Redes de Computadores

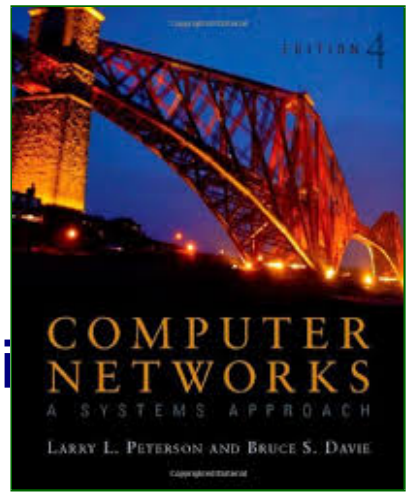
Tema 6 – QoS y Control de Congestión

Natalia Ayuso, Juan Segarra y Jesús Alastruey

Departamento de
Informática e Ingeniería
de Sistemas

Universidad Zaragoza

Índice



Capítulo 6

1 Introducción

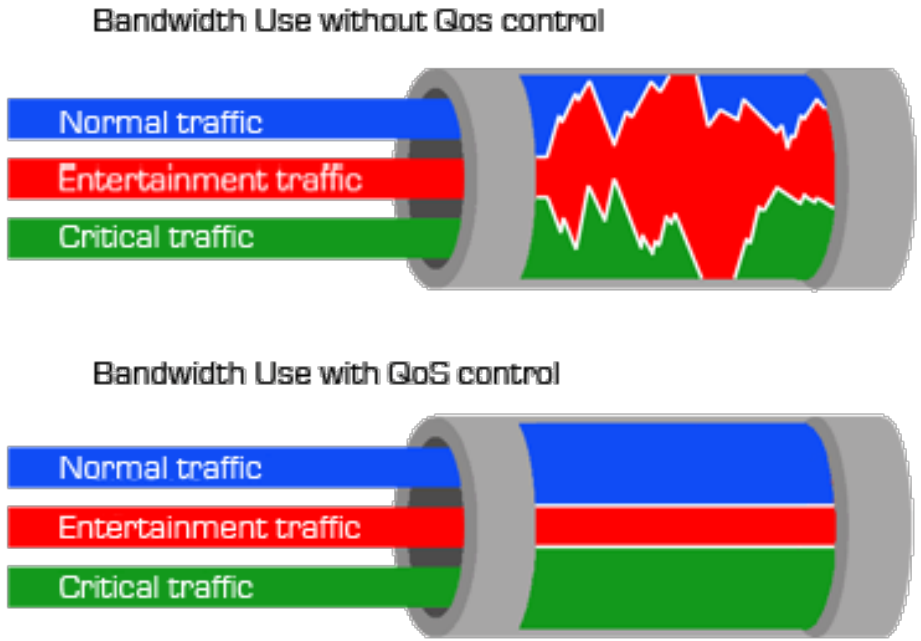
- La red tiene una serie de recursos compartidos: ancho de banda de enlaces, buffers en routers y switches
- Cuando muchos paquetes compiten en un router por un mismo enlace, la memoria del router puede agotarse y producirse pérdidas de paquetes
- Si las pérdidas de paquetes son frecuentes, se dice que la red está *congestionada*

Objetivos:

1. Repartir los recursos para que todos perciban la mayor *calidad de servicio* posible
2. Reducir la pérdida de paquetes mediante mecanismos de *control de congestión*

2 Calidad de Servicio (QoS)

- *Calidad de servicio (QoS)*: rendimiento promedio percibido por los usuarios, o más específicamente en redes, los mecanismos para proporcionar dicho rendimiento
  - Reserva de recursos
  - Clasificación de paquetes y tratamiento en contexto



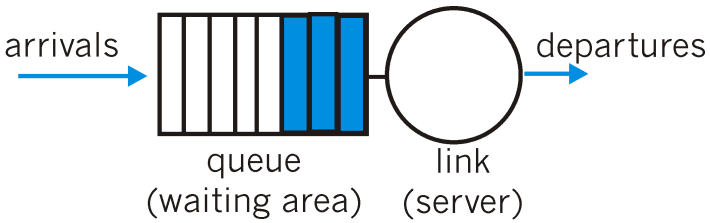
2 Calidad de Servicio (QoS) (II)

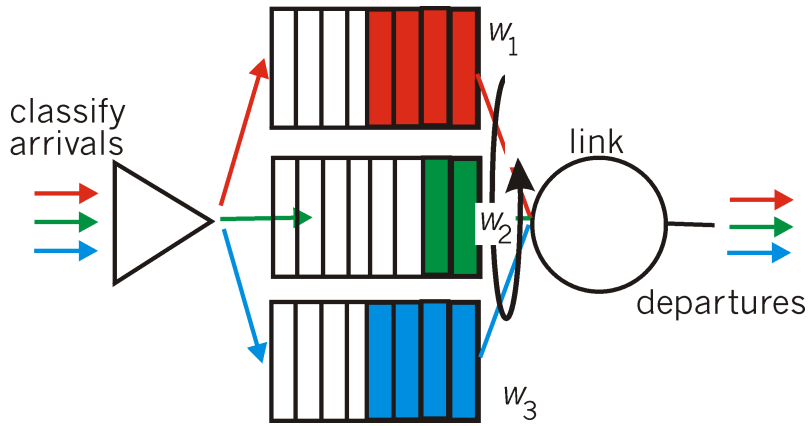
La calidad percibida depende de nuestro tráfico *y del tráfico del resto del mundo*. Opciones:

- No dar ningún tipo de calidad de servicio
  - Los paquetes son encolados y procesados por estricto orden de llegada al encaminador (FIFO, *First-In-First-Out*)
  - Sencillo y eficaz con poco tráfico
- Ofrecer QoS «perfecta»
  - Negociar entre red y aplicación el tráfico que se va a enviar
  - Reservar recursos de red para tratar ese tráfico
  - Denegar el uso de la red cuando no sea posible proporcionar los recursos necesarios
  - Complejo (lento), calidad independiente del resto del tráfico
- Cualquier opción intermedia entre las dos anteriores

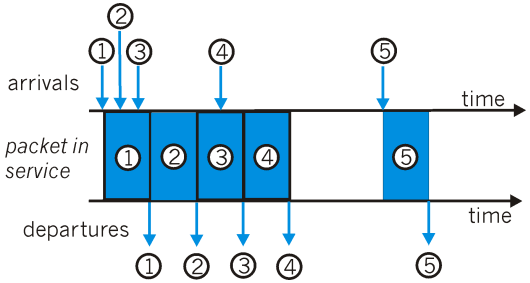
2.1 Disciplina de colas

- *FIFO*: una única cola en el encaminador
  - No proporciona ningún tipo de calidad de servicio
- *Colas equitativas* (*Fair Queuing, FQ*)
  - Varias colas FIFO en el encaminador
  - Las colas se atienden de forma equitativa (*Round-Robin*)
  - Al llegar un paquete, se clasifica y se mete en la cola correspondiente
    - ¿Qué política se sigue para clasificar? [p. 8][p. 9]
  - Colas equitativas *ponderadas* (WFQ): se asigna un peso (w_i) a cada cola

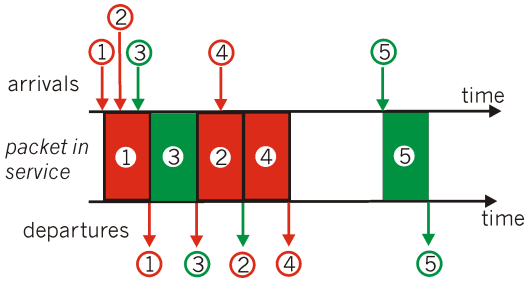




2.1 Disciplina de colas (II)

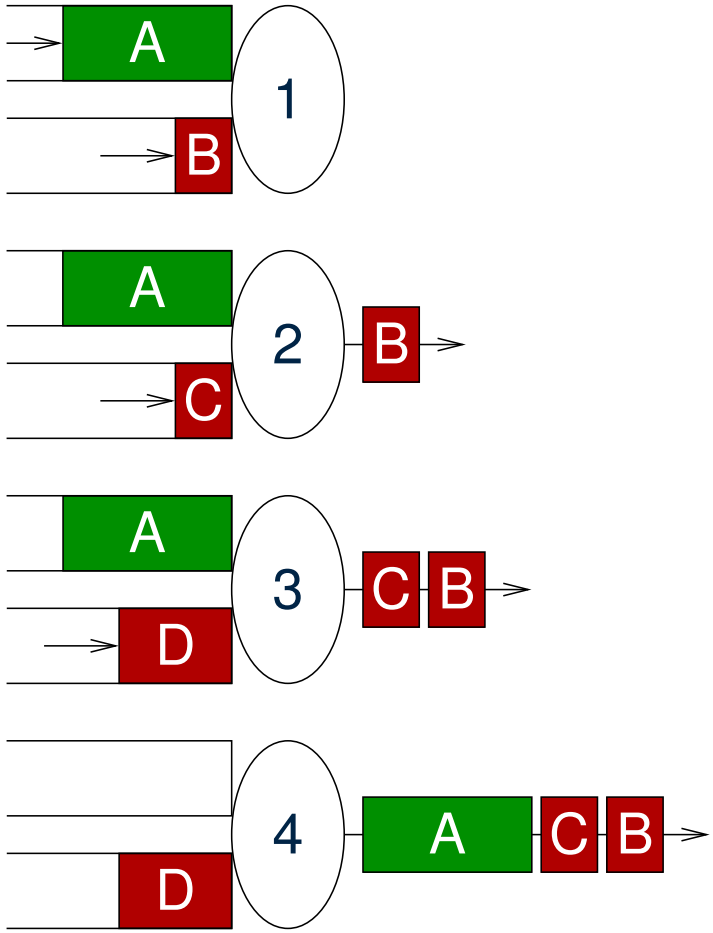


FIFO



FQ

Virtual Finish Time=
Virtual Start Time +
Virtual Service Time



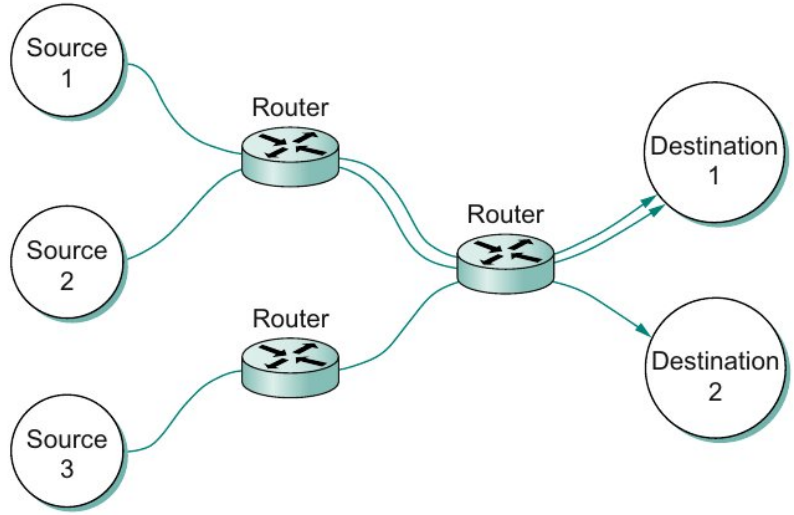
FQ con tamaños distintos

2.2 Servicios diferenciados (*DiffServ*)

- Clasificar paquetes y su correspondiente asignación a colas de encaminadores por el *tipo de tráfico* que contengan, es decir, *diferenciar servicios*. Por ejemplo:
  - Paquetes «Premium»
  - Paquetes «Best effort» (sin garantías)
- IPv4: campo «TOS»
- IPv6: campo «TrafficClass»
- El valor de dicho campo puede ser establecido por el origen, por el encaminador de entrada a un SA, etc.
- La gestión por parte del encaminador es sencilla
- No permite dar un servicio particularizado

2.3 Servicios integrados (*IntServ*)

- Clasificación de paquetes y su correspondiente asignación a colas de encaminadores por *flujo* al que pertenezcan
  - E.g. paquetes de transmisión de vídeo entre x y y
  - IPv4 debe analizar IPs y puertos
  - IPv6 incorpora «FlowLabel» para facilitar la clasificación



- Combina colas WFQ con circuitos virtuales o protocolos de *reserva de recursos*
  - La aplicación especifica a la red sus requisitos de tráfico. La red o los garantiza o deniega el servicio
  - Gestión compleja en el encaminador: necesita mantener mucha información (una cola/circuito/reserva por flujo)

2.3 Ejemplo

- Índice de velocidades de ISP para Netflix España

SPAIN

ISP LEADERBOARD - DECEMBER 2016

RANK	ISP	SPEED Mbps		PREVIOUS Mbps	RANK CHANGE	TYPE
1	Telecable	3.86		3.68		Cable, Wireless
2	Vodafone	3.76		3.65	+1	Fiber, DSL
3	ONO	3.73		3.67	-1	Cable, DSL
4	Orange	3.73		3.62		Fiber, DSL
5	Jazztel	3.70		3.60		Fiber, DSL
6	Euskaltel	3.62		3.49	+1	Cable, DSL, Wireless
7	R	3.27		3.53	-1	Fiber, Cable, Wireless
8	Telefonica - Movistar	2.54		2.42		Fiber, DSL, Wireless

Fiber Cable DSL Satellite Wireless

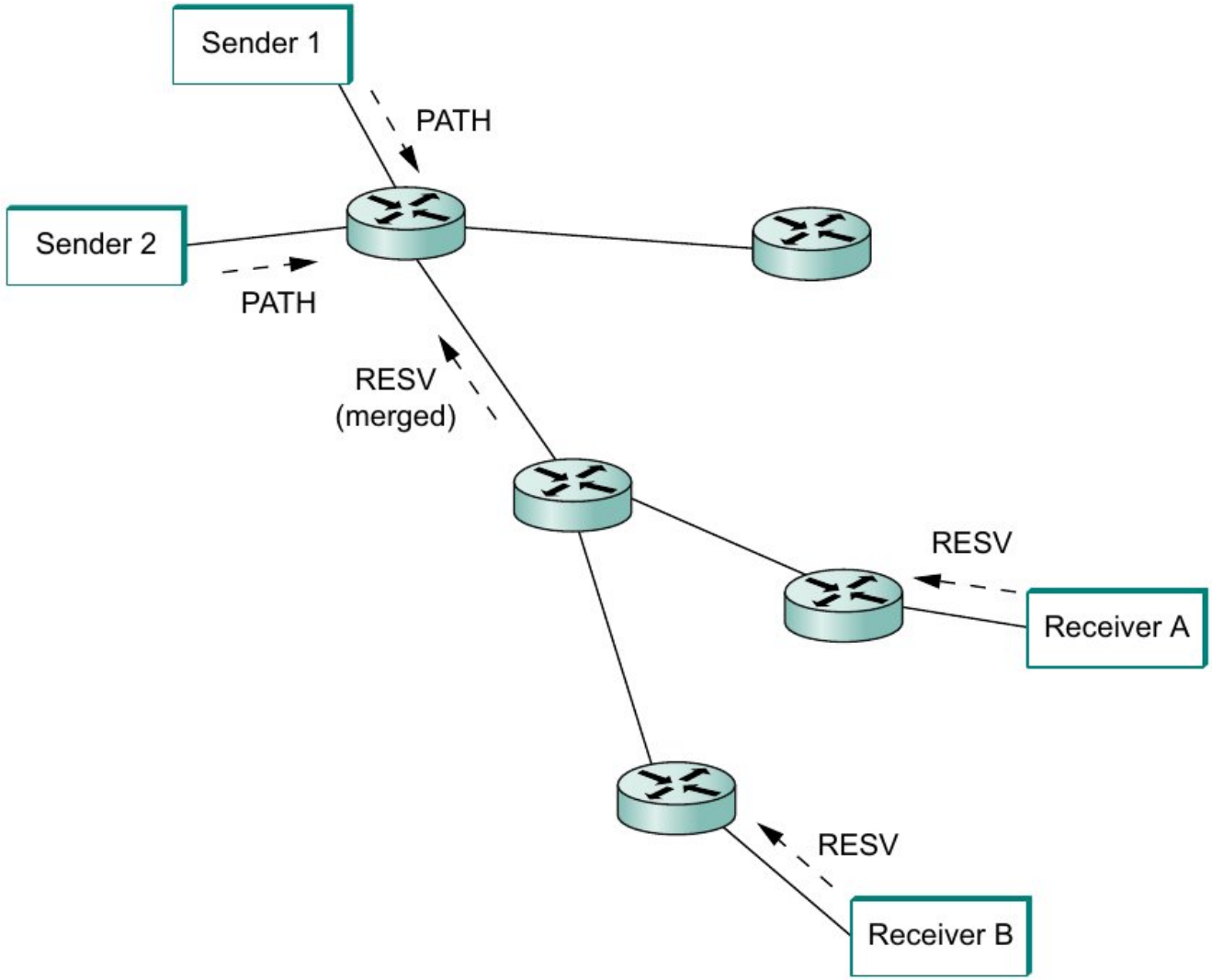
Imagen: Comunidad Movistar

2.3.1 Protocolo RSVP

- Resource ReSerVation Protocol (RSVP)
- Diseñado para funcionar sobre IP
- Utiliza refresco periódico como alternativa a los circuitos virtuales
- Permite cambios dinámicos de configuración
- Diseñado para ofrecer multicast
  - Fusiona requerimientos de receptores
  - Puede especificar varios interlocutores
- Basado en mensajes periódicos PATH/RESV
  - PATH (origen $\rightarrow$ destino): recoge datos sobre recursos de encaminadores
  - RESV (destino $\rightarrow$ origen): reserva recursos

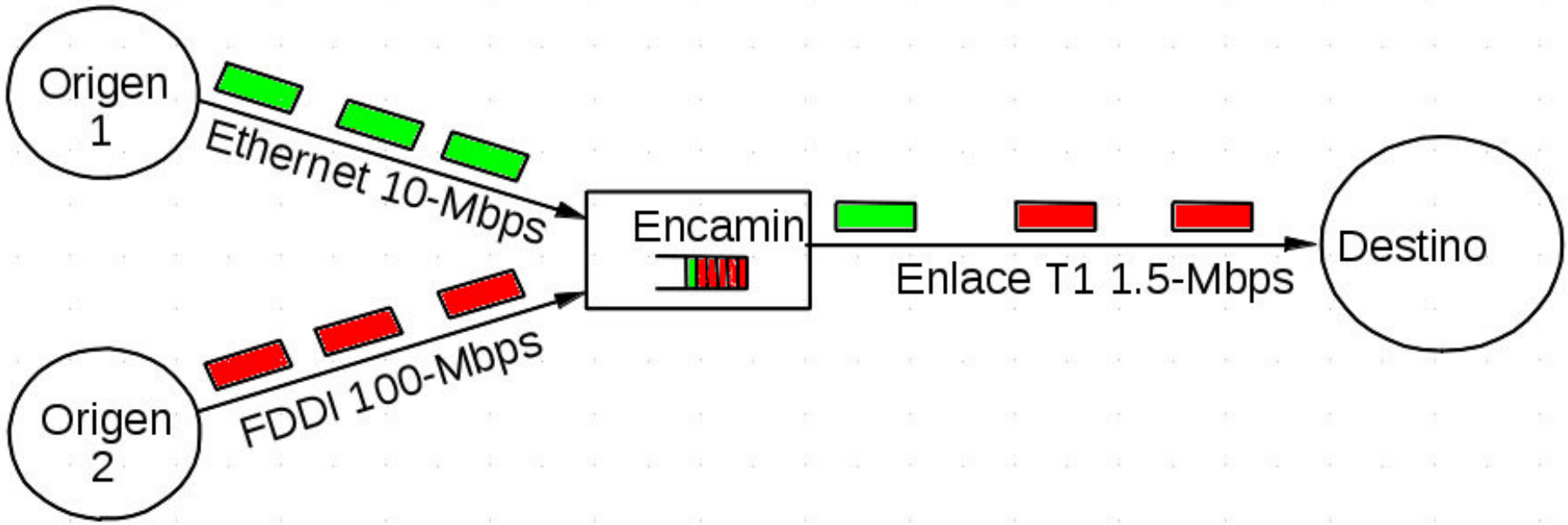
2.3.1 Protocolo RSVP (II)

E.g. Reservas en multicast



3 Control de congestión

- *Congestión*: pérdida frecuente de paquetes
  - Si un equipo recibe paquetes a más velocidad de la que los puede reenviar, su buffer se puede desbordar
  - Mecanismos de *control de congestión* para tratarla



3 Control de congestión (II)

- La tasa a la que llega el tráfico depende de todas las capas
  - Tiempos de espera aleatorios en redes locales
  - Tiempo en colas de encaminadores en capa de red
  - Velocidad efectiva de tráfico dependiente de ventana, retransmisiones, etc. en capa de transporte

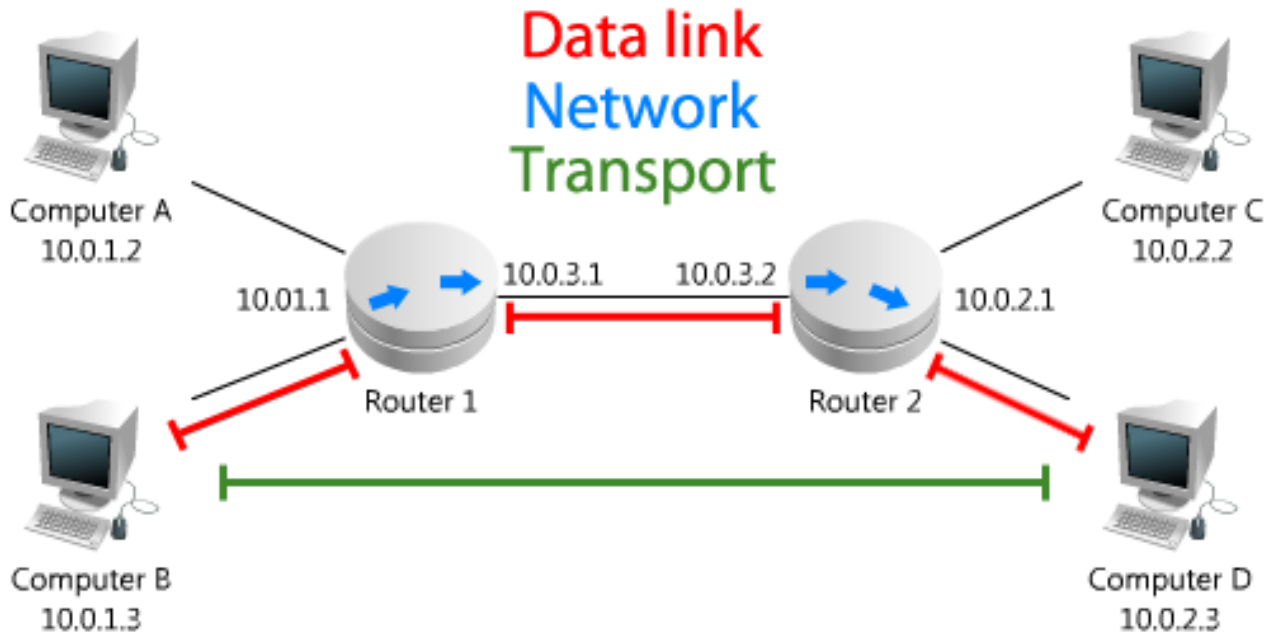


Imagen: OSI model – Layer 4: Transport (TCP and UDP with Scapy)

3 Control de congestión (III)

- ¿Cómo actúa cada capa para tratar la congestión?
  - Independientemente $\rightarrow$ peligro de entorpecerse
  - Coordinadas $\rightarrow$ se pierde la abstracción al asumir que por encima/debajo se actúa de cierta forma
- Mecanismos de control de congestión desde . . .
  - TCP: Tahoe (*comienzo lento y prevención de congestión*), Reno (*retransmisión/recuperación rápida*), Vegas, SACK, BIC/CUBIC, Compound, etc.
  - Capa de red: *DECbit/ECN*, *RED*, etc.

3 Control de congestión en TCP

- Una ventana de emisión demasiado pequeña no utiliza el ancho de banda disponible
- Una ventana de emisión demasiado grande puede congestionar la red, cuyo efecto es más perjudicial que usar una ventana pequeña
- Originalmente la tasa de emisión estaba limitada solamente por el control de flujo (ventana anunciada por el receptor)
  - Paquetes descartados por encaminadores $\rightarrow$ emisores retransmiten $\rightarrow$ congestión de la red
  - 1986, colapso por congestión de la red NSF: 32 Kbps $\rightarrow$ 40 bps
- Solución: algoritmos de control de congestión desarrollados por Van Jacobson
  - Nodos ajustan su tasa de emisión basándose en eventos de la red (pérdidas ...)

3 Control de congestión en TCP (II)

- RFC 5681 (TCP-Reno) describe 4 algoritmos
  - Comienzo lento, *slow start* (SS)
  - Prevención de congestión, *congestion avoidance* (CA)
  - Retransmisión rápida, *fast retransmit*
  - Recuperación rápida, *fast recovery*
- Desarrollados en [Jac88] y [Jac90]
- Asumen que las pérdidas de paquetes son debidas a congestión

[Jac88] Jacobson, V., **Congestion Avoidance and Control**, Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[Jac90] Jacobson, V., **Modified TCP Congestion Avoidance Algorithm**, end2end-interest mailing list, April 30, 1990. ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail

3.1 Slow Start + Congestion Avoidance

- Cuando se establece una conexión TCP, los extremos no conocen ni su ancho de banda (BW) ni su latencia (RTT)

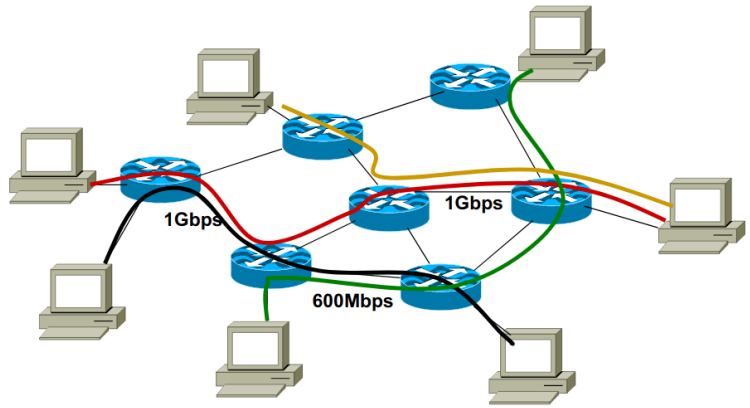


Imagen: Sylvia Ratnasamy. TCP: Congestion Control. CS 168, Univ. Berkeley.

- Si el emisor envía demasiados segmentos, puede congestionar la red y provocar pérdidas
- El emisor ajusta dinámicamente su ventana de emisión
  - Comienza enviando ventanas de pocos segmentos
  - Si no hay pérdidas, incrementa el tamaño de la ventana
  - Si hay pérdidas, reduce el tamaño de la ventana

3.1 Slow Start + Congestion Avoidance (II)

- Velocidad de incremento de la ventana
  - Exponencial: fase descubrimiento de BW
    $\rightarrow$ comienzo lento
  - Lineal: fase ajuste por las variaciones del BW
    $\rightarrow$ prevención de congestión

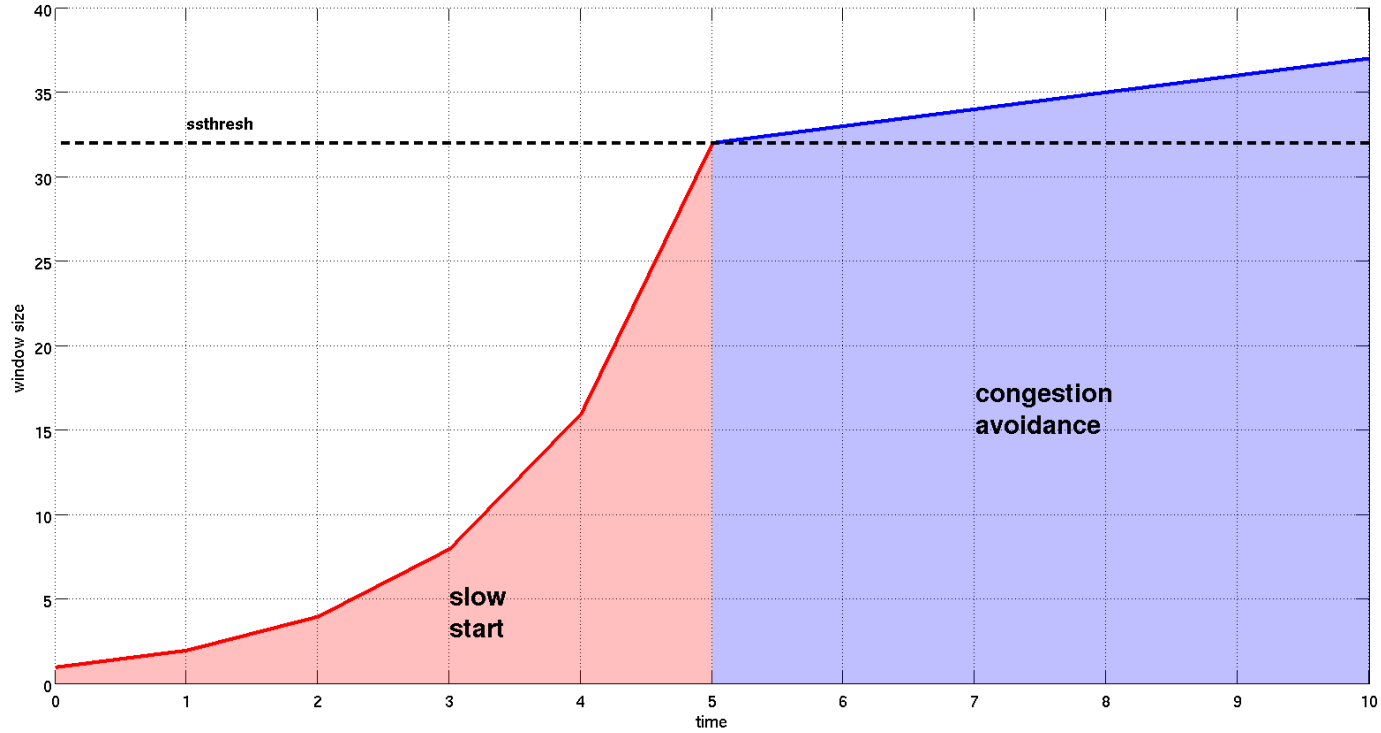


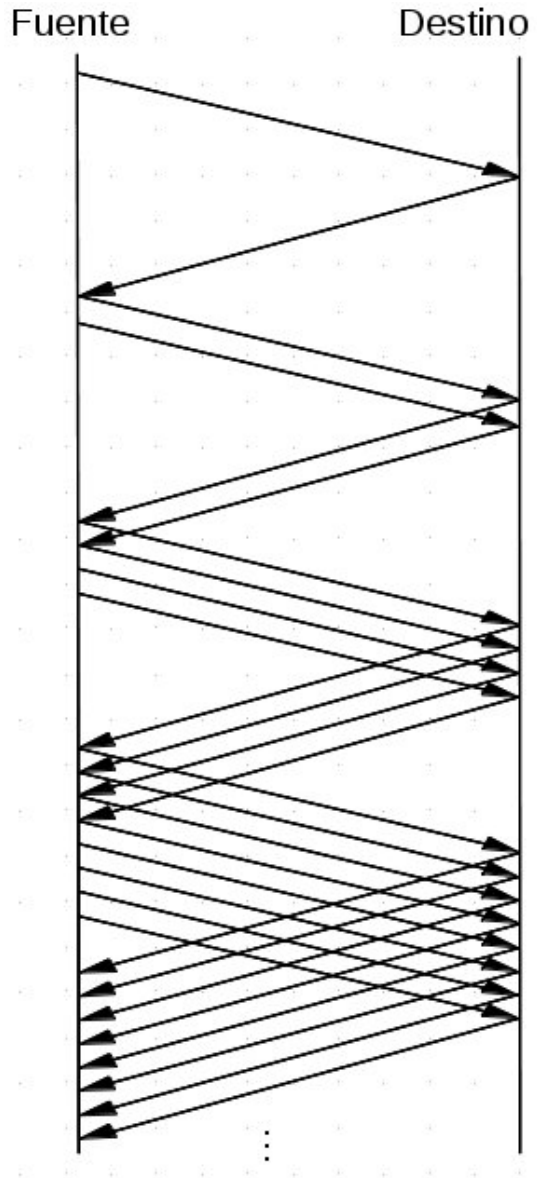
Imagen: elaboración propia

3.1 SS + CA: implementación

- Variables por conexión TCP
- *Ventana de congestión*, $cwnd$
  - Bytes que pueden enviarse sin desbordar routers
  - Calculada por emisor
- *Ventana de recepción*, $rwnd$
  - Bytes que pueden enviarse sin desbordar al receptor
  - Comunicada por receptor a emisor en cada ACK
- *Umbral comienzo lento*, $ssthresh$
  - Determina si la emisión es controlada por el algoritmo de comienzo lento ($cwnd < ssthresh$) o el de prevención de congestión ($cwnd \geq ssthresh$)
  - Calculado por emisor
- Transmisión de datos limitada por $min(cwnd, rwnd)$
- Eventos: ACK, dupACK (ACK duplicado), RTO

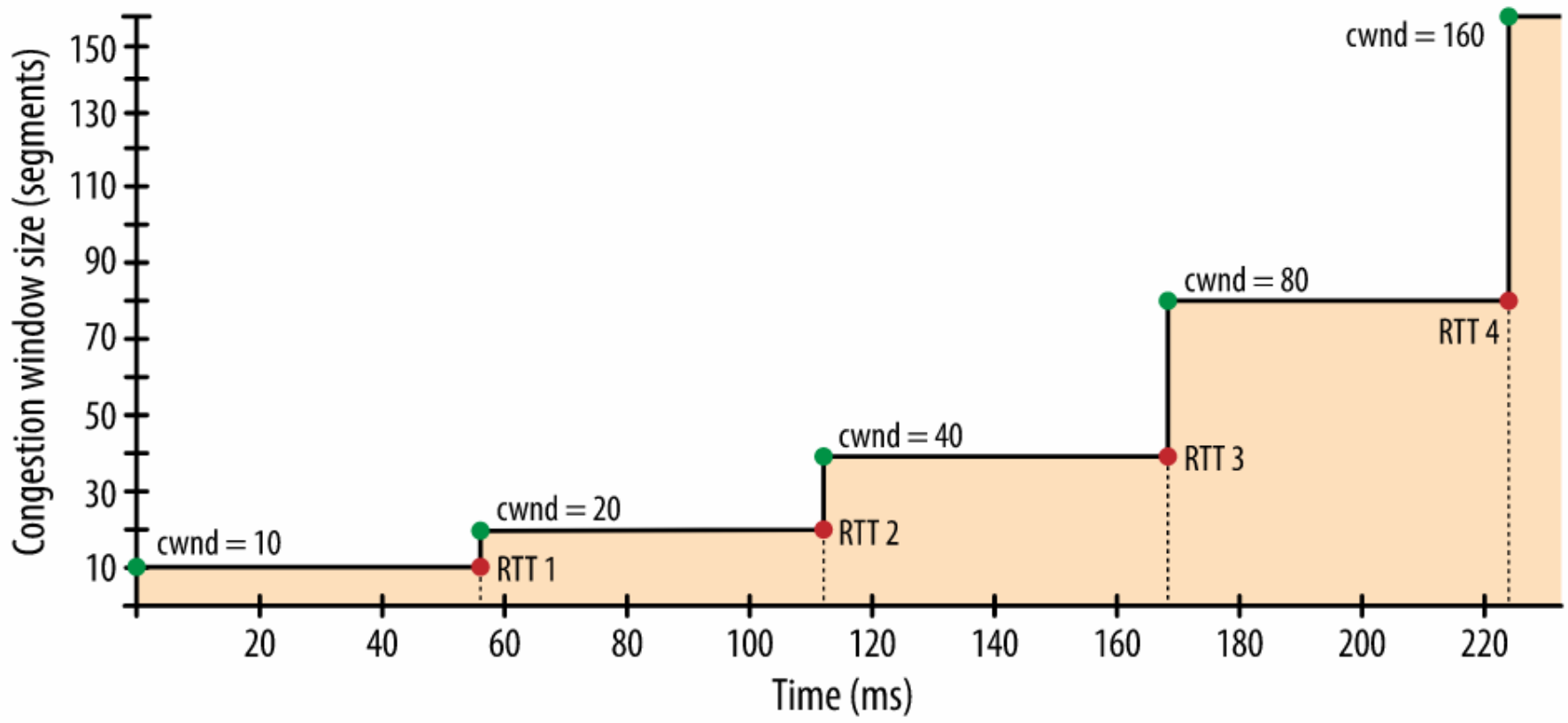
3.1 Comienzo lento

- Objetivo: estimar BW
- Inicializar $cwnd$ a número reducido de segmentos: IW, *initial congestion window*:
  - Enero-1997: $IW = 1$, RFC 2001
  - Abril-1999: $IW = 4$, RFC 2581
  - Abril-2013: $IW = 10$, RFC 6928
  - Large-Scale Scanning of TCP's Initial Window
- Aumentar $cwnd$ de forma rápida
  - Cada ACK: $cwnd = cwnd + 1$
    $\rightarrow$ cada RTT: $cwnd = 2 * cwnd$



3.1 Comienzo lento: ejemplo

- Contexto:
  - $IW = 10\ segmentos$, $RTT = 56\ ms$
- Evolución temporal de $cwnd$



Fuente: High Performance Browser Networking: Building Blocks of TCP

3.1 Comienzo lento: ejercicio

✍ Dado un enlace TCP de las siguientes características:

➤ $rwnd = 64\ KiB$, $IW = 1\ segmento$, $RTT = 56\ ms$

Calcula el tiempo necesario para que la ventana de congestión ($cwnd$) alcance el valor de la ventana anunciada de recepción ($rwnd$).
Repetir el cálculo con $IW = 4$ y $IW = 10\ segmentos$

3.1 Prevención de congestión

- Objetivo: ajustar ventana de congestión según las variaciones del BW
- Si no hay pérdidas, aumentar $cwnd$ de forma lineal
  - Cada ACK: $cwnd = cwnd + 1/cwnd$
    $\rightarrow$ cada RTT: $cwnd = cwnd + 1$
- Si hay pérdidas (vence RTO), reinicializar $cwnd$

3.1 SS+CA: cambio de fase

- ¿Cuándo se cambia de comienzo lento a prevención de congestión? Cuando el valor de la ventana de congestión, $cwnd$, alcanza el umbral de comienzo lento, *slow start threshold* ($ssthresh$)
- Evolución de $ssthresh$
  - Se inicializa a valor elevado
  - En caso de pérdida (RTO) $\rightarrow ssthresh = cwnd/2$

3.1 SS + CA: recapitulando

- Inicializar variables
  - $ssthresh = \infty$
  - $cwnd = IW = 1/4/10$ segmentos,
    según RFC 2001/RFC 2581/RFC 6928
- Inicio lento
  - Cada ACK: $cwnd = cwnd + 1 \rightarrow$ cada RTT: $cwnd = 2 * cwnd$
- Si vence temporizador (RTO)
  - $ssthresh = cwnd/2$
  - Reinicializar $cwnd \rightarrow cwnd = IW$
  - Comienzo lento mientras $cwnd < ssthresh$
  - Cuando $cwnd \geq ssthresh \rightarrow$ prevención de congestión,
    cada ACK: $cwnd = cwnd + 1/cwnd$
    $\rightarrow$ cada RTT: $cwnd = cwnd + 1$

3.1 SS + CA: ejemplo

- Evolución temporal de la ventana de congestión

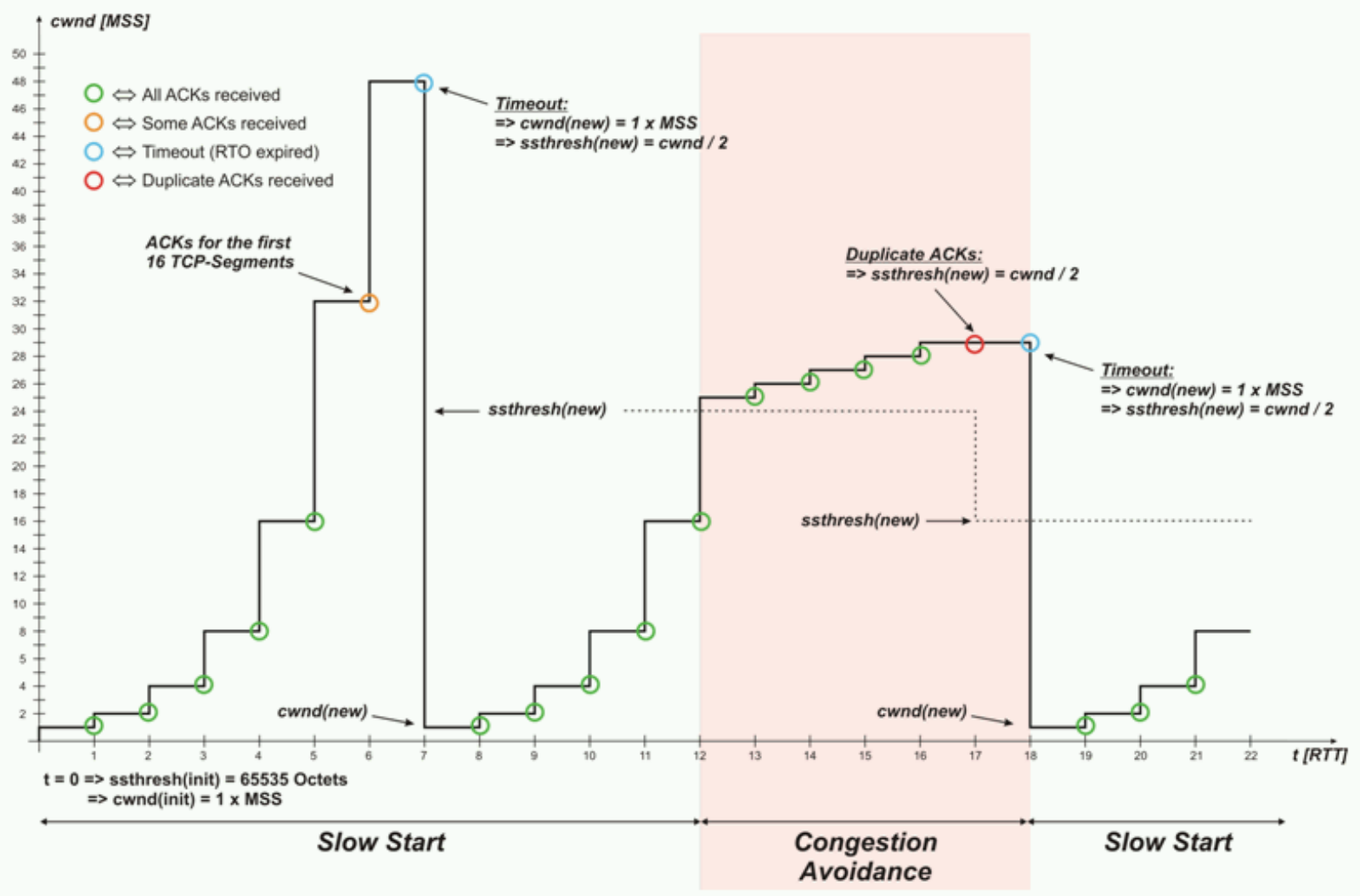
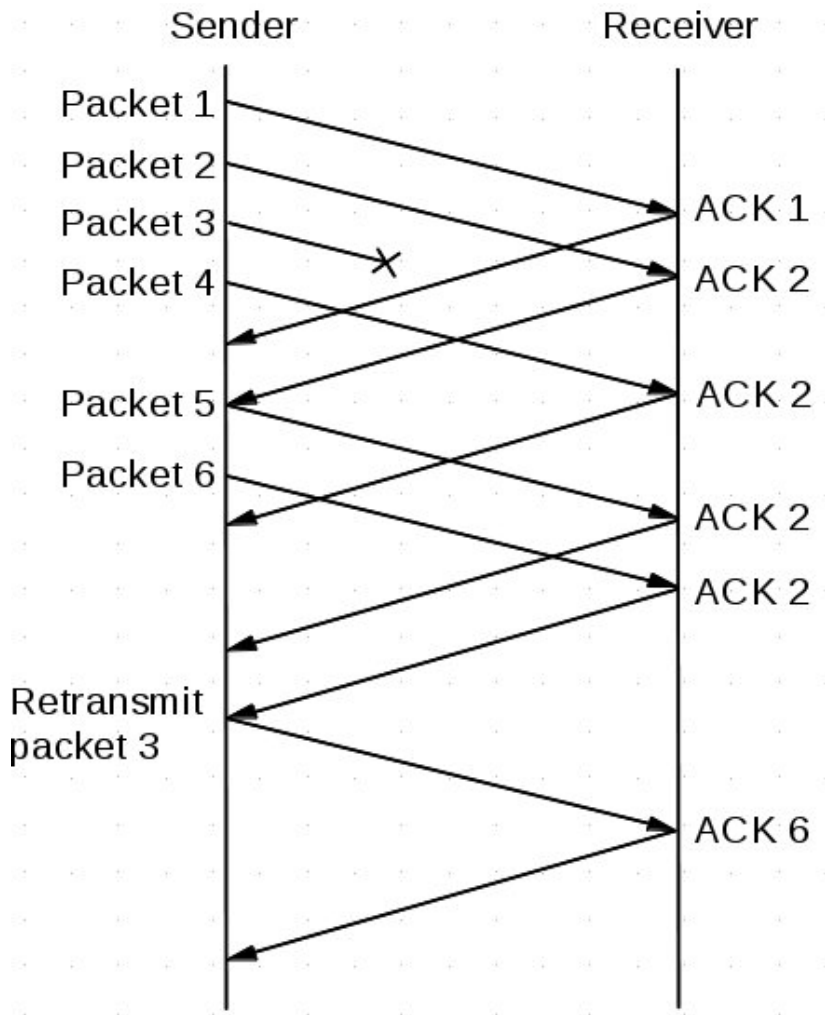


Imagen: INACON Protocol Help

3.2 Retransmisión rápida

- Objetivo: anticipar detección de pérdida de segmentos
- El receptor envía dupACK cuando recibe un segmento fuera de orden
- El emisor retransmite el segmento perdido cuando haya recibido 3 dupACKs, sin esperar a que venza RTO



3.3 Recuperación rápida

- Gestión desde la retransmisión rápida de un segmento perdido hasta la recepción de un ACK no duplicado
- Tras recibir 3 dupACKs: retransmisión + paso a modo prevención de congestión
  - $ssthresh = cwnd/2$, $cwnd = ssthresh + 3$
  - Por cada dupACK adicional: $cwnd = cwnd + 1$
- Tras recibir ACK: fin de recuperación rápida
  - $cwnd = ssthresh$

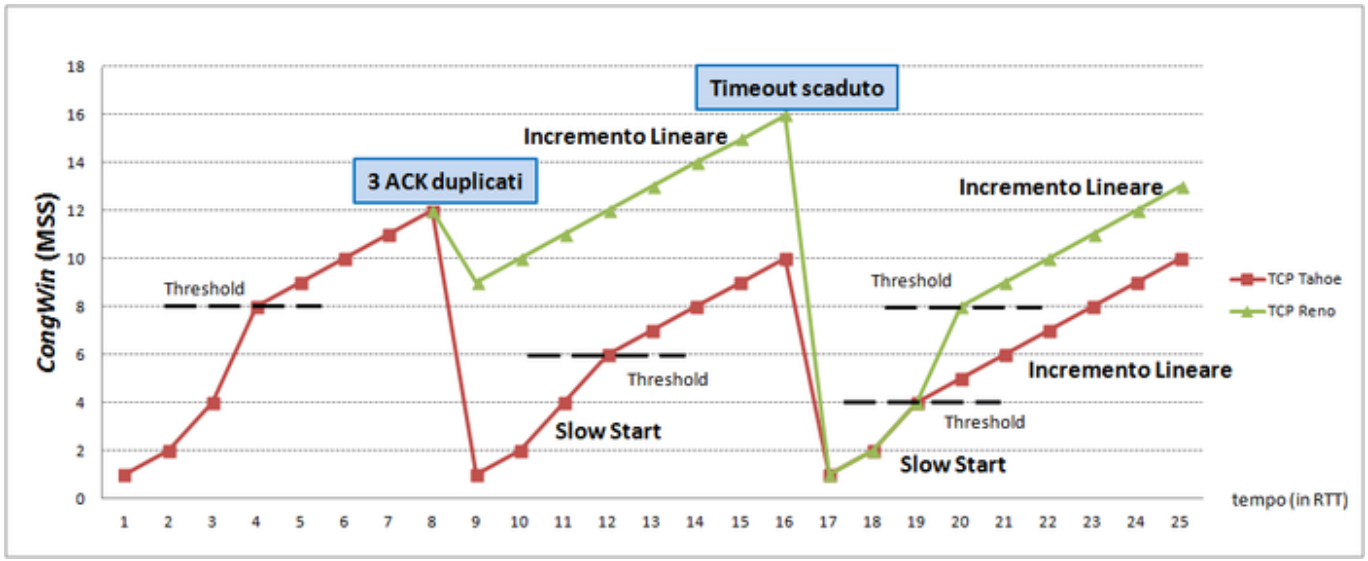
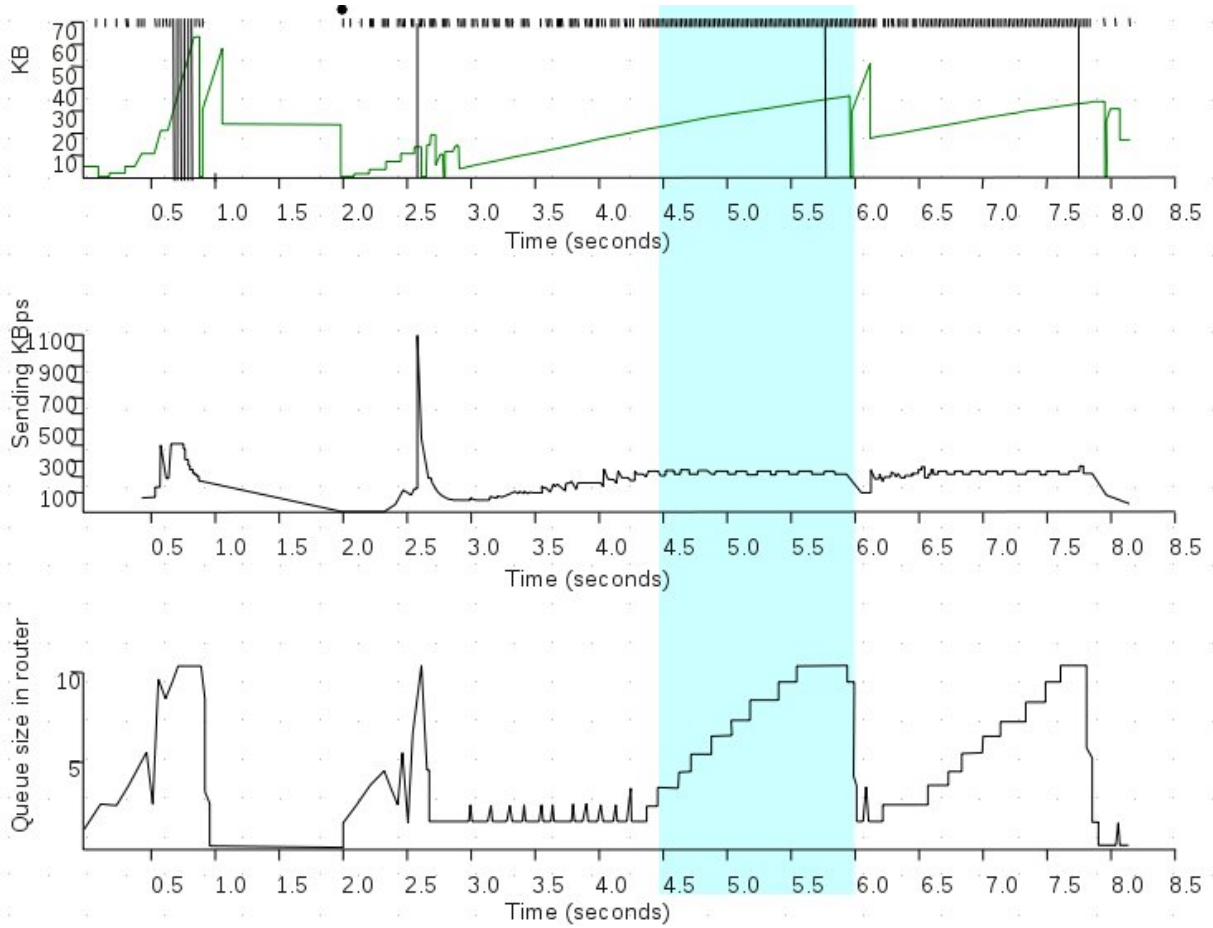


Imagen: Wikipedia: Controllo della congestione in TCP

3.4 Prevención desde origen: TCP Vegas

- Control de congestión basado en retardo, no en pérdidas
- Emisor monitoriza RTT para detectar crecimiento de las colas en los encaminadores $\rightarrow$ proximidad de congestión

- Si aumenta RTT $\rightarrow$ crece ocupación de las colas $\rightarrow$ $cwnd--$ cada RTT
- Si no aumenta RTT $\rightarrow$ $cwnd++$ cada RTT



3.5 Detección Explícita de Congestión

- DECbit (*Detection Explicit Congestion*)
- Detección en capa de red; control en capa de transporte
- Encaminador monitoriza la ocupación media de cola
  - Marca *bit de congestión* en la cabecera del paquete si $ocup\_media\_cola > umbral$
- Destino reenvía a origen el valor del bit
- Origen ajusta su tasa de envío en base a los bits marcados
  - $<$ 50 % de última ventana $\rightarrow$ incr. ventana en 1 paq.
  - $\geq$ 50 % o más con bit $\rightarrow$ decr. ventana en 0.875
- RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP

3.6 Detección temprana aleatoria (RED)

- *Random Early Detection* (RED), RFC 2309
- Detección y control de congestión en capa de red
- Notifica descartando paquetes (coopera con TCP)
- Encaminador monitoriza longitud media de la cola a la llegada de cada paquete
  - Si $OcupMedia > UmbralMax$, descarta paquete
  - Si $OcupMedia > UmbralMin$, descarta paquete con una probabilidad P
  - Con cada paquete no descartado, incrementar P
- La probabilidad de descarte en un flujo es proporcional a su cantidad de tráfico
- Posibilidad de «castigar» a quienes no regulan congestión