Sistemas Concurrentes y Distribuidos.
Tema 4. Introducción a los sistemas de tiempo real.

Dpt. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
Universidad de Granada

Curso 16-17

Índice

Sistemas Concurrentes y Distribuidos.
Tema 4. Introducción a los sistemas de tiempo real.

Sección 1
Concepto de sistema de tiempo real. Medidas de tiempo y modelo de tareas.

Subsección 1.1
Definición, tipos y ejemplos

Sistemas de Tiempo Real

- Constituyen un tipo de sistema en el que la ejecución del sistema se debe producir dentro de unos plazos de tiempo predefinidos para que funcione con la suficiente garantía.
- En un sistema además concurrente será necesario que todos los procesos sobre un procesador o sobre varios se ejecuten en los plazos de tiempo predefinidos.

Definición de un Sistema de Tiempo Real

Stankovic (1997) da la siguiente definición:

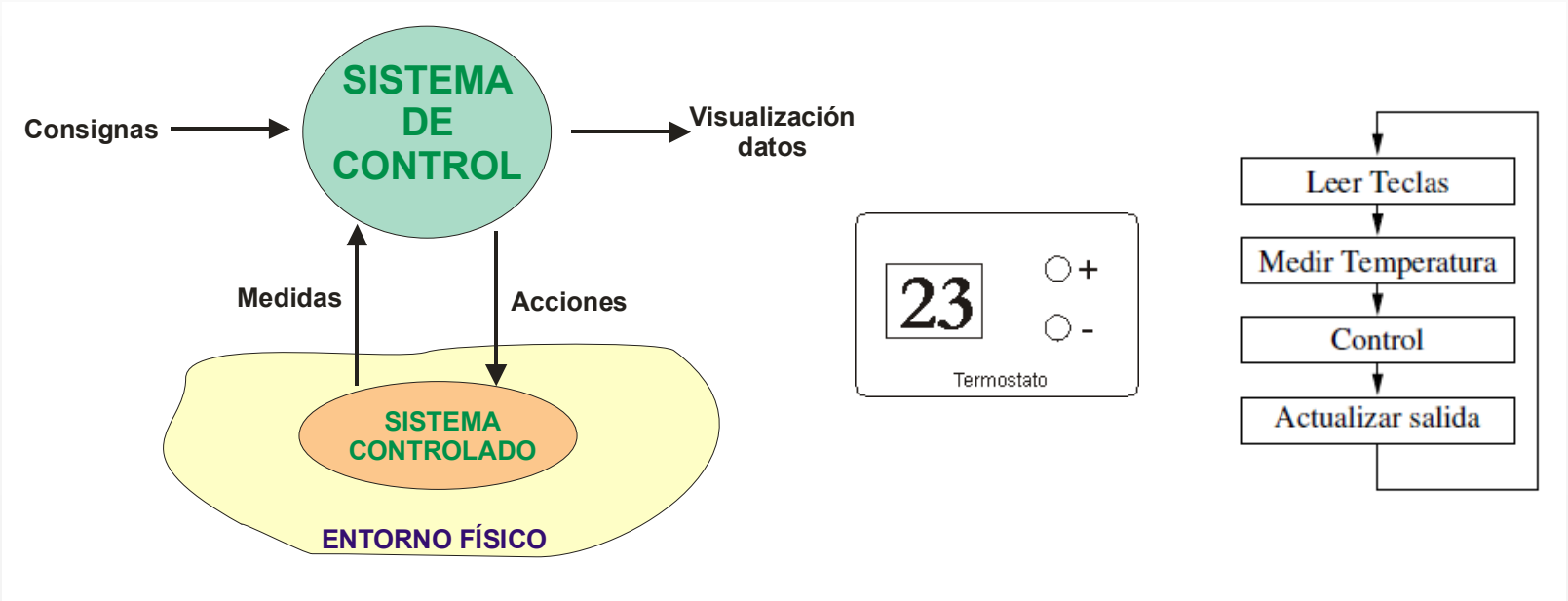
> *Un sistema de tiempo real es aquel sistema cuyo funcionamiento correcto depende no sólo de resultados lógicos producidos por el mismo, sino también del instante de tiempo en el que se producen esos resultados*

Corrección Funcional + Corrección Temporal

El no cumplimiento de una restricción temporal lleva a un **fallo** del sistema

Ejemplo: Sistema de Control

- Objetivo: Ejecutar el lazo de control del sistema en instantes de tiempo prefijados.
- Condición de tiempo real: no puede haber retrasos en la ejecución del lazo de control, ya que afecta al rendimiento y provoca perdida de estabilidad.



Tipos de Sistemas de Tiempo Real

Habitualmente suele asociarse la denominación de sistemas de tiempo real a los siguientes tipos de sistemas:

- **Sistema "en-línea":** Siempre está disponible, pero no se garantiza una respuesta en intervalo de tiempo acotado.
  - Ejemplos: Cajeros automáticos, sistemas de reservas de vuelo.
- **Sistema interactivo:** El sistema ofrece una respuesta en un tiempo, aunque no importa el tiempo que necesita para su ejecución.
  - Ejemplos: Reproductor DVD, sistema aire acondicionado, juegos, ..
- **Sistema de respuesta rápida:** El sistema ofrece una respuesta en el menor tiempo posible y de forma inmediata.
  - Ejemplos: Sistema anti incendios, alarmas, etc.

Tipos de Sistemas de Tiempo Real

Ejemplos de Sistemas de Tiempo Real

En la actualidad hay muchos ejemplos de uso de Sistemas de Tiempo Real, podemos citar algunos de ellos:

- **Automoción:** sistema de ignición, transmisión, dirección asistida, frenos antibloqueo (ABS), control de la tracción, ...
- **Electrodomésticos:** televisores, lavadoras, hornos de microondas, reproductores de videos o DVDs, sistemas de seguridad y vigilancia, ...
- **Aplicaciones aeroespaciales:** control de vuelos, controladores de motores, pilotos automáticos, ...
- **Equipamiento médico:** como sistemas de monitorización de anestesia, monitores ECG,
- **Sistemas de defensa:** como sistemas radar de aviones, sistemas de radio, control de misiles, ...

Subsección 1.2
Propiedades de los Sistemas de Tiempo Real

Propiedades de un STR

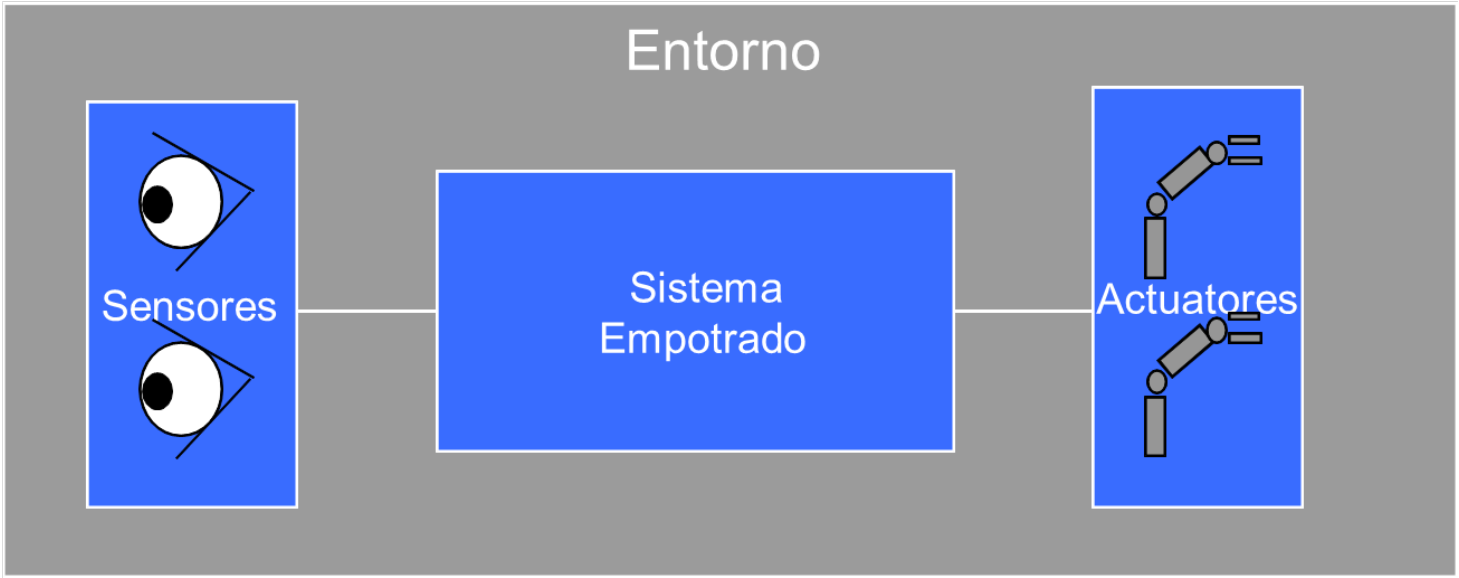
Al depender la corrección del sistema de las restricciones temporales, los sistemas de tiempo real tienen que cumplir una serie de propiedades:

- **Reactividad.**
- **Predecibilidad.**
- **Confiabilidad.**

a continuación veremos cada una de estas propiedades con más detalle.

Propiedad: Reactividad

El sistema tiene que interaccionar con el entorno, y responder de la manera esperada a los estímulos externos dentro de un intervalo de tiempo previamente definido.



Propiedad: Predecibilidad

Tener un comportamiento predecible implica que la ejecución del sistema tiene que ser determinista, y por lo tanto, se debe garantizar su ejecución dentro del plazo de tiempo definido.

- Las respuestas han de producirse dentro de las restricciones temporales impuestas por el entorno (sistema controlado), y que suele ser diferente para cada proceso del sistema.
- Es necesario conocer el comportamiento temporal de los componentes software (SO, middleware, librería, etc) y hardware utilizados, así como del lenguaje de programación.
- Si no se puede tener un conocimiento temporal exacto, hay que definir marcos de tiempo acotados; p.e. conocer el tiempo de peor ejecución de un algoritmo, el tiempo máximo de acceso a un dato de E/S)

Propiedad: Confiabilidad

- La **Confiabilidad** (***Dependability***) mide el grado de confianza que se tiene del sistema. Depende de:
  - Disponibilidad (*avalaibility*). Capacidad de proporcionar servicios siempre que se solicita.
  - Robustez o tolerancia a fallos (*fault tolerant*). Capacidad de operar en situaciones excepcionales sin poseer un comportamiento catastrófico.
  - Fiabilidad (*reliability*). Capacidad de ofrecer siempre los mismos resultados bajo las mismas condiciones.
  - Seguridad: Capacidad de protegerse ante ataques o fallos accidentales o deliberados (*safety*), y a la no vulnerabilidad de los datos (*security*).
- Cuando esta propiedad es crítica (su no cumplimiento lleva a pérdida humana o económica), el sistema se denomina sistema de tiempo real **crítico** (***safety-critical system***)
  - Ejemplos: Sistemas de Aviónica, Sistemas de automoción, Centrales nucleares, etc.

Tipos de STRs (otra clasif. distinta)

Tipo	Características	Ejemplos
No permisivos o estrictos (*hard*)	Plazo de respuesta dentro del límite preestablecido.	Control de vuelo de una aeronave. Sistema automático de frenado ABS.
Permisivos, flexibles o no estrictos (*soft*)	Aunque el plazo de respuesta es importante, el sistema funciona correctamente aunque no respondan en el plazo de tiempo fijado	Sistema de adquisición de datos meteorológicos.
Firmes	Se permiten algunos fallos en el plazo de respuesta, pero un número excesivo provoca una fallo completo del sistema	Sistema de control de reserva de vuelos. Sistema de video bajo demanda.

Los sistemas de tiempo real pueden tener componentes de las tres clases.

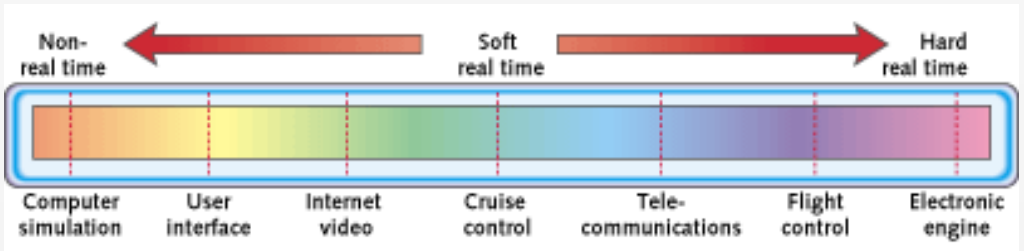


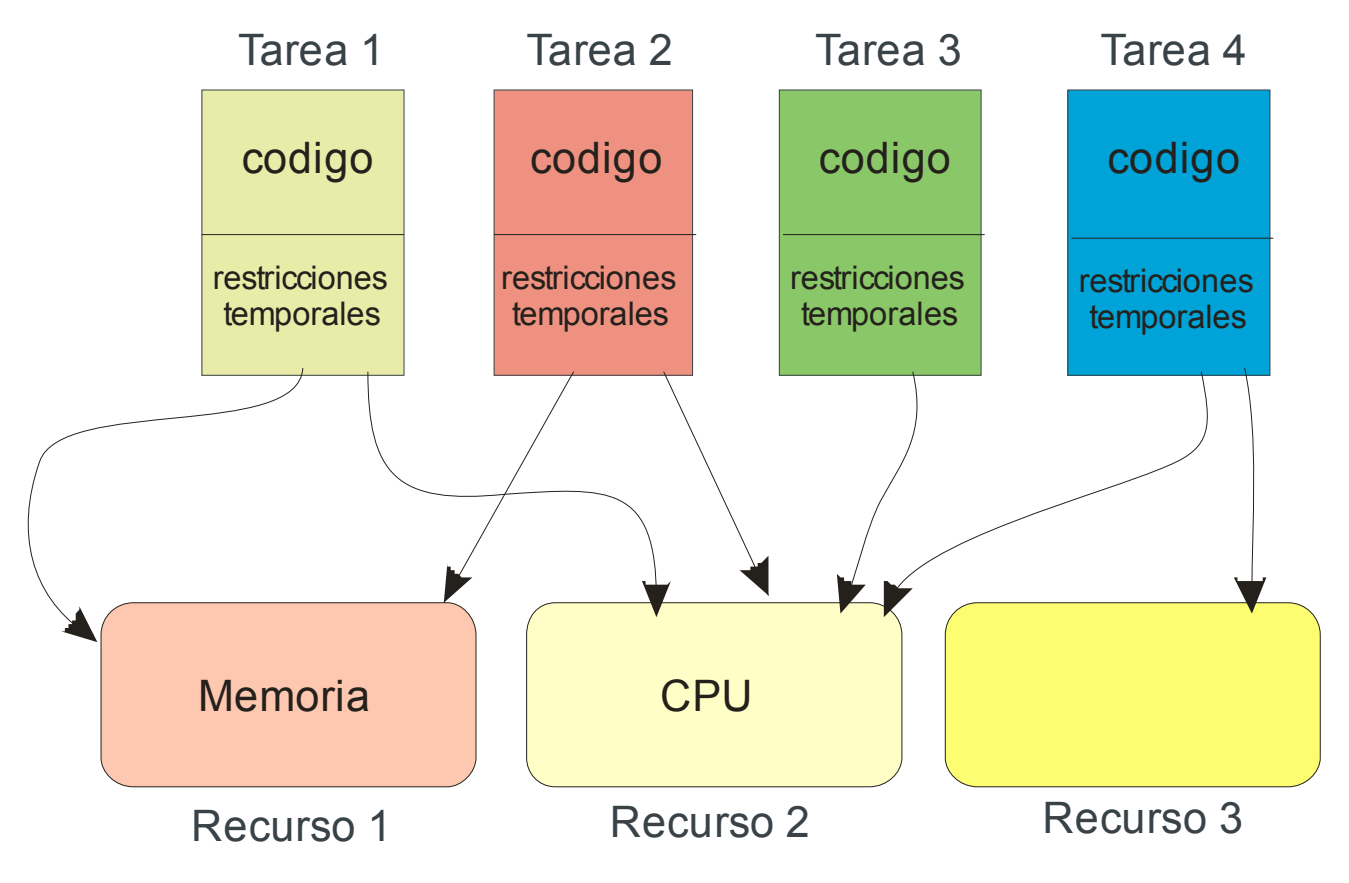
Imagen obtenida de:

http://www.embedded.com/electronics-blogs/beginner-s-corner/4023859/Introduction-to-Real-Time

Subsección 1.3
Modelo de Tareas

Modelo de Tareas

- Un sistema de tiempo real se estructura en **tareas** que acceden a los recursos del sistema.



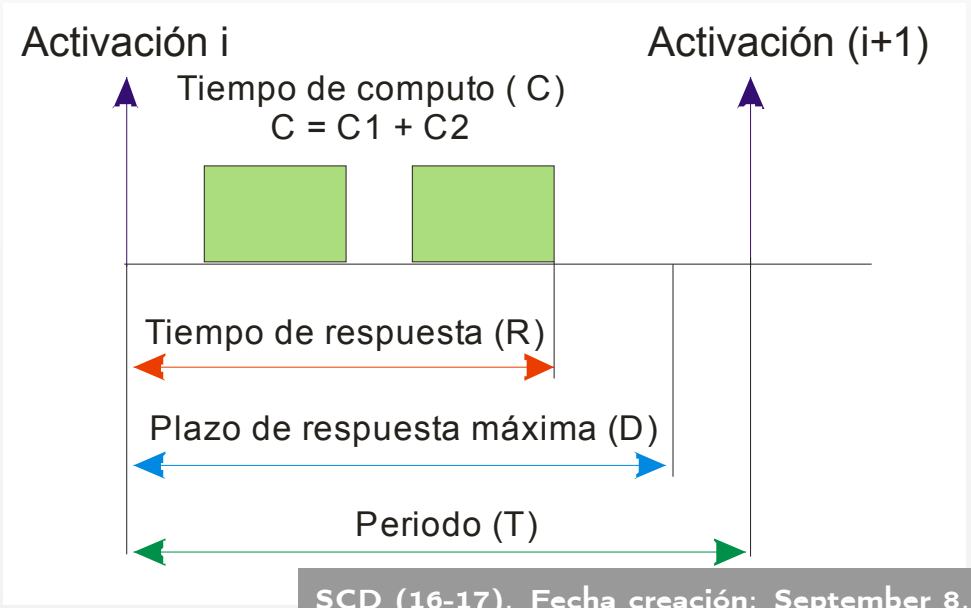
Tareas y recursos. Definición.

En los STR cabe distinguir estos dos tipos de elementos:

- **Tarea:** El conjunto de acciones que describe el comportamiento del sistema o parte de él en base a la ejecución secuencial de una sección de código (o programa). Equivalente al *proceso* o *hebra*.
  - La tarea satisface una necesidad funcional concreta.
  - La tarea tiene definida unas restricciones temporales a partir de los **Atributos Temporales**.
  - Una **tarea activada** es una tarea en ejecución o pendiente de ejecutar.
  - Decimos que una tarea *se activa* cuando se hace necesario ejecutarla una vez.
  - El **instante de activación** de una tarea es el instante de tiempo a partir del cual debe ejecutarse (cuando pasa de desactivada a activada).
- **Recursos:** Elementos disponibles para la ejecución de las tareas.
  - Recursos Activos: Procesador, Red, ...
  - Recursos Pasivos: Datos, Memoria, Dispositivos de E/S, ...

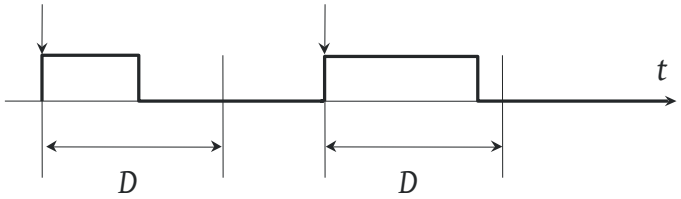
Atributos temporales

- **Tiempo de computo o de ejecución** (C): Tiempo necesario para la ejecución de la tarea.
- **Tiempo de respuesta** (R): Tiempo que ha necesitado el proceso para completarse totalmente a partir del instante de activación.
- **Plazo de respuesta máxima** (*deadline*) (D): Define el máximo tiempo de respuesta posible.
- **Periodo** (T): Intervalo de tiempo entre dos activaciones sucesivas en el caso de una tarea periódica.

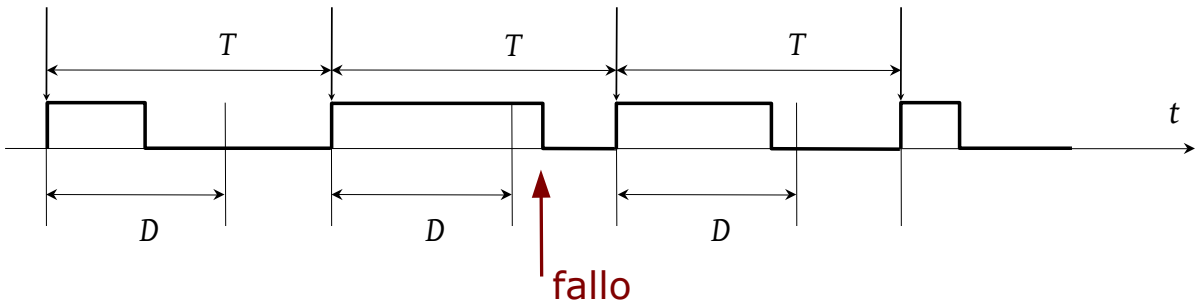


Tipos de Tareas

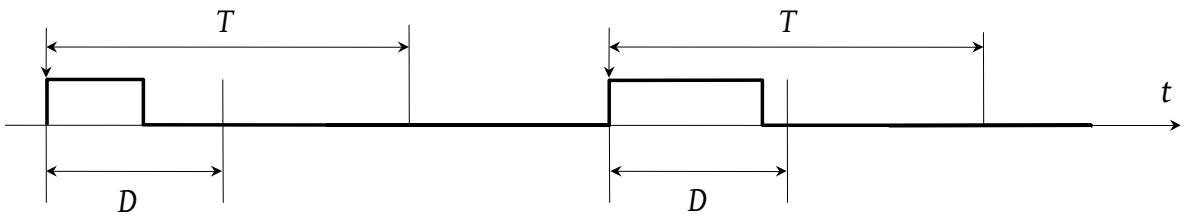
- **Aperiódica:** activación en instantes arbitrarios.



- **Periódica:** repetitiva, T es el **período** (tiempo exacto entre act.)

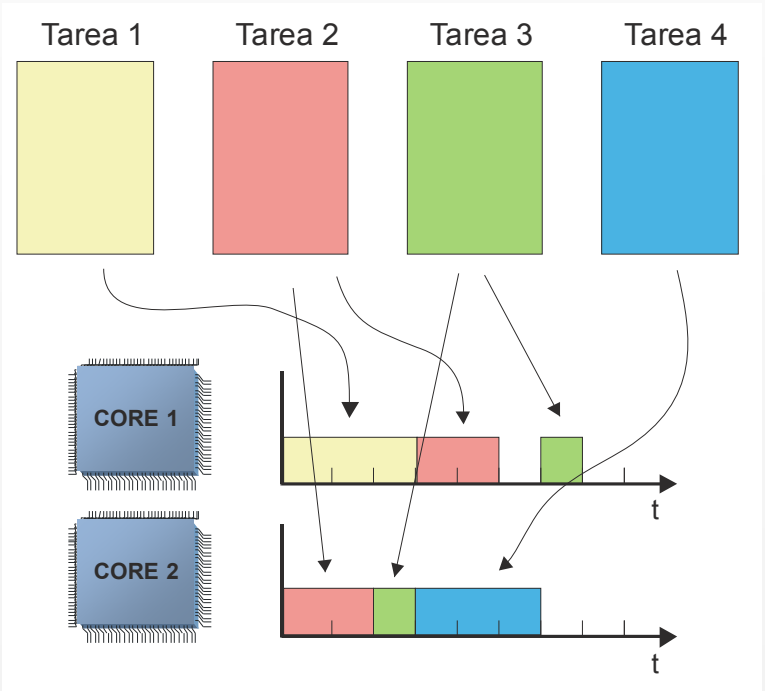


- **Esporádica:** repetitiva, T es intervalo mínimo entre activaciones



Planificación de Tareas

- La **planificación** de tareas consiste en asignar las tareas a los recursos activos de un sistema (principalmente el procesador o procesadores) para garantizar la ejecución de todas las tareas de acuerdo a un conjunto de restricciones especificas.
  - En tareas de tiempo real las restricciones suelen estar asociadas a restricciones temporales como los plazos límites.



Diseño de la planificación de tareas.

El problema de la planificación supone:

- Determinar los procesadores disponibles a los que se puede asociar las tareas.
- Determinar las relaciones de dependencias de las tareas:
  - Relaciones de precedencia que hay entre las distintas tareas.
  - Determinar los recursos comunes a los que acceden las distintas tareas.
- Determinar el orden de ejecución de la tareas para garantizar las restricciones especificadas.

Esquema de planificación de tareas

Para determinar la planificabilidad de un conjunto de tareas se requiere un esquema de planificación que cubra los dos siguientes aspectos:

- Un **algoritmo de planificación**, que define un criterio (política de planificación) que determina el orden de acceso de las tareas a los distintos procesadores.
- Un **método de análisis** (test de planificabilidad) que permite predecir el comportamiento temporal del sistema, y determina si la planificabilidad es factible bajo las condiciones o restricciones especificadas
  - Se pueden comprobar si los requisitos temporales están garantizados en **todos los casos posibles**.
  - En general se estudia el **peor comportamiento posible**, es decir, con el WCET (*Worst Case Execution Time*).

Cálculo del WCET.

Todos los métodos de planificación parten de que se conoce el WCET (C_w), el tiempo más largo posible de ejecución de una tarea, es decir, el máximo valor de C para cualquier ejecución posible de dicha tarea.
Hay dos formas de obtener los valores de C_w:

- **Medida directa** del tiempo de ejecución (en el peor caso) en la plataforma de ejecución.
  - Se realizan múltiples experimentos, y se hace una estadística.
- **Análisis del código**, se basa en calcular el peor tiempo:
  - 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.

En los ejemplos y ejercicios, y mientras no se diga lo contrario, asumimos que siempre se tarda lo mismo (C) en ejecutar una tarea determinada. Por tanto, se cumple $C = C_w$.

Restricciones temporales

Para determinar la planificación del sistema necesitamos conocer las **restricciones temporales** de cada tarea del sistema.

Aquí vemos dos ejemplos de restricciones temporales:

- $C = 2\,\text{ms}$, $T = D = 10\,\text{ms}$.
  Es una tarea periódica que se activa cada $10\,\text{ms}$, y se ejecuta en un tiempo máximo de $2\,\text{ms}$ en el peor de los casos.
- $C = 10\,\text{ms}$, $T = 100\,\text{ms}$; $D = 90\,\text{ms}$.
  Es una tarea periódica que se activa cada $100\,\text{ms}$, se ejecuta en cada activación un máximo de $10\,\text{ms}$, y desde que se inicia la activación hasta que concluye no puede pasar más de $90\,\text{ms}$.

Sección 2
Esquemas de planificación

2.1. Planificación Cíclica.
2.2. Planificación con prioridades

Tipos de Esquemas de Planificación

Los esquemas de planificación para un sistema monoprocesador son:

- **Planificación estática *off-line* sin prioridades**
  - Planificación cíclica (ejecutivo cíclico)
- **Planificación basada en prioridades**
  - **Prioridades estáticas:**
    Cada tarea tiene asociada una prioridad fija durante toda la ejecución del sistema:
    - **RMS:** (*Rate Monotonic Scheduling*)
      prioridad al más frecuente (menor periodo de activación)
    - **DMS:** (*Deadline Monotonic Scheduling*)
      prioridad al más urgente (menor *deadline*)
  - **Prioridades dinámicas:**
    Las prioridades de las tareas cambian durante la ejecución:
    - **EDF:** (*Earliest Deadline First*):
      prioridad al de siguiente plazo de respuesta máximo (*deadline*) más próximo
    - **LLF:** (*Least Laxity First*):
      prioridad al de menor holgura

Subsección 2.1
Planificación Cíclica.

Planificación Cíclica

La planificación se basa en construir un plan de ejecución (**plan principal**) de forma explicita y fuera de línea (*off-line*) en el que se especifica el entrelazado de las tareas periódicas de tal forma que su ejecución cíclica garantiza el cumplimiento de los plazos de las tareas.

- La estructura de control o programa cíclico se denomina **ejecutivo cíclico**.
- El entrelazado es fijo y se define en el **plan principal** que se construye antes de poner en marcha el sistema (*off-line*).
- La duración del **ciclo principal** se denomina **hiperperiodo** (T_M):
  - $T_M = \text{mcm}(T_1, T_2, \ldots, T_n)$ (el hiperperiodo es el mínimo comun múltiplo de los periodos de las tareas)
  - se supone tiempo entero (ticks de reloj)
  - el comportamiento temporal del sistema se repite cada ciclo principal
- El ciclo principal se descomponen en varios **ciclos secundarios** de igual duración todos ellos (T_S)

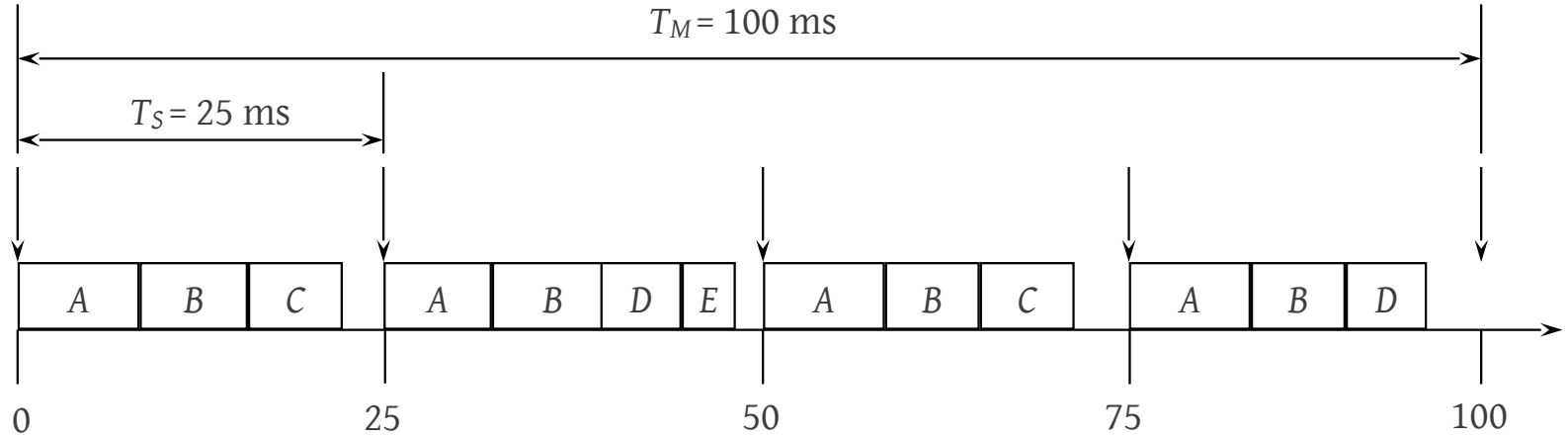
Ejemplo de planificación cíclica.

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

- El ciclo principal dura 100 ms

$$T_M = \text{mcm}(25, 25, 50, 50, 100) = 100$$

- Se compone de 4 ciclos secundarios, con $T_S = 25$ ms.



(en estos ejemplos, si no se dice lo contrario , se asume que $D_i = T_i$, el deadline es el periodo).

Implementación de la Planificación Cíclica

Esta planificación se puede implementar en C con una función similar a la que aparece aquí (cada tarea se implementa con una función C).

```
void EjecutivoCiclico()
{
   const int nciclos = 4 , // número de ciclos secundarios (constante elegida en el diseño)
   int       frame   = 0 ; // número del ciclo secundario a ejecutar a continuación
   while( true )
   {  // ejecutar la secuencia de tareas correspondiente al núm. de ciclo secundario
      switch( frame )
      {  case 0 : A(); B(); C();       break ;
         case 1 : A(); B(); D(); E(); break ;
         case 2 : A(); B(); C();       break ;
         case 3 : A(); B(); D();       break ;
      }
      frame = ( frame+1 ) % nciclos ; // avanzar al siguiente ciclo secundario
      esperar_tick_reloj( 25 ) ; // espera inicio siguiente periodo de 25 miliseg.
   }
}
```

- Cada iteración del bucle **while** ejecuta un ciclo secundario.
- Cada 4 iteraciones del bucle **while** ejecutan un ciclo principal.

Diseño de un ejecutivo cíclico

Se puede utilizar un método sistemático que ayuda a construir el plan de ejecución cíclico, determinando un "buen" valor de T_s. Para ello se tienen en cuenta estas

- **Restricciones:** necesariamente se cumple
  - el ciclo secundario tiene que ser un divisor del ciclo principal, es decir, siempre existirá un entero k tal que:

$$T_M = k \cdot T_s$$

  - el ciclo secundario tiene que ser mayor o igual que el tiempo de cómputo de cualquier tarea,

$$\text{máximo}(C_1, C_2, \ldots, C_n) \leq T_s$$

- **Sugerencias:** es aconsejable intentar en principio que:
  - el ciclo secundario sea menor o igual que el minimo deadline (plazo de respuesta máximo)

$$T_s \leq \text{mínimo}(D_1, D_2, \ldots, D_n)$$

Propiedades de la Planificación Cíclica

- No hay concurrencia en la ejecución.
  - Cada ciclo secundario es una secuencia de llamadas a **procedimientos**
  - No se necesita un núcleo de ejecución multitarea
- Los procedimientos pueden **compartir datos**.
  - No se necesitan mecanismos de exclusión mutua como los semáforos o monitores
- No hace falta analizar el comportamiento temporal.
  - El sistema es correcto por construcción.

Problemas de la Planificación Cíclica

- Dificultad para incorporar tareas con periodos largos.
- Las tareas esporádicas son difíciles de tratar.
  - Se puede utilizar un servidor de consulta.
- El plan ciclo del proyecto es difícil de construir.
  - Si los periodos son de diferentes órdenes de magnitud el número de ciclos secundarios se hace muy grande.
  - Puede ser necesario partir una tarea en varios procedimientos.
  - Es el caso más general de sistemas en tiempo real críticos.
- Es poco flexible y difícil de mantener.
  - Cada vez que se cambia una tarea hay que rehacer toda la planificación.

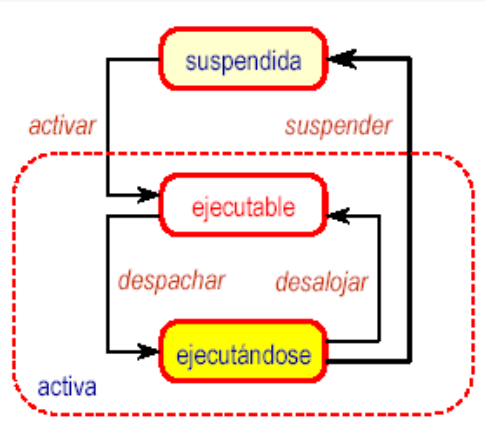
Subsección 2.2
Planificación con prioridades

Planificación con prioridades

La **planificación con prioridades** permite solventar los problemas descritos. Cada tarea tiene asociado un valor **entero positivo**, llamado **prioridad** de la tarea:

- Es un atributo de las tareas normalmente ligado a su importancia relativa en el conjunto de tareas.
- Por convención se asigna números enteros mayores a procesos más urgentes.
- La prioridad de una tarea la determina sus necesidades temporales; no es importante el rendimiento o comportamiento del sistema.

- Una tarea puede estar en varios estados:
- Las tareas ejecutables se despachan para su ejecución en orden de prioridad.



Planificación RMS

RMS: *Rate Monotonic Scheduling*

Es un método de planificación estático on-line con asignación mayor prioridad a las tareas más frecuentes (es decir: con menor periodo T_i)

- A cada tarea i se le asigna una prioridad (P_i) basada en su período (T_i): cuanto menor sea el periodo (mayor frecuencia) mayor será su prioridad.

$$\forall i,j \ : \quad T_i < T_j \implies P_i > P_j$$

- Esta asignación de prioridades es optima en el caso de que todas las tareas sean periódicas, y el plazo de respuesta máxima D coincida con el periodo.

Ejemplo de planificación RMS (1)

Supongamos que tenemos un conjunto de tres tareas (1,2 y 3) para las cuales se han fijado las siguientes restricciones temporales:

Tareas	C	D	T
1	10	30	30
2	5	40	40
3	9	50	50

La aplicación del método de planificación RMS indica que:

- la tarea 1 tiene la mayor prioridad,
- la tarea 2 tiene prioridad intermedia
- la tarea 3 tiene la menor prioridad.

Tests de Planificabilidad

Los tests de planificabilidad permiten determinar si el conjunto de tareas del sistema es planificable según un algoritmo de planificación antes de su ejecución
Existen diversos tipos de tests aplicables según el algoritmo de planificación

- **Tests de planificabilidad suficientes**:
  - En caso de éxito en la aplicación del test el sistema es planificable.
  - En caso contrario no tenemos información: podría ser planificable o no serlo.
- **Tests de planificabilidad exactos**: (suficiente y necesario)
  - En caso de éxito en la aplicación del test el sistema es planificable.
  - En caso contrario el sistema no es planificable: habrá fallos inevitablemente (no se cumplirá algún deadline).

Test de Liu & Layland

Es un test suficiente, determina la planificabilidad para un sistema de n tareas periódicas independientes con prioridades RMS.

- Se usan dos valores reales, el **factor de utilización** (U), y el **factor de utilización máximo** ($U_0(n)$) (para n tareas). Se definen así:

$$U \equiv \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad U_0(n) \equiv n\left(\sqrt[n]{2}-1\right)$$

- El valor de $U_0(1)$ es 1. A partir de ahí decrece al crecer n, hasta el límite (para n infinito), que es $\ln 2 \approx 0.693147 \leq U_0(n)$, $\forall n$.
- Un sistema **pasa el test** si el factor de utilización es menor o igual que el máximo posible:

$$U \leq U_0(n)$$

en ese caso el sistema es planificable. En caso contrario, no se puede afirmar nada.

Ejemplo RMS **cumpliendo** el Test Liu & Layland

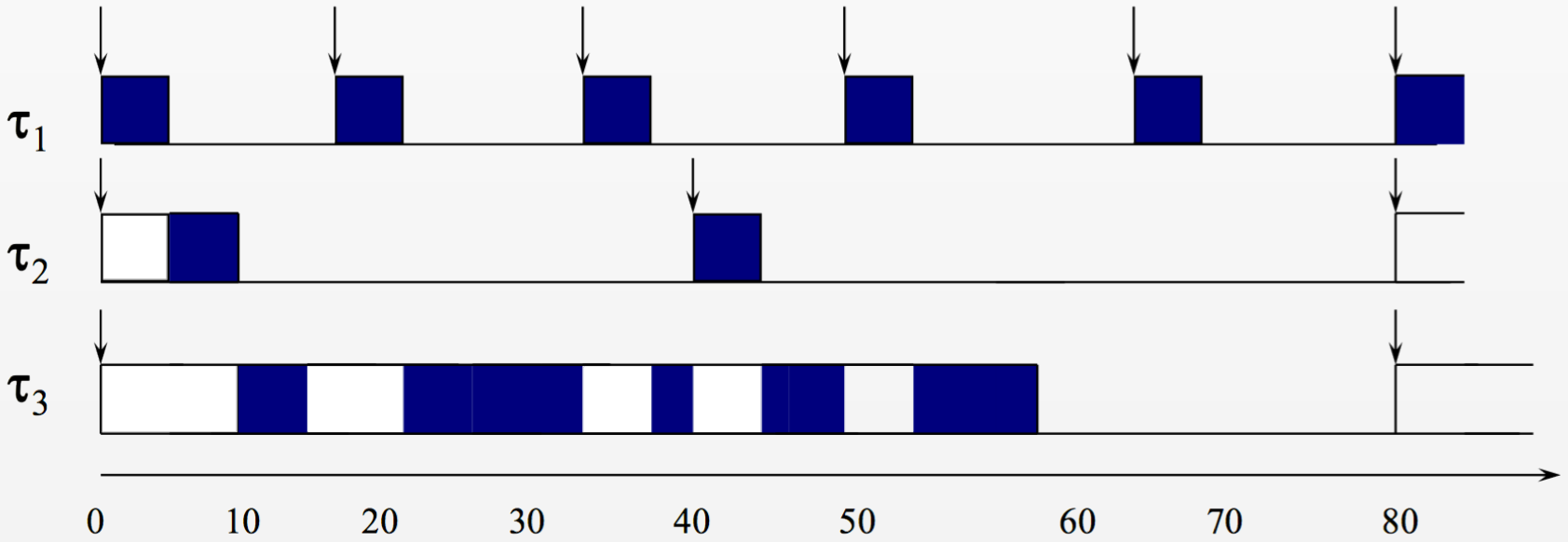
Supongamos ahora que tenemos estos datos de entrada para un sistema:

Tareas	C	D	T	C/T
τ_1	4	16	16	0,250
τ_2	5	40	40	0,125
τ_3	32	80	80	0,400

El sistema **pasa** el Test de Liu & Layland, ya que:

$$U = 0,775 \leq 0,779 = U_0(3)$$

El cronograma de este sistema es este:



Ejemplo RMS **no cumpliendo** el Test Liu & Layland

Supongamos ahora que tenemos estos datos de entrada para un sistema:

Tareas	C	D	T
Tarea 1	10	30	30
Tarea 2	10	40	40
Tarea 3	10	50	50

Podemos calcular U y compararlo con $U_0(3)$:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{10}{30} + \frac{10}{40} + \frac{10}{50} = 0,783333 \not\leq 0,779 = U_0(3)$$

es decir:

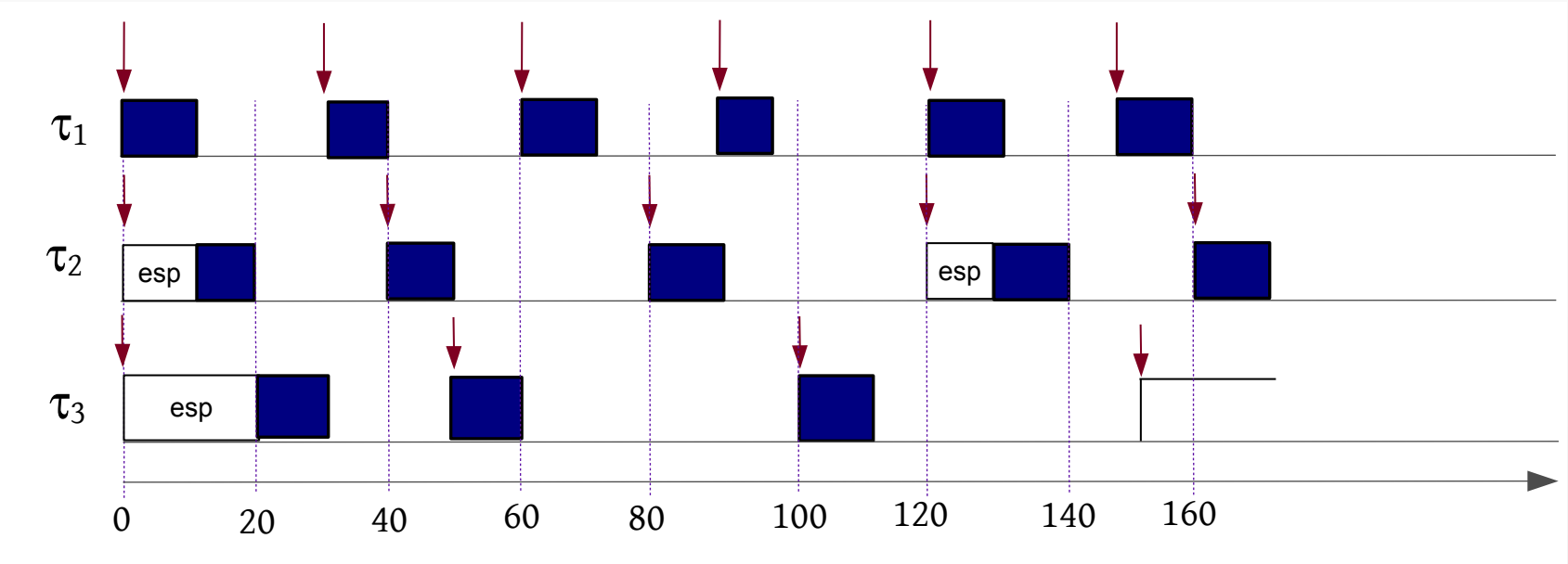
- El sistema **no pasa el test**.
- **No podemos decir si es planificable o no lo es.**
- Para saberlo, podemos analizar el cronograma.

Ejemplo RMS sin cumplir test: cronograma

Para analizar la planificabilidad del sistema con dichas restricciones, hay que hacer el cronograma y verificar que:

- para cada tarea i, se cumple el plazo de respuesta: $R < D_i$
- esto se debe verificar para un hiperperiodo (después se repite).

En el ejemplo, se debería hacer el cronograma completo hasta $t = 600$ ($= \text{mcm}\{30, 40, 50\}$), aquí vemos una primera parte (hasta $t = 160$):



Planificación **EDF**

Es un esquema de planificación con **prioridades dinámicas**. Se denomina **EDF** (por ***Earliest Deadline First***), o bien **primero al más urgente**:

> *La asignación de prioridad se establece una prioridad más alta a la que se encuentre más próxima a su plazo de respuesta máxima (deadline).*

Características:

- En caso de igualdad se hace una elección no determinista de la siguiente tarea a ejecutar.
- Es un algoritmo de planificación dinámica, dado que la prioridad de cada tarea cambia durante la evolución del sistema.
- Es más óptimo porque no es necesario que las tareas sean periódicas.
- En menos pesimista que RMS.

Test de planificabilidad para **EDF**

El test de planificación de Liu & Layland se puede aplicar también para el caso de planificación EDF.

- En este caso, un sistema pasa el test si el factor de utilización es menor o igual a la unidad:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

- Esto implica que la planificación EDF puede aplicarse a más sistemas que la planificación RMS (la cota máxima de U es mayor en EDF que en RMS)
- El test ahora **es exacto** (necesario y suficiente):
  - Si un sistema pasa el test, es planificable con EDF.
  - Si un sistema no pasa el test, no es planificable con EDF.

Ejemplo de planificación EDF

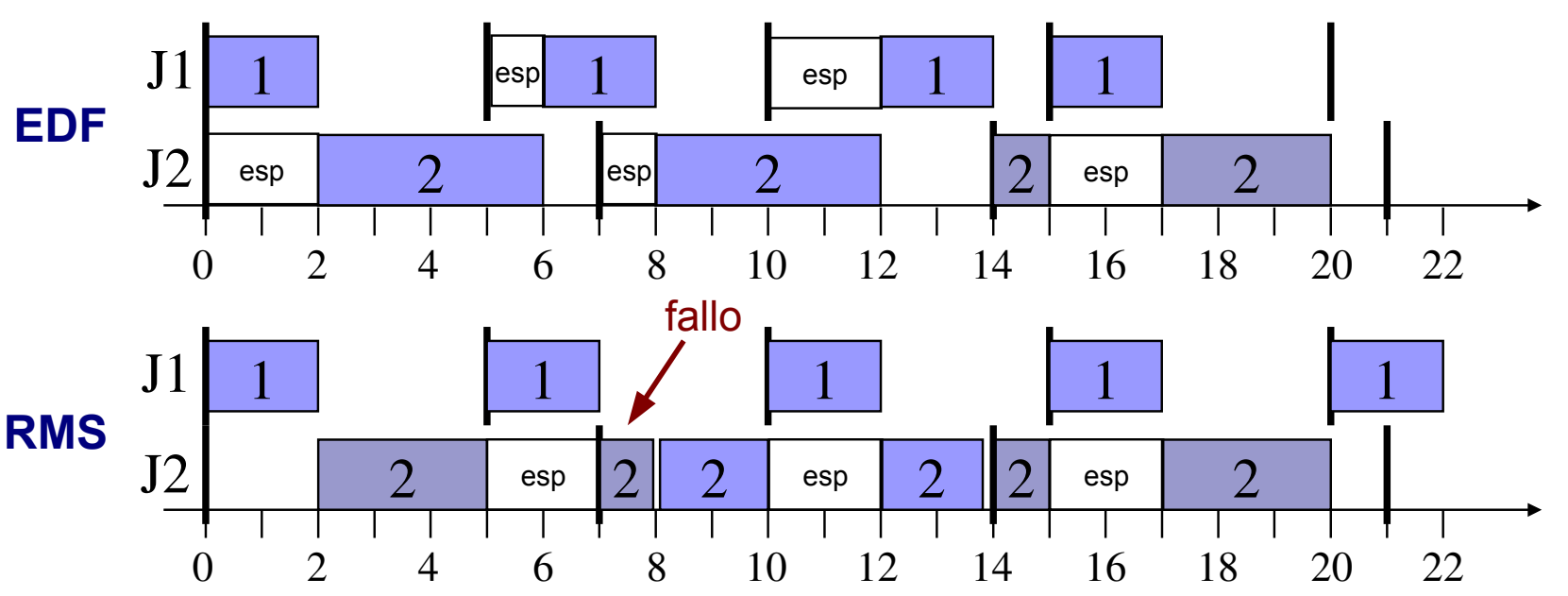
Vemos un sistema de ejemplo, y aplicamos el test

Tareas	C	D	T
J1	2	5	5
J2	4	7	7

Pasa el test de Liu & Layland (EDF):

$$U = \frac{2}{5} + \frac{4}{7} = 0,97 \leq 1$$

Se verifica que el sistema **es planificable con EDF**, pero no con RMS:



Bibliografía del tema 4.

Para más información, ejercicios, bibliografía adicional, se puede consultar:

4.1. **Concepto de sistemas de tiempo real. Medidas de tiempo y modelo de tareas.**
Burns (2005), capítulos 1, 9 y 10 (en Inglés).

4.2. **Esquemas de planificación.**
Burns (2002), capítulo 13 (en Español).
Burns (2005), capítulos 11 y 12 (en Inglés).