

Gestión de cola

Area de Ingeniería Telemática
<http://www.tlm.unavarra.es>

Grado en Ingeniería en Tecnologías de
Telecomunicación, 3º

Control de congestión en TCP

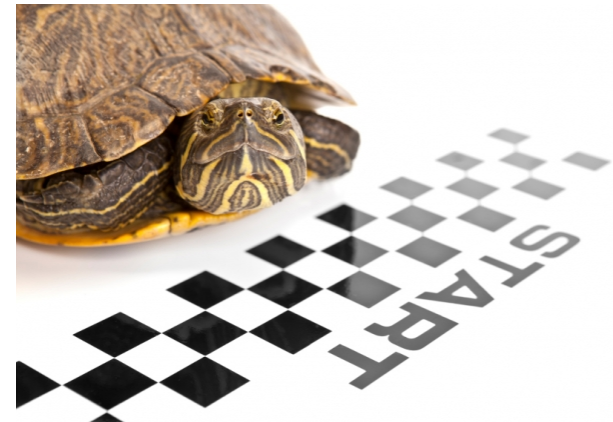
Congestion Avoidance

- Vamos a ver lo que se llama *Active Queue Management*
- Su objetivo es evitar que interfaces o colas se congestionen
- Diseñado para TCP pues es un protocolo que reacciona ante congestión reduciendo la tasa de envío
- Primero un recordatorio sobre *congestion avoidance* en TCP (solo el mecanismo básico)



Slow start

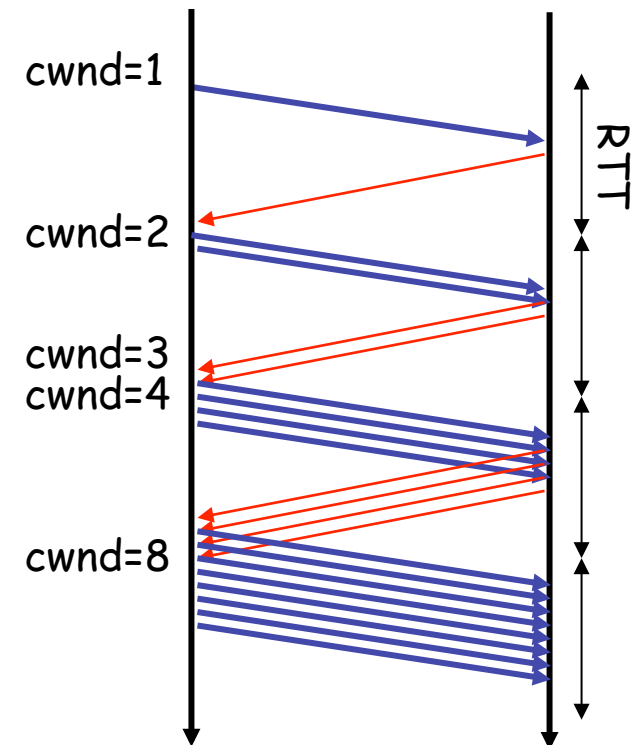
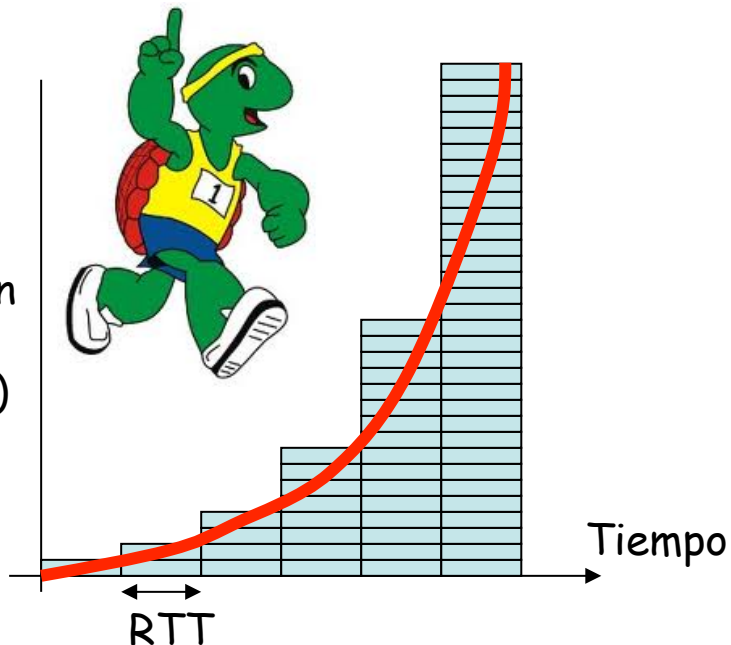
- TCP puede tener enviado sin confirmar la ventana de control de flujo (win)
- Inicialmente TCP no conoce las condiciones de la red
- Comienza enviando en modo muy conservador (1 paquete?)
- $cwnd = 1$ (*congestion window*)
- Puede tener enviado sin confirmar el **mínimo** entre $cwnd$ y la ventana de control de flujo
- Si ve que los datos llegan (recibe ACK) prueba a enviar más de golpe (tener más “en vuelo”)
- Para eso aumenta $cwnd$



Slow start

- TCP incrementa cwnd en como mucho SMSS por cada ACK que confirma nuevos datos
- Hasta que alcance/supere ssthresh o se detecte una pérdida
- ssthresh (slow start threshold) comienza siendo grande

Número de
segmentos
que se envían
(IW=1, no
delayed-ack)



Congestion avoidance

- Inicialización: **cwnd**=1, **ssthresh**=65535
- *Slow Start* (...)
- Si se detecta una pérdida (...):
 - **ssthresh** = mín{ cwnd, win }/2
 - [RFC2581] ssthresh = máx {FlightSize/2, 2} (...)

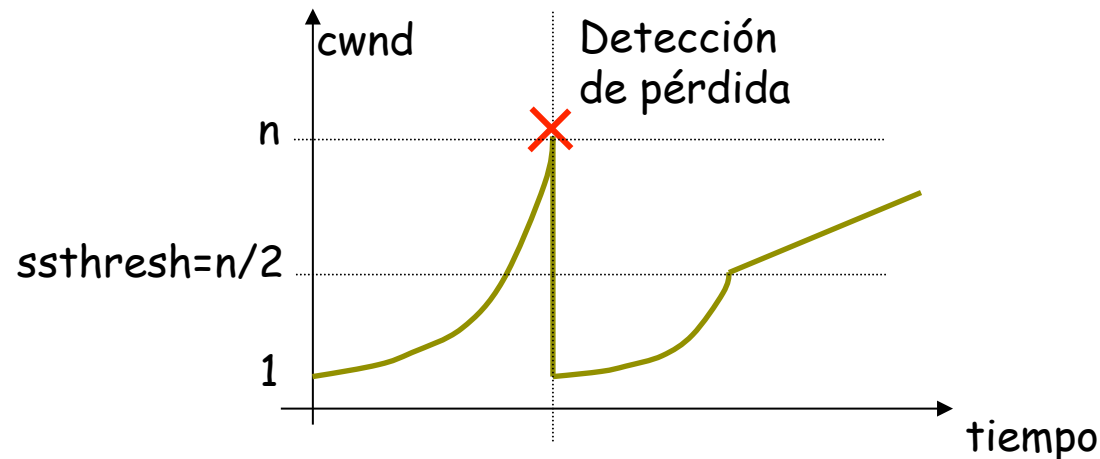
TCP Tahoe

Por **Timeout**:

- cwnd=1, comienza *Slow-start* de nuevo (...)

Nuevos datos son confirmados:

- Si $cwnd \leq ssthresh$: *Slow-start*, $cwnd += 1$ (crecimiento exp.) (...)
- Si $cwnd > ssthresh$: **Congest. avoidance**, $cwnd += 1/cwnd$ (crecimiento lineal) (...)

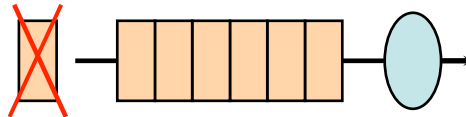


Gestión pasiva de cola

Passive Queue Management

Drop-Tail (la más habitual)

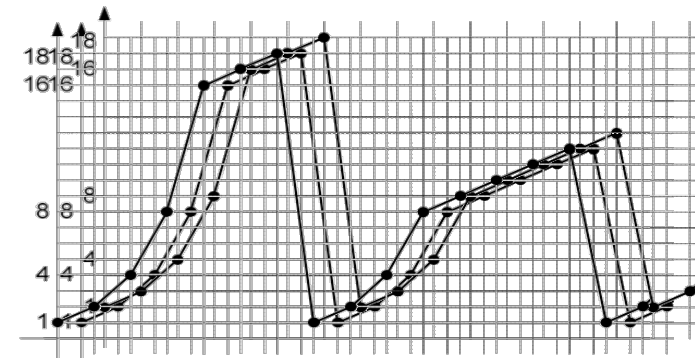
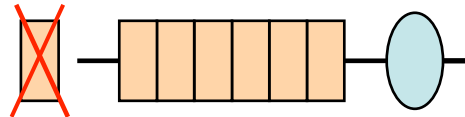
- Simple
- Descarta un paquete independientemente de su importancia
- Controla la congestión pero no la evita
- (...)



Passive Queue Management

Drop-Tail (la más habitual)

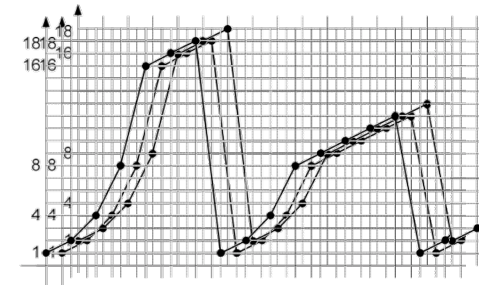
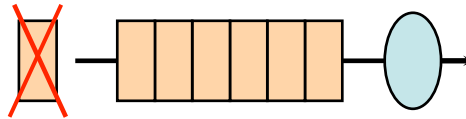
- TCP puede enviar ráfagas (limitadas por el tamaño de la ventana)
- Si llega una ráfaga a una cola casi llena se perderán varios paquetes
- Ante pérdidas TCP reduce ventana de congestión y así velocidad de envío
- Varias conexiones pueden entrar simultáneamente en este proceso de control de la congestión y reducirse en gran medida el throughput global
- Introduce **sincronización global** con varias conexiones TCP en el enlace
- (...)



Passive Queue Management

Drop-Tail (la más habitual)

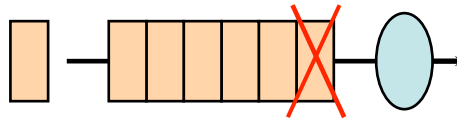
- Unos pocos flujos pueden monopolizar el recurso
- La cola se mantiene llena por largos periodos
- Pero la cola es para absorber ráfagas, luego no se podrán absorber
- Colas llenas no lleva a mayor throughput sino a menor throughput
- **Colas poco ocupadas llevan a mayor throughput y menor retardo**



Passive Queue Management

Head-drop

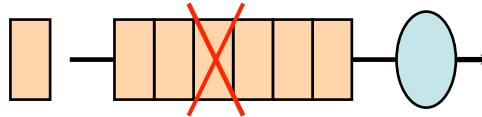
- Tira los paquetes que más tiempo llevan en el buffer
- Probablemente ya han sido retransmitidos (TCP)
- Probablemente ya llegan tarde (UDP/RTP)
- Controla la congestión pero no la evita
- Evita que unos pocos flujos monopolicen el recurso
- Posible sincronización



Passive Queue Management

Random-Drop (ante cola llena)

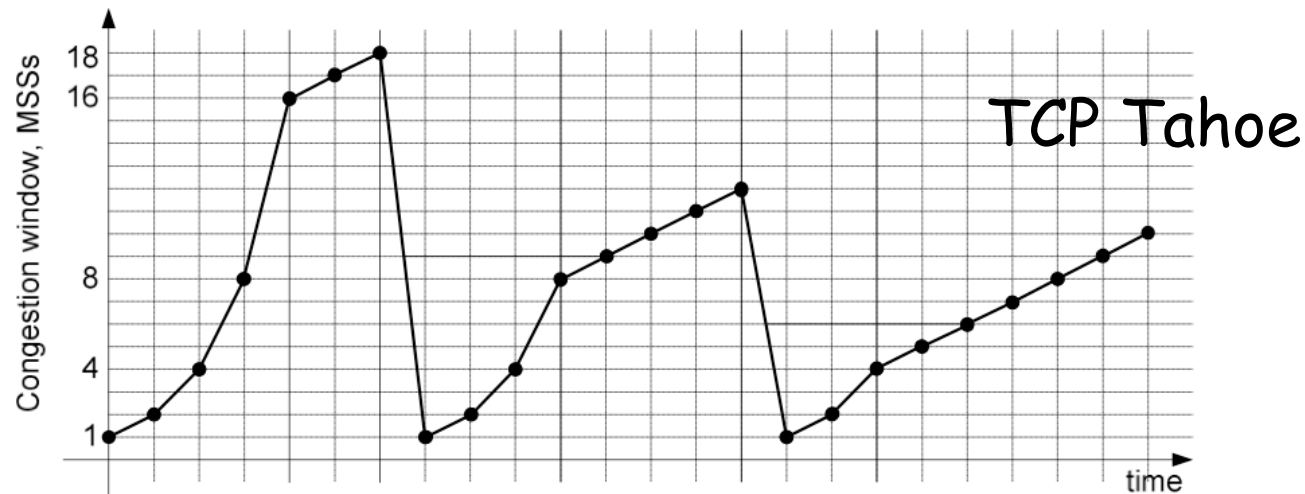
- Se puede reducir la sincronización global pero no controlar UDP
- Evita que unos pocos flujos monopolicen el recurso
- Controla la congestión pero no la evita



AQM

Active Queue Management

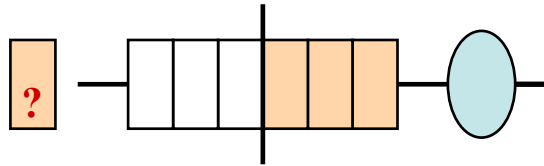
- Pensando en TCP, no controla UDP igual de bien
- Evita sincronizaciones, menores retardos y fluctuaciones
- TCP regula su tasa al detectar pérdidas (*Congestion avoidance*)



Active Queue Management

Early-Random-Drop (cola no llena)

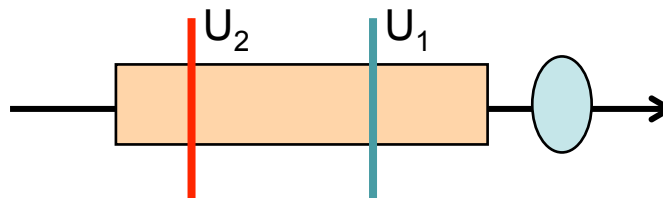
- Si la cola excede un nivel se tira cada paquete que llega con una probabilidad fija



Active Queue Management

Weighted Tail Drop

- Se asignan umbrales en la cola a diferentes clases de tráfico
- Cuando se alcanza el umbral U_1 se descartan los paquetes que lleguen de la clase 1
- Cuando se alcanza el umbral U_2 se descartan también los paquetes que lleguen de la clase 2
- De la tercera clase se descartarán solo cuando se llene la cola

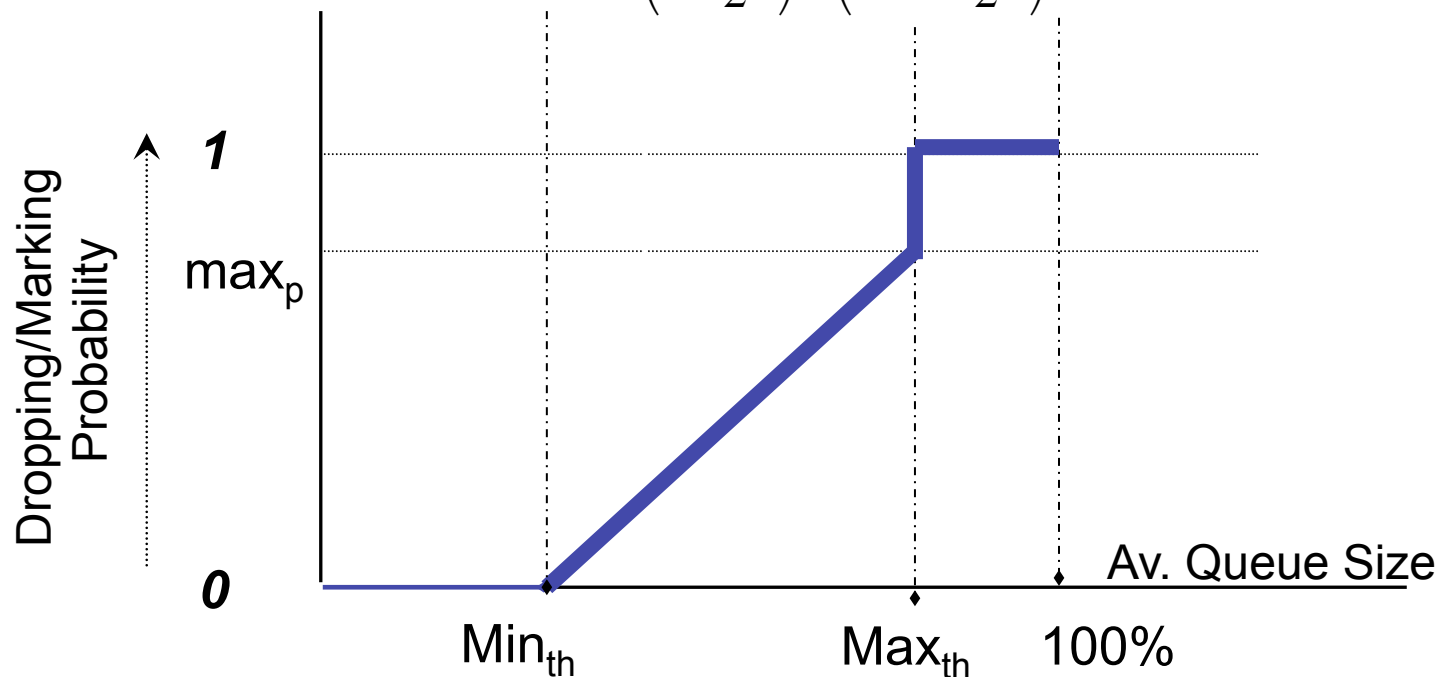


Active Queue Management

RED (Random Early Detection)

- RFC 2309 “Recommendations on Queue Management and Congestion Avoidance in the Internet”
- Descartar paquetes probabilísticamente antes de la congestión
- Evalúa la ocupación media del buffer
- Cálculo mediante *exponential moving average*
- Para parámetro w bajo la media sigue los cambios rápidos del valor instantáneo

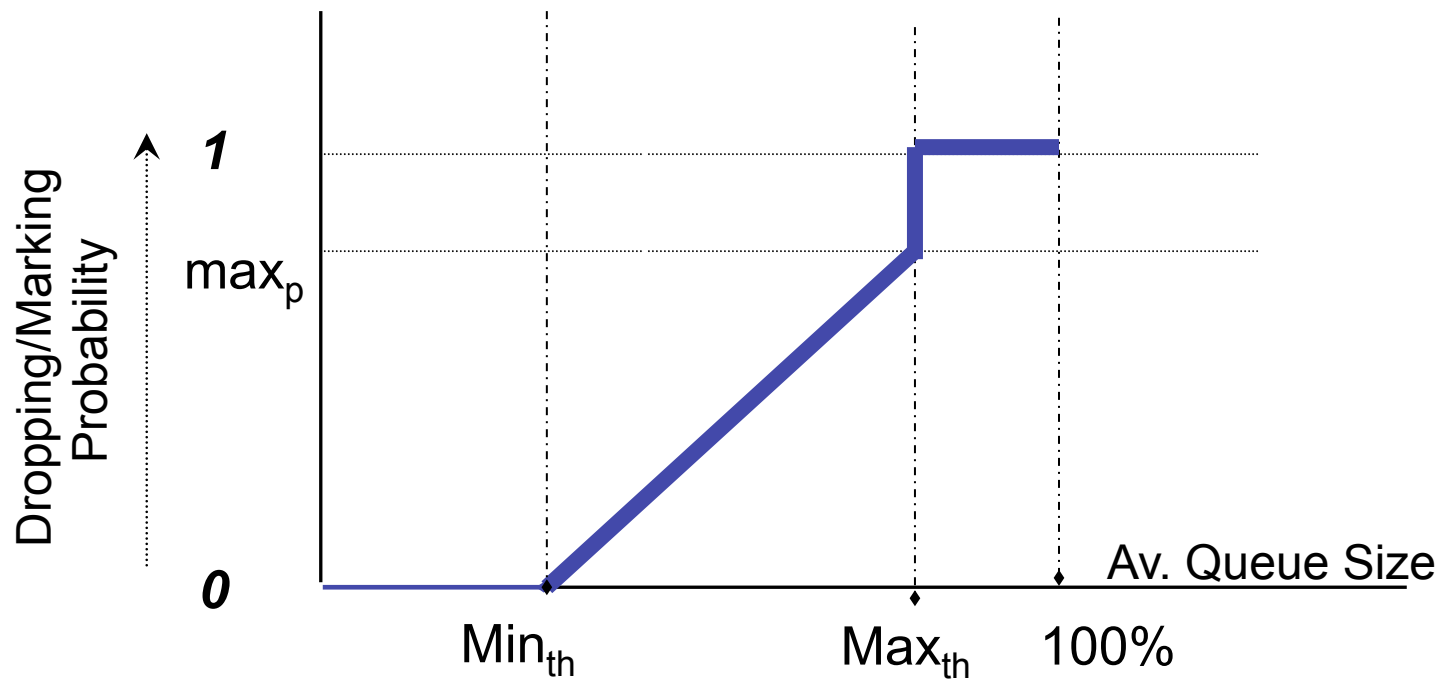
$$q_{avg} = q_{avg_old} \left(1 - \frac{1}{2^w}\right) + \left(q_{current} \frac{1}{2^w}\right)$$



Active Queue Management

RED (Random Early Detection)

- Difícil medir sus beneficios
- Con mala configuración podría comportarse peor que *drop-tail*
- Se han propuesto bastantes más algoritmos de AQM
- RED es el más extendido en implementaciones

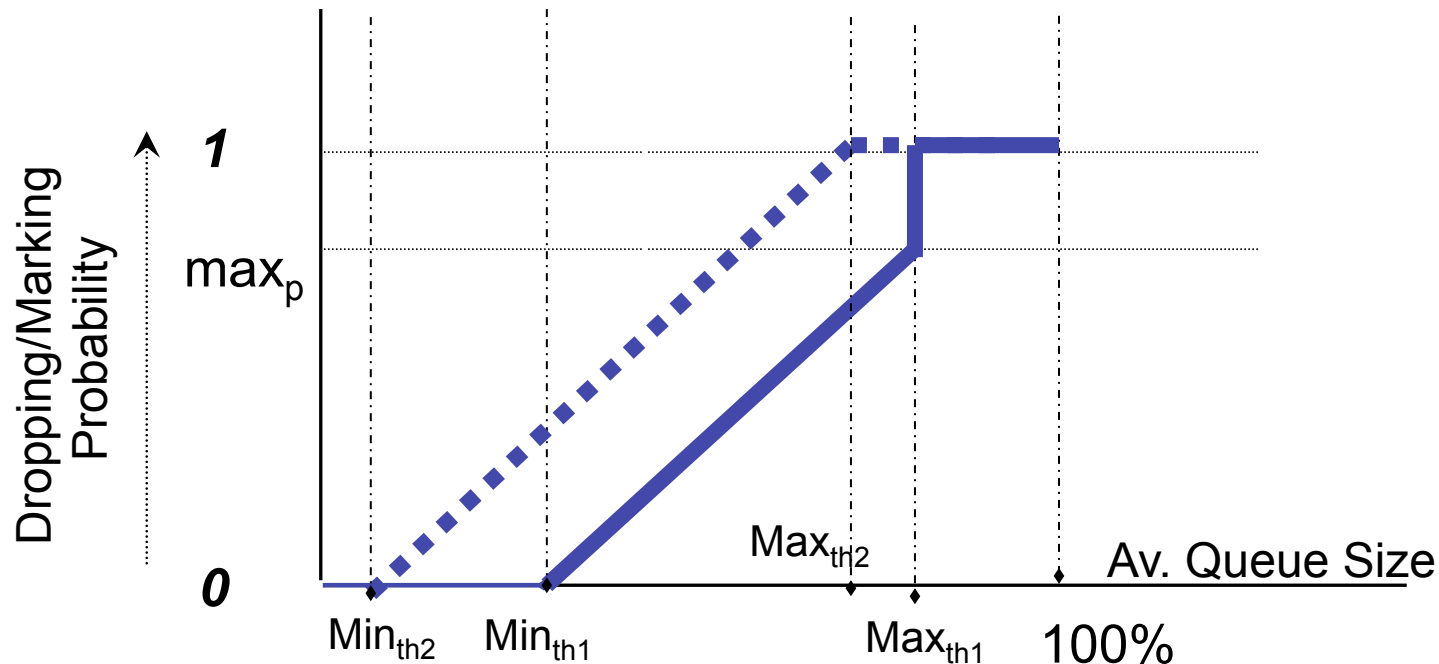


Active Queue Management

WRED (Weighted RED)

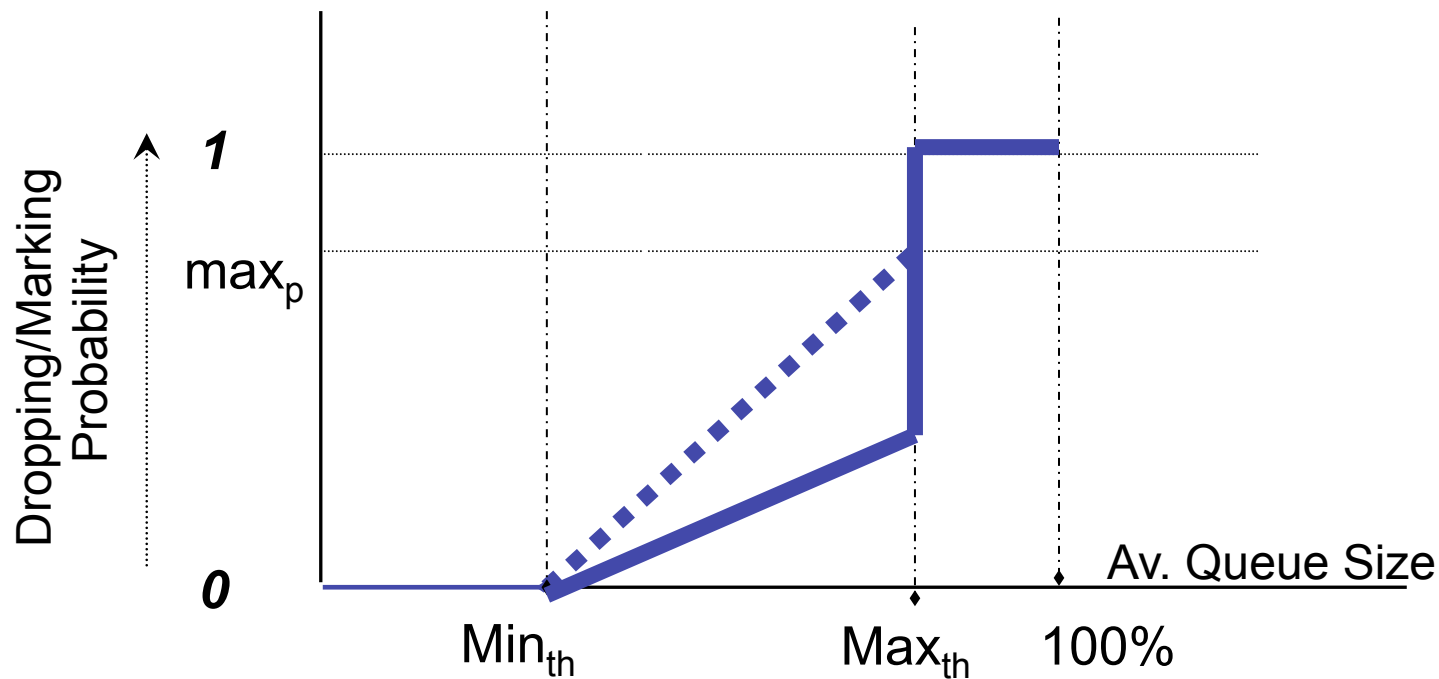
- Emplea un Min_{th} y Max_{th} diferente para diferentes clases de tráfico
- Mayor cuanto mayor es el valor de precedencia

Otras propuestas: Adaptive RED (ARED), RED In & Out (RIO), Flow weighted RED (FRED)...



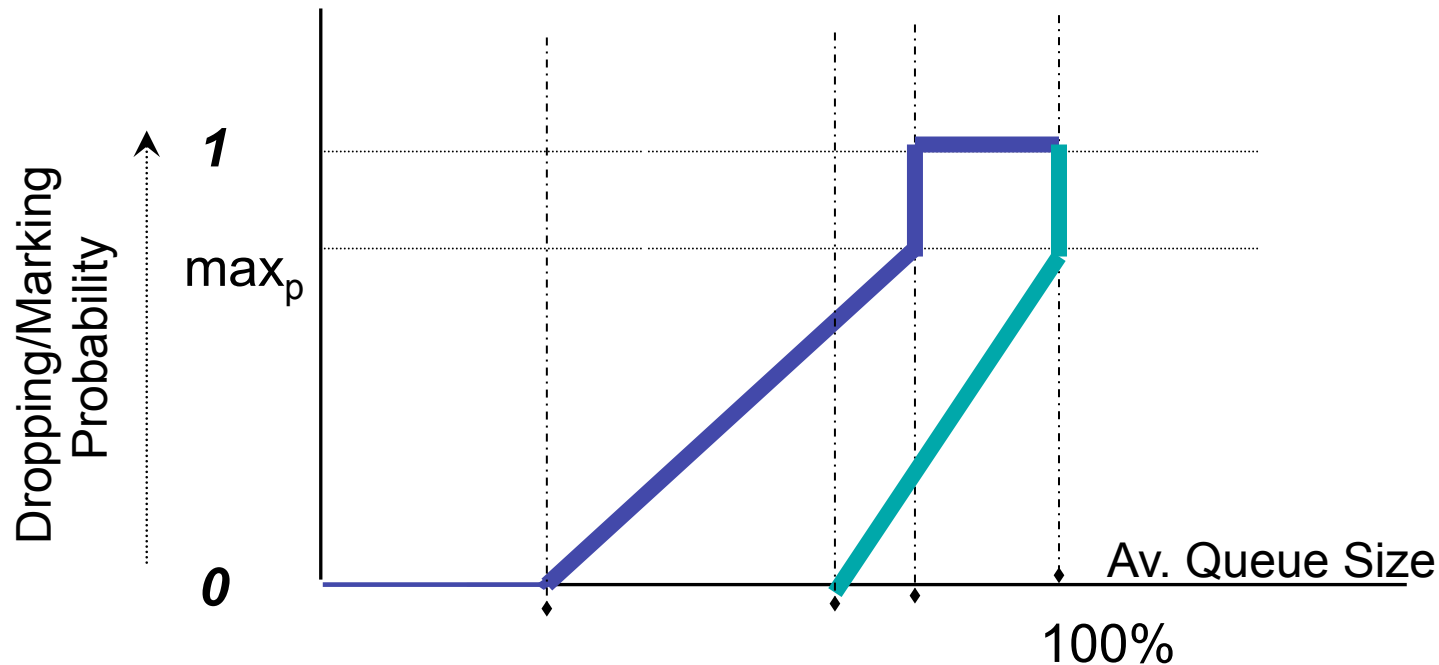
Active Queue Management

- *ARED (Adaptive RED)*
 - Que los parámetros cambien en función del tráfico
 - Difícil de implementar



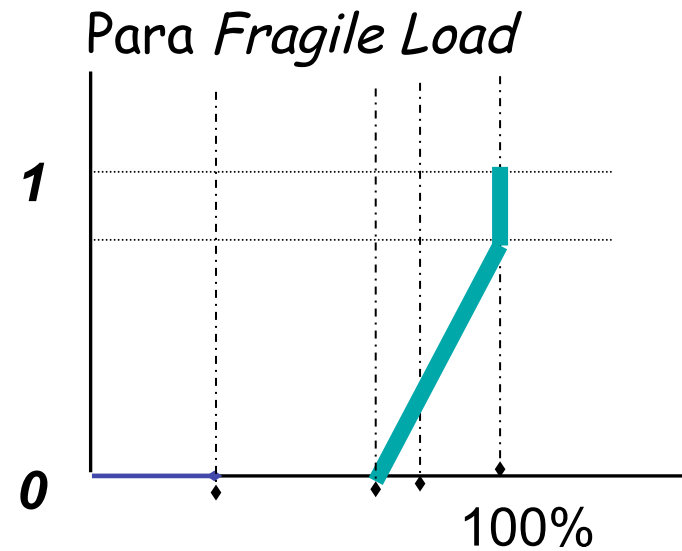
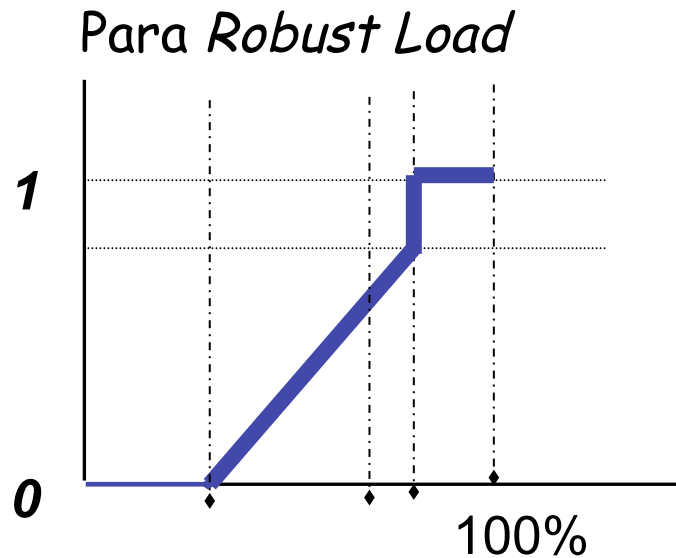
Active Queue Management

- *RIO (RED In & Out)*
 - Mantiene dos REDs simultáneos: para el tráfico *In* y el *Out*
 - Tráfico *In*: *confirming (in-profile)* (cumplen SLA)
 - Tráfico *Out*: *non-conforming (out of profile)*
 - Hay versiones en que *In* pierde solo tras entrar *Out* en $p=1$
 - Hay versiones en que ambos llegan a $p=1$ con el mismo valor de ocupación de cola



Active Queue Management

- FRED (Flow weighted RED)
 - Distinguir entre flujos
 - Carga no adaptativa (audio, video): *drop-tail*
 - *Robust Load*: TCP con pequeño RTT, reaccionan rápido
 - *Fragile Load*: TCP con gran RTT, reaccionan despacio
 - Complejo



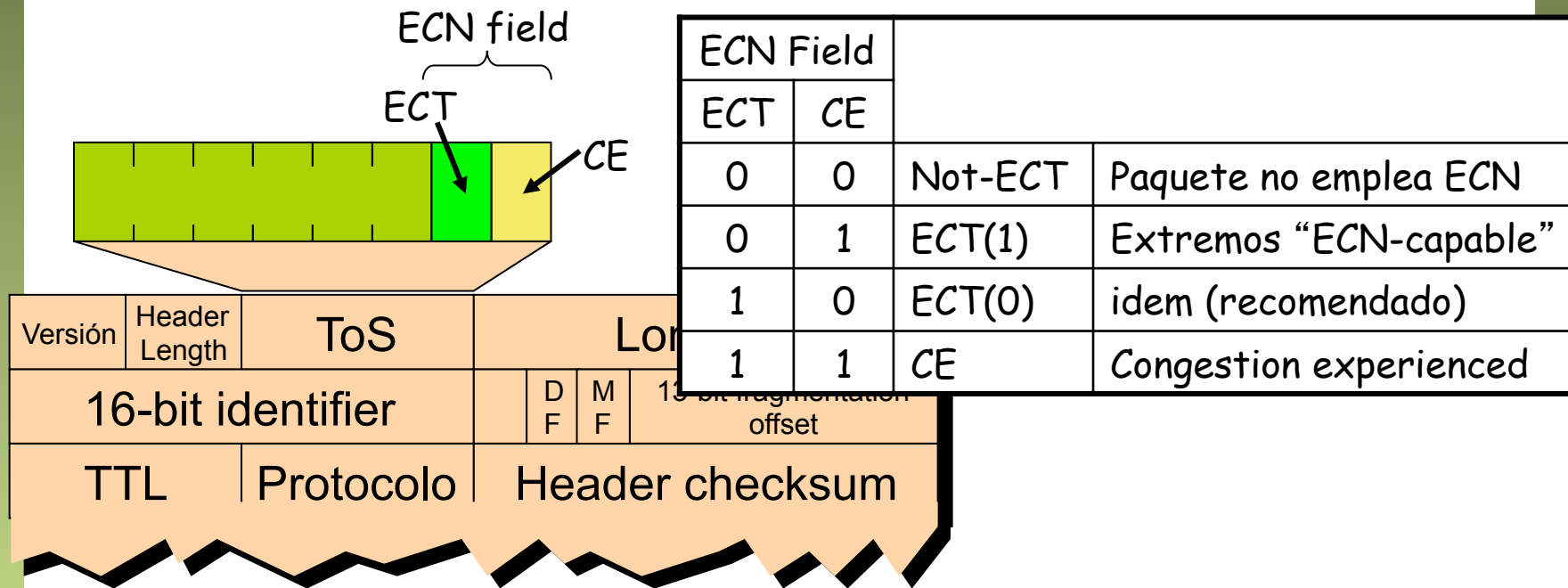


ECN



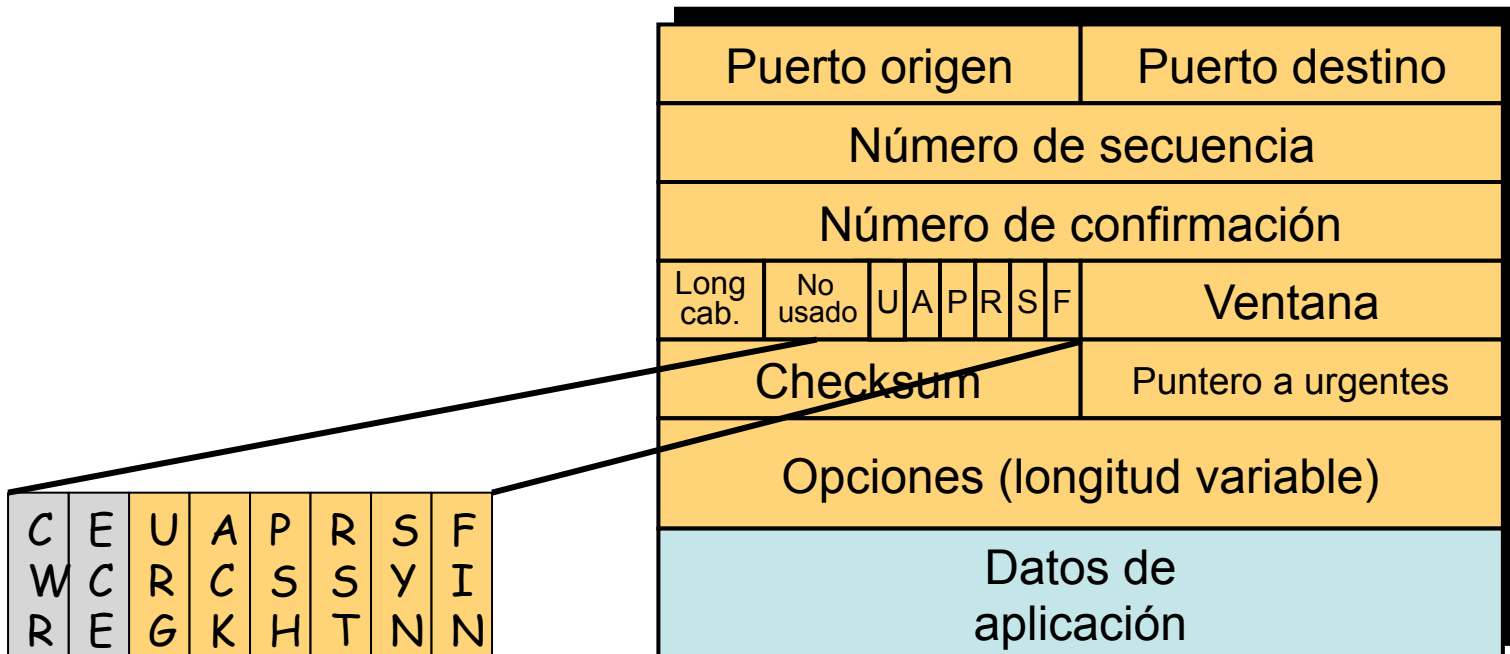
ECN

- *Explicit Congestion Notification*
- RFC 3168 “The Addition of Explicit Congestion Notification (ECN) to IP
- Extensión a RED: **marcar en vez de descartar** (salvo cuando se alcanza ocupación máxima que sí se descarta)
- Bit ECT = *ECN-Capable Transport*
- Bit CE = *Congestion Experienced*
- Requiere extender el control de congestión de TCP



ECN y TCP

- Emplea dos flags nuevos en la cabecera TCP
- ECN-Echo (ECE): para que el receptor del paquete con CE activo devuelva esta indicación a emisor
- Emisor reacciona como si hubiera detectado una pérdida
- CWR flag: emisor notifica a receptor en el siguiente paquete de que ha recibido el ECE
- Poco extendida la implementación de ECN en routers y hosts

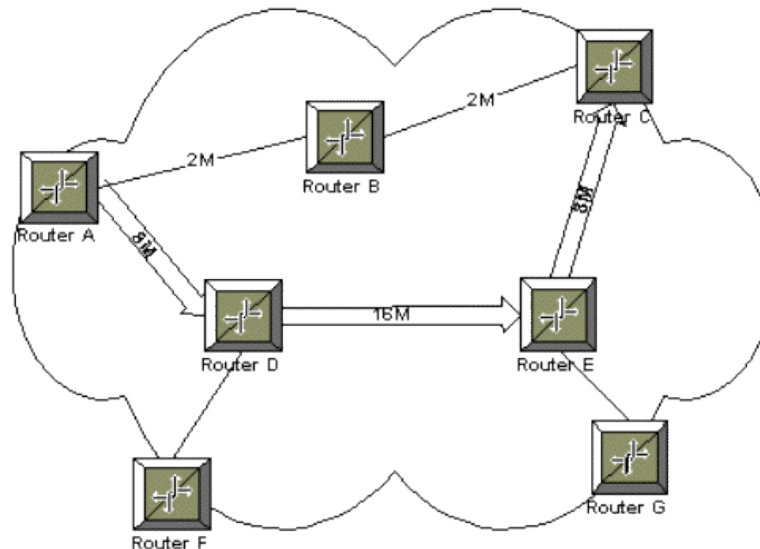


QoS routing

Elementos

- **QoS routing / Traffic Engineering**
 - Encontrar caminos “buenos” para flujos con requisitos específicos de QoS
 - Usar la red de forma eficiente: aumentar la probabilidad de aceptar peticiones futuras
 - Es complicado:
 - Información precisa sobre el estado de la red es difícil de mantener
 - Calcular caminos que cumplan requisitos de QoS es costoso (computacionalmente hablando)
 - *Constraint-based Routing*
 - Calcular caminos teniendo en cuenta no solo QoS sino también políticas

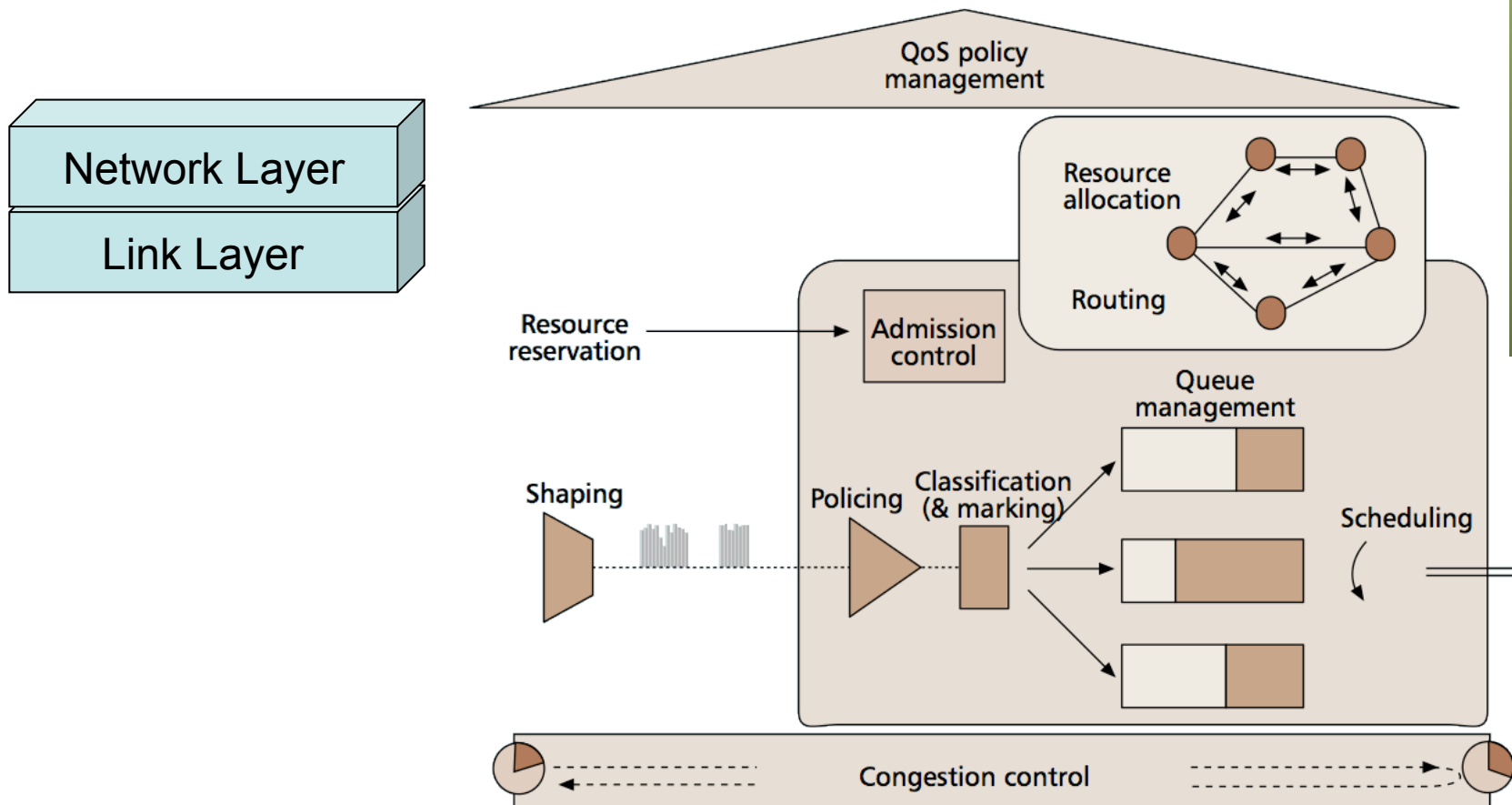
A → C: 4M



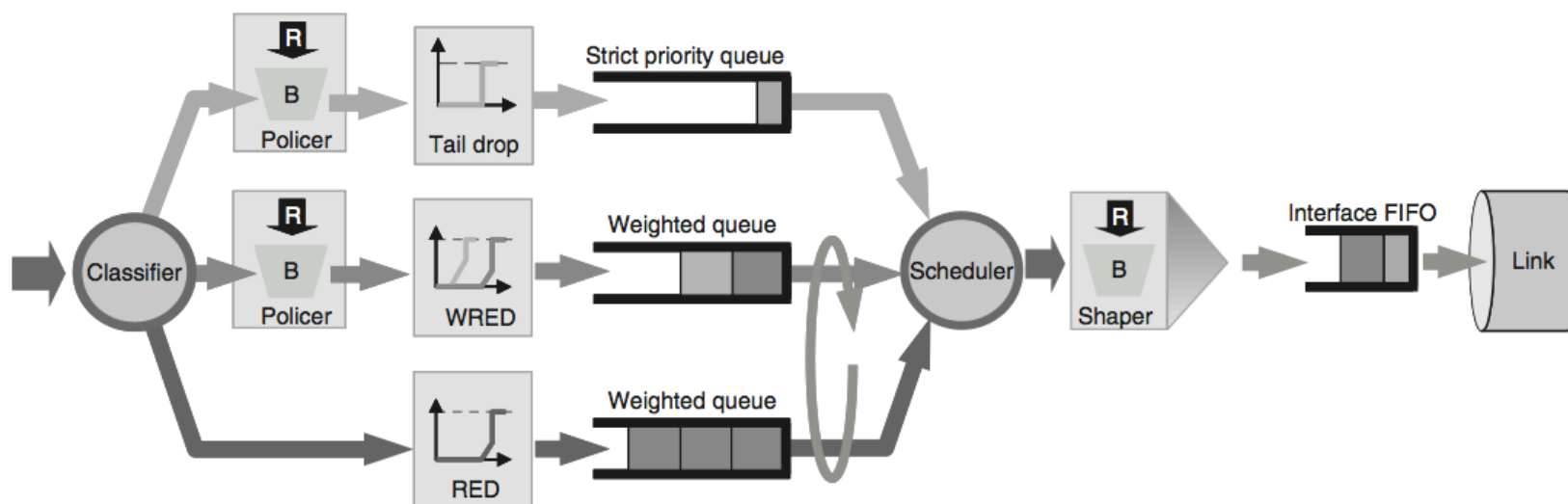
Dónde se implementa

Localización de los elementos

- Router, switch o similar
- Tienen sentido en más de una capa



Implementación



Implementación

