Agenda

- 3.1 Servicios de la capa de transporte
- 3.2 Multiplexación y demultiplexación
- 3.3 Transporte no orientado a conexión: UDP
- 3.4 Principios de la transferencia de datos confiable
- 3.5 Transporte orientado a conexión: TCP
  - estructura del segmento
  - transferencia de datos confiable
  - control de flujo
  - gestión de la conexión
- 3.6 Principios del control de congestión
- 3.7 Control de congestión de TCP

Principios del Control de Congestión

Congestión:

- informalmente: "demasiadas fuentes enviando muy rápido demasiado tráfico a ser manejado por la red"
- ¡Diferente de control de flujo!
- manifestaciones:
  - pérdida de paquetes (*buffer overflow* en los routers)
  - grandes retardos (encolamiento en los *buffers* de los routers)

Principios del Control de Congestión

- Resulta en una injusta y pobre utilización de los recursos de la red
  - Los recursos son utilizado por paquetes posteriormente descartados
  - Retransmisiones
  - Pobre asignación de recursos
- ¡Un problema importante!

Perspectiva histórica

- *"Congestion avoidance y control" – Van Jacobson & Michael Karels – November, 1988*
- Si se respeta el "principio de conservación de los paquetes", la congestión debería ser la excepción y no la regla
- Dicho principio dice que para una conexión establecida y "en equilibrio", un nuevo paquete no puede ser introd. en la red antes que otro paquete salga de ella.
- En equilibrio: conexión estable con una ventana completa "*in-flight*"

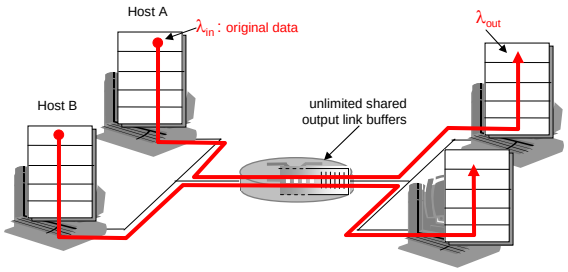
Introduction

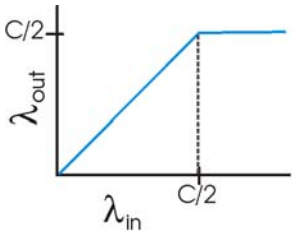
Computer networks have experienced an explosive growth over the past few years and with that growth have come severe congestion problems. For example, it is now common to see internet gateways drop 10% of the incoming packets because of local buffer overflows. Our investigation of some of these problems has shown that much of the cause lies in transport protocol implementations (*not* in the protocols themselves):

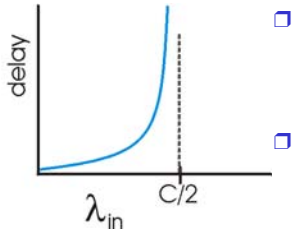
In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and two IMP hops) dropped from 32 Kbps to 40 bps. We were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. In particular, we wondered if the 4.3BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes".

Causas/costos de la congestión: escenario 1

- 2 *senders*, 2 *receivers*
- 1 router, *buffers* infinitos
- sin retransmisiones



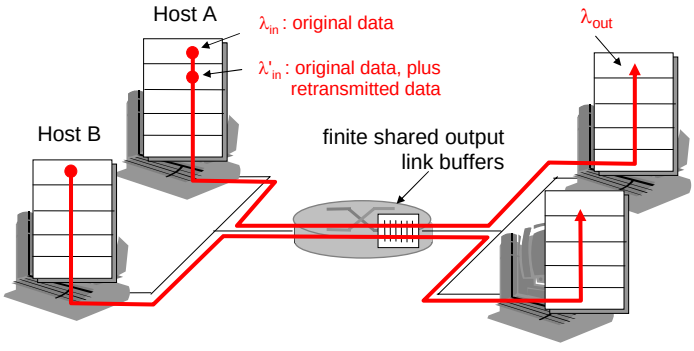




- Cuando hay congestión, tenemos grandes retardos
- máximo *throughput* alcanzable

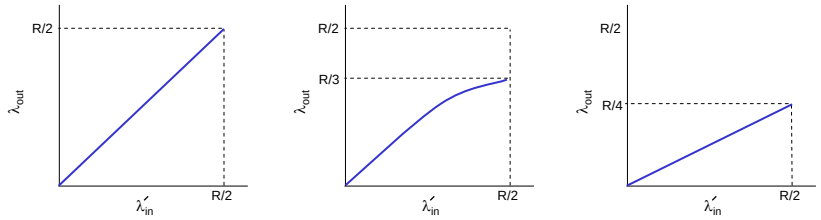
Causas/costos de la congestión: escenario 2

- 1 router, *buffers finitos*
- el *sender* retransmite paquetes perdidos



Causas/costos de la congestión: escenario 2

- siempre: $\lambda_{in} = \lambda_{out}$ (*goodput*)
- retransmisión "perfecta" sólo cuando hay pérdidas: $\lambda'_{in} > \lambda_{out}$
- La retransmisión de paquetes retrasados (no perdidos) hace λ'_{in} mayor (respecto al caso perfecto) para el mismo λ_{out}



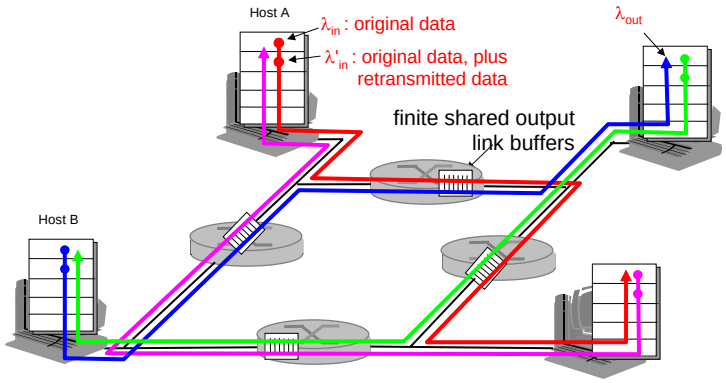
"costos" de la congestión:

- más trabajo (retransmisiones) para un *goodput* dado
- Retransmisiones innecesarias: los enlaces llevan múltiples copias de los paquetes

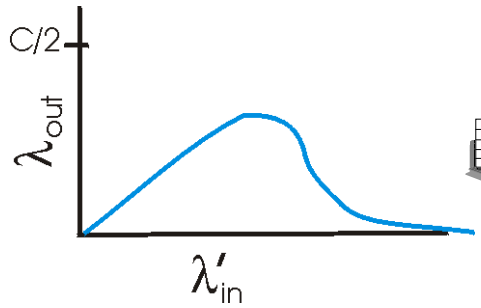
Causas/costos de la congestión: escenario 3

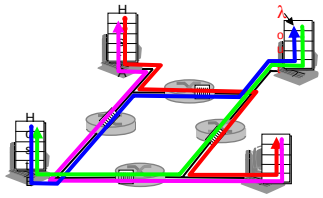
- 4 *senders*
- caminos *multihop*
- *timeout/retransmit*

P: ¿Qué ocurre si λ_{in} y λ'_{in} aumentan?



Causas/costos de la congestión: escenario 3





Otro "costo" de la congestión:

- cuando un paquete es descartado, ¡toda la capacidad de transmisión "aguas arriba" utilizada por dicho paquete fue desperdiciada!

Aproximaciones al control de congestión (1/2)

Objetivo: estrangular al emisor para asegurar que la carga en la
red es "razonable"

Dos estrategias posibles

- Entre extremos
  - No hay soporte explícito de la red
  - Los extremos la infieren
  - Tradicional de TCP

Aproximaciones al control de congestión (2/2)

Objetivo: estrangular al emisor para asegurar que la carga en la
red es "razonable"

- Asistida por la red
  - Los routers realimentan explícitamente a los extremos
  - Directa
    - Con un mensaje explícito
  - Indirecta
    - Marcando un campo en algún paquete
    - Involucra RTTs
    - *Explicit Congestion Notification* (ECN) – RFCs 3168 y 3540 y http://www.icir.org/floyd/ecn.html

Aclaración

- Si el emisor y el receptor de la conexión TCP están unidos por un cable (p.e. UTP) es muy probable que podamos llegar a tener "in-flight" la ventana de recepción
- Si entre ellos hubiera un *switch* "razonable" y con una carga "razonable", quizás también
- La congestión puede aparecer "con fuerza" cuando tenemos una red entre las entidades TCP, y debemos tratar de evitarla-controlarla

Agenda

- 3.1 Servicios de la capa de transporte
- 3.2 Multiplexación y demultiplexación
- 3.3 Transporte no orientado a conexión: UDP
- 3.4 Principios de la transferencia de datos confiable
- 3.5 Transporte orientado a conexión: TCP
  - estructura del segmento
  - transferencia de datos confiable
  - control de flujo
  - gestión de la conexión
- 3.6 Principios del control de congestión
- 3.7 Control de congestión de TCP

Control de congestión de TCP

- RFC 2581, abril 1999
- Entre sistemas finales
  - Sin asistencia de la red
- Límites de transmisión del emisor:
  LastByteSent-LastByteAcked ≤ min {CongWin, RcvWin}
- Sin mucha rigurosidad,

$$\text{tasa} = \frac{\text{CongWin}}{\text{RTT}} \text{ bytes/sec}$$

- La **CongWin** es dinámica; función de la congestión de la red percibida por el emisor

¿Cómo percibe la congestión el emisor?

- evento de pérdida = *timeout* o 3 ACKs duplicados
- El emisor TCP reduce la tasa (**CongWin**) después de de un evento de pérdida

Cuatro algoritmos:

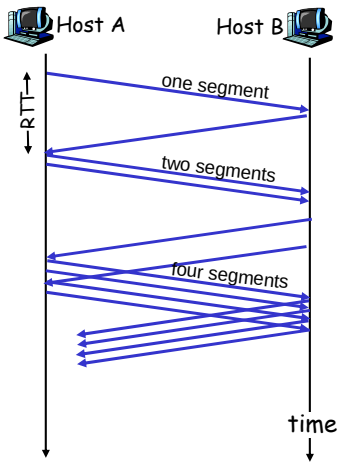
- *slow start*
- *congestion avoidance*
- *fast retransmit*
- *fast recovery*

Control de congestión de TCP: Slow Start ("arranque lento o en frío")

- Cuando la conexión comienza,
  CongWin = 1 MSS
  - Ejemplo: MSS = 500 bytes & RTT = 200 msec
  - Velocidad inicial = 20 kbps
- El ancho de banda disponible podría ser >> MSS/RTT
  - Es deseable que rápidamente lleguemos a una velocidad respetable
- Cuando la conexión comienza, se incrementa la velocidad **exponencialmente** (no linealmente) hasta el primer evento de pérdida

TCP Slow Start (más)

- Cuando la conexión comienza, se incrementa la velocidad exponencialmente hasta el primer evento de pérdida
  - La **CongWin** se incrementa en a lo sumo SMSS bytes por cada ACK "nuevo" recibido ("conservación de los paquetes")
- Resumen: la velocidad inicial es lenta pero crece exponencialmente



Refinamiento: inferiendo la pérdida

- Luego de un evento "3 ACKs duplicados":
  - **CongWin** se baja a la mitad
  - Luego la ventana crece **linealmente**, en 1 SMSS por RTT
- Pero, luego de un evento "*timeout*":
  - **CongWin** vale 1 MSS;
  - La ventana crece **exponencialmente** hasta un umbral, y luego, linealmente

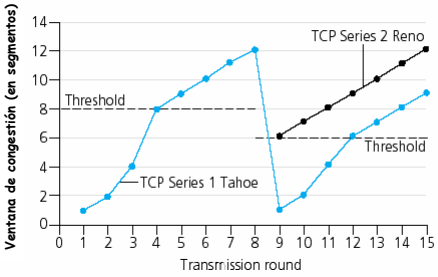
Filosofía:

- 3 ACKs duplicados indica que la red es capaz de entregar algunos segmentos
- *timeout* indica un escenario de congestión "más preocupante"

Refinamiento

P: ¿Cuándo debería el incremento **exponencial** cambiar a **lineal?**

R: Cuando **CongWin** alcanza la mitad de su valor previo al *timeout*.



Implementación:

- Variable *Threshold*
  - *ssthresh*: arbitrariamente alto (por ejemplo, *RcvWnd*)
- Reno: en un evento de pérdida, el *Threshold* es fijado a la mitad del valor de **CongWin** justo antes del evento mencionado

Algunos sabores de TCP

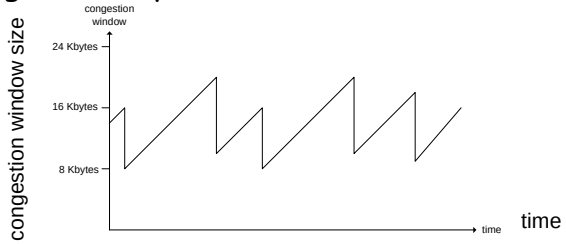
- Reactivos
  - Tahoe (Van Jacobson 1988)
    - Slow Start, Congestion Avoidance y Fast Retransmit
  - Reno (Van Jacobson 1990)
    - RFC 2581 – abril 1999
    - Tahoe + Fast Recovery
  - New Reno
    - RFC 3782 – abril 2004
    - Para contemplar el caso de pérdida de segmentos consecutivos y no poder utilizar la opción SACK
- Proactivos
  - Vegas
    - Busca evitar la congestión tratando de detectar cuando es inminente que ella ocurra (RTT) y así, reduciendo la tasa de envío. Cambios en parte de la implementación de SS y CA

Control de congestión de TCP:

Incremento Aditivo, Decremento Multiplicativo (AIMD)

- Propuesta: incrementar la tasa de transmisión (tamaño de ventana), probando por el ancho de banda utilizable, hasta que ocurra una pérdida
  - *incremento aditivo:* incrementa **CongWin** en 1 MSS cada RTT hasta que se detecta una pérdida
  - *decremento multiplicativo*: recorta a la mitad la **CongWin** luego de una pérdida

Comportamiento de "diente de sierra": Probando en busca de ancho de banda



Resumen: Control de Congestión de TCP

- Cuando **CongWin** está debajo de **Threshold**, el emisor está en la fase *slow-start*, la ventana crece **exponencialmente**.
- Cuando **CongWin** está sobre **Threshold**, el emisor está en la fase *congestion-avoidance*, la ventana crece **linealmente**.
- Cuando ocurre un triple ACK duplicado, el **Threshold** pasa a valer **CongWin/2** y **CongWin se pone al valor de Threshold**.
- Cuando ocurre un *timeout*, el **Threshold** pasa a valer **CongWin/2** y **CongWin,** 1 MSS.

TCP: control de congestión del emisor

Estado	Evento	Acción del Emisor TCP	Comentario
Slow Start (SS)	Recepción de ACK para datos previamente no-ACKed	CongWin = CongWin + MSS, If (CongWin > Threshold) pasa a estado "Congestion Avoidance"	Se dobla CongWin por cada RTT
Congestion Avoidance (CA)	Recepción de ACK para datos previamente no-ACKed	CongWin = CongWin + MSS x (MSS/CongWin)	Incremento aditivo. Aumento de CongWin en 1 MSS por cada RTT
SS or CA	Evento de pérdida detectado: Triple ACK duplicado	Threshold = CongWin/2, CongWin = Threshold pasa a estado "Congestion Avoidance"	*Fast recovery*, implementando disminución multiplicativa. CongWin no puede ser menor a 1 MSS.
SS or CA	*Timeout*	Threshold = CongWin/2, CongWin = 1 MSS, pasa a estado "Slow Start"	
SS or CA	ACK duplicado	Incrementa el contador de ACK duplicado para el segmento que es ACKed	CongWin y Threshold no cambian

¿Pérdida de paquetes = congestión?

- Respuesta parcial: "*depende*"
- Completando la respuesta
  - En medios no hostiles (redes UTP actuales, redes de fibra óptica) podríamos llegar a afirmar que sí
  - En medios hostiles (redes inalámbricas) podemos afirmar: "no necesariamente"
  - TCP para redes inalámbricas, porque las "señales de congestión" del TCP tradicional en este contexto, no siempre son adecuadas

TCP *throughput* ("para diente de sierra")

- ¿Cuál es el *throughout* promedio de TCP en función del tamaño de ventana y del RTT?
  - No tenemos en cuenta la fase *slow start*
- Sea **w** el tamaño (en bytes) de ventana cuando ocurre una pérdida.
- Cuando la ventana es w, el *throughput* es aprox. **w/RTT**
- Justo después de la pérdida, la ventana cae a **w/2**, y el *throughput* pasa a **w/2RTT**.
- Como el throughput se incrementa linealmente entre estos dos valores,
  - *throughput* promedio: **0.75 w/RTT**

TCP Futuro: TCP sobre "elefantes" (LFN: *Long and Fat Networks*)

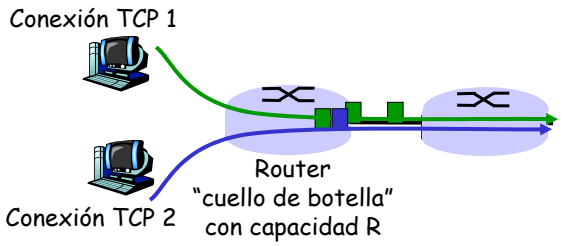
- Las aplicaciones cambian y por lo tanto los servicios requeridos a TCP también
- RFC 3649: *High Speed TCP for Large Congestion Windows*
- Ejemplo : segmentos de 1500 bytes, 100ms RTT, precisa 10 Gbps de *throughput*
- Requiere un tamaño de ventana de W = 83.333 segmentos en tránsito
- *Throughput* en función de la tasa de pérdida (L):

$$\frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

- ➜ aprox. L = 2 x 10^{-10} o sea, una pérdida cada 5 x 10^{9} segmentos, o sea, 1 pérdida cada 1 hora y 40 minutos… glup!!
- Investigación sobre nuevas versiones de TCP para alta velocidad

TCP Imparcialidad

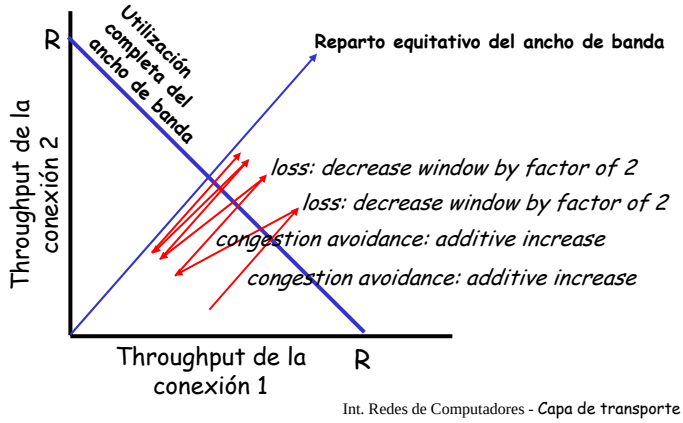
Objetivo de la imparcialidad: si **K** sesiones TCP comparten el mismo enlace "cuello de botella" de ancho de banda **R**, cada uno debería tener una tasa promedio de **R/K**



¿Por qué es imparcial TCP? Fase CA

Dos sesiones compitiendo:

- Incremento aditivo
- Decremento multiplicativo



Imparcialidad (más)

Imparcialidad y UDP

- Las aplicaciones multimedia frecuentemente no utilizan TCP
  - no quieren que la tasa se vea estrangulada por el control de congestión
- En su lugar utilizan UDP:
  - inyectan audio/video a una tasa constante, toleran pérdidas de paquetes
- Área de investigación
  - *TCP friendly*
  - Protocolos de control de congestión

Imparcialidad y conexiones TCP paralelas

- nada impide a una aplicación basada en TCP abrir varias conexiones TCP paralelas entre 2 *hosts*.
- Los *web browsers* hacen ésto
- Ejemplo: enlace de tasa R soportando 9 conexiones;
  - Si nueva aplic. requiere 1 conexión TCP, recibe tasa R/10
  - Pero, si nueva aplic. requiere 11 conexiones TCP, ¡ recibe R/2 !

Capítulo 3: Resumen

- principios detrás de los servicios de la capa de transporte:
  - Multiplexación, demultiplexación
  - Transferencia de datos confiable
  - Control de flujo
  - Control de congestión
- Instanciación e implementación en Internet
  - UDP
  - TCP

Lo que se viene:

- dejamos el borde (capas de aplicación y de transporte)
- nos metemos en el "*core*" de la red