Algoritmos planificador corto plazo

Se atienden los procesos en orden de llegada a la cola de READY

ES **SIN** DESALOJO

- Proceso finaliza
- Proceso se bloquea
- Proceso libera CPU voluntariamente

→ **Libera CPU**

- No optimiza métricas (throughput, tiempo de espera, tiempo de respuesta...)

- Es simple
- Tiene poco overhead
- Es “justo” respecto al orden de llegada

FIFO

Proceso	Tiempo de arribo	Ráfaga CPU	Ráfaga IO	Ráfaga CPU
P1	0	4	1	3
P2	1	1	3	1
P3	5	2	-	-

En t = 5 **P1** volverá de una IO y **P3** ingresará al sistema.. Qué ocurre??

Tiempo de espera -> el tiempo en el que un proceso está esperando en la cola de READY

SJF (Shortest Job First)

Atiende al proceso cuya siguiente ráfaga de CPU es más corta

ES **SIN** DESALOJO

Libera CPU

Para = ráfaga FIFO desempata

- Puede generar starvation
- Un proceso puede monopolizar la CPU

Favorece

- Throughput
- T de espera promedio
- T de respuesta

SRT (Shortest Remaining Time FIrst - SJF con desalojo)

Atiende al proceso cuyo remanente de la siguiente ráfaga de CPU (lo que no ejecutó aún) es más corto

ES **CON** DESALOJO

- Libera CPU
- LLega un proceso nuevo al sistema
- Un proceso se desbloquea

Un proceso en ejecución podrá ser **desalojado** en caso de que llegue a READY un proceso con siguiente ráfaga más corta

Todo muy lindo pero...

... cómo lo implementamos?

No podemos saber cuánto va a durar la próxima ráfaga..
...pero podemos estimarla

Se calcula como un promedio de la duración de las ráfagas anteriores:

Est(n) = Estimado de la ráfaga anterior
R(n) = Lo que realmente ejecutó la ráfaga anterior en la CPU
Est(n+1) = El estimado de la próxima ráfaga

$$Est(n+1) = \alpha\, R(n) + (1 - \alpha)\, Est(n) \qquad \alpha \in [0,1]$$

Atiende al proceso con mayor prioridad

PUEDE SER **CON/SIN** DESALOJO

Replanifica cuando ...

SIN
- Libera CPU

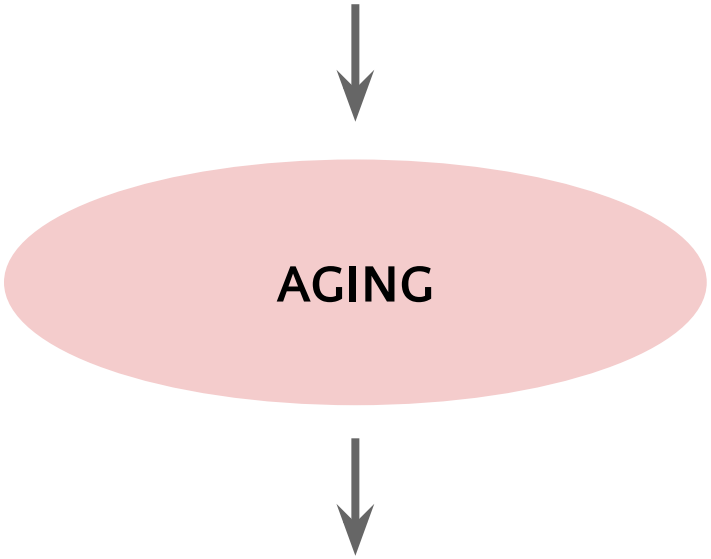
CON
- Libera CPU
- Desbloqueo
- Proceso nuevo

Para = ráfaga
FIFO
desempata

- Puede generar starvation

Favorece
- Permite garantizar prioridades establecidas

Cómo se podría resolver el problema de starvation??



Vamos aumentando la prioridad del proceso hasta que llega a la máxima (0) y puede ser elegido


APLIQUEMOS AGING EN SJF!!
NO SE PUEDE MODIFCAR LA PRÓXIMA RÁFAGA!

HRRN (Highest Response Ratio Next)

Elige al proceso con mayor RR (Response Ratio)

$$RR = (S + W) / S \Rightarrow 1 + W / S$$

S = Duración próxima ráfaga CPU (service time)
W = Tiempo de espera en READY (wait time)

A mayor tiempo de espera → Mayor RR → Mayor prioridad

A mayor duración de ráfaga → Menor RR → Menor prioridad

ES SIN DESALOJO

Libera CPU

- Genera mucho overhead
- También requiere estimación

- Prioriza procesos IO bound pero sin generar starvation

Atiende por orden de llegada dejando que ejecute como máximo un QUANTUM de tiempo

ES **CON** DESALOJO

Replanifica cuando ...

- Libera CPU
- Fin de Q (interrupción timer)

- Puede generar mucho overhead (Process switch)

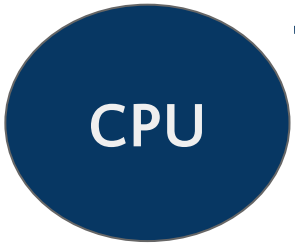
- Respeta orden de llegada
- Permite que todos ejecuten concurrentemente

PRÁCTICA

VRR (Virtual Round Robin)

Q' = Q – lo que ejecutó –> lo que le queda por consumir del Q fijo

READY – RR quantum fijo **Q**



Int Fin Q

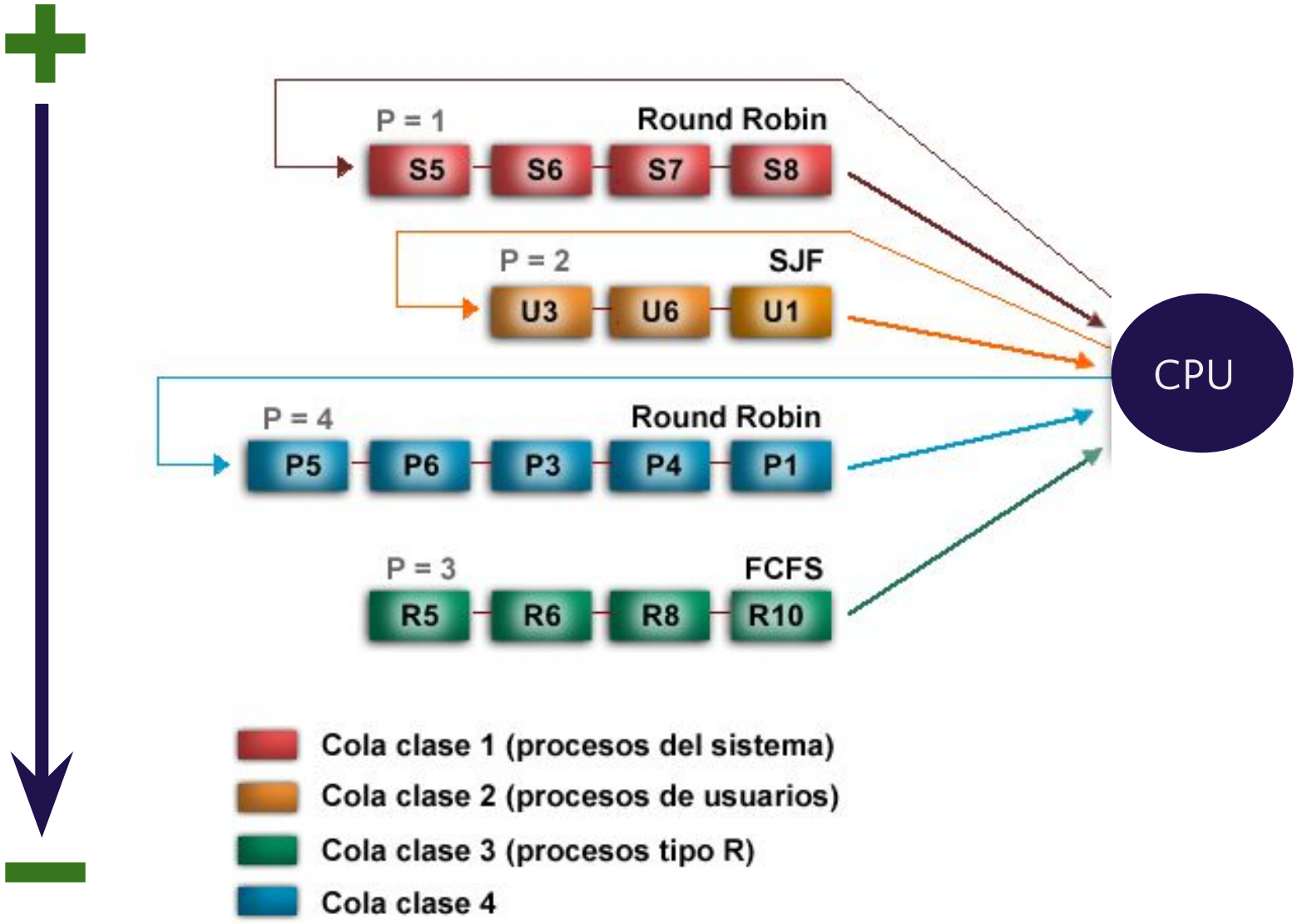
BLOCKED (no usó todo el Q)

READY – RR quantum variable **Q'**

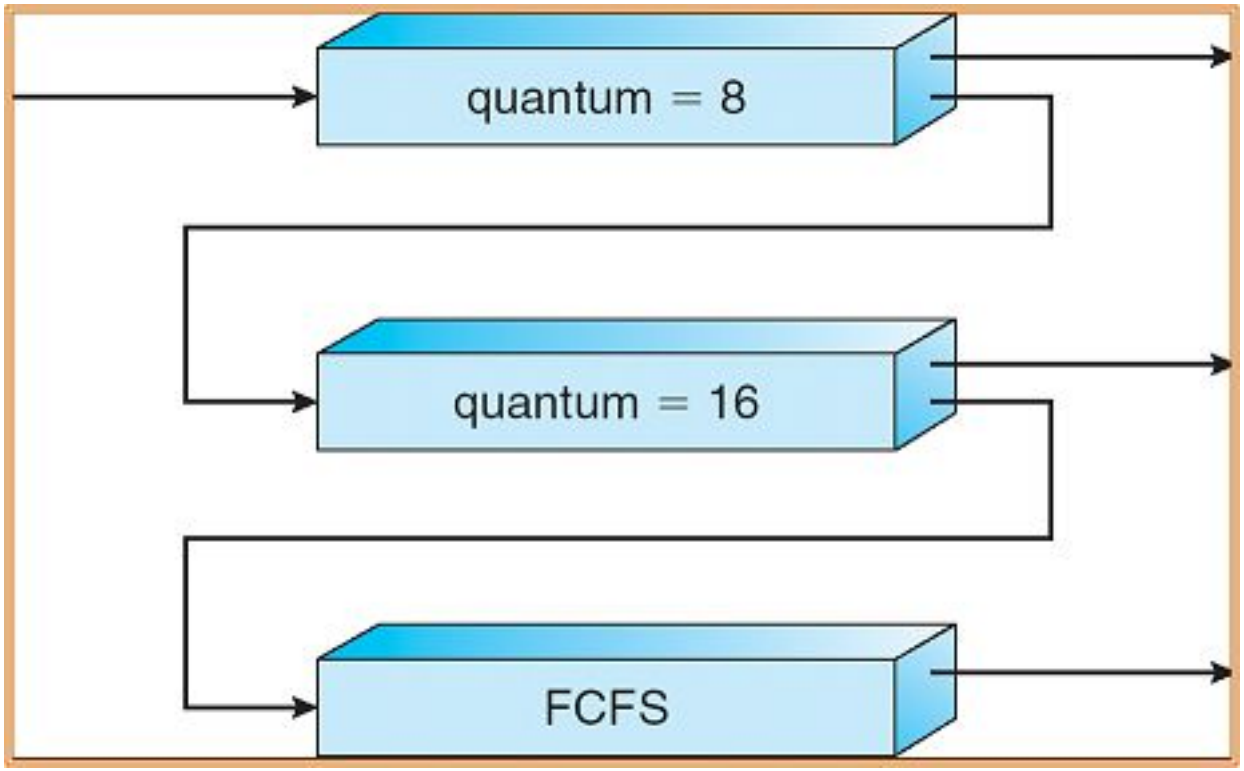
Ocurre el evento

PRIODIDAD

COLAS MULTINIVEL



COLAS MULTINIVEL RETROALIMENTADAS (FEEDBACK)



CPU

Preguntas?