Capítulo 3: Capa Transporte:Control de congestión en TCP

ELO322: Redes de Computadores

Agustín J. González

Este material está basado en:

- Material de apoyo al texto *Computer Networking: A Top Down Approach*. Jim Kurose, Keith Ross.

Capítulo 3: Continuación

- 3.1 Servicios de la capa transporte
- 3.2 Multiplexing y demultiplexing
- 3.3 Transporte sin conexión: UDP
- 3.4 Principios de transferencia confiable de datos
- 3.5 Transporte orientado a la conexión: TCP
  - Estructura de un segmento
  - Transferencia confiable de datos
  - Control de flujo
  - Administración de conexión
- 3.6 Principios del control de congestión
- 3.7 Control de congestión en TCP

Control de Congestión en TCP

- Usa control extremo a extremo (sin asistencia de la red)
- Tx limita su ventana de transmisión:

$$Bytes\ en\ tránsito: LastByteSent - LastByteAcked \leq min\{CongWin, RcvWindow\}$$

- Si Rx tiene espacio, se tiene:

$$\text{tasa}_{\text{Aprox}} = \frac{\text{CongWin}}{\text{RTT}} \text{ [Bytes/sec]}$$

- **CongWin** es dinámica y función de la congestión percibida de la red
- **RcvWindow** es el número de bytes que el Rx puede recibir en su buffer, aquí lo suponemos grande y no limita la tasa de envío.

¿Cómo el Tx percibe la congestión?

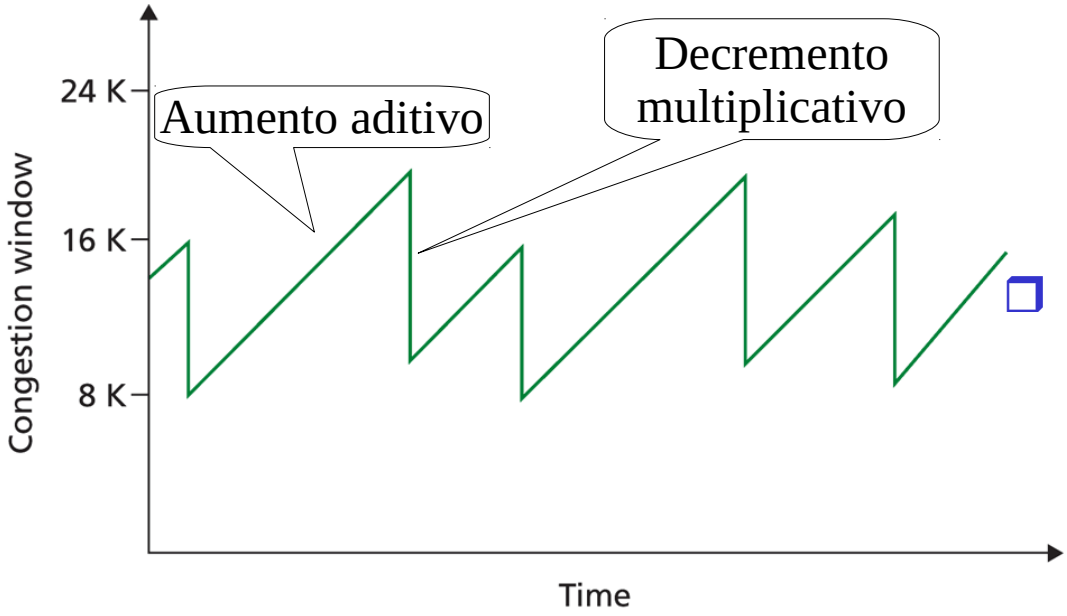
- Evento de pérdida = timeout ó 3 acks duplicados
- Tx TCP reduce tasa (**CongWin**) después de un evento de pérdida

Hay tres mecanismos:

- AIMD (Additive-Increase, Multiplicative-Decrease)
- "Partida lenta"
- Conservativo después de evento de timeout

TCP AIMD (Additive-Increase, Multiplicative-Decrease)

- Decrecimiento multiplicativo: reducir **CongWin** a la mitad luego de pérdida



- Aumento aditivo: aumenta **CongWin** en 1 MSS cada RTT en ausencia de pérdida. En algunas implementaciones **CongWin** incrementa en MSSx(MSS/CongWin) por cada ACK recibido.
- MSS (Maximum Segment Size) es la máxima cantidad de datos que se envía en cada segmento sin fragmentarse.

Figure 3.50 ♦ Additive-increase, multiplicative-decrease congestion control

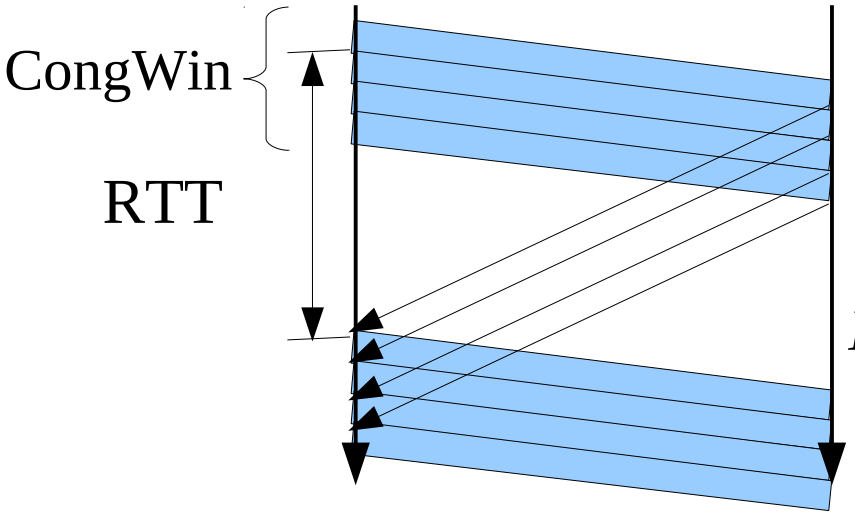
Indique qué protocolo usa el tamaño de segmento máximo (MSS, Maximum Segment Size). ¿A qué corresponde?

?

- El MSS es usado por TCP. Corresponde al MTU (Maximum Transmission Unit) más pequeño en la ruta de la fuente al destino. Usando segmentos de tamaño MSS, TCP asegura que sus paquetes no serán fragmentados.

Aumento aditivo

- La idea es aumentar un MSS luego de un RTT.
- Podemos aproximarnos aumentando la CongWin cada vez que se recibe un ACK de manera que al completar 1 RTT hayamos sumado un MSS.



- Se envía como máximo CongWin bytes y esperamos por el acuse de recibo.

$$NumSegmentos = NumAkcs = \frac{CongWin}{MSS}$$

$$Incr. = \frac{MSS}{NumAkcs} = \frac{MSS}{\frac{CongWin}{MSS}} = \frac{MSS * MSS}{CongWin}$$

- Incr. : Incremento por cada ACK

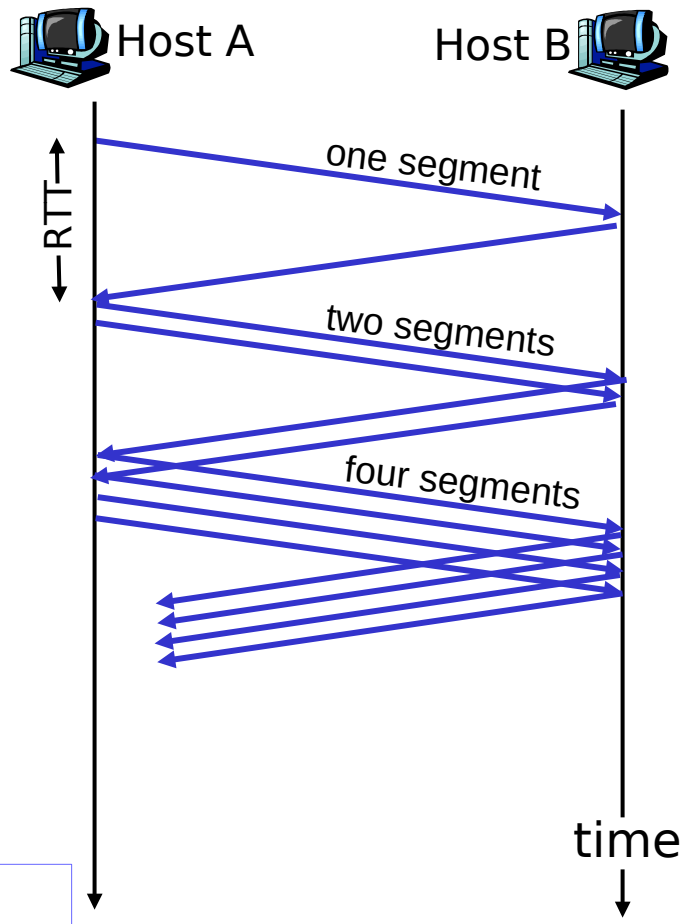
Si hay ACK retardados, el incremento debe ser mayor

Partida lenta en TCP (slow start)

- Cuando la conexión comienza, **`CongWin`** = 1 MSS
  - Ejemplo: MSS = 500 bytes & RTT = 200 msec
  - Tasa inicial = 20 kbps
- Pero la tasa disponible de la ruta puede ser >> MSS/RTT
  - Es deseable aumentar tasa rápidamente hasta una tasa respetable
- IDEA: Cuando la conexión comienza, aumentar tasa exponencialmente rápido hasta llegar a un umbral predefinido o tener 3 ACKs duplicados
- Se le llama Slow Start porque parte desde tasa muy abaja.

Partida Lenta en TCP (más detalles)

- Cuando la conexión comienza, aumentar tasa exponencialmente hasta umbral o primera pérdida:
  - Idea: Duplicar **`CongWin`** cada RTT
  - Lo logra incrementando **`CongWin`** en 1 MSS por cada ACK recibido
- Resumen: tasa inicial es lenta pero se acelera exponencialmente rápido

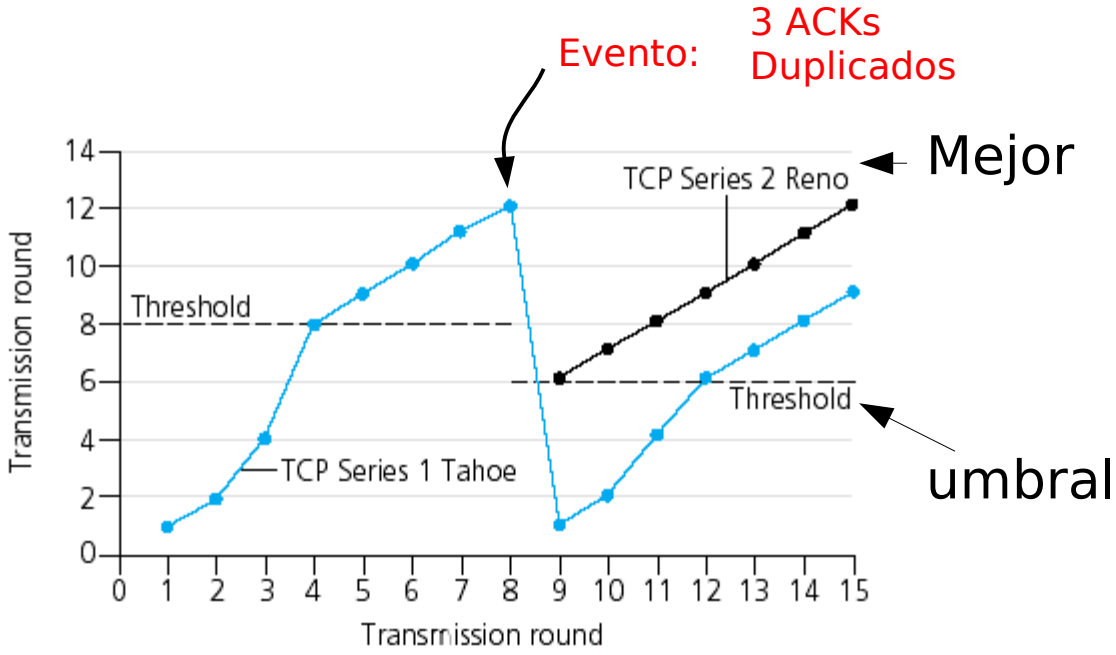


Todo esto pasa en la medida que Tx tenga muchos datos que enviar.

Reacción ante eventos de pérdida

Q: ¿Cuándo debería cambiar el aumento de exponencial a lineal?

A: Un buen criterio es: Cuando **CongWin** llega a 1/2 de su valor antes del timeout.



Implementación:

- Umbral variable (variable threshold)
- Ante evento de pérdidas, el umbral (threshold) es fijado en 1/2 de CongWin justo antes de la pérdida

Tahoe: primera versión de control de congestión en TCP. No distinguía entre timeout o ACK duplicados.

Reno: versión siguiente en TCP. Sí distingue timeout de ACK duplicados. Hay otras versiones.

Reacción ante eventos de timeout (Reno)

- Después de 3 ACKs duplicados:
  - **CongWin** baja a la mitad
  - Luego la ventana crece linealmente
- Después de un timeout:
  - **CongWin** es fijada en 1 MSS;
  - Luego la ventana crece exponencialmente hasta un umbral (mitad de ventana antes del timeout), luego crece linealmente

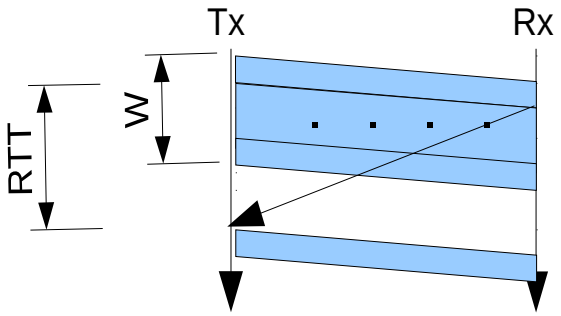
Filosofía:

- 3 ACKs duplicados indican que la red es capaz de transportar algunos segmentos (solo llegan fuera de orden en el Rx). Se perdió uno pero llegaron los otros y por eso tenemos ACKs duplicados
- timeout antes de 3 duplicados es “más alarmante” (no llegan paquetes!)

En ausencia de errores y cuando el buffer de recepción es muy grande, la tasa promedio de transferencia de TCP durante un RTT se puede aproximar por (Tamaño de Ventana de congestión)/RTT. Muestre un diagrama que explique la deducción de esta expresión. ¿Es esta expresión válida para todo valor de “Ventana de Congestión”? Explique.

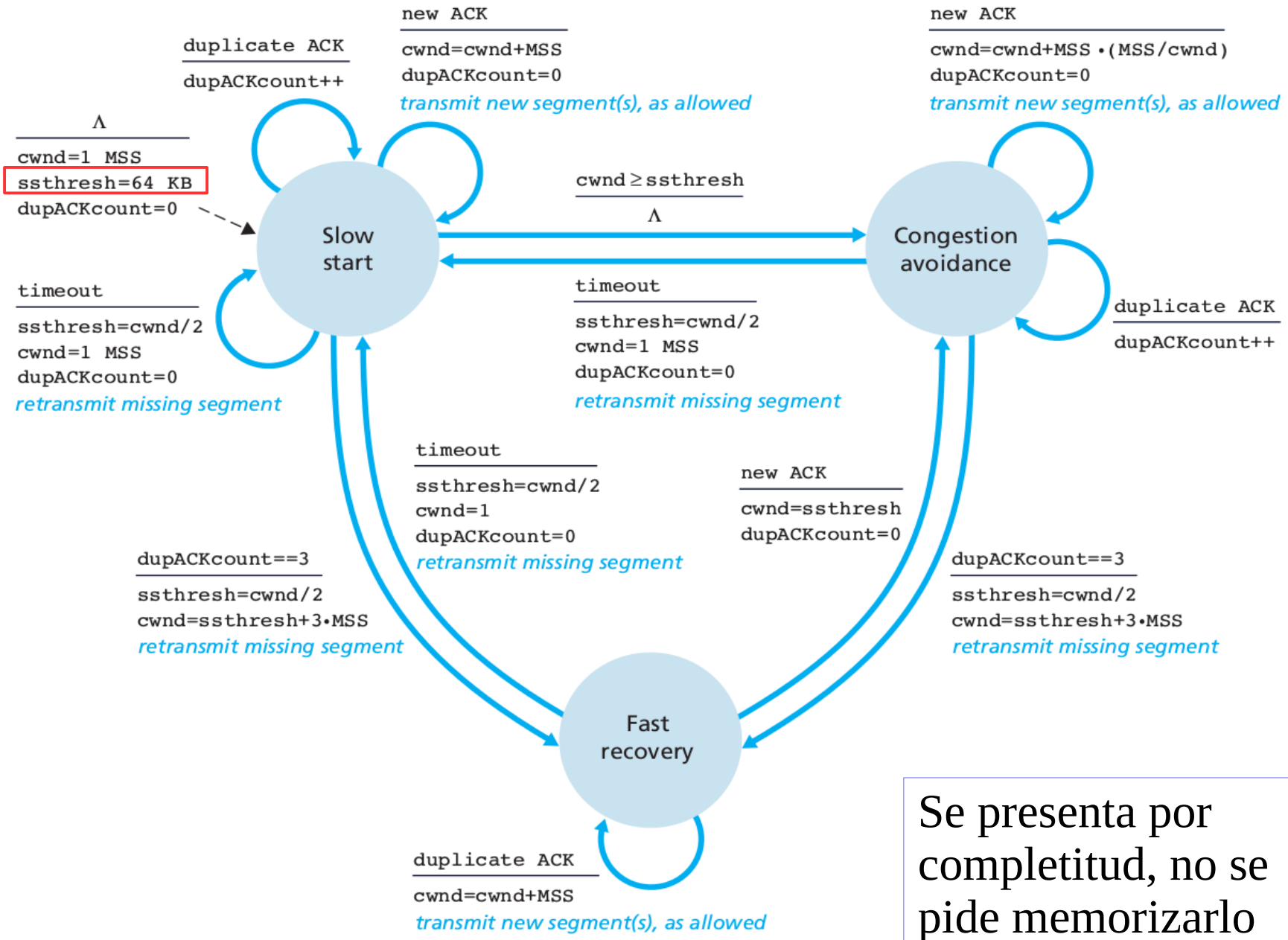
?

- Luego de enviar la ventana W, el Tx debe esperar la llegada del acuse de recibo más antiguo para retomar la transmisión.
- No es válida siempre, pues conforme la ventana de congestión aumenta, aumenta la utilización del canal y cuando ésta alcanza 100% no es posible seguir aumentando la tasa pues la tasa de transmisión del enlace pone una cota máxima.



Resumen: Control de Congestión en TCP

- Cuando **CongWin** está bajo el **Threshold (umbral)**, Tx está en fase slow-start, la ventana de transmisión crece exponencialmente (un MSS por cada ACK).
- Cuando **CongWin** está sobre **Threshold**, Tx está en fase abolición de congestión, la ventana crece linealmente (aprox. un MSS por cada RTT).
- Al tercer ACK duplicados, **Threshold** pasa a **CongWin/2** y **CongWin** pasa a **Threshold**.
- Cuando ocurre un timeout, **Threshold** pasa a **CongWin/2** y **CongWin** se lleva a 1 MSS. Pasa a slow-start.



Se presenta por completitud, no se pide memorizarlo

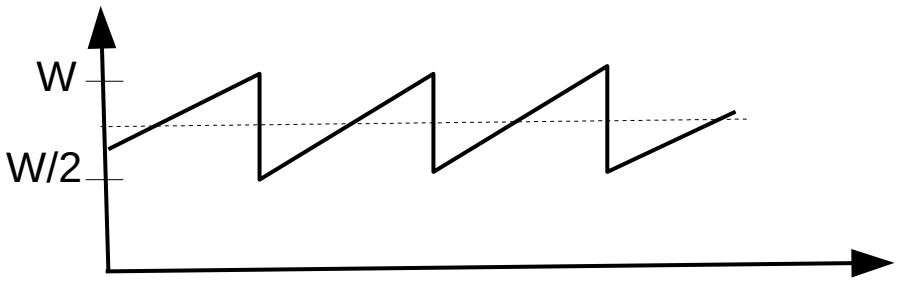
Figure 3.52 ♦ FSM description of TCP congestion control

Control de congestión del Tx TCP
(similar información que diagrama previo)

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to “Congestion Avoidance”	Resulta en una duplicación de CongWin cada RTT.
Congestion Avoidance (CA)	ACK receipt for previously unacked data	CongWin = CongWin+MSS * (MSS/ CongWin)	Aumento aditivo, resulta en aumento de CongWin en apox. 1 MSS cada RTT
SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, Set state to “Congestion Avoidance”	Recuperación rápida, implementando reducción multiplicativa. CongWin no caerá a 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, Set state to “Slow Start”	Ingresa a Partida Lenta (slow start)
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin y Threshold no cambian

Throughput Simplificado de TCP (tasa de transferencia de datos lograda)

- ¿Cuál es el throughput promedio de TCP como una función del tamaño de ventana **`CongWin`** y RTT?
  - Ignoremos slow start ya que al ser exponencial es una fase muy corta, supondremos pocas pérdidas.
- TCP pide ancho de banda adicional al incrementar W en 1 MSS por cada RTT hasta una pérdida
- Sea W el tamaño de la ventana (en bytes) cuando ocurre una pérdida.
- Cuando la ventana es W, el throughput es ~W/RTT
- Justo después de la pérdida, la ventana cae a W/2, y el throughput cae a W/2RTT.
- Throughput promedio entre W/2RTT y W/RTT es 0.75 W/RTT
- Esto debido a que el throughput crece linealmente entre ambos valores.



Futuro de TCP: TCP es enlaces largos y de alta tasa

- Ejemplo: segmentos de 1500 bytes, RTT de 100ms, queremos throughput de 10 Gbps
- Requiere tamaño de ventana **`CongWin`** W = 83333 (segmentos en tránsito) para utilización =1.
- Throughput en términos de tasa de pérdida (L) es:

$$Avg.Throughput = \frac{1{,}22 * MSS}{RTT\sqrt{L}}$$

  L=(bytes perdidos)/(Número total enviados)
- Para alcanzar 10 Gbps de throughput, con el algoritmo de control de congestión actual, se toleran probabilidades de pérdida de solo $L = 2 \cdot 10^{-10}$ *!! (1 cada 5 mil millones de segmentos)*
- Se requieren nuevas versiones de TCP para enlaces de alta velocidad (interesados ver RFC 3649)

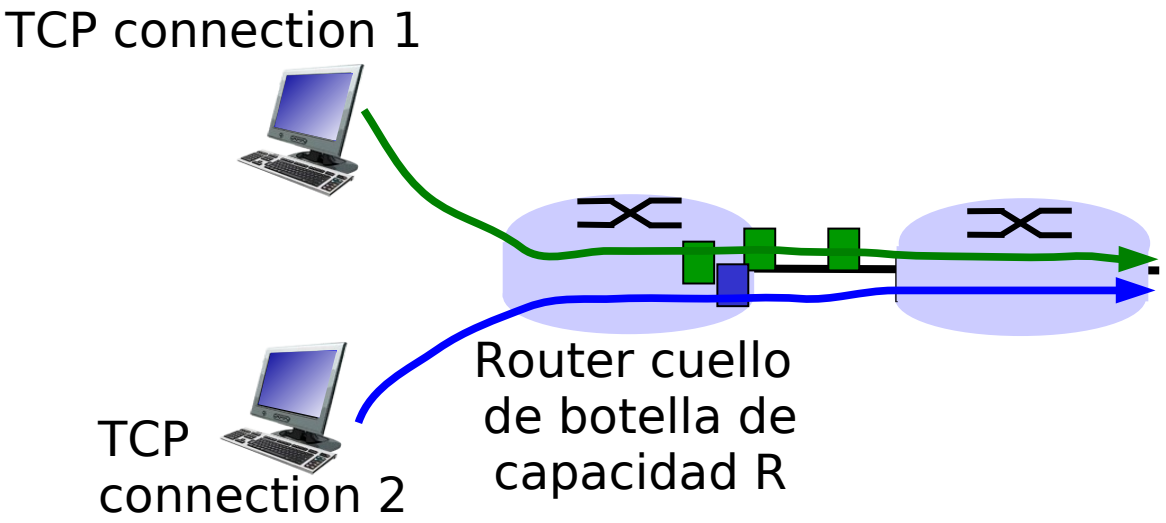
¿Por qué la ventana de congestión de TCP sólo se reduce a la mitad cuando la pérdida es detectada por 3 ACKs duplicados, mientras que se reduce a 1 MSS cuando la pérdida es detectada por timeout?

?

Porque la llegada de 3 ACKs duplicados es una indicación que paquetes posteriores al perdido sí llegaron, luego esta situación de congestión es menos crítica que cuando hay timeout sin 3 ACKs duplicados.

Equidad en TCP

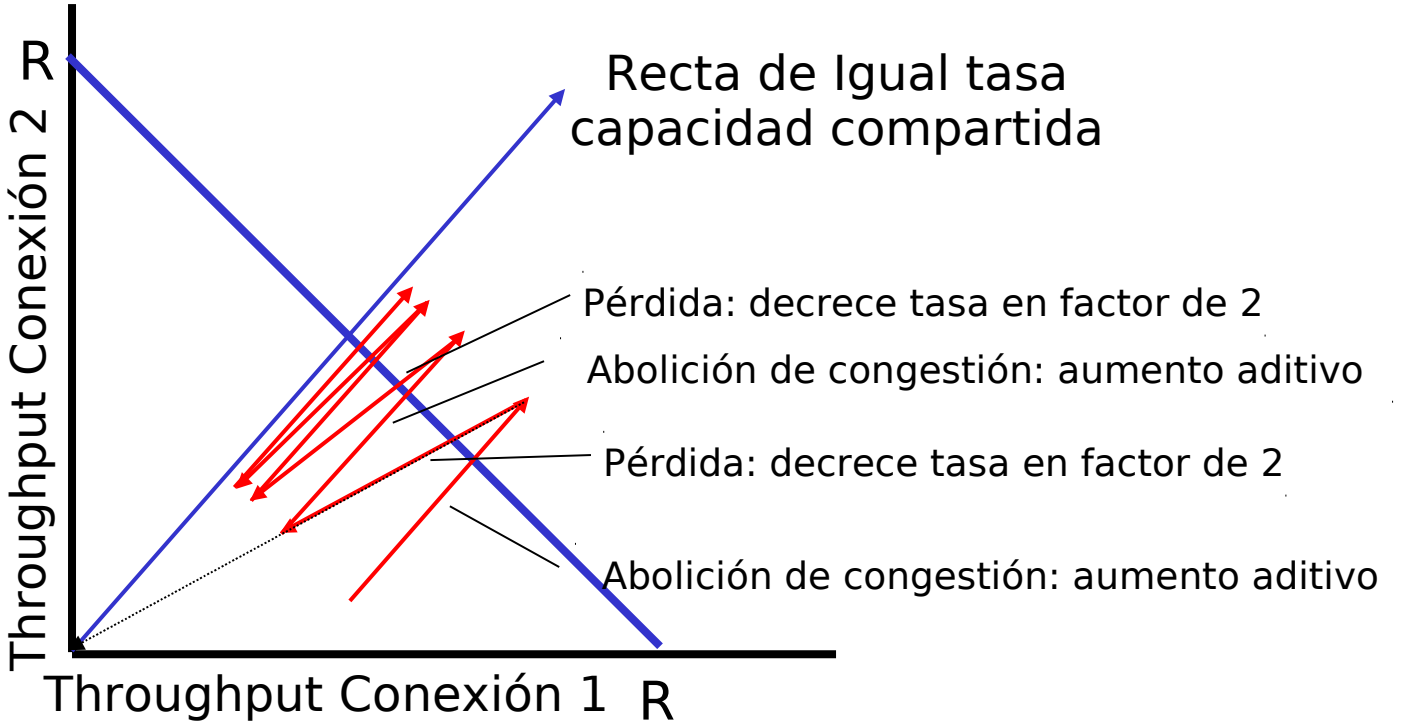
- Objetivo de la Equidad (fairness): Si K sesiones TCP comparten un mismo enlace de ancho de banda R, cada una debería tener una tasa promedio de R/K



¿Por qué TCP es justa?

Supongamos dos sesiones compitiendo:

- Aumento aditivo da pendiente de 1, como aumento de throughout
- Reducción multiplicativa reduce throughput proporcionalmente



Equidad (más)

Equidad y UDP

- Aplicaciones Multimedia no usan TCP
  - No quieren tasa limitada por control de congestión
- En su lugar usan UDP:
  - Envían audio/vídeo a tasa constante y toleran pérdidas de paquetes
- Área de investigación: Hacerlas amistosas con TCP (TCP friendly)

Equidad y conexiones TCP paralelas

- Nada previene a las aplicaciones de abrir conexiones paralelas entre dos hosts.
- Navegadores WEB hacen esto
- Ejemplo: Sea un enlace de tasa R con 9 conexiones;
  - Una aplicación nueva pide 1 conexión TCP, obtendrá R/10
  - Si la aplicación nueva pide 11 conexiones TCP, ésta obtendrá 11R/20 , más de R/2!

En una subred hay 6 usuarios viendo vídeos de Youtube.com vía conexiones TCP. ¿Si éstos fueran los únicos usuarios, qué fracción de la capacidad de un enlace congestionado le debería corresponder a cada uno?

?

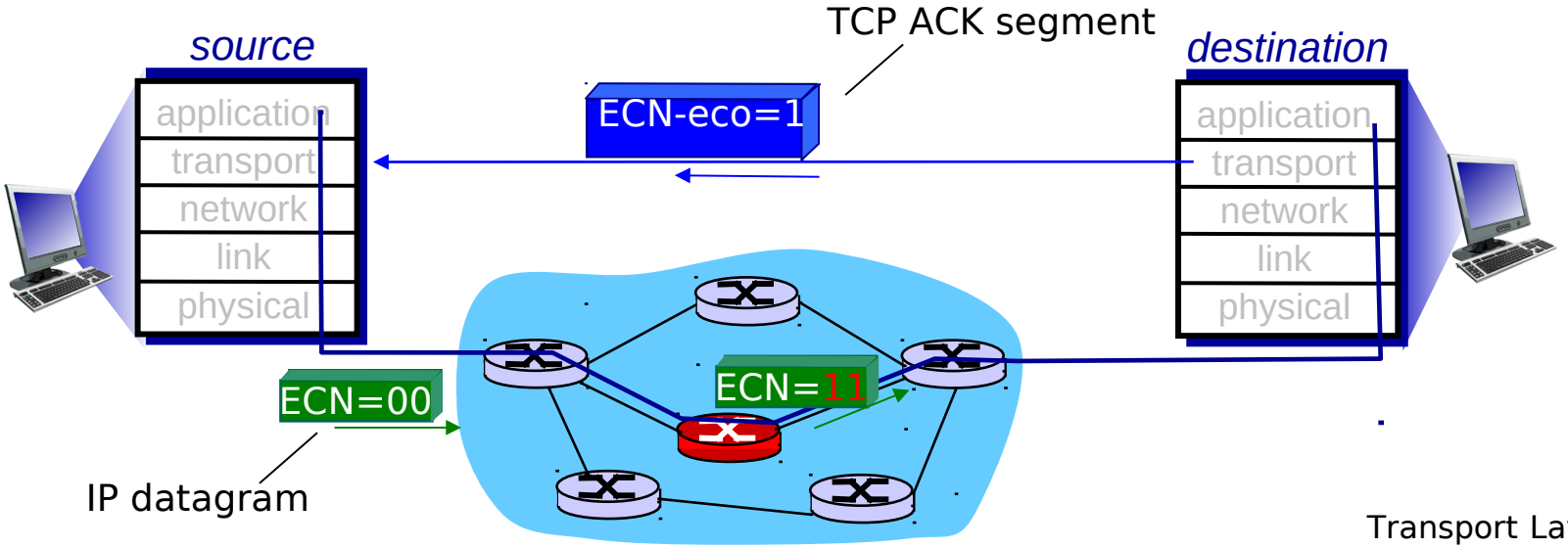
1/6.

Nota: Se supone que ese es el único tráfico en el enlace congestionado; en otro caso será la misma fracción del tráfico para cada conexión.

Explicit Congestion Notification (ECN)

Control de congestión asistido por la red:

- Dos bits en encabezado IP (ToS field) son marcados *por router de la red* para indicar congestión
- Indicación de congestión viaja hasta host receptor
- Receptor (ve indicación de congestión en datagrama IP) fija bit ECN-eco en segmento ACK de receptor-a-transmisor para notificar congestión al transmisor



Capítulo 3: Resumen

- Principios detrás de los servicios de capa transporte:
  - multiplexing, demultiplexing
  - Transferencia confiable de datos
  - Control de flujo
  - Control de congestión
- Uso e implementación en Internet
  - UDP
  - TCP

A continuación

- Dejaremos la "periferia" o "edge" de la red (capas aplicación y transporte)
- Nos internaremos en el centro de la red "network core"