UD 2: Programación concurrente y sistemas de tiempo real

Tema 7 - Planificación

« Me lo explicaron y lo olvidé, lo ví y lo aprendí, lo hice y lo entendí.»

- Confucio -

industriales

etsii UPCT

Universidad Politécnica de Cartagena

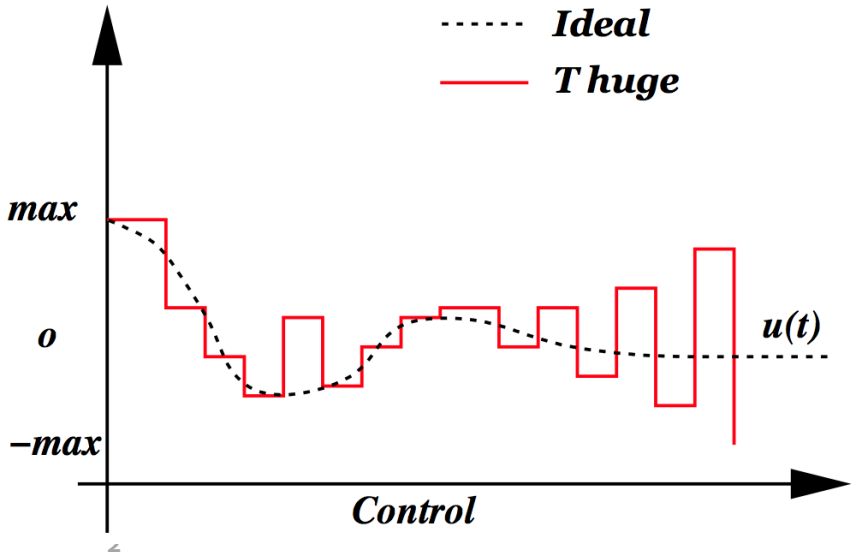
Recordatorio

- Un sistema de tiempo-real es aquel en el que no solo el cómputo debe ser correcto, sino que además debe obtenerse a tiempo
- El requisito temporal estricto se debe a la interacción con el mundo físico, particularmente al periodo de muestro (T) en control discreto

Control continuo
Laplace (s)

$$s = \frac{z-1}{T}$$

Control discreto
Transformada Z



Tareas y Trabajos en Tiempo-Real

- **Tarea**: secuencia de instrucciones que manipulan datos
- **Trabajo**: instancia de una tarea con sus datos específicos
- Una tarea se caracteriza por:
  - **Requisitos** de la tarea: características requeridas por el diseñador
  - **Propiedades** de la tarea: dependen de la implementación específica de la tarea y del sistema en que se ejecuta. Entre éstas destacan su tiempo de ejecución (peor, medio y mejor), tiempo de respuesta, tiempo de activación, etc.

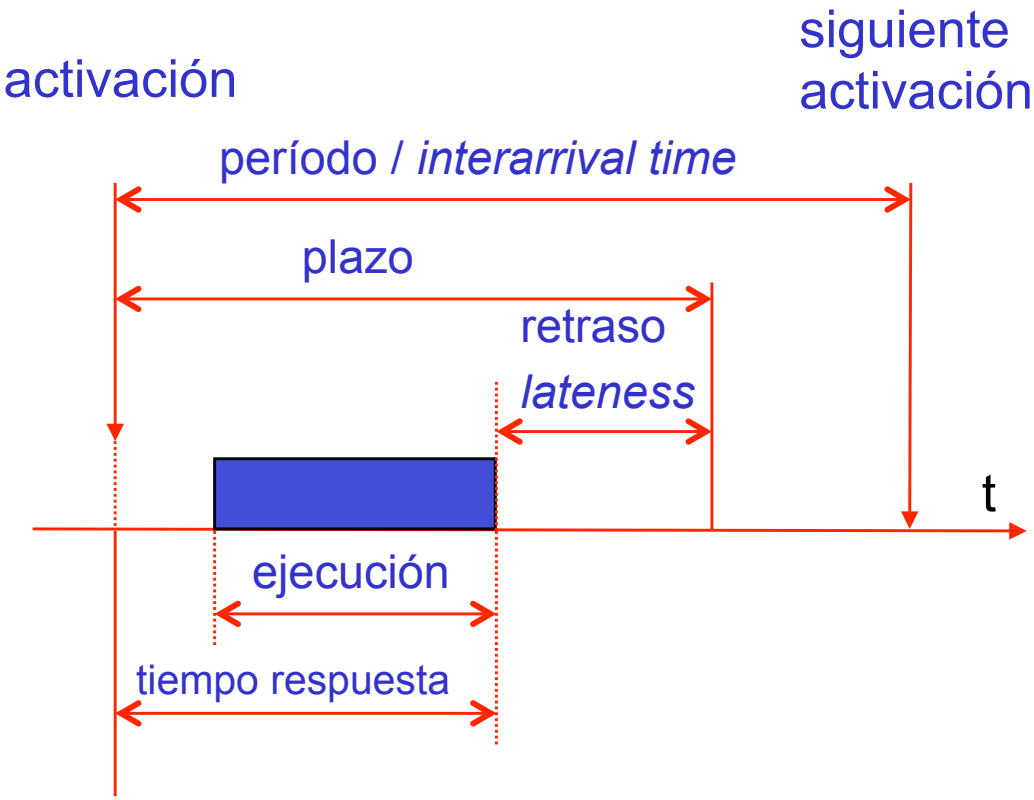
Requisitos Temporales de una Tarea

- Tareas **periódicas**: aquellas en que los trabajos se activan en intervalos regulares de tiempo (periodo de la tarea). Se define fase como la primera vez que se activa un trabajo
- Tareas **esporádicas**: aquellas en que la activación consecutiva de dos trabajos está separada un tiempo mínimo (*interarrival time*)
- Tareas **aperiódicas**: aquellas en que los trabajos se pueden activar en intervalos arbitrarios
- Plazo (*deadline*): tiempo máximo en que tiene que acabar el trabajo
- También es posible establecer relaciones de precedencia entre tareas

Requisitos de una Tarea: Importancia

- **Crítica (hard):** la pérdida de una plazo tiene consecuencias muy importantes sobre el funcionamientos del sistema
- **Firme (firm):** la pérdida de un plazo no tiene fuertes consecuencias sobre el sistema, pero el cómputo tardío es inútil
- **Normal (soft):** la pérdida de un plazo no tiene fuertes consecuencias sobre el sistema, y el cómputo tardío todavía conserva cierto valor
- En una aplicación es normal definir tareas con distintos niveles de importancia

Terminología (I)



- La variación en el tiempo de los trabajos se llama en general *Jitter.* Puede ser de activación, del tiempo de respuesta, etc.

Terminología (II)

- **N**: Número de tareas
- $\mathbf{T_i}$: Período de activación de la tarea i-ésima
- $\mathbf{C_i}$: Tiempo de ejecución medio de la tarea i-ésima
- $\mathbf{D_i}$: Plazo de respuesta de la tarea i-ésima
- $\mathbf{R_i}$: Tiempo de respuesta máximo de la tarea i-ésima
- $\mathbf{P_i}$: Prioridad de la tarea i-ésima

Problema de Planificación

- Dado un conjunto de tareas más sus requisitos temporales, hay que encontrar una ordenación (planificación) de las mismas tal que se cumpla que $R_i<D_i$. Se dice entonces que el sistema es *planificable*
- Un método de planificación define:
  - Un **algoritmo de planificación**, que determina el orden de acceso de las tareas a los recursos del sistema (CPU principalmente)
  - Un **método de análisis,** que permite calcular el comportamiento temporal del sistema
    - Se puede comprobar si los requisitos temporales están garantizados en todos los casos posibles
    - En general, se estudia el peor comportamiento posible

Método 1: Ejecutivo Cíclico

- Si todas las tareas son periódicas se puede confeccionar un plan de ejecución fijo, un esquema que se repite cada ciclo principal:

$$T_M = \text{mcm}(T_i) \quad \text{hiperperiodo}$$

- El ciclo principal se divide en ciclos secundarios con período T_S

$$T_M = k \cdot T_S \qquad \exists i: \ {}^{T_i}/_{T_s} - \left\lfloor {}^{T_i}/_{T_s} \right\rfloor = 0$$

$$T_S \geq max(C_i) \qquad \forall i: \ 2 \cdot T_S - mcd(T_S, T_i) \leq D_i \ \text{(a)}$$

- Las tareas se reparten entre los ciclos secundarios de forma que el sistema sea planificable

- El programa se ejecuta en un único proceso

Ejemplo Ejecutivo Cíclico (I)

- $T_m = mcm(T_i) \rightarrow 100$
- $T_m > T_s > \max(C_i) \rightarrow 100 > T_s > 10$
  $T_s = \{50, 25, 20, 10\}$

Tarea	T	C
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2

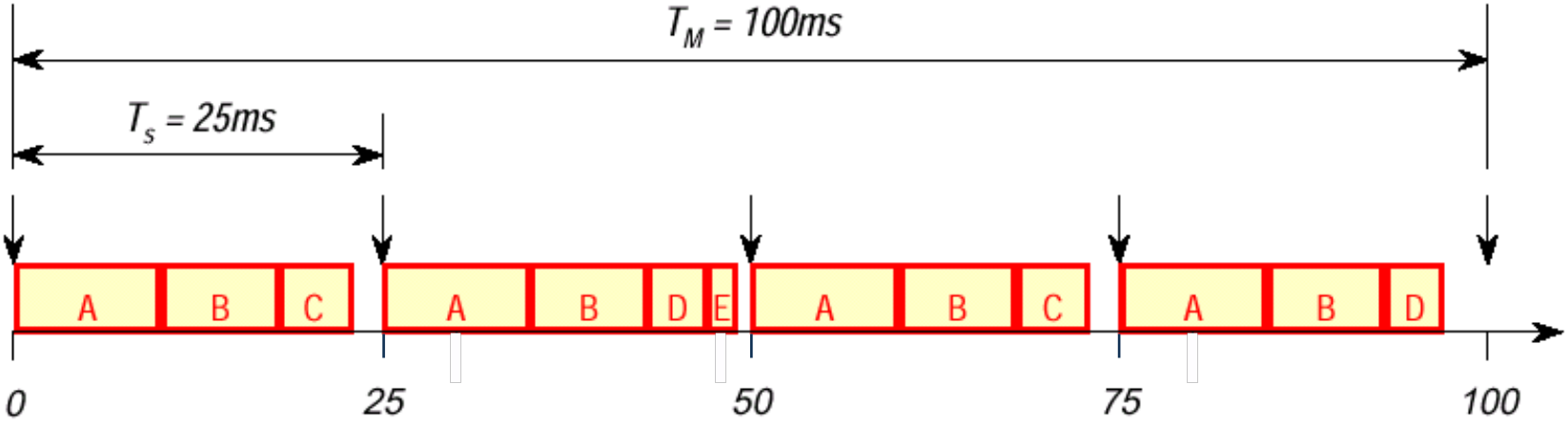
$T_s = 50$	$T_s = 25$
$2 \cdot 50 - mcd(50, 25) \nleq 25$ ✗	$2 \cdot 25 - mcd(25,25) \leq 25$ $2 \cdot 25 - mcd(25,50) \leq 50$ $2 \cdot 25 - mcd(25,100) \leq 100$ ✓

$T_s = 20$	$T_s = 10$
$2 \cdot 20 - mcd(20, 25) \nleq 25$ ✗	$2 \cdot 10 - mcd(10,25) \leq 25$ $2 \cdot 10 - mcd(10,50) \leq 50$ $2 \cdot 10 - mcd(10,100) \leq 100$ ✓

Ejemplo Ejecutivo Cíclico (II)

Tarea	T	C
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2

- El ciclo principal dura 100ms
- Se compone de 4 ciclos secundarios de 25ms cada uno



Cronograma

Implementación del Ejemplo

```
void main (void) {
   long ciclo=0, siguiente = clock(), periodo = 25;
   while (1) {
      switch (ciclo) {
         case 0 : A(); B(); C(); ++ciclo; break;
         case 1 : A(); B(); D(); E(); ++ciclo; break;
         case 2 : A(); B(); C(); ++ciclo; break;
         case 3 : A(); B(); D(); ciclo=0; break;
      }
      siguiente += periodo;
      sleep_until(siguiente);
   }
}
```

Ventajas del Ejecutivo Cíclico

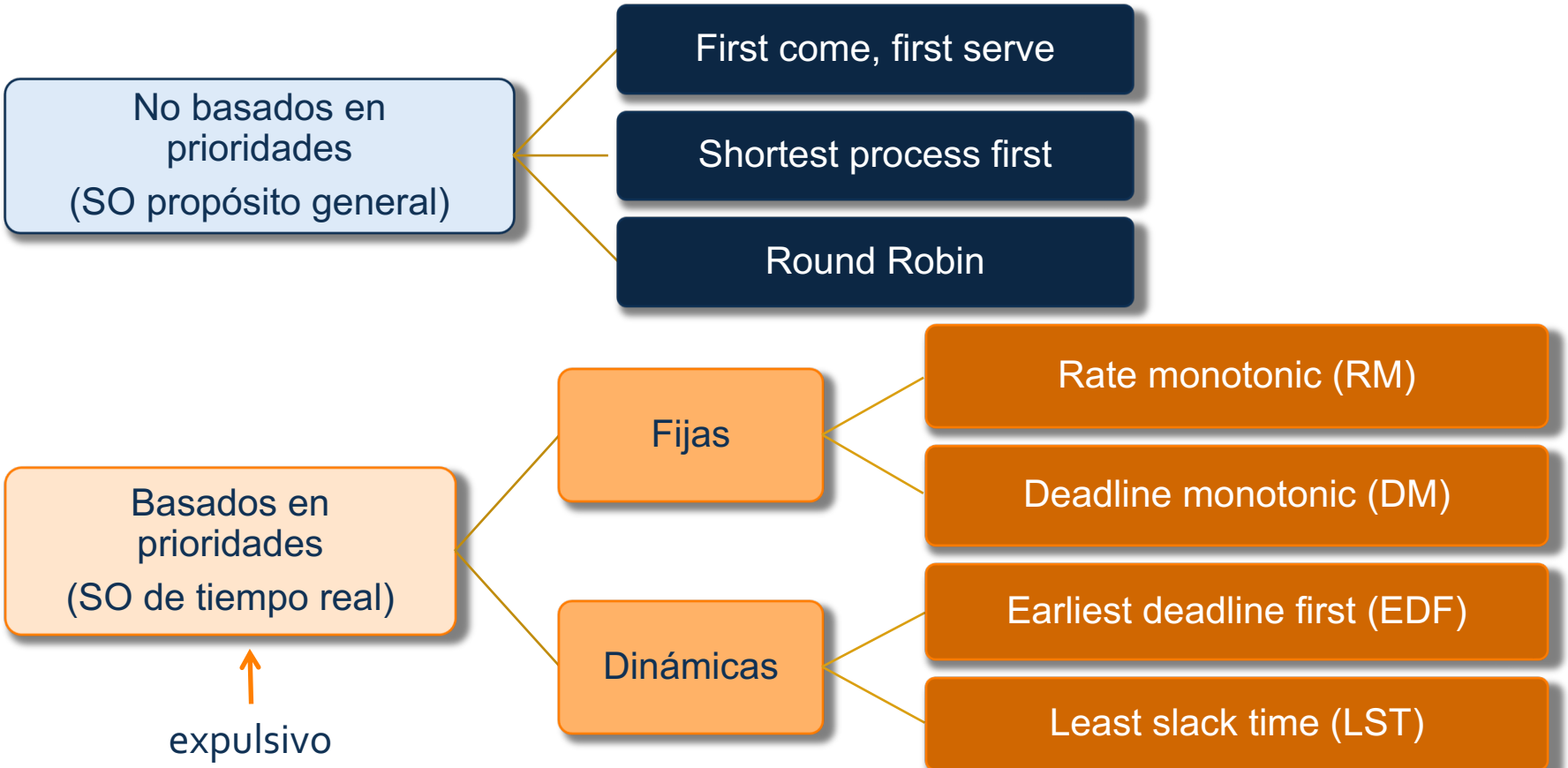
- No hay concurrencia, el programa es puramente secuencial
- Las tareas comparten datos sin necesidad de exclusión mutua
- No hace falta analizar el comportamiento temporal: el sistema es correcto por construcción
- La implementación es trivial (una vez desarrollado el código)
- No se necesita SO, aplicaciones sobre máquina desnuda

En determinados sectores es lo único que se permite y certifica

Inconvenientes del Ejecutivo Cíclico

- Las tareas esporádicas son difíciles de tratar (generalmente se añade un servidor esporádico)
- El plan cíclico es difícil de construir:
  - Si los períodos son muy dispares, el número de ciclos secundarios puede ser muy grande
  - Puede ser necesario partir una tarea en muchas actividades
- Poco flexible y difícil de mantener, ya que cada vez que cambia una tarea hay que rehacer la planificación
- Para resolver estos problemas: planificación multi-tarea

Algoritmos de Planificación para Sistemas de Tiempo Real



Planificación con Prioridades Fijas

- Es un método estático, el más utilizado
- La prioridad es un parámetro relacionado con la urgencia o la importancia de la tarea
- En cada momento se ejecuta la tarea con mayor prioridad de todas las ejecutables
- Si se activa una tarea de mayor prioridad que la que se está ejecutando, se expulsa a ésta de la CPU (planificación expulsiva)
- Existen dos métodos fundamentales: prioridades monótonas en frecuencia y en plazo

Modelo de Tareas Simple

- El conjunto de tareas es estático (se conoce en tiempo de diseño)
- Todas las tareas son periódicas (se activan a intervalos regulares)
- Las tareas son independientes unas de otras (no hay comunicación entre ellas)
- El tiempo de ejecución máximo de cada tarea es conocido
- El plazo (*deadline*) coincide con el periodo de la tarea
- Las operaciones del SO son instantáneas (no hay tiempo para expulsar una tarea y asignar el procesador a una nueva)

Método 2: Prioridades Monótonas en Frecuencia

- La asignación de mayor prioridad a las tareas de menor período (*rate monotonic scheduling*) es óptima para el modelo de tareas simple
- 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
- Este método de planificación tiene un método de análisis asociado, *rate monotonic analysis* (RMA)

Factor de Utilización (teorema de Liu y Layland)

- Es la fracción de tiempo que la CPU está ocupada ejecutando instrucciones de las tareas del sistema, definida como: $U = \sum_{i=1}^{i=N} \frac{C_i}{T_i}$
- Es una medida de la carga de la CPU para un conjunto de tareas, que en un sistema monoprocesador debe ser U <= 1
- Para el modelo simple, con prioridades monótonas en frecuencia, los plazos están garantizados si:

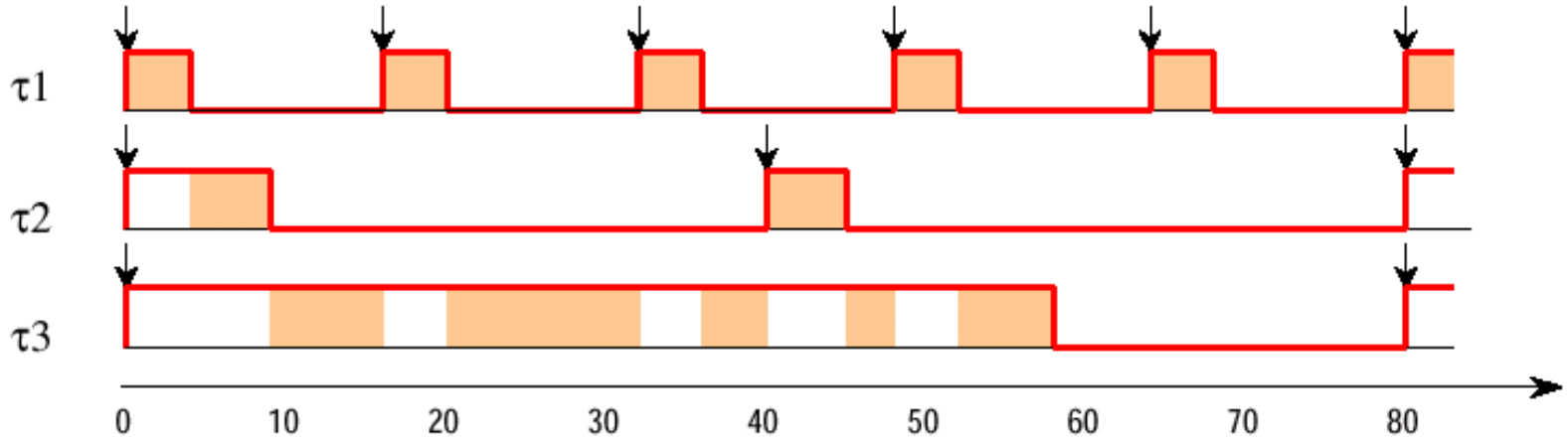
$$U = \sum_{i=1}^{i=N} \frac{C_i}{T_i} \leq N \cdot \left(2^{\frac{1}{N}} - 1\right)$$

N	U
1	1,0
2	0,828
3	0,779
4	0,756
5	0.743
∞	0,693

Ejemplo 1

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 ($U < 0{,}779$)



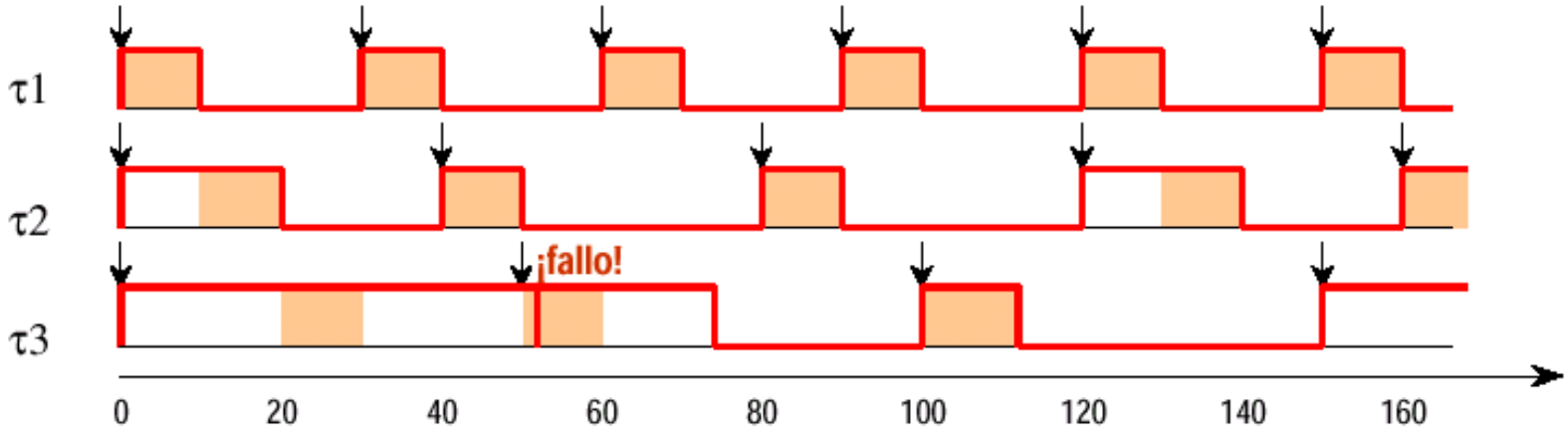
Ejemplo 2

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)

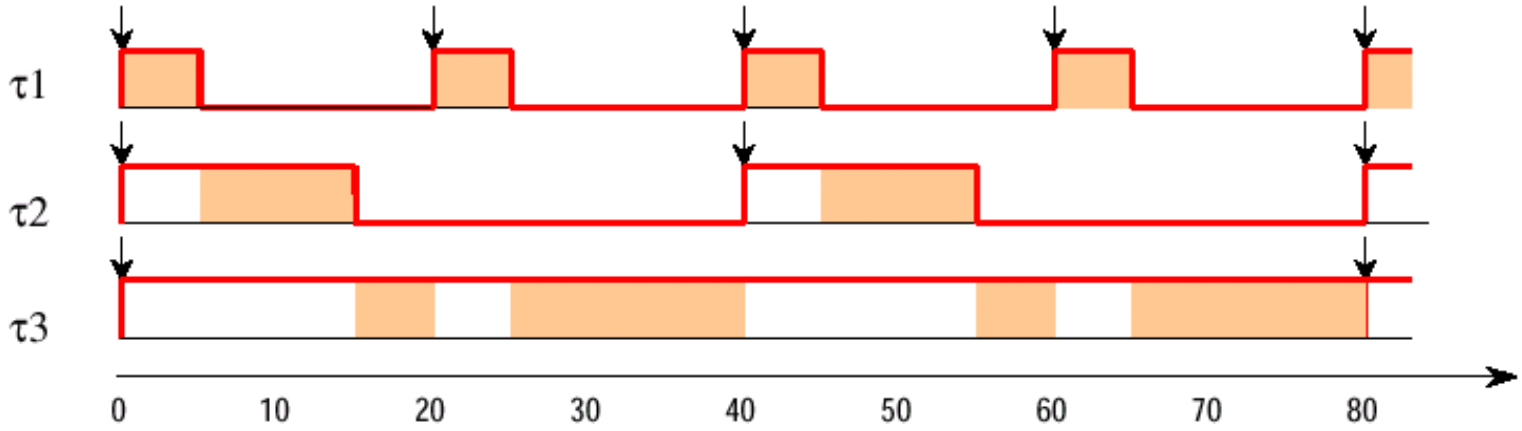
La tarea 3 falla en $t = 50$



Ejemplo 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

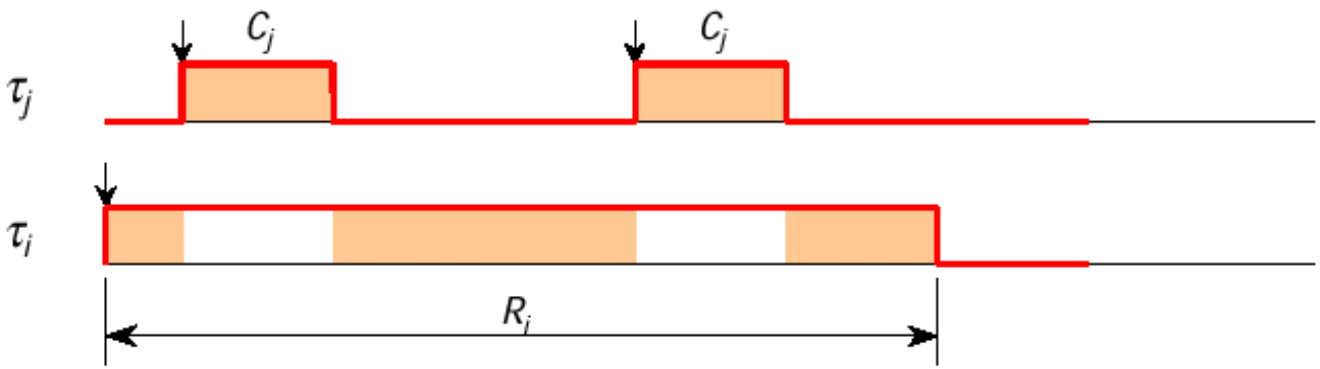
Este sistema no pasa la prueba ($U > 0{,}779$),
pero se cumplen los plazos



Análisis del Tiempo de Respuesta

- Es inexacto: da una condición suficiente pero no necesaria
- Solo funciona con el modelo simple, pero no se ajusta a modelos más complejos
- La construcción de un cronograma es compleja, incluso considerando crítico el instante inicial
- El análisis basado en el cálculo del tiempo de respuesta de cada tarea sí que proporciona un resultado definitivo

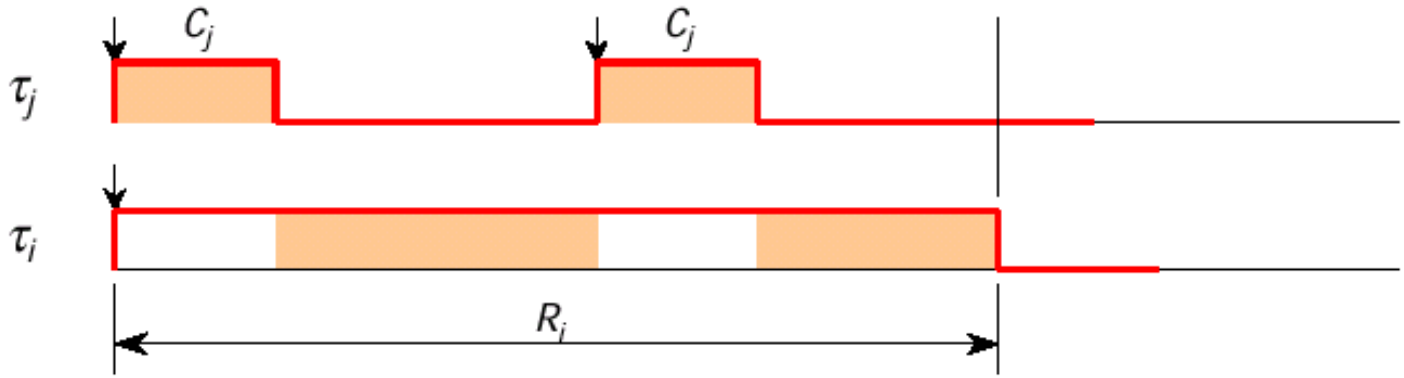
Tiempo de Respuesta



$$R_i = C_i + I_i$$

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

Cálculo de la Interferencia Máxima



Para una tarea
de prioridad superior

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

Para todas las tareas
de prioridad superior

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

Cálculo del Tiempo de Respuesta

- La ecuación del tiempo de respuesta es la siguiente:

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

'j' toma como valor los índices de las tareas que tienen mayor prioridad que 'i'

- Es una ecuación que no es continua ni lineal
- No se puede resolver analíticamente, sino iterativamente

Cálculo Iterativo del Tiempo de Respuesta

- La solución iterativa requiere que la incógnita aparezca a ambos lados de la ecuación

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

- Un valor inicial aceptable para comenzar el cálculo de cada R_i es:

$$W_i^0 = C_i + \sum_{j \in mp(i)} C_j$$

- La iteración termina cuando se cumple una de estas dos condiciones

$$\begin{cases} W_i^{n+1} = W_i^n < T_i \ (planificable) \\ W_i^{n+1} > T_i \ (no\ planificable) \end{cases}$$

Ejemplo 4

Tarea	T	C	P	R
τ1	7	3	3	3
τ2	12	3	2	6
τ3	20	5	1	20

$$\tau_1: \quad w_1^0 = 3;$$

$$\tau_2: \quad w_2^0 = 3 + 3 = 6;$$

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

$$\tau_3: \quad w_3^0 = 5 + 3 + 3 = 11;$$

$$w_3^1 = 5 + \left\lceil \frac{11}{7} \right\rceil \cdot 3 + \left\lceil \frac{11}{12} \right\rceil \cdot 3 = 14;$$

$$w_3^2 = 5 + \left\lceil \frac{14}{7} \right\rceil \cdot 3 + \left\lceil \frac{14}{12} \right\rceil \cdot 3 = 17;$$

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

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

- A pesar de que U=0,928 , todas las tareas tienen sus plazos garantizados
- Es condición suficiente y necesaria, ya que se ha comprobado el tiempo de respuesta de todas las tareas

Mejora 1: Inclusión de Tareas Esporádicas

- Para incluir tareas esporádicas hace falta modificar el modelo de tareas simple:
  - El parámetro T representa la separación mínima entre dos sucesos de activación consecutivos.
  - El plazo de respuesta puede ser menor que el periodo (D < T)

Método 3: Prioridad Monótona en Plazo

- 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:

$$W_i^{n+1} = W_i^n < D_i \ (planificable)$$

$$W_i^{n+1} > D_i \ (no\ planificable)$$

Ejemplo 5

Monótona en plazo

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

Monótona en frecuencia

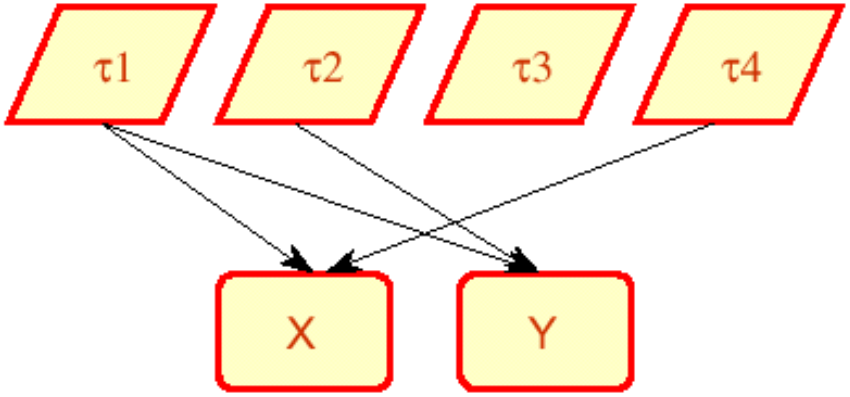
Tarea	T	D	C	P	R
τ3	10	10	4	4	4
τ2	15	7	3	3	7
τ1	20	5	3	2	10
τ4	20	20	3	1	20

- Con prioridades monótonas en frecuencia los plazos de respuesta no están garantizados
- Solo la asignación de prioridades monótona en plazo lo asegura

Mejora 2: Comunicación entre Tareas

- Lo normal es que las tareas interaccionen mediante datos comunes o paso de mensajes
- Aparece un nuevo problema: la **inversión de prioridad**
  - Una tarea de mayor prioridad queda bloqueada en un mutex cogido por otra de menor prioridad, que es a su vez bloqueada por otra tarea de prioridad intermedia
- Se utilizan dos métodos para limitar su duración:
  - Herencia de prioridad
  - Techo de prioridad

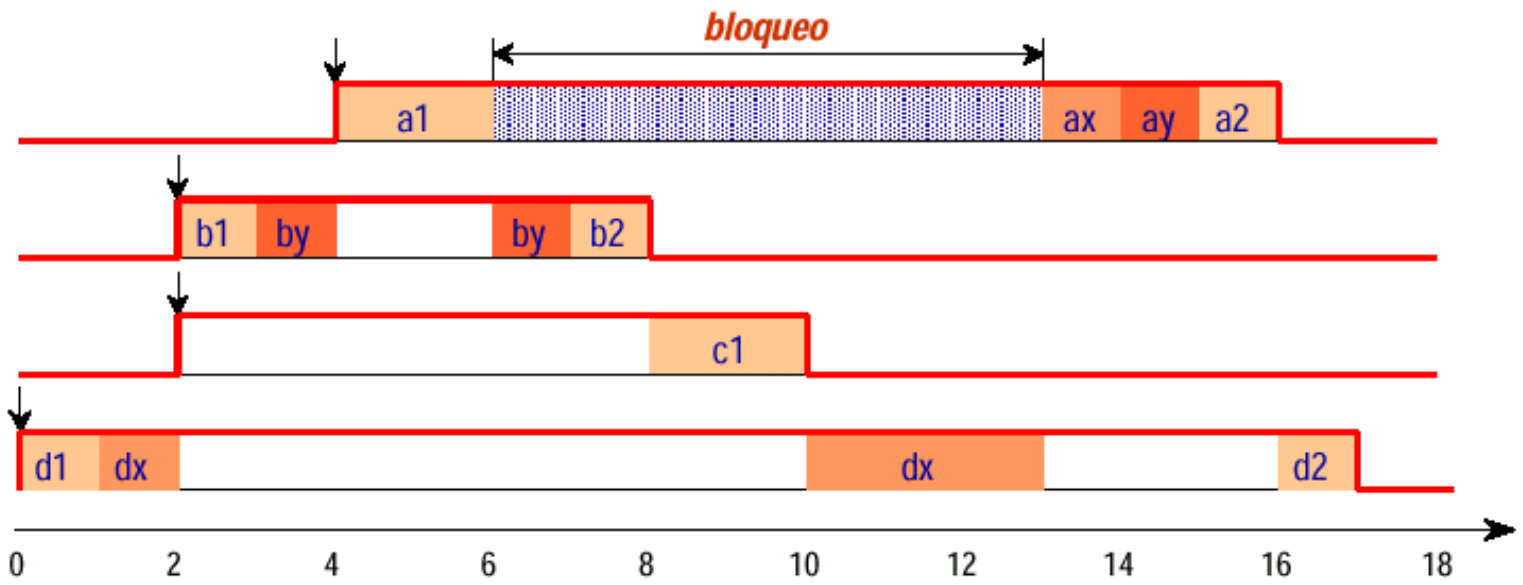
Ejemplo 6



Tarea	P	ta	Acciones
τ1	4	4	a1; ax; ay; a2
τ2	3	2	b1; by; b2
τ3	2	2	c1
τ4	1	0	d1; dx; d2

Acción	P	C	usa
a1	4	2	
ax	4	1	X
ay	4	1	Y
a2	4	1	
b1	3	1	
by	3	2	Y
b2	3	1	
c1	2	2	
d1	1	1	
dx	1	4	X
d2	1	1	

Ejemplo 6: Inversión de Prioridad

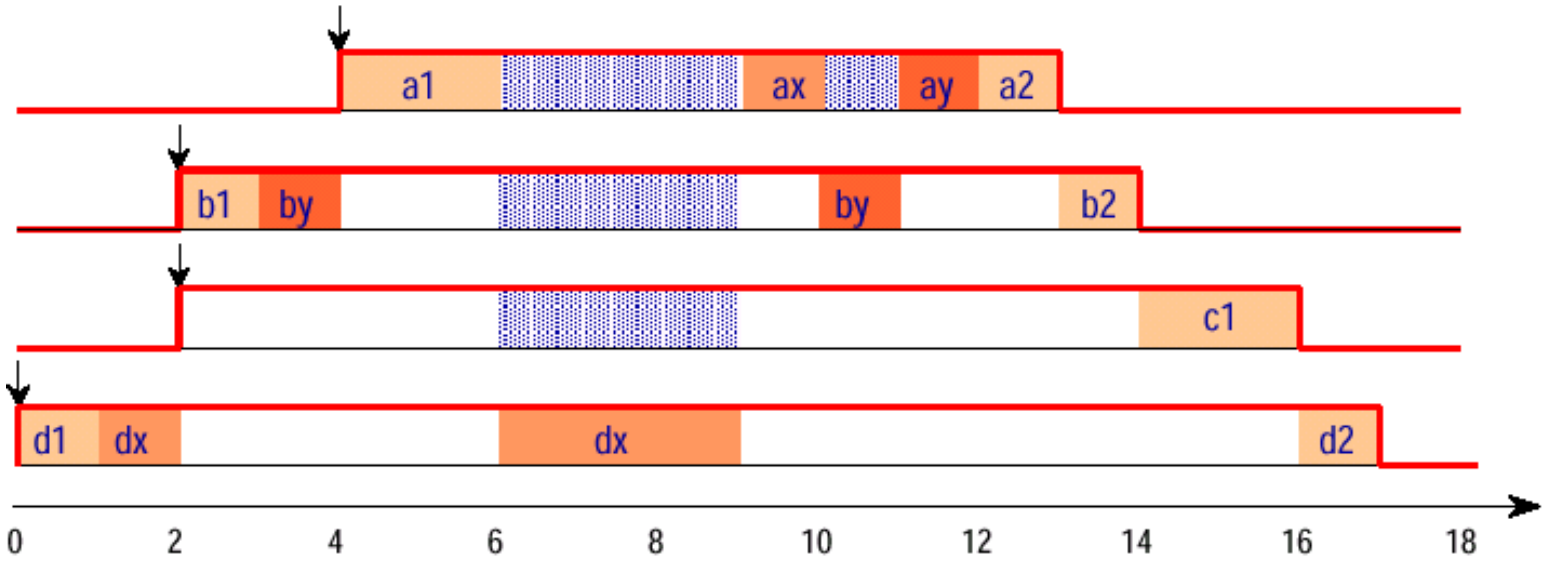


- El proceso más prioritario sufre una importante inversión de prioridad
- No sólo es bloqueado por t4 que accede a región crítica x, sino también por t2 y t3

Protocolo de Herencia de Prioridad

- Una forma de reducir la duración de los bloqueos es variar dinámicamente la prioridad de las tareas
- Cuando una tarea está bloqueando a otra más prioritaria, hereda la prioridad de ésta. Por tanto, las prioridades se vuelven dinámicas, aunque con un "dinamismo" controlado
- La prioridad dinámica de una tarea es el máximo de
  - Su prioridad básica
  - Las prioridades de todas las tareas a las que va bloqueando

Ejemplo 6: Herencia de Prioridad



- Cuando t1 intenta acceder a x, la tarea t4 hereda su prioridad y termina su ejecución de la región crítica x
- La tarea t3 sufre bloqueo sin acceder a recursos y t1 sufre un segundo bloqueo pero su tiempo de respuesta pasa de 12 a 9 unidades temporales

Cálculo de la Duración del Bloqueo

- Con el protocolo de herencia de prioridad, una tarea puede bloquear como máximo
  - una vez por cada tarea de prioridad inferior
  - una vez por cada sección crítica
- La duración de bloqueo es:

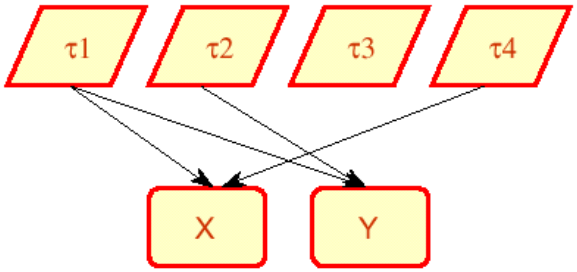
$$B_i = \sum_{k=1}^{K} usa(\tau_i, R_k) \cdot C(R_k)$$

$C(R_k)$ es el WCET del recurso R_k

R_k referencia el recurso compartido 'k'

$usa(\tau_i, R_k)$ es una función que vale '1' si el recurso R_k es utilizado por, al menos, una tarea con prioridad menor que τ_i y otra con prioridad igual o superior a τ_i. 'o' en caso contrario

Cálculo del Bloqueo en el Ejemplo 6



$C(R_x) = \max(C_x) = \max(1,4) = 4$

$C(R_y) = \max(C_y) = \max(1,2) = 2$

$B_1 = usa(\tau_1, R_x) \cdot C(R_x) + usa(\tau_1, R_y) \cdot C(R_y) = 4+2 = 6$

$B_2 = usa(\tau_2, R_x) \cdot C(R_x) + usa(\tau_2, R_y) \cdot C(R_y) = 4+0 = 4$

$B_3 = usa(\tau_3, R_x) \cdot C(R_x) + usa(\tau_3, R_y) \cdot C(R_y) = 4+0 = 4$

$B_4 = usa(\tau_4, R_x) \cdot C(R_x) + usa(\tau_4, R_y) \cdot C(R_y) = 0+0 = 0$

- Una tarea puede bloquearse por recursos a los que accede (por ejemplo, τ_2)
- Una tarea puede sufrir bloqueo aunque no acceda a recursos compartidos (por ejemplo, τ_3)
- La tarea de menor prioridad (τ_4) no sufre bloqueo

Tiempo de Respuesta con Herencia de Prioridad

- Cuando hay bloqueos, la ecuación del tiempo de respuesta es:

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

- La solución se obtiene mediante la relación de recurrencia:

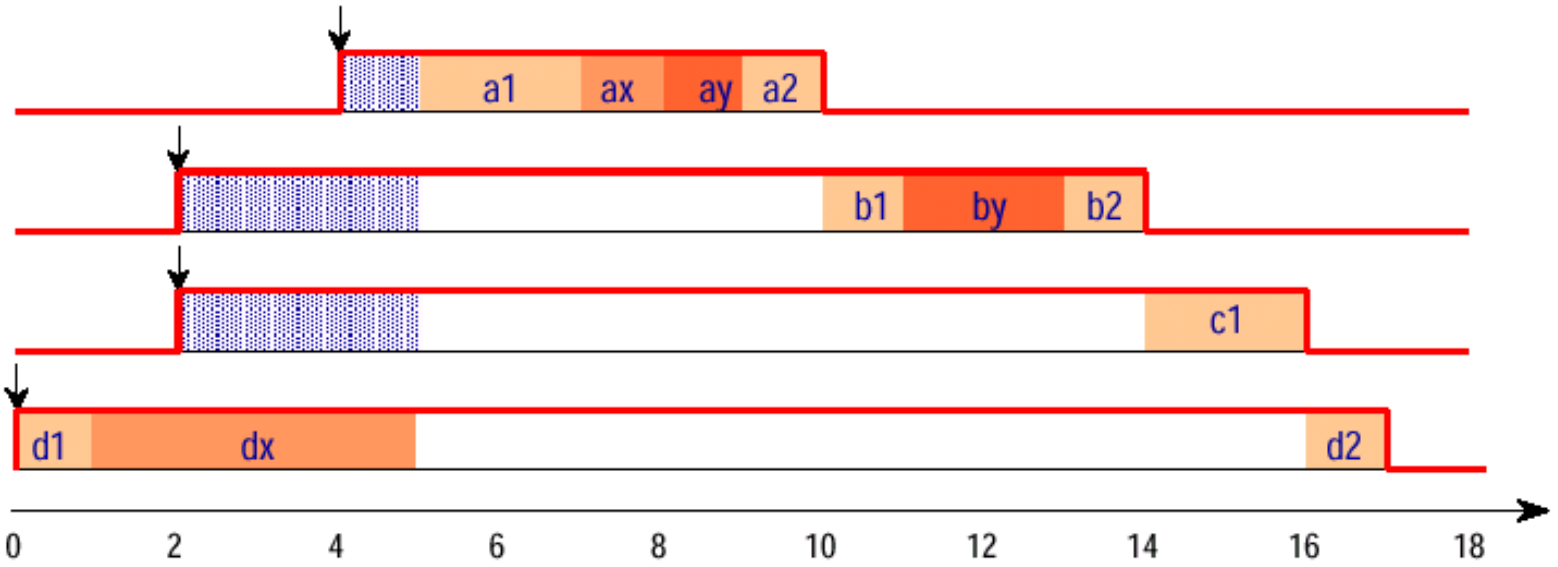
$$W_i^{n+1} = C_i + B_i + \sum_{j \in mp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil \cdot C_j$$

- En este caso, el análisis del tiempo de respuesta responde a una solución (muy) pesimista

Protocolo de Techo de Prioridad Inmediato

- Sustituye al Techo de Prioridad ya que es más fácil de implementar y es más eficiente (menos cambios de contexto al producirse el bloqueo antes de cada ejecución)
- El techo de prioridad de un recurso es la máxima prioridad de todas las tareas que usan dicho recurso (se calcula y fija manualmente)
- Con este protocolo, una tarea que accede a un recurso hereda inmediatamente el techo de prioridad del recurso
  - La prioridad dinámica de una tarea es el máximo de su prioridad básica y los techos de prioridad de los recursos que usa (se calcula a mano)
- Las tareas se bloquean solo al principio del ciclo

Ejemplo 6: Techo de Prioridad Inmediato



- Cuando t_4 accede a x hereda la prioridad 4 y ejecuta hasta liberar el recurso
- A partir de entonces, todas las tareas ya activadas se ejecutan en orden de prioridad

Cálculo de la Duración del Bloqueo

- Con este protocolo, una tarea puede bloquear como máximo
  - una vez por ciclo, con valor máximo igual a la sección crítica **más larga**
  - no puede haber bloqueos encadenados
- La duración de bloqueo es:

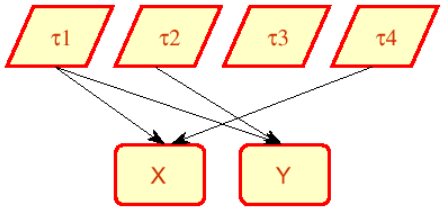
$$B_i = \max_k \left(usa(\tau_i, R_k) \cdot C(R_k) \right)$$

$C(R_k)$ es el WCET del recurso R_k

R_k referencia el recurso compartido 'k'

$usa(\tau_i, R_k)$ vale '1' si el recurso R_k es utilizado por, al menos, una tarea con prioridad menor que τ_i y otra con prioridad igual o superior a τ_i. '0' en caso contrario

Cálculo del Bloqueo en el Ejemplo 6



$C(R_x) = \max(C_x) = \max(1,4) = 4$

$C(R_y) = \max(C_y) = \max(1,2) = 2$

$B_1 = \max(usa(\tau_1, R_x) \cdot C(R_x), usa(\tau_1, R_y) \cdot C(R_y)) = \max(4,2) = 4$

$B_2 = \max(usa(\tau_2, R_x) \cdot C(R_x), usa(\tau_2, R_y) \cdot C(R_y)) = \max(4,0) = 4$

$B_3 = \max(usa(\tau_3, R_x) \cdot C(R_x), usa(\tau_3, R_y) \cdot C(R_y)) = \max(4,0) = 4$

$B_4 = \max(usa(\tau_4, R_x) \cdot C(R_x), usa(\tau_4, R_y) \cdot C(R_y)) = \max(0,0) = 0$

- Una tarea puede bloquearse por recursos a los que accede (por ejemplo, τ_2)
- Una tarea puede sufrir bloqueo aunque no acceda a recursos compartidos (por ejemplo, τ_3)
- La tarea de menor prioridad (τ_4) no sufre bloqueo