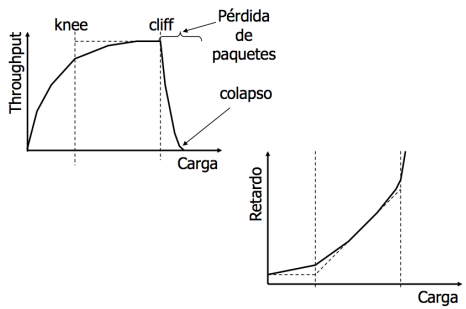
Redes de Datos II
Nivel de Transporte – TCP
Opciones – performance

Luis Marrone

LINTI–UNLP

21 de octubre de 2021



Contenidos

Contenidos

Contenidos

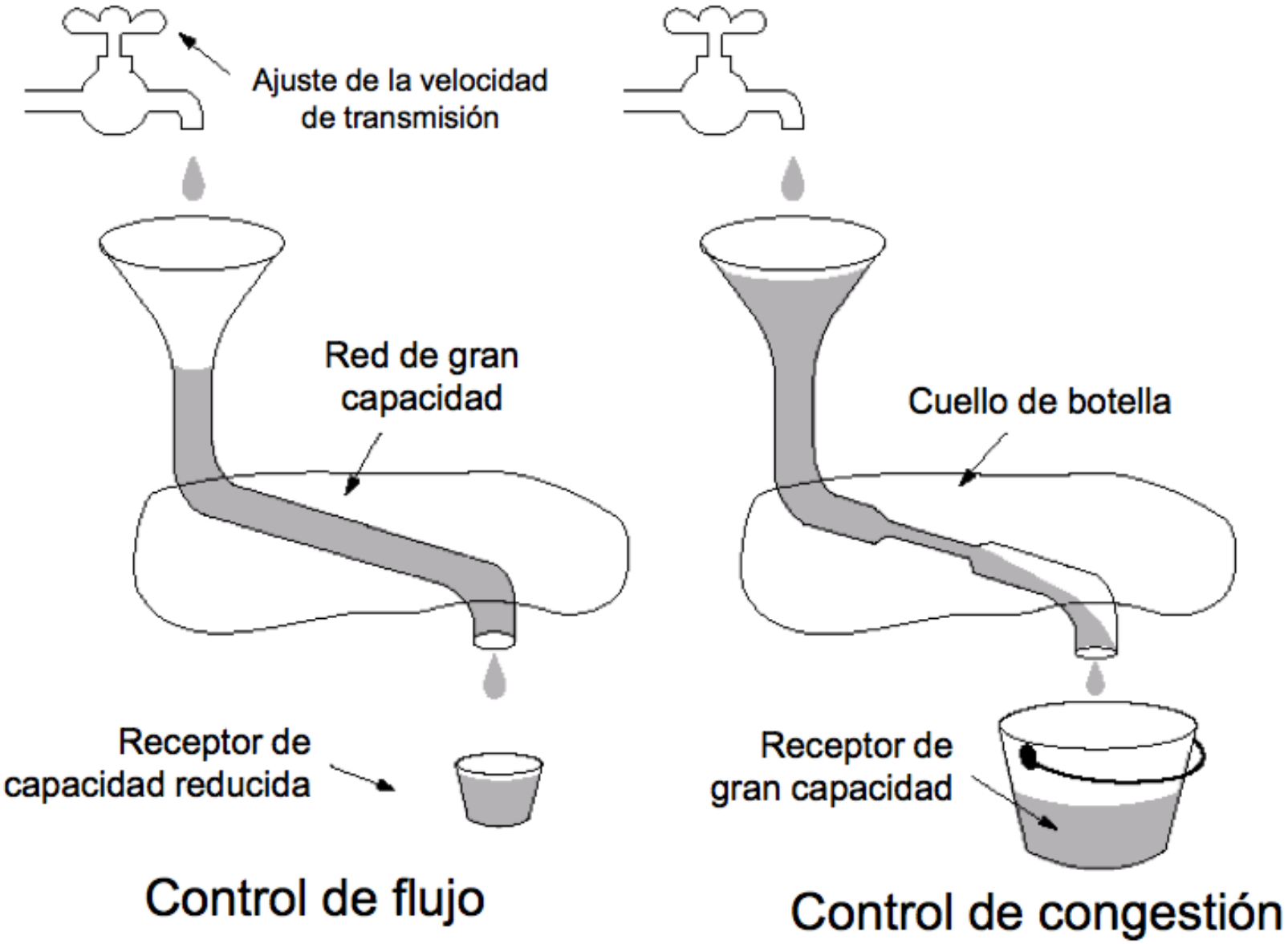
Contenidos

Contenidos

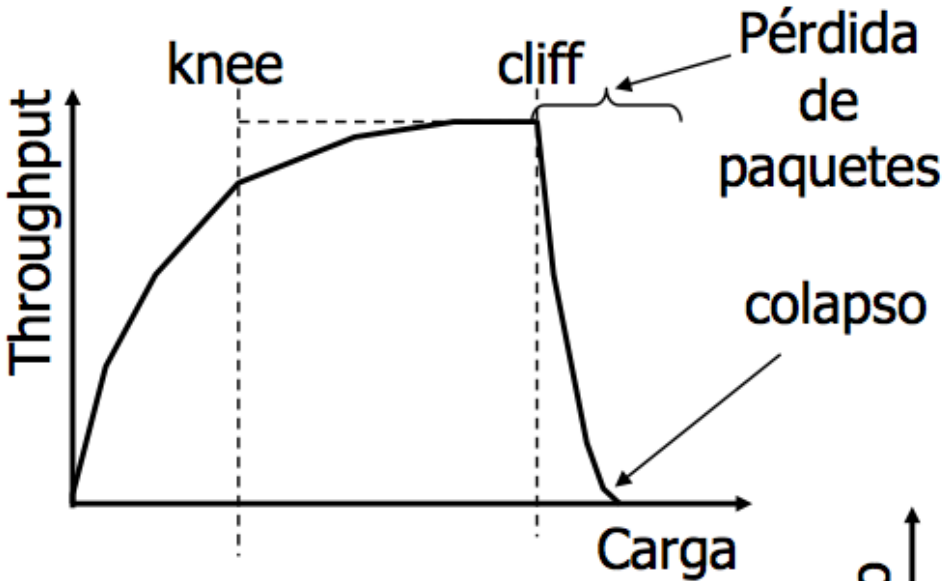
Contenidos

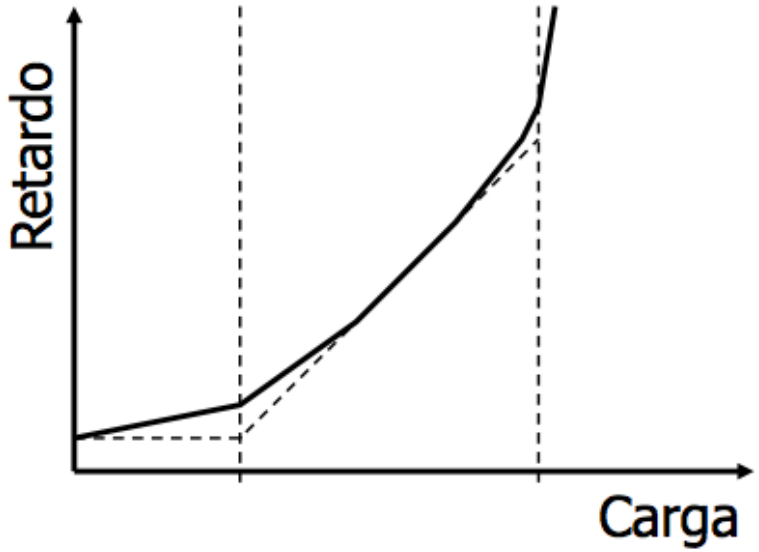
Estamos en:

Control de Flujo vs. Control de Congestión



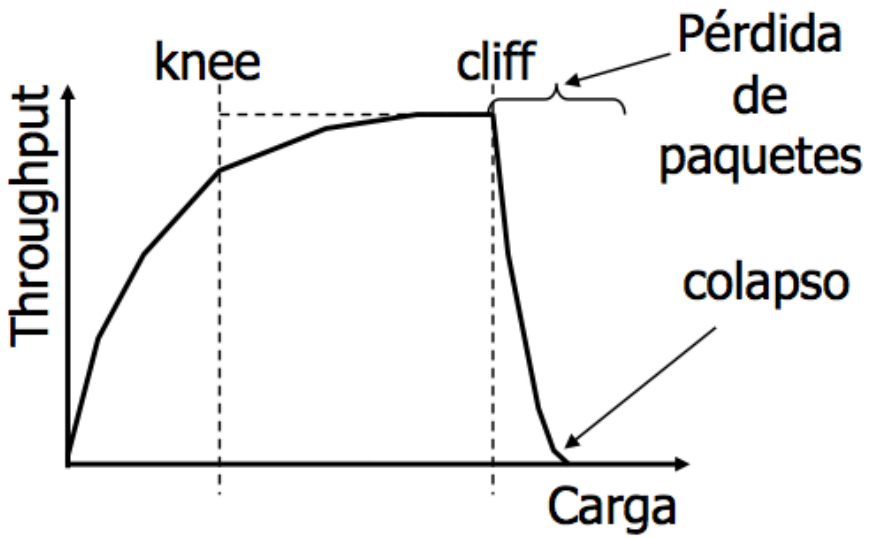
Throughput vs. Congestión





Controlar vs. Evitar

- Controlar
  - Permanecer a la izquierda del “cliff”
- Evitar
  - Permanecer a la izquierda del “knee”



Modelos

- Modelo end-to-end
  - Los extremos son la fuente de la demanda
  - Los extremos deben estimar los tiempos y grado de congestión y reducir la demanda
  - Los nodos intermedios deben monitorear el estado de la red.

- Modelo basado en la red
  - Los extremos no son confiables
  - El nodo de la red tiene control sobre el tráfico
  - Acciones más rápidas

TCP – Modelo end-to-end

- Utiliza tres variables:
  - cwnd: ventana de congestión
  - rcv_win: ventana del receptor. Publicada en el segmento.
  - ssthresh: valor del umbral de “slow start”. Actualiza cwnd.

- Para el envío
  - ventana = min(rcv_win,cwnd)

Slow Start

- ✓ Inicializa el sistema y descubre la congestión rápidamente.
- ✓ Incrementa cwnd hasta la congestión
- ✓ Estima el óptimo cwnd
- ✓ Detecta congestión por pérdida de segmentos
- ✓ Desventajas
  - Detección tardía
  - Enlaces de alta velocidad $--> $ ventanas mayores $--> $ mayor pérdida
  - Interacción con el algoritmo de retransmisión

Slow Start ...

- ✓ En el comienzo o después de congestión

$$cwnd = 1$$

- ✓ Después de cada segmento validado

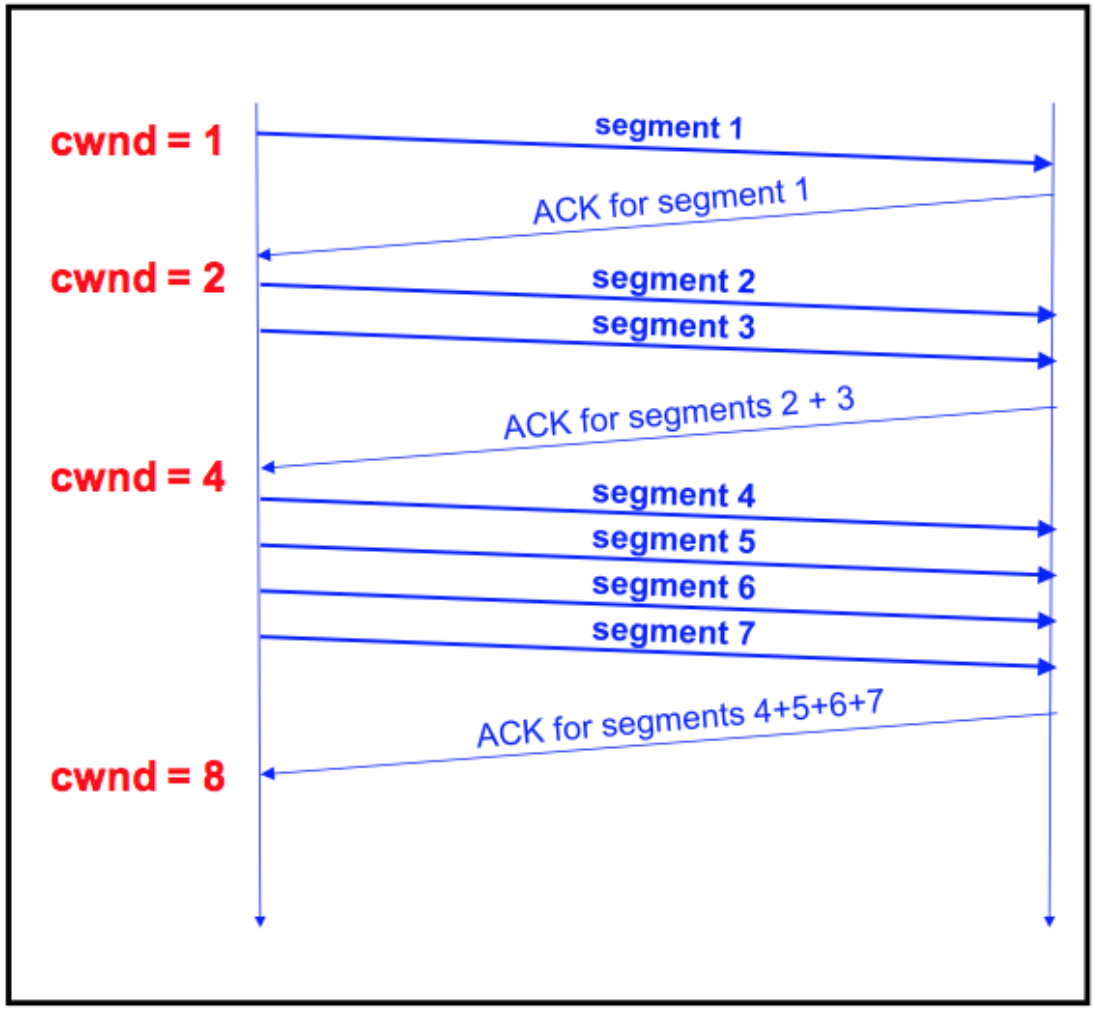
$$cwnd <-- cwnd + 1$$

- ✓ TCP detiene el crecimiento de cwnd cuando

$$cwnd \geq ssthresh$$

- ✓ Pese al incremento unitario resulta exponencial

Slow Start – Ejemplo



Estrategia del control de Congestión

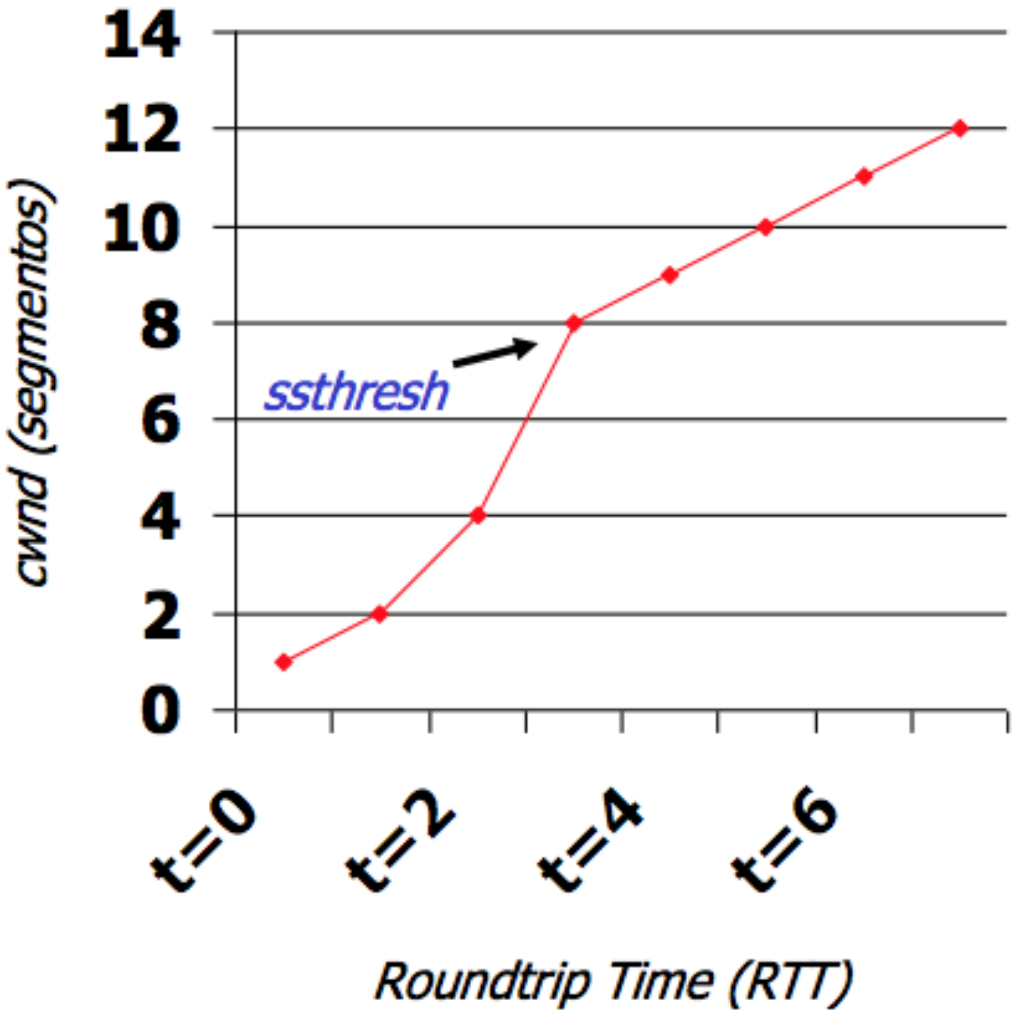
- ✓ Incremento aditivo, comenzar con ssthresh, incrementar cwnd lentamente.
- ✓ Disminución multiplicativa
  Cortar la ventana de congestión drásticamente si se detecta una pérdida.
- ✓ Si *cwnd* $>$ *ssthresh* entonces, por cada ACK

$$cwnd <-- cwnd + \frac{1}{cwnd}$$

- ✓ cwnd se incrementa en 1 si todos los segmentos recibieron su validación.

Combinando...

ssthresh = 8



Armando el Rompecabezas

Inicialmente

$cwnd = 1$

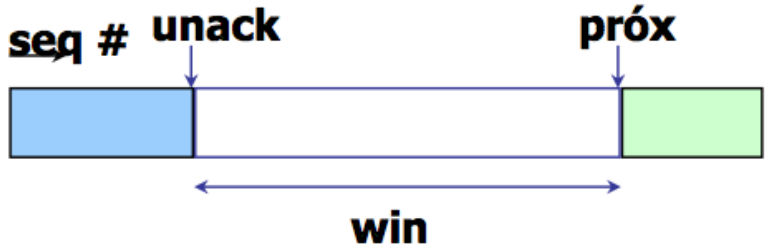
$ssthresh = \infty$

ack recibido
si (cwnd $<$ ssthresh)
/* Slow Start*/
cwnd = cwnd + 1
Tiemout
/* Multiplicative Decrease */

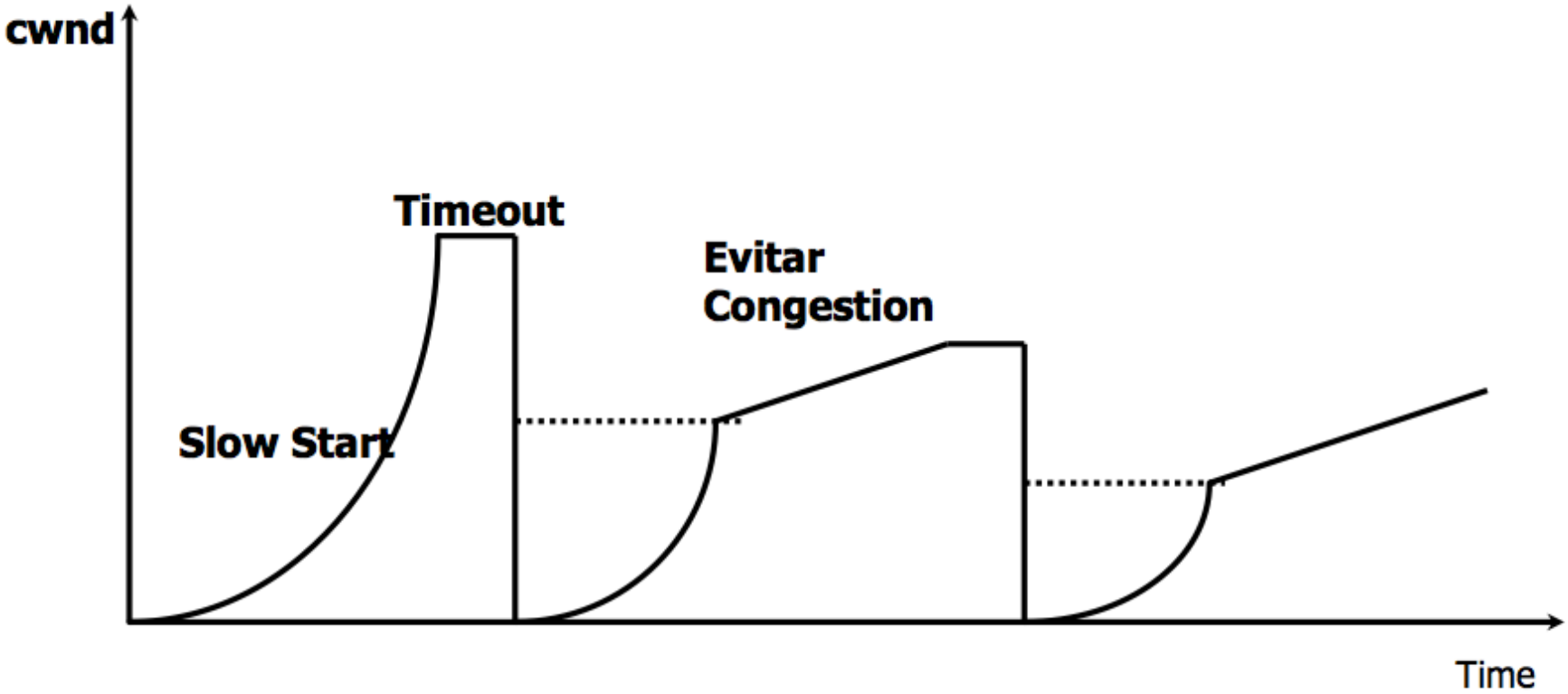
$ssthresh = win/2$

$cwnd = 1$

mientras (próx $<$ unack + win)
transmitir próx segmento
según win = min(cwnd,rcv_win)



Finalmente...



Detección de paquetes perdidos

- ✓ Esperar RTO
- ✓ RTO es usualmente dos veces RTT
- ✓ Degradación de performance
- ✓ No esperar RTO
  - Utilizar mecanismos alternativos
  - Utilizar RTO si no actúan los anteriores

Fast Retransmit y Fast Recovery

- ✓ Frente a un segmento fuera de orden se debe enviar un ACK.
- ✓ Provoca duplicación de ACKs.
- ✓ Esta duplicación se ve como debida a:
  Paquetes perdidos
  Reordenamiento de paquetes
- ✓ No se puede discriminar
- ✓ Si se reciben 3 ACKs duplicados se considera que se debe a un paquete perdido
- ✓ Al recibir el tercer ACK repetido se retransmite sin esperar el RTO.
- ✓ Eso es Fast Retransmit
- ✓ Luego se ejecuta “congestion avoidanc”?, no slow start.
- ✓ Eso es Fast Recovery.

Integrando...

- ✓ Slow Start.
- ✓ Congestion Avoidance.
- ✓ Si aparecen ACKs duplicados
  Fast Retransmit y Fast Recovery.
  Congestion Avoidance.
- ✓ Si RTO
  Slow Start
- ✓ Resumiendo, TCP Reno.

TCP Vegas

- ✓ 1994
- ✓ Crecimiento más lento que el slow start
- ✓ Nuevo mecanismo de retransmisión
- ✓ Se chequea el TO al recibir el primer ACK duplicado
- ✓ Nuevo algoritmo de congestion avoidance
- ✓ Evita las oscilaciones de Reno
- ✓ Monitorea la diferencia entre el throughput estimado y el real
- ✓ Trata de reducir a cero los paquetes almacenados en los buffers de los routers

Ventana TCP Vegas

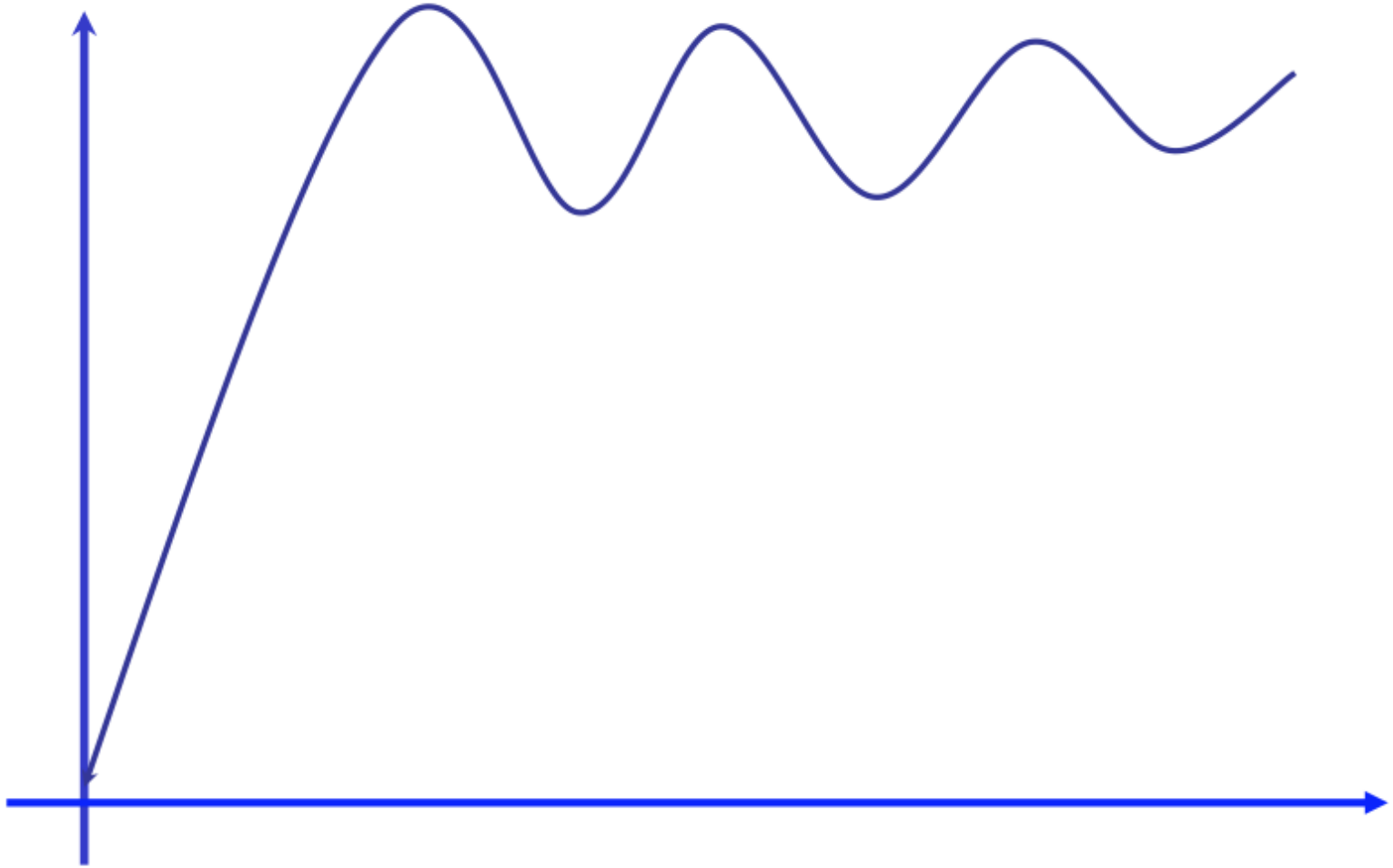
$$w_s(t+1) = \begin{cases} w_s(t) + \frac{1}{D_s(t)}, & si\ \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} < \alpha_s \\ w_s(t) - \frac{1}{D_s(t)}, & si\ \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} > \beta_s \\ w_s(t), & \alpha_s < \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} < \beta \end{cases}$$

$D_s(t) =$ *RTT de la fuente s*

$d_s =$ *mínimo RTT de la fuente s*

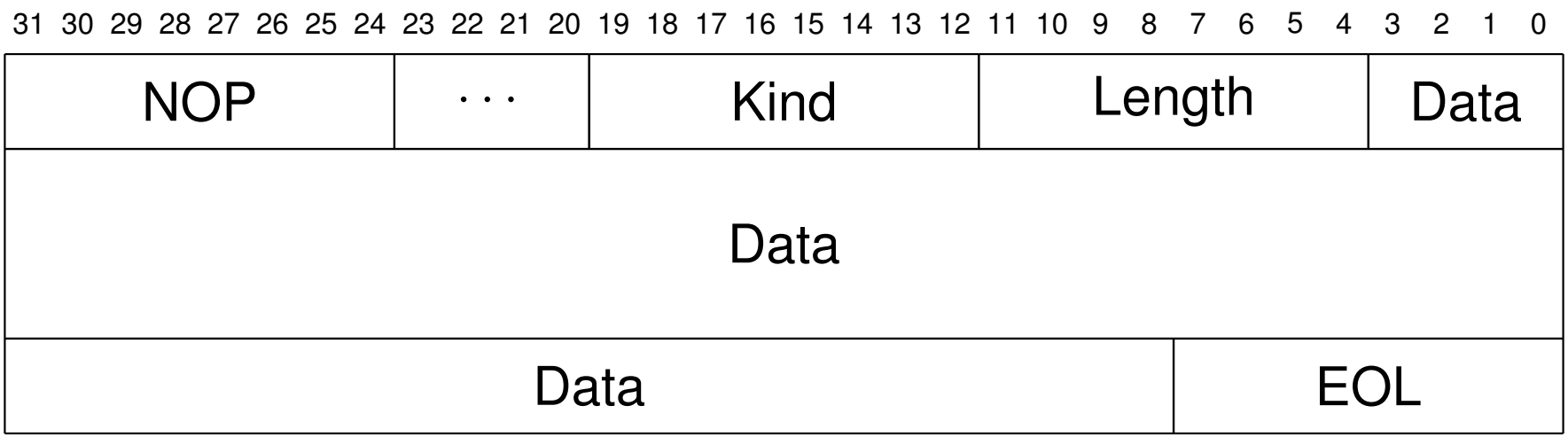
α *y* β, *parámetros*

Throughput TCP Vegas



Estamos en:

Estructura Opciones TCP



Opciones

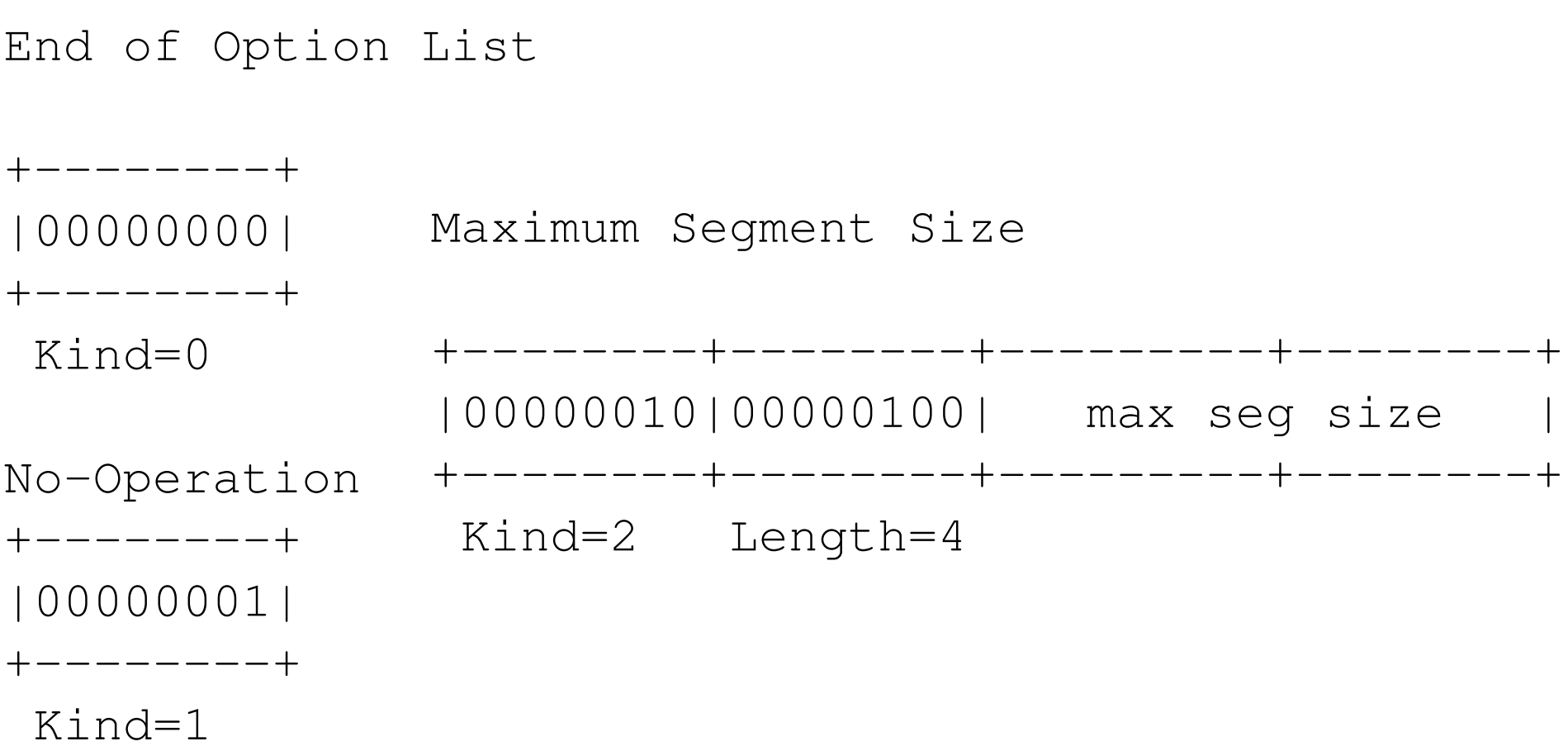
Table 13-1 The TCP option values. Up to 40 bytes are available to hold options.

Kind	Length	Name	Reference	Description and Purpose
0	1	EOL	[RFC0793]	End of Option List
1	1	NOP	[RFC0793]	No Operation (used for padding)
2	4	MSS	[RFC0793]	Maximum Segment Size
3	3	WSOPT	[RFC1323]	Window Scaling Factor (left-shift amount on window)
4	2	SACK-Permitted	[RFC2018]	Sender supports SACK options
5	Var.	SACK	[RFC2018]	SACK block (out-of-order data received)
8	10	TSOPT	[RFC1323]	Timestamps option
28	4	UTO	[RFC5482]	User Timeout (abort after idle time)
29	Var.	TCP-AO	[RFC5925]	Authentication option (using various algorithms)
253	Var.	Experimental	[RFC4727]	Reserved for experimental use
254	Var.	Experimental	[RFC4727]	Reserved for experimental use

.-Extraído de TCP/IP Illustrated 2nd. edition

.- RFC 1323 fue actualizada por RFC 7323

Opciones...

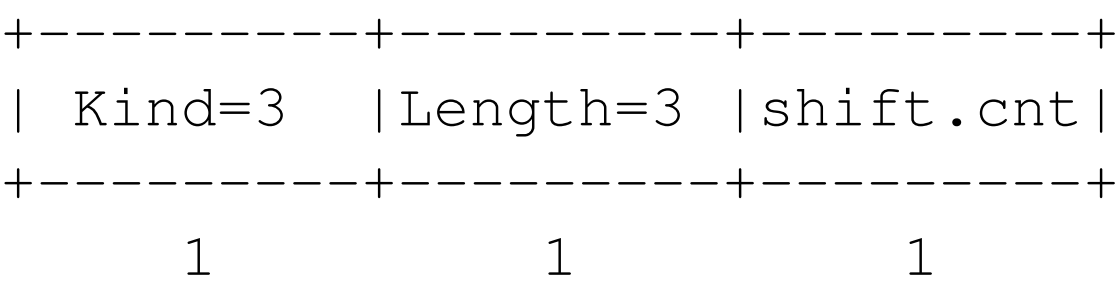

End of Option List
+--------+
|00000000|
+--------+
Kind=0
No-Operation
+--------+
|00000001|
+--------+
Kind=1
Maximum Segment Size
+--------+--------+---------+---------+
|00000010|00000100| max seg size |
+--------+--------+---------+---------+
Kind=2   Length=4

Opciones...

TCP Window Scale option (WSopt):

Kind: 3

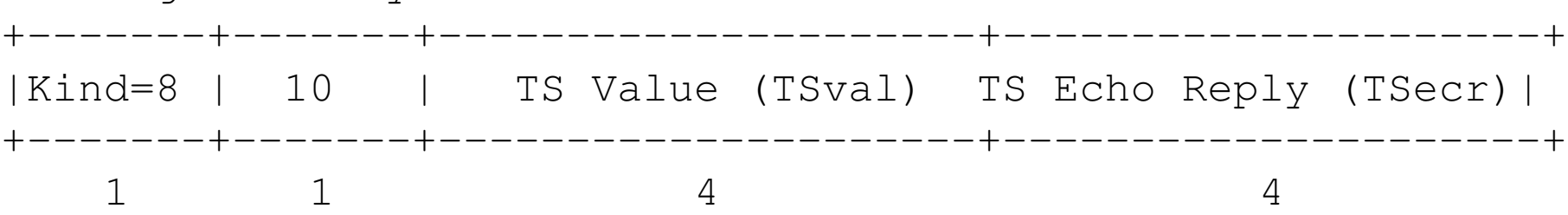
Length: 3 bytes



TCP Timestamps option (TSopt):

Kind: 8

Length: 10 bytes



.- Se le anteponen dos NOPs para alinerarlo a enteros de 32 bits

Estamos en:

TCP Persist Timer - RFC 1122

- ✓ Es necesario hacer una suerte de polling
- ✓ Pueden producirse “deadlocks”
- ✓ Al recibirse una ventana de 0 se activa el persist timer
- ✓ Normalmente 5 segundos
- ✓ Cuando expira se envía un segmento de 1 byte para verificar el estado del receptor
- ✓ El receptor le contesta acorde con el estado en que se encuentra
- ✓ Si el receptor vuelve a constestar con ACK , wind=0, entonces el persist timer hace un back-off binario (10, 20, 40 seg...)
- ✓ Al contestar el ACK no validando el byte recibido el emisor continúa enviando este byte de prueba
- ✓ La diferencia con el RTO es que en este caso el emisor envía permanentemente esta prueba hasta que la ventana se incremente o se haga un reset de la conexión

Keepalive Timer - RFC 1122

- ✓ En TCP si no hay intercambio de datos no hay tráfico alguno, pero la conexión persiste
- ✓ Hasta que haya una caída en algún extremo o reboot de alguno de los hosts
- ✓ Se define para interrogar al otro extremo por su estado
- ✓ No es parte de la especificación de TCP
- ✓ No se recomienda porque puede provocar caídas en caso que la falla sea transitoria
- ✓ También incrementa el uso del ancho de banda disponible como cualquier otra acción de control
- ✓ En algunos casos es necesario. Ej, Telnet

Keepalive Timer... I

- ✓ En el extremo en que esté habilitado si no hay actividad en un lapso de tiempo configurable envía un segmento de prueba
- ✓ El segmento de prueba es similar al de persist timer
- ✓ Puede también enviarse vacío como un ACK pero con un número de secuencia inesperado
- ✓ El receptor se puede encontrar en alguno de estos estados:

  Activo: Responderá al segmento de prueba. El emisor resetea el timer por otras 2 horas. Si en ese intervalo aparece tráfico entonces se vuelve a resetear

  Caído: o en proceso de reboot. El emisor no recibirá respuesta y se genera un timeout a los 75 segundos. Repite 10 veces en intervalos de 75 seg. Si no recibe respuesta finaliza la conexión

  Fin-reboot: el emisor recibirá una respuesta que será un reset de la conexión

Keepalive Timer... II

Receptor activo **pero inalcanzable por el emisor:** el emisor no recibe respuesta y genera un timeout de 75 seg al cabo de los cuales retransmite el byte de prueba. Es similar al caso 2

“Wrap around”

- ✓ El número de secuencia tiene 32 bits
- ✓ En velocidades altas el espacio de 32 bits puede reciclar dentro del intervalo de tiempo que un segmento es retardado en la red
- ✓ Para conseguir una operación libre de este error:

$$2^{31}/B > MSL(seg)$$

donde B es la velocidad efectiva del enlace en bytes/seg

$$Twrap = 2^{31}/B$$

Red	B (MBps)	Twrap (seg)
Ethernet	1.25	1700
FDDI	12.5	170
Gigabit	125	17

PAWS - RFC 1123-7323

- ✓ Definido para rechazar segmentos duplicados y reencarnaciones
- ✓ Utiliza la opción de timestamp
- ✓ Asume que los segmentos de datos y ACK recibidos contienen un valor de TS monótono no-decreciente
- ✓ Un segmento puede ser descartado como duplicado si se recibe con un valor de TS menor que uno anterior
- ✓ “Menor que” significa que si s y t son valores de TS, entonces

$$s < t \; si \; 0 < (t - s) < 2^{31}$$

- ✓ Los valores de TS enviados en <SYN> y/o <SYN,ACK> inicializan PAWS
- ✓ No requiere sincronización de relojes entre emisor y receptor

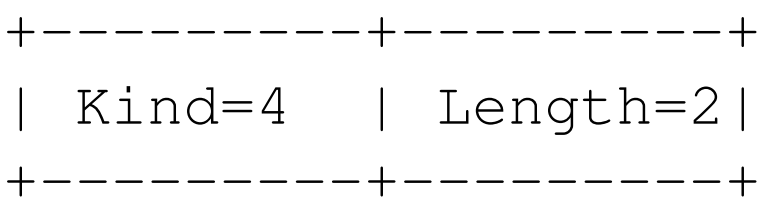
Estamos en:

SACK - RFC 2018

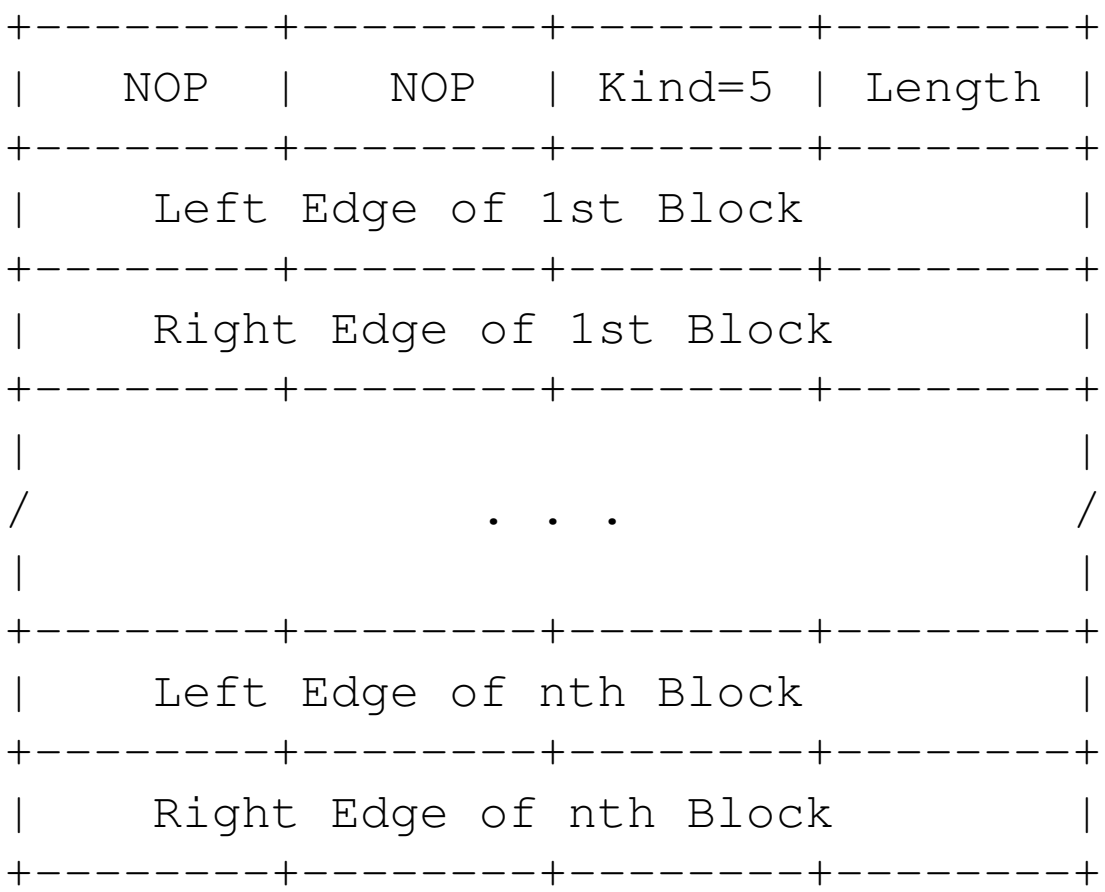
- ✓ Produce la retransmisión selectiva
- ✓ RFC 2018
- ✓ Comprende dos opciones de TCP
  - ✓ “Sack-Permitted Option”
  - ✓ “Sack Option”

Sack Permitted Option

- ✓ Enviada con el SYN
- ✓ Habilita el uso de SACK



Sack Option – Estructura



Los NOP se agregan para alinear la opción a palabras de 32 bits (4 bytes).

Left Edge of Block

Es el primer número de secuencia del block.

Right Edge of Block

Es el número de secuencia inmediato siguiente al último numero de secuencia del block. Los bytes por debajo del block(Left Edge of Block - 1), y justo por encima del block (Right Edge of Block), no se recibieron.

Dado que las opciones tienen una longitud máxima de 40 bytes esto hace que no se puedan indicar más de 4 huecos.

Al recibirse se produce la retransmisión de lo que indica la opción

Los bloques representan bytes disponibles en el buffer.

Ejemplos SACK

El transmisor envía 8 segmentos de 500 bytes de datos con SN:5000
Caso 1:
Se pierde el primer segmento y llegan OK los 7 restantes:

Triggering Segment	ACK	Left Edge	Right Edge
5000	(lost)		
5500	5000	5500	6000
6000	5000	5500	6500
6500	5000	5500	7000
7000	5000	5500	7500
7500	5000	5500	8000
8000	5000	5500	8500
8500	5000	5500	9000

Ejemplos SACK...

El transmisor envía 8 segmentos de 500 bytes de datos con SN:5000
Caso 2:
Se pierden los segmentos pares:

Triggering Segment	ACK	1er Bloque		2do Bloque		3er Bloque	
		Left Edge	Right Edge	Left Edge	Right Edge	Left Edge	Right Edge
5000	5500						
5500	(lost)						
6000	5500	6000	6500				
6500	(lost)						
7000	5500	7000	7500	6000	6500		
7500	(lost)						
8000	5500	8000	8500	7000	7500	6000	6500
8500	(lost)						

Ejemplos SACK...

Continuando con el caso anterior, supongamos que se recibe el cuarto paquete a continuación. Entonces:

Triggering Segment	ACK	1er Bloque		2do Bloque		3er Bloque	
		Left Edge	Right Edge	Left Edge	Right Edge	Left Edge	Right Edge
6500	5500	6000	7500	8000	8500		

Si luego llega el segundo segmento:

Triggering Segment	ACK	1er Bloque		2do Bloque		3er Bloque	
		Left Edge	Right Edge	Left Edge	Right Edge	Left Edge	Right Edge
5500	7500	8000	8500				

Estamos en:

Generalidades

- ✓ Retomar el modelo basado en la red para control de congestión (FR, ATM)
- ✓ Participan IP y TCP
- ✓ Colabora con RED
- ✓ RED procede al marcado de IP
- ✓ TCP interpreta y completa la acción de control
- ✓ RFC 3168

ECN – IP

IP Header

```
0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|          DS FIELD, DSCP                 | ECN FIELD |
+-----+-----+-----+-----+-----+-----+-----+-----+
        DSCP: differentiated services codepoint
        ECN:  Explicit Congestion Notification

      +-----+-----+
      | ECN FIELD |
      +-----+-----+
        ECT   CE
         0     0          Not-ECT
         0     1          ECT(1)
         1     0          ECT(0)
         1     1          CE
```

ECN

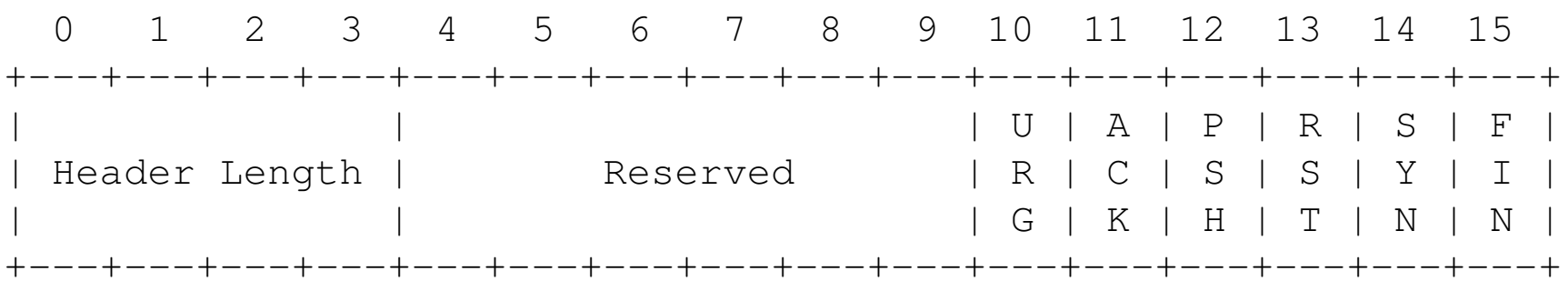
- ✓ Not-ECT indica que no se soporta ECN
- ✓ ECT(1)(01) y ECT(0)(10) son activados por los extremos
- ✓ ECT : ECN Capable Transport
- ✓ CE (11) es activado por el router para indicar congestión a los extremos
- ✓ CE :Congestion Experienced

ECN – Funcionalidad de TCP

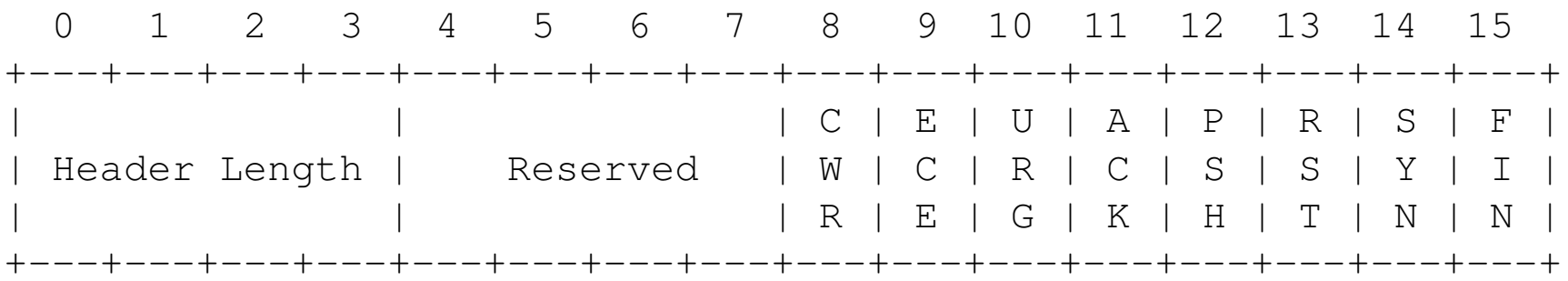
- ✓ Negociación en el set-up para acordar soporte de ECN
- ✓ Activar el bit ECN-Echo (ECE) por parte del receptor. Avisa al emisor la recepción de un CE.
- ✓ Activar el bit CWR (Congestion Window Reduced) para informar al receptor que la ventana de congestión se redujo a la mitad

Header TCP

Antes



Ahora



ECN – Secuencia de Eventos

- ✓ Se activa ECT “codepoint”en paquetes transmitidos por el emisor indicando que se soporta ECN.
- ✓ Un router ECN detecta congestión y detecta que el ECT “codepoint”está activo en el paquete que está pronto a marcar (antes lo descartaba).
- ✓ El router activa el CE y forwardea el paquete.

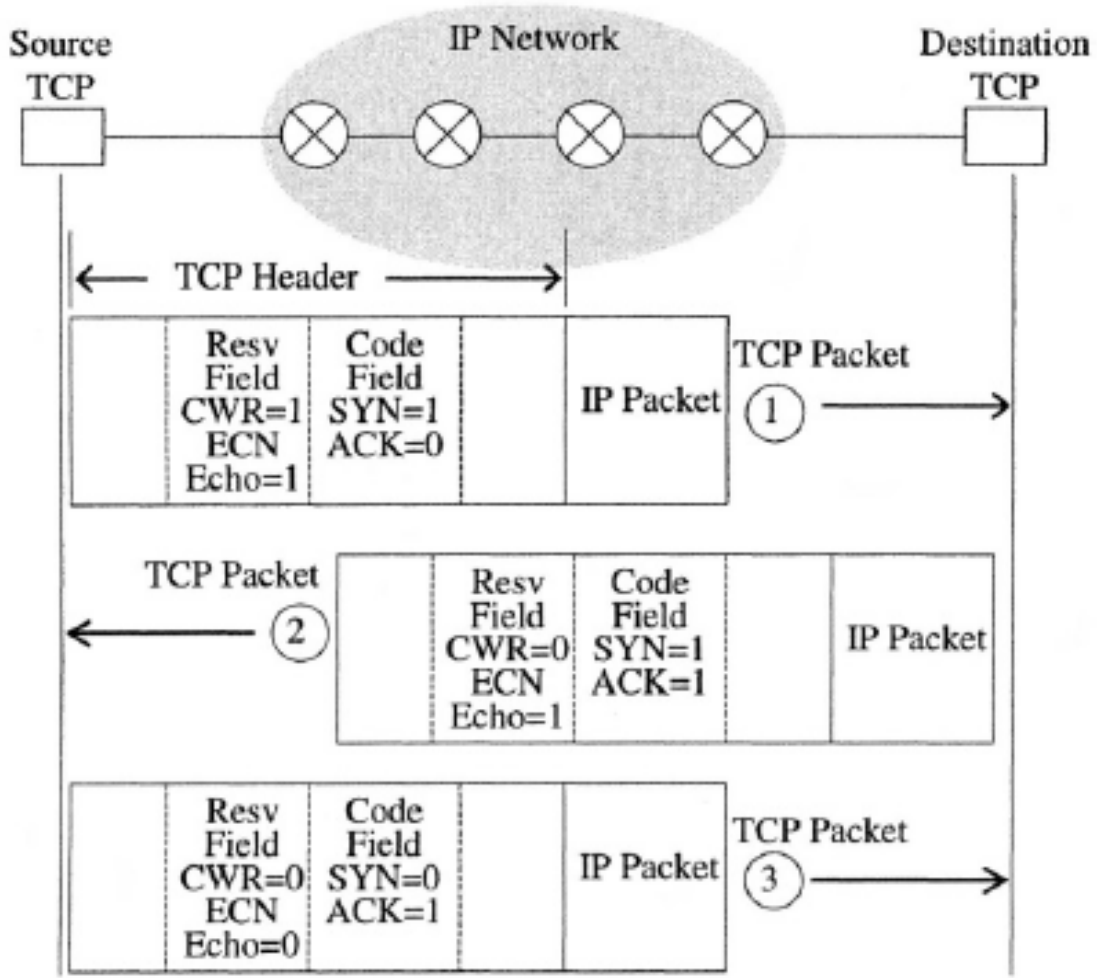
ECN – Secuencia de Eventos

- ✓ El receptor recibe el paquete con CE activo y activa el flag ECN-E en el header del próximo paquete TCP que enviará al emisor.
- ✓ El emisor recibe el paquete TCP con el bit de Flag ECN-Echo activo y actúa como si hubiera detectado un paquete perdido.
  Acorde con su mecanismo de control de congestión.
- ✓ El emisor activa el Flag de CWR en el header TCP del próximo paquete que envía al receptor.

ECN – Set-up

- ✓ Suponemos un set-up de A a B
- ✓ Un paquete de SYN con los flags ECE y CWR activos se lo llama paquete de SYN ECN set-up
- ✓ Un paquete de SYN con a lo sumo uno de los dos bits activos (ECE o CWR) lo llamamos de NO-SYN ECN set-up
- ✓ Un paquete de SYN-ACK con el bit de ECE activo pero no el de CWR se lo considera un paquete de SYN-ACK ECN set-up.
- ✓ Cualquier otra combinación será NO-SYN-ACK ECN set-up

ECN – Set-up


Source
TCP
IP Network
Destination
TCP
TCP Header
Resv Field CWR=1 ECN Echo=1
Code Field SYN=1 ACK=0
IP Packet
TCP Packet
1
TCP Packet
2
Resv Field CWR=0 ECN Echo=1
Code Field SYN=1 ACK=1
IP Packet
Resv Field CWR=0 ECN Echo=0
Code Field SYN=0 ACK=1
IP Packet
TCP Packet
3

Estamos en:

Características

- RFC 2309 – Reemplazada por RFC 7567(2015)
- Modelo de control de congestión basado en la red
- Controlar la longitud de la cola de salida de los datagramas
- Monitorear el estado de la cola y calcular la longitud promedio
- Marcar los datagramas según esa longitud promedio
- Completa el mecanismo ECN.
- Propone diversos algoritmos
- Ejemplo: RED (Random Early Detection)

RED – Algoritmo

Dos algoritmos

1. Longitud promedio de la cola
   - Filtro pasabajos con promedio ponderado exponencial
   - Determina el índice de rafagosidad que se admitirá en el “buffer ”
2. Marcado de paquetes
   - Determina la frecuencia de marcado de paquetes en función del nivel de congestión en la red
   - Se compara la longitud promedio con dos umbrales
     - Si está por debajo del menor no se marcan
     - Si está entre ambos se los marca con una probabilidad p_a
     - Si está por encima del mayor se lo marca
   - El marcado puede ser sólo eso o el descarte.

RED – Algoritmo – Longitud promedio

Initialization:

$avg \longleftarrow 0$

$count \longleftarrow -1$

for each packet arrival
calculate the new average queue size avg:
if the queue is nonempty

$avg \longleftarrow (1 - w_q)\, avg + w_q\, q$

else

$m \longleftarrow f(time - q_{time})$

$avg \longleftarrow (1 - w_q)^m\, avg$

RED – Algoritmo – Marcado de paquetes

if $min_{th} \leq avg < max_{th}$
increment count
calculate probability p_a:

$p_b \longleftarrow max_p\,(avg - min_{th})/(max_{th} - min_{th})$

$p_a \longleftarrow p_b/(1 - count \times p_b)$

with probability p_a:
mark the arriving packet

$count \longleftarrow 0$

else if $max_{th} \leq avg$
mark the arriving packet

$count \longleftarrow 0$

else $count \longleftarrow -1$
when queue becomes empty

$q_{time} \longleftarrow time$

Variaciones del marcado

- Medir la longitud de la cola en bytes en vez de paquetes
- La longitud refleja el retardo promedio en el router
- Se deben hacer cambios:

$$p_b \longleftarrow max_p\,(avg - min_{th})/(max_{th} - min_{th})$$

$$p_b \longleftarrow p_b\,PacketSize/MaximumPacketSize$$

$$p_a \longleftarrow p_b/(1 - count \times p_b)$$

- Es más probable que se marque un paquete FTP que uno de Telenet

RED-Algoritmo-Parámetros

Variables:

avg: longitud promedio de la cola

q_{time}: Comienzo del tiempo ocioso de la cola

count: Paquetes desde la última marcación

Constantes:

w_q: peso de la cola

min_{th}: Umbral mínimo

max_{th}: Umbral máximo de la cola

max_p: valor máximo para p_b

Otros:

p_a: probabilidad de marcado

q: longitud actual de la cola

time: tiempo actual

$f(t)$: función lineal del tiempo t

RED - Probabilidad de marcado

$$p_b \longleftarrow max_p(avg - min_{th})/(max_{th} - min_{th})$$

Consideramos la *VA* , X el número de paquetes que llegan entre dos marcas sucesivas
Si se marcan los paquetes con prob p_b entonces:

$$Prob[X = n] = (1 - p_b)^{n-1} p_b$$

Resulta una distribución geométrica, generándose marcas en forma de ráfagas y con intervalos largos entre marcas.
Mejor alternativa es que X sea uniformemente aleatoria en el intervalo $\{1, 2, \ldots, 1/p_b\}$
Entonces:

$$p_a = \frac{p_b}{1 - count \times p_b}$$

RED - Implementación

- ✓ *avg* se calcula en cada arribo
- ✓ Se debe recalcular cuando llega a una cola vacía
- ✓ Se estima un nro de paquetes que debería haber llegado en el intervalo en que la cola estuvo vacía.

$$m = \frac{time - q_{time}}{s}$$

donde s es el tiempo de inserción de un paquete pequeño

$$avg \longleftarrow (1 - w_q)^m avg$$

RED - Implementación

- ✓ Si *avg* está dentro de las cotas se debe marcar con probabilidad
- ✓ ¿Qué paquete se marca?
- ✓ $R = Random[0, 1]$
- ✓ Se marca si:

$$R < \frac{p_b}{1 - count \times p_b}$$