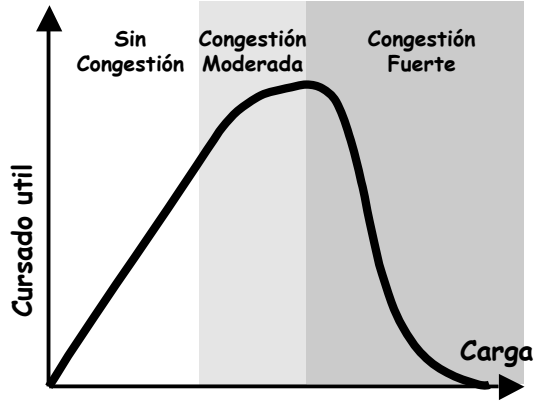
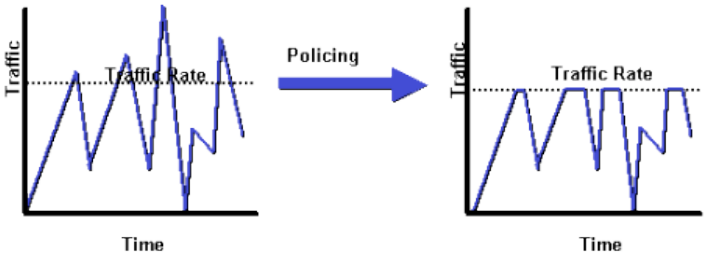
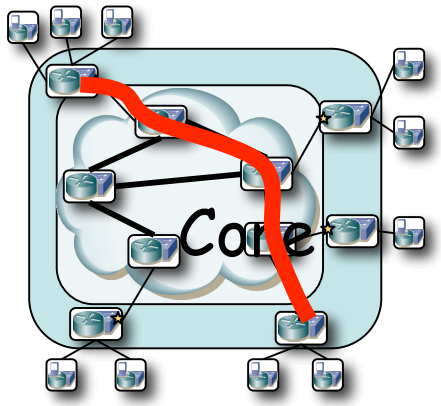
Tema 1 (2ª parte): QoS
Quality of Service

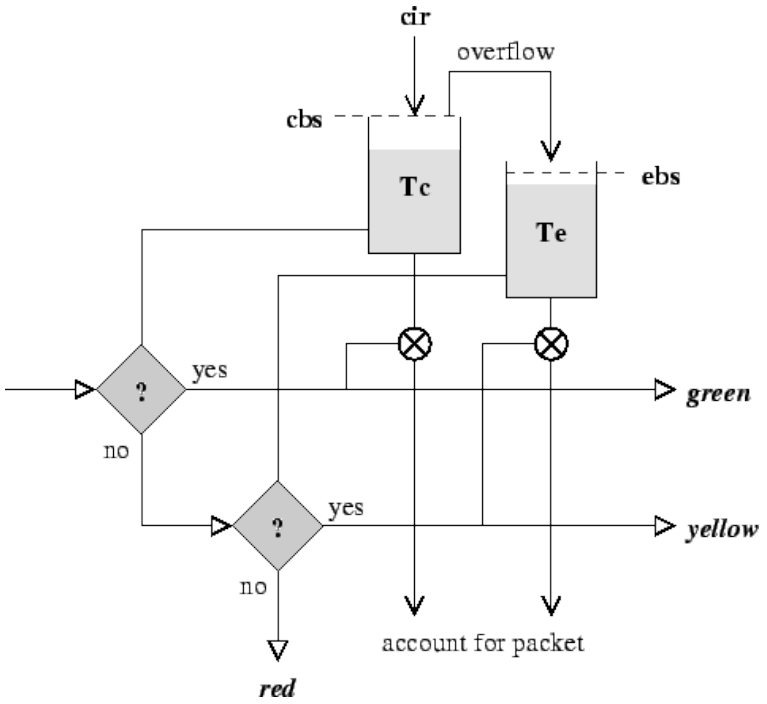
Previously on NSRI…

- Congestión y QoS
- Clasificación y marcado
- CAC
- Policing y Shaping









A continuación…

- Scheduling
- Queue management
- Arquitecturas

Scheduling

Scheduling

- Recursos compartidos
  - Buffer space
  - Capacidad en el enlace de salida
  - Tiempo de procesador
- Tipos de schedulers
  - Work-conserving
  - Non-work-conserving (no veremos)
- Schedulers sin prioridades
  - FCFS, RR, …
- Schedulers con prioridades
  - GPS, WFQ, SCFQ, WF2Q, …
- Características deseables
  - Sencillo de implementar
  - Reparto justo (*max-min fair share*) y protección
  - Performance bounds (deterministas o estadísticos)
  - Que permita implementar un CAC simple

The Conservation Law

- Sea un conjunto de N flujos en un planificador
- Para el flujo *i* la tasa media de llegadas es λ_i
- El tiempo medio de servicio de los paquetes del flujo *i* es x_i
- La utilización media del enlace debido al flujo *i* es $\rho_i = \lambda_i x_i$
- El tiempo medio de espera en cola de los paquetes del flujo *i* es q_i

Conservation Law

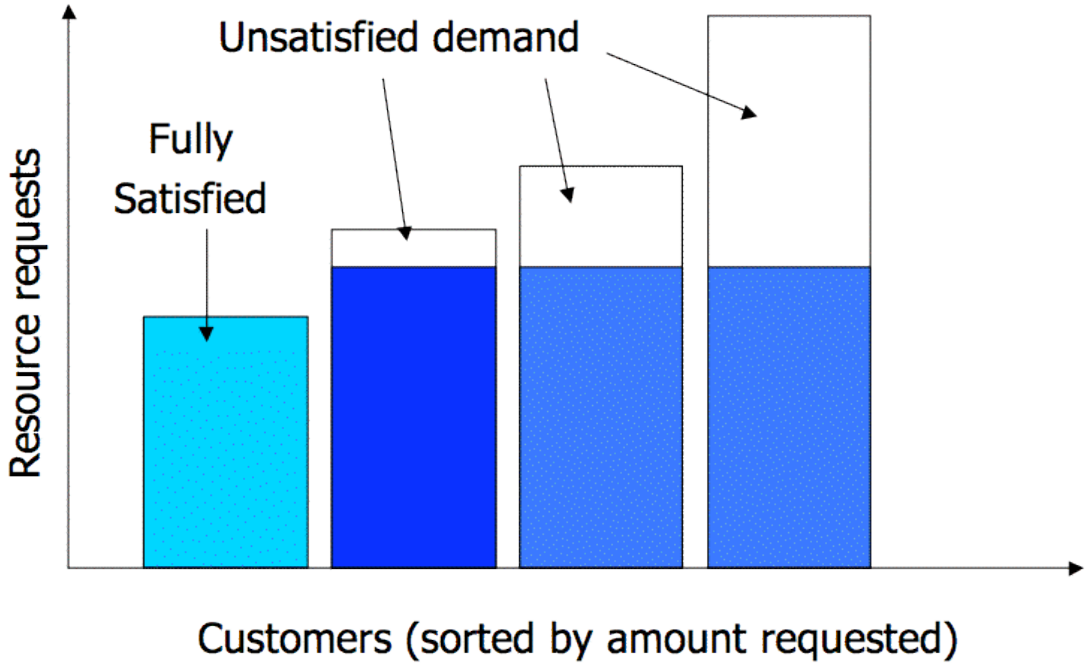
- Si el planificador es conservativo en trabajo (*work-conserving*) entonces

$$\sum_{i=1}^{N} \rho_i q_i = \text{Constante}$$

- Es independiente del planificador en concreto
- Implica que para reducir el retardo medio de una clase debemos aumentar el de otra(s)

Max-min Fair

- Asignar recursos en orden creciente de demanda
- Ningún cliente recibe más de lo que solicita
- Aquellos cuya demanda no se pueda satisfacer se reparten el remanente del recurso
- Se pueden incluir pesos

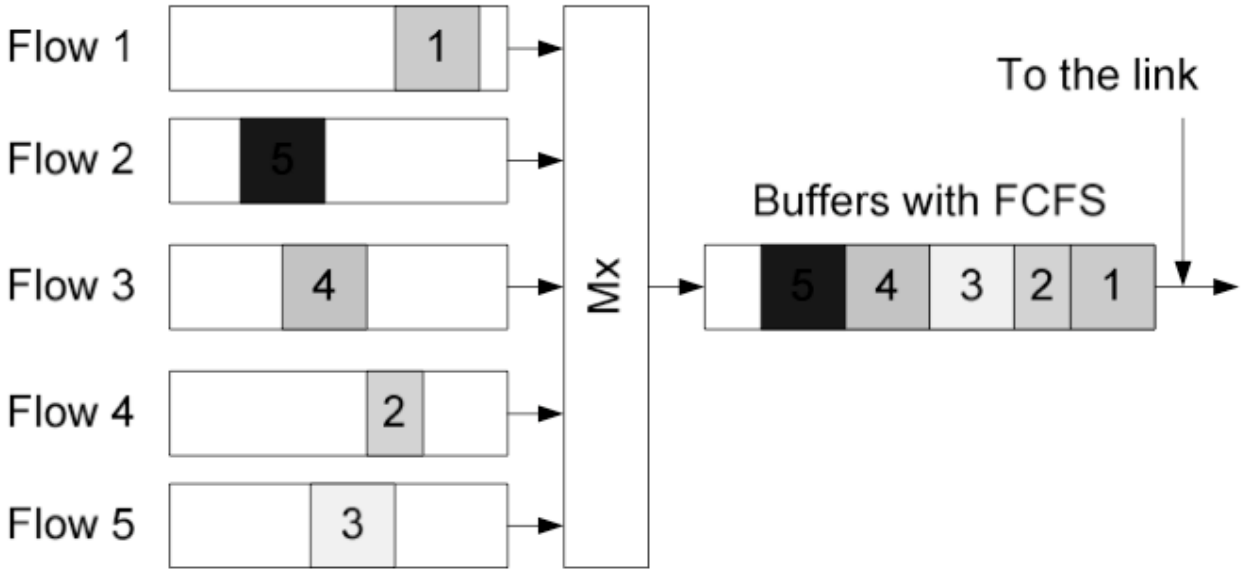


Max-min Fair (Ejemplo)

- Recurso: 10
- Demandas: 2, 2.6, 4 y 5
- 10/4 = 2.5
  - Demasiado para el primer cliente
  - Asignarle 2 y queda 0.5
- Ese 0.5 repartirlo entre los otros 3:
  - 0.5/3 = 0.167
  - Asignaciones [2, 2.67, 2.67, 2.67]
  - Demasiado para el segundo cliente
  - Asignarle 2.6 y quedan 0.07
- Repartir ese 0.07 entre los otros 2:
  - 0.07/2 = 0.035
  - Asignaciones [2, 2.6, 2.703, 2.705]

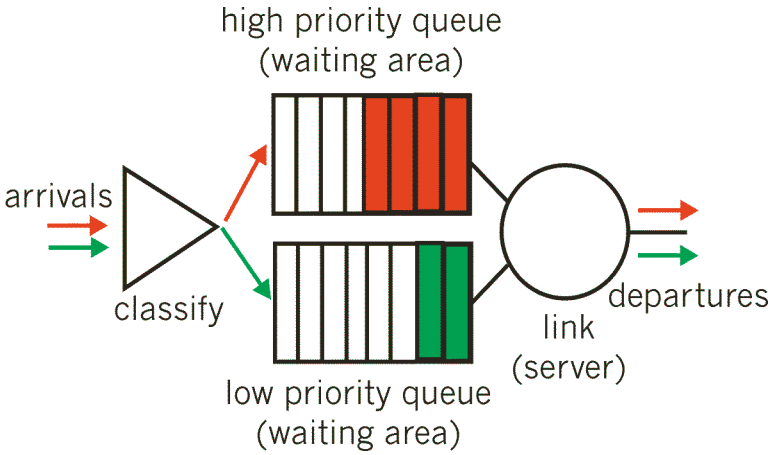
FCFS (FIFO)

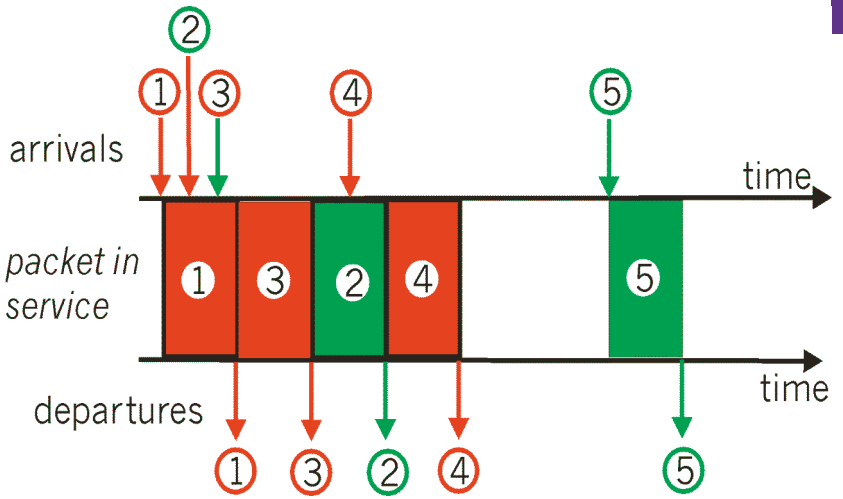
- Almacenamiento y reenvío
- Es el método más rápido y sencillo de implementar
- Se suele utilizar por defecto (B*est Effort*)
- Limitado por la capacidad del buffer ante congestión
- No permite diferenciar entre distintos tipos de paquete



Priority Queueing (PQ)

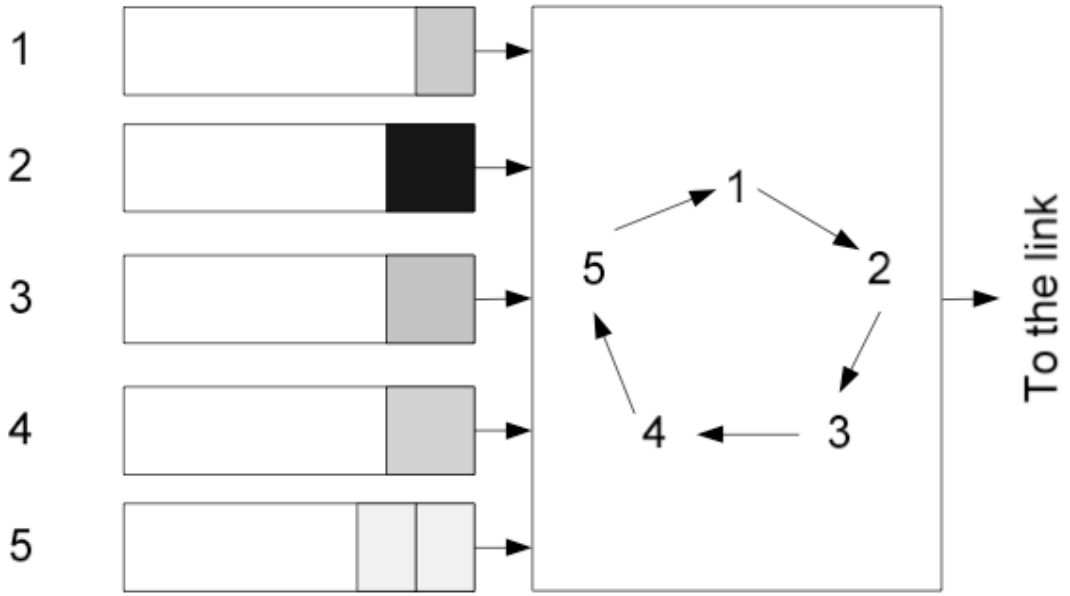
- Packets in the high-priority queue are always sent first
- Packets in the low-priority queue are not sent until all the high-priority queues become empty (*multilevel priority with exhaustive service*)
- En cada cola FCFS
- Asegura que el tráfico importante reciba un servicio rápido
- Puede crear inanición, es decir dejar fuera de servicio a tráfico menos prioritario.





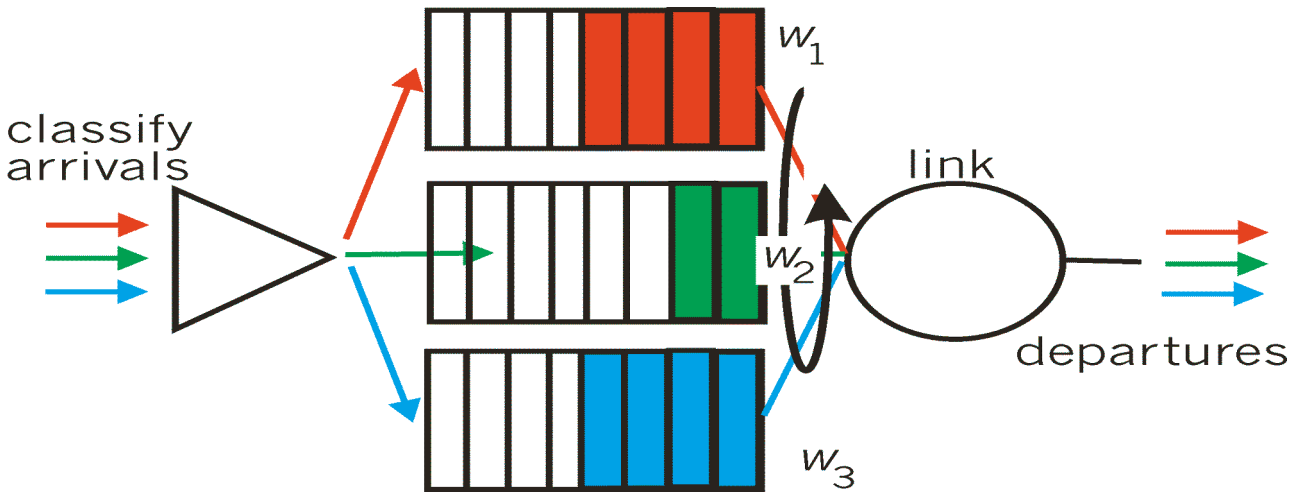
Round Robin (RR)

- Opera en “turnos” (*rounds*)
- En cada turno visita cada cola (en *round-robin*)
- En cada cola FCFS
- Se sirven un número de paquetes o paquetes durante un cierto tiempo fijo



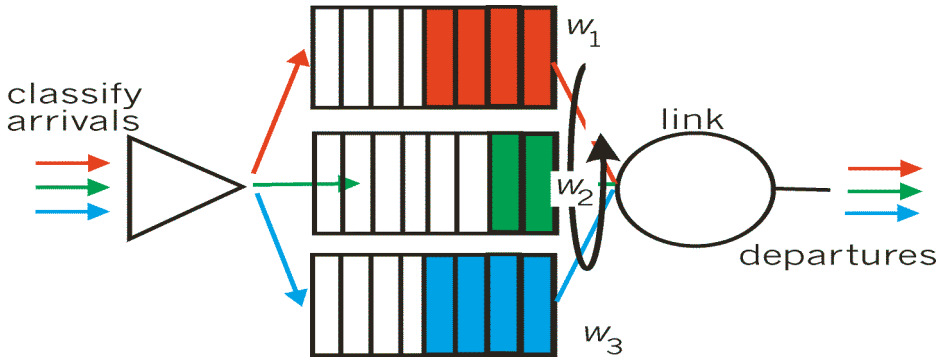
GPS

- Algoritmo basado en flujos, cada uno a una cola
- PS = *Processor Sharing* : Cada uno de los N flujos recibe 1/N
- GPS = *Generalized Processor Sharing* : pesos a cada uno
- A gran escala cada clase obtiene un servicio proporcional al peso asignado
- Asegura que las diferentes colas no se queden privadas de un mínimo ancho de banda
- No da garantías totales como PQ
- Max-min fair (y por ser *fair* ofrece protección)



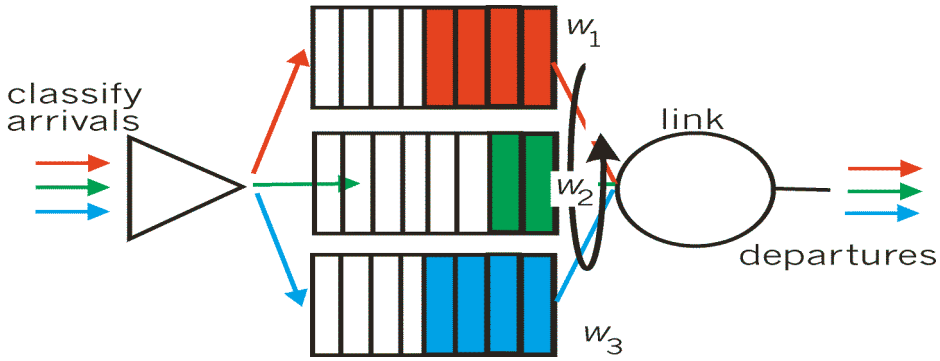
WRR

- *Weighted Round Robin*
- Aproximación de GPS (*Generalized Packet Sharing*) para el caso de paquetes
- Opera en “turnos” (*rounds*). En cada turno visita cada cola (en *round-robin*)
- Divide el peso por el tamaño medio de los paquetes del flujo
- En la visita sirve uno o más paquetes de forma que la cantidad sea proporcional al peso asignado a la cola
- *Fair* por encima de la escala del turno



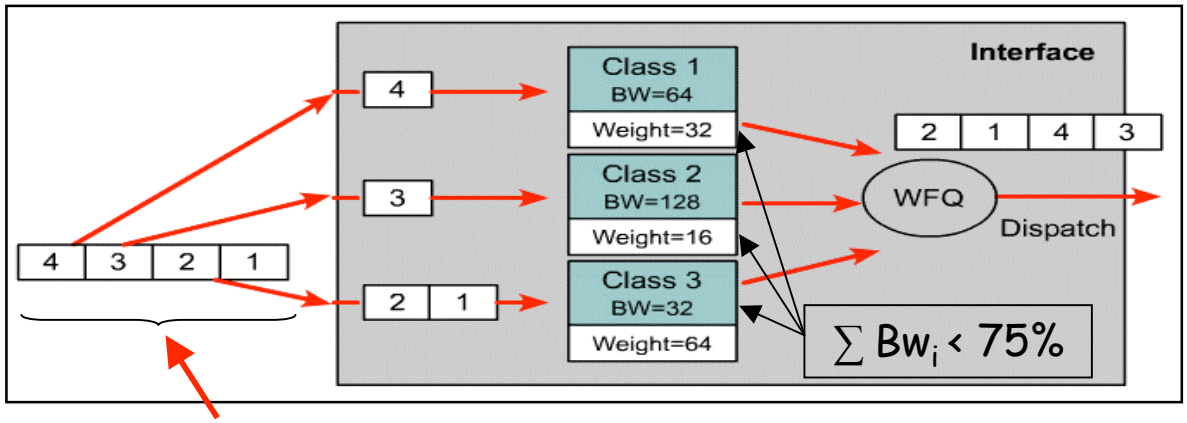
WFQ

- *Weighted Fair Queueing*
- Aproximación de GPS (*Generalized Packet Sharing*) para el caso de paquetes
- Equivalente a PGPS
- Emplea un reloj virtual
- Calcula el comienzo y final virtual en que se enviaría cada paquete en el caso ideal GPS
- Se envían en orden de tiempo final virtual
- Más complejo de implementar
- Puede ofrecer *worst-case bounds*



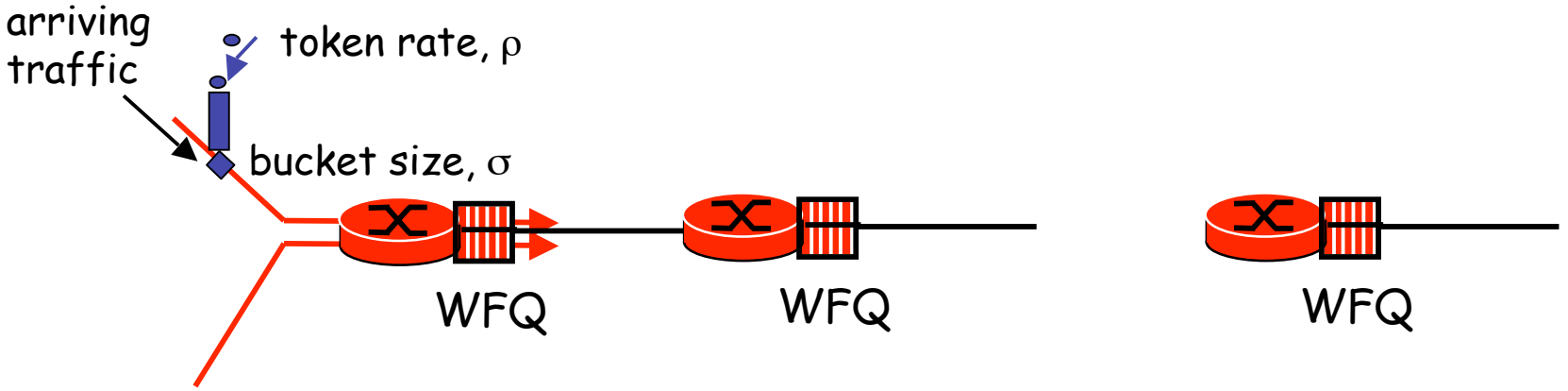
Típica implementación de WFQ

- Cada flujo es una “conversación” reconocida por info. de layer 3 (direcciones IP, precedencia) y de layer 4 (puertos)
- Pesos en función de los bits de precedencia de los paquetes
- No requiere configuración
- No escala (una cola por conversación)
- CBWFQ
  - *Class Based WFQ*
  - Especificar los filtros (clases) que determinan los paquetes que van a cada cola (una por clase, no por flujo)
  - Especificar peso para cada cola



Tocken bucket + WFQ

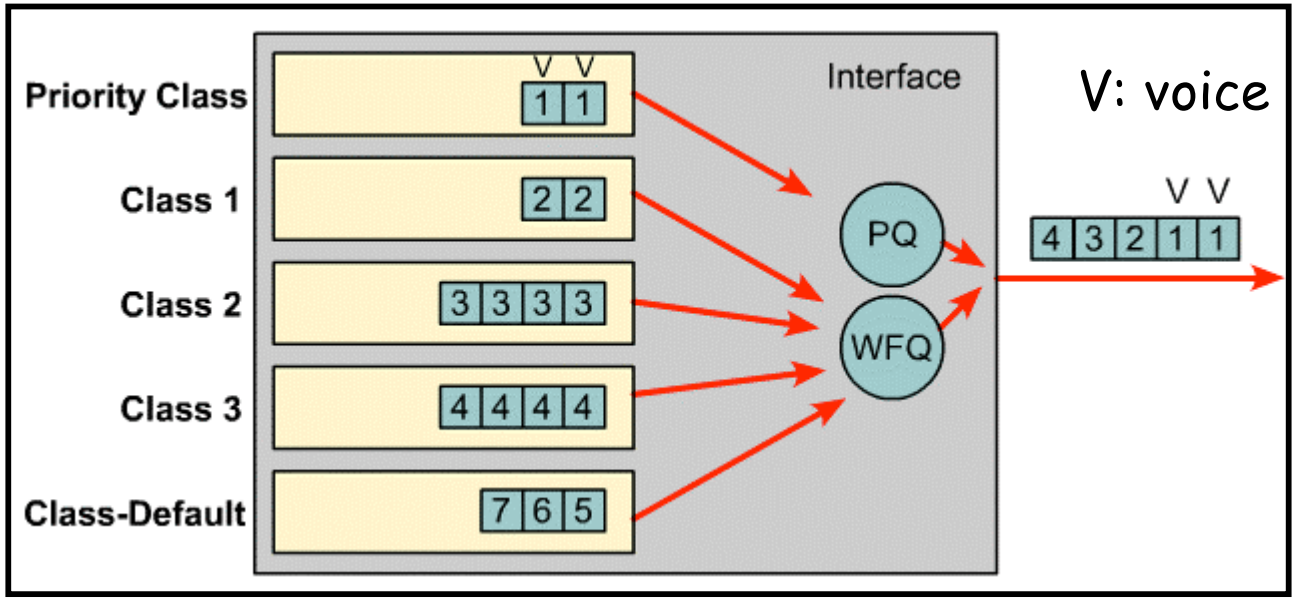
- PGPS permite ofrecer garantías de límite de retardo
- Si:
  - Flujo restricción (σ, ρ) (el resto puede no estar conformado)
  - Camino con *h* saltos (todos WFQ)
  - Se le ha asignado al menos una tasa de ρ en todos ellos
- Entonces:
  - El retardo end-to-end está acotado



Low Latency Queueing (LLQ)

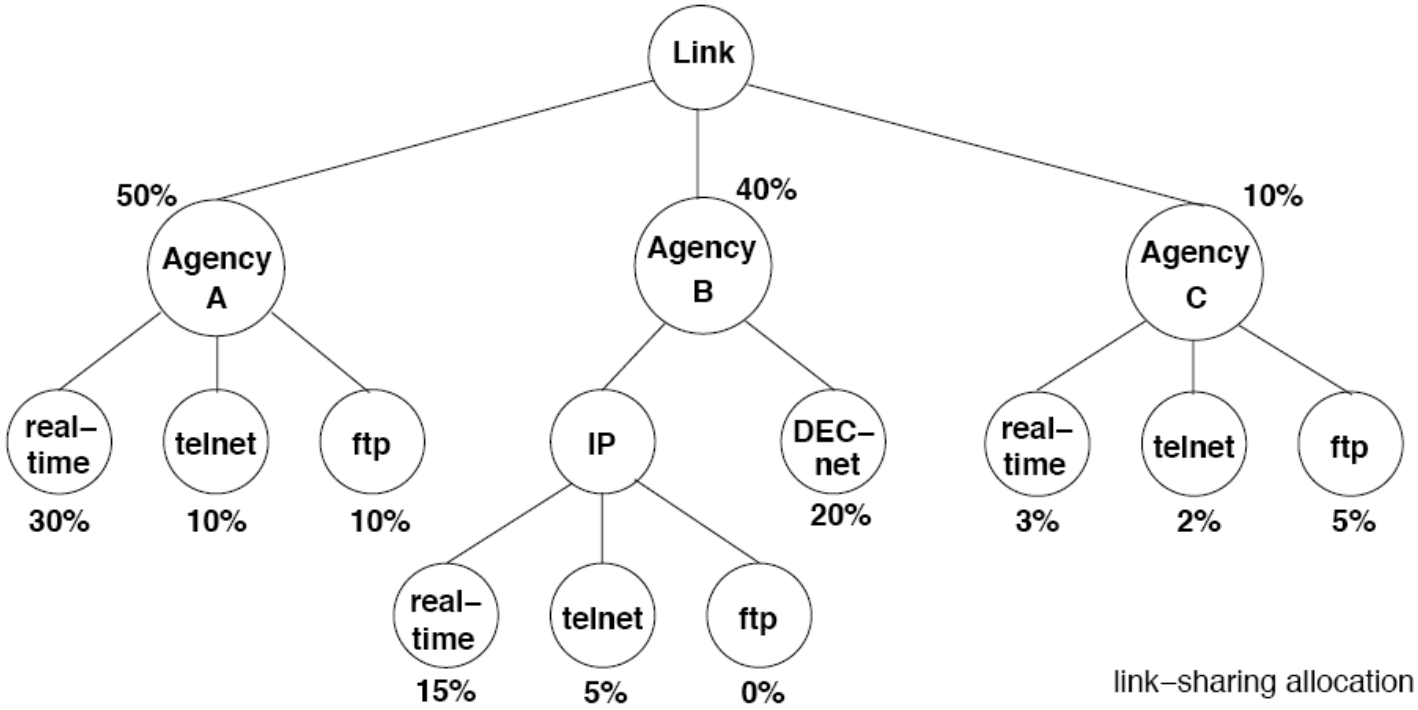
- Añade una PQ (*Priority Queue*) a CBWFQ = PQ-CBWFQ = LLQ
- Recomendable para tráfico multimedia (VoIP): bajo retardo y jitter.
- Se puede configurar junto al resto de colas CBWFQ como una cola más asociada a una clase determinada.

LLQ se comporta como una *Priority Queue*.



Class Based Queueing (CBQ)

- Puede contener diferentes planificadores

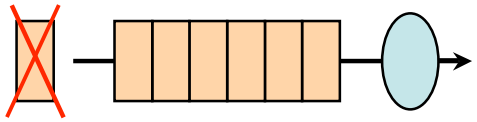


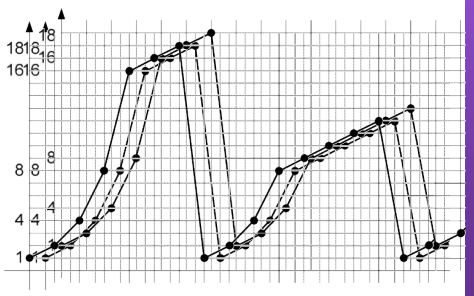
Queueing

Queue Management

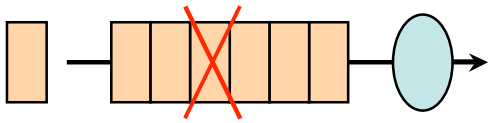
Pasivo

- *Drop-Tail* (la más habitual)
  - Simple
  - Introduce sincronización global cuando hay varias conexiones TCP atravesando ese enlace
  - Controla la congestión pero no la evita, posible synch.





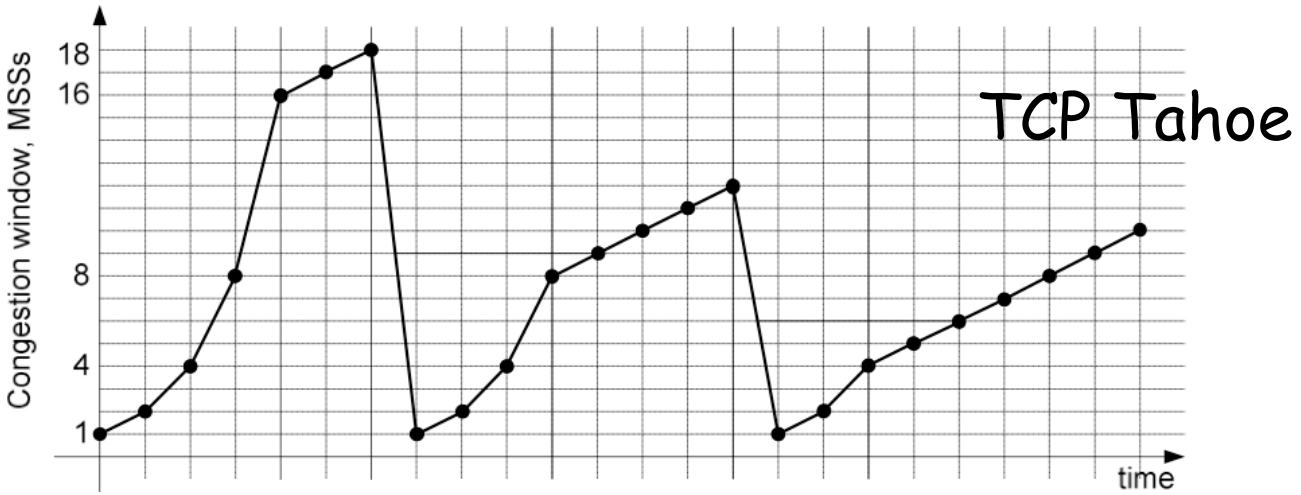
- *Head-drop*
  - Tira los paquetes que más tiempo llevan en el buffer
  - Probablemente ya han sido retransmitidos (TCP)
  - Probablemente ya llegan tarde (UDP/RTP)
  - Controla la congestión pero no la evita, posible synch
- *Random-Drop* (ante cola llena)
  - Se puede reducir la sincronización global pero no controlar UDP
  - Controla la congestión pero no la evita



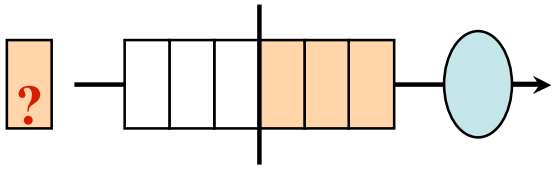
Queue Management

Activo (AQM)

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

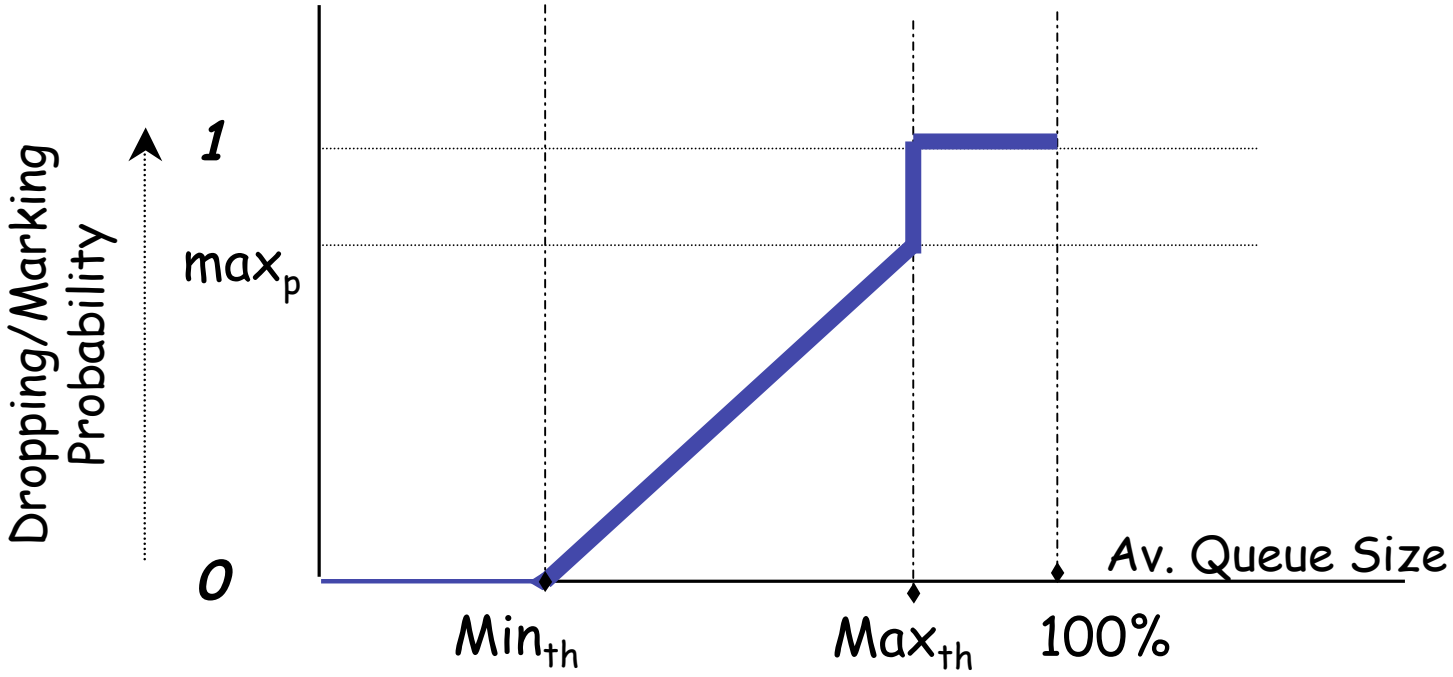


- *Early-Random-Drop* (cola no llena)
  - Si la cola excede un nivel se tira cada paquete que llega con una probabilidad fija



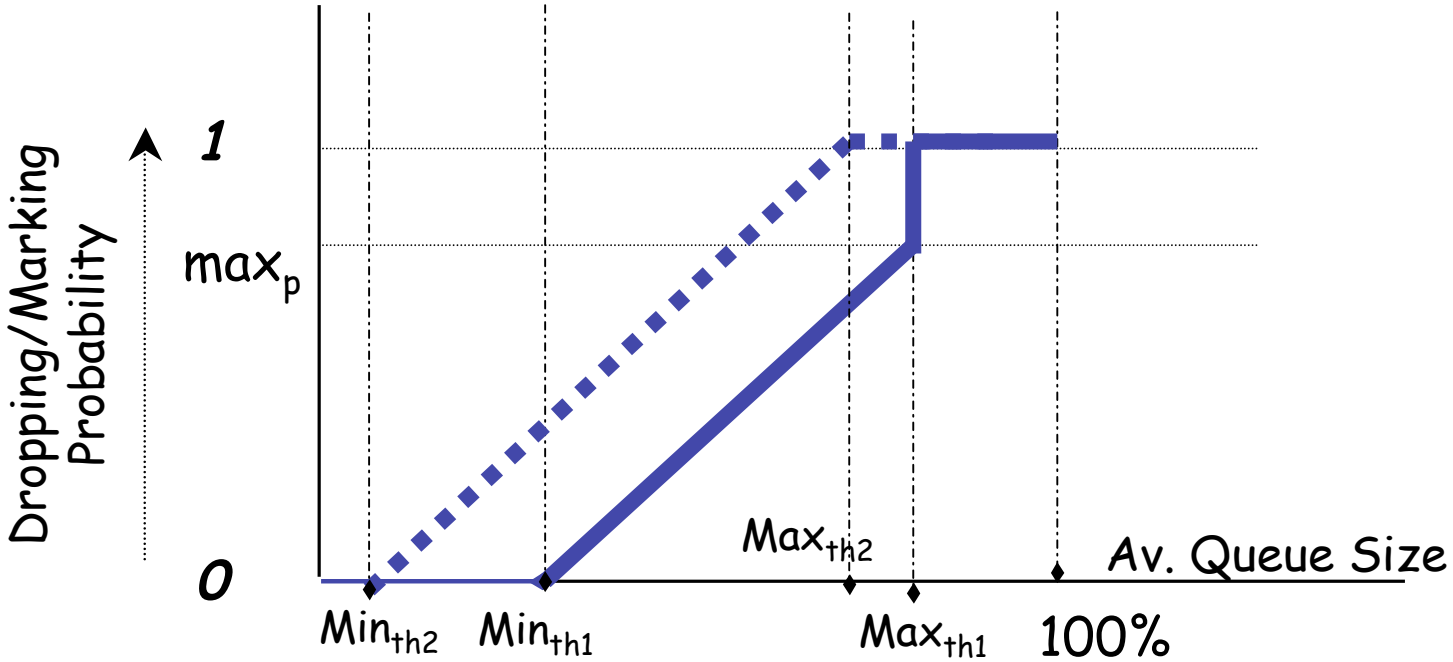
Active Queue Management

- *RED (Random Early Detection)*
  - RFC 2309
  - Evalúa la ocupación media del buffer (*exponential weighted moving average*)
  - Descartar paquetes probabilísticamente antes de la congestión
  - Ojo: Con mala configuración se comporta peor que *drop-tail*



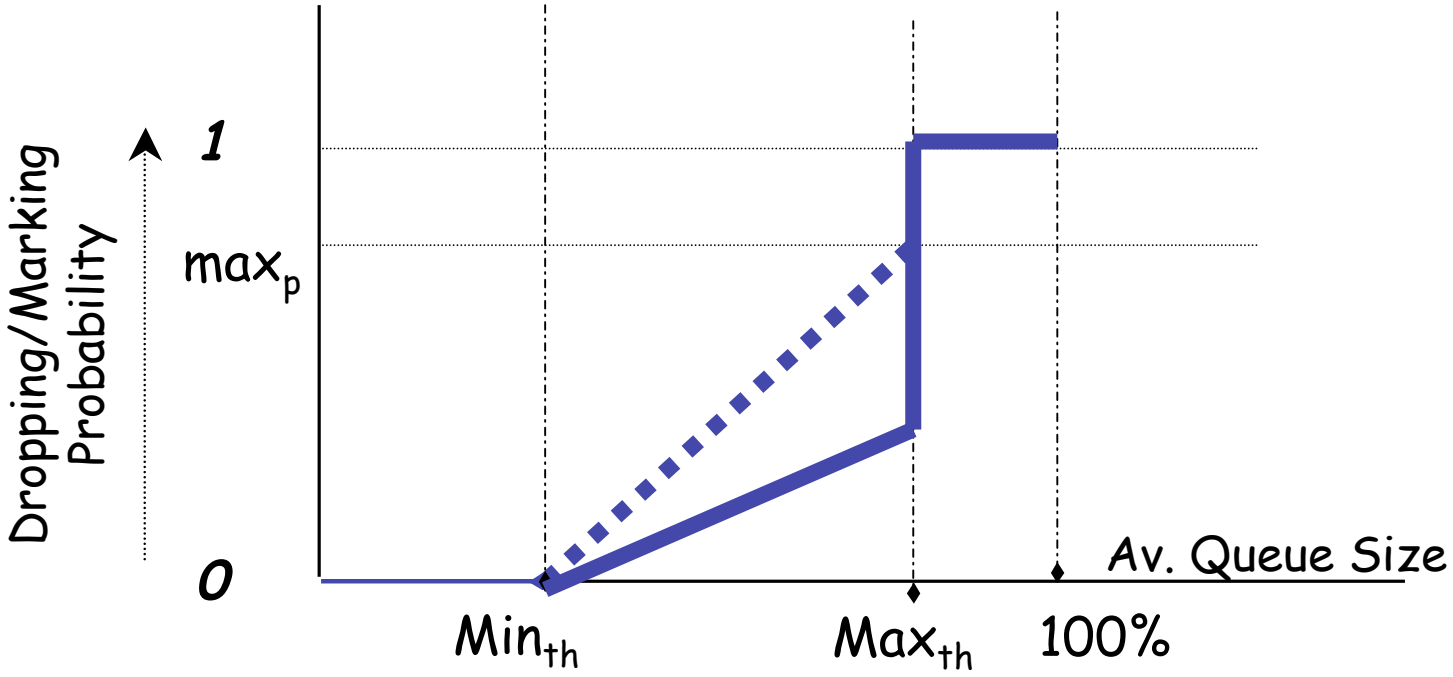
Active Queue Management

- *WRED (Weighted RED)*
  - Emplea un Min_{th} diferente para diferentes clases de tráfico
  - Mayor cuanto mayor es el valor de precedencia



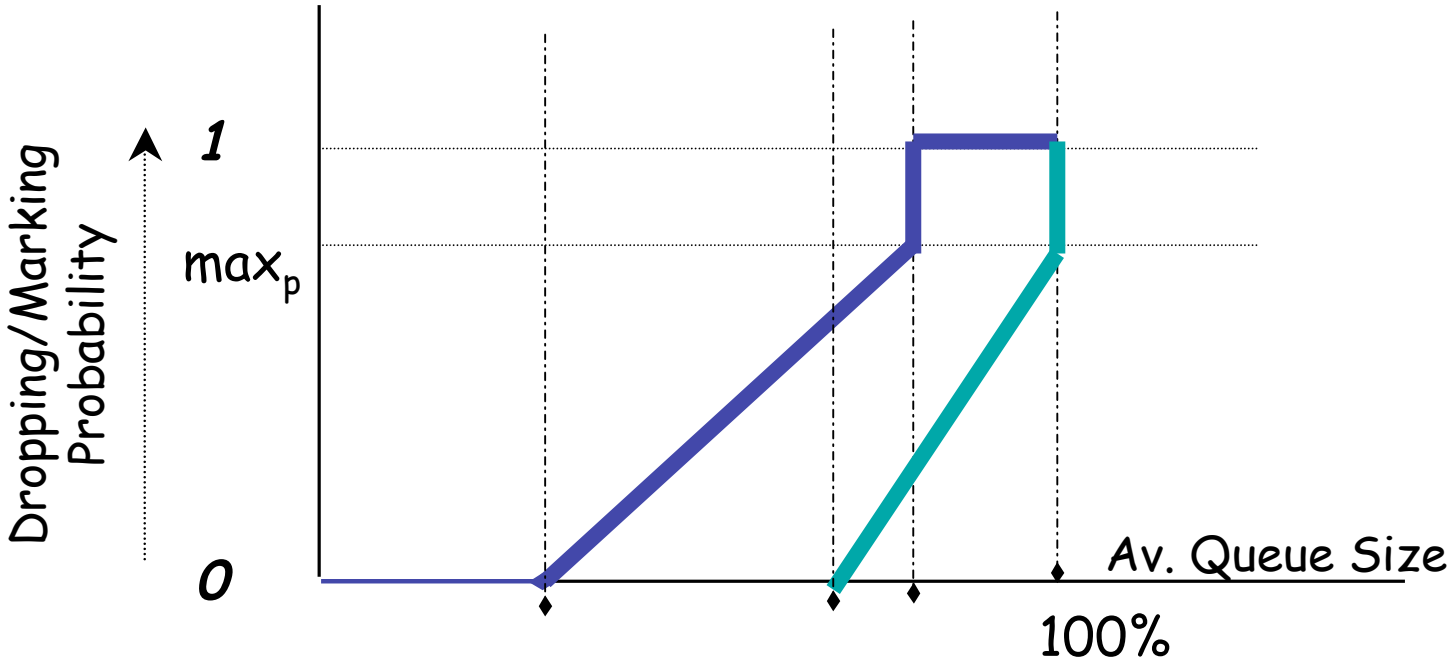
Active Queue Management

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



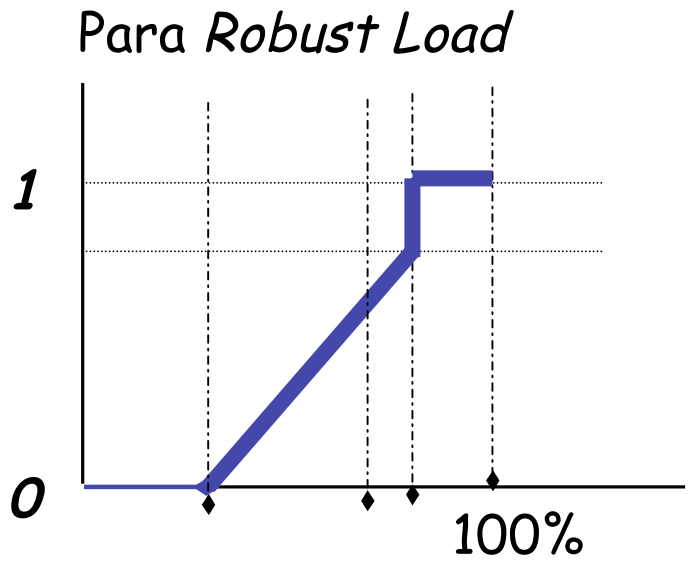
Active Queue Management

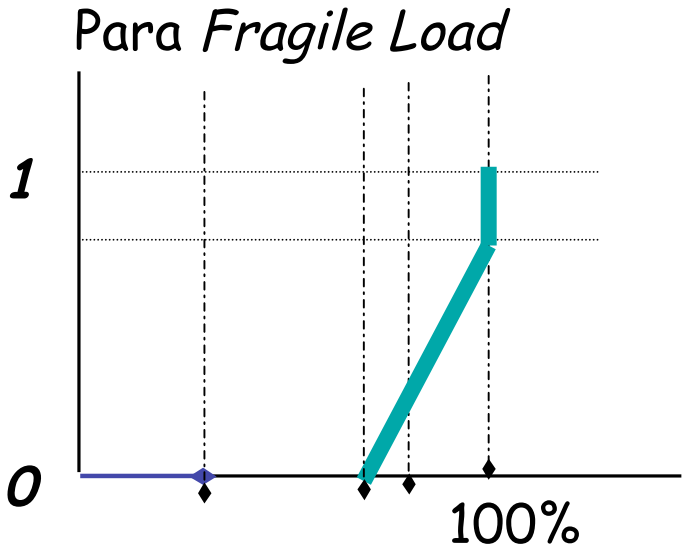
- *RIO (RED In & Out)*
  - Mantiene dos REDs simultáneos: para el tráfico *In* y el *Out*
  - Tráfico *In*: *confirming (in-profile)* (cumplen SLA)
  - Tráfico *Out*: *non-conforming (out of profile)*
  - Hay versiones en que *In* pierde solo tras entrar *Out* en p=1
  - Hay versiones en que ambos llegan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



Active Queue Management

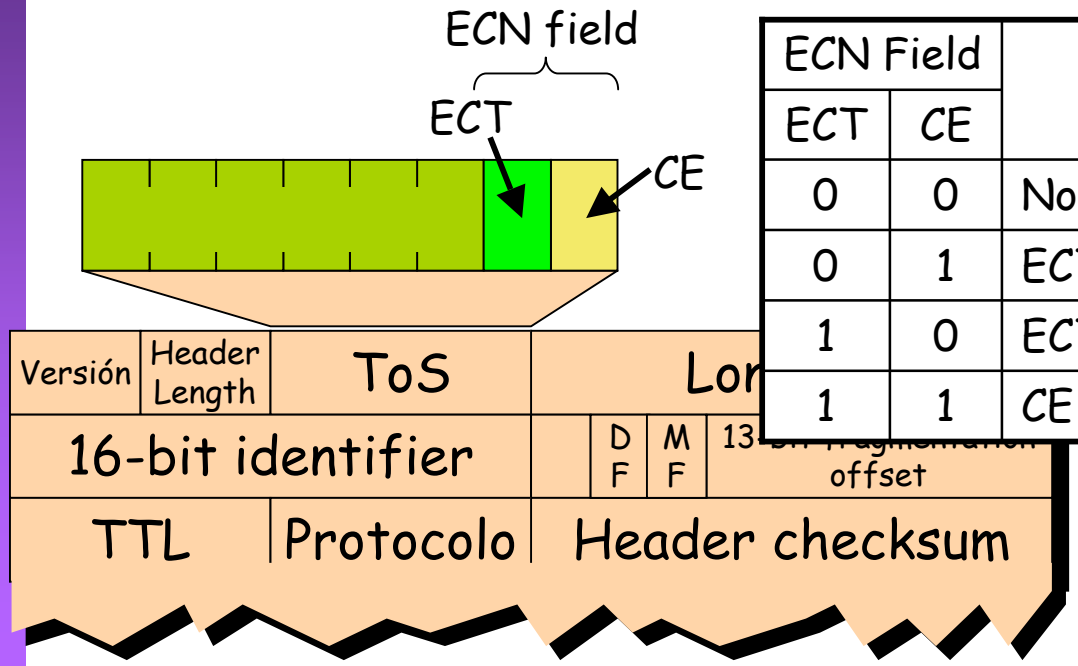
- FRED (Flow weighted RED)
  - Distinguir entre flujos
  - Carga no adaptativa (audio, video): *drop-tail*
  - *Robust Load*: TCP con pequeño RTT, reaccionan rápido
  - *Fragile Load*: TCP con gran RTT, reaccionan despacio
  - Complejo





ECN

- *Explicit Congestion Notification*
- RFC 3168
- Extensión a RED: marcar en vez de descartar
- Bit ECT = *ECN-Capable Transport*
- Bit CE = *Congestion Experienced*
- Requiere extender el control de congestión de TCP



ECN Field			
ECT	CE		
0	0	Not-ECT	Paquete no emplea ECN
0	1	ECT(1)	Extremos "ECN-capable"
1	0	ECT(0)	idem (recomendado)
1	1	CE	Congestion experienced

LFI

Link Fragment and Interleaving (LFI)

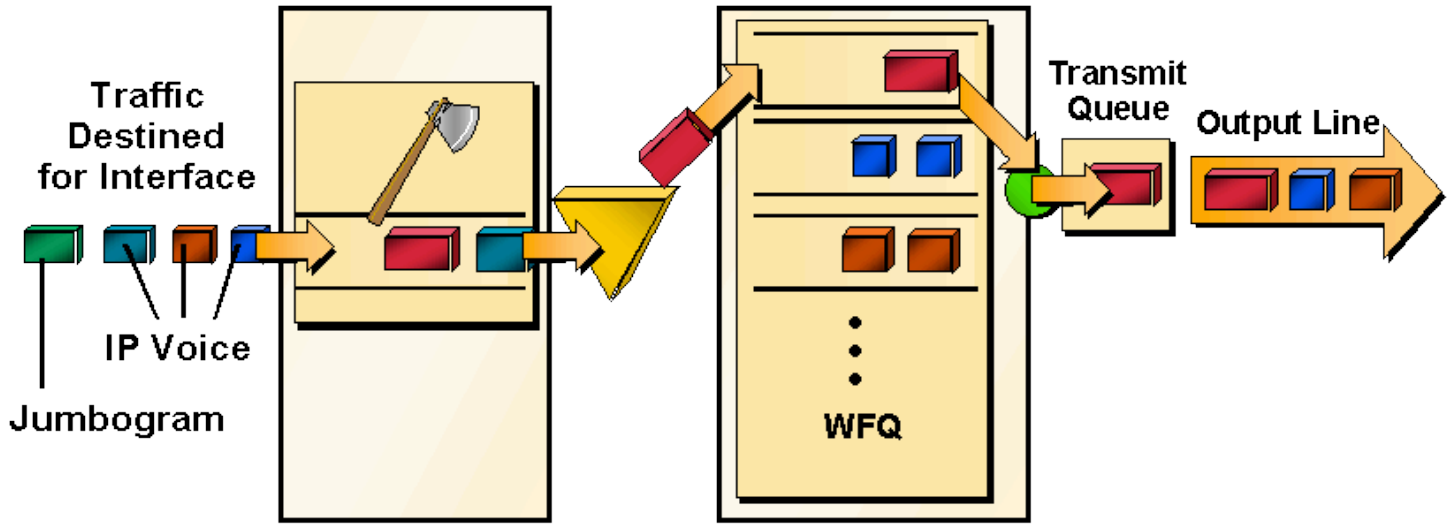
Problema:

- Llega paquete IP a su cola de alta prioridad (estando esta vacía)
- Mientras está saliendo otro paquete de clase con menor prioridad
- Retardo máximo producido si el paquete es de 1500 bytes y la línea de 256Kbps:

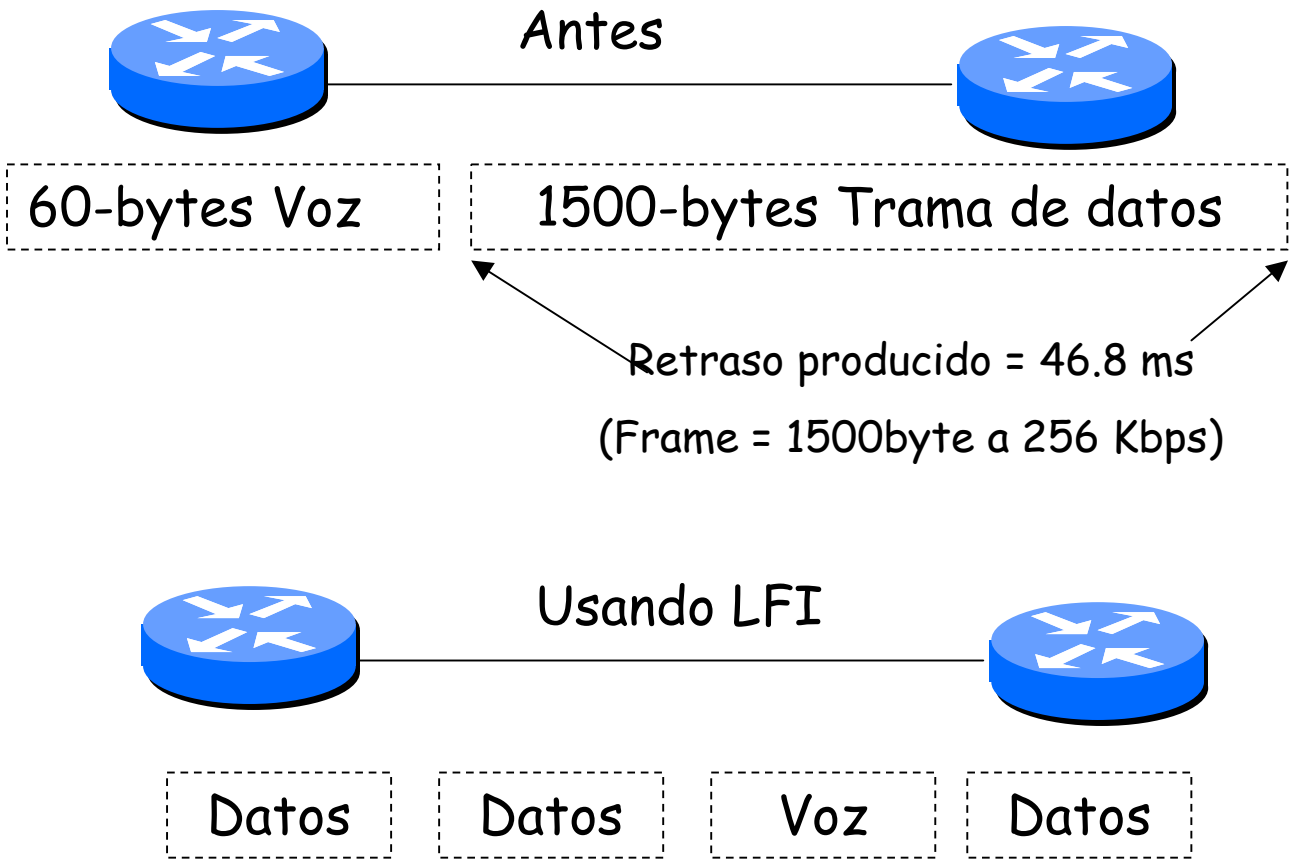
  (1500*8) bits / 256 Kbps = 46.8 ms!

Solución:

- Fragmentar los paquetes de datos
- Ej.: límite fragmentos “de 10ms”
- Es decir, tamaño de un paquete igual a máximo que se pueda enviar en 10 ms
- Insertar paquete de VoIP entre estos paquetes
- Asegura un retraso mucho menor
- ¡¡ Los paquetes VoIP no deben fragmentarse !!



Ejemplo de uso de LFI

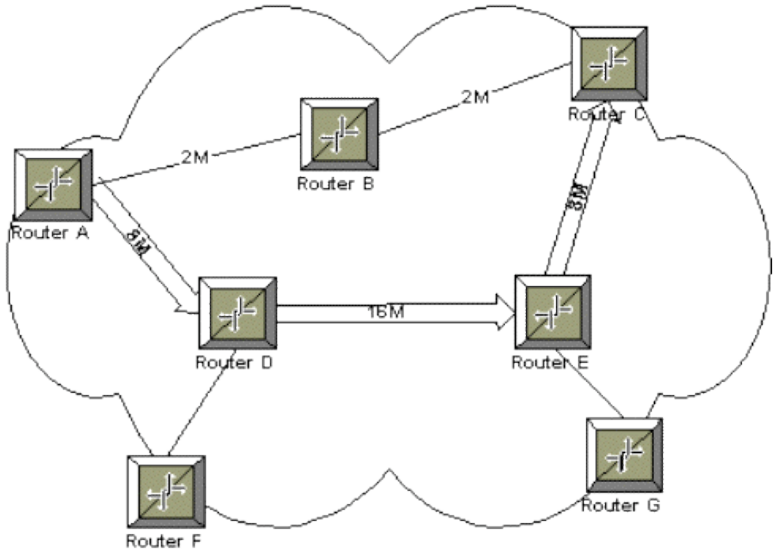


QoS routing

QoS Routing

- 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 costos (computacionalmente hablando)
- *Constraint-based Routing*
  - Calcular caminos teniendo en cuenta no solo QoS sino también políticas

$A \rightarrow C$: 4M



Arquitecturas

Propuestas del IETF

- **IntServ** (Integrated Services)
  - Filosofía: reserva de recursos
  - Cada router del trayecto ha de tomar nota y efectuar la reserva solicitada
- **DiffServ** (Differentiated Services)
  - Filosofía: priorización de tráfico
  - El usuario marca los paquetes con un determinado nivel de prioridad
  - Los routers van agregando las demandas de los usuarios y propagándolas por el trayecto
  - Esto le da al usuario una confianza razonable de conseguir la QoS solicitada
- Pueden coexistir

IntServ: Características

- RFC 1633
- Para cada flujo (puede ser agregado) reserva recursos en todo el camino
- Orientado a conexión
- Requiere un protocolo de señalización que soporten todos los routers
- No requiere modificar los protocolos existentes

IntServ: Servicios

- *Best Effort*
- *Controlled load service*
  - RFC 2211
  - *"commitment … to provide … with service closely equivalent to unloaded best-effort"*
  - Prácticamente sin pérdidas
  - No da garantías estrictas
- *Guaranteed service*
  - RFC 2212
  - *"provides firm (mathematically provable) bounds on end-to-end datagram queueing delays."*
  - Garantías de BW
  - Retardo acotado
  - Sin pérdidas en buffers
  - Garantías estrictas

IntServ: *Flow parametrization*

filterspec (*Filter specification*)

- Determina qué paquetes forman el flujo
- Flujo identificado en base a IPs + puertos
- Separa en diferentes colas

flowspec (*Flow specification*)

- **Tspec** (*Traffic specification*)
  - Descripción del tráfico
  - Parámetros de un *tocken bucket* por el que pasa el tráfico
  - Mean rate, tocken bucket depth, max rate, max packet length
- **Rspec** (*Service Request specification*)
  - Requisitos de QoS impuestos a la red
  - BW, retardo, probabilidad de pérdida

IntServ: *Signaling*

- Requisitos
  - Debe poderse usar en redes IP
    - Emplear tablas de rutas existentes
    - Reaccionar ante cambios de rutas
  - Soportar multicast
    - Flujos que se agregan en árbol
  - Pequeña sobrecarga
    - Pocos mensajes y pequeños
  - Modular y fácil de extender

- Resultado:
  - RSVP (*Resource reSerVation Protocol*)
  - RFC 2205
  - *Soft state (periodic updates)*
  - ¡ No sirve para calcular el camino !
  - Empleado en IntServ, DiffServ, MPLS, …

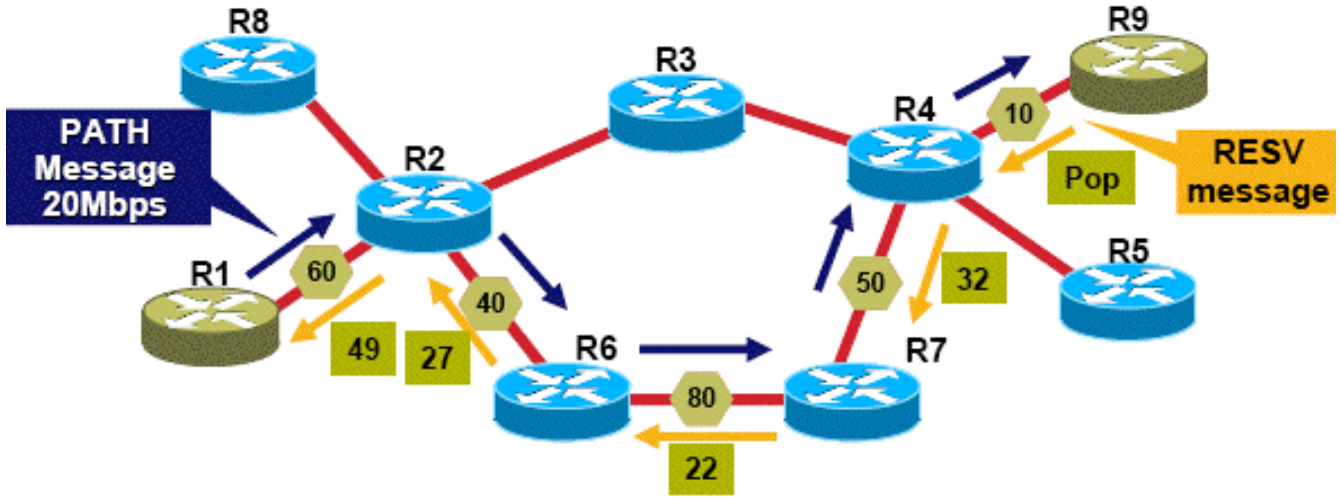
RSVP: Mensajes

PATH

- Desde fuente de tráfico, Tspec
- Establecer camino
- Puede hacerse CAC

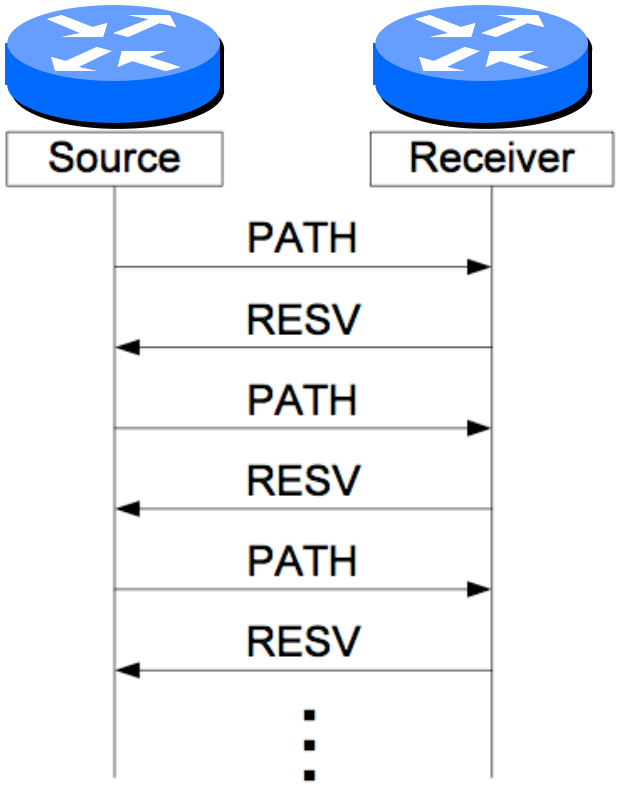
RESV

- Camino inverso al PATH
- Incluye Rspec
- Hacer la reserva en los routers



RSVP: *states*

- Actualizaciones periódicas refrescan el estado
- Se libera al dejar de recibir actualizaciones
- Alternativa (no soportada): Hard state
  - Se mantiene hasta liberarlo explícitamente
  - Requiere algoritmo ante errores

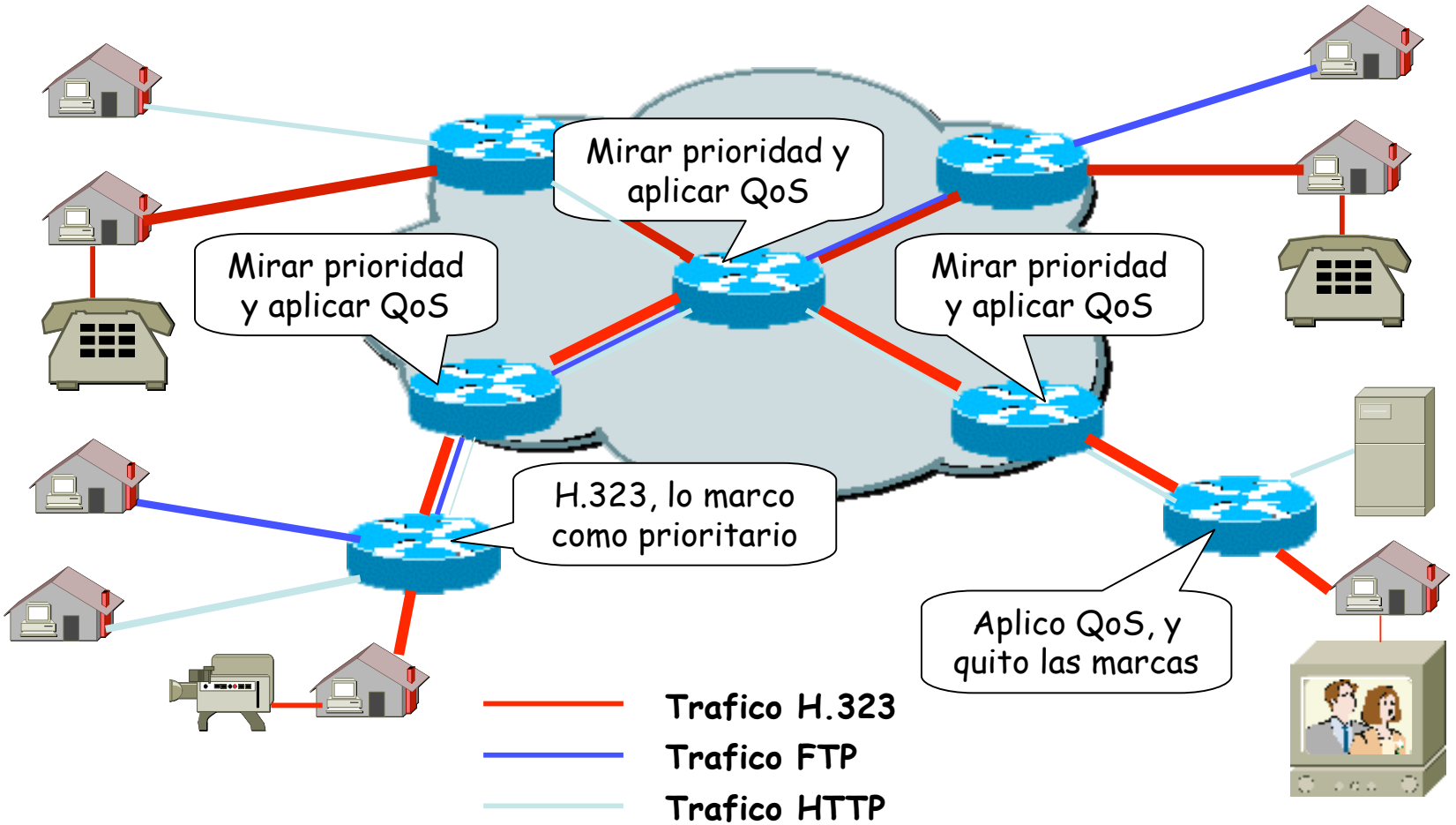


DiffServ

- IntServ no escala bien

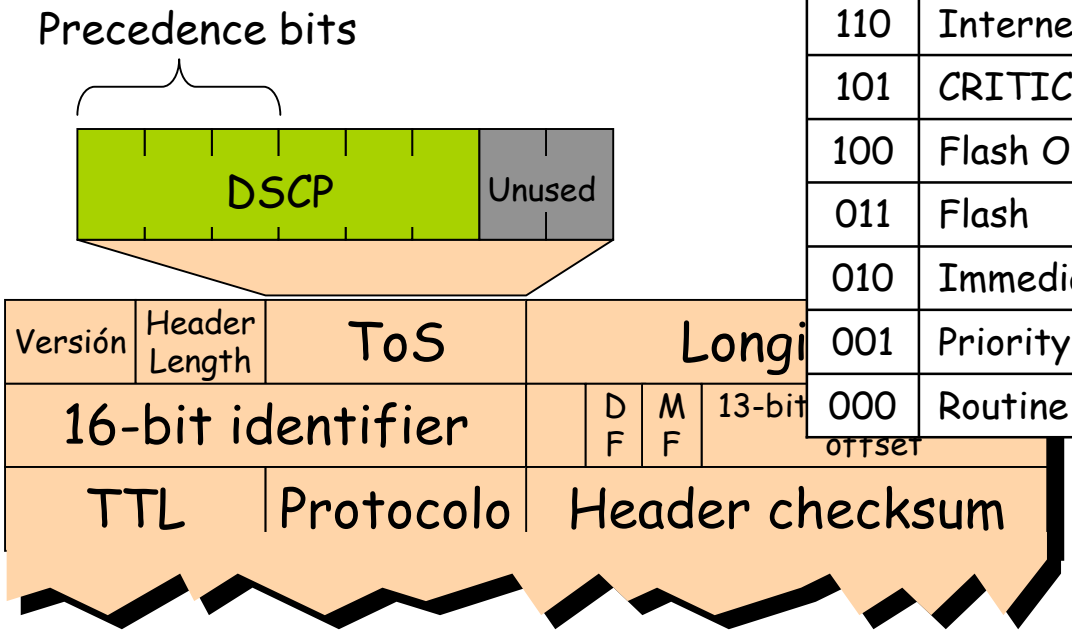
- RFC 2475, 2638
- Clasificar el tráfico en pocas clases
- Clasifican los *ingress routers* (complejidad en la frontera) con un *codepoint* en la cabecera IP
- DiffServ mapea en cada nodo el *codepoint* en el paquete a un PHB en concreto
- PHB = *Per Hop Behavior*
  - El tratamiento que se le da al paquete en cuestión de scheduling y gestión de cola en ese nodo
  - El mapeo *codepoint* ↔ *PHB* debe ser configurable
- No es sensible a los requisitos de un flujo individual

Ejemplo


Mirar prioridad y aplicar QoS
Mirar prioridad y aplicar QoS
Mirar prioridad y aplicar QoS
H.323, lo marco como prioritario
Aplico QoS, y quito las marcas
Trafico H.323
Trafico FTP
Trafico HTTP

DiffServ: DSCP

- Originalmente 3 primeros bits del ToS = *Precedence* bits
- Fácil mapeo a 802.1p bits
- ToS ahora se llama DS (*Differentiated Services*)
- 6 de sus bits son el DSCP (*Differentiated Services CodePoint*)
- *Class Selector Codepoint:*
  - CSx = XXX000
  - Compatibilidad con *precedence*



CSx	Significado histórico	Uso generalizado
111	Network Control	Tráfico de control (ej: routing)
110	Internetwork Control	
101	CRITIC/ECP	Voz
100	Flash Override	Vconf., streaming
011	Flash	Call signaling
010	Immediate	Libres para clasificar tráfico de datos
001	Priority	
000	Routine	*default*

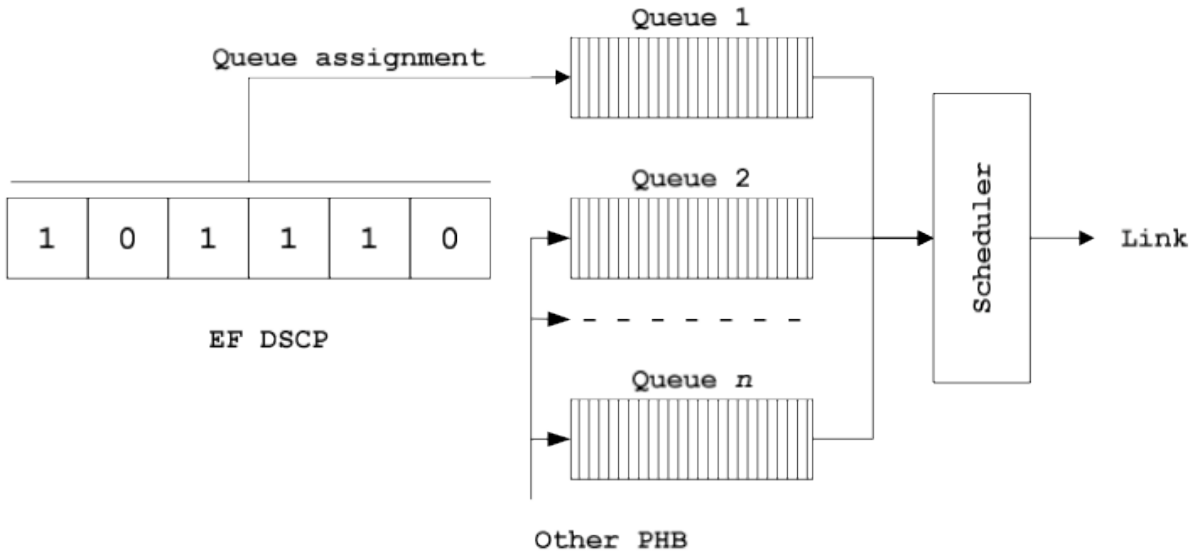
PHBs

PHBs

- *Best-Effort* (BE)
- *Assured Forwarding* (AFxy)
- *Expedited Forwarding* (EF)

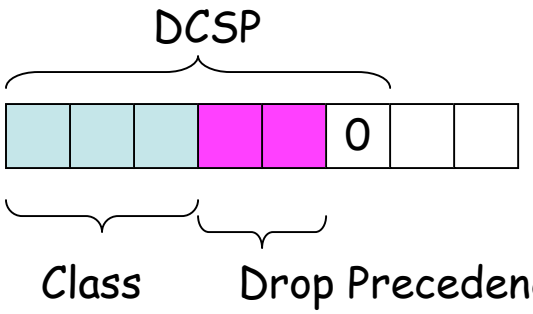
PHB: Expedited Forwarding (EF)

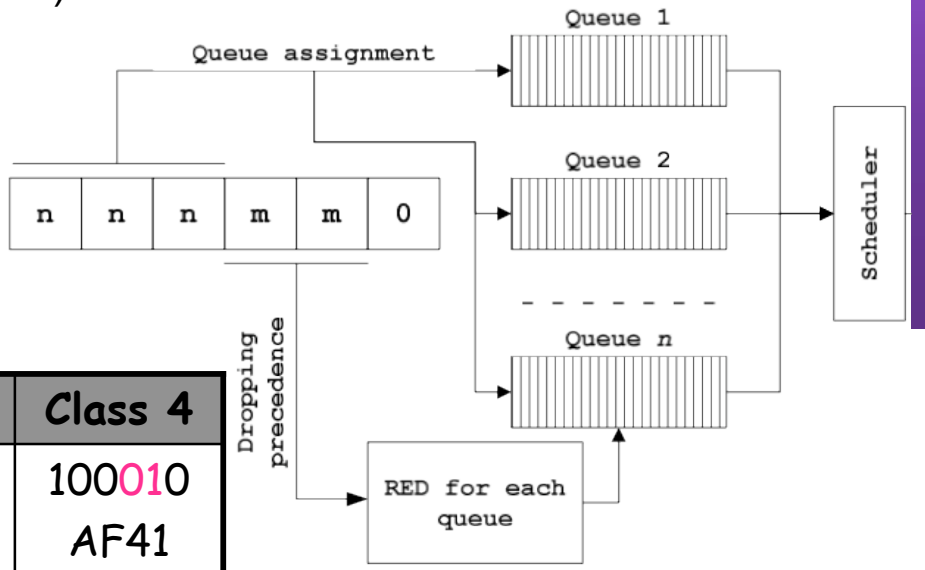
- RFC 2598
- Alta prioridad
- DSCP 10110
- *"…the departure rate of the aggregate's packets from any diffserv node must equal or exceed a configurable rate."*
- *"The EF traffic SHOULD receive this rate independent of the intensity of any other traffic attempting to transit the node. "*
- Este PHB se puede implementar con PQ, WRR, CBQ, etc.



PHB: Assured Forwarding (AF)

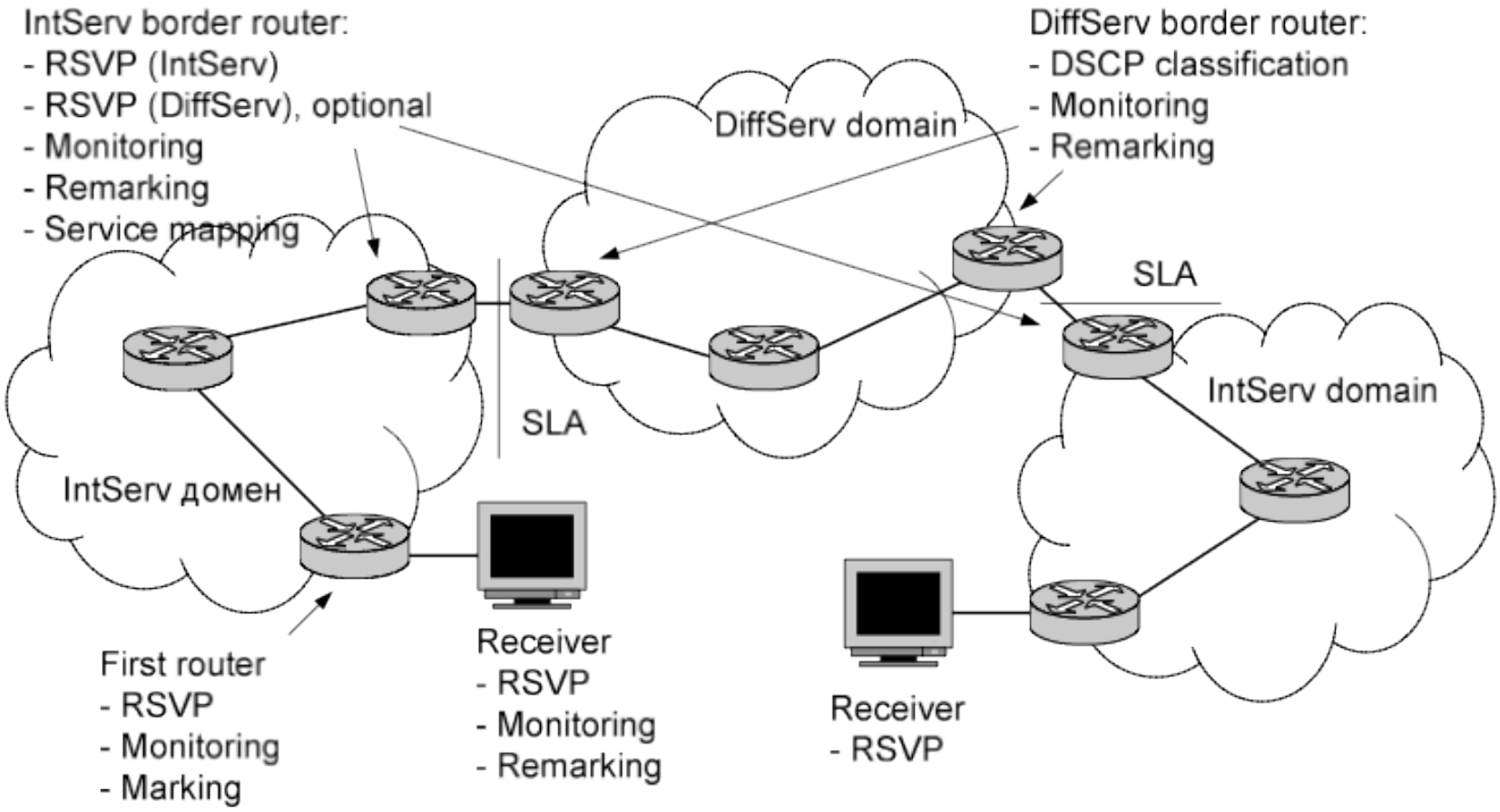
- RFC 2597
- Se definen 4 clases (AF1x, AF2x, AF3 y AF4x)
- Cada una tiene una reserva en cada nodo (BW, buffer)
- Cada una con 3 probabilidades de descarte (*drop*)
- DSCP xxxyy0 : xxx la clase, yy la prob. Descarte
- Debe emplear AQM (RED, WRED, …)





Drop	Class 1	Class 2	Class 3	Class 4
Low	001010 AF11	010010 AF21	011010 AF31	100010 AF41
Medium	001100 AF12	010100 AF 22	011100 AF32	100100 AF42
High	001110 AF13	010110 AF23	011110 AF33	100110 AF43

IntServ over DiffServ



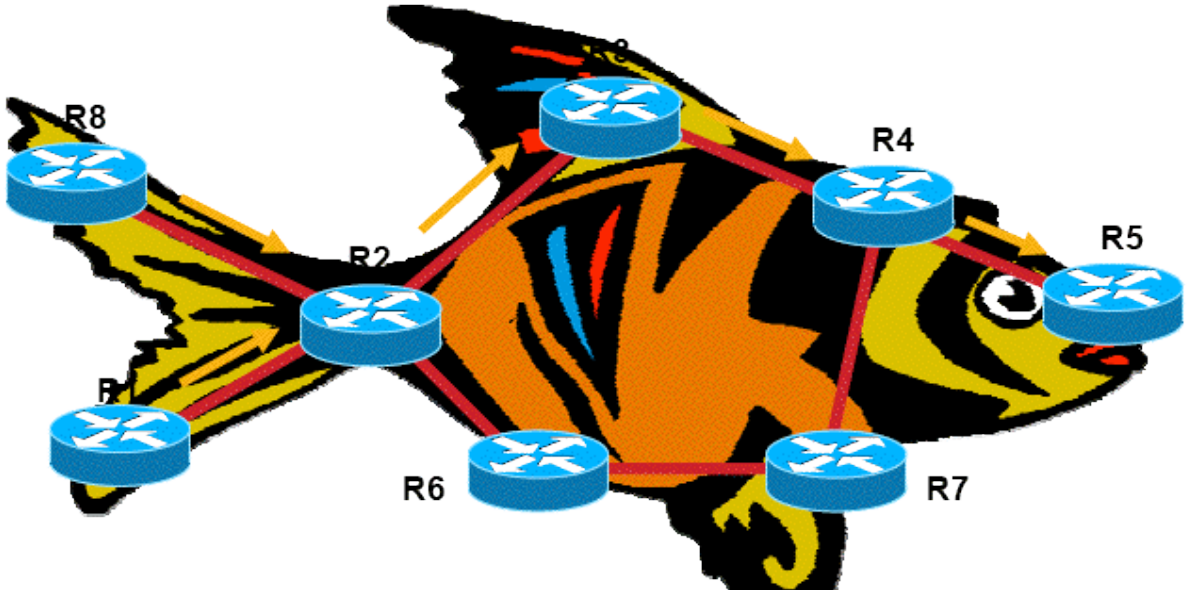
Traffic Engineering

Traffic Engineering (TE)

- RFC 3272 (Overview and Principles of Internet Traffic Engineering)
- *".. that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks."*
- *"[TE] encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic."*
- Existe desde las redes telefónicas clásicas
- Proceso:
  - *Measurement*: desde el nivel de paquete al de flujo, usuario, agregado de tráfico o red
  - *Modeling, Analysis and Simulation*
  - *Optimization*: desde real-time optimization a network planning

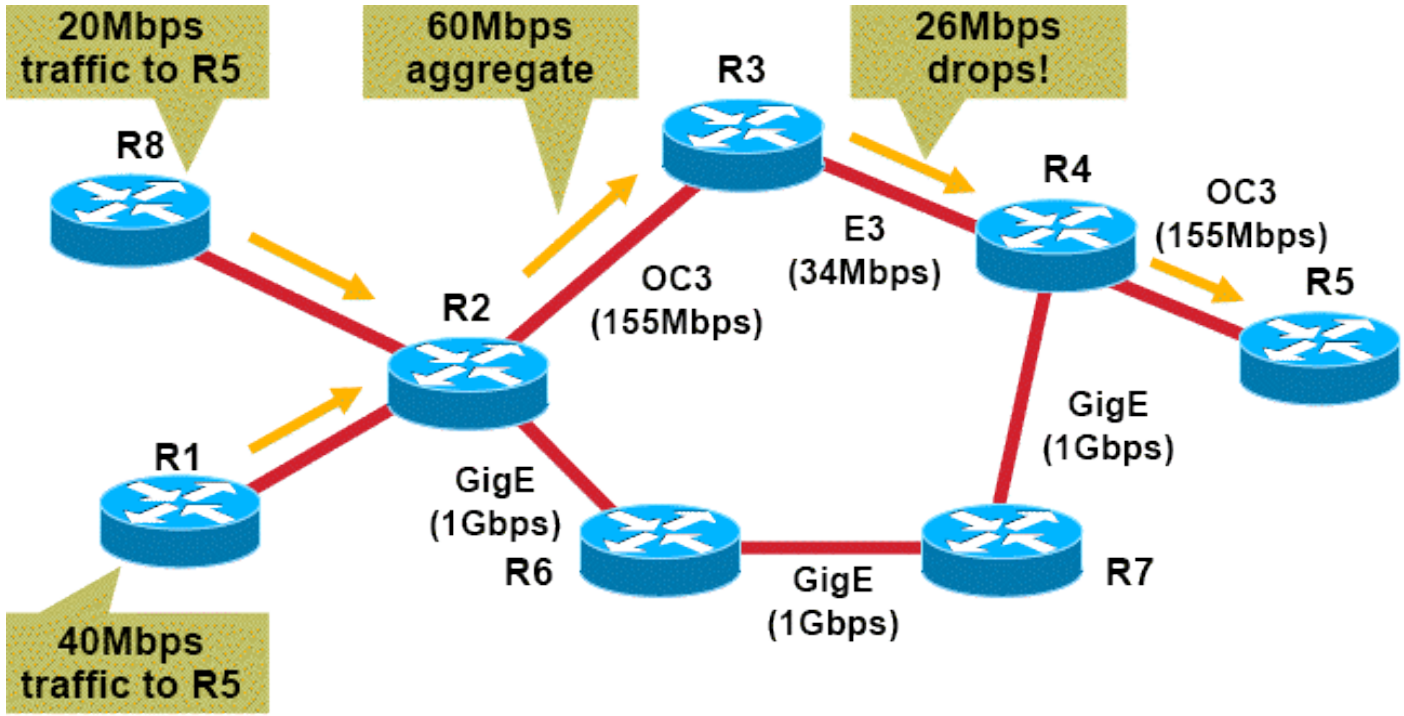
Traffic Engineering

- Network Engineering
  - Construir la red para transportar el tráfico esperado (¡predecir!)
- Traffic Engineering
  - Manipular el tráfico para encajar en la red
  - Prevenir enlaces congestionados y otros infrautilizados
- No podemos contar con predecir los patrones de tráfico
- Seguramente tendremos una red con BW simétricos pero flujos asimétricos
- RFC 2702 - Requirements for Traffic Engineering over MPLS
- Ejemplo: *"The Fish"*



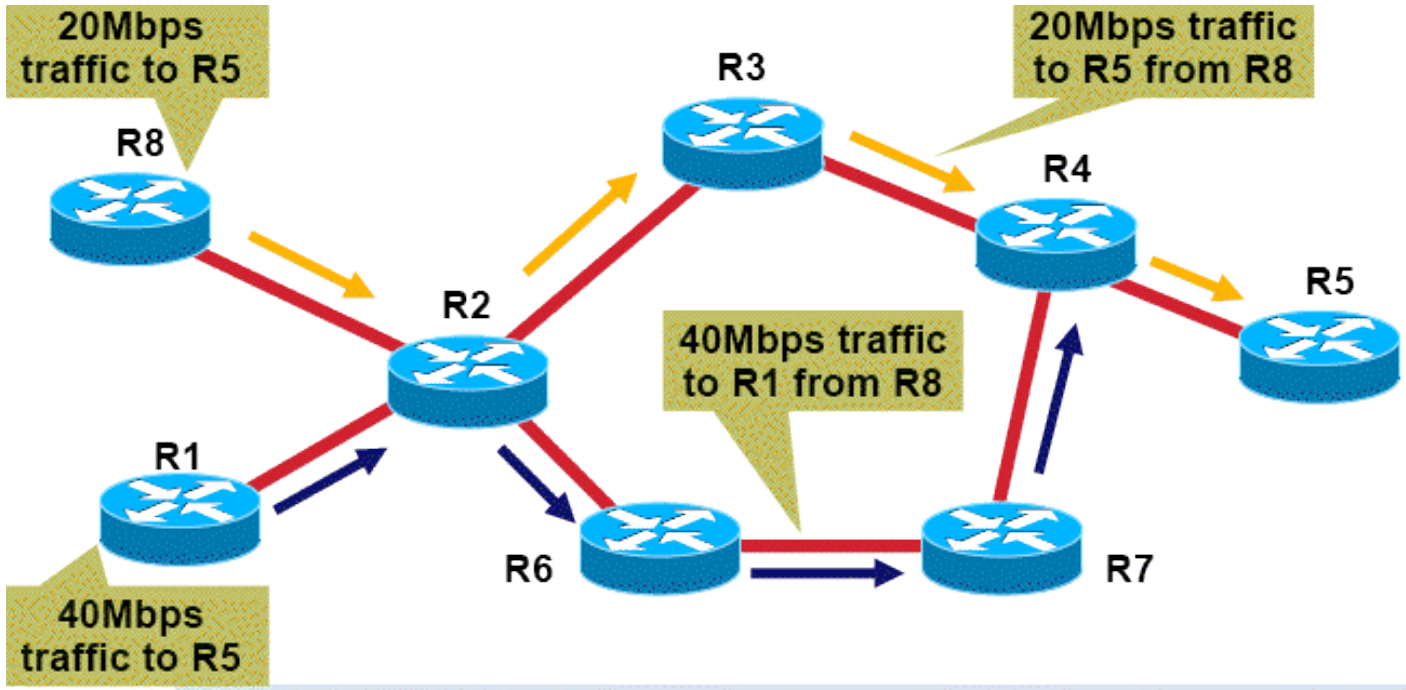
The Fish

- Sin TE



The Fish

- Con TE

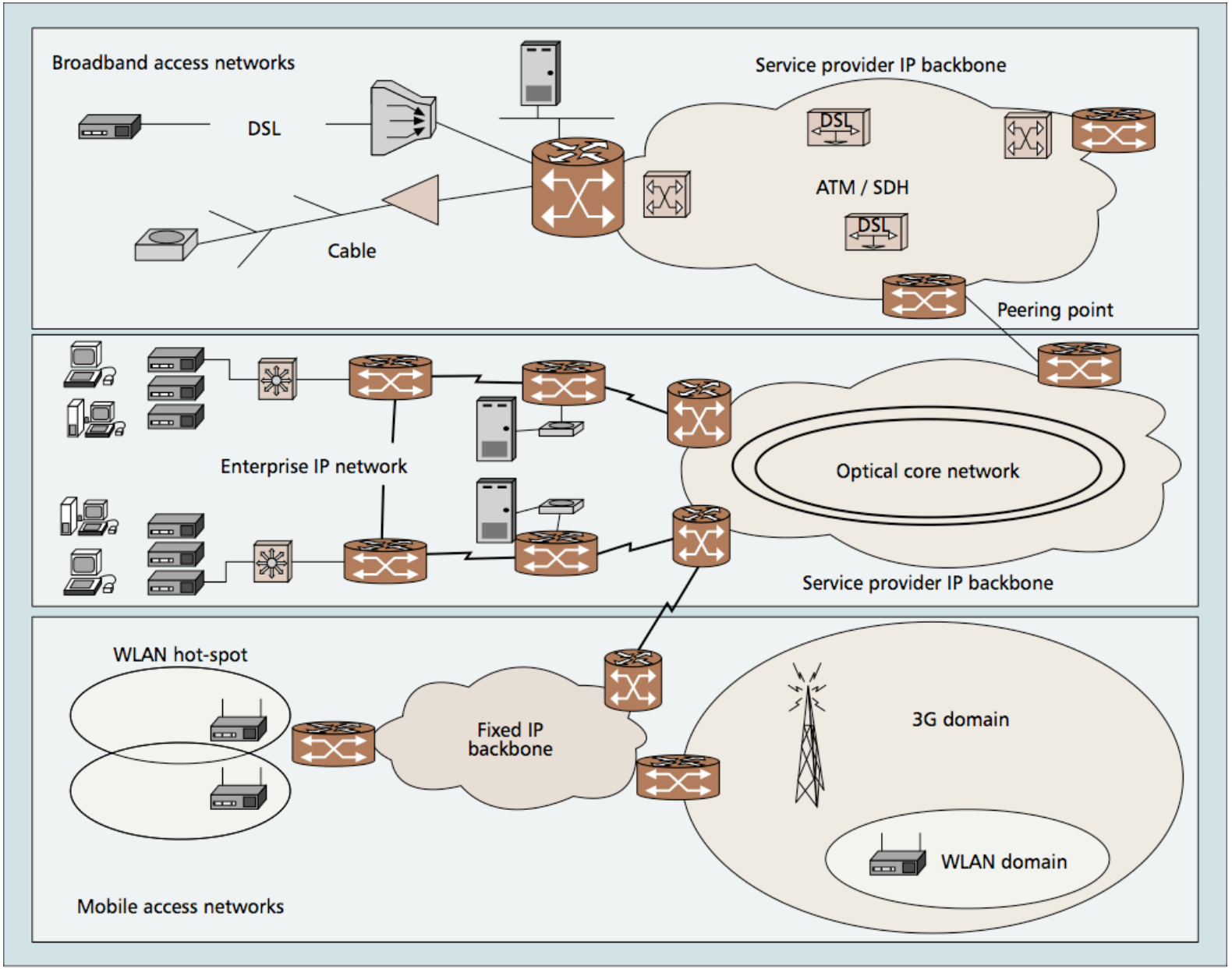


- MPLS Labels can be used to engineer explicit paths
- Tunnels are UNI-DIRECTIONAL

Normal path: R8 → R2 → R3 → R4 → R5

Tunnel path: R1 → R2 → R6 → R7 → R4

Redes heterogéneas



Resumen

- Técnicas:
  - Clasificación y marcado
  - Policing y shaping
  - Scheduling
  - Queue management
  - CAC
  - QoS routing
- Arquitecturas:
  - IntServ
  - DiffServ