dit
UPM

Sistemas de tiempo real basados en ejecutivos cíclicos

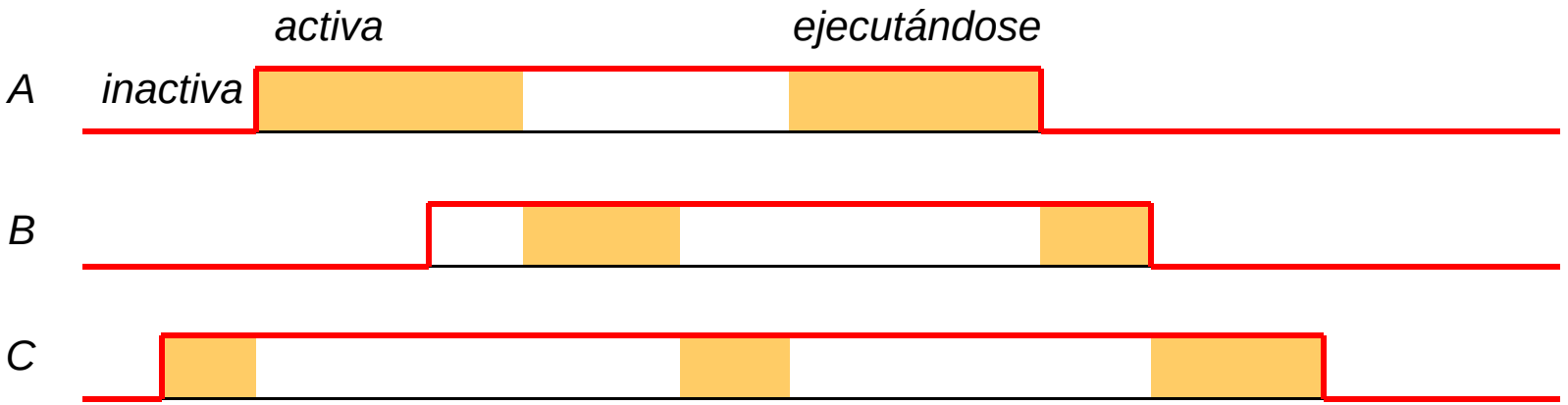
Juan Antonio de la Puente
DIT/UPM

Objetivos

- Repasar algunos problemas concretos relacionados con la realización de sistemas de tiempo real
- Introducir un método de diseño de sistemas basado en una **arquitectura síncrona**
  - el elemento fundamental es un **ejecutivo cíclico**
- Evaluar las ventajas e inconvenientes de este tipo de arquitectura

Concurrencia

- Los sistemas de tiempo real controlan actividades del mundo exterior que son simultáneas
- Para ello deben ejecutar varias actividades o **tareas** en paralelo (concurrentemente)
- La ejecución de las tareas se multiplexa en el tiempo en uno o varios procesadores

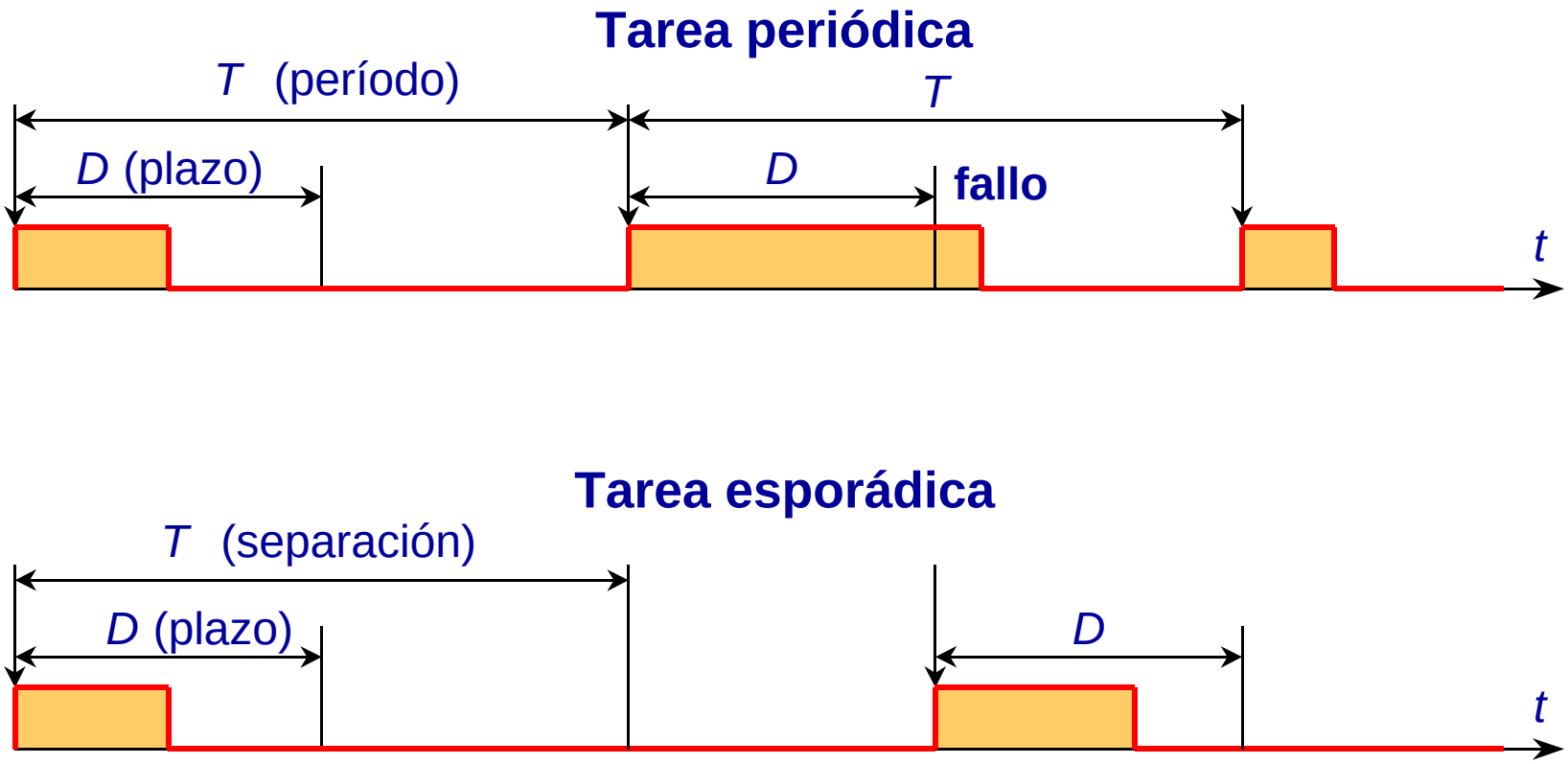


Requisitos temporales

Los requisitos de tiempo real suelen referirse a

- El *principio* del intervalo de ejecución (**esquema de activación**)
  - **Tareas periódicas:** se ejecutan a intervalos regulares
  - **Tareas esporádicas:** se ejecutan cuando ocurren determinados **sucesos**
    (en instantes distribuidos irregularmente)
- El *final* del intervalo de ejecución
  - Se suele especificar un **plazo** (relativo al instante de activación) para terminar la ejecución

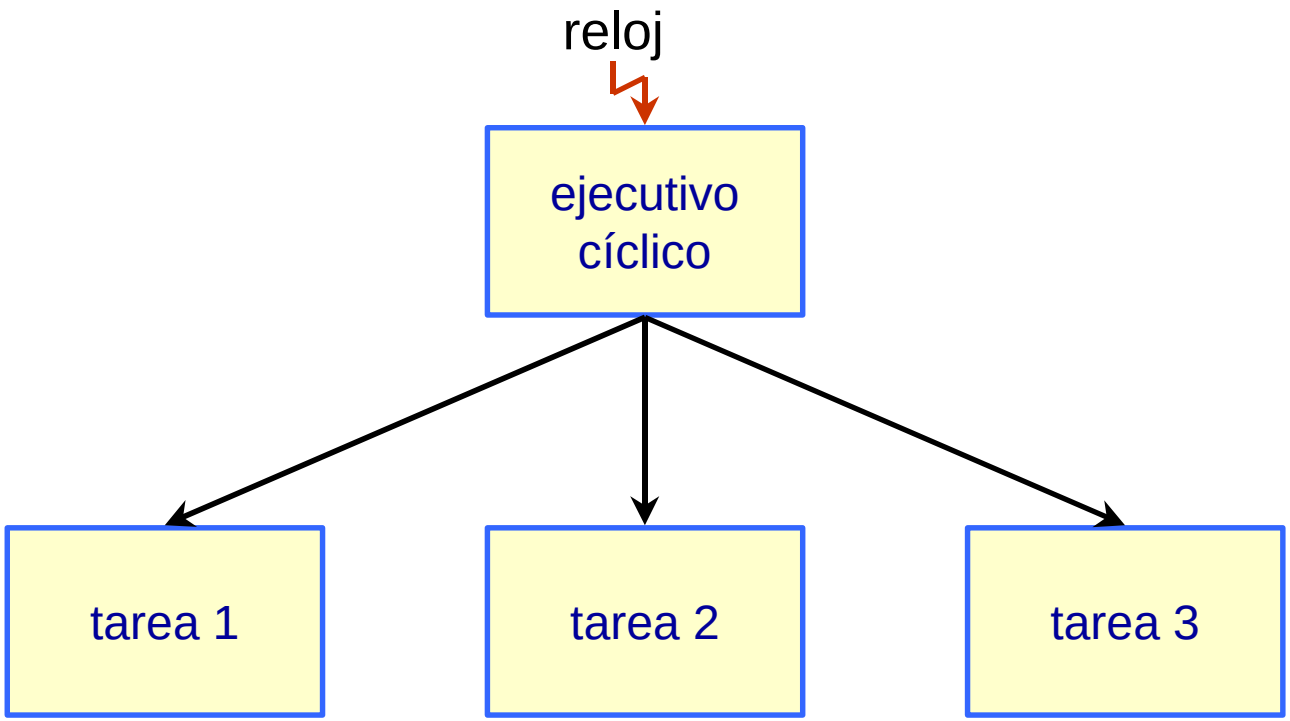
Tareas periódicas y esporádicas



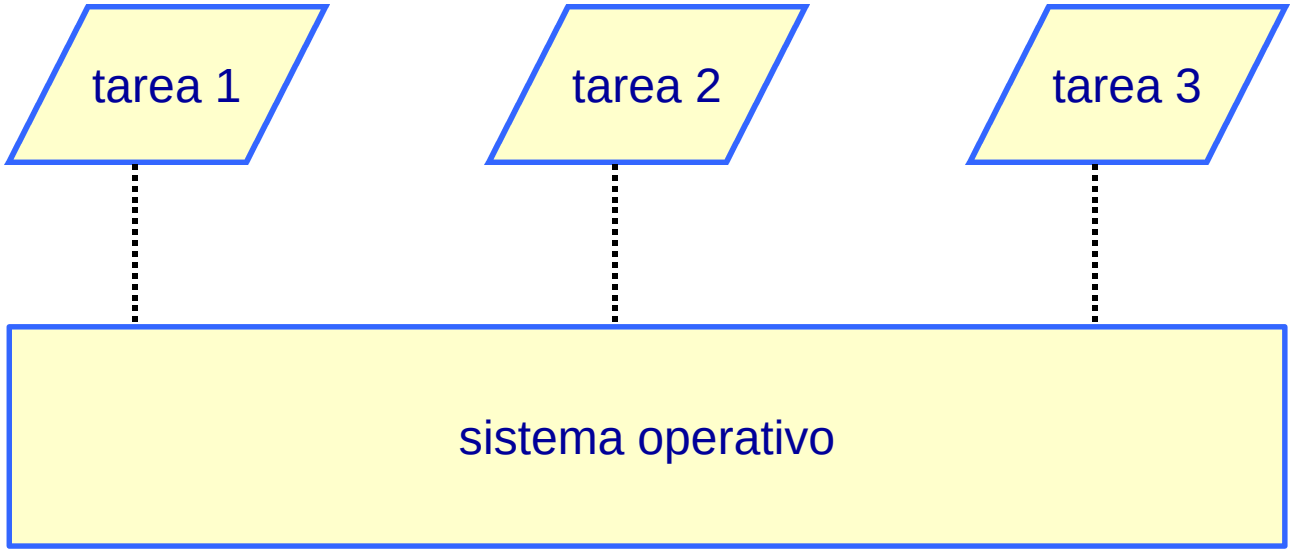
Arquitecturas de software

- Definen la estructura general de una clase de sistemas de tiempo real
- Hay dos tipos principales de arquitecturas
  - **Arquitectura síncrona**
    - » las tareas se ejecutan según un **plan de ejecución** fijo (realizado por el diseñador)
    - » el sistema operativo se reemplaza por un **ejecutivo cíclico**
  - **Arquitectura asíncrona**
    - » las tareas se reparten el procesador de forma dinámica (invisible para el diseñador)
    - » cada tarea tiene una **prioridad**
    - » en cada momento se ejecuta la tarea activa de mayor prioridad

Arquitectura síncrona



Arquitectura asíncrona



Planificación cíclica

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

  $T_M = \text{mcm}(T_i)$ (**ciclo principal**)
- El ciclo principal se divide en **ciclos secundarios**, con período T_S $(T_M = k \cdot T_S)$

En cada ciclo secundario se ejecutan las actividades correspondientes a determinadas tareas

Ejemplo 1

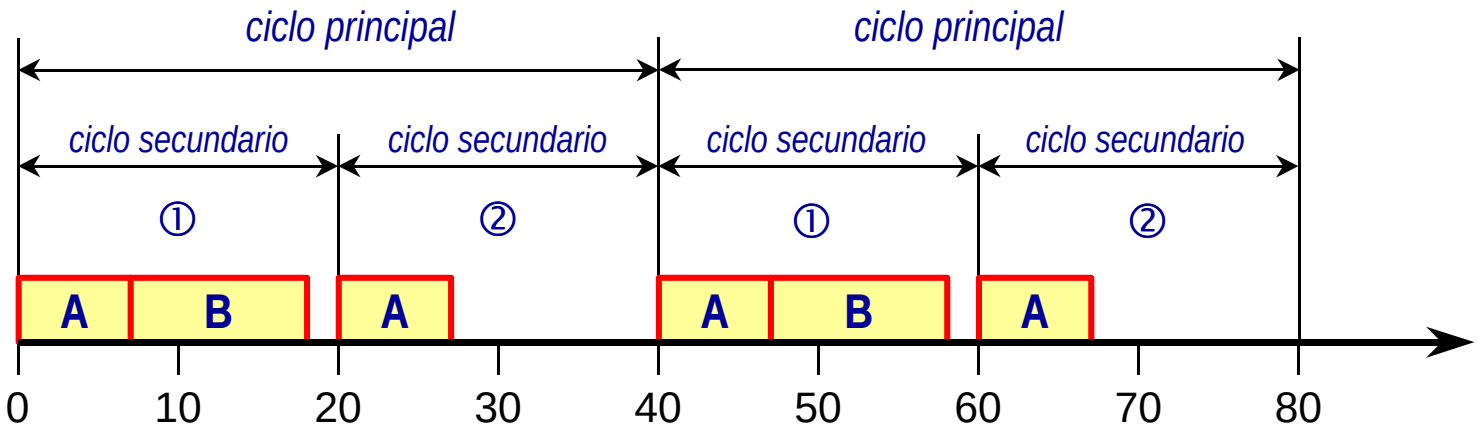
- Sistema con dos tareas periódicas

  A : T = 20 ms D = 20 ms

  B : T = 40 ms D = 40 ms

  T_M = 40 ms

  T_S = 20 ms



Ejecutivo cíclico

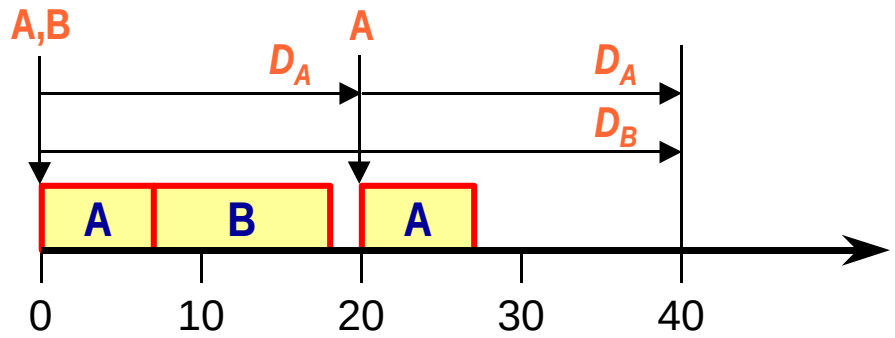
```
procedure Cyclic_Executive is
   type Frame is mod 2;
   Index :Frame := 0;
begin
   loop
      Wait_Clock_Interrupt; -- cada 20ms
      case Index is
         when 0 => A; B;
         when 1 => A;
      end case;
      Index := Index + 1;
   end loop;
end Cyclic_Executive;
```

Plazos de respuesta

- Se comprueba que se cumplen los plazos directamente sobre el plan de ejecución
- Para ello hace falta conocer el tiempo de cómputo de cada tarea

  Ejemplo:

A :	T = 20 ms	D = 20 ms	C = 8 ms
B :	T = 40 ms	D = 40 ms	C = 12 ms



Factor de utilización

- La cantidad $U = \sum_{i=1}^{N} \frac{C_i}{T_i}$

  se denomina **factor de utilización** del procesador
- Es una medida de la carga del procesador para un conjunto de tareas
- Para poder elaborar un plan de ejecución que garantice los plazos de todas las tareas, debe ser $U \leq 1$

Parámetros del plan cíclico

- Los períodos de los ciclos principal y secundario deben cumplir ciertas condiciones (Baker&Shaw, 1989)
  1. $T_S \geq \max C_i$
  2. $\exists i : \lfloor T_i / T_S \rfloor - T_i / T_S = 0$
  3. $\forall i : T_S + [T_S - \text{mcd}(T_S, T_i)] \leq D_i$
- La última condición permite detectar fallos de tiempo de respuesta
  - hay un ciclo secundario completo entre la activación y el límite del plazo de respuesta de una tarea
  - a veces es aceptable exigir únicamente

    $\forall i : T_S + [T_S - \text{mcd}(T_S, T_i)] \leq T_i$

Ejemplo 2

Tarea	C	T	D
T1	10	40	40
T2	18	50	50
T3	10	200	200
T4	20	200	200

$U = 0{,}76$

$T_M = 200$

1) $T_S \geq 20$

2) $T_S \in \{20,40,50,100,200\}$

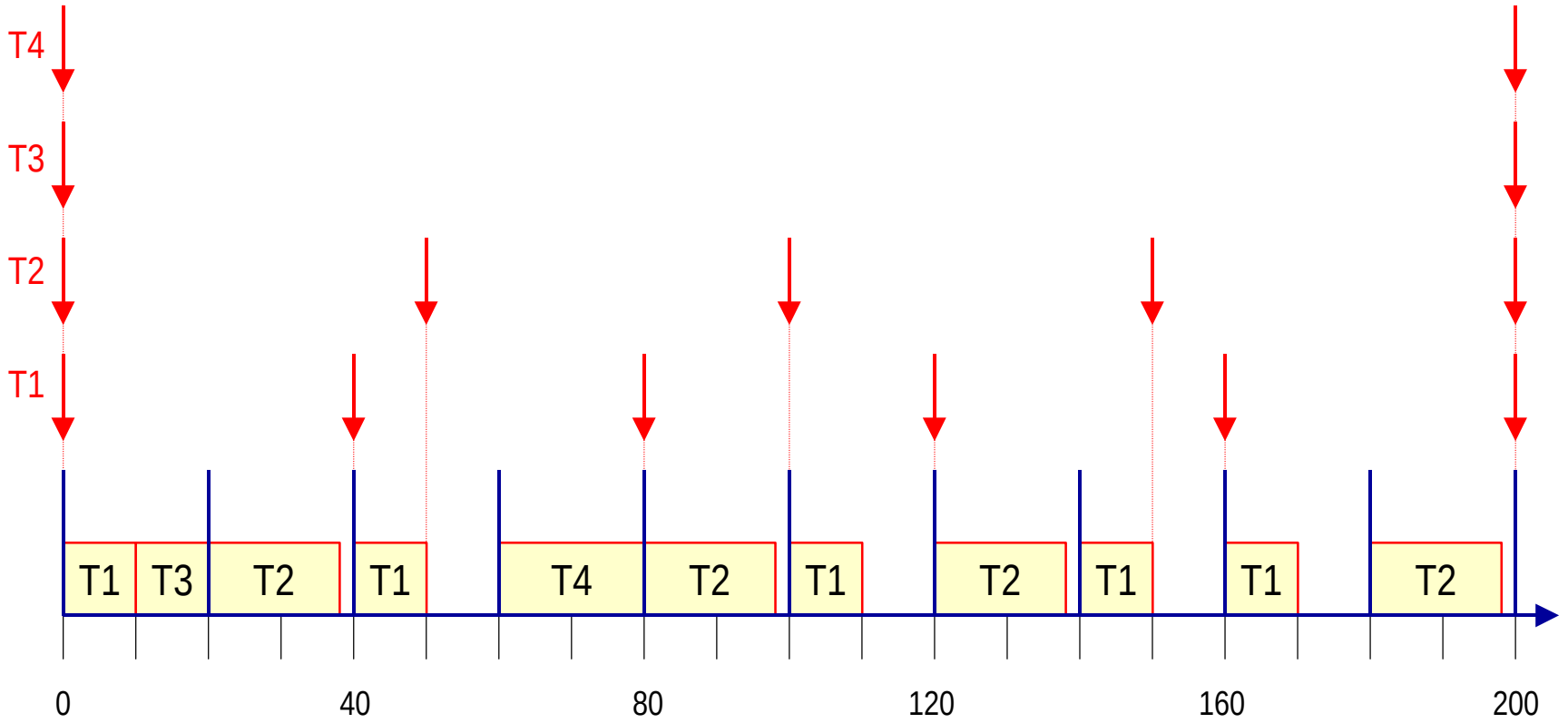
3) $2T_S - \text{mcd}(T_S,40) \leq 40$

$2T_S - \text{mcd}(T_S,50) \leq 50$

$2T_S - \text{mcd}(T_S,200) \leq 200$

$T_S = 20$

Ejemplo 2: plan cíclico



Tareas aperiódicas

- El ejecutivo cíclico sólo permite ejecutar tareas periódicas
- Las tareas esporádicas se ejecutan con un **servidor de consulta** *(polling server)*
- Es una tarea periódica que consulta si se ha producido el suceso esporádico o no
  - el período depende de la separación mínima entre eventos y del plazo de respuesta

Recursos compartidos

- Una tarea (o segmento) se ejecuta sin interrupción hasta que termina
- No es necesario proteger los recursos compartidos
  - la exclusión mutua es automática

Segmentación de tareas

- A veces no es posible confeccionar un plan cíclico que garantice los plazos
- Si $U \leq 1$, es posible planificar la ejecución *segmentando* una o más tareas
- Los segmentos son secuencias de instrucciones de la tarea con un tiempo de cómputo conocido

Ejemplo 3

Tarea	C	T	D
T1	40	80	80
T2	4	20	5
T3	10	40	40

$U = 0{,}95$

$T_M = 80$

$$
\begin{aligned}
&1) \quad T_s \geq 40 \\
&2) \quad T_s \in \{40,80\} \\
&3) \quad 2T_s - \text{mcd}(T_s,80) \leq 80 \\
&\qquad 2T_s - \text{mcd}(T_s,20) \leq 5 \\
&\qquad 2T_s - \text{mcd}(T_s,40) \leq 40
\end{aligned}
$$

- Ningún valor cumple (3)
- No hay solución aceptable

Ejemplo 3: segmentación

Tarea	C	T	D
T1.1	6	80	80
T1.2	16	80	80
T1.3	6	80	80
T1.4	12	80	80
T2	4	20	5
T3	10	40	40

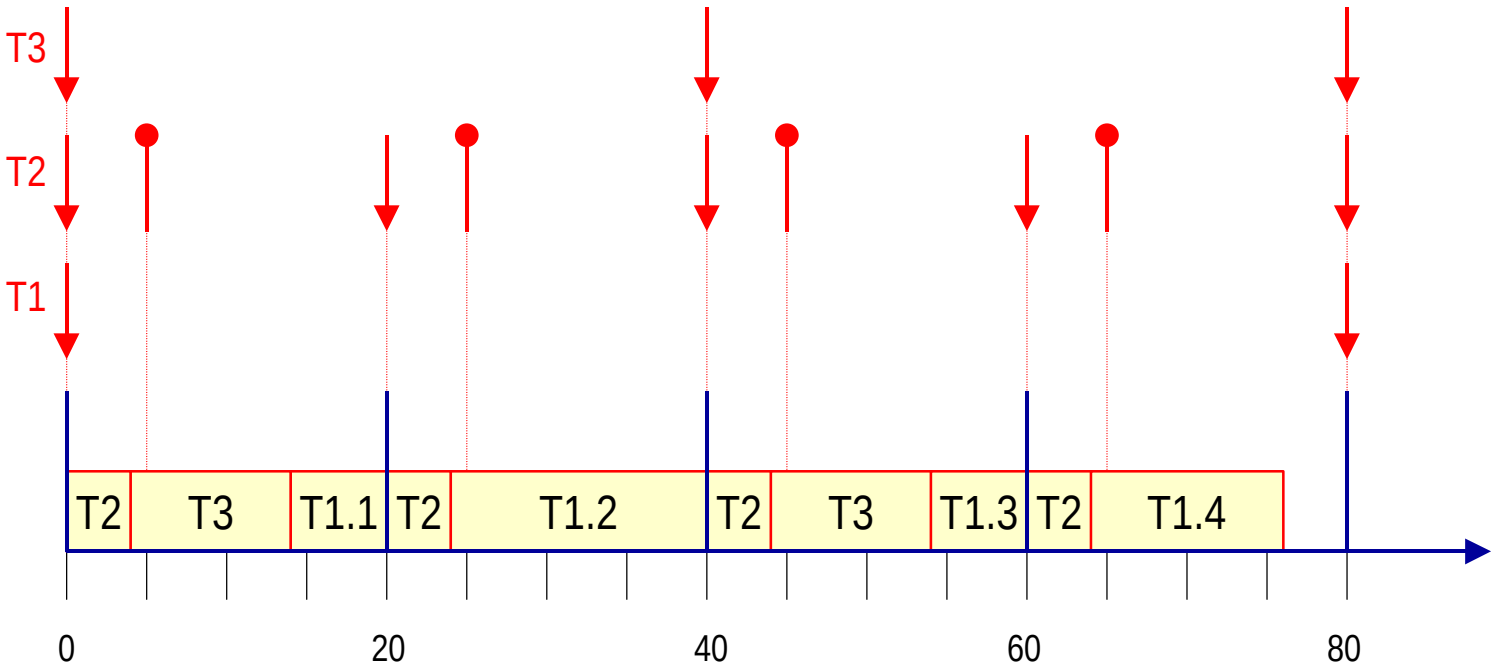
$$
\begin{aligned}
&1) \quad T_S \geq 16 \\
&2) \quad T_S \in \{20,40,80\} \\
&3) \quad 2T_S - \text{mcd}(T_S,80) \leq 80 \\
&\qquad 2T_S - \text{mcd}(T_S,20) \leq 5 \\
&\qquad 2T_S - \text{mcd}(T_S,40) \leq 40
\end{aligned}
$$

$U = 0{,}95$

$T_M = 80$

- Ningún valor cumple (3)
- Hay una solución aceptable con $T_S = 20$

Ejemplo 3: plan de ejecución



Ejemplo 3: ejecutivo cíclico

```
procedure Cyclic_Executive is
   type Frame is mod 4;
   Index :Frame := 0;
begin
   loop
      Wait_clock_Interrupt; -- cada 20ms
      case Index is
         when 0 => T2; T3; T1_1;
         when 1 => T2;     T1_2;
         when 2 => T2; T3; T1_3;
         when 3 => T2;     T1_4;
      end case;
   end loop;
end Cyclic_Executive;
```

Problemas de la segmentación

- A veces es difícil ajustar el tiempo de cómputo de los segmentos
- Si hay recursos compartidos, cada sección crítica debe estar incluida en un solo segmento
- Si se modifica una sola tarea hay que rehacer la planificación completa
  - y posiblemente volver a segmentar de otra manera

Construcción del plan cíclico

- En general, el problema es NP-duro
  - no hay algoritmos eficientes que resuelvan todos los casos
- Se usan algoritmos heurísticos
  - se construye un árbol de soluciones parciales
  - se puede empezar colocando las tareas más urgentes
  - se podan las ramas según algún criterio heurístico
- Es más fácil cuando el sistema es armónico
  - pero esto puede forzar una mayor utilización del procesador
- Cuando los períodos son muy dispares es más difícil
  - muchos ciclos secundarios en cada ciclo principal

Conclusiones

- Los sistemas cíclicos, con arquitectura síncrona, tienen muchas ventajas
  - implementación sencilla y robusta
  - determinismo temporal
  - es posible certificar que son seguros
- Pero tienen inconvenientes importantes
  - mantenimiento difícil y costoso
    - » si se cambia algo hay que empezar desde el principio
    - » la segmentación añade mucha complejidad
  - es difícil incluir tareas esporádicas
- En general, es un método de bajo nivel

Bibliografía

- Burns & Wellings,
  Real-Time Systems and Programming Languages,
  capítulos 7 y 13
- Liu, *Real-Time Systems,* capítulo 5
- Kopetz, *Real-Time Systems*, capítulo 11