Diseño y Operación de Redes Telemáticas

Tema 4. Análisis de Técnicas y Protocolos de Transporte: TCP

Ramón Agüero Calvo

Departamento de Ingeniería de Comunicaciones

Índice

Índice

Control de congestión en TCP

- Se trata de limitar la tasa a la que se envía la información en función de la congestión que se 'percibe' en la red
  - Si hay poca congestión se incrementa la tasa
  - Si aparece congestión, la tasa se reduce
- ¿Cómo se limita la tasa?
  - Uso de la ventana de congestión: `cwnd`

$$\text{Bytes por confirmar} \leq \textbf{min}\{\texttt{cwnd}, \texttt{rwnd}\}$$

  - El buffer de recepción (control de flujo), `rwnd` se supone mayor que el `cwnd`, por lo que la tasa se puede estimar como

$$\beta = \frac{\texttt{cwnd}}{RTT}$$

Control de congestión en TCP

- ¿Cómo se percata el transmisor TCP de la congestión?
  - Ante la pérdida (y retransmisión) de segmentos: *3ACK* y *timeout*
- ¿Qué algoritmos se usan para modificar la tasa?
  - Cuando se confirman nuevos segmentos se incrementa el `cwnd` y, al detectar pérdidas, se reduce
  - Un problema es establecer la velocidad a la que crece el `cwnd`
  - Algoritmos *Slow Start*, *Congestion Avoidance* y *Fast Recovery*

Control de congestión en TCP

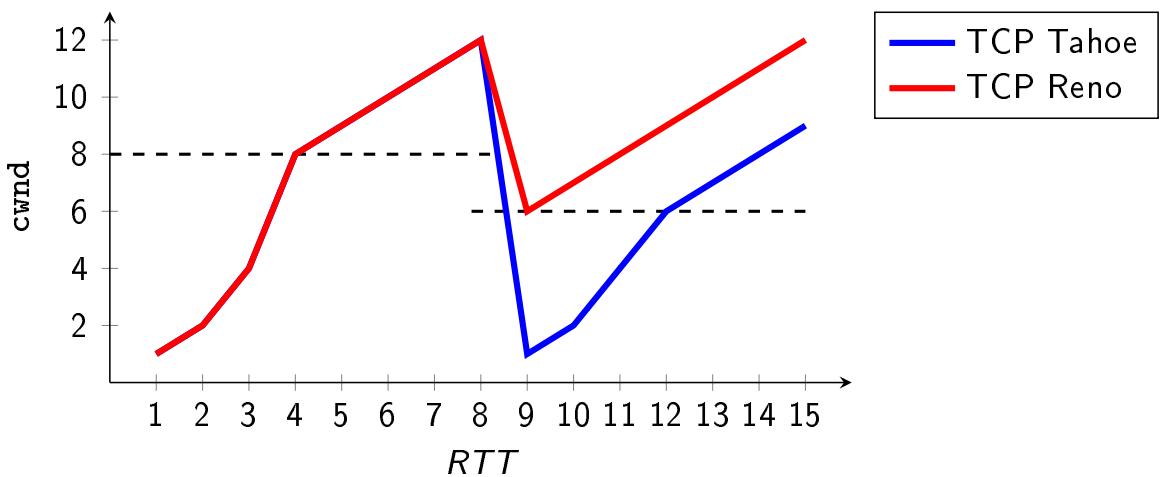
Slow Start y Fast Recovery

- Se pretende tener un incremento exponencial ($\times 2$) en cada RTT
- Cada vez que llega un ACK nuevo se incrementa el `cwnd` en un segmento (MSS bytes)
- ¿Hasta cuándo se mantiene esa tasa de crecimiento?
  - Hasta que alcanza el valor del `ssthresh`
  - El umbral `ssthresh` se fija a la mitad del `cwnd` cuando se produce una retransmisión
- Cuando se produce una retransmisión
  - Por *timeout*: $\texttt{cwnd} = 1$
  - Por *3ACK* (si se usa *Fast Recovery*): $\texttt{cwnd} = \frac{\texttt{cwnd}}{2}$

    De manera temporal la ventana se *infla* para que valga $\frac{\texttt{cwnd}}{2} + 3$.

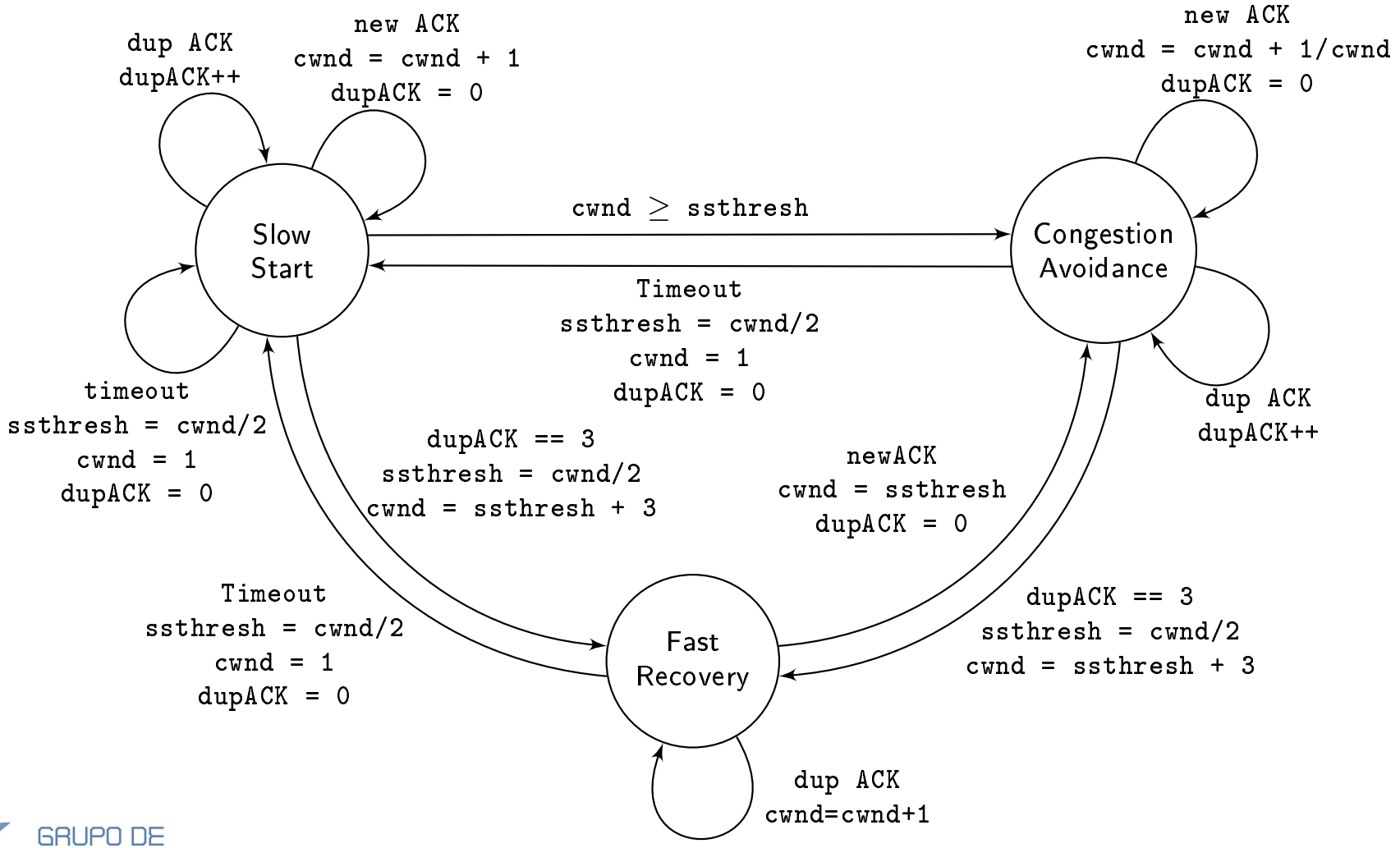
Control de congestión en TCP
Congestion Avoidance

- Se pretende que el ritmo al que crece la tasa sea menor, 1 MSS por RTT, crecimiento lineal
- Para ello cada vez que se recibe un ACK nuevo se incrementa `cwnd` en $\frac{MSS}{\texttt{cwnd}}$
- La reacción ante situaciones de pérdida (*timeout* y *3ACK*) es la misma que en *Slow Start*



Control de congestión en TCP

Máquina de estados



Control de congestión en TCP

Implementaciones

- Tahoe: Slow Start y Congestion Avoidance
- Reno: Tahoe + Fast Recovery
- New Reno: procedimientos para gestionar mejor la pérdida de múltiples segmentos
- Vegas: se 'trata' de anticipar la aparición de pérdidas antes de que suceda
- La opción TCP SACK, que permite establecer mecanismos de recuperación *selectivos*, también se emplea de manera frecuente

Índice

Evolución del control de congestión en TCP

- Las características de las redes han cambiado notablemente desde que se originó TCP
- Se hace necesario adaptar el funcionamiento de los algoritmos de control de congestión de TCP
  - Para redes con alto *Bandwidth Delay Product* (BDP)
  - En redes inalámbricas
- La evolución natural vista anteriormente no es suficiente para asegurar un comportamiento óptimo en estas redes, por lo que se hacen necesarios cambios más importantes
- El survey de Afanasyev *et al.*[§] proporciona una discusión extensa acerca de las diferentes propuestas que se han llevado a cabo

[§] A. Afanasyev y col. "Host-to-Host Congestion Control for TCP". En: *Communications Surveys Tutorials, IEEE* 12.3 (2010), págs. 304-342

TCP en redes con elevado BDP

- El ritmo de crecimiento 'conservador' de TCP en *congestion avoidance* puede resultar muy lento para redes con alto BDP
- Se necesitan $D \times$ `cwnd` *RTTs* para alcanzar la tasa que soporta la red
  - Para una red de 10 Gbps con 100 ms de retardo y un MSS $\approx$ 1500 Bytes se necesitaría alrededor de 2 horas para llegar a la tasa máxima, asumiendo que no se produjera ninguna pérdida
- Se han propuesto algoritmos para este tipo de redes
  - Alta capacidad: redes ópticas
  - Retardos elevados: enlaces satelitales

TCP en redes con elevado BDP

Requisitos

- Uso eficiente de los recursos
- Reaccionar rápido ante los cambios
- Distribución equitativa de los recursos
  - **Intra:** con flujos que usen el mismo algoritmo
  - **Inter:** con flujos que usen otros algoritmos
  - **RTT-Fairness:** flujos con RTT diferentes que comparte un mismo enlace *cuello de botella*

TCP en redes con elevado BDP
TCP Cubic

- Uno de los principales algoritmos es el TCP Cubic
- Se ha convertido en el segundo algoritmo de congestión más extendido, ya que es el que incorpora el kernel de Linux a partir de la versión 2.6.x
- La ventana de congestión crece según la siguiente expresión

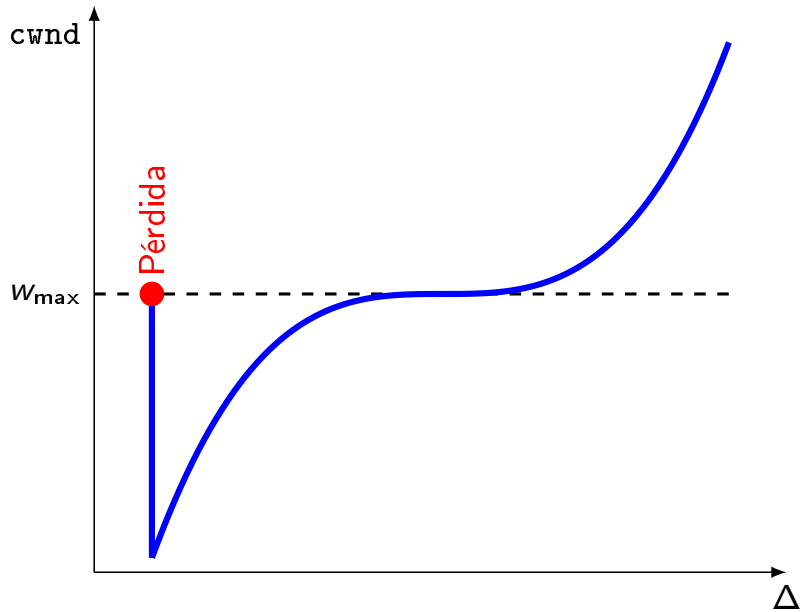
$$w = C\left(\Delta - \sqrt[3]{\beta\frac{w_{max}}{C}}\right)^3 + w_{max}$$

  - C y β son constantes.
  - Δ es el tiempo transcurrido desde el último evento de pérdida.
  - w_{max} es el valor de la `cwnd` al producirse ese último evento de pérdida.

TCP en redes con elevado BDP

TCP Cubic

- Se consigue un crecimiento rápido cuando w está lejos de w_{max} y más lento cuando se está en torno a dicho valor



TCP en redes inalámbricas

- En el caso de enlaces inalámbricos es muy probable que se produzcan errores debido a la hostilidad del canal
- En esas circunstancias reducir la tasa no es una acción adecuada
- Las soluciones se pueden dividir entre aquella que 'rompen' en comportamiento extremo-a-extremo de TCP y las que únicamente modifican sus algoritmos de control de congestión
- Soluciones que introducen elementos adicionales
  - Explicit Loss Notifications, mandados por los nodos (routers) intermedios
  - Elementos intermedios, capa de enlace o agente *snoop*
  - Indirect-TCP, que parte la conexión TCP en dos

TCP en redes inalámbricas

- Si se pretende 'conservar' el comportamiento extremo a extremo de TCP se modifican los algoritmos de control de congestión
- Se incorporan técnicas que buscan 'decidir' la causa de una pérdida
- Una de las propuestas más interesantes es TCP Westwood, que trata de establecer un valor óptimo para la tasa, β
  - Si se trata de una pérdida aleatoria no se modifica la tasa de envío de información
  - Cuando hay congestión, la tasa se igual a aquella que se haya observado recientemente

TCP en redes inalámbricas

- El reto es establecer mecanismos para que el transmisor estime β, sin necesidad de añadir nuevos paquetes

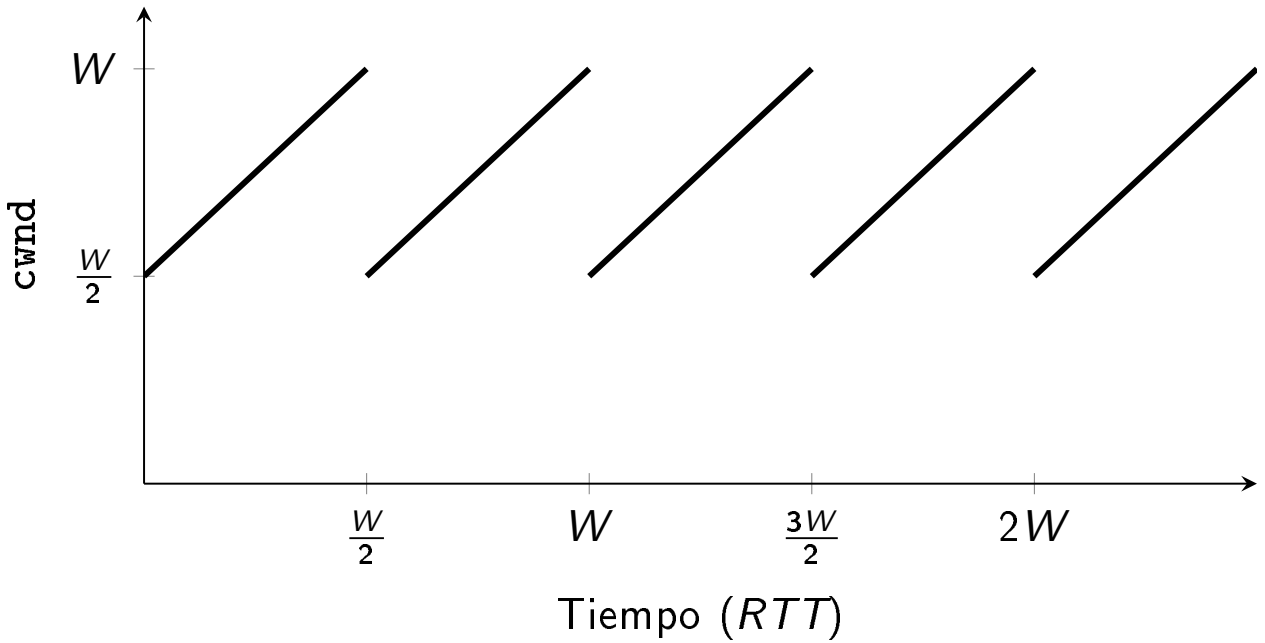
$$b = \frac{d}{\Delta} \qquad \beta = \alpha(\Delta)\beta_{-1} + [1 - \alpha(\Delta)] \left(\frac{b + b_{-1}}{2} \right)$$

- Donde
  - b es la estimación instantánea del ancho de banda
  - d son los bytes confirmados por un ACK
  - Δ es el tiempo desde que se recibió el anterior ACK
  - $\alpha(\Delta)$ es un coeficiente para promediar (filtro)
  - Los términos con subíndice $_{-1}$ se corresponde con los de la estimación previa

Índice

Tasa TCP en conexiones 'largas'

- Se asume que no hay retransmisiones por *timeout*, con lo que la única causa de errores son los *3ACK*
- La evolución de la ventana de congestión es *Additive Increase Multiplicative Decrease* (AIMD)
- Si se confirmaran todos los segmentos, la `cwnd` crece linealmente hasta que haya una pérdida, bajando a $\frac{\texttt{cwnd}}{2}$



Tasa TCP en conexiones 'largas'

- Número de segmentos recibidos por ciclo es $N \approx \frac{3W^2}{8}$, como se pierde un único segmento, se puede obtener el valor de W en función de p (probabilidad de pérdida)

$$W = \sqrt{\frac{8}{3p}}$$

- Por otra parte, la tasa TCP, β se puede calcular como

$$\beta = \frac{MSS \cdot \frac{3}{8}W^2}{RTT \cdot \frac{W}{2}} = \frac{MSS}{RTT} \cdot \frac{3}{4}W$$

- Llegando a la expresión que relación la tasa TCP con la probabilidad de pérdida, el MSS y el RTT

$$\beta = \frac{MSS}{RTT} \cdot \sqrt{\frac{3}{2p}} \approx 1.22 \frac{MSS}{RTT} \frac{1}{\sqrt{p}}$$

Tasa TCP en conexiones 'largas'

- El anterior modelo[§] se puede extender para considerar *Delayed ACK*, que limita el número de ACKs transmitidos (un ACK cada dos segmentos de datos)
- Sin embargo, su principal limitación es que no tiene en cuenta las retransmisiones por *timeout*
- Padhye deriva una expresión que sí tiene en cuenta este aspecto, y que es la que mayor aceptación tiene por parte de la comunidad investigadora
- Considera que se puede confirmar más de un segmento por ACK (parámetro b) y el *Retransmission TimeOut* (RTO), T_0

[§] Matthew Mathis y col. "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm". En: *SIGCOMM Comput. Commun. Rev.* 27.3 (jul. de 1997), págs. 67-82

Tasa TCP en conexiones 'largas'

- Expresión de *Padhye*[§]

$$\beta = \frac{1}{RTT\sqrt{\frac{2\,b\,p}{3}} + T_0 \min\left[1, 3\sqrt{\frac{3\,b\,p}{8}}\right] p\,(1+32p^2)}$$

- Padyhe también considera que se puede alcanzar el tamaño del buffer en el receptor, lo que limitaría la tasa a $\frac{\texttt{rwnd}}{RTT}$

[§] J. Padhye y col. "Modeling TCP Reno performance: a simple model and its empirical validation". En: *Networking, IEEE/ACM Transactions on* 8.2 (2000), págs. 133-145

Rendimiento TCP en conexiones 'cortas'

- En los modelos anteriores se asume que la conexión TCP es lo suficientemente larga, con lo que llega al estado *Congestion Avoidance*
- En conexiones cortas (tráfico `http`) es más razonable asumir que estas no llegan a *Congestion Avoidance*, permaneciendo en *Slow Start*
- El crecimiento de la ventana de congestión es exponencial en cada RTT (crece un segmento por cada ACK nuevo recibido)
- Cardwell *et al*[§] analizan el comportamiento de TCP en estas condiciones, estudiando el tiempo necesario para transmitir M bytes de información

[§] Neal Cardwell, Stefan Savage y Tom Anderson. *Technical Report: Modeling the Performance of Short TCP Connections*. Inf. téc. University of Washington, 1998

Rendimiento TCP en conexiones 'cortas'

Canal ideal

- Inicialmente se asume que el canal es ideal, por lo que no se producen pérdidas
- Se asume el uso de *Delayed ACK*, enviando un ACK cada b segmentos de datos recibidos
- En esas circunstancias la progresión de la ventana de congestión `cwnd` se rige por la siguiente expresión

$$\texttt{cwnd}_{i+1} = \texttt{cwnd}_i + \texttt{acks}_i = \texttt{cwnd}_i + \frac{\texttt{cwnd}_i}{b} = \\ = \left(1 + \frac{1}{b}\right) \texttt{cwnd}_i = \gamma \cdot \texttt{cwnd}_i$$

- Como se pretenden transmitir M `bytes` se necesitarán $\frac{M}{MSS}$ segmentos

Rendimiento TCP en conexiones 'cortas'

Canal ideal

- ¿Cuántos RTT (ciclos de `cwnd`) son necesarios?

$$\frac{M}{MSS} = \sum_{k=1}^{N} \gamma^k \cdot w_0 = w_0 \frac{\gamma^N - 1}{\gamma - 1} \quad \rightarrow$$

$$\rightarrow \quad N = \log_\gamma \left[\frac{M(\gamma - 1)}{MSS \cdot w_0} + 1 \right]$$

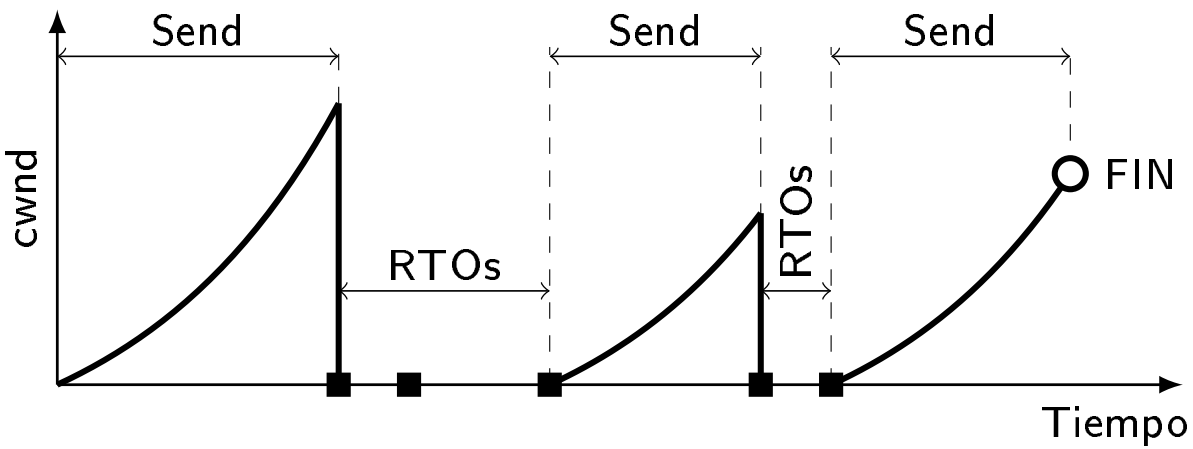
- Luego el tiempo total necesario pará transmitir los M `bytes` será

$$T_{\text{total}} = RTT \cdot \log_\gamma \left[\frac{M(\gamma - 1)}{MSS \cdot w_0} + 1 \right] + \underbrace{RTT}_{\text{3-way handshake}} + \underbrace{t_{Delayed\ ACK}}_{\text{1er segmento}}$$

Rendimiento TCP en conexiones 'cortas'

Canal con errores

- Se supone que en el periodo en el que se transmite el fichero se van sucediendo dos tipos de intervalos
  - Intervalos de transmisión, en los que TCP se encuentra en el estado *Slow Start*: crecimiento exponencial de `cwnd`
  - Intervalos de timeout, en el que hay una o más retransmisiones por expiración de *RTO* consecutivas



Rendimiento TCP en conexiones 'cortas'

Canal con errores

- El número de intervalos de transmisión, v es igual al número de intervalos de timeout, u más 1: $v = u + 1$
- La probabilidad de pérdida se puede calcular en función del número de segmentos que hay que transmitir, $s = \frac{M}{MSS}$ y de los que se pierden, l

$$p = \frac{l}{s + l} \quad \rightarrow \quad l = \frac{s \cdot p}{1 - p}$$

- Probabilidad de que una pérdida origine un RTO (Padhye)

$$Q = \begin{cases} 0 & p = 0 \\ \min\left\{1, \frac{3}{\sqrt{\frac{8}{3\,b\,p}}}\right\} & p > 0 \end{cases}$$

Rendimiento TCP en conexiones 'cortas'

Canal con errores

- Con lo que se puede calcular el número total de timeouts en la conexión: $to = l \cdot Q$
- Además el número de pérdidas consecutivas se puede modelar como una variable aleatoria geométrica, con media $\frac{1}{1-p}$
- Por tanto se puede calcular el número de intervalos de timeout que se producen en la conexión

$$u = \frac{to}{\frac{1}{1-p}} = l \cdot Q \cdot (1-p) \qquad v = u + 1$$

- Con lo que se puede calcular la cantidad de segmentos que se transmite en un intervalo de transmisión 'promedio', $e = \frac{s+l}{v}$

Rendimiento TCP en conexiones 'cortas'

Canal con errores

- Llegando, finalmente, a la duración de dicho intervalo 'promedio'

$$t_{\text{tx}} = RTT \cdot \log_{\gamma}\left[\frac{e\,(\gamma-1)}{w_0}+1\right]$$

- Por otra parte, Padhye también estima la duración media de los intervalos de timeout

$$t_{\text{tout}} = T_0 \frac{1+p+2p^2+4p^3+8p^4+16p^5+32p^6}{1-p}$$

- Con lo que el tiempo total se podría estimar como sigue

$$T_{\text{total}} = v \cdot t_{\text{tx}} + u \cdot t_{\text{tout}} + \underbrace{RTT}_{\text{3-way handshake}} + \underbrace{t_{Delayed\ ACK}}_{\text{1er segmento}}$$

Reparto de la capacidad entre flujos TCP

- Además del funcionamiento 'aislado' de los algoritmos de control de congestión de TCP también es relevante el estudio del reparto de recursos entre varios flujos
- ¿Cómo se reparte la capacidad de un enlace (*cuello de botella*) por el que pasan varias conexiones TCP, cada una con caminos *end-to-end* diferentes?
- Si la capacidad del enlace es R y hay K conexiones, un reparto *equitativo* asignaría a cada flujo una capacidad $\approx \frac{R}{K}$
- Los algoritmos de control de congestión de TCP se comportan de manera adecuada cuando las condiciones (en concreto, el RTT) son similares

Reparto de la capacidad entre flujos TCP

- Cuando el *RTT* de los flujos es diferente, el reparto deja de ser apropiado
  - Los flujos con menor *RTT* acaparan la capacidad de manera más rápida, ya que su ventana de congestión crece a un mayor ritmo
  - El problema es más relevante si se comparte la capacidad con conexiones que usen otros protocolos de transporte, como UDP, que no usa ningún control de congestión
  - Otro problema aparece con aplicaciones que inician varias conexiones TCP (navegación web), ya que el reparto sería poco equitativo (desde el punto de vista de la aplicación)

Reparto de la capacidad entre flujos TCP

Índice de Jain

- Chiu y Jain propusieron[§] un parámetro (*Jain's fairness index*) para 'medir' lo equitativo del reparto de los recursos de red

$$F = \frac{\left(\sum_{i=1}^{k} f_i\right)^2}{k \cdot \sum_{i=1}^{k} f_i^2}$$

- F está acotado entre 0 y 1
- Si el reparto es completamente equitativo (todas las f_i's iguales), $F = 1$
- Si un flujo 'acapara' todos los recursos $F = \frac{1}{k}$, notar que $\lim_{k\to\infty} F = 0$

[§] Dah-Ming Chiu y Raj Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks". En: *Comput. Netw. ISDN Syst.* 17.1 (jun. de 1989)