dit
UPM

Planificación de tareas de tiempo real

Juan Antonio de la Puente
DIT/UPM

Objetivos

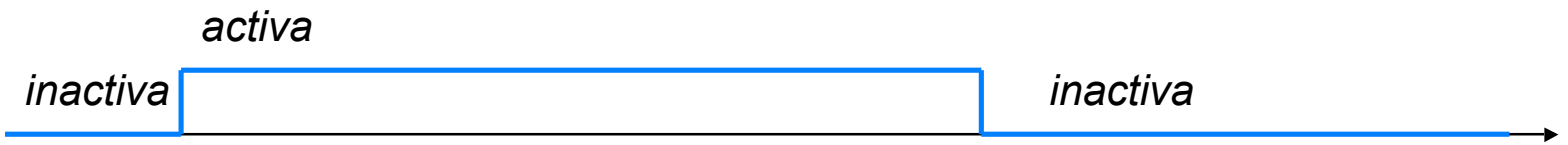
- Plantear los problemas básicos relacionados con el cumplimiento de los requisitos temporales
- Conocer los principales métodos de planificación de tareas y sus características temporales
- Evaluar las ventajas e inconvenientes de los diversos métodos de planificación de tareas

Tareas y requisitos temporales

Tareas de tiempo real

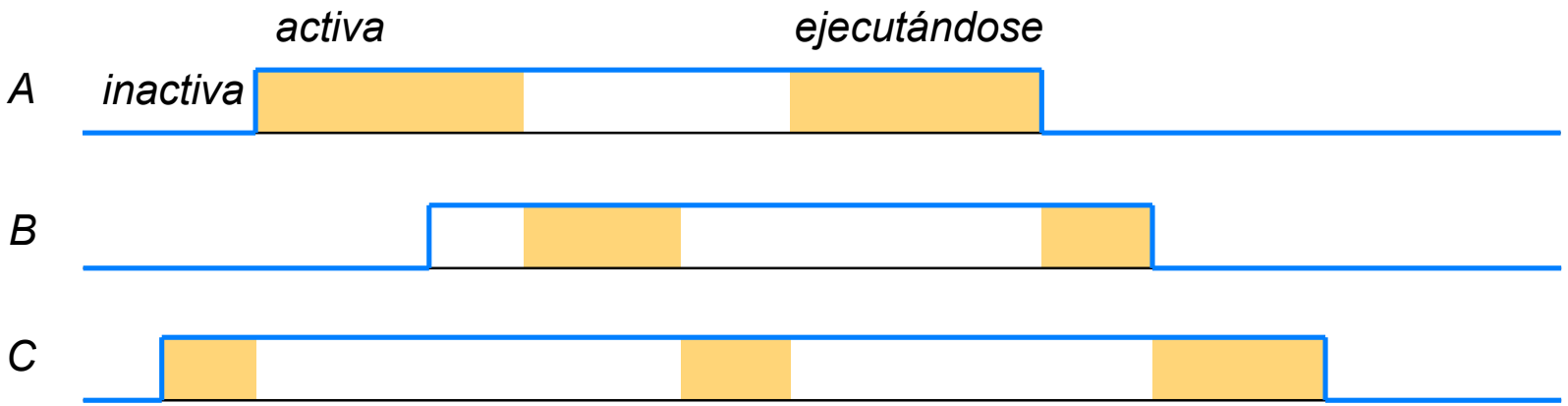
- En un sistema de tiempo real se ejecutan una o más *tareas*

- Cada tarea ejecuta una *actividad* de forma repetida
  - cada vez que ejecuta la actividad se produce un *ciclo de ejecución*
  - durante el ciclo de ejecución la tarea permanece *activa*
  - cuando termina la actividad pasa a estar *inactiva* en espera de que comience el siguiente ciclo de ejecución



Concurrencia

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

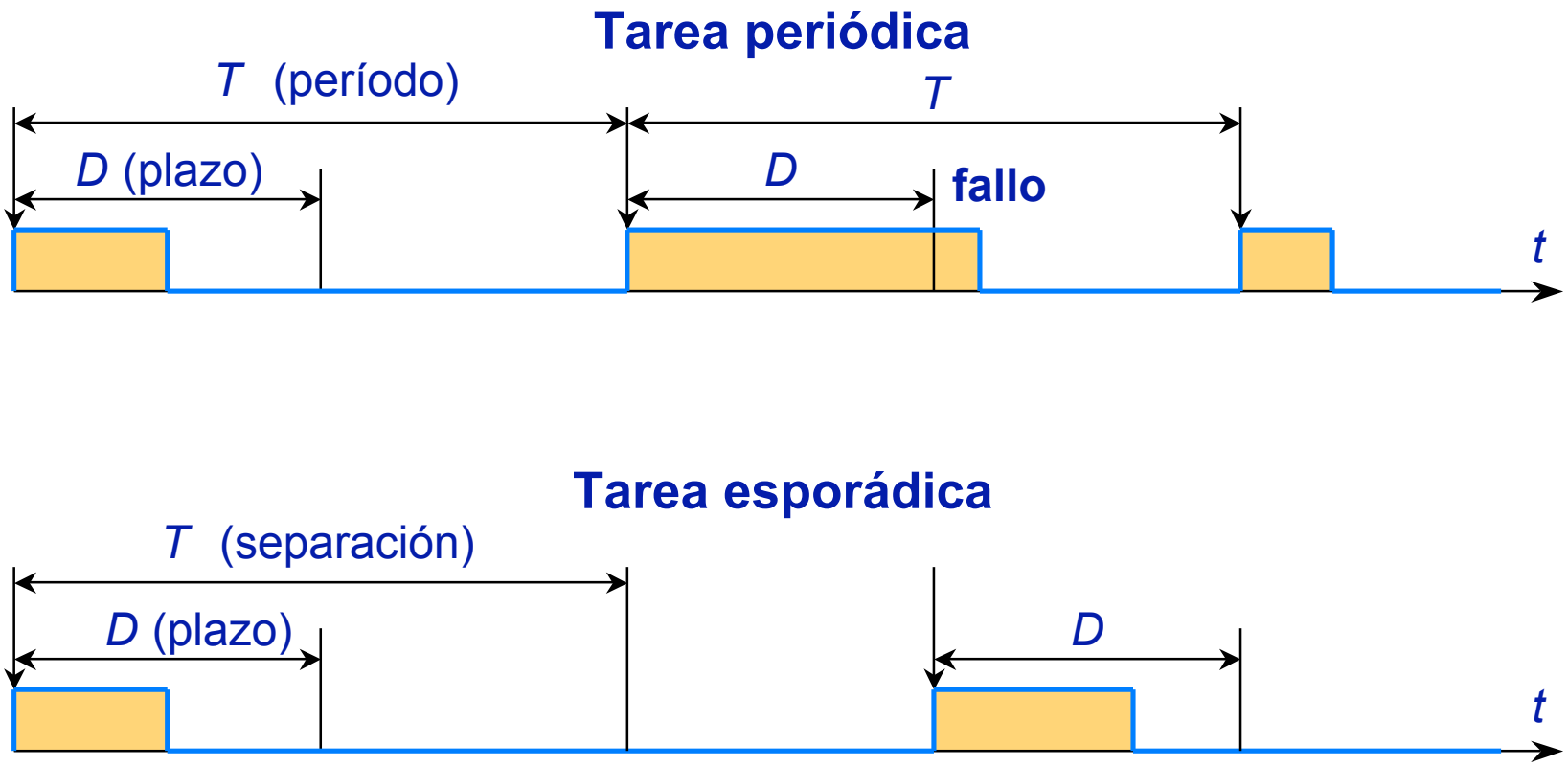


Requisitos temporales

Los requisitos de tiempo real se refieren a

- El *principio* del ciclo 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



Planificación de tareas

Planificación de tareas

- Se trata de repartir el tiempo de procesador entre varias tareas ***de forma que se satisfagan los requisitos temporales***

- La relación biunívoca entre acciones y procesadores es un *plan de ejecución (schedule)*

- El componente del sistema que hace esto es el *planificador (scheduler)*
  - para ello utiliza un *algoritmo de planificación*

Esquemas de planificación

- Planificación dirigida por tiempo *(time/clock-driven)*
  - el planificador se ejecuta cada vez que llega una señal de reloj
    - » ejemplo: planificación cíclica
- Planificación por turno circular *(round-robin)*
  - las acciones listas para ejecutarse se agrupan en una cola FIFO
  - cada acción se ejecuta durante una rodaja de tiempo y después se pone al final de la cola
    - » variante: rodajas de tiempo desiguales (ponderadas)
- Planificación por prioridades
  - cada acción tiene una prioridad
  - se ejecuta siempre la acción de mayor prioridad entre las listas
    - » la planificación está dirigida por sucesos *(event-driven)*

Planificación con y sin desalojo

- Planificación con desalojo (preemptive scheduling)
  - se puede desalojar del procesador una acción que se está ejecutando para dar paso a otra
    - » se usa normalmente con prioridades
- Planificación sin desalojo (non-preemptive scheduling)
  - una acción que ha comenzado a ejecutarse sólo deja el procesador si
    - » termina su ejecución
    - » necesita un recurso que no está disponible
    - » abandona el procesador voluntariamente

Prioridades fijas y variables

- Planificación con prioridades fijas
  - la prioridad de las acciones de una misma tarea es siempre la misma
    - » puede variar si hay cambios de modo
  - ejemplo: prioridades monótonas en frecuencia *(rate-monotonic scheduling)*
    - » mayor prioridad a la tarea con período más corto

- Planificación con prioridades variables (dinámicas)
  - la prioridad de una acción se decide en el momento de ejecutarla
  - ejemplo: primero el más urgente *(earliest deadline first)*
    - » mayor prioridad a la acción que deba terminar antes

Modelos de tareas

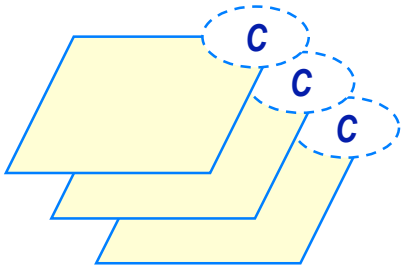
- Un modelo de tareas especifica las características de las tareas de un sistema de tiempo real
  - se restringen para poder analizar el sistema y garantizar los requisitos temporales
- Ejemplos:
  - sólo tareas periódicas independientes
  - tareas periódicas y esporádicas independientes
  - tareas con comunicación y sincronización
  - tareas estáticas o dinámicas
- Empezamos con modelos sencillos
  - tareas periódicas independientes

Planificación estática

Modelo de tareas cíclico

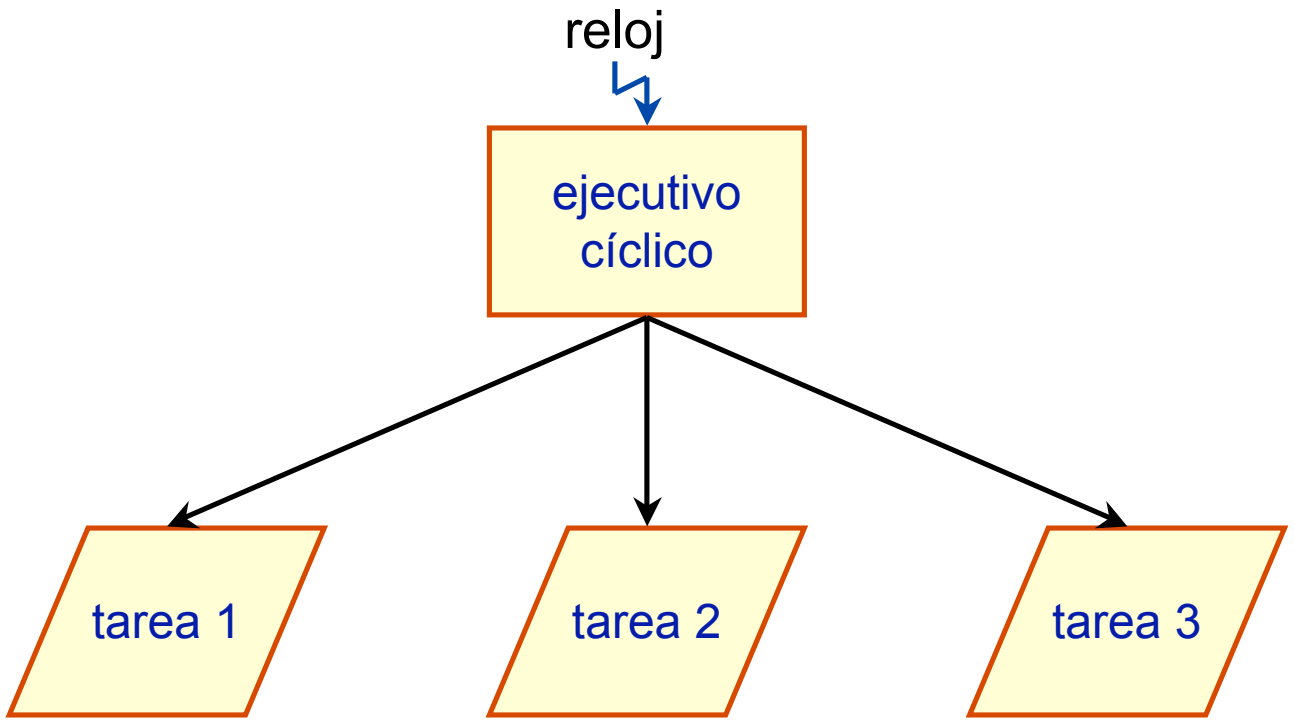
- Hay muchos sistemas de tiempo real que sólo tienen tareas periódicas
  - son más fáciles de construir
  - su comportamiento está completamente determinado

- Inicialmente consideramos que no hay comunicación entre tareas (tareas independientes)



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**



Parámetros temporales

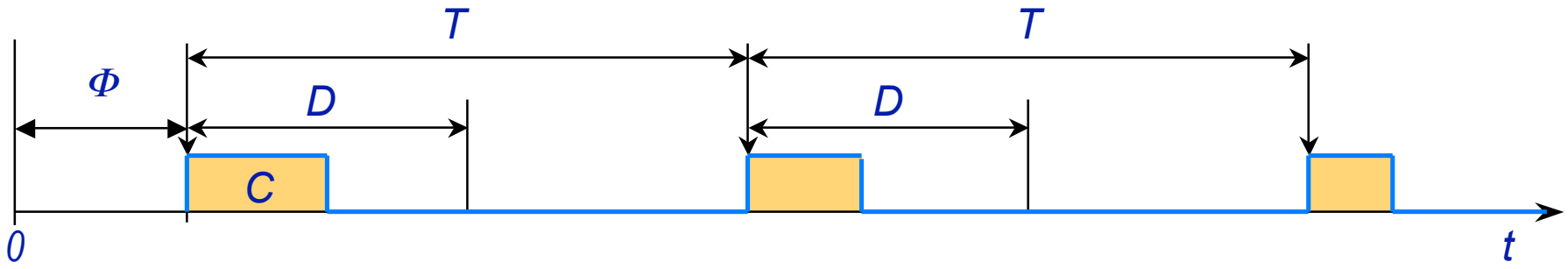
Una tarea periódica se define por su parámetros (Φ,T,C,D)

Φ es la *fase*

T es el *período* de activación de la tarea

C es su *tiempo de cómputo* en el peor caso

D es el *plazo de respuesta* relativo a la activación

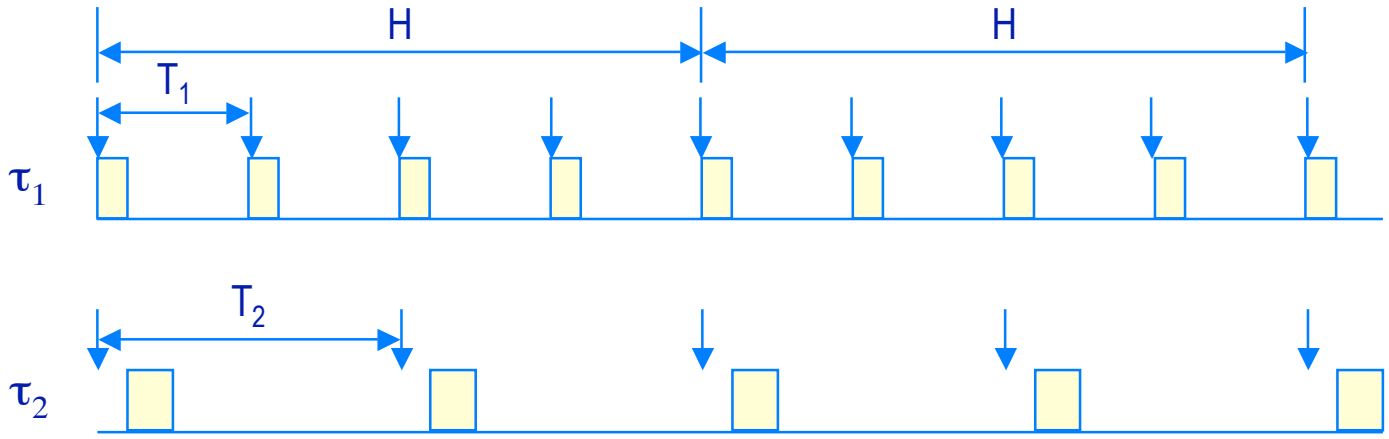


Hiperperíodo

- En un sistema formado únicamente por tareas periódicas con períodos T_i, $i = 1..N$, el comportamiento global se repite con un período

  $H = \text{mcm}\ (T_i)$

  H es el **hiperperíodo** del sistema



Planificación estática

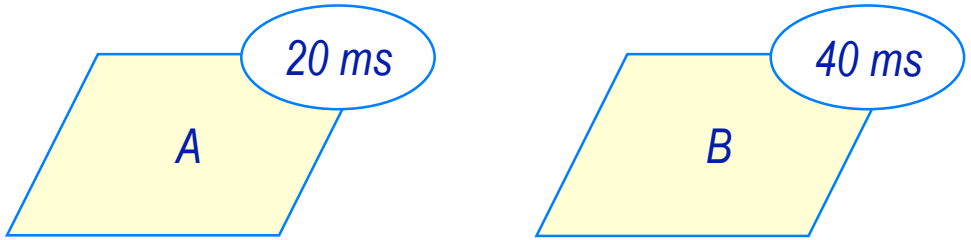
- Si todas las tareas son periódicas, se puede confeccionar un *plan de ejecución* fijo
- Se trata de un esquema que se repite cada

$$T_M = \text{mcm}(T_i)$$ *(ciclo principal)*

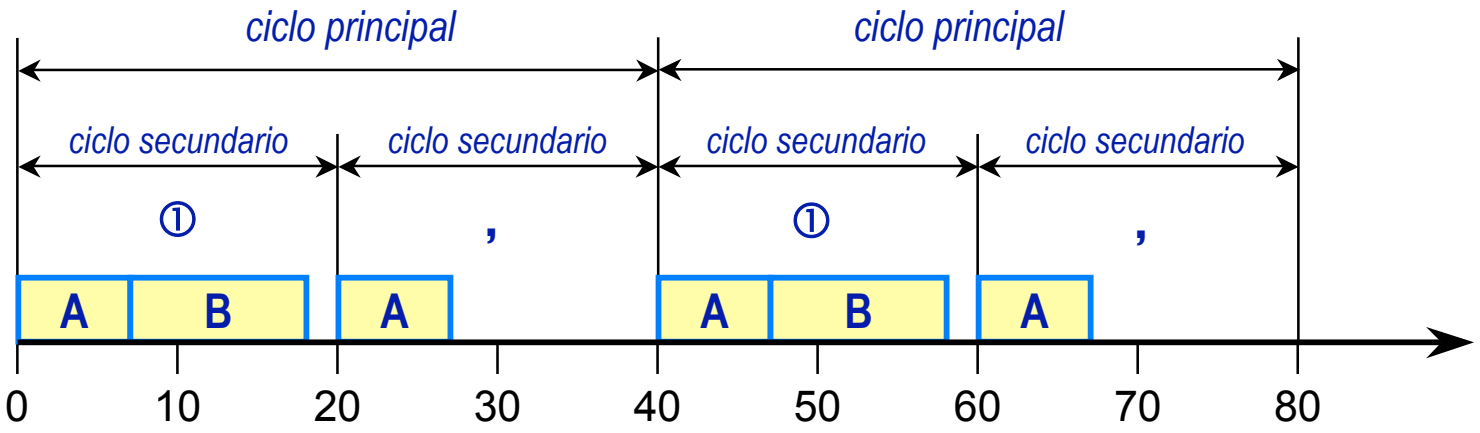
  - » el período del ciclo principal es igual al hiperperíodo del sistema

- 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



Plan cíclico: T_M = 40 ms; T_S = 20 ms



Ejecutivo cíclico

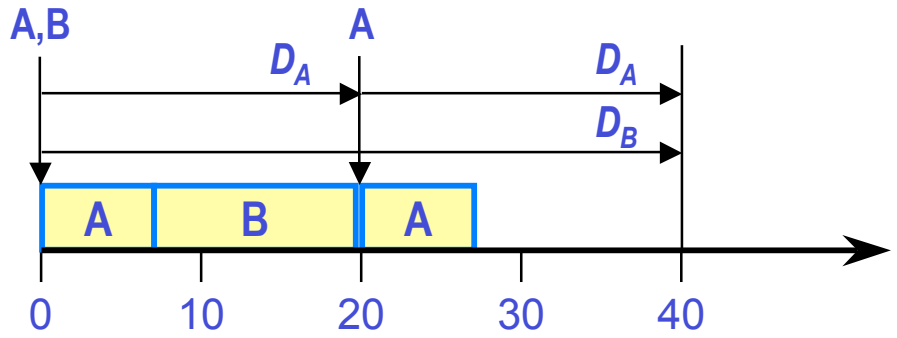
```
procedure Cyclic_Executive is
   type Frame is mod 2;
   Index :Frame := 0;
begin
   Set_Timer (Periodic, 0.020);
   loop
      Wait_Clock_Interrupt; -- cada 20ms
      case Index is
         when 0 => A; B;
         when 1 => A;
      end case;
      Index := Index + 1;
   end loop;
end Cyclic_Executive;
```

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

$$(1) \qquad T_s \geq \max C_i$$

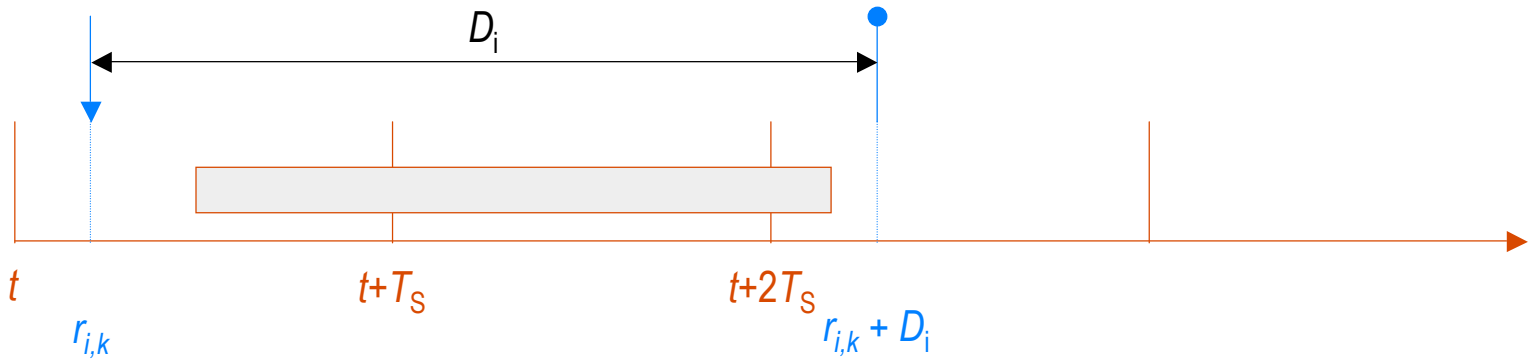
$$(2) \quad \exists i: \quad T_i / T_S - \lfloor T_i / T_s \rfloor = 0$$

$$(3) \quad \forall i: \quad 2T_S - \text{mcd}(T_S, T_i) \leq D_i$$

El período secundario debe cumplir ciertas condiciones (Baker&Shaw, 1989):

1. Todas las acciones deben caber en un marco
2. El período secundario divide al menos al período de una tarea (y por tanto al período principal)
3. Entre el instante de activación y el tiempo límite de cada acción debe haber al menos un marco completo (para poder comprobar si la acción termina a tiempo)
   - si la acción se activa al comienzo de un marco, basta que $T_S \leq D_i$

Condición (3)



$$t + 2T_s \le r_{i,k} + D_i$$

$$2T_S - (r_{i,j} - t) \le D_i$$

$$r_{i,j} - t \ge \text{mcd}\,(T_i, T_S)$$

$$2T_S - \text{mcd}\,(T_i, T_S) \le D_i$$

Si $r_{i,k} = t$, basta con que sea

$T_S \le D_i$

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) $T_s \in \{20,25,40,50,100,200\}$

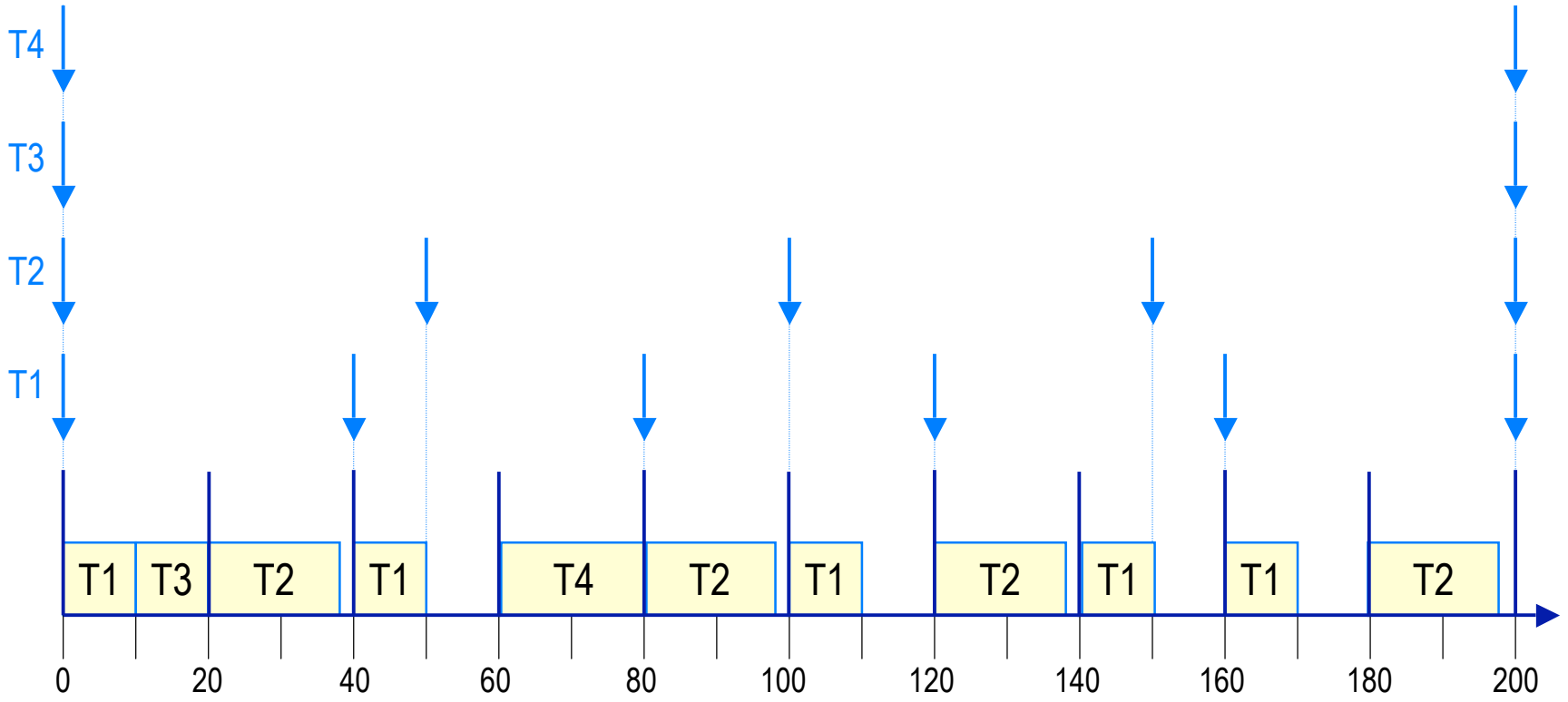
3) $2T_s - \text{mcd}(T_s,40) \leq 40$

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

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

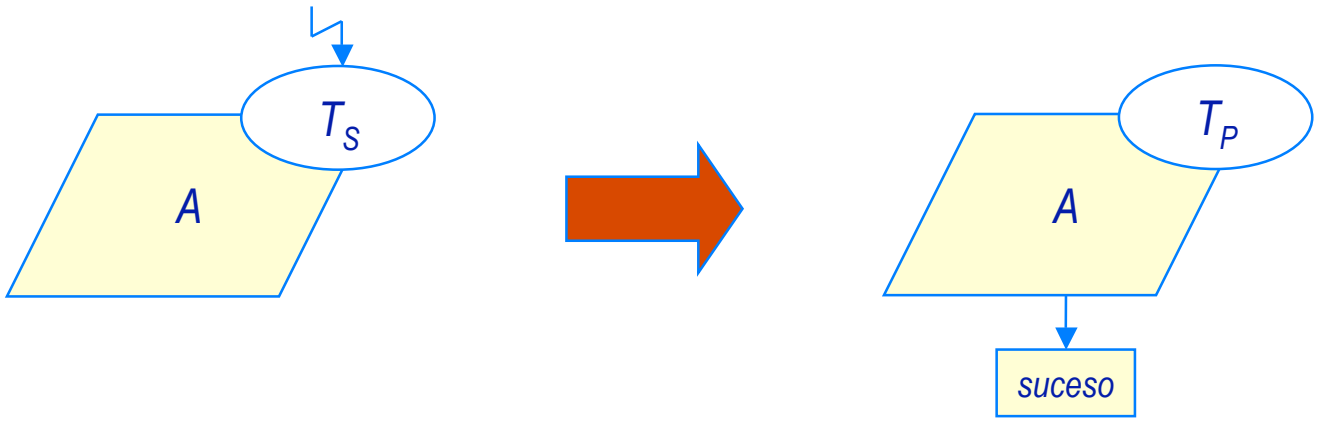
$T_S = 20$

Ejemplo 2: plan cíclico



Tareas esporádicas

- 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

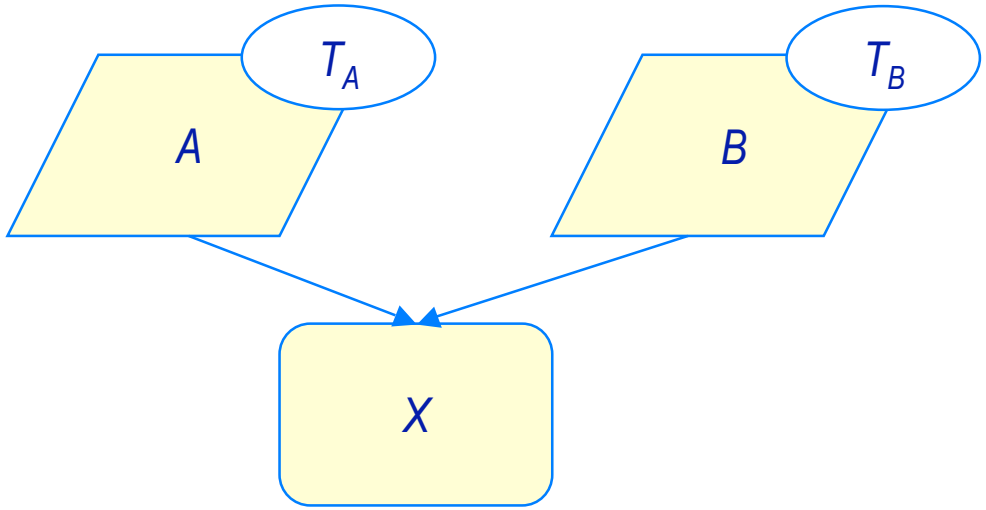


Ejemplo de servidor de consulta

```
procedure Polling_Server is
   Event_Occurred : Boolean := False;
begin
   -- invocado periódicamente por el ejecutivo cíclico
   Check (Event_Occurred);
  if Event_Occurred then
     Sporadic_Activity;
   end if;
end Polling_Server;
```

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
τ_1	10	40	40
τ_2	20	100	100
τ_3	50	200	200

$$1)\quad T_S \geq 50$$
$$2)\quad T_S \in \{50,100,200\}$$
$$3)\quad 2T_S - \text{mcd}(T_S,40) \leq 40$$
$$2T_S - \text{mcd}(T_S,100) \leq 100$$
$$2T_S - \text{mcd}(T_S,200) \leq 200$$

$U = 0{,}95$

$T_M = 200$

- Ningún valor cumple (1) y (3)
- No hay solución aceptable

Ejemplo 3 — segmentación

Tarea	C	T	D
τ1	10	40	40
τ2	20	100	100
τ3.1	10	200	200
τ3.2	30	200	200
τ3.3	10	200	200

(1) $T_s \geq 30$

(2) $T_s \in \{40,50,100,200\}$

(3) $2T_s - mcd(T_s,40) \leq 40$

$2T_s - mcd(T_s,100) \leq 100$

$2T_s - mcd(T_s,200) \leq 200$

T_S = 40 cumple todas las condiciones

Ejemplo 3: ejecutivo cíclico

```
procedure Cyclic_Executive is
   type Frame is mod 5;
   Index :Frame := 0;
begin
    loop
       Wait_clock_Interrupt; -- cada 40ms
      case Index is
         when 0 => T1; T2; T3_1;
         when 1 => T1;     T3_2;
         when 2 => T1;     T3_3;
         when 3 => T1; T2;
          when 4 => T1;
      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

- Tres tipos de decisiones interdependientes:
  - ajustar el tamaño de los marcos
  - segmentar acciones
  - colocar los segmentos en marcos
- 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
  - sólo es apropiado para sistemas que no se modifican unavez construidos