dit
UPM

Planificación con prioridades

Juan Antonio de la Puente
DIT/UPM

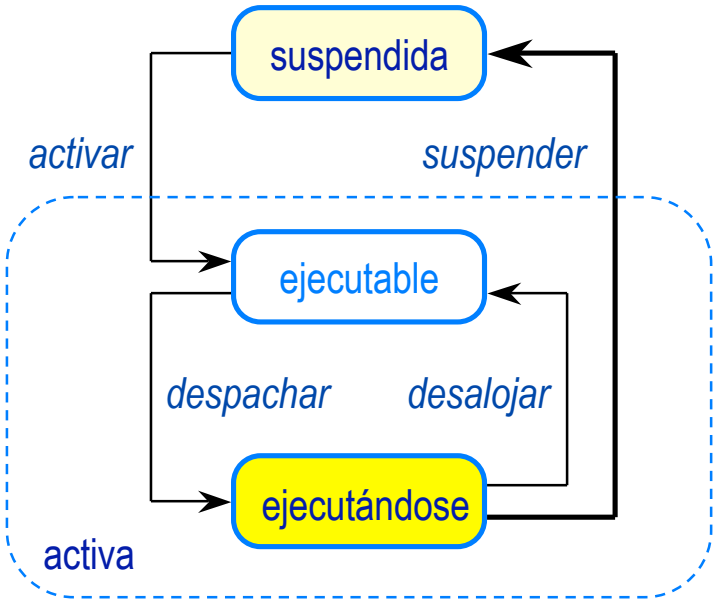
Planificación

- El objetivo de los métodos de **planificación** *(scheduling)* es repartir el tiempo de procesador entre varias tareas de forma que se garanticen los requisitos temporales de todas ellas
  - activación periódica o esporádica
  - plazo de respuesta *(deadline)*

- Para poder analizar el comportamiento del sistema hay que definir un modelo de cómputo adecuado
  - tareas estáticas o dinámicas
  - tareas periódicas, esporádicas, aperiódicas
  - tareas independientes, sincronización, comunicación

Multiprogramación

- Las tareas se realizan como hebras concurrentes
- Una tarea puede estar en varios estados
- Las tareas ejecutables se **despachan** para su ejecución de acuerdo con un *método de planificación:*
  - prioridades fijas (*fixed-priority scheduling,* FPS)
  - primero el más urgente (*earliest deadline first,* EDF)
  - primero el más valioso (*value-based scheduling,* VBS)



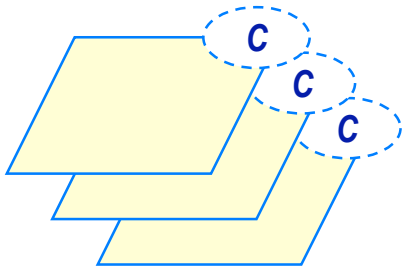
Planificación con prioridades fijas

- Es el método más corriente en sistemas operativos de tiempo real
- Cada tarea tiene una prioridad fija
  - planificación estática
- Las tareas ejecutables se **despachan** para su ejecución en orden de **prioridad**
- El despacho puede hacerse
  - **con desalojo**
  - **sin desalojo**
  - **con desalojo limitado**
- En general supondremos **prioridades fijas con desalojo**
  - **mejor tiempo de respuesta para las tareas de alta prioridad**

Tareas periódicas

Diseño de sistemas

- Cuando se diseña un sistema planificado con prioridades fijas hay dos problemas:
  - cómo asignar prioridades a las tareas
  - cómo analizar el sistema para ver si se garantizan los requisitos temporales
- La solución depende del modelo de tareas
- Empezamos con un modelo sencillo
  - conjunto estático de tareas periódicas e independientes



Parámetros de planificación

N	Número de tareas
T	Período de activación
C	Tiempo de ejecución máximo
D	Plazo de respuesta
R	Tiempo de respuesta máximo
P	Prioridad

De momento supondremos que para todas las tareas τ_i :

$$C_i \leq D_i = T_i$$

Se trata de asegurar que

$$R_i \leq D_i$$

Prioridades monótonas en frecuencia

- La asignación de mayor prioridad a las tareas de menor período *(rate monotonic scheduling)* es **óptima** para un modelo de tareas con
  - tareas periódicas,
  - independientes,
  - con plazos iguales a los períodos

Si se pueden garantizar los plazos de un sistema de tareas con otra asignación de prioridades, se pueden garantizar con la asignación monótona en frecuencia

(Liu & Layland, 1973)

Condición de garantía de los plazos basada en la utilización

- Para este modelo de tareas, con prioridades monótonas en frecuencia, los plazos están garantizados si

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N \cdot \left(2^{1/N} - 1\right)$$

- La cantidad

$$U_0(N) = N \cdot \left(2^{1/N} - 1\right)$$

es la **utilización mínima garantizada** para N tareas

Utilización mínima garantizada

N	U0
1	1,000
2	0,828
3	0,779
4	0,756
5	0,743

$$\lim_{n\to\infty} U_0(N) = \log 2 \approx 0{,}693$$

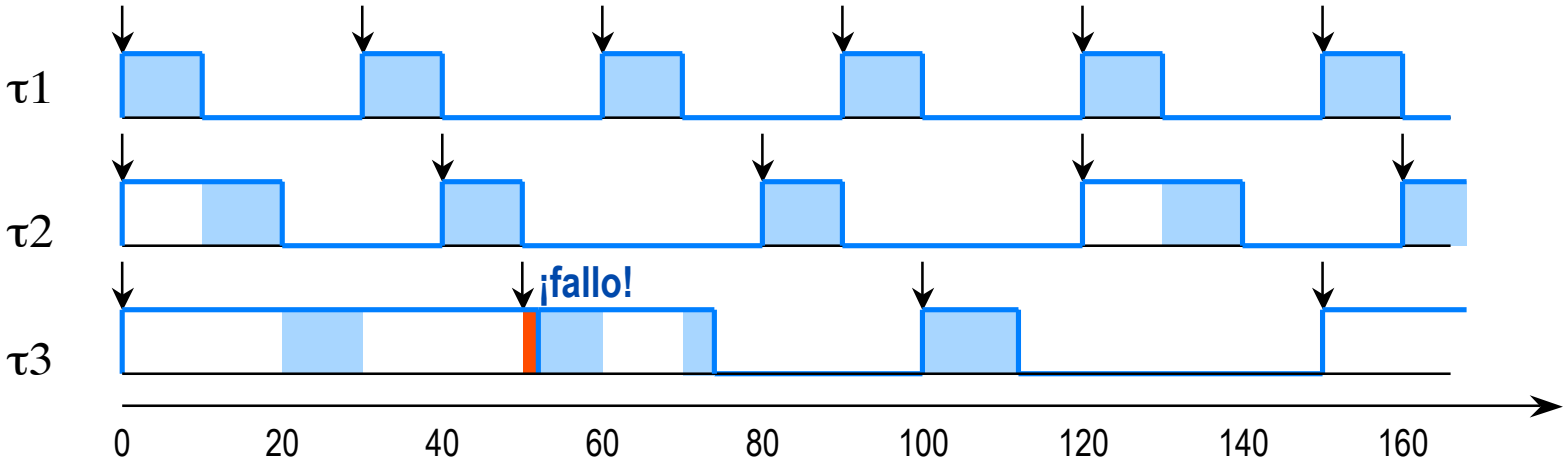
Ejemplo 1

Tarea	T	C	P	U
τ1	30	10	3	0,333
τ2	40	10	2	0,250
τ3	50	12	1	0,240
				0,823

El sistema no cumple la prueba de utilización

(U > 0,779)

La tarea 3 falla en *t* = 50

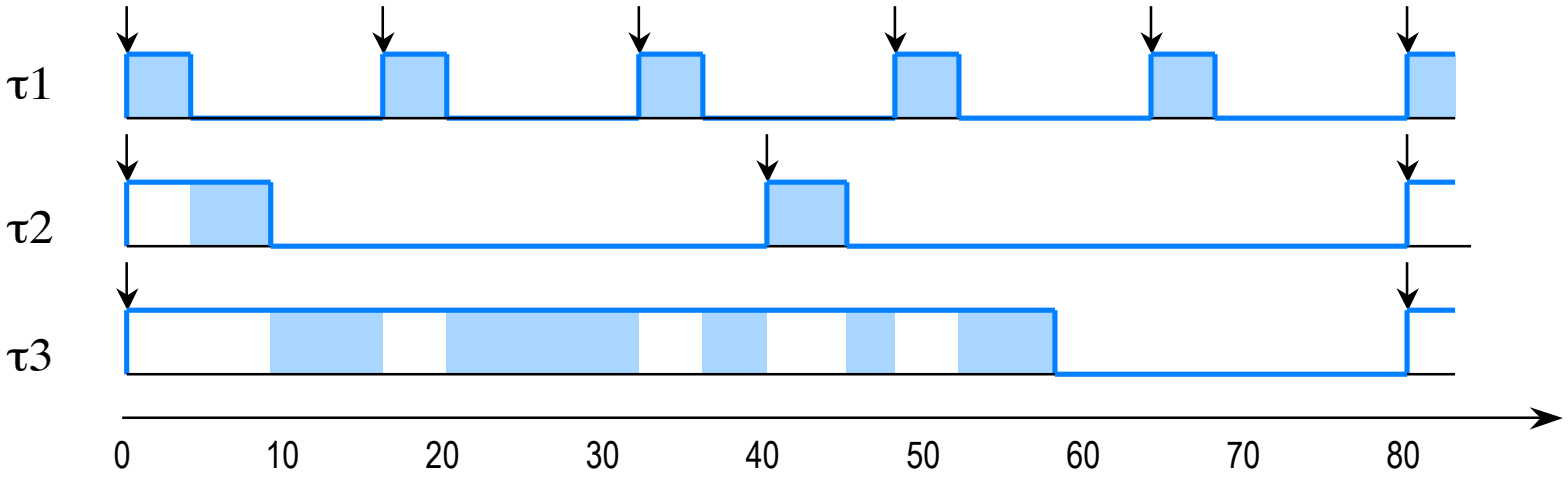


Ejemplo 2

Tarea	T	C	P	U
τ1	**16**	**4**	**3**	0,250
τ2	**40**	**5**	**2**	0,125
τ3	**80**	**32**	**1**	0,400
				0,775

Este sistema está garantizado

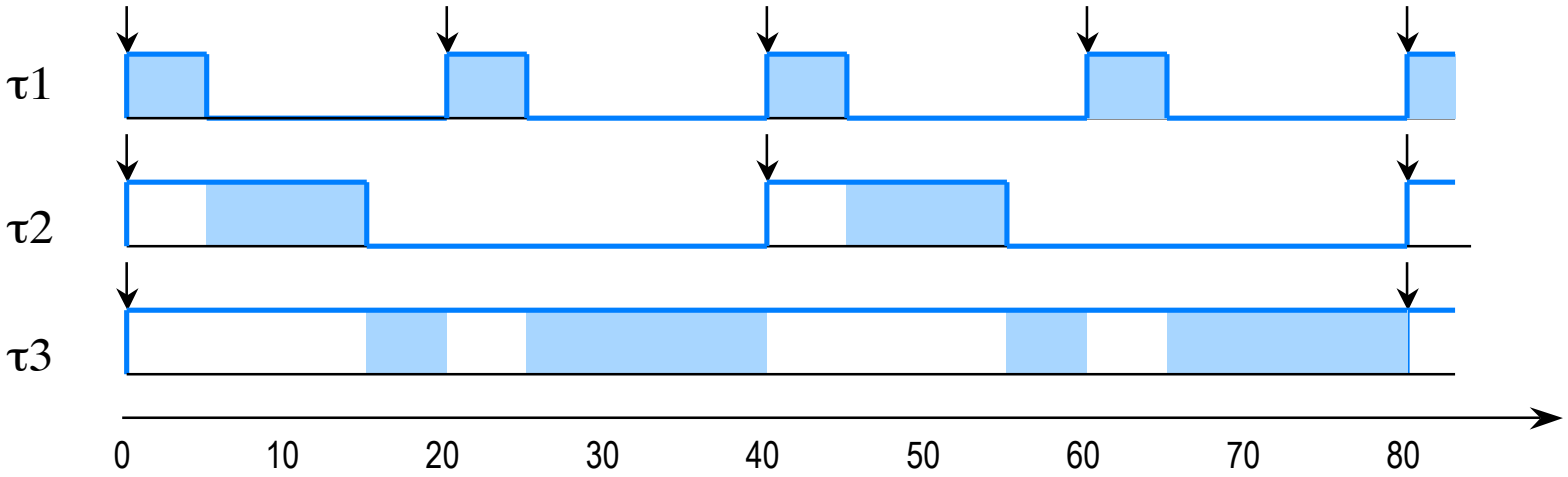
(U < 0,779)



Ejemplo 3

Tarea	T	C	P	U
τ1	20	5	3	0,250
τ2	40	10	2	0,250
τ3	80	40	1	0,500
				1,000

Este sistema no pasa la prueba ($U > 0{,}779$), pero se cumplen los plazos



Problemas del análisis de utilización

- La prueba del factor de utilización no es exacta, ni se puede generalizar a modelos de tareas más complejos
  - pero es eficiente, *O(N)*
- Veremos una prueba basada en el cálculo del tiempo de respuesta de cada tarea

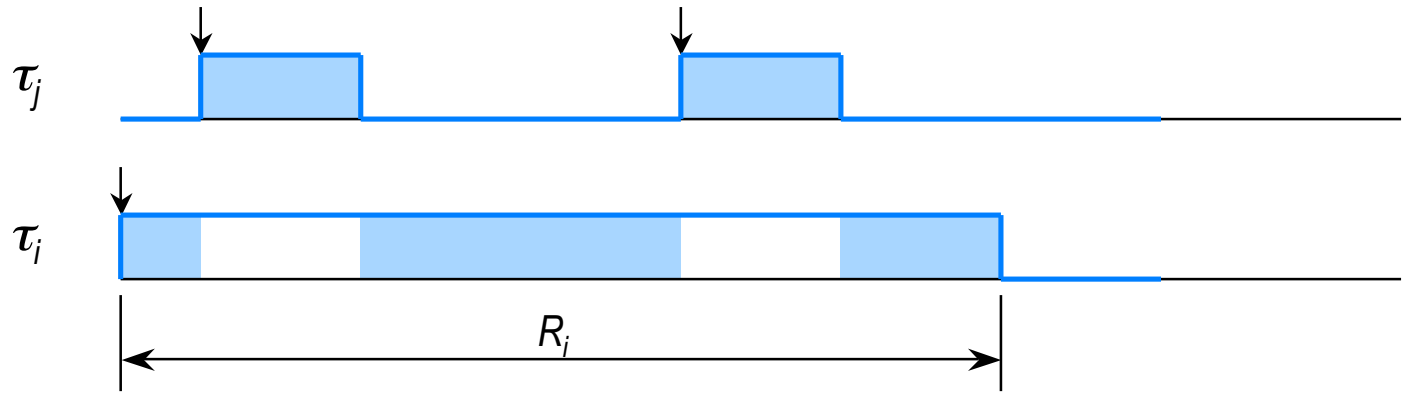
Análisis del tiempo de respuesta

- Es un método más completo y flexible que el del factor de utilización para FPS
  - es fácil de generalizar a otros modelos de tareas
  - proporciona una condición necesaria y suficiente para que los plazos estén garantizados

- Se trata de calcular el tiempo de respuesta en el peor caso de cada tarea, R_i, y comprobar directamente que es menor que el plazo correspondiente:

$$R_i \leq D_i$$

Ecuación del tiempo de respuesta

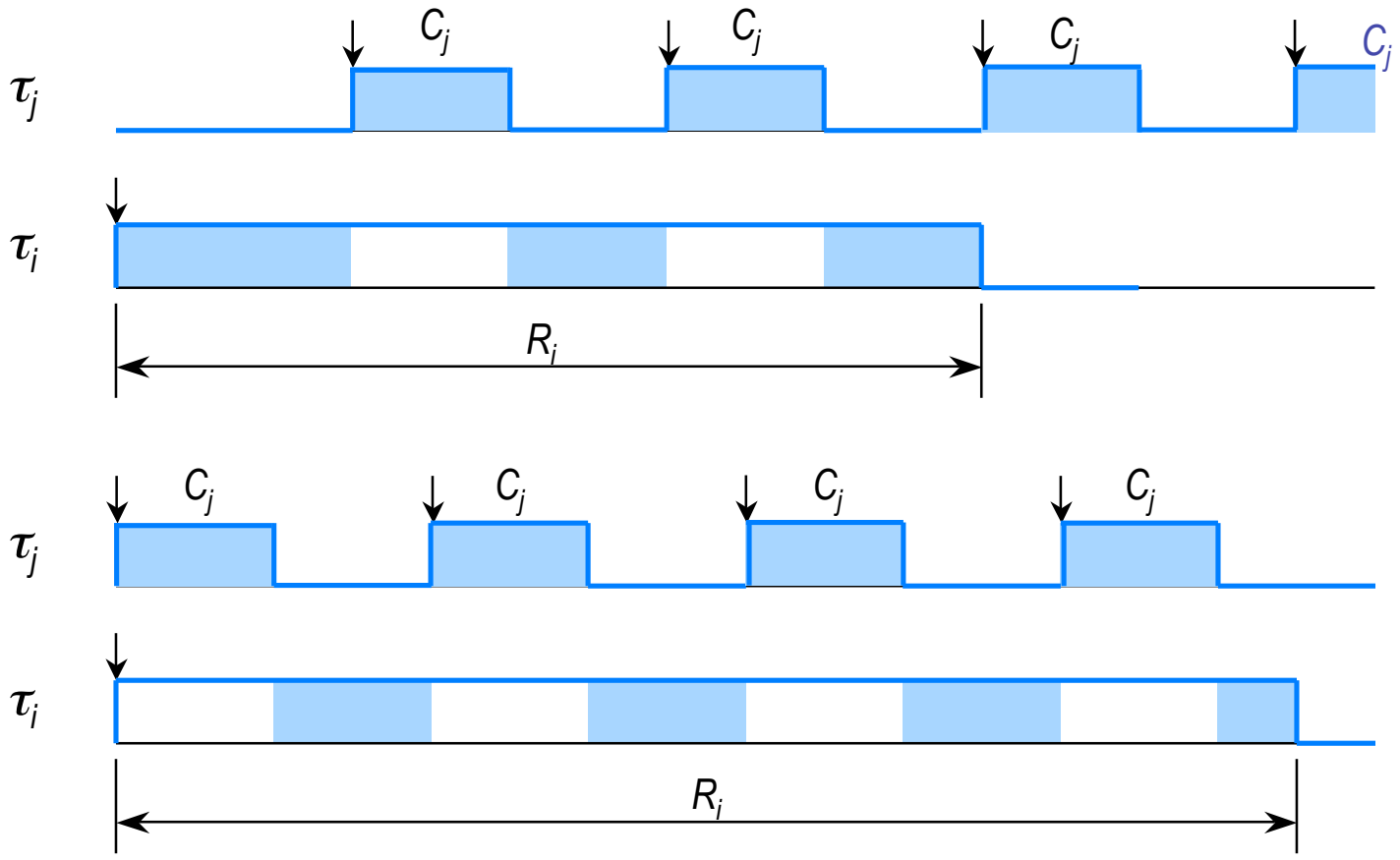


$$R_i = C_i + I_i$$

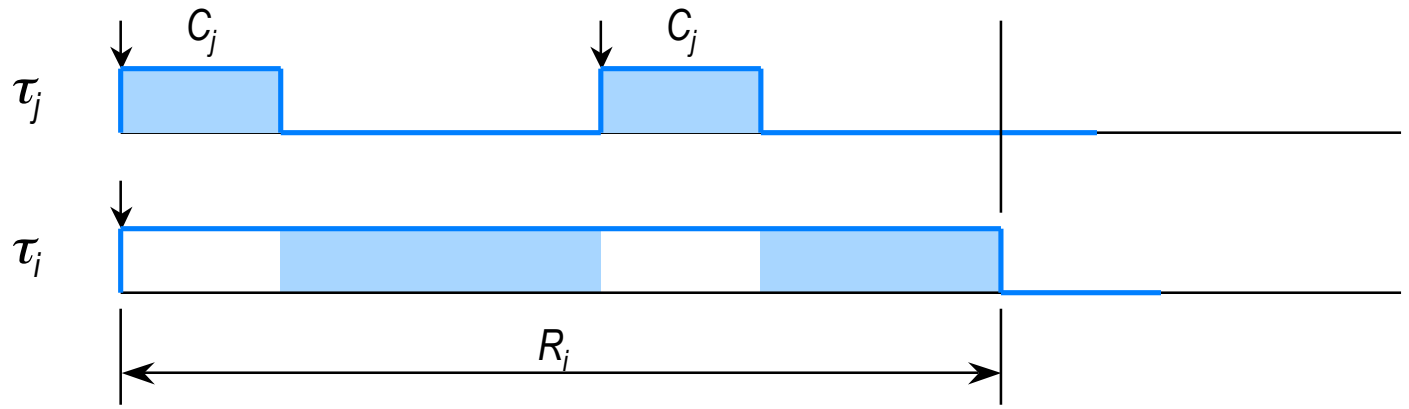
El tiempo de respuesta de una tarea es la suma de su tiempo de cómputo más la interferencia que sufre por la ejecución de tareas más prioritarias

Instante crítico

- La interferencia es máxima cuando todas las tareas se activan a la vez
  - el instante inicial se denomina **instante crítico**



Cálculo de la interferencia



- El número de veces que una tarea de prioridad superior τ_j se ejecuta durante el intervalo $[0,R_i)$ es:

$$\left\lceil \frac{R_i}{T_j} \right\rceil$$

función *techo:* $\lceil x \rceil = \min k \in Z : k \geq x$

Por tanto, el valor de la interferencia de τ_j sobre τ_i es

$$I_i^j = \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j$$

Cálculo del tiempo de respuesta

- La interferencia total que sufre τ_i es

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \qquad hp(i) = \{ j : 1..N \mid P_j > P_i \}$$

- La ecuación del tiempo de respuesta queda así:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j$$

  - La ecuación no es continua ni lineal
  - No se puede resolver analíticamente

Iteración lineal

- La ecuación del tiempo de respuesta se puede resolver mediante la relación de recurrencia

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil \cdot C_j$$

  - la sucesión $\left(w_i^0, w_i^1, \ldots w_i^n, \ldots\right)$ es monótona no decreciente
  - un valor inicial aceptable es $w_i^0 = C_i$
  - se termina cuando
    - a) $w_i^{n+1} = w_i^n$ (y entonces $R_i = w_i^n$), o bien
    - b) $w_i^{n+1} > T_i$ (no se cumple el plazo)
  - converge siempre que $U < 100\%$

Ejemplo 4

Tarea	T	C	P	R
$\tau 1$	7	3	3	3
$\tau 2$	12	3	2	6
$\tau 3$	20	5	1	20

$$R_1 = 3$$

$$w_2^0 = 3$$

$$w_2^1 = 3 + \left\lceil \frac{3}{7} \right\rceil \cdot 3 = 6$$

$$w_2^2 = 3 + \left\lceil \frac{6}{7} \right\rceil \cdot 3 = 6; \quad R_2 = 6$$

$$w_3^0 = 5$$

$$w_3^1 = 5 + \left\lceil \frac{5}{7} \right\rceil \cdot 3 + \left\lceil \frac{5}{12} \right\rceil \cdot 3 = 11$$

$$w_3^2 = 5 + \left\lceil \frac{11}{7} \right\rceil \cdot 3 + \left\lceil \frac{11}{12} \right\rceil \cdot 3 = 14$$

$$w_3^3 = 5 + \left\lceil \frac{14}{7} \right\rceil \cdot 3 + \left\lceil \frac{14}{12} \right\rceil \cdot 3 = 17$$

$$w_3^4 = 5 + \left\lceil \frac{17}{7} \right\rceil \cdot 3 + \left\lceil \frac{17}{12} \right\rceil \cdot 3 = 20$$

$$w_3^5 = 5 + \left\lceil \frac{20}{7} \right\rceil \cdot 3 + \left\lceil \frac{20}{12} \right\rceil \cdot 3 = 20 \quad R_3 = 20$$

Todas las tareas tienen sus plazos garantizados

Ejemplo 3 (repaso)

Tarea	T	C	P	U	R
$\tau 1$	**20**	**5**	**3**	0,250	**5**
$\tau 2$	**40**	**10**	**2**	0,250	**15**
$\tau 3$	**80**	**40**	**1**	0,500	**80**
				1,000	

$$R_1 = 5$$

$$w_2^0 = 10$$

$$w_2^1 = 10 + \left\lceil \frac{10}{20} \right\rceil \cdot 5 = 15$$

$$w_2^2 = 10 + \left\lceil \frac{15}{20} \right\rceil \cdot 5 = 15; \quad R_2 = 15$$

$$w_3^0 = 40$$

$$w_3^1 = 40 + \left\lceil \frac{40}{20} \right\rceil \cdot 5 + \left\lceil \frac{40}{40} \right\rceil \cdot 10 = 60$$

$$w_3^2 = 40 + \left\lceil \frac{60}{20} \right\rceil \cdot 5 + \left\lceil \frac{60}{40} \right\rceil \cdot 10 = 75$$

$$w_3^3 = 40 + \left\lceil \frac{75}{20} \right\rceil \cdot 5 + \left\lceil \frac{75}{40} \right\rceil \cdot 10 = 80$$

$$w_3^4 = 40 + \left\lceil \frac{80}{20} \right\rceil \cdot 5 + \left\lceil \frac{80}{40} \right\rceil \cdot 10 = 80 \quad R_3 = 80$$

Todas las tareas tienen sus plazos garantizados

Propiedades del análisis de tiempo de respuesta

- Proporciona una condición necesaria y suficiente para la garantía de los plazos

$$\forall i \; R_i \leq D_i$$

- Permite un análisis del comportamiento temporal del sistema más exacto que la prueba del factor de utilización
- El elemento crítico es el cálculo del tiempo de cómputo de cada tarea
  - optimista: los plazos puedes fallar aunque el análisis sea positivo
  - pesimista: él análisis puede ser negativo aunque los plazos no fallen en realidad

Tiempo de cómputo

- Interesa el tiempo de ejecución en el peor caso *(WCET, worst case execution time)*
- Hay dos formas de obtener el WCET de una tarea:
  - **Medida** del tiempo de ejecución
    - » no es fácil saber cuándo se ejecuta el peor caso posible
  - **Análisis** del código ejecutable
    - » se descompone el código en un grafo de bloques secuenciales
    - » se calcula el tiempo de ejecución de cada bloque
    - » se busca el camino más largo

    Puede ser muy pesimista
    - » es difícil tener en cuenta los efectos de los dispositivos de hardware (caches, pipelines, estados de espera de la memoria, etc..)
    - » hace falta tener un modelo adecuado del procesador

Análisis estático del WCET

- Generalmente se hace en tres pasos:
  1. Descomposición del código en un grafo dirigido compuesto por bloques básicos (secuencias)
  2. Cálculo del WCET de cada bloque básico a partir del código de máquina y del modelo del procesador
  3. Cálculo del WCET total a partir del camino más largo del grafo

Mejora con información semántica

- Ejemplo:

```
for I in 1.. 10 loop
  if Cond then
    -- bloque básico con coste 100
  else
    -- bloque básico con coste 10
  end if;
end loop;
```

  - Sin más información, el coste peor es 10 × 100 = 1000
  - Si sabemos que `Cond` sólo es verdadera 3 veces, entonces el coste es 3×100 + 7×10 = 370

Restricciones en el código

- Para poder calcular el tiempo de cómputo hay que evitar utilizar estructuras con tiempo de cómputo no acotado, como:
  - bucles no acotados
  - recursión no acotada
  - objetos dinámicos
  - tareas dinámicas
- Para construir sistemas de tiempo real estricto se utilizan subconjuntos del lenguaje de programación que no usan estos elementos

  Ejemplos:
  - » SPARK (parte secuencial de Ada)
  - » Ravenscar (parte concurrente)

Tareas esporádicas y aperiódicas

Tareas esporádicas

- Para incluir tareas esporádicas hace falta modificar el modelo de tareas:
  - El parámetro ***T*** representa la **separación** mínima entre dos sucesos de activación consecutivos
  - Suponemos que en el peor caso la activación es pseudoperiódica (con período *T)*
  - El plazo de respuesta puede ser menor que el período ($D \leq T$)
- El análisis de tiempo de respuesta sigue siendo válido
- Funciona bien con cualquier orden de prioridad

Prioridades monótonas en plazos

Cuando los plazos son menores o iguales que los períodos, la asignación de mayor prioridad a las tareas de menor plazo de respuesta *(deadline monotonic scheduling)* es **óptima**

- El tiempo de respuesta se calcula de la misma forma que con la asignación monótona en frecuencia
  - se termina cuando $w_i^{n+1} = w_i^n$,
  - o cuando $w_i^{n+1} > D_i$

Ejemplo 5

Tarea	T	D	C	P	R
τ1	20	5	3	4	3
τ2	15	7	3	3	6
τ3	10	10	4	2	10
τ4	20	20	3	1	20

Con prioridades monótonas en frecuencia los plazos no están garantizados:

Tarea	T	D	C	P	R
τ3	10	10	4	4	4
τ2	15	7	3	3	7
τ1	20	5	3	2	10
τ4	20	20	3	1	20

Tareas críticas y acríticas

- A menudo los tiempos de cómputo en el peor caso de las tareas esporádicas son mucho más altos que los medios
  - interrupciones en rachas, tratamiento de errores
  - planteamiento demasiado pesimista
- No todas las tareas esporádicas son críticas
  - Deben garantizarse los plazos de todas las tareas en condiciones "normales"
    - » con separación entre activaciones y tiempos de cómputo medios

    Puede haber una **sobrecarga transitoria**
  - Deben garantizarse los plazos de las tareas críticas en las peores condiciones
    - » con separación entre activaciones y tiempos de cómputo peores

    Esto asegura un comportamiento correcto en caso de sobrecarga transitoria

Tareas aperiódicas

- Son tareas acríticas sin separación mínima
- Se pueden ejecutar con prioridades más bajas que las tareas críticas (periódicas y esporádicas)
  - el tiempo de respuesta puede ser muy largo
  - en condiciones normales sobra tiempo de cómputo de las tareas críticas
- Es mejor utilizar un **servidor**
  - el servidor asegura que las tareas críticas tienen asegurados sus recursos
  - pero asignan los recursos que no se utilizan a las tareas acríticas

Servidor esporádico

- Un **servidor esporádico** *(SS, sporadic server)* es un proceso periódico
  - Parámetros: período T_s, tiempo de cómputo C_s, prioridad máxima
    - » C_s es la *capacidad inicial* del servidor
    - » T_s y C_s se eligen de forma que las tareas críticas estén garantizadas
  - Cuando se activa una tarea aperiódica, se ejecuta con prioridad máxima mientras quede capacidad disponible
  - Cuando se agota la capacidad se ejecuta con prioridad baja
  - La capacidad se rellena cuando ha pasado un tiempo T_s desde la activación de la tarea aperiódica