Control de Congestión en TCP

Teoría de la Comunicaciones

17 de Octubre de 2017

Request for comments

The RFC series contains **technical** and **organizational** documents about the Internet, including the **specifications** and policy documents produced by four streams: the Internet Engineering Task Force (IETF), the Internet Research Task Force (IRTF), the Internet Architecture Board (IAB), and Independent Submissions.

Abstract

This document defines TCP's four intertwined congestion control algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery. In addition, the document specifies how TCP should begin transmission after a relatively long idle period, as well as discussing various acknowledgment generation methods. This document obsoletes RFC 2581.

Definiciones

- **Segment:** "Paquete" TCP/IP
- **Sender Maximum Segment Size** (SMSS): Máximo payload puede tener cada segmento
- **Receiver Maximum Segment Size** (RMSS): Máximo que va a recibir el receptor
- **Full-Sized Segment:** Un segmento con la cantidad máxima (SMSS) de bytes

Definiciones

- **Reciever Window** (RWND): Última Advertised Window recibida.
- **Congestion Window** (CWND): Estimación de la congestión.

- **Initial Window** (IW): Valor de CWND después del handshake.
- **Loss Window** (LW): Valor de CWND después de un timeout.
- **Restart Window** (RW): Valor de CWND después de un período idle.

Definiciones

- **Flight Size:** (LastByteSent - LastByteACKed)
- **Slow Start Threshold** (SSTHRESH): Umbral que define si se usa SS o CA.
- **Duplicate Acknowledgement:** Un ACK es duplicado, si:
  1. El receptor del ACK tiene datos en vuelo.
  2. El ACK no tiene datos.
  3. No Hay SYN ni FIN.
  4. El número de ACK es igual al máximo recibido.
  5. La advertised window es igual a la última recibida.

La ventana de congestión

CWND =

"Es una estimación de la cantidad de información que puedo meter en la red sin que se vea afectada su performance."

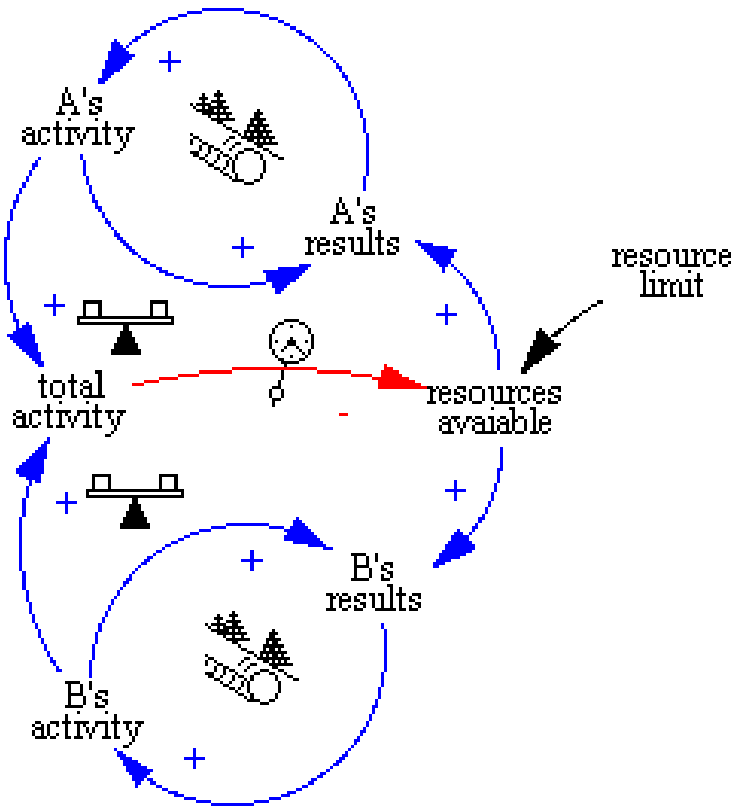
MaxWindow = Min(RWND, CWND)

★ "Define quién lidera la ventana."

EffectiveWindow = MaxWindow - (LastByteSent - LastByteAcked)

"Cuántos bytes puedo despachar."

Marco teórico



A activity $\Rightarrow$ + A results

+ A results $\Rightarrow$ + A activity

B activity $\Rightarrow$ + B results

+ B results $\Rightarrow$ + B activity

★ Total activity = B activity + A activity
★ Recursos Disponibles -= Total activity

mientras Recursos Disponibles $>$ 0:
+ B results;
+ A results;

Comportamiento de una fuente ...

... con retroalimentación binaria ($+$, $-$)

Incremento:
Mientras no haya signos de congestión ($+$), aumentar la ventana.

Decremento:
Ante un indicio de congestión ($-$), decrementar la ventana.

$$cwnd_{t+1} = \begin{cases} a_i + b_i * cwnd_t & \text{si } + \\ a_d + b_d * cwnd_t & \text{si } - \end{cases}$$

Comportamientos

$$cwnd_{t+1} = \begin{cases} a_i + b_i * cwnd_t & \text{si } + \\ a_d + b_d * cwnd_t & \text{si } - \end{cases}$$

- $a_i = 0; a_d = 0; b_i > 1; 0 < b_d < 1$

  Multiplicative Increase, Multiplicative Decrease (MIMD)

- $a_i > 0; a_d < 0; b_i = 1; b_d = 1$

  Additive Increase, Additive Decrease (AIAD)

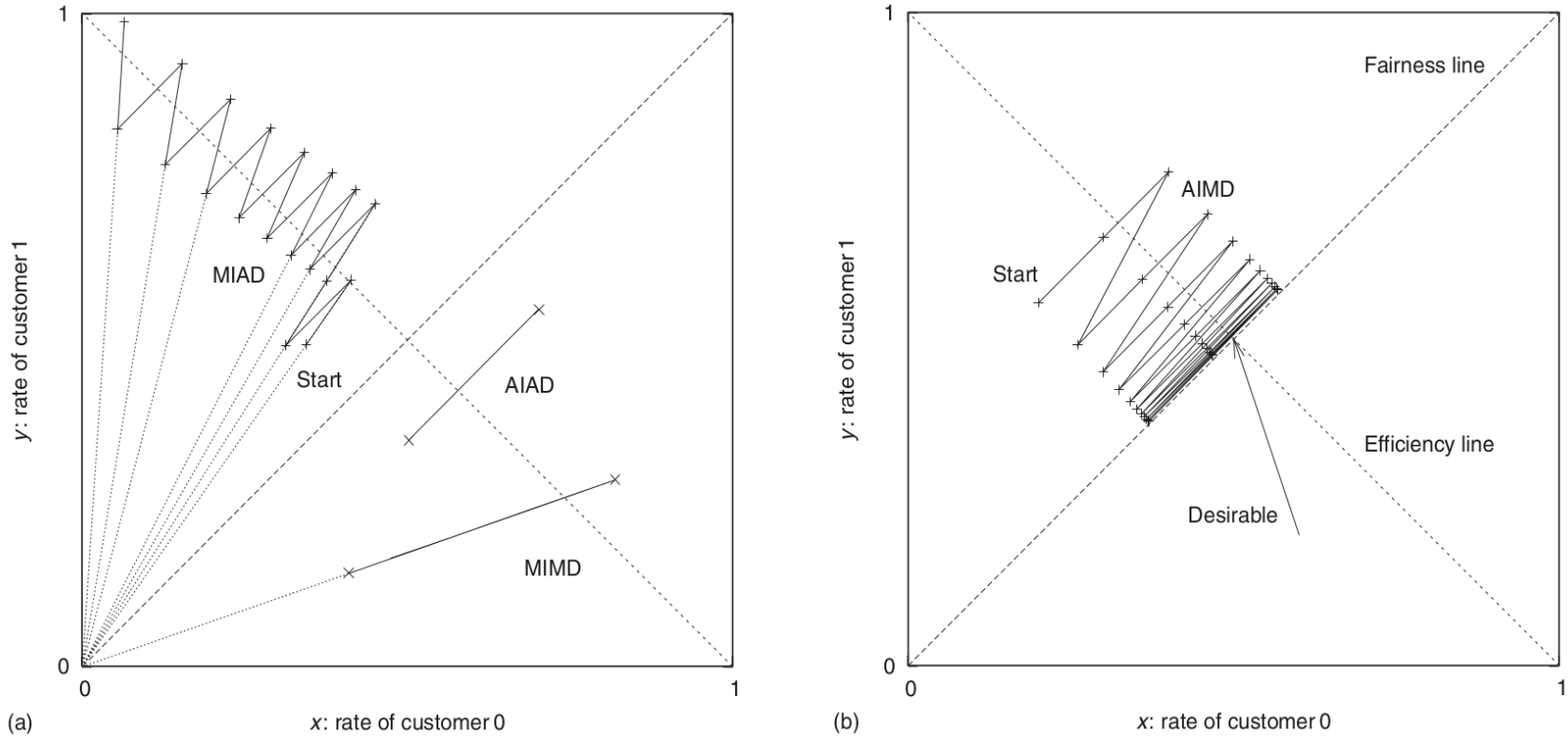
★ $a_i > 0; a_d = 0; b_i = 1; 0 < b_d < 1$

  Additive Increase, Multiplicative Decrease (AIMD)

- $a_i = 0; a_d < 0; b_i > 1; b_d = 1$

  Multiplicative Increase, Additive Decrease (MIAD)

Comportamientos



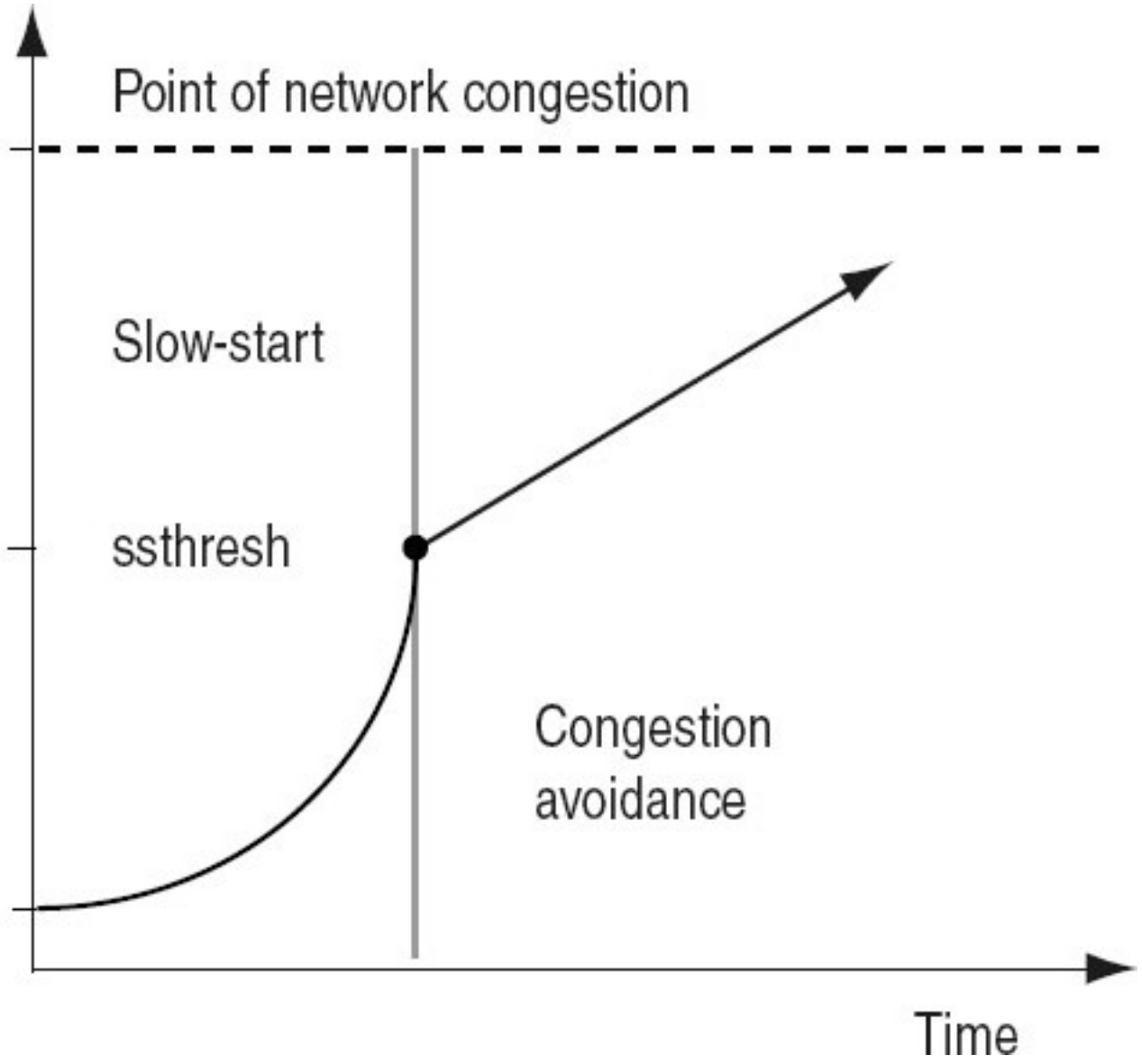
Fairness line: Los recursos consumidos por el cliente 0 deberían ser iguales a los consumidos por el cliente 1

Efficiency line: La suma total de los recursos consumidos por los clientes no debe superar un cierto limite.

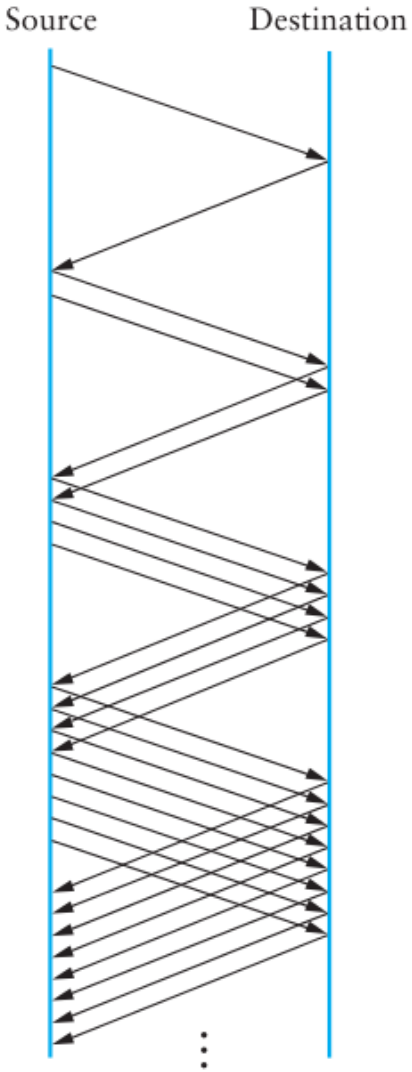
Algoritmos de Control de Congestión

- **Slow Start:** Comenzar enviando pocos datos
- **Congestion Avoidance:** Aumentar un SMSS por RTT
- **Fast Retransmit / Fast Recovery:** No esperar al time out, para recuperarse de un error.

Algoritmos: Slow Start + Congestion Avoidance



Algoritmos: Slow Start



Inicialmente:

- ★ $CWND = IW = 2 * SMSS$
- ★ $SSTHRESH = alta(max\ advertised\ window)$

si $CWND < SSTHRESH$:

Hacer $CWND+ = min(N, SMSS)$ **por cada ACK**

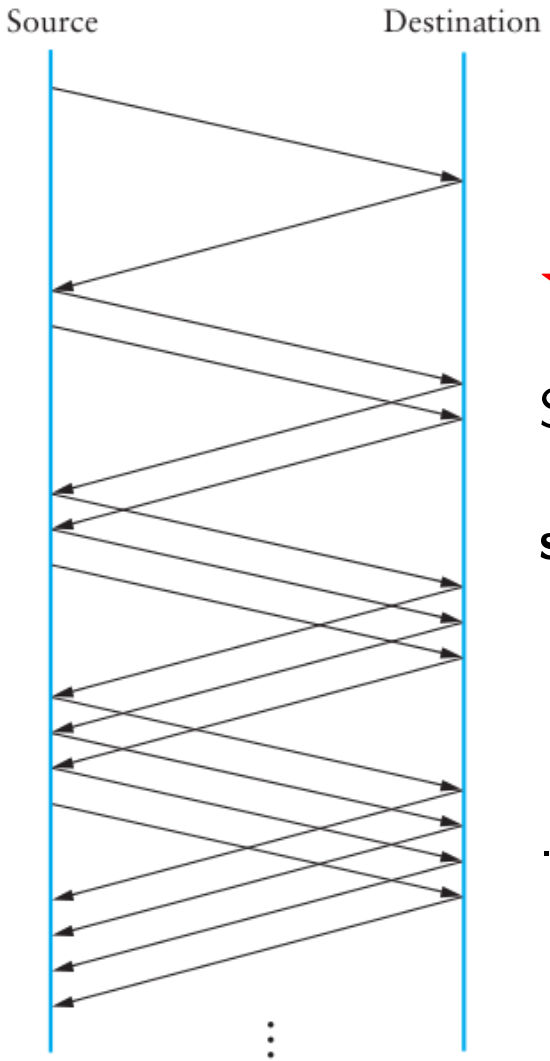
N es la cantidad de bytes reconocidos por el ACK

"Se usa la llegada de ACKs como retroalimentación positiva"

Ejercicio

Considere el efecto de usar Slow Start en una conexión TCP recién establecida ($IW = 2 * SMSS$, $SSTHRESH = 64KB$, $SMSS = 2KB$), que tiene un RTT de 10 mseg y sin congestión ni errores presentes en la red. Si la RWND es de 24 KB, ¿Cuánto tiempo transcurre antes de que pueda ser enviada la primera ventana de recepción llena?

Algoritmos: Congestion Avoidance



★ Aumentar la *CWND* en un *SMSS* por *RTT*

Se puede lograr con ...

si $CWND > SSTHRESH$:

Hacer $CWND+ = SMSS * SMSS/CWND$
por cada ACK

... aunque es mas conservador

Algoritmos: Time out

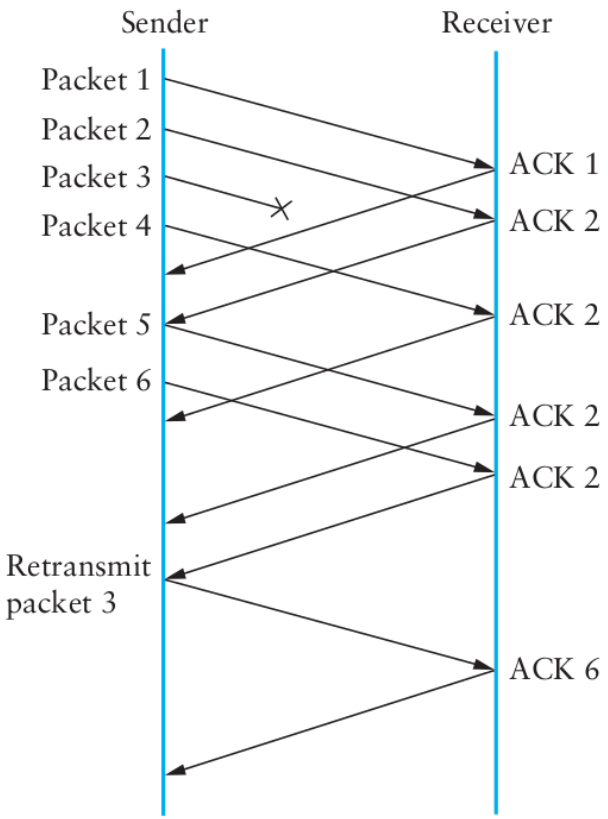
Ante un *time-out* se cambian los valores a:

- ★ $CWND = LW(1SMSS)$
- ★ $SSTHRESH = max(FlightSize/2, 2 * SMSS)$

$\Rightarrow$ **Se comienza de nuevo con Slow Start**

"Se usa el time-out como retroalimentación negativa"

Algoritmos: Fast Recovery / Fast Retransmit



Al 3er ACK duplicado, el sender debería retransmitir el segmento perdido.

- ★ SSTHRESH = max(FlightSize/2, 2 * SMSS)
- ★ CWND = SSTHRESH + 3 * SMSS

mientras no se reconozcan nuevo datos:
Hacer CWND += SMSS
por cada ACK Duplicado

Termina cuando llega el primer ACK que reconoce nuevos datos.

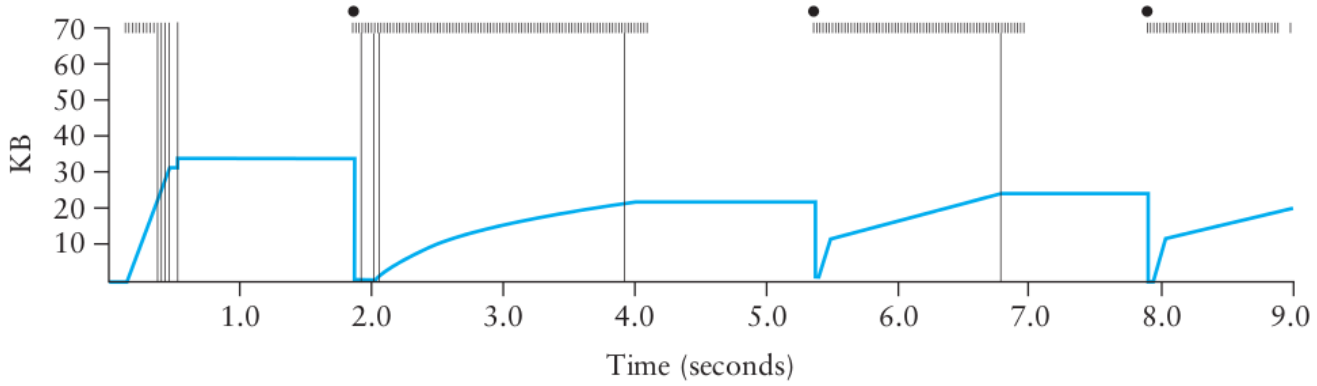
- ★ CWND = SSTHRESH

⇒ **Se continúa con Congestion Avoidance**

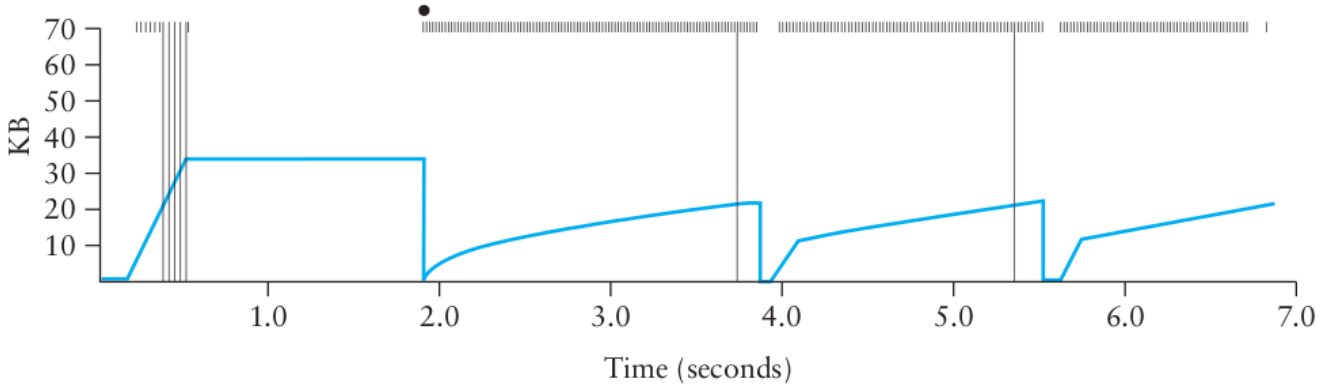
"Se usa el 3er ACK duplicado como retroalimentación negativa"

Algoritmos: Fast Recovery / Fast Retransmit

Sin FR/FR



Con FR. FastRecovery?



Reiniciando conexiones idle.

"Si una conexión no envía datos no tiene retroalimentaciones."

Si no hay actividad por mas de un RTO

★ CWND = RW = min(IW,cwnd)

Generando reconocimientos.

"Se puede esperar antes de enviar un ACK:"

★ A lo sumo 500ms o 2*SMSS bytes sin reconocer

Ejercicio

En una conexión recién establecida con RTT=200ms, el host receptor siempre anuncia una *AdvertisedWindow* de 16KB. La red está cargada al punto que si una ráfaga fuera de 16KB o mas, se perderían todos los segmentos de la misma.

a. ¿Cuánto vale la CWND luego de enviar un archivo de 40KB?

b. 3 segundos después del envió del archivo, se envía otro archivo de 30KB ¿Cuánto tiempo tarda?

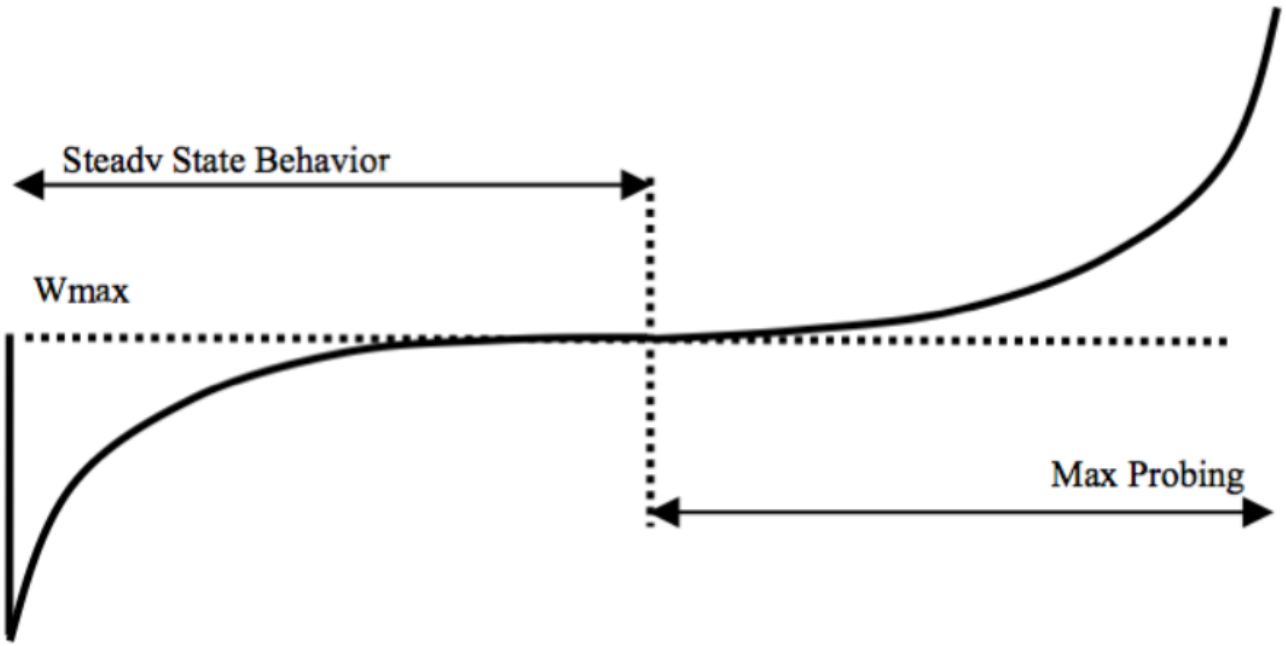
CUBIC: Linux kernels 2.6.19

Llegada de ACK: $W_{cubic} = C(t - K)^3 + W_{max}$

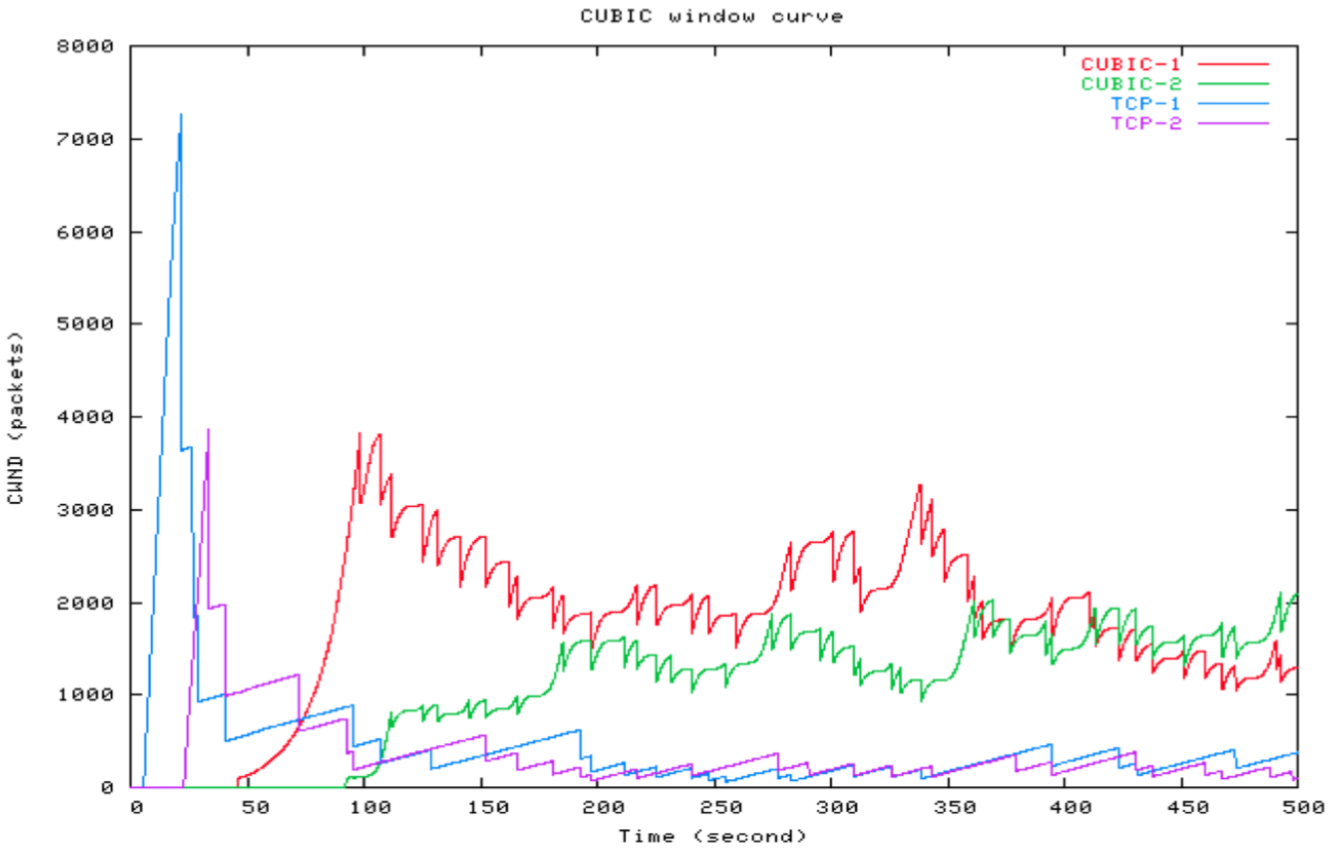
Perdida de paquete: $W_{max} = \beta W_{max}$

siendo $K = \sqrt[3]{W_{max}\beta C}$ con $\beta = 0{,}8$ y $C = 0{,}4$

y t el tiempo desde la ultima pérdida de paquete



Simulación: 2 Flujos CUBIC y 2 Flujos TCP compitiendo en una red con un cuello de botella de 500Mbps y un RTT de 100ms



Ejercicio Parcial 1c16

Conexión TCP recién establecida de 50ms mediada por un router que también conecta otras redes y descarta ráfagas $>=$ a 20KB. El emisor tiene que transmitir por horas y el receptor siempre anuncia una ventana de 28KB.

a. ¿Cuánto tiempo tarda la conexión en alcanzar el momento a partir del cual SSTHRESH no cambia más y cuál es su valor?

b. Si host receptor anunciara una ventana de 18KB, ¿tardaría más o menos tiempo la transferencia?