

Capítulo 3

Control de Congestión

(Referencia: Computer Networks. Kurose-Ross)

Andres Arcia Moret
andres@arcia.net.ve

Principios del Control de la Congestión

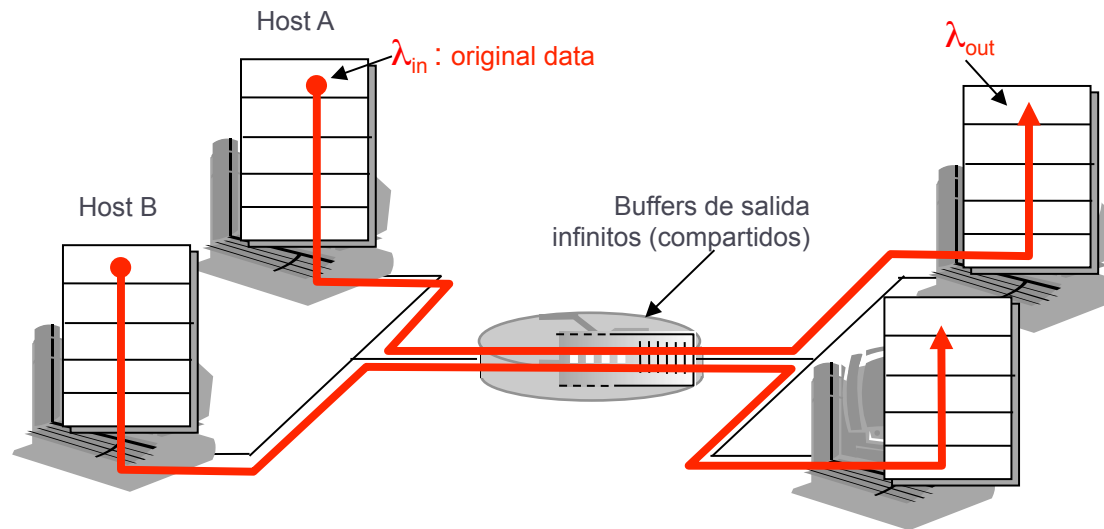
- ▶ Retransmisión trata pérdidas pero **NO** la causa de las pérdidas:
 - ▶ *Muchas fuentes envían data a la vez y de forma rápida.*

Causas y Costos de la Congestión

- ▶ Pero, ¿Qué sucede cuando hay varios transmitiendo a una tasa alta?
- ▶ Cada nodo tiene su visión (descentralizada) basada en feed-back implícito.
 - ▶ **ACK**: indica que la red no está congestionada.
 - ▶ **Segmento perdido**: Se perdió porque la red está congestionada, entonces hay que disminuir la tasa de emisión

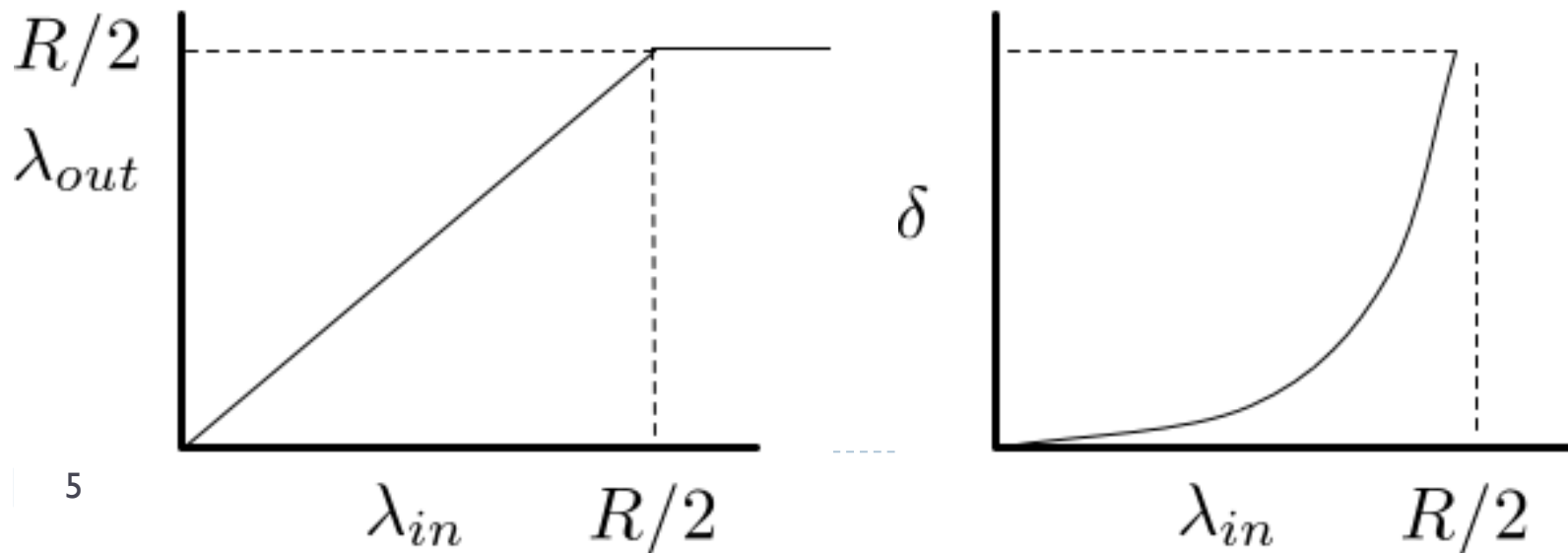
Escenario 1: 2 emisores, 1 enrutador con buffer infinito.

- La aplicación de **Host A y Host B** manda data a capa transporte en bytes/sec a **Host C y D** respectivamente.
- Protocolo de transporte es simple, la data se encapsula y se envía (hay recuperación de errores) —> no hay control de flujo ni de congestión.
- R es la capacidad del cuello de botella.
- Como el buffer es infinito, los paquetes de exceso se guardan. —> no hay retransmisión.



Performance del Sistema

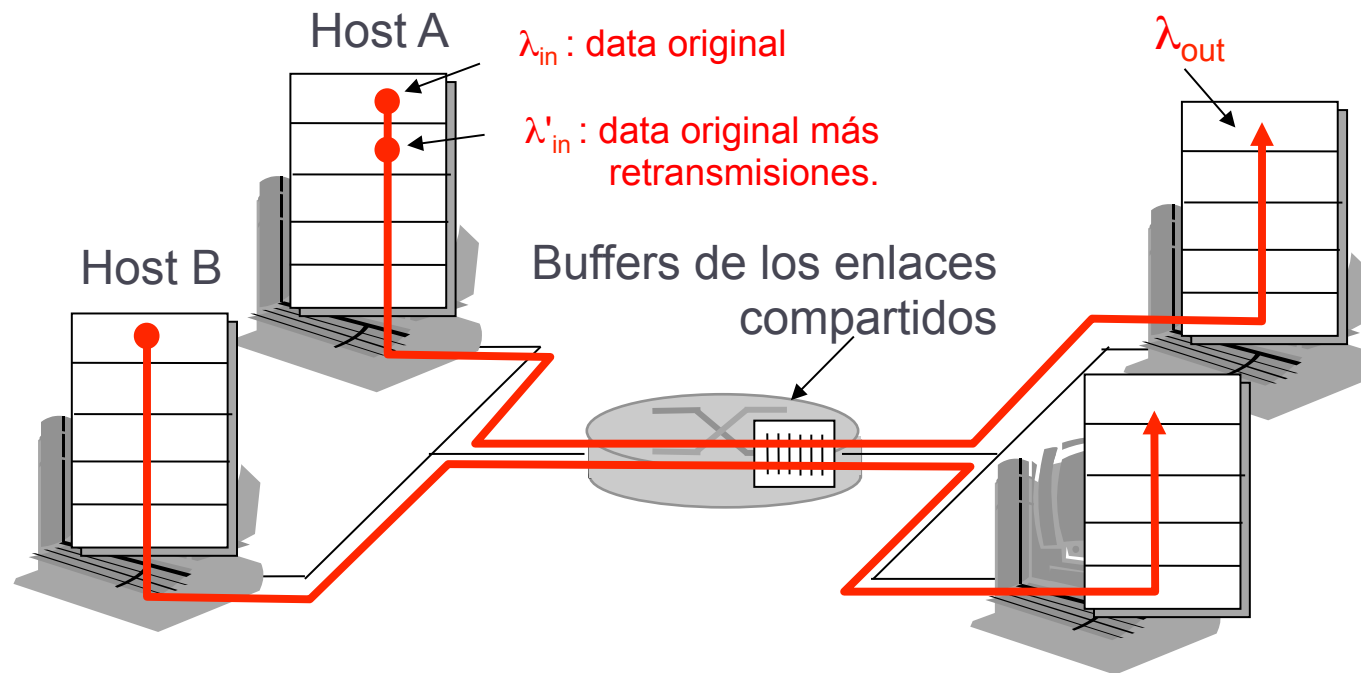
- ▶ El emisor nunca ve un rendimiento mejor que $R/2$
- ▶ La espera promedio incrementa cuando la conexión excede la capacidad del enlace.
- ▶ Aún en este escenario idealista, la congestión hace estragos. El delay aumenta cuando la tasa de emisión alcanza la capacidad del enlace.



Escenario 2: Dos emisores, un ruteador con buffers finitos

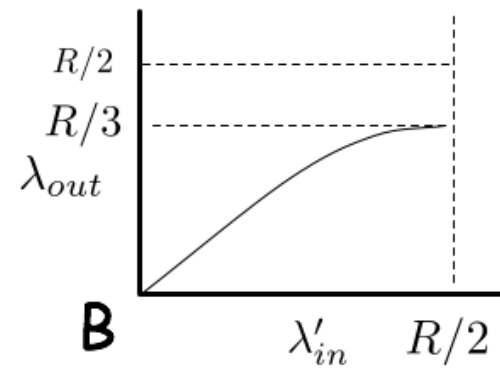
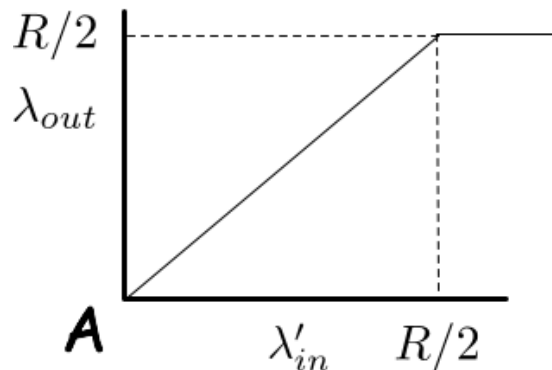
- ▶ **Modificación pequeña del Escenario I**
 - ▶ Buffer *finito* → hay pérdidas
 - ▶ La aplicación envía data a λ_{in} (en bps)
 - ▶ La tasa de envío de la capa transporte es λ'_{in} (data +retransmisión) → llamada “carga ofrecida a la red”
 - ▶ $\lambda'_{in} \leq \lambda_{in}$
 - ▶ Depende de cómo se efectúan las retransmisiones
 - ▶ Si el que envía “supiese” el estado del buffer podría enviar paquetes y no tener pérdidas

Escenario 2: Topología



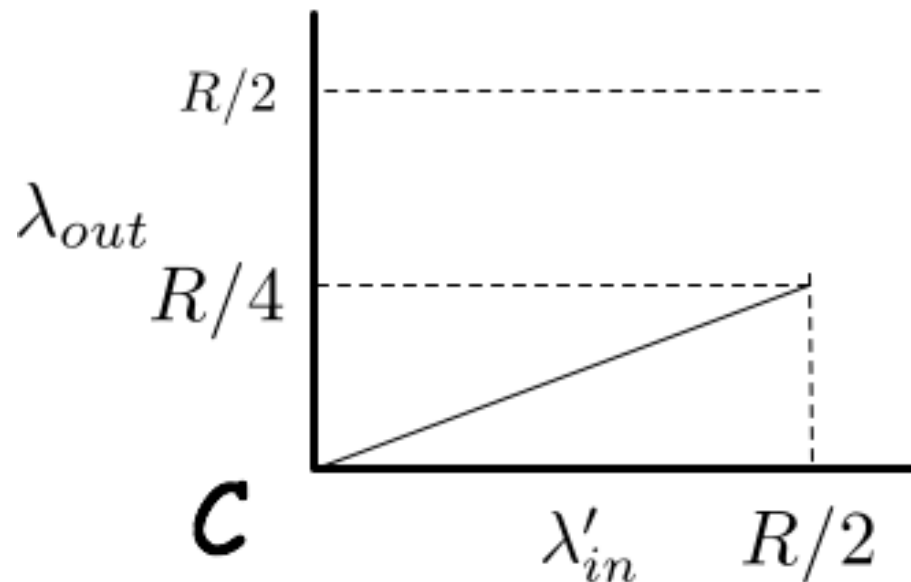
Escenario 2: continuación...

- ▶ Considere que hay retransmisión cuando se pierden los paquetes (el valor timeout es suficientemente grande).
- ▶ El emisor nunca supera los $R/2$ de tasa de emisión (puesto que hay pérdidas).
- ▶ Observe en “B” que, de $0.5R$, 0.33 es data original y 0.17 es retransmisión (puede ser peor, si se retransmite 1:2).
 - ▶ Congestión \rightarrow retransmisión de paquetes perdidos.



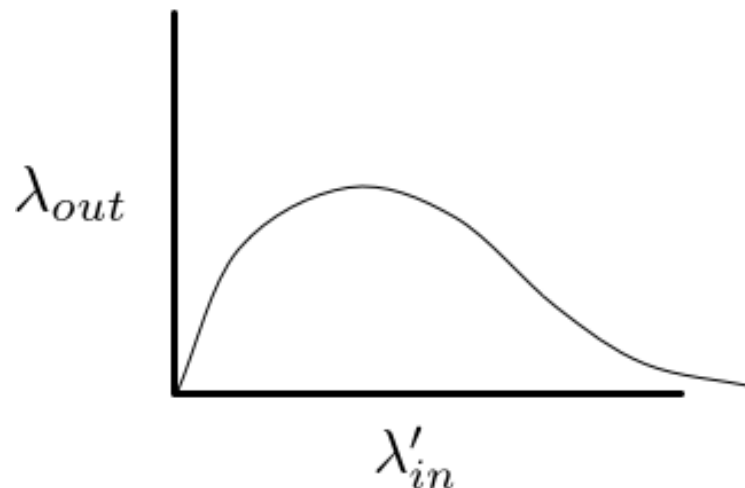
Escenario 2: continuación

- ▶ Los paquetes detenidos (*no perdidos*) generan *timeout*.
- ▶ Trabajo del ruteador perdido en la retransmisión.
- ▶ Costo de la Congestión: Hace perder recursos para retransmisiones innecesarias. Vea en “C” lo que sucede si cada paquete es reenviado 2 veces.

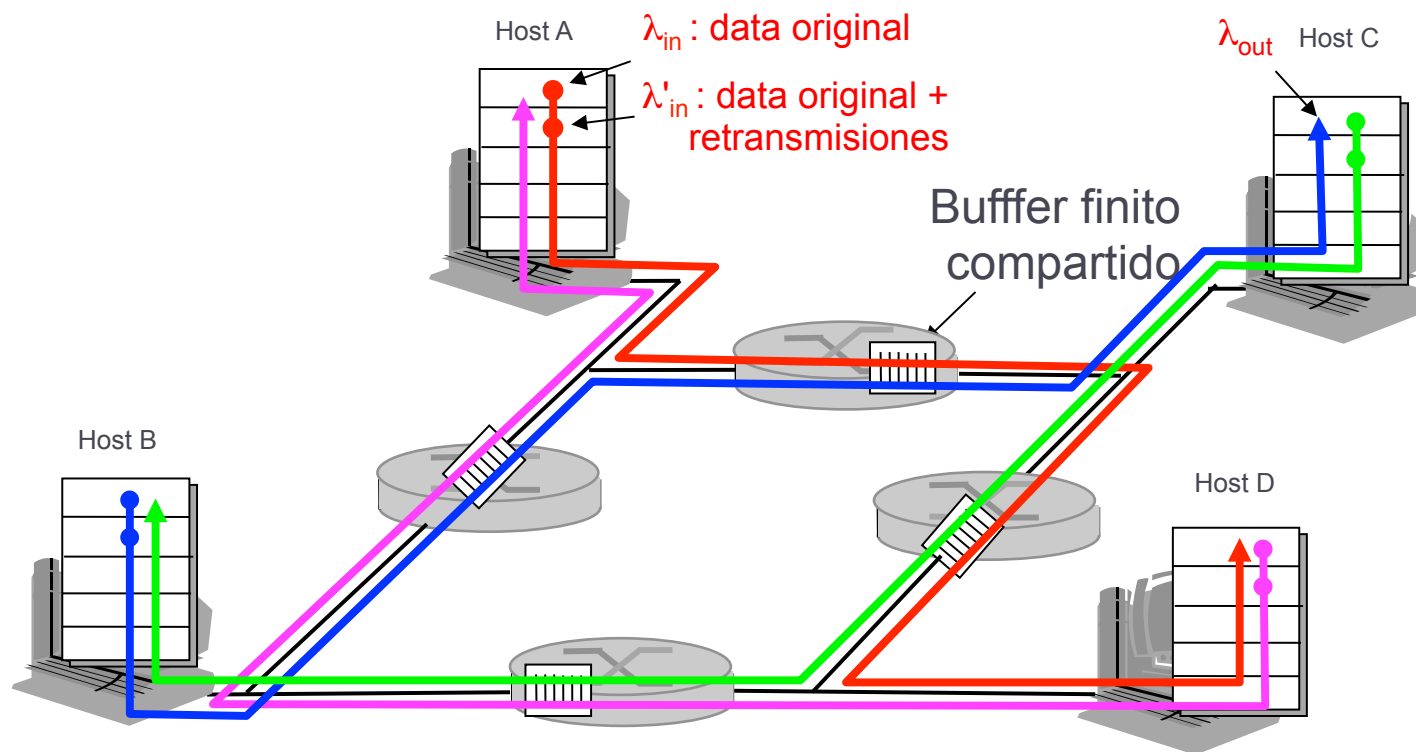


Escenario 3: 4 emisores, enrutadores con buffers finitos y caminos multi-saltos.

- ▶ Los hosts transmiten en caminos solapados de 2 ruteadores (A-C comparten R1, B-D comparten R2).
- ▶ Se usa timeout/retransmisión para la confiabilidad.
- ▶ Para tasas de transmisión muy pequeñas no hay overflows —> throughput es la carga ofrecida (λ'_{in} incrementa λ_{out} también).



Escenario 3: topología.

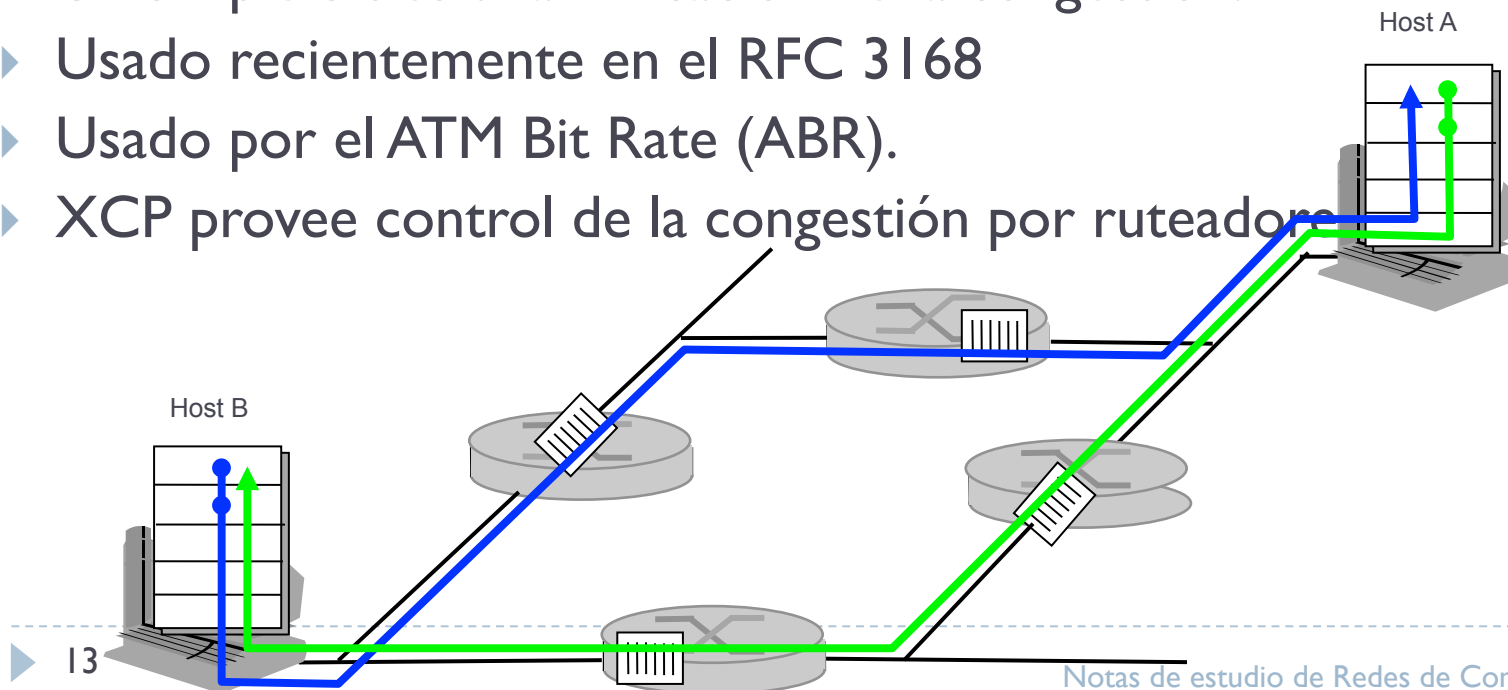


Escenario 3: continuación

- ▶ ¿Qué sucede cuando λ_{in} crece mucho?
 - ▶ Cada vez que crece pasa menos tráfico útil.
 - ▶ La fuente con tráfico más cercano al ruteador compartido se lo apropia más rápidamente.
- ▶ Costo de la congestión: un paquete botado en un ruteador, *¡hace perder recursos a ruteadores previos!*

Aproximaciones al control de la congestión.

- ▶ **De extremo a extremo**
 - ▶ Sin apoyo de la capa de red → tradicional
 - ▶ Con apoyo de la capa de red → moderno (ECN en IP)
 - ▶ Una pérdida TCP es un signo de congestión.
- ▶ **Asistido por la red**
 - ▶ Un simple bit es una indicación de la congestión.
 - ▶ Usado recientemente en el RFC 3168
 - ▶ Usado por el ATM Bit Rate (ABR).
 - ▶ XCP provee control de la congestión por ruteadores.



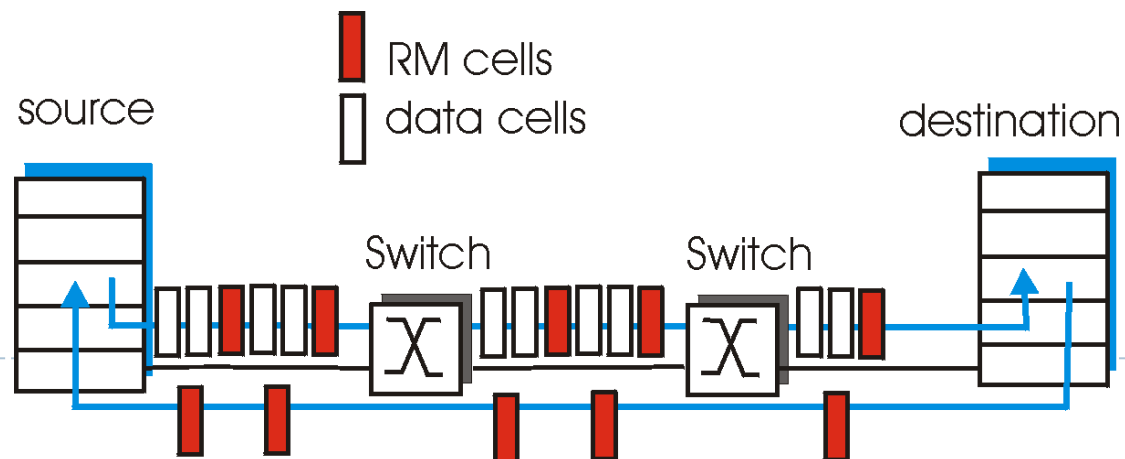
Caso de Estudio:

Control de Congestión en Redes ATM.

- ▶ **Control de Congestión asistido por la red.**
 - ▶ Señala al emisor explícitamente cuando reducir la tasa de emisión en período de congestión.
- ▶ **Toma el ancho de banda remanente y lo asigna a TCP.**
 - ▶ Es un servicio elastico
- ▶ **Marco de Trabajo (framework):**
 - ▶ Paquetes de Manejo de Recursos que circulan constantemente.
- ▶ **Basado en la tasa de transferencia (calculada por el que envía). ¿Qué conducta garantiza la administración del AB?**
 - ▶ Si hay ancho de banda → use todo el que pueda
 - ▶ Si **no** hay ancho de banda → dar garantía mínima

Manejo de Recursos (por celdas RM)

- ▶ Las celdas especiales (RM) son enviadas por el emisor
- ▶ Mecanismos para la señalización de la congestión :
 - ▶ Notificación explícita en el sentido de ida (forward) EFCI. Cuando es recibido por el destino, se hace eco.
 - ▶ Bits de notificación de congestión y parada inmediata por congestión (CI y NI). CI → congestión severa (congestion indication). NI → congestión intermedia (non increase).
 - ▶ Tasa específica (ER) → dice en 2 bytes cuanto debe ser la tasa de emisión.



Control de Congestion de TCP

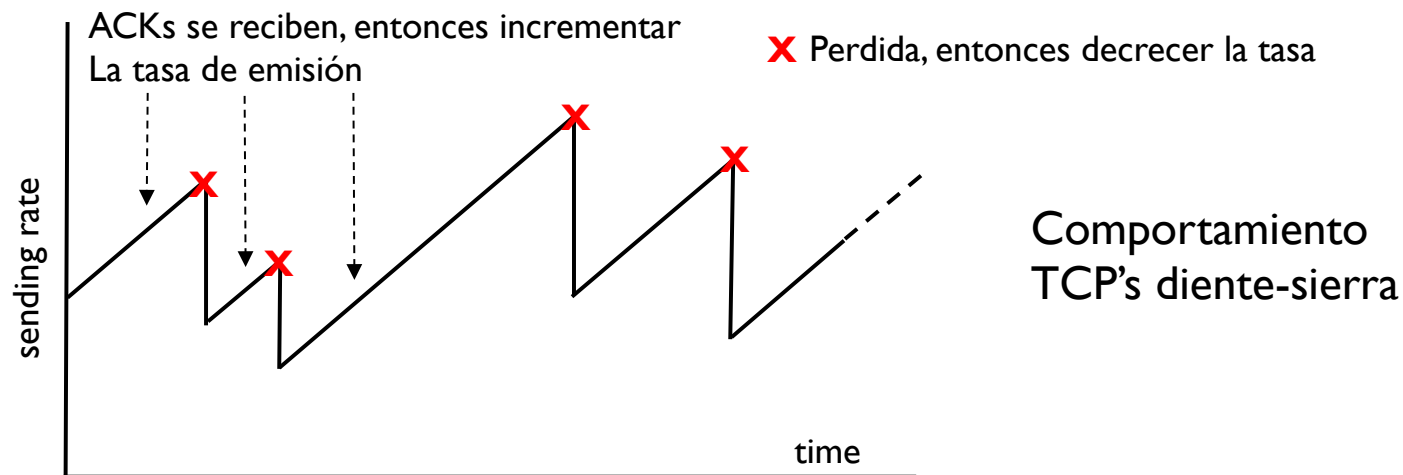
- ▶ Utiliza el control de congestión end-to-end
- ▶ Ajusta la tasa de emisión de acuerdo a la congestión percibida.
 - ▶ Incrementa si hay poca congestión
 - ▶ Decrementa si hay pérdidas
- ▶ ¿Cómo limitar la tasa de envío?
- ▶ ¿Cómo se percibe que hay congestión?
- ▶ ¿Cómo debe reaccionar el emisor (algorítmicamente) cuando se percibe congestión de extremo a extremo?

Control de Congestión de TCP

- ▶ **Objetivo:** TCP debe transmitir lo más rápido posible SIN congestionar la red.
 - ▶ Pero, ¿Cómo?
- ▶ Hay un feedback implícito para TCP
 - ▶ Recepción de ACK: la red puede aguantar más tráfico
 - ▶ Perdida de un segmento: hay congestión

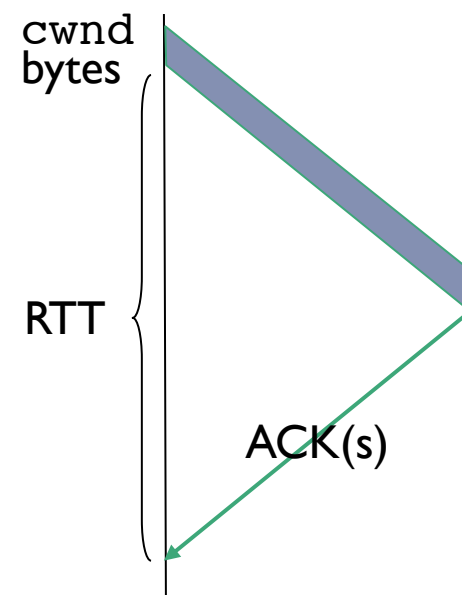
Prueba del Ancho de Banda

- ▶ “Prueba del Ancho de Banda”: A la recepción de ACKs, y hasta que se produzca una pérdida, la tasa de emisión se incrementa.



Mas detalles del control de congestión

- ▶ El emisor limita su tasa de emisión de los **bytes no confirmados** en el tubo:
 - ▶ $\text{LastByteSent} - \text{LastByteAked} \leq \text{cwnd}$
 - ▶ Cwnd: no es lo mismo que rwnd (ve por que?)
 - ▶ Emisor se limita a $\min(\text{cwnd}, \text{rwnd})$
- ▶ Aproximadamente
 - ▶ Tasa = cwnd / RTT bytes/sec
- ▶ La ventana de congestión es función de la congestión percibida de la red.



Control de Congestión: más detalles.

Reducción de la cwnd: ante una pérdida

- ❑ timeout: no hay respuesta del servidor
 - Cortar `cwnd` to 1
- ❑ 3 ACKs duplicados: hay segmentos que llegan (retransmisión rápida)
 - Cortar `cwnd` a la mitad, menos agresivo que un timeout

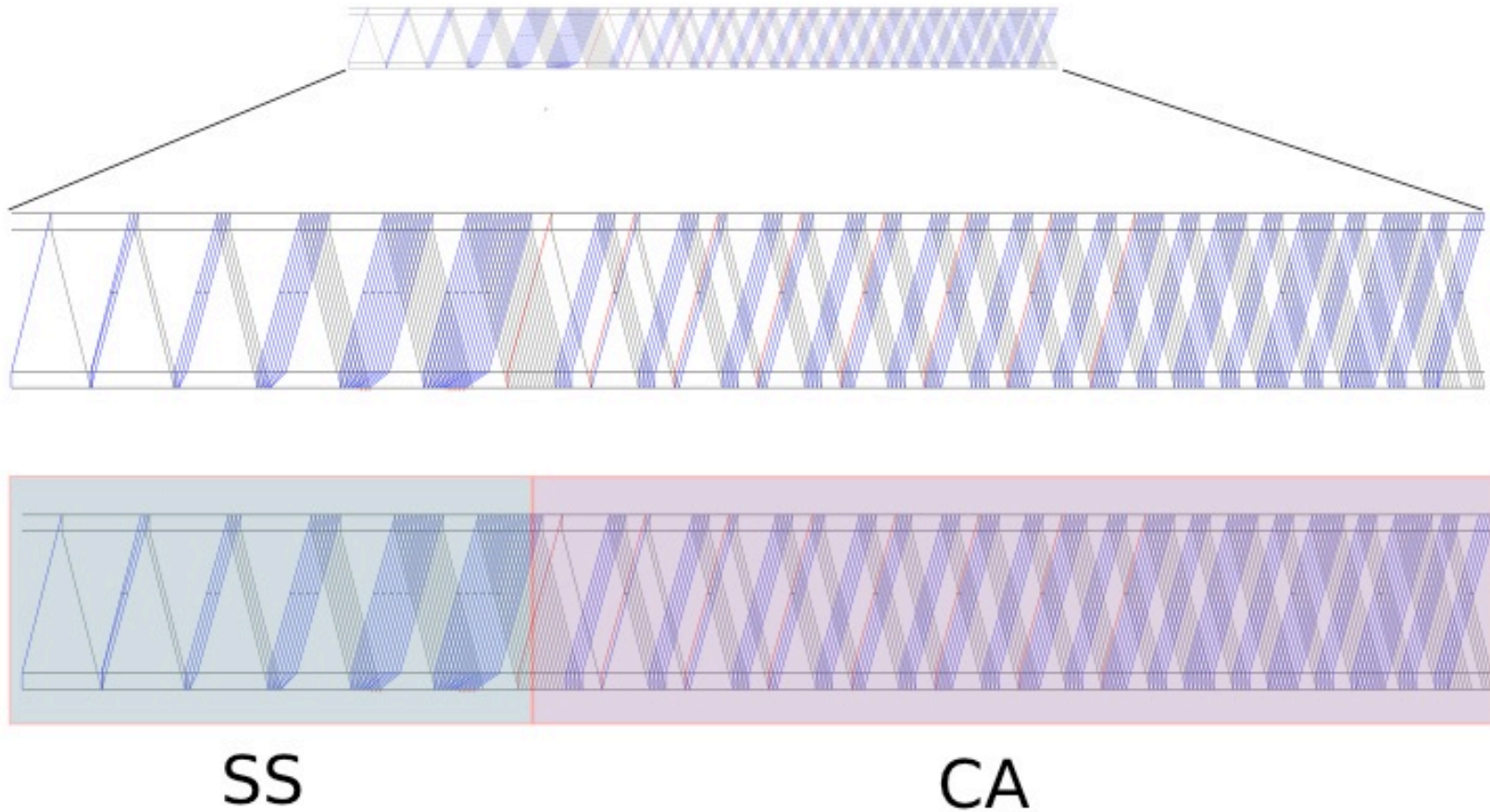
Recepción de ACK: incremento de la cwnd

- ❑ Slowstart:
 - Incrementa rápidamente al inicio de la conexión o después de un timeout
- ❑ Congestion Avoidance:
 - Incremento pseudo-lineal.

TCP en Slow Start

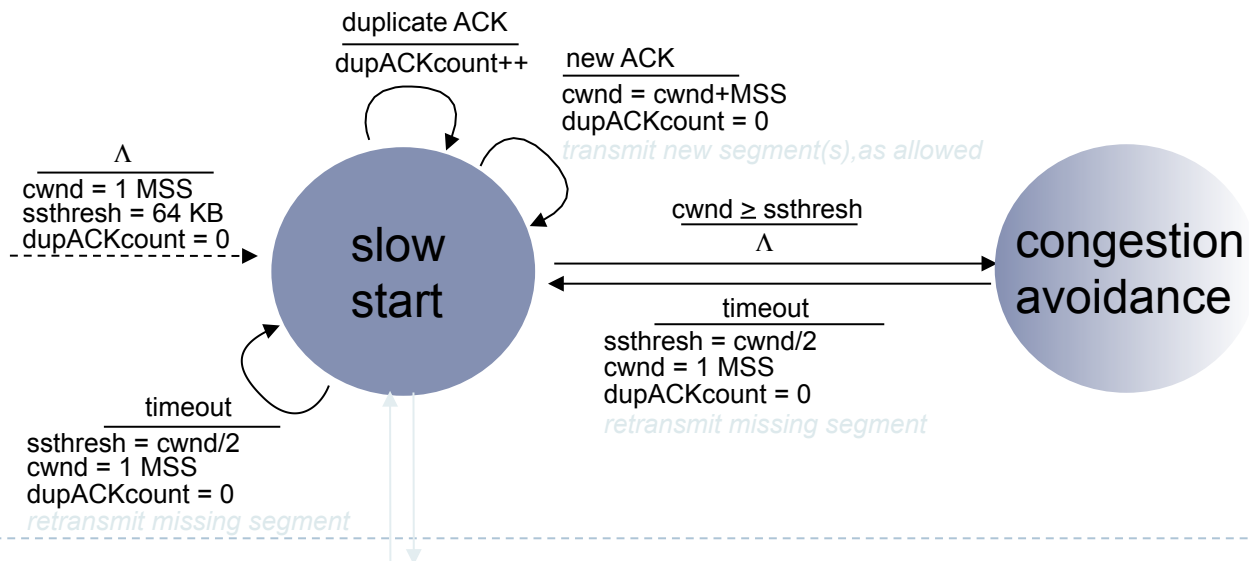
- ❑ Al comienzo, **cwnd** = 1 MSS
 - Ej: MSS = 500 bytes & RTT = 200 msec
 - Tasa inicial = 20 kbps
- ❑ BW disponible puede ser \gg MSS/RTT
 - Buenas noticias si se quiere acelerar rápidamente.
- ❑ Se incrementa exponencialmente hasta una pérdida o hasta que se llegue al límite
 - Se dobla la cwnd cada RTT
 - $Cwnd = cwnd + 1$ por cada ACK.

Slow Start & Congestion Avoidance



Entrada y salida de Slow Start

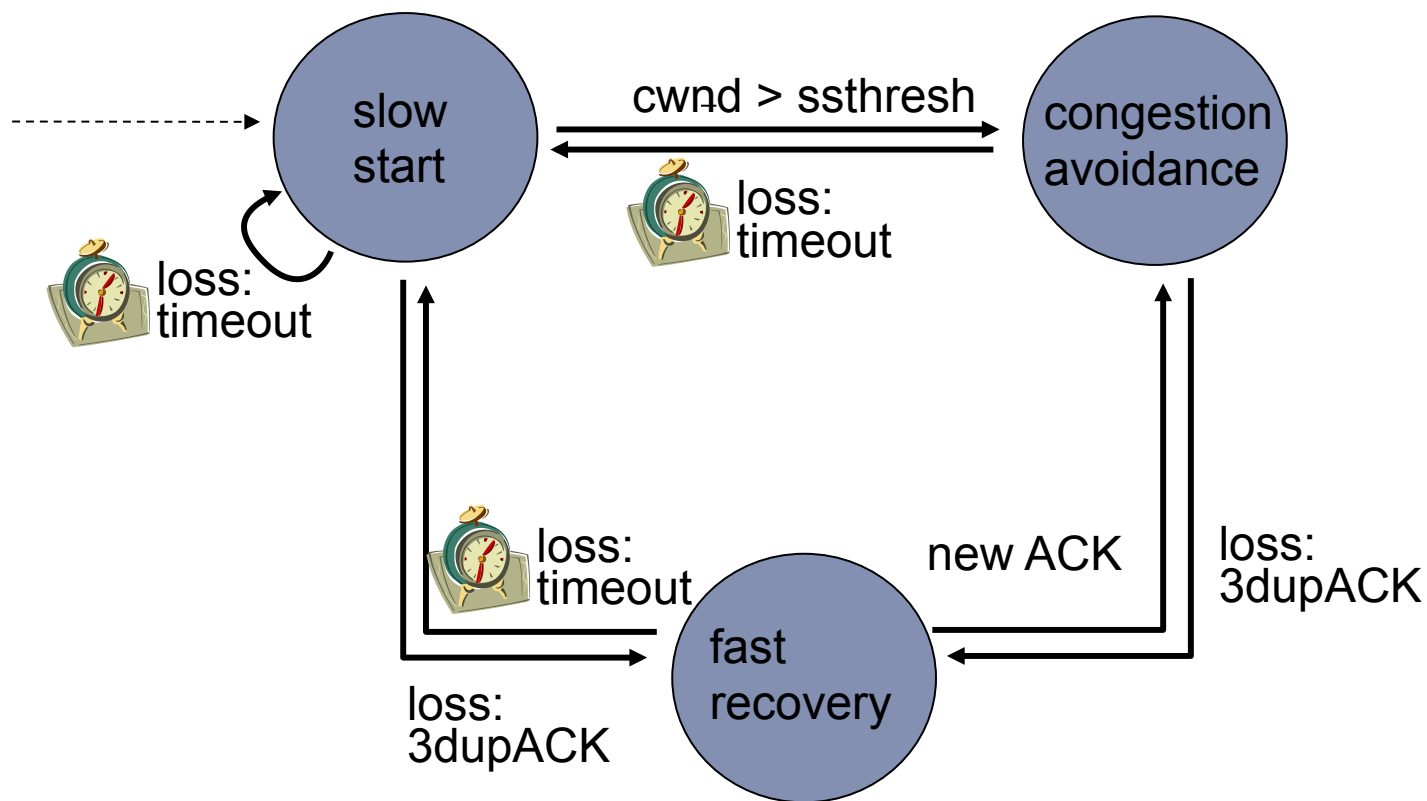
- ▶ Ssthresh: limite de la cwnd mantenido por TCP
- ▶ Cuando hay perdida: $ssthresh = cwnd/2$
 - ▶ Recuerda la mitad de la tasa TCP cuando la última congestion ocurrió
- ▶ Cuando $cwnd \geq ssthresh$: se pasa de SS a CA.



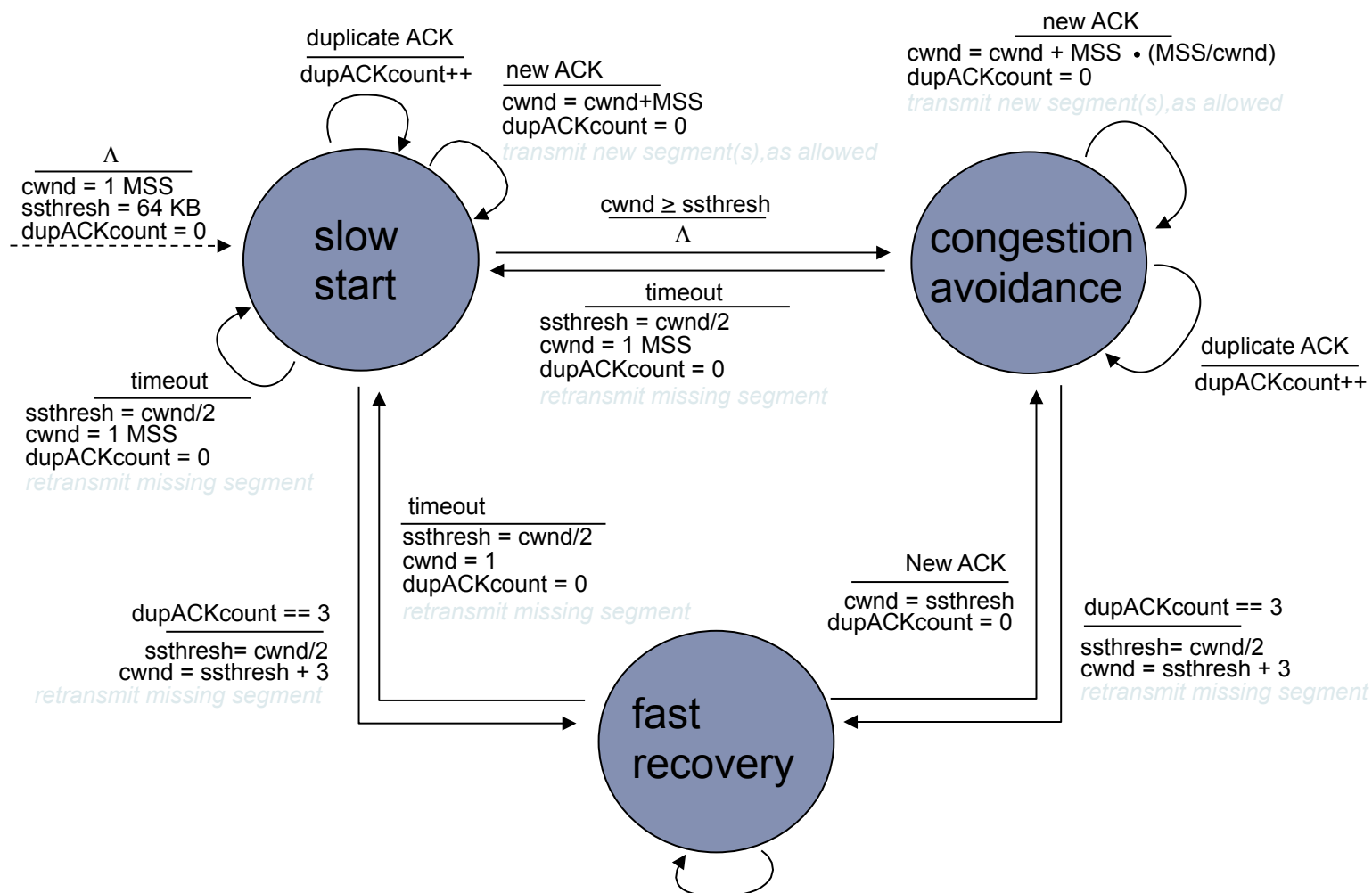
Congestion Avoidance

- ▶ Cuando $cwnd > ssthresh$ hay que crecer linealmente
 - ▶ Incrementar $cwnd$ de 1 MSS cada RTT
 - ▶ Se aproxima a la congestión más lentamente que en Slow Start
 - ▶ Algoritmo: $cwnd += MSS/cwnd$ por cada ACK
- ▶ Principio del control de la congestión:
 - ▶ AIMD: additive increase multiplicative decrease
 - ▶ ACK: incrementar **cwnd** por 1 MSS cada RTT. (additive Increase)
 - ▶ **Perdida**: picar la ventana a la mitad (Multiplicative Decrease)

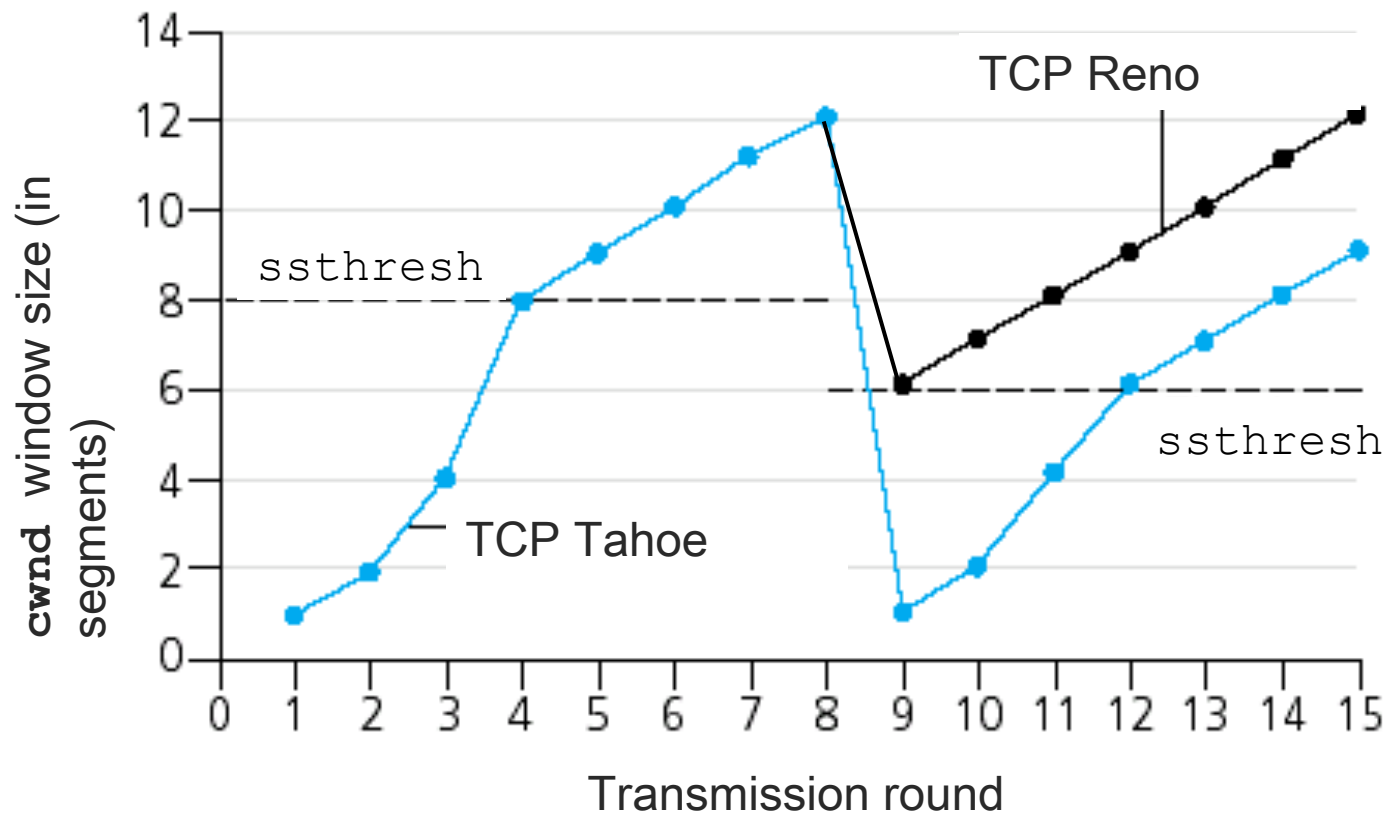
Máquina de Estados para Control de Congestión



TCP: Máquina de Estado Detallada.



Versiones Populares de TCP



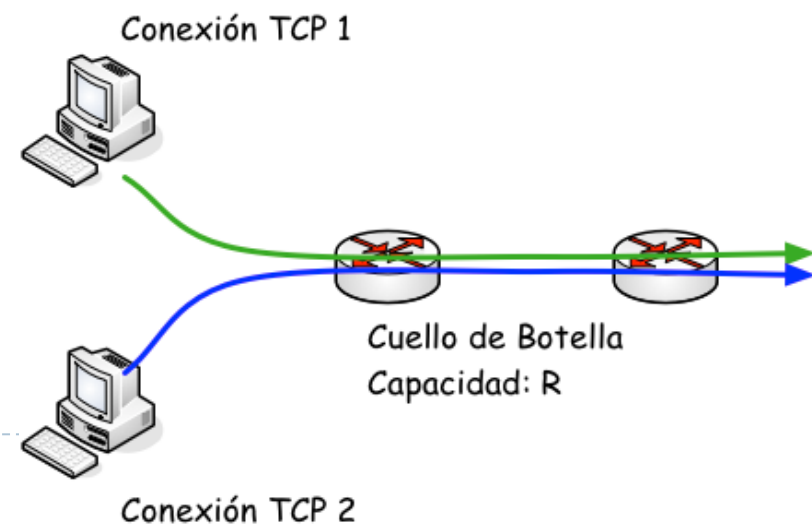
Rendimiento de TCP

- ▶ ¿Cuál es el rendimiento promedio de TCP en función del tamaño de la ventana y del RTT?
 - ▶ Hay que ignorar slow start
- ▶ Si el tamaño de la ventana es W :
 - ▶ Cuando la ventana se ha abierto hasta W , el rendimiento es W/RTT
 - ▶ Cuando hay una pérdida la ventana será $W/2$ y la tasa de rendimiento $W/(2RTT)$
 - ▶ Rendimiento promedio:

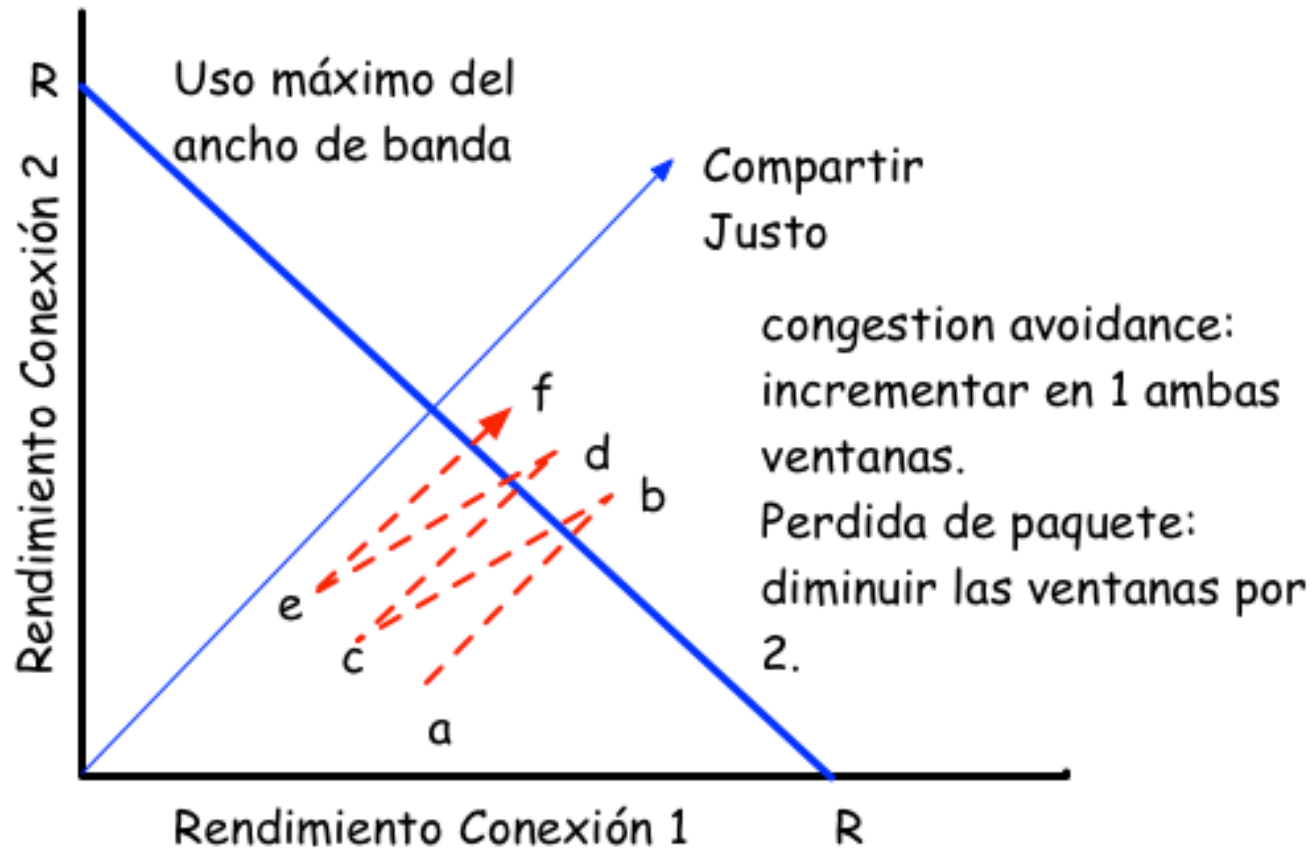
$$\frac{W/RTT + W/2RTT}{2} = 0.75W/RTT$$

Justicia en el uso del Ancho de Banda

- ▶ Considere K conexiones TCP cada una con un camino de extremo a extremo diferente, con un cuello de botella en común
- ▶ Tasa de transmisión del cuello de botella es R
- ▶ Se dice que hay justicia si para cada conexión se tiene un Ancho de Banda de R/K
 - ▶ Se logra gracias al algoritmo AIMD



¿Cómo funciona la justicia en el uso del ancho de banda?



Más sobre Justicia

▶ Justicia y UDP

- ▶ Aplicaciones multimedia casi no usan UDP
- ▶ No les conviene la tasa de emisión impedida por el control de congestión
- ▶ Se prefiere enviar audio/video a tasa constante y tolerar pérdidas de paquete
- ▶ ¡Pueden impedir el paso de tráfico TCP!

▶ Conexiones TCP paralelas:

- ▶ Nada impide que una aplicación abra más de una conexión, p.e., los navegadores.
 - ▶ Suponga un enlace que recibe 9 conexiones TCP.
 - ▶ Si una aplicación establece 1 conexión TCP: $tasa = R/10$
 - ▶ Si una aplicación establece 9 conexiones TCP: $tasa = R/2$