# Protocolos de control de congestión

Su formulación como un problema NUM

# Protocolo TCP

- Protocolo de capa de transporte (extremo a extremo) orientado a conexión y confiable
- Garantiza un flujo confiable de información entre un cliente y un servidor

# Control de congestión

- Para controlar la congestión hay que tener realimentación de la red
- TCP supone que si el reconocimiento de un segmento no llega dentro del tiempo establecido, se debe a que la red está congestionada
- En esta hipótesis, la ventana del trasmisor se ajusta ante la ocurrencia de timeouts en los reconocimientos

# Control de congestión en TCP

- Cuando se detecta una situación de congestión, cwnd se reduce a 1 segmento.
- Sin embargo como se detectó congestión, ahora se crecerá exponencialmente hasta la mitad de la ventana máxima que se logró y luego linealmente.
- El punto medio se almacena en una variable llamada ssthresh

# Algoritmo

- Cuando se establece una conexión
  - cwnd= segsize
  - ssthresh= 65535
- En cada ACK se crece cwnd = cwnd+segsize hasta rwnd o que ocurra congestión.
- Cuando ocurre un timeout
  - ssthresh = max(cwnd/2,segsize*2)
  - cwnd = segsize
- Cuando se recibe un ACK
  - Si cwnd <= ssthresh, entonces cwnd = cwnd +segsize
  - En otro caso
  - cwnd= cwnd + segsize*segsize/cwnd

# Los ACKs en TCP

- TCP reconoce el próximo byte que espera recibir
- ACK = x indica que todos los bytes hasta e incluido el x-1 han sido recibidos satisfactoriamente
- Cuando se recibe x-1,x+1,x+2,x+3,.. Se responde los ACK de x,x,x,x,... Indicando que se espera recibir un segmento con número de secuencia x
- ACKs duplicados son fuertes indicaciones de que se perdió el paquete x.

# Triple recibo de ACK

- Cuando se reciben un ACK(x) por triplicado se retransmite x inmediatamente ( fast retransmission)
- Queremos además responder a esta situación disminuyendo la ventana pero no de manera tan pesimista como cuando se da un time-out.
- Lo que se hace luego de retransmitir x es reducir cwnd a la mitad.
- A partir de este punto se comienza con la parte lineal salteando la fase de slow start (fast recovery)

# Fast retransmit

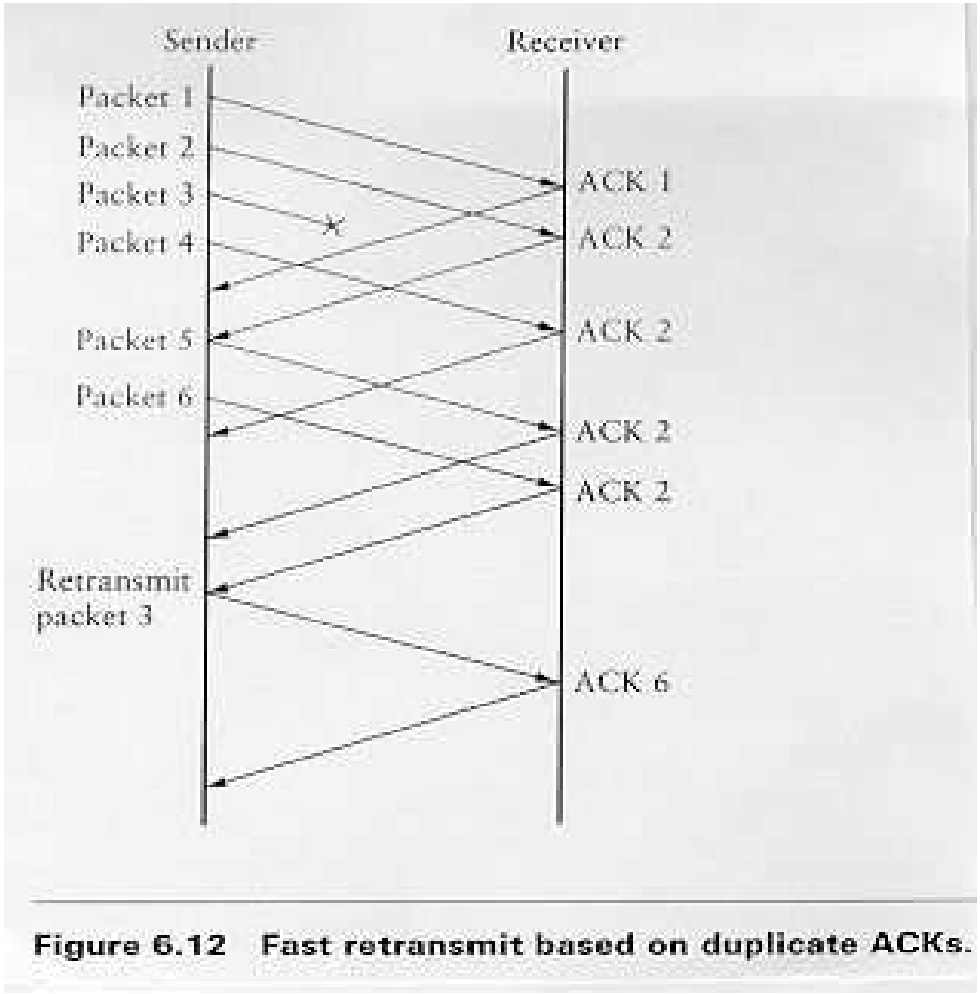



**Figure 6.12 Fast retransmit based on duplicate ACKs.**

# Indicaciones de congestión

- Una fuente TCP recibe dos tipos de indicaciones de pérdida de paquetes
  - Triplicaciones de ACK (TDA)
  - Time-outs (TO)
- TDA reduce cwnd a la mitad
- TO reduce a 1 segmento.

# Time Outs

- Inicialmente se setea el período de TO a un valor inicial $T_0$
- Luego de un TO cwnd es reducida a 1 segmento, permitiendo solo la retransmisión del paquete perdido
- Si la retransmisión falla, el período TO es incrementado a $2T_0$
- Si las retransmisiones continúan fallando se sigue incrementado a $4T_0$ , etc. hasta $64T_0$
- $T_0$ se setea en un valor inicial y luego se ajusta en función del RTT medido

# Análisis de performance de TCP

- Se han desarrollado diversos modelos de TCP en los últimos años
- Todos ellos realizan simplificaciones al protocolo y a la red para poder llegar a modelos tratables analíticamente
- La mayoría de los modelos se aplican
  - ó a largas transferencias de archivos
  - ó a conexiones de corta duración

# Análisis de performance de TCP

- Los modelos para largas transferencias descartan las fases de establecimiento de la conexión y desconexión y la fase de slow start. Nosotros analizaremos las hipótesis y los resultados a los que llegan. El detalle de los cálculos, los pueden encontrar por ejemplo en:
  - Modeling TCP throughput: a simple model and its empirical validation- Padhye, Firoiu, Towsley, Kurose-SIGCOMM 98
- Los modelos para flujos cortos se concentran por el contrario en el modelado del establecimiento de la conexión y en la fase de slow start. Nosotros analizaremos las hipótesis y los resultados a los que llegan. El detalle de los cálculos, los pueden encontrar por ejemplo en:
  - TCP model for short lived flow- Mellia, Stoica, Zhang, IEEE Communication Leters, 2002.

# Análisis de performance de TCP

- Analizaremos el caso de conexiones largas y llegaremos a resultados similares a los de Padhye et al. Pero veremos el problema como maximización de utilidades y no a través del análisis probabilístico del control de congestión.

- Ejercicio: Leer el paper de Padhye et al.

# Modelo de TCP

(Misra, Gong, Towsley, Hollot '00/'01, Low-Paganini-Doyle '02)

Reno:

$$\frac{dw_i}{dt} = \frac{x_i(t-\tau_i)(1-q_i(t))}{w_i} - \frac{w_i(t)}{2} x_i(t-\tau_i) q_i(t)$$

**Incremento aditivo**

**Decremento multiplicativo**

$\frac{1}{w_i}$ incremento por cada ACK,

$x_i(t-\tau_i)(1-q_i(t))$: tasa de ACKs

$-\frac{w_i(t)}{2}$ decremento por pérdida,

$x_i(t-\tau_i)q_i(t)$ tasa de pérdidas

$\tau_i$ : RTT

$w_i$ : ventana

$b_l$ : cola enlace

$x_i \approx \frac{w_i}{\tau_i}$ :tasa de la fuente

$y_l$: tasa del enlace

$q_i$ : probabilidad de paquetes perdidos vista en la fuente

$p_l$ : probabilidad de pérdida de paquetes en los enlaces

# Un problema de optimización (Kelly)

- Una forma de modelar los problemas de asignación de recusros
- Asume que las fuentes tienen una función de utilidad *Ui (x) que es monótona creciente y cóncava* donde expresa la demanda por esa tasa de transmisión.
- Interpreta las tasas de equilibrio como la solución a un problema de optimización convexa

$$\max_x \sum_i \underbrace{U_i(x_i)}_{\text{función utilidad fuente}}, \quad \text{sujeto a} \quad \underbrace{Rx \le c}_{\text{Restricciones de capacidad}} \qquad (\#)$$

- El problema se puede resolver como un problema descentralizado,donde cada fuente trata de maximizar su beneficio

$$\text{Max}_{x_i} \underbrace{Ui(xi)}_{\text{UTILIDAD}} - \underbrace{\mathbf{q_i}\ \mathbf{x_i}}_{\text{"COSTO" DE LA RUTA}} .$$

cuando la red se congestiona

Usa precios que dependen de la capacidad de la red, es decir aumentan

# ¿Cómo se relaciona esto con TCP ?

Cada protocolo define una función utilidad, las características en régimen y de la dinámicas están definidas por esta función

$$\max_{x_i} \left(U_i(x_i) - q\ x_i\right)$$

En el caso de TCP RENO

$$\frac{dw_i}{dt} = \frac{x_i(t-\tau_i)(1-q_i(t))}{w_i} - \frac{w_i(t)}{2} x_i(t-\tau_i) q_i(t) = 0$$

$$\Rightarrow\ x^* = f(q^*) = \frac{1}{\tau}\sqrt{\frac{2(1-q^*)}{q^*}} \Rightarrow\ U_i^{reno}(x_i) = \frac{\sqrt{2}}{\tau_i}\tan^{-1}\left(\frac{x_i\tau_i}{2}\right)$$

**Steven H. Low: A duality model of TCP and queue management algorithms. IEEE/ACM Trans. Networking. 11(4): 525-536 (2003)**

# Padhye vs NUM

- Padhye cuando p es pequeña, y solo se consideran las indicaciones de triple ACK

$$x = \frac{1}{RTT}\sqrt{\frac{3}{2bq}}$$

- NUM (b=1)

$$x^* = f(q^*) = \frac{1}{RTT}\sqrt{\frac{2(1-q^*)}{q^*}} \approx \frac{1}{RTT}\sqrt{\frac{2}{q^*}}$$

Más allá de constantes se llega al mismo resultado fundamental:
•El throughput es inversamente proporcional al RTT
•El throughput es inversamente a la raiz de la probabilidad de pérdidas

# Análisis de la estabilidad de TCP-Reno/RED

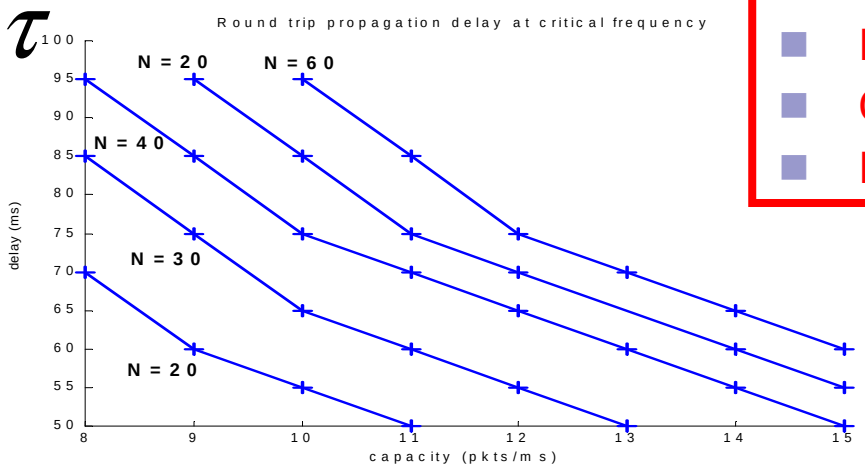
•Si no se tiene en cuenta el retardo el sistema es estable en el mismo sentido de los resultados vistos para los algoritmos primal, dual, etc.

•Los puntos de equilibrio se pueden calcular y los resultados consistentes con los modelos previos.

• Se estudia la estabilidad entorno al punto de equilibrio linealizando. Resultado: Inestable para ventanas grandes, es decir cuando el **Rtt es granade** o cuando la **capacidad es grande**!

Inestable para
- Retardo grande
- Capacidad grande
- N pequeño

Región de estabilidad: N fuentes idénticas, un enlace cuello de botella




capacidad

# Modelos de TCP

- Sobre esta base se pueden analizar otros sabores de TCP. Ej.TCP-Vegas
- También esto sirve de base para proponer nuevas versiones de TCP mejores en el sentido que no presenten oscilaciones y logren mejor throughput. Ej. FAST