QoS: Scheduling (2)

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

Grado en Ingeniería en Tecnologías de Telecomunicación, 3º

Temas de teoría

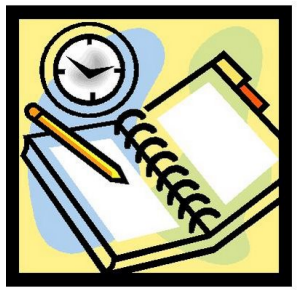
1. Introducción
2. QoS
3. Encaminamiento dinámico en redes IP
4. Tecnologías móviles
5. Otros temas

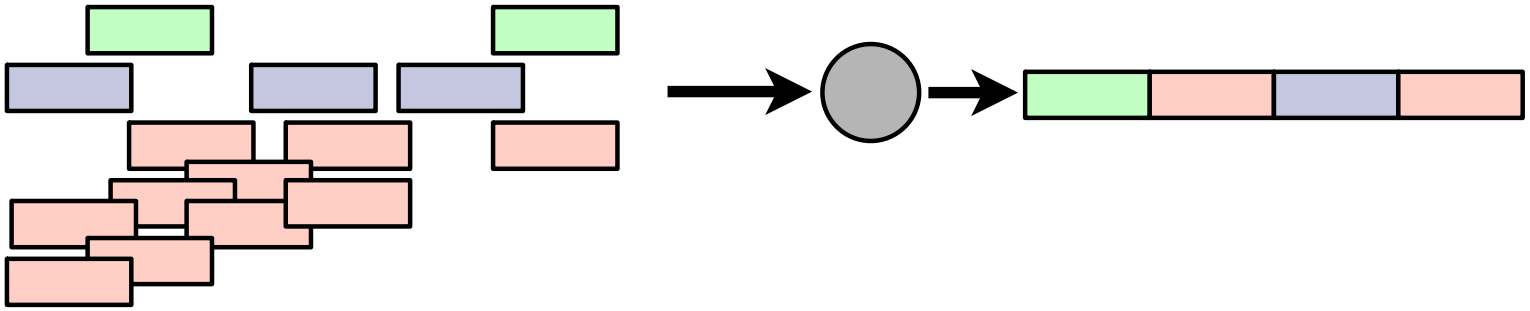
Objetivos

- Conocer las características y el funcionamiento de los planificadores más habituales

Scheduling, decíamos…

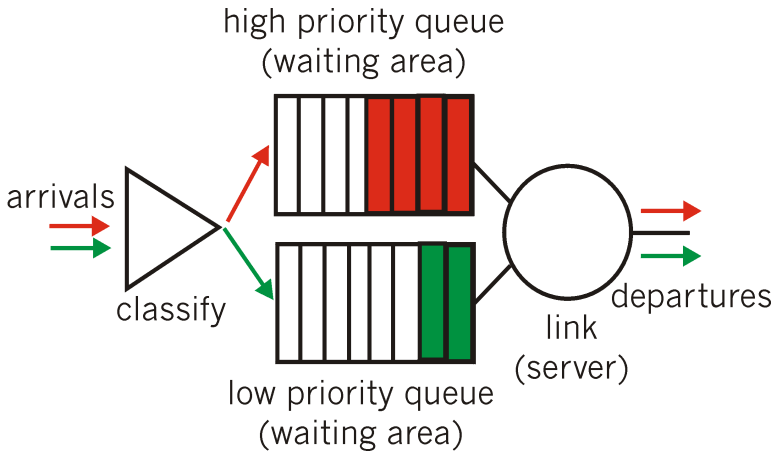
- Permite compartir recursos
- Emplea una disciplina de planificación para decidir la siguiente petición a atender
- Puede tener lugar en diferentes niveles de una pila de protocolos
- Nos centraremos en compartir la capacidad de un enlace
- Por ejemplo en el nivel de aplicación sería necesario para decidir la siguiente petición a un servidor que atender
- Nos centraremos en planificadores conservativos en trabajo (*work conserving*): están inactivos solo si la cola está vacía

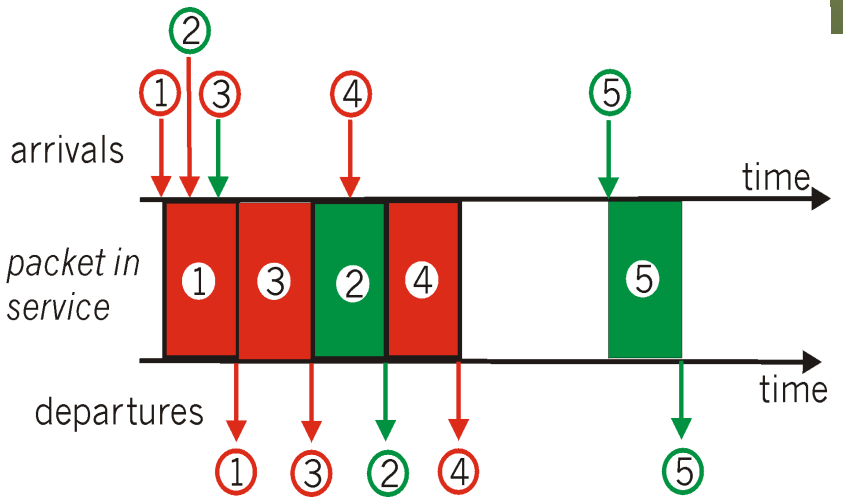




Priority Queueing (PQ)

- Paquetes en cola de mayor prioridad se envían siempre antes que paquetes en colas de menor prioridad
- *Multilevel priority with exhaustive service:* Los paquetes en una cola de menor prioridad no se envían hasta que todas las colas de mayor prioridad están vacías
- En cada cola FCFS
- Asegura que el tráfico importante reciba un servicio rápido
- Puede crear inanición (*starvation*), es decir, dejar fuera de servicio a tráfico menos prioritario
- Menor retardo en cola medio para un flujo a costa de mayor para otros.





Priority Queueing

- El número de niveles de prioridad depende del número de clases de retardo a crear
- Son típicas al menos 3:
  - Prioridad alta: mensajes urgentes, por ejemplo protocolos de control de red
  - Prioridad media: servicio garantizado
  - Prioridad baja: best-effort
- Otra posibilidad:
  - Prioriad alta: voz
  - Prioridad media: vídeo
  - Prioridad baja: resto de datos

- Un flujo de alta prioridad puede “ahogar” a otros de prioridad más baja
- Es vital un correcto control de admisión y policing para todo lo que no sea la clase más baja
- Sencillo de implementar
- El reparto del BW entre las clases no es max-min fair

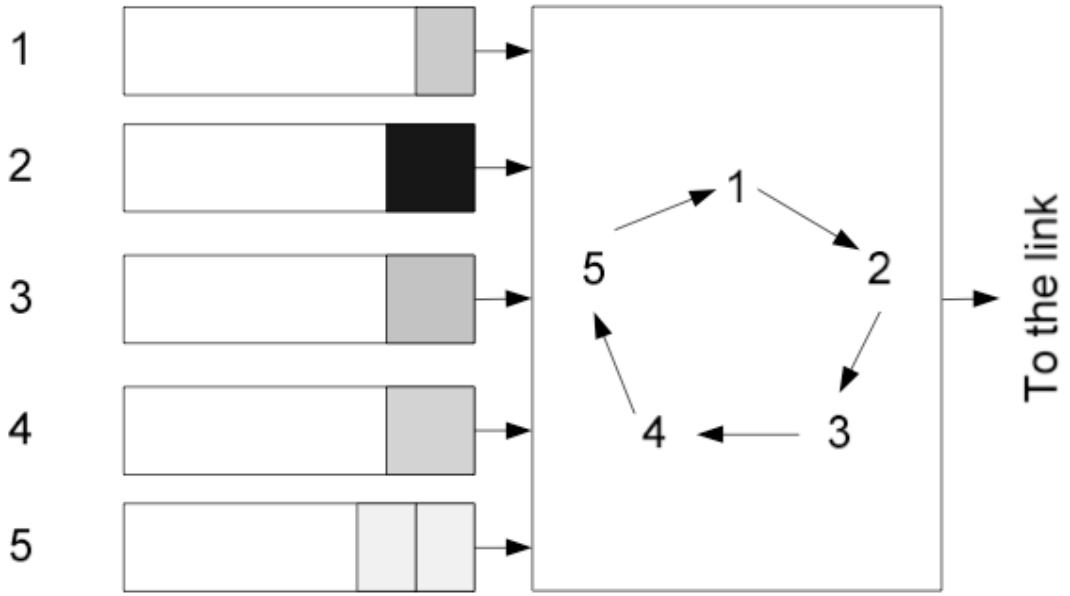
Priority Queueing

Ejemplo

- Se conoce el tamaño de la ráfaga más larga que puede llegar de un flujo (b_i)
- Para el flujo de prioridad mayor i = 1 el b_1 debe ser inferior al retardo de peor caso que se busque
- Para el flujo de prioridad i = 2 el retardo máximo es b_1+b_2
- El flujo de prioridad i = k sufre un retardo de caso peor de $\sum_{i=1}^{k} b_i$

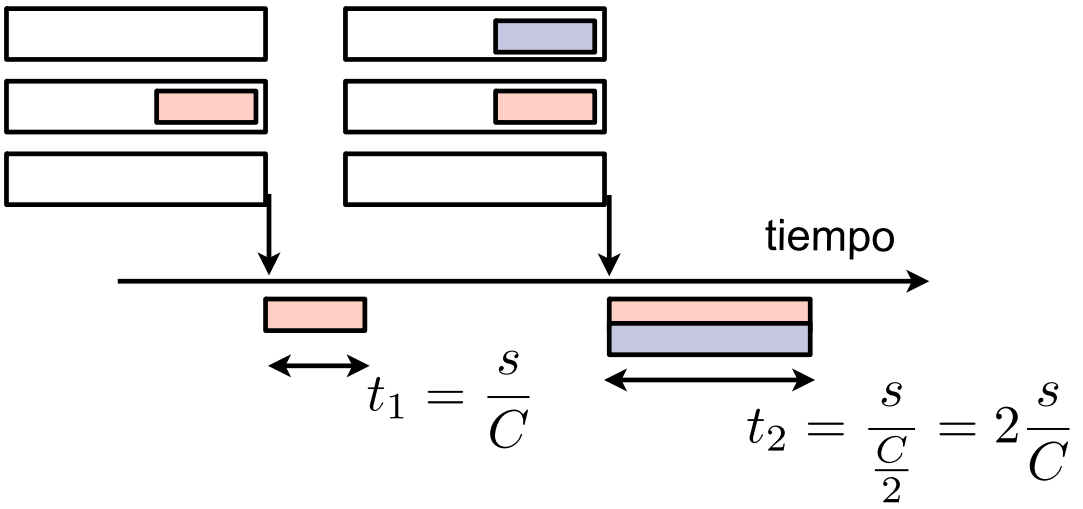
Round Robin (RR)

- Opera en “turnos” (*rounds*), conservativo en trabajo
- En cada turno visita cada cola (en *round-robin*)
- En cada cola FCFS
- Se sirve un número de paquetes o paquetes durante un cierto tiempo fijo (la diferencia es cómo afectan sus tamaños)



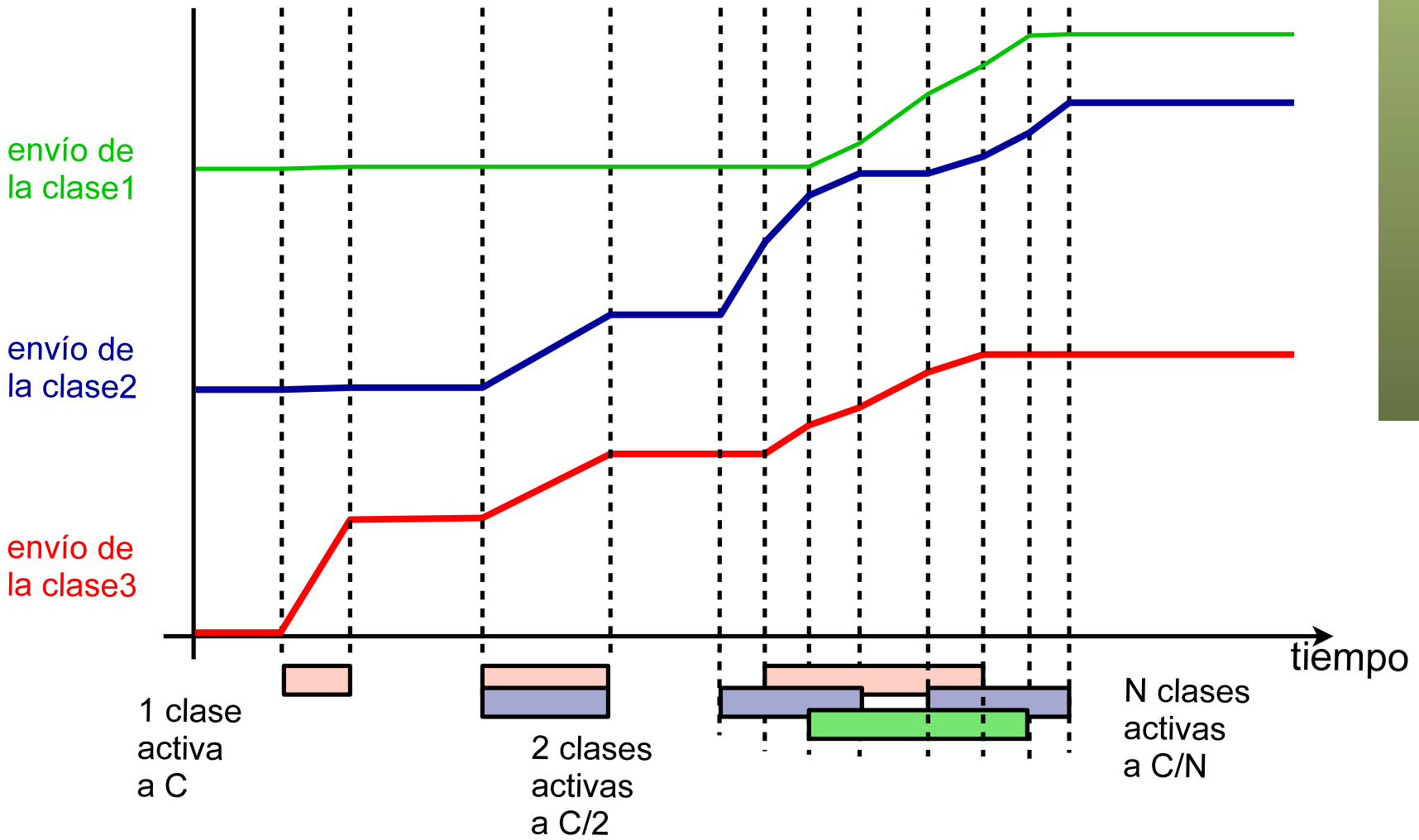
PS

- Para best-effort querríamos un reparto *max-min fair*
- Esto se puede lograr con un scheduler llamado *Processor Sharing*
- Es un planificador *work-conserving*
- Sirve de forma simultanea todas las colas, repartiendo la capacidad
- O se puede decir que las sirve por turnos (round robin) pero sirviendo una cantidad infinitesimal de cada una
- Si una cola está vacía pasa a la siguiente, de forma que su tiempo se está repartiendo entre el resto (y de ahí el max-min)
- Aproximación de tráfico como un fluido
- Es un planificador ideal e imposible de implementar, aunque se puede aproximar



Processor Sharing

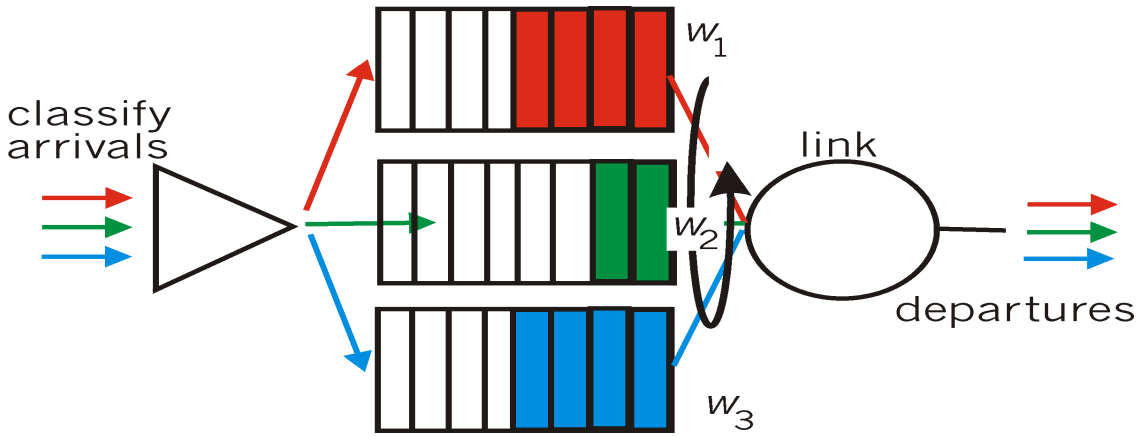
Ejemplo



GPS

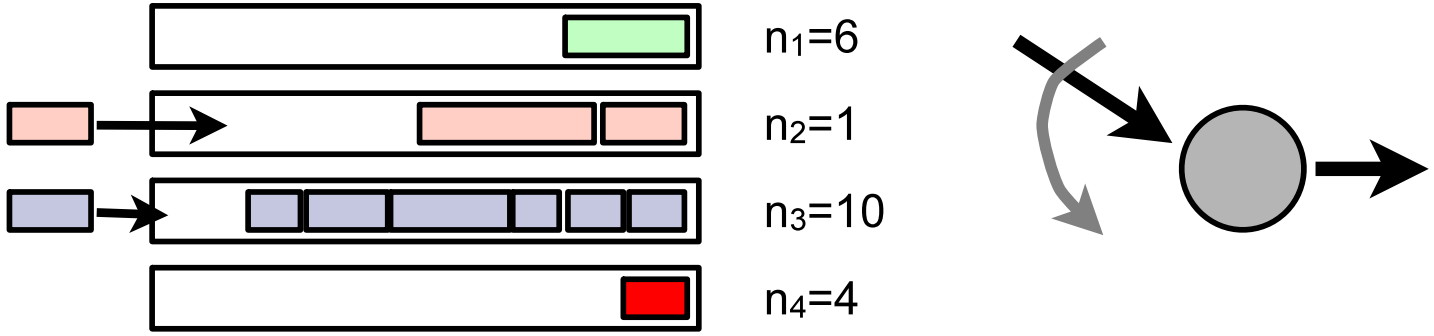
- Se puede asociar un peso $\phi(i)$ a cada cola y entonces la cantidad de servicio es proporcional al mismo
- Ofrece *weighted max-min fairness* y lo llamamos *Generalized Processor Sharing (GPS)*
- En cualquier caso, en la realidad no podemos servir fluidos sino que servimos paquetes así que solo podremos aproximarlo
- Round Robin es una aproximación a PS

$$c_i = C\frac{\phi(i)}{\sum \phi(i)}$$



Weighted Round Robin (WRR)

- Opera por “turnos”
- Se normaliza el peso por el tamaño medio de paquete en la clase $\frac{\phi(i)}{s_i}$
- Normaliza el resultado para que sean enteros
- En cada turno visita cada cola (en RR) y sirve tantos paquetes como su peso normalizado
- Ejemplo:
  - Pesos: 0.03, 0.05, 1 y 0.5
  - Tamaños medios: 50, 500, 1000 y 1200 bytes
  - Renormalizados según tamaños medios: 0.0006, 0.0001, 0.001 y 0.0004
  - Renormalizados a enteros: 6, 1, 10, 4



Weighted Round Robin (WRR)

- Necesita saber el tamaño medio de paquete de cada clase
- Más sencillo si los paquetes son de tamaño constante
- Es justo solo por encima de la escala de la duración del turno. Ejemplo:
  - Enlace T3 (45Mbps)
  - 500 PVCs ATM con peso 1 y 500 PVCs con peso 10
  - Supongamos que todos los PVCs tiene tráfico
  - Un turno requiere enviar: 500 x 1 + 500 x 10 = 5500 celdas ➔ 51.82 ms
  - Por debajo de una escala de 50ms unos PVCs reciben más que otros
- El retardo que sufre una clase depende del número de clases que haya
- Hay implementaciones que lo combinan con una cola de prioridad
- SRR (*Shaped Round Robin*):
  - Modo *Shaped*: Reserva una porción del BW a cada clase reservando un tiempo en cada round y dentro de él enviando los paquetes equiespaciados
  - Modo *Shared*: el tiempo no utilizado por una clase lo usan el resto
  - A partir de un cierto intervalo sirven los mismos paquetes de cada cola WRR y SRR
  - SRR los envía con un orden diferente, más entremezclados de las diferentes clases

Deficit Round Robin (DRR)

- Permite hacer un RR con pesos sin conocer tamaños medios de paquetes
- Veamos primero versión **sin pesos**
- Cada clase mantiene un contador de déficit inicializado a 0
- En cada turno se añade **q** (el *quantum*) al contador de cada clase si tiene paquetes por servir, si no se resetea
- El planificador visita cada clase y sirve el primer paquete de la cola si su tamaño es menor que su contador de déficit
- y decrementa el contador en el tamaño del paquete
- Ejemplo:
  - q = 1000 bytes
  - Tres clases A, B y C con paquetes de 1500, 800 y 1200 bytes
  - Turno 1: Clase A contador a 1000, clase B se sirve paquete y el contador se queda en 200, clase C contador a 1000
  - Turno 2: Clase A se sirve paquete y contador a 500, clase B se resetea pues no tiene paquetes (para que no acumule), clase C se sirve paquete y contador a 800

Deficit Round Robin (DRR)

- En la versión con pesos el *quantum* es el peso de cada clase
- El *quantum* debería ser al menos del tamaño máximo de paquete para servir alguno en todos los turnos
- Es sencillo de implementar

Resumen

- Priority Queueing
- Round Robin
- Weighted Round Robin
- Deficit Round Robin