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)
- El emisor limita su tasa:

```
LastByteSent-LastByteAcked
≤ min {CongWin, RcvWindow}
```

- Se busca lograr que, aproximadamente,

$$\text{tasa} = \frac{\texttt{CongWin}}{\text{RTT}} \text{ Bytes/s}$$

- **`CongWin`** es dinámica y función de la congestión percibida de la red

- ¿Cómo percibe el emisor la congestión?
  - Pérdidas = timeout ó 3 acks duplicados
  - Emisor TCP reduce tasa (CongWin) después de pérdidas
- Existen tres mecanismos
  - AIMD (Additive-Increase, Multiplicative-Decrease)
  - Arranque lento (slow start)
  - Conservativo después de timeout

AIMD en TCP

- Decrecimiento multiplicativo
  - Reducir CongWin a la mitad luego de pérdida
- Aumento aditivo
  - Aumentar `CongWin` en 1 MSS cada RTT en ausencia de pérdida
  - En algunas implementaciones `CongWin` incrementa en `MSS(MSS/CongWin)` por cada ACK.

Conexión TCP en el tiempo

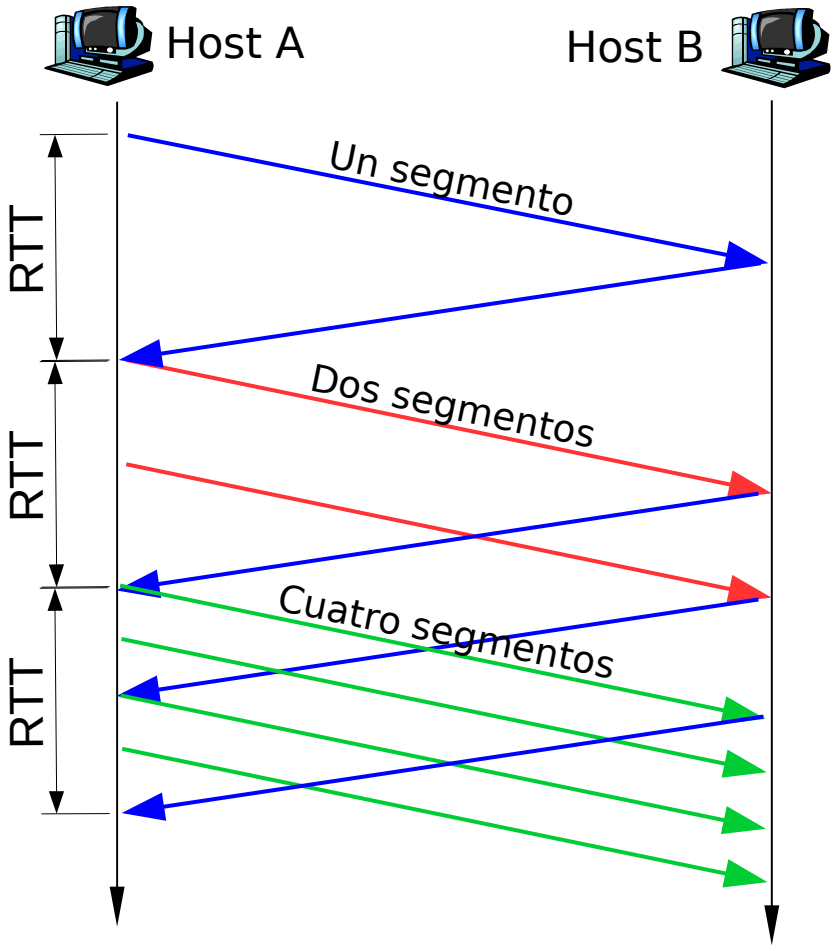
Arranque lento en TCP

- Cuando la conexión comienza, `CongWin` = 1 MSS
  - Ejemplo: MSS = 500 bytes & RTT = 200 msec
  - Tasa inicial = 20 kbps
- El ancho de banda disponible puede ser >> MSS/RTT
  - Es deseable aumentar rápidamente hasta una tasa respetable

- Cuando la conexión comienza, aumentar tasa exponencialmente rápido hasta la primera pérdida

Arranque lento en TCP

- Cuando la conexión comienza, aumentar la tasa exponencialmente hasta primera pérdida:
  - Duplicar `CongWin` cada RTT
  - Se logra al incrementar `CongWin` por cada ACK recibido
- En resumen: la tasa inicial es lenta, pero se acelera exponencialmente rápido

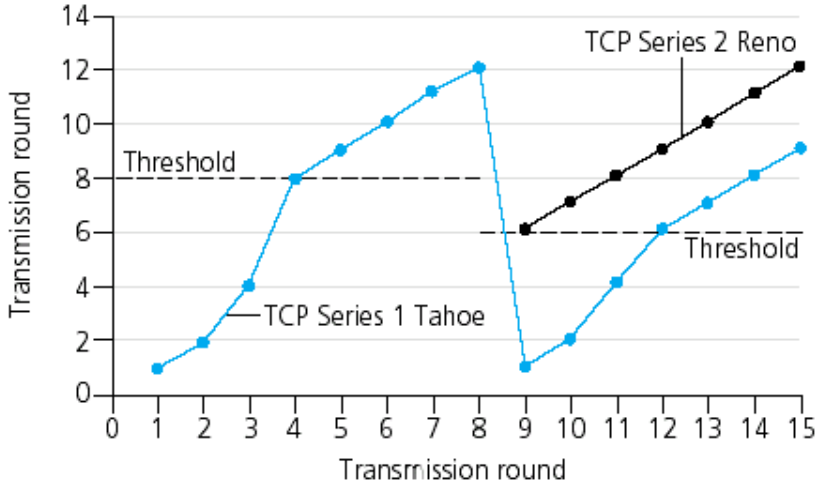


Refinamiento

- Después de 3 ACKs duplicados:
  - `CongWin` baja a la mitad
  - Luego la ventana crece linealmente
- Pero luego de un timeout:
  - `CongWin` es fijada en 1 MSS;
  - La ventana crece exponencialmente
  - Hasta un umbral, luego crece linealmente

- 3 ACKs duplicados indican que la red es capaz al menos de transportar algunos segmentos
- Timeout antes de 3 duplicados es “más alarmante”

Refinamiento



- Tahoe: primera versión de control de congestión en TCP. No distinguía entre timeout o ACK duplicados.
- Reno: versión siguiente de TCP. Distingue timeout de ACK duplicados. Es como opera hoy TCP.

- ¿Cuándo cambiará de exponencial a lineal el crecimiento?

  Cuando `CongWin` llega a 1/2 de su valor antes del timeout.
- Implementación:
  - Umbral variable
  - Ante pérdidas, el umbral es fijado en 1/2 de `CongWin` justo antes de la pérdida

Control de Congestión en TCP

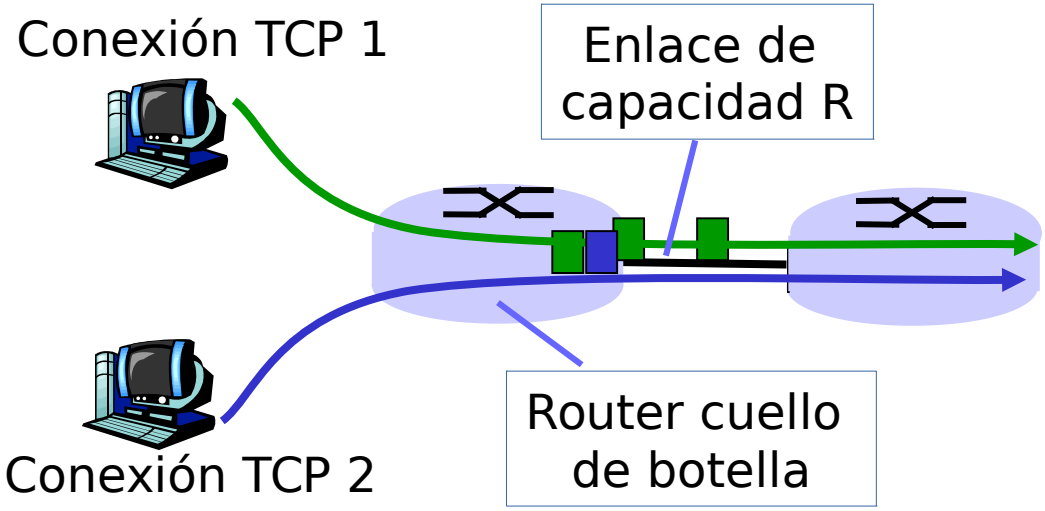
- Cuando **CongWin** está por debajo de **Threshold** (umbral), el emisor está en fase slow-start, la ventana crece exponencialmente.
- Cuando **CongWin** está sobre **Threshold**, el emisor está en fase de evitar congestión, la ventana crece linealmente.
- Cuando ocurre un triple duplicado de ACK, Threshold 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.

Control de congestión

Estado	Evento	Acción del emisor TCP	Comentario
Slow Start (SS)	Se recibe ACK para datos no reconocidos previamente	CongWin = CongWin + MSS, If (CongWin > Threshold) state = CA	Resulta en una duplicación de CongWin cada RTT
Congestion Avoidance (CA)		CongWin = CongWin+MSS * (MSS/CongWin)	Aumento aditivo, resulta en aumento de CongWin en 1 MSS cada RTT
SS o CA	Pérdida detectada por triple ACK duplicado	Threshold = CongWin/2, CongWin = Threshold, state = CA	Recuperación rápida, implementando reducción multiplicativa. CongWin no caerá por debajo de 1 MSS.
	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, state = SS	Ingresa en Arranque Lento (Slow Start)
	ACK duplicado	Incrementar la cuenta de ACK duplicados para ese segmento	CongWin y Threshold no cambian

Equidad en TCP

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



¿Por qué TCP es justo?