Tareas de Tiempo-Real

Desde el punto de vista de la planificación, el sistema operativo considera a las tareas como procesos que consumen una cierta cantidad de tiempo de procesador, y a las que asignarles esa cantidad cada cierto tiempo. Tanto los datos que necesita cada tarea, como el código que ejecutan y los resultados que producen son totalmente irrelevantes para el planificador.

Concepto de Planificación

Planificación: forma o criterio que se sigue a la hora de decidir que proceso debe entrar en ejecución.

Multiprogramación: Ejemplo

Un sistema con dos procesos P1 y P2. Cada proceso se ejecuta durante 1 seg. y espera otro seg. Este esquema se repite 60 veces.

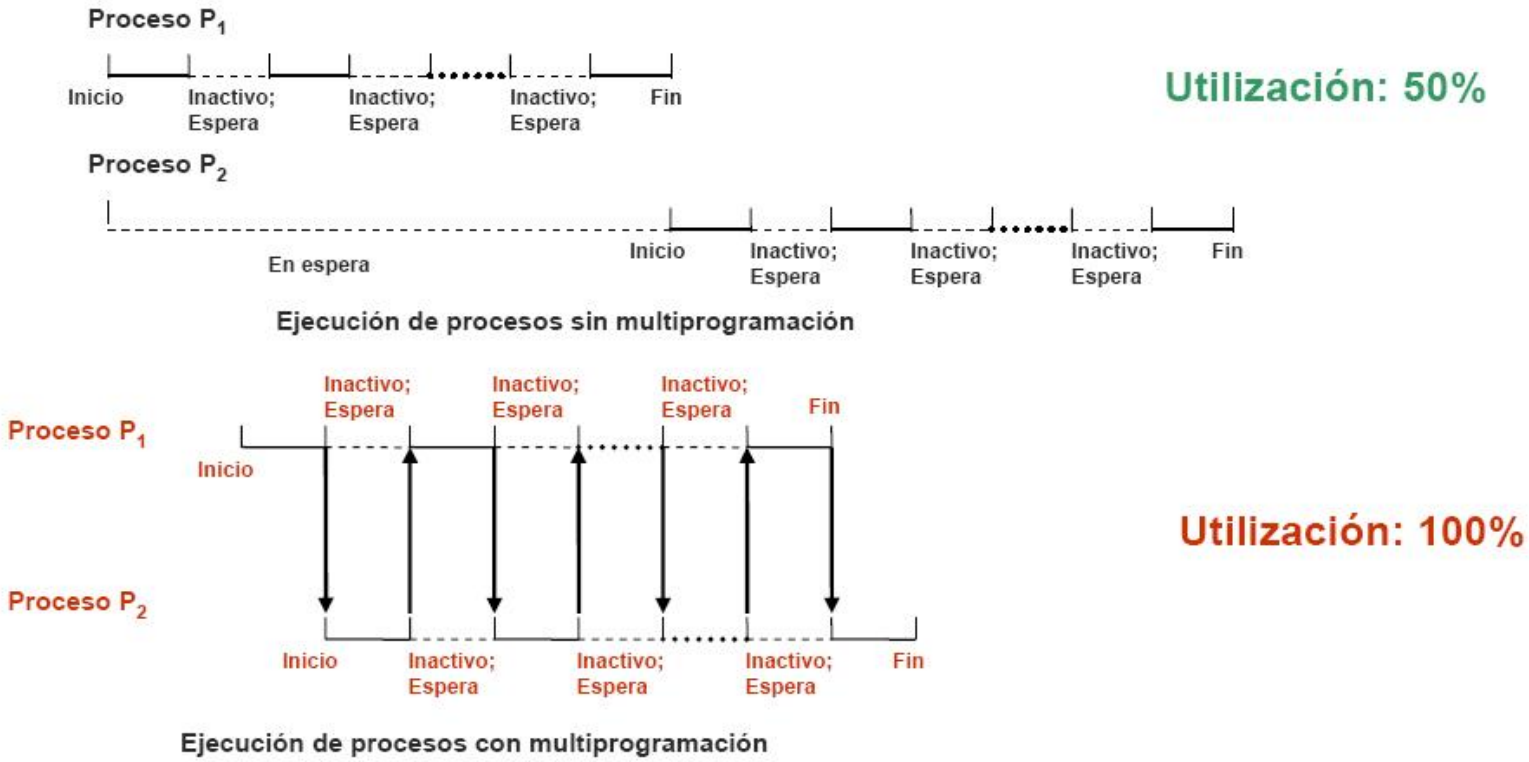


Figura 1: multiprogramacion

El planificador

- **Planificador:** parte del SO que se encarga de tomar la decisión de qué proceso entra en ejecución
- **Algoritmo de planificación:** criterio que utiliza el planificador para designar el proceso que entra en ejecución
- **Objetivos de un buen planificador:**
  - Equidad
  - Eficiencia (100 % utilización)
  - Minimizar el tiempo de espera
  - Aumentar el rendimiento (máximo número de trabajos por u.t)
- **Problemas de un planificador:**
  - Alcanzar todos los objetivos provoca contradicciones
  - El comportamiento de los procesos es único e impredecible

- El SO debe evitar que un proceso "monopolice"el uso del procesador
- El SO debe ejecutar cada cierto tiempo el planificador. Si el planificador es capaz de quitar a un proceso el procesador, la **planificación denomina expulsiva (preemptive)**
- Si cuando un proceso consigue el procesador ya no lo cede hasta que termina, se dice que la **planificación es no expulsiva**

Características de las políticas de planificación

Los objetivos que persigue toda política de planificación de tiempo real son:

- Garantizar la correcta ejecución de todas las tareas críticas.
- Ofrecer un buen tiempo de respuesta a las tareas aperiódicas sin plazo.
- Administrar el uso de recursos compartidos.
- Posibilidad de recuperación ante fallos software o hardware.
- Soportar cambios de modo, esto es, cambiar en tiempo de ejecución el conjunto de tareas. Por ejemplo: un cohete espacial tiene que realizar acciones muy distintas durante el lanzamiento, estancia en órbita y regreso; en cada fase, el conjunto de tareas que se tengan que ejecutar ha de ser distinto

Clasificación de las politicas de planificación

Los primeros planificadores se diseñaban a "mano", esto es, se construían durante la fase de diseño del sistema un plan con todas las acciones que tenía que llevar a cabo el planificador durantte la ejecución. Durante la ejecución, el planificador que estaba en la tabla se repite constantemente. A estos planificadores se les llama **CÍCLICOS**

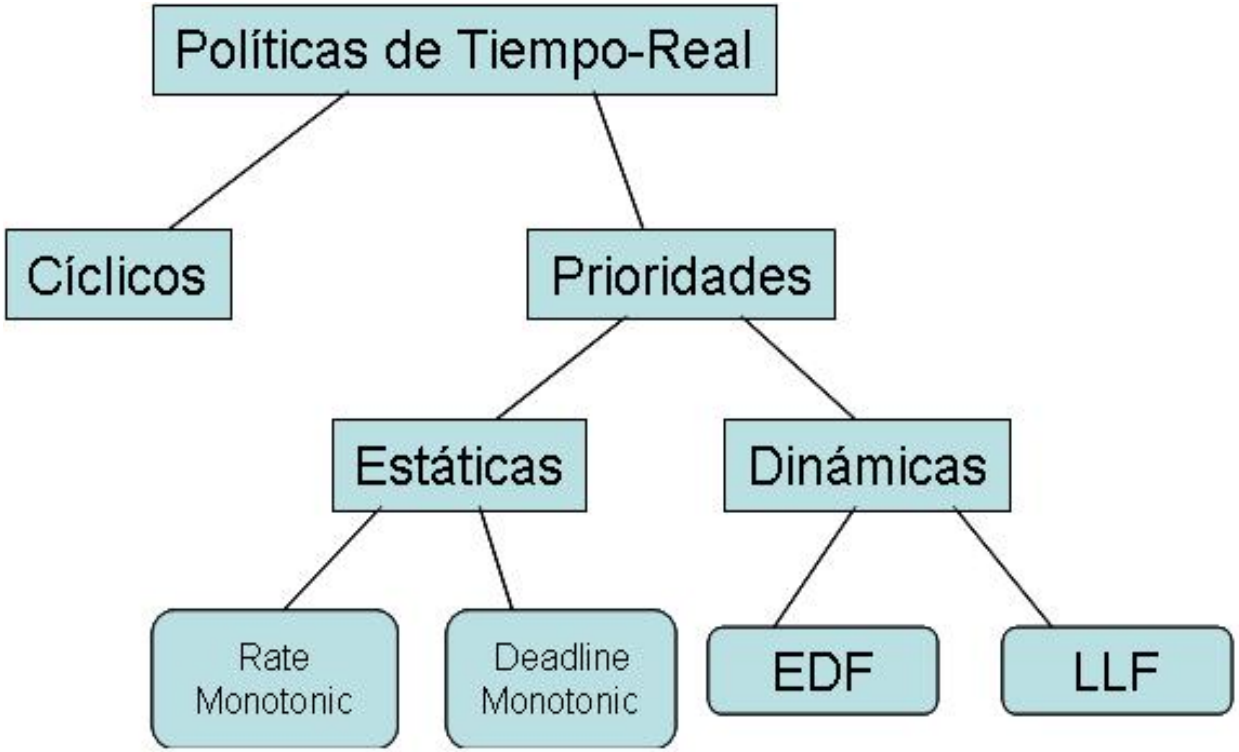


Figura 2: Clasificación de políticas planificación

Algoritmos de planificación

- Los distintos algoritmos de planificación tienen propiedades diferentes y pueden favorecer o perjudicar a un tipo u otro de procesos.
- Para comparar los algoritmos de planificación se han propuesto varios criterios:
  - Utilización de la CPU: mantener la CPU tan ocupada como sea posible (maximizar)
  - Rendimiento (Productividad): número de procesos que se completan por unidad de tiempo (maximizar)
  - Tiempo de retorno: tiempo transcurrido desde que se presenta el proceso hasta que se completa (minimizar)
  - Tiempo de espera: tiempo que un proceso pasa en la cola de procesos listos esperando la CPU (minimizar)
  - Tiempo de respuesta: tiempo que tarda un proceso desde

que se le presenta una solicitud hasta que produce la primera respuesta (minimizar)

- productividad, y minimizar los tiempos de retorno, de espera y de respuesta
- Puesto que conseguir todo lo anterior es imposible (contradictorio) lo que se desea es llegar a un compromiso entre todos los criterios de forma que se optimice el promedio
- Algoritmos:
  - Por orden de llegada **(FCFS)** ("First Come First Served")
  - Prioridad al trabajo más breve **(SJF)** ("Shortest Job First")
  - Prioridad al que resta menos tiempo **(SRTF)** ("Shortest Remaining Time First")
  - Planificación por prioridades (estáticas o dinámicas)

- Planificación circular o Round Robin"**(RR)**
- Planificación con clases de prioridades
- Planificación con múltiples colas realimentadas

Algoritmo FCFS ("First Come First Served")

- La CPU se asigna a todos los procesos en el mismo orden en que lo solicitan .
- Propiedades
  - No optimiza el tiempo de espera: muy variable en función del orden de llegada y de la duración de intervalos de CPU
  - Efecto convoy: los trabajos largos retrasan a los cortos
  - No adecuado para sistemas interactivos
  - Muy fácil de implementar (cola FIFO)

Proceso	Instante de llegada	Tiempo de CPU
P1	0	24
P2	0	3
P3	0	3

Consideremos los procesos P1, P2 y P3 cuyo comportamiento se muestra en la tabla adjunta

- Caso 1: orden de llegada P1, P2, P3. Tiempo medio de espera (0 + 24 + 27)/3 = 17
- Caso 2: orden de llegada P2, P3, P1. Tiempo medio de espera (6 + 0 + 3)/3 = 3

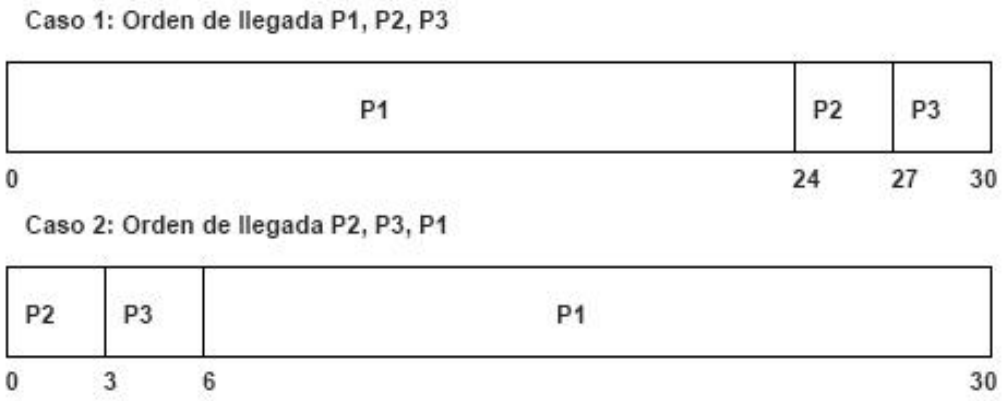


Figura 3: planificador FCFS

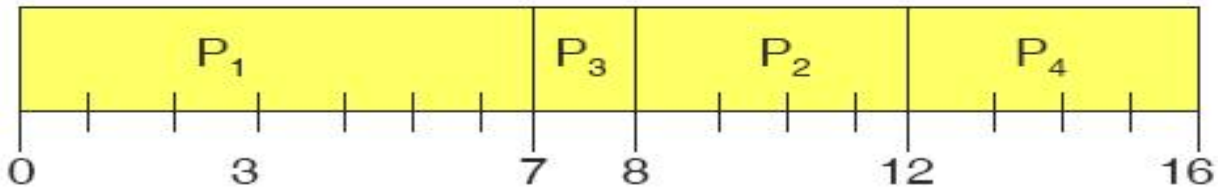
Algoritmo SJF ("Shortest Job First")

- Este algoritmo da prioridad al proceso que va a necesitar menos tiempo de CPU (mejora el tiempo medio de espera)
- Funcionamiento:
  - Asocia a cada proceso un tiempo aproximado de utilización de CPU
  - Asigna la CPU al proceso con menor tiempo asociado
  - Cuando un proceso consigue la CPU la conserva hasta que decide liberarla (no existe expulsión)
- Inconvenientes
  - Estimación del tiempo de utilización de CPU por parte de un proceso (a veces se modela con técnicas estadísticas)

Algoritmo SJF ("Shortest Job First")
Ejemplo

Procesos	Llegada	Tiempo CPU (ms)
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (*no expulsivo*)



- Tiempo de espera medio = (0 + 6 + 3 + 7)/4 = 4

Figura 4: planificador SJF $t_m = [0 + (7 - 4) + (8 - 2) + (12 - 5)]/4$

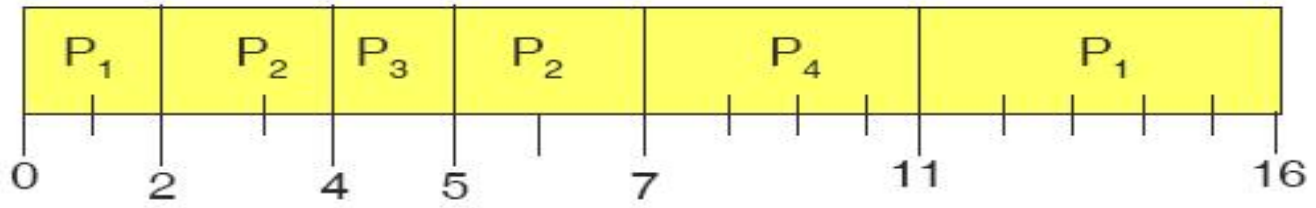
Algoritmo SRTF ("Shortest Remaining Time First")

- Da prioridad al proceso que le resta menos tiempo de CPU para terminar (variante del SJF con expulsión)
- Optimiza la media del tiempo de espera
- Funcionamiento:
  - Los procesos llegan a la cola y solicitan un intervalo de CPU
  - Si dicho intervalo es inferior al que le falta al proceso en ejecución para abandonar la CPU, el nuevo proceso pasa a la CPU y el que se ejecutaba a la cola de preparados.
- Inconvenientes:
  - El intervalo de CPU es difícil de predecir
  - Posibilidad de **inanición**:los trabajos largos no se ejecutarán mientras hayan trabajos cortos

Algoritmo SRTF ("Shortest Remaining Time First") Ejemplo

Procesos	Llegada	Tiempo CPU (ms)
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SRTF (*expulsivo*)



- Tiempo de espera medio = (9 + 1 + 0 +2)/4 = 3

Figura 5: planificador SRTF $t_m = [(11-2)+(5-4)+(4-4)+(7-5)]/4$

Planificación por prioridades

- Se asocia a cada proceso un número entero llamado prioridad de acuerdo con algún criterio.
- Se asigna la CPU al proceso con mayor prioridad
- Funcionamiento:
  - Algoritmos con **expulsión o sin expulsión**
  - Prioridades estáticas o dinámicas
    - Estáticas: se asigna antes de la ejecución y no cambia
    - Dinámicas: cambia con el tiempo
- Propiedades:
  - Con prioridades estáticas aparece el problema de inanición: los procesos con baja prioridad no se ejecutan nunca (poco equitativo)
  - El problema anterior se soluciona con la actualización de

prioridades (dinámicas): la prioridad de un proceso aumenta con el tiempo de espera

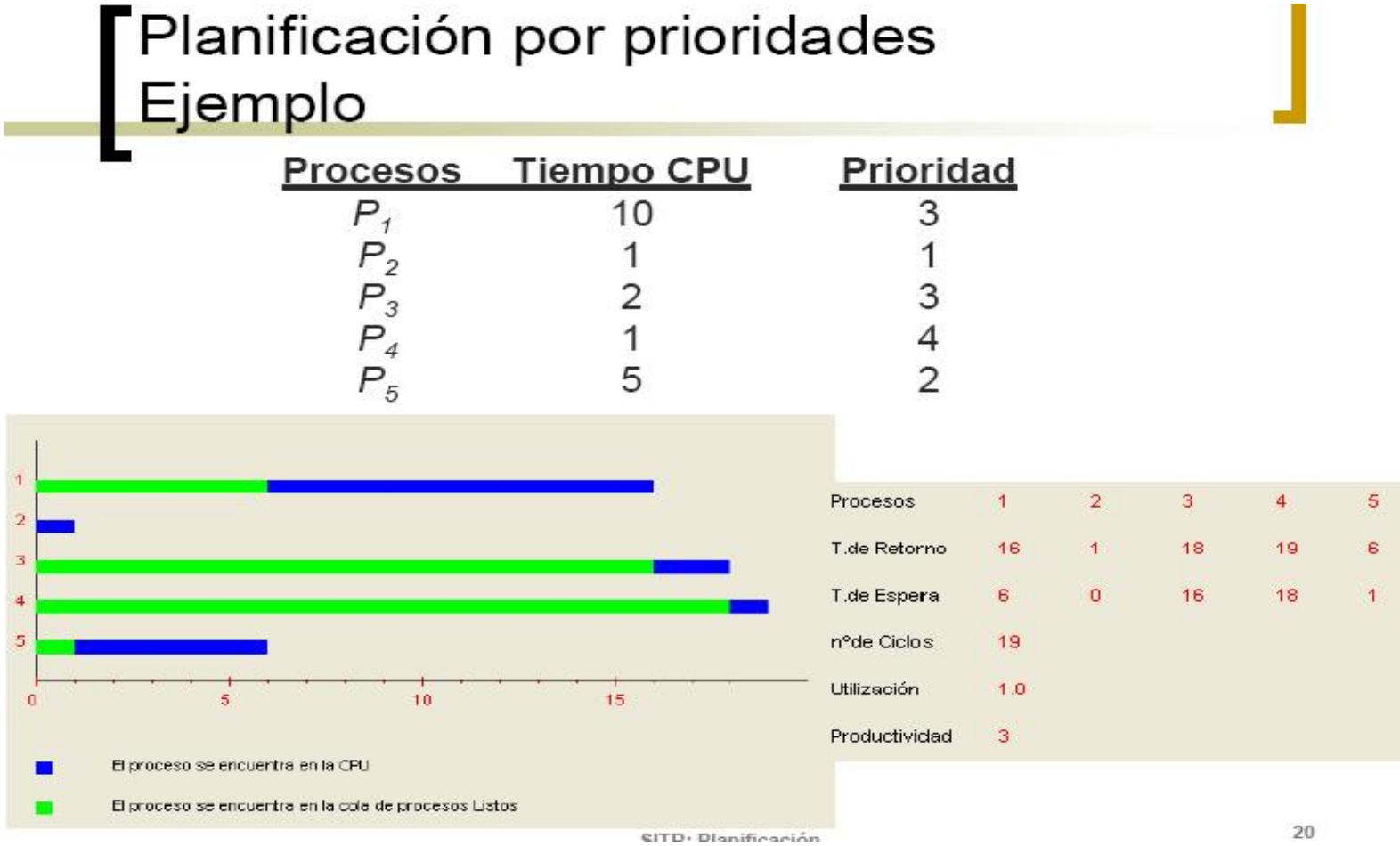


Figura 6: prioridad

Turno Rotatorio o Round Robin (RR)

- Es de los más utilizados, sencillos y equitativos.
- A cada proceso se le asigna un intervalo de tiempo llamado cuanto o **quantum**.(de 10 a 100ms)
- Un proceso se ejecuta durante ese cuanto de tiempo. Si cuando acaba el cuanto no ha terminado su ejecución, se le expulsa de la CPU dando paso a otro proceso.
- Si un proceso termina antes del cuanto, se planifica un nuevo proceso

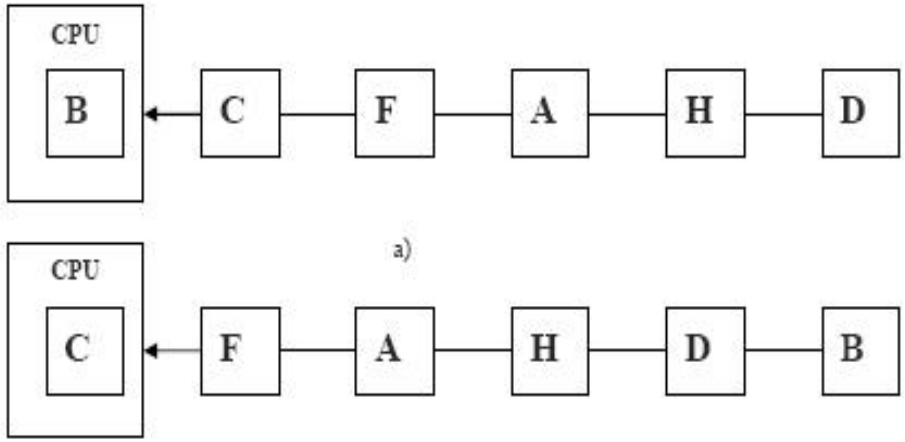


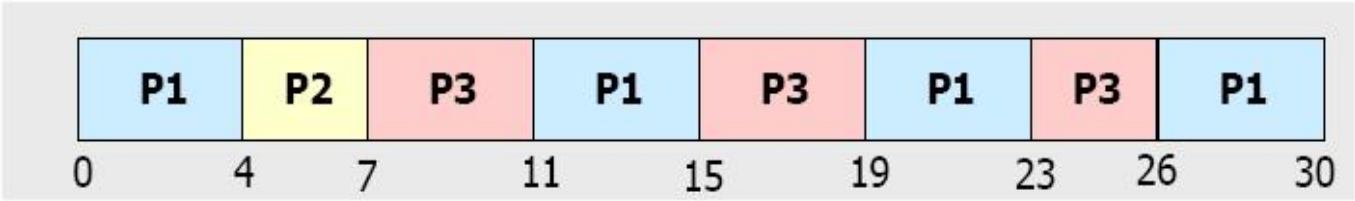
Figura 7: planificado RR

Turno Rotatorio o Round Robin (II)

Quantum q=4

Procesos	T. Llegada	Duración
P1	0	16
P2	0	3
P3	0	11

Diagrama de Gantt



Cronograma por procesos

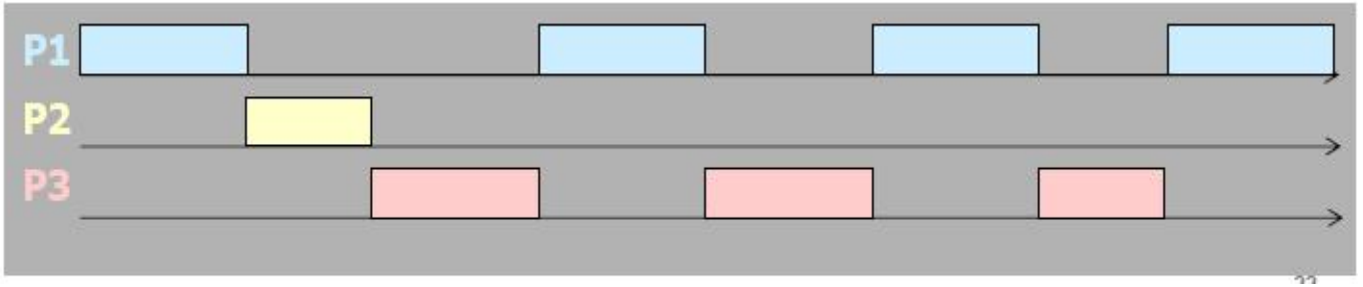


Figura 8: planificador RR2

Turno Rotatorio o Round Robin (RR)

- Valor del "quantum"de tiempo
  - Para **q grandes:** el algoritmo degenera en un algoritmo FCFS.
  - Para **q pequeños:** q ha de ser grande respecto al tiempo necesario para el cambio de contexto, sino la sobrecarga introducida es muy alta.
  - **Regla práctica:** El 80 % de los intervalos de CPU han de ser inferiores al "quantum- de tiempo.
- Si hay **n** procesos en la cola de listos y el quantum es **q**, cada proceso recibe **1/n** del tiempo de CPU. Ningún proceso espera más de (n-1)q unidades de tiempo.
- Funcionamiento:
  - Propiedades

- Equitativo
- Fácil de implementar
- Normalmente el tiempo de retorno medio es mayor que en **SJF**, pero el tiempo de respuesta es mejor

Problema ejemplo

- Porque cuando quemo un disco no puedo hacer nada mas en mi maquina?
  - Si otro proceso me quita el procesador, no puedo terminar "a tiempo"
- DOS soluciones
  - Mas memoria en hardware (controlador de CD) para evitar "buffer underrun".
  - Planificacion "adecuada" que permita mantener suficientes datos en buffer de escritura.

Mas memoria en Hardware

- Cuanta memoria?
- Cuanto cuesta?

- Funciona en todos los casos? NO!

Planificacion adecuada

- Que es?
  - Planificacion predecible que permite analizar si es posible terminar tareas a tiempo
    - Otro proceso no me puede quitar el procesador a menos que se garantice que podré terminar a tiempo
- Cuanto cuesta? 0
- Funciona en todos los casos? SI

Que es planificación adecuada?

- Planificar de acuerdo a tiempos requeridos de terminación y frecuencia de ejecución
- Verificar si todas las tareas (o procesos) pueden cumplir sus tiempos de terminación

Que necesito saber de mis tareas?

- Con que frecuencia corre? **(T ó P)**
  - Ejemplos:
    - Video de 30 frames por segundo:
      - Se ejecuta cada 1/30 segundo para mostrar un frame (PERIODO)
    - Sonido de alta fidelidad 50 Khz
      - Ejecuta cada 1/100000 = 100 micro segundos para reproducir una "muestra"de sonido
- Cual es el tiempo requerido de termino? **(D↔ deadline)**
  - Cada vez que corre
    - Antes de que se ejecute una segunda vez = periodo
- Cual es el tiempo máximo de ejecución de mi tarea? **(C ↔ costo de la ejecución)**
  - Cada vez que corre

- Ejemplo:
  - Video: 10 milisegundos de tiempo de CPU

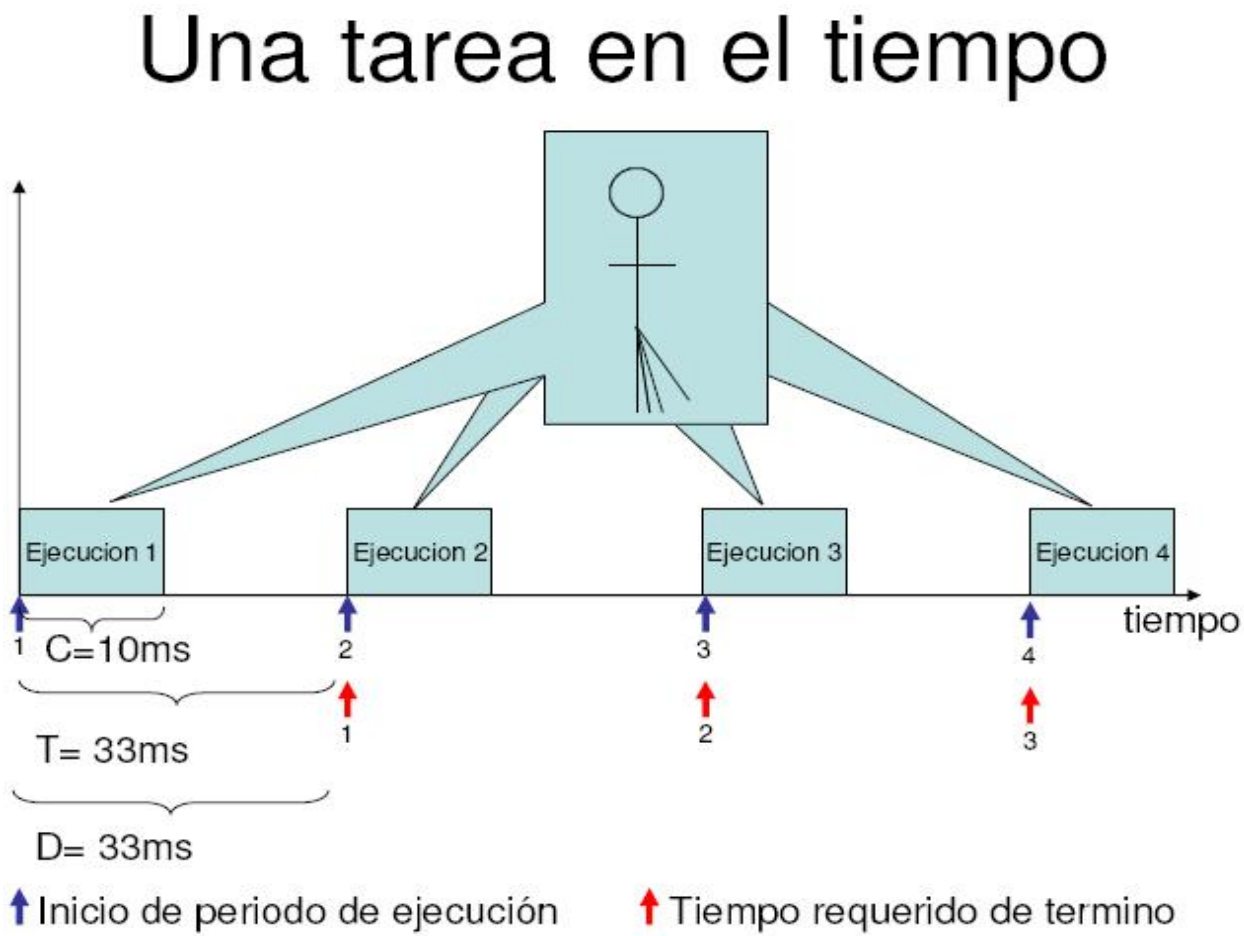

Una tarea en el tiempo
Ejecucion 1
Ejecucion 2
Ejecucion 3
Ejecucion 4
tiempo
1
C=10ms
2
3
4
1
2
3
T= 33ms
D= 33ms
Inicio de periodo de ejecución
Tiempo requerido de termino

Figura 9: t1

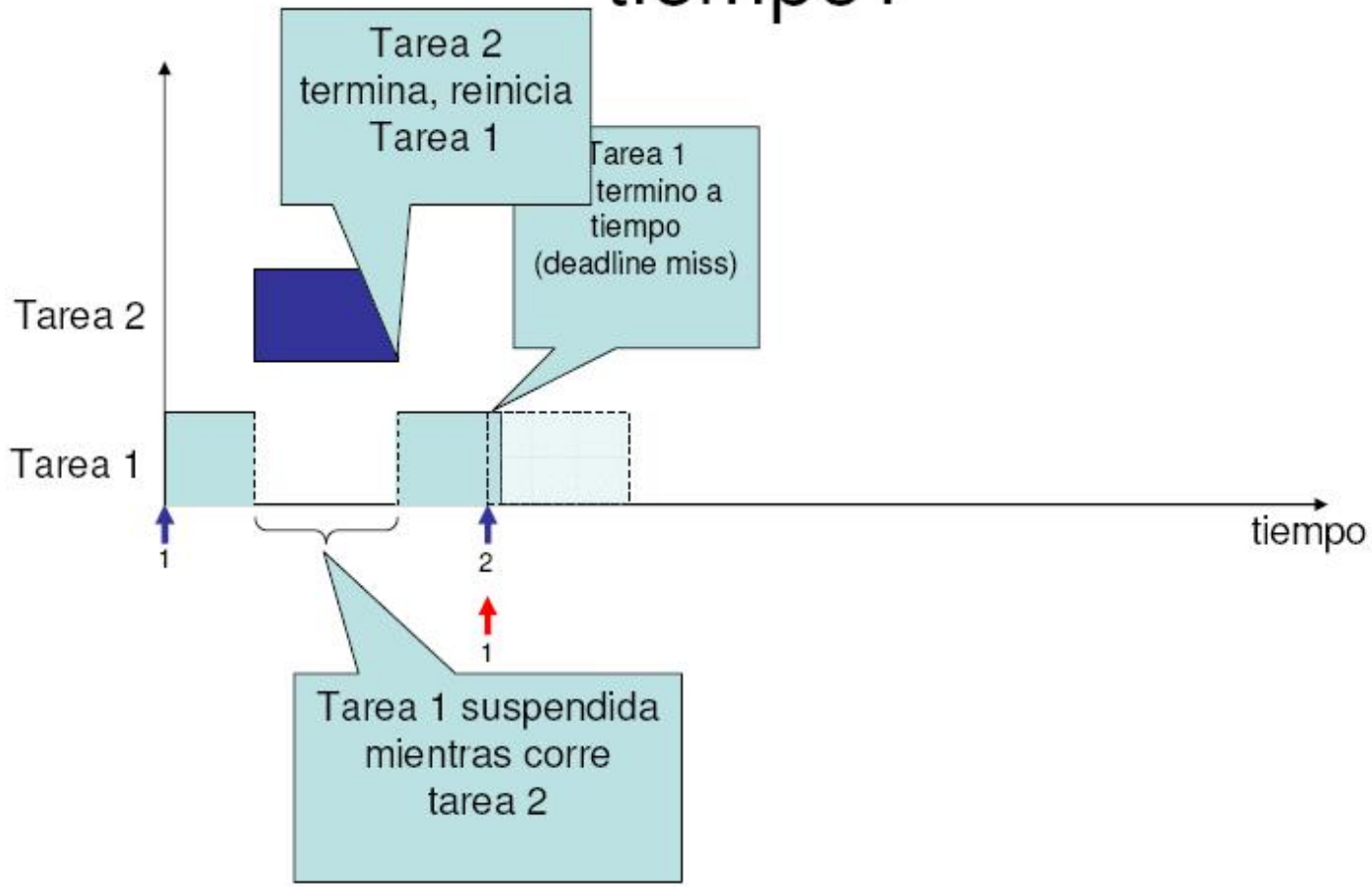

Porque puede NO terminar a tiempo?
Tarea 2 termina, reinicia Tarea 1
Tarea 1 termino a tiempo (deadline miss)
Tarea 2
Tarea 1
tiempo
1
2
1
Tarea 1 suspendida mientras corre tarea 2

Figura 10: t2

Que hacer para que terminen a tiempo?

- Poner a correr la tarea con el tiempo de termino (deadline) mas corto
  - EDF (Earliest Deadline First)
  - Cada vez que una tarea se pone lista para correr:
    - Si esta tarea tiene deadline mas corto que tarea al procesador
      - Suspender tarea actual
      - Poner a correr nueva tarea
  - Desventaja
    - Comparaciones de tiempo de termino de tareas demasiado frecuentes

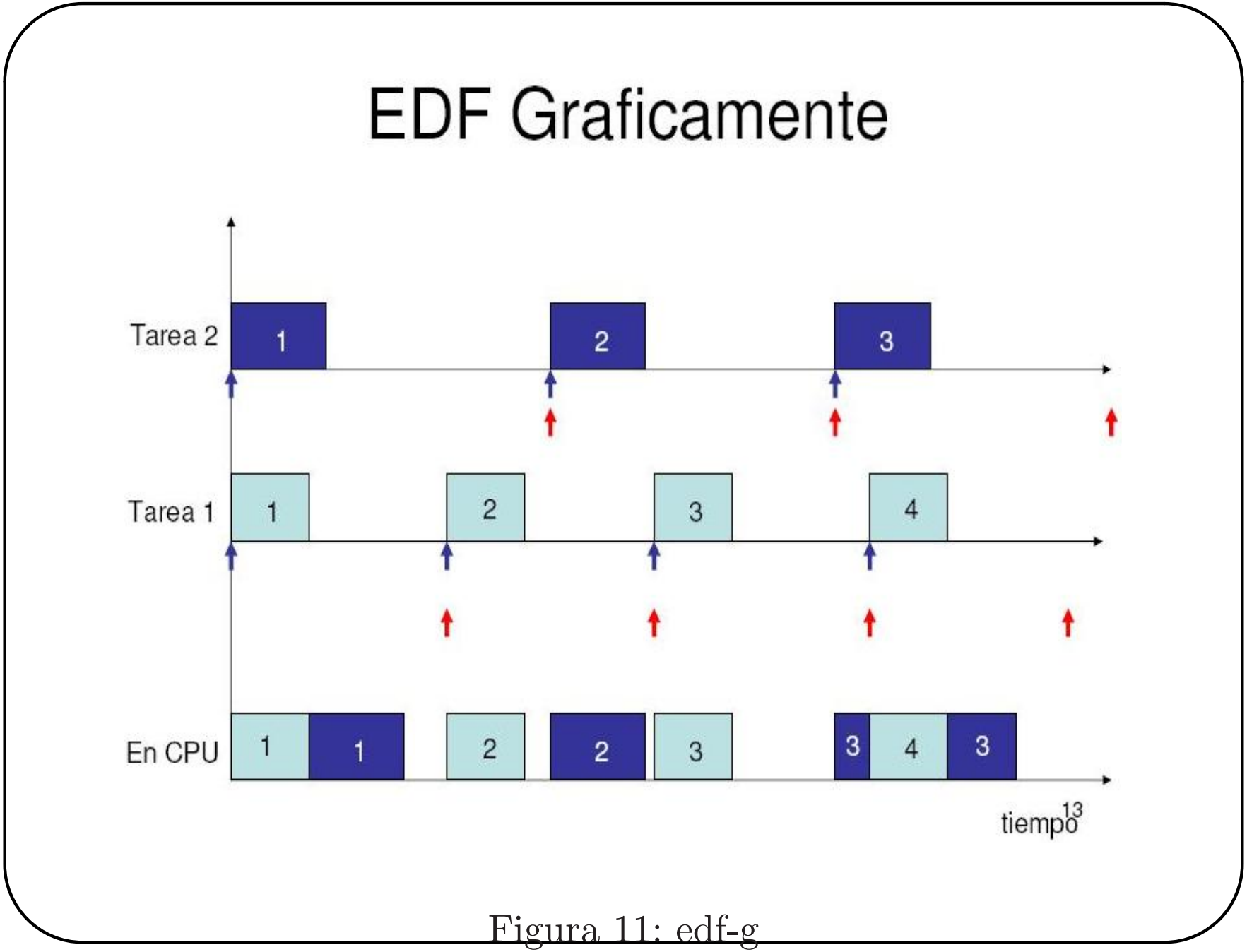


Figura 11: edf-g

Como verifico que mis tareas pueden terminar a tiempo?

EDF

$$\sum_{\forall i \in Tareas} \frac{C_i}{T_i} \leq 1$$

Ejemplo

Tarea 1

$$C_1 = 10,\ T_1 = 100,\ D_1 = 100$$

Tarea 2

$$C_2 = 2,\ T_2 = 10,\ D_2 = 10$$

$$\frac{10}{100} + \frac{2}{10} + \frac{30}{100} \leq 1$$

Como reduzco numero de comparaciones de tiempo de terminación?

- Planificación de prioridad fija (fixed priority)
  - Asignar prioridad al iniciar la tarea
  - Una vez dado el procesador a tarea, no lo devuelve hasta que termina su ejecución actual (y espera el siguiente periodo)
  - Solo compara tiempos una vez antes de ejecutar el conjunto de tareas
  - Cada vez que una tarea se pone lista para ejecutarse solo compara prioridades
    - Comparación más simple de realizar
    - Necesita menos bits

Asignación de prioridades para tareas de tiempo real

- Rate-Monotonic
  - Asignar prioridad mas alta a tareas con periodo más pequeño
  - Ejemplo:
    - Tarea 1

$$C_1 = 10,\ T_1 = 100,\ D_1 = 100$$

    - Tarea 2

$$C_2 = 2,\ T_2 = 10,\ D_2 = 10$$

    - Prioridades
      - Tarea 1: prioridad 2
      - Tarea 2: prioridad 1 (1 mas alta prioridad que 2)

Verificación de tiempo de termino en Rate-Monotonic

Por limite

$$\sum_{\forall i \in Tareas} \frac{C_i}{T_i} \leq 0.69$$

- El limite es menor del 100 % (1) por pequeñas ineficiencias de esta planificacion
  - Esto es: algunas veces la ejecución de una tarea con deadline menor que otra tarea tendra una prioridad mas baja

Tareas periódicas:

- **n** número de tareas del conjunto de tareas
- C_i *or* (e_i) Tiempo de cómputo en el peor de los casos. (Para obtener este valor se ha analizado el codigo de la tarea).
- P_i Periodo de repeticion. Cada T_i unidades de tiempo se ha de activar la tarea.
- D_i Plazo maximo de finalizacion (deadline). Tiempo maximo que puede transcurrir entre la activación de la tarea y la finalización u obtencion de los resultados.
- r_i Tiempo de liberación de la tarea T_i

Caracteristicas de las políticas de planificación

Los objetivos que se persiguen toda política de planificación de tiempo real son:

- Garantizar la correcta ejecución de todas las tareas críticas
- Ofrecer un buen tiempo de respuesta a las tareas periódicas sin plazo.
- Administrar los recursos compartidos.
- Posibilidad de recuperación ante fallos software o hardware.
- Soportar cambios de modo, esto es, cambiar en tiempo de ejecución el conjunto de tareas.
- Las tareas son independientes.
- No comparten recursos.
- Todas las tareas son periódicas.

- Los tiempos de cambio de contexto son despreciables.

Definición previas

Un **planicador** es un método para asignar recurso (el tiempo de procesador).

Diremos que un conjunto de tareas es factible o planificable si existe algún planificador que sea capaz de cumplir las restricción de todas las tareas (en nuestro caso las retricciones temporales: los plazos de ejecución de tareas factible.

El **factor de utilización** es

$$U = \sum_{i=1}^{k} \frac{C_i}{P_i}$$

El **hiperperiodo** es el mínimo común múltiplo de los periodos de las tareas

Test de garantía (factor de utilización)

Un conjunto de n tareas será planificable bajo el ***Rate-Monotonic*** si se cumple que el factor de utilización del conjunto de las tareas es menor que la expresión $n(2^{\frac{1}{n}} - 1)$ esto es:

$$\frac{C_1}{P_1} + \cdots + \frac{C_n}{P_n} = U \leq U(n) = n(2^{\frac{1}{n}} - 1)$$

n	$U(n)$
1	1.0
2	0.828
3	0.779
4	0.756
5	0.743
...	...
∞	0.693

Cuadro 1: Valores $U(n)$

Si aplicamos este test al siguiente conjunto de tareas:

$T1 = (1, 5)$; $T2 = (2, 8)$; $T3 = (3, 14)$

Tenemos lo siguiente

$$\frac{1}{5} + \frac{2}{8} + \frac{3}{14} = 0.2 + 0.25 + 0.214 = 0.664 \leq U(3) = 3(2^{\frac{1}{3}} - 1) = 0.779$$

Con lo que podemos asegurar que este conjunto de tareas es planificable por RateMonotonic.

Test de garantía (tiempos de finalización)

Un conjunto de $\ll n \gg$ tareas será planificable bajo cualquier asignación de prioridades si y sólo si: Cada tarea cumple su plazo de ejecución en el peor caso.

Siendo el peor caso de cada tarea aquel en el que todas las tareas de prioridad superior se activa a la vez que ésta.

si lo expresamos matemáticamente el test queda formulado como sigue:

$$\forall \quad 1 \leq i \leq n, \qquad W_i \leq D_i$$

Donde:
W_i respresenta el instante en que finaliza la ejecución en el peor caso. Este valor se obtiene de la siguiente expresión.

$$W_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_i}{P_j} \right\rceil C_j$$

Donde:
$hp(i)$ representa el conjunto de tareas con prioridad mayor que la tarea i. Tal como vemos a ambos lados de la igualdad tenemos el valor W_i, que no se puede despejar. La solución de esta expresión se obtiene de forma iterativa:

$$
\begin{aligned}
W_i^0 &= C_i \\
W_i^1 &= C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_i^0}{P_j} \right\rceil C_j \\
W_i^2 &= C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_i^1}{P_j} \right\rceil C_j \\
&\cdots \\
W_i^k &= W_i^{k+1}
\end{aligned}
$$

La iteración se repite hasta encontrar dos valores consecutivos (W_i^k y W_i^{k+1}) iguales. El último valor W obtenido es el que luego emplearemos en la expresión de test de garantia. Si en alguna iteración se obtiene un valor de W mayor que el plazo máximo de ejecución de la tarea, entonces esta tarea no será planificable y por lo tanto el sistema tampoco.

Ejemplo de aplicación de test

Tenemos:

$$T1 = (C = 1, P = 4); T2 = (C = 2, P = 9); T3 = (C = 4, P = 10)$$

Primera tarea

$$\begin{aligned} W_1^0 &= C_1 &&= 1 \\ W_1^1 &= C_1 + 0 &&= 1 \leq D_1 = 4 \end{aligned}$$

Segunda tarea

$$
\begin{aligned}
W_2^0 &= C_2 &&= 2 \\
W_2^1 &= C_2 + \textstyle\sum_{\forall j \in 1} \left\lceil \frac{W_2^0}{P_j} \right\rceil C_j &&= 2 + \left\lceil \frac{2}{4} \right\rceil = 3 \\
W_2^2 &= C_2 + \textstyle\sum_{\forall j \in 1} \left\lceil \frac{W_2^1}{P_j} \right\rceil C_j &&= 2 + \left\lceil \frac{3}{4} \right\rceil = 3 \\
W_2^2 &= W_2^1 &&= 3 \leq D_2 = 9
\end{aligned}
$$

Tercera tarea

$$
\begin{aligned}
W_3^0 &= C_3 &&= 4 \\
W_3^1 &= C_3 + \textstyle\sum_{\forall j \in \{1,2\}} \left\lceil \frac{W_3^0}{P_j} \right\rceil C_j &&= 4 + \left\lceil \frac{4}{4} \right\rceil 1 + \left\lceil \frac{4}{9} \right\rceil 2 = 7 \\
W_3^2 &= C_3 + \textstyle\sum_{\forall j \in \{1,2\}} \left\lceil \frac{W_3^1}{P_j} \right\rceil C_j &&= 4 + \left\lceil \frac{7}{4} \right\rceil 1 + \left\lceil \frac{7}{9} \right\rceil 2 = 8 \\
W_3^3 &= C_3 + \textstyle\sum_{\forall j \in \{1,2\}} \left\lceil \frac{W_3^2}{P_j} \right\rceil C_j &&= 4 + \left\lceil \frac{8}{4} \right\rceil 1 + \left\lceil \frac{8}{9} \right\rceil 2 = 8 \\
W_3^3 &= W_3^2 &&= 8 \leq D_3 = 10
\end{aligned}
$$

Si utilizaramos el primer test propuesto (**factor de utilización**) veremos que **NO cumple el test**, y por lo tanto no sabriamos si el conjunto de tareas es planificable o no. **El primero es un test necesario pero no suficiente**. Mientras que este nuevo test es necesario pero suficiente.

Schedulability test for RM

$$t_i \qquad W_i(t) \leq t_i \qquad \forall i$$

Where

$$W_i(t) = \sum_{j=1}^{i} c_j \left\lceil \frac{t}{P_j} \right\rceil$$

and

$$t_i = \left\{ \ell P_i | j = 1, \ldots, i; \ell = 1, \ldots, \left\lfloor \frac{P_i}{P_j} \right\rfloor \right\}$$

Example Rate Monotonic Scheduling

checking for schedulability using RM

Taks	c_i	P_i	r_i	
t_1	0.5	2	0	r_i release time
t_2	2.0	6	1	c_i execution time
t_3	1.75	10	3	P_i period

1. The periods are in the range $P_1 < P_2 < P_3$
   the prority orden is $t_1 > t_2 > t_3$
2. schedule graph
3. schedulability test for RM
   $a)$ The necessary and

$$t_i \qquad W_i(t) \leq t_i \qquad \forall i$$

$$W_i(t) = \sum_{j=1}^{i} c_j \left\lceil \frac{t}{P_j} \right\rceil$$

$$t_i = \left\{ \ell P_i | j = 1, \ldots, i; \ell = 1, \ldots, \left\lfloor \frac{P_i}{P_j} \right\rfloor \right\}$$

$b)$ t_i calculation i in the number y task, so i will be 1,2,3

$$t_1 = \{\ell P_1\} = \left\{ \left\lfloor \frac{P_1}{P_1} \right\rfloor P_1 \right\} = \{2\}$$

$$t_2 = \{\ell P_1, \ell P_2\} = \left\lfloor \frac{P_2}{P_1}, \frac{P_2}{P_2} \right\rfloor = \left\lfloor \frac{6}{2}, \frac{2}{2} \right\rfloor = \{3, 1\}$$

$$t_2 = \{3P_1, P_1, 3P_2, P_2\} = \{2, 6, 6, 18\}$$

remove the duplicate and remove periods above the maximum

$$t_2 = \{2, 6\}$$

$$t_3 = \{\ell P_1, \ell P_2, \ell P_3\} = \left\lfloor \frac{P_3}{P_1}, \frac{P_3}{P_2}, \frac{P_3}{P_3} \right\rfloor = \left\lfloor \frac{10}{2}, \frac{10}{6}, \frac{10}{10} \right\rfloor = \{5, 1, 1\}$$

$$t_3 = \{5P_1, 5P_2, 5P_3, P_1, P_2, P_3\}$$

$$t_3 = \{10, 30, 50, 2, 6, 10\} = \{2, 6, 10\}$$

$c)$ $W_1(t)$

$$W_1(1) = \sum c_j \left\lceil \frac{t}{P_j} \right\rceil \leq t_1 = 2$$

$$W_1(2) = 0.5 \left\lceil \frac{2}{2} \right\rceil = 0.5 \leq 2$$

t_1 is schedulable

$d)$ W_2

$$W_2(2) = c_1 \left\lceil \frac{2}{P_1} \right\rceil + c_2 \left\lceil \frac{2}{P_2} \right\rceil \leq 2$$

$$= 0.5 \left\lceil \frac{2}{2} \right\rceil + 2 \left\lceil \frac{2}{6} \right\rceil \leq 2$$

$$= 2.5 \leq 2 \bigstar$$

$$W_2(6) = c_1 \left\lceil \frac{6}{P_1} \right\rceil + c_2 \left\lceil \frac{6}{P_2} \right\rceil \leq 6$$

$$= 0.5 \left\lceil \frac{6}{2} \right\rceil + 2 \left\lceil \frac{6}{6} \right\rceil \leq 6$$

$$3.5 \leq 6 \checkmark$$

if any one condition true, then the task in schedulable (t_2)

$e)$ W_3

$$W_3(2) = c_1 \left\lceil \frac{2}{P_1} \right\rceil + c_2 \left\lceil \frac{2}{P_2} \right\rceil + c_3 \left\lceil \frac{2}{P_3} \right\rceil \leq 2$$

$$= 0.5 \left\lceil \frac{2}{2} \right\rceil + 2 \left\lceil \frac{2}{6} \right\rceil + 1.75 \left\lceil \frac{2}{10} \right\rceil \leq 2$$

$$= 4.25 \leq 2 \bigstar$$

$$W_3(6) = c_1 \left\lceil \frac{6}{P_1} \right\rceil + c_2 \left\lceil \frac{6}{P_2} \right\rceil + c_3 \left\lceil \frac{6}{P_3} \right\rceil \leq 6$$

$$= 1.5 + 2 + 1.75 \leq 6$$

$$= 5.25 \leq 6\checkmark$$

t_3 is schedulable

$$W_3(10) = 2.5 + 4 + 1.75 \leq 10$$

$$= 8.25 \leq 10\checkmark$$

the results can alse be obained using time donand función (TDF) which is a graph Hotled against T is $W(t)$

t	W_1	W_2	W_3
2	0.5	2.5	4.25
6		3.5	5.25
10			8.25

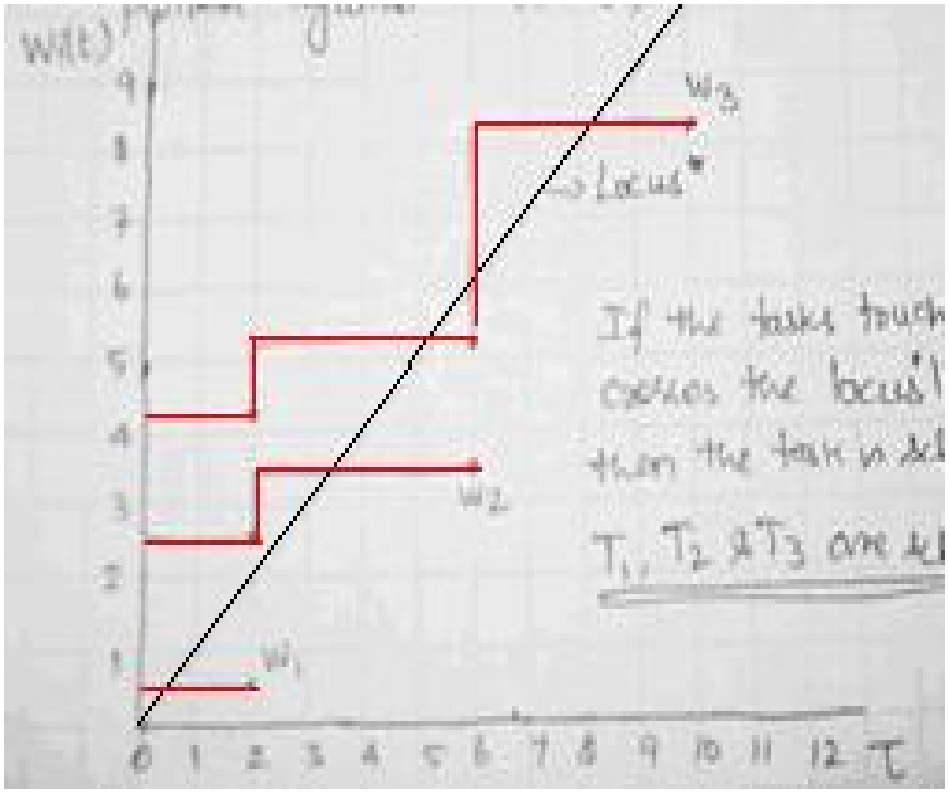


Figura 12: Gráfica hotled

Example: consider the set of four tasks where

i	c_i	P_i
1	20	100
2	30	150
3	80	210
4	100	400

Then

$\tau_1 = \{100\}$

$\tau_2 = \{100, 150\}$

$\tau_3 = \{100, 150, 200, 210\}$

$\tau_4 = \{100, 150, 200, 210, 300, 400\}$

Let us check the RM-schedulability of each task. Figura **??** contains plots of $W_i(t)$ for $i = 1, 2, 3, 4$. Task T_i is RM-schedulable iff any part of the plot of $W_i(t)$ falls on or below the $W_i(t) = t$ line

In algebraic terms, we have:

- task T_1 is RM-schedulable iff $c_1 \leq 100$
- task T_2 is RM-schedulable iff

$$c_1 + c_2 \leq 100 \quad OR$$
$$2c_1 + c_2 \leq 150$$

- task T_3 is RM-schedulable iff

$$\begin{aligned} c_1 + c_2 + c_3 &\leq 100 \quad OR \\ 2c_1 + c_2 + c_3 &\leq 150 \quad OR \\ 2c_1 + 2c_2 + c_3 &\leq 200 \quad OR \\ 3c_1 + 2c_2 + c_3 &\leq 210 \end{aligned}$$

- task T_4 is RM-schedulable iff

$$\begin{aligned} c_1 + c_2 + c_3 + c_4 &\leq 100 \quad OR \\ 2c_1 + c_2 + c_3 + c_4 &\leq 150 \quad OR \\ 2c_1 + 2c_2 + c_3 + c_4 &\leq 200 \quad OR \\ 3c_1 + 2c_2 + c_3 + c_4 &\leq 210 \quad OR \\ 3c_1 + 2c_2 + 2c_3 + c_4 &\leq 300 \quad OR \\ 4c_1 + 3c_2 + 2c_3 + c_4 &\leq 400 \end{aligned}$$