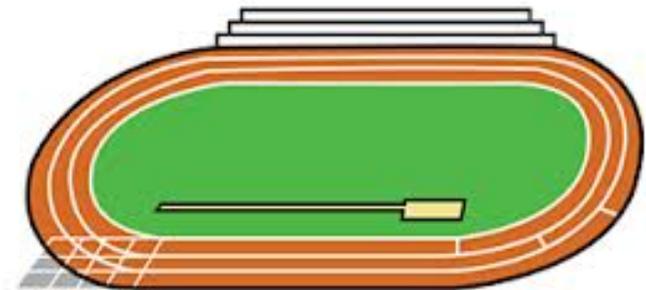
“High Performance TCP”

- **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”

- **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)



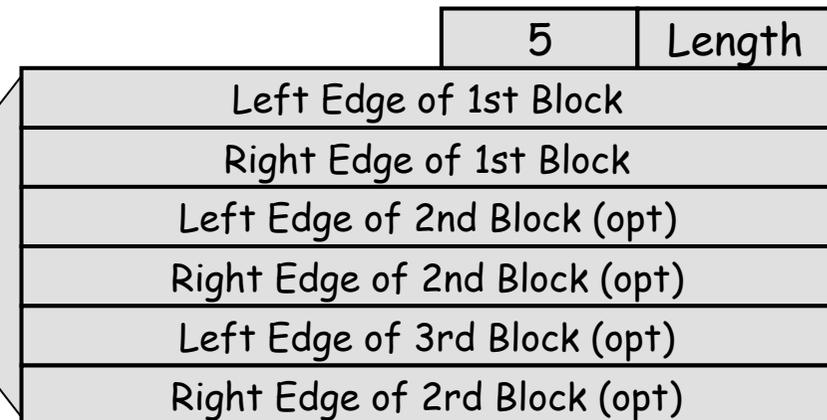
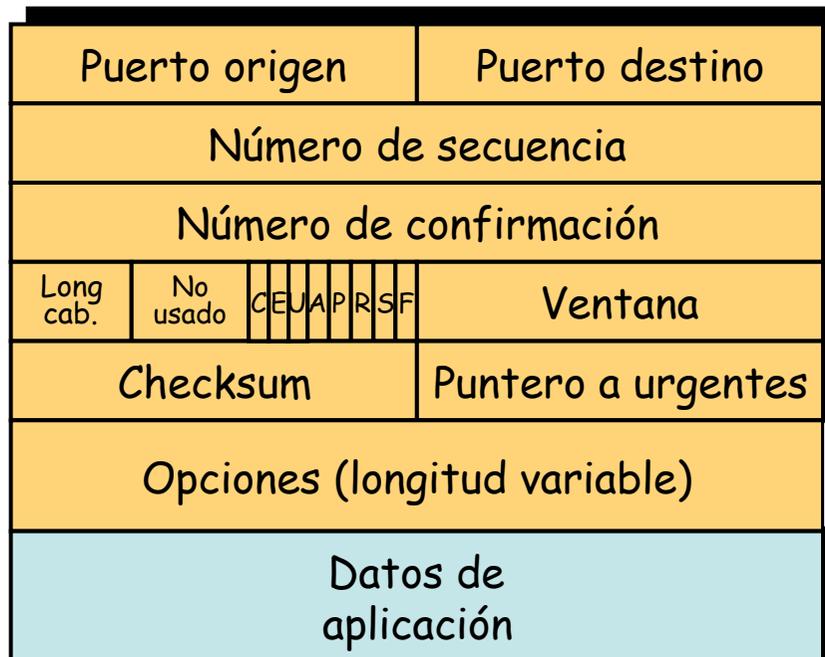
“High Performance TCP”

- **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 (...)



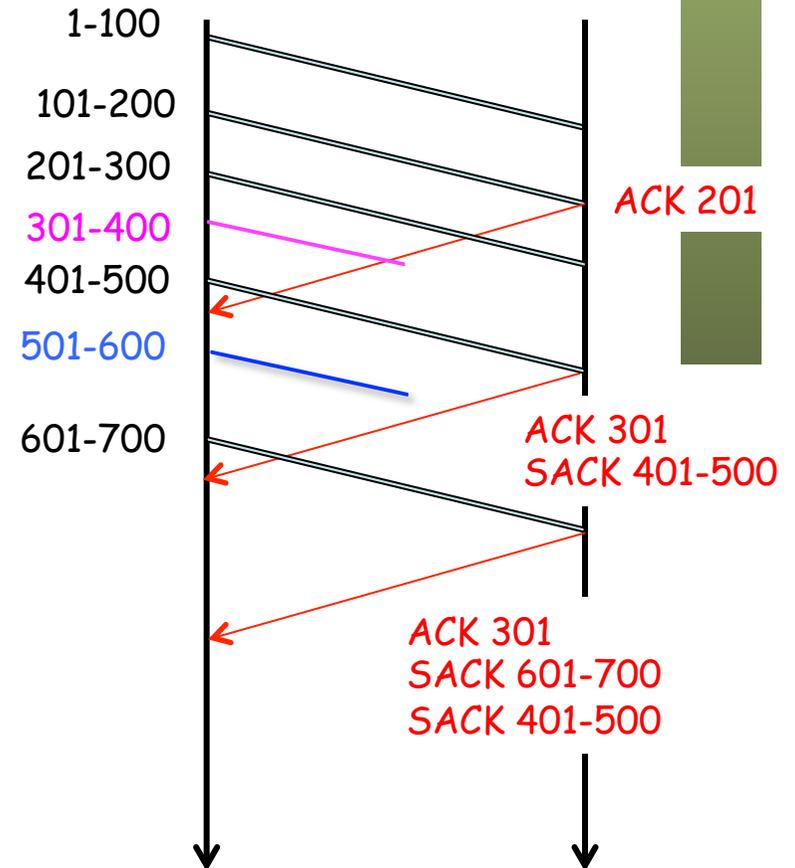
SACK

- 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

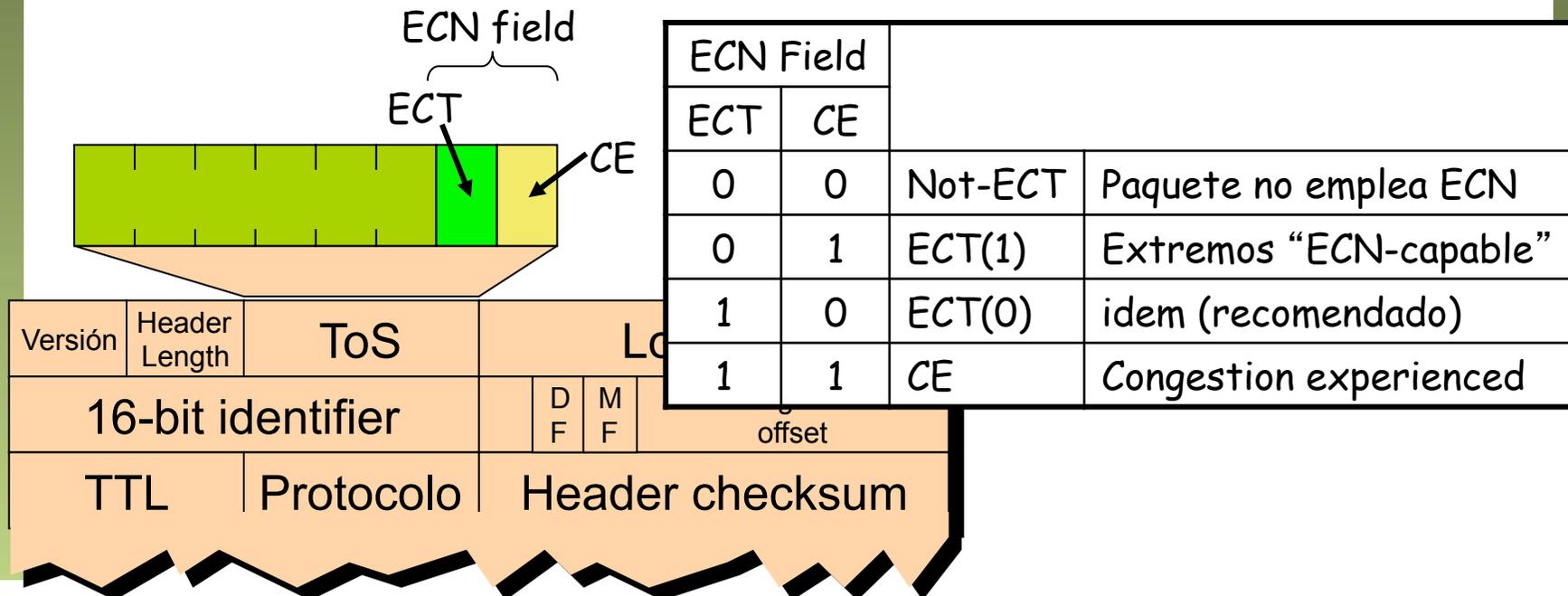
- No cambia el significado del campo de número de ACK
- El receptor podría descartar más tarde esos datos para los que ha confirmado con SACK (si le falta buffer)
- No puede eliminar los segmentos aunque se hayan confirmado por SACK hasta que lo sean por el nº de ACK
- Si caduca el timer de retransmisión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



ECN y TCP

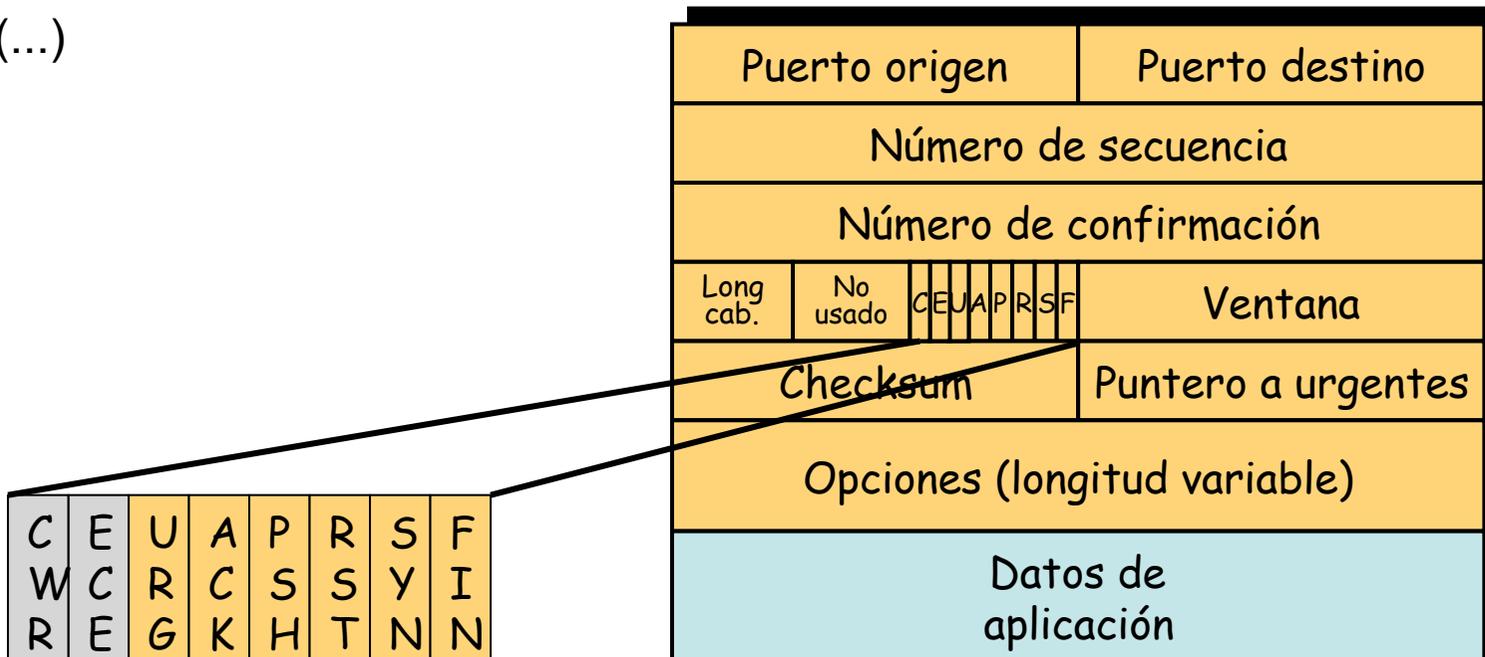
ECN

- 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

- 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)

C	E	U	A	P	R	S	F
W	C	R	C	S	S	Y	I
R	E	G	K	H	T	N	N

Rendimiento con pérdidas

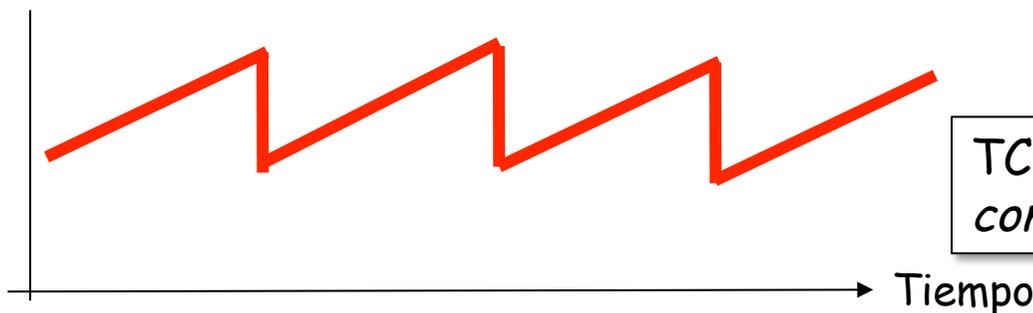
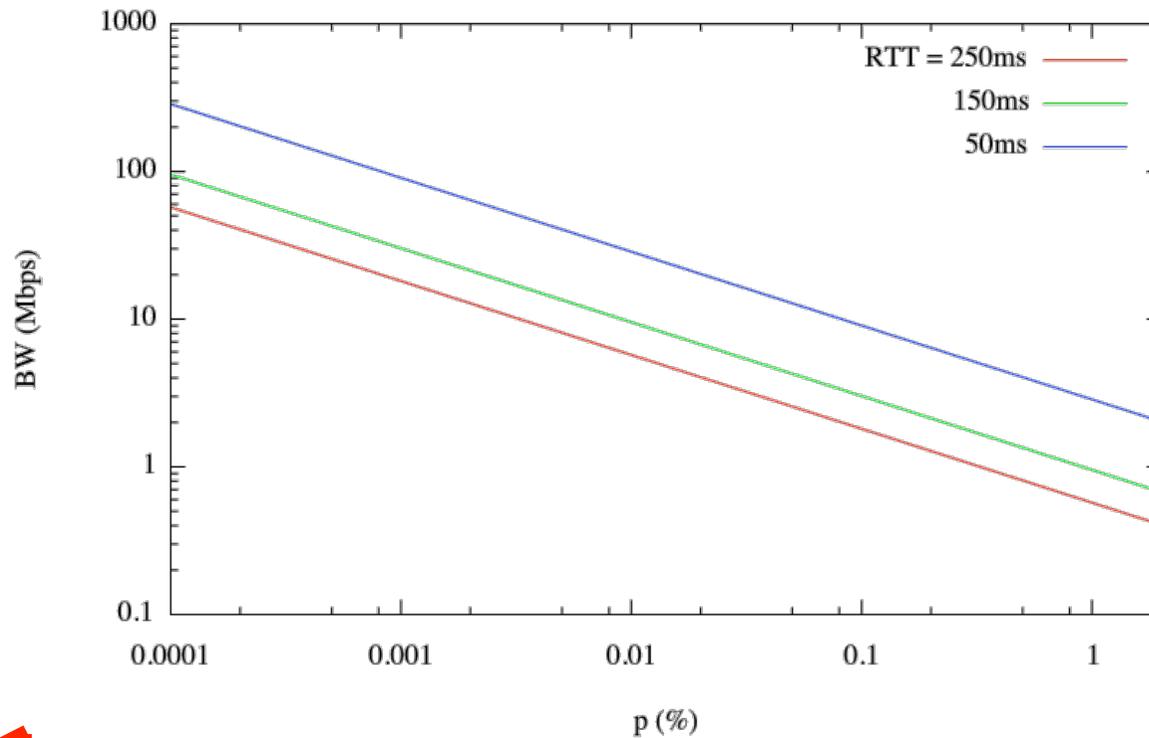
Rendimiento con pérdidas

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

MSS = 1460 bytes

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

*M. Mathis
 (ACM SIGCOMM'97)*



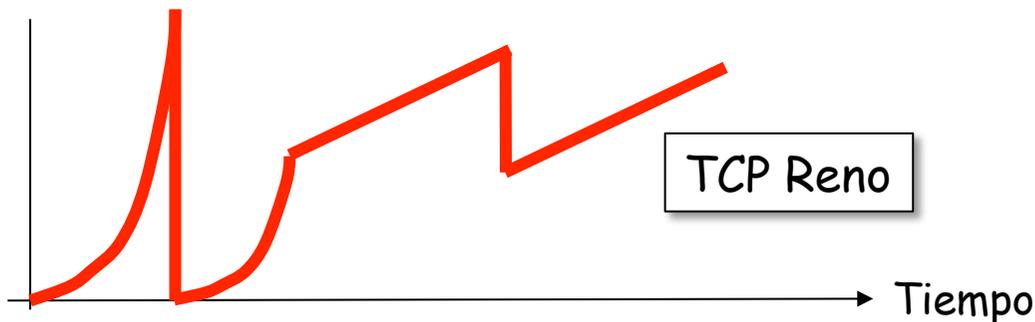
TCP Reno en
congestion avoidance

Rendimiento con pérdidas

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

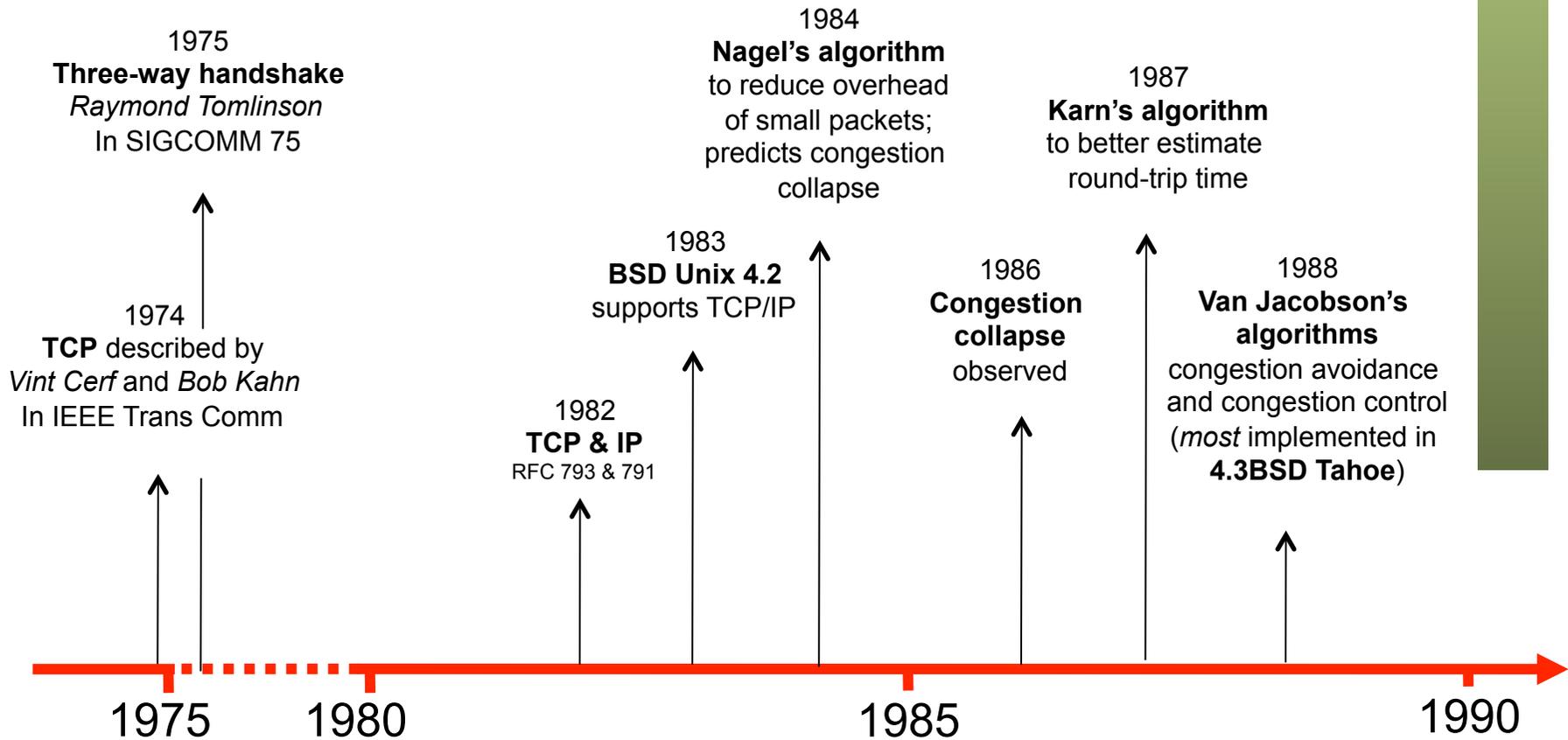
$$BW = \frac{MSS}{RTT \sqrt{\frac{2p}{3}} + t_{RTO} \left(3 \sqrt{\frac{3p}{8}} \right) p(1 + 32p^2)}$$

*J. Padhye
 (ACM SIGCOMM'98)*

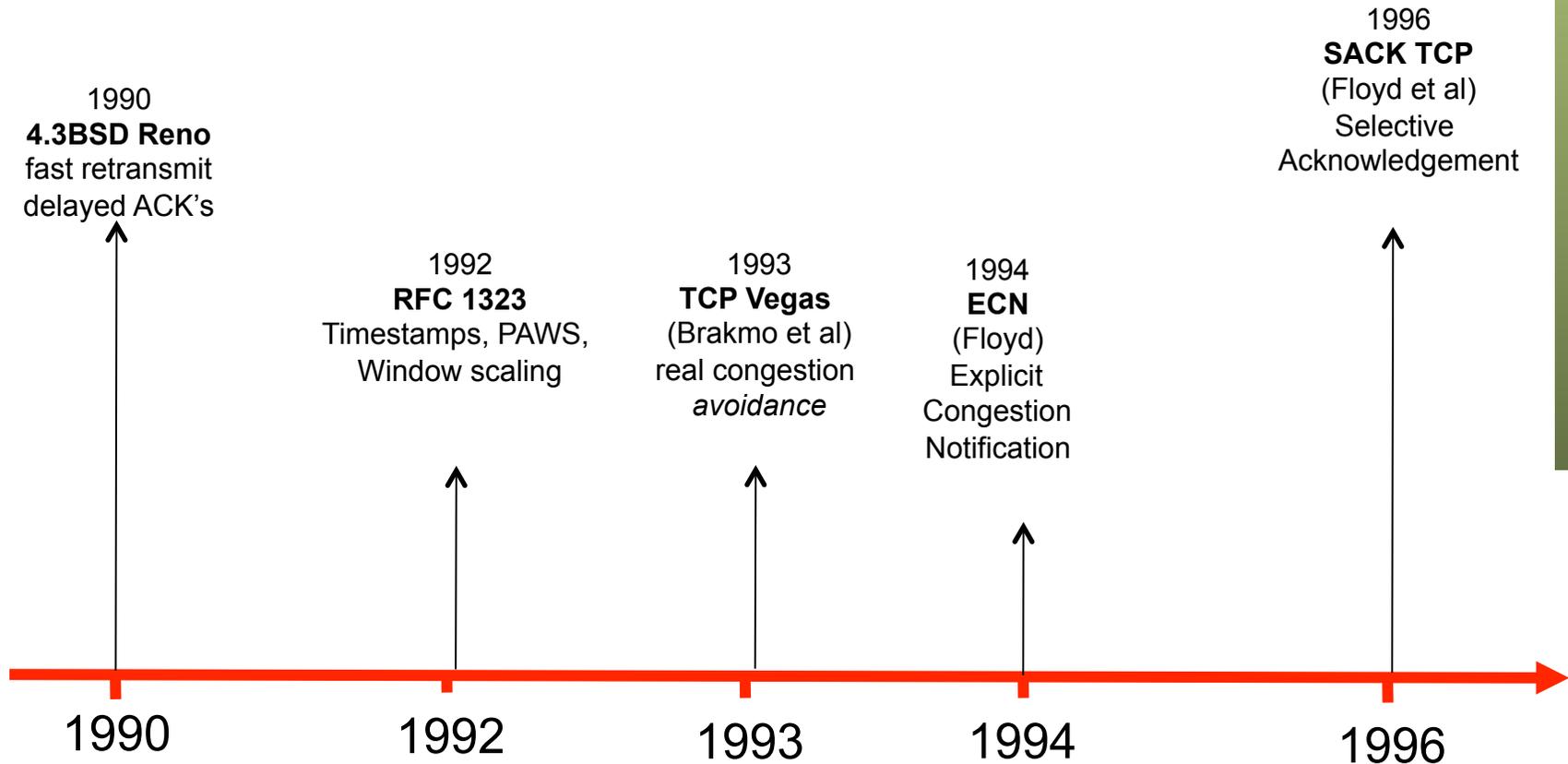


Evolución de TCP

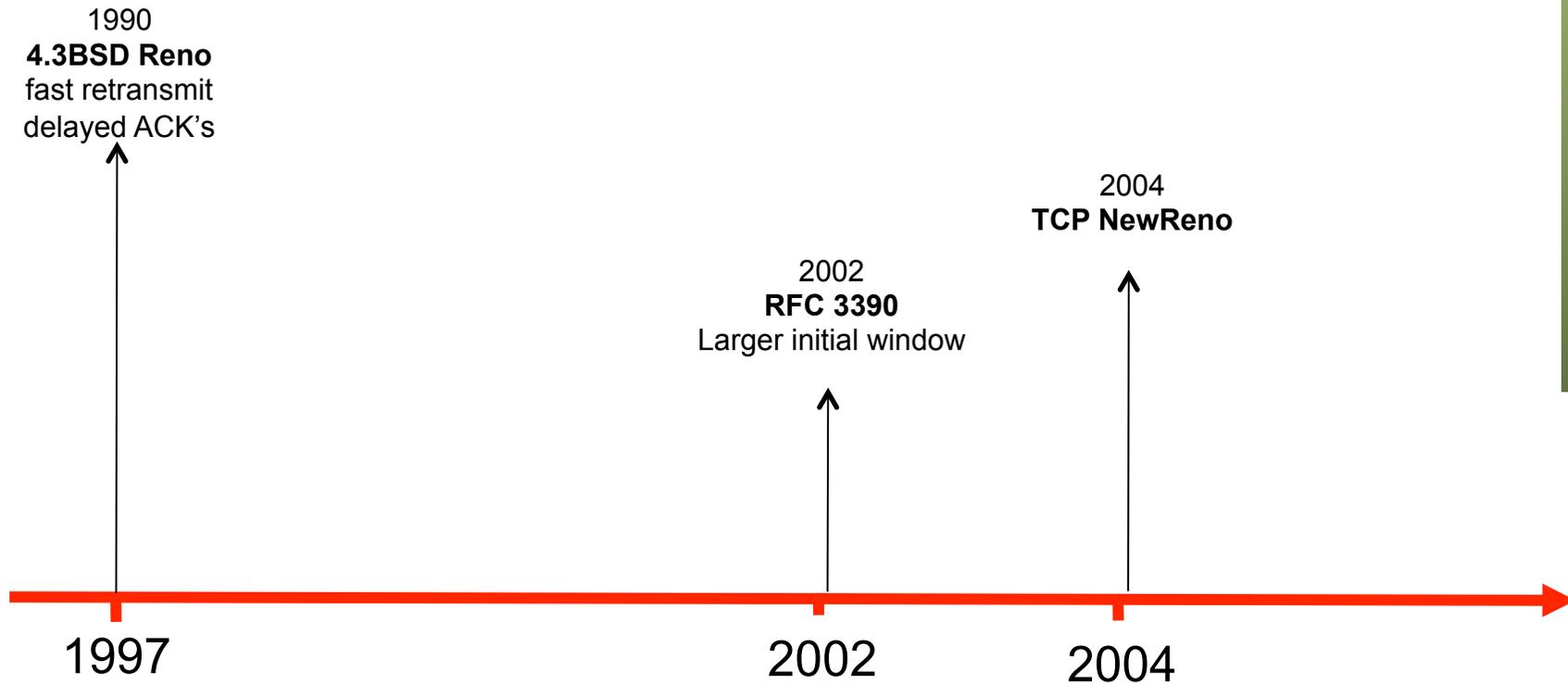
Evolución de TCP



TCP Through the 1990s



TCP Through the 1990s



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.

