Departamento de Arquitectura y Tecnología de Computadores
UNIVERSIDAD DE SEVILLA

Tema 6:
Sistemas en Tiempo Real

INDICE:

- Introducción al Tiempo Real: Conceptos Básicos
- Secuenciamiento de tareas
- Soporte para Sistemas en Tiempo Real

Introducción al Tiempo Real

- Definiciones:

  - *Cualquier sistema en el que el tiempo en el que se produce la salida es significativo. Esto generalmente es porque la entrada corresponde a algún movimiento del mundo físico, y la salida está relacionada con dicho movimiento. El intervalo entre el tiempo de entrada y el de salida debe ser lo suficientemente pequeño para una temporalidad aceptable.*

  - *Sistema en tiempo real son aquellos que deben producir respuestas correctas dentro de un intervalo de tiempo definido. Si el tiempo de respuesta excede ese límite, se produce una degradación del funcionamiento y/o un funcionamiento erróneo.*

Introducción al Tiempo Real

➢ Clasificación (según requisitos temporales)

- **Tiempo real estricto** *(hard real time)*: Cuando es absolutamente necesario que la respuesta se produzca dentro del límite especificado. Ej.: control de vuelo.
- **Tiempo real no estricto** *(soft real time)*: Cuando se permite la pérdida ocasional de especificaciones temporales, aunque debe cumplirse normalmente. Ej.: sistema de adquisición de datos
- **Tiempo real firme** *(firm real time)*: Cuando se permite la pérdida ocasional de especificaciones temporales, pero dicha perdida no implica beneficios ya que la respuesta retrasada es descartada. Ej.: sistema multimedia.

Introducción al Tiempo Real

- Clasificación (según aplicaciones)
  - Control de procesos industriales: Conseguir que un variables siga una evolución determinada (temperatura, caudal, presión, etc.) → La misión del computador es generar las señales que permiten conseguir el objetivo, a partir de la medida de la variable a controlar, del valor especificado para ésta y de un determinado algoritmo de control.
  - Manufactura: control sobre los procesos de fabricación, con el objetivo de reducción de costes y/o aseguramiento de la calidad. → el computador se encarga de coordinar las tareas a realizar por los distintos componentes del sistema como son, las máquinas herramientas, las cintas transportadoras, etc.
  - Comunicación, mando y control: recopilación y mantenimiento de información como ayuda a la toma de decisiones (reserva de billetes, monitorización de pacientes, control de trafico aéreo, etc..).

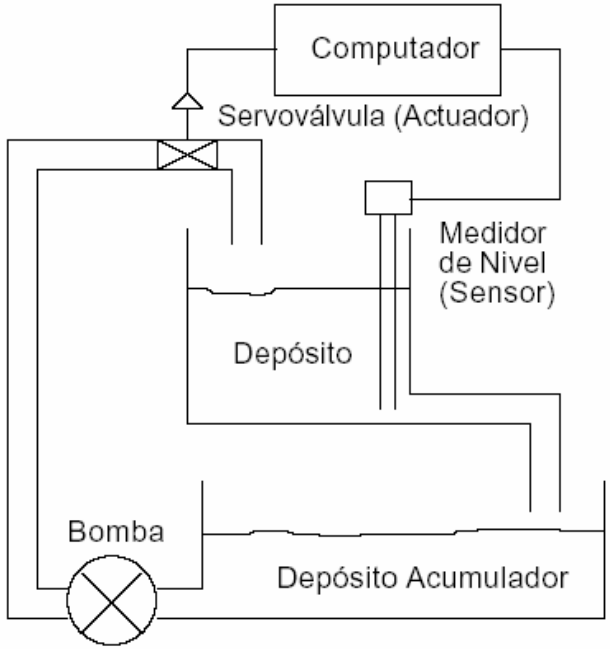
Introducción al Tiempo Real

SISTEMA EMPOTRADO

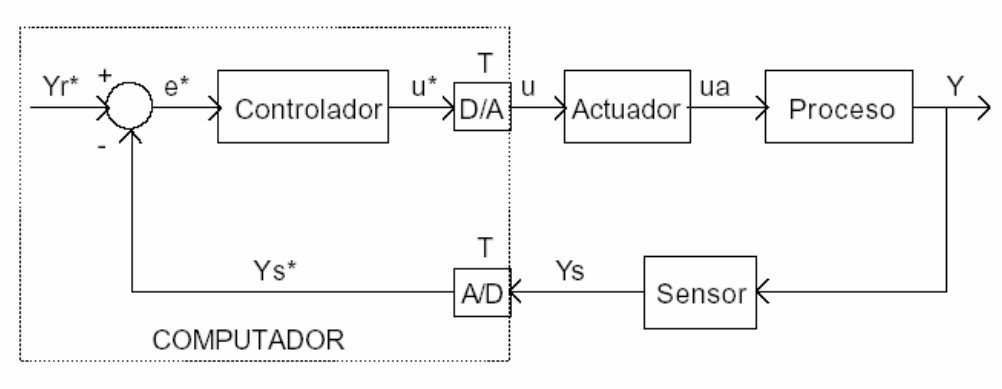
Cuando en un sistema en Tiempo Real el computador forma parte del núcleo del sistema y es el encargado de obtención y procesado de la información y de la generación de las salidas.

Primer ejemplo de sistemas en Tiempo Real

- Control del nivel de un deposito: *conseguir que el nivel de líquido dentro del depósito alcance cierto valor especificado:*
  - Mediante un *conducto* se aporta al *depósito* un determinado caudal de líquido, variable mediante una *servoválvula*.
  - La altura que alcanza el líquido dentro del depósito se determina mediante un *medidor de nivel* y el resultado de la medida se envía a un *computador*.
  - En el *computador*, a partir del *nivel especificado*, de la *medida del nivel* realizada por el *sensor* (medidor de nivel) y de un determinado *algoritmo de control*, se calcula el valor de la *señal de control* que permite posicionar la servoválvula (*actuador*) de tal forma que se pueda conseguir el objetivo fijado.
  - El valor de la *señal de control* calculado en el *computador* se envía a la *servoválvula*.



Primer ejemplo de sistemas en Tiempo Real



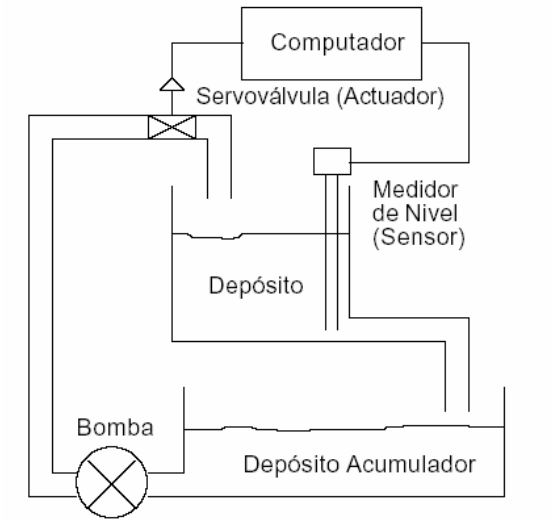
- Un *sensor*, que mide la salida continua del proceso Y(t) y genera la señal continua de medida Ys(t).
- Un *conversor analógico digital A/D*, que toma muestras de Ys(t), con período de muestreo T, y genera la señal de medida discreta Ys* que posteriormente digitaliza.
- Un *comparador*, que entre períodos de muestreo, calcula el valor de la señal de error e*, como la diferencia entre el último valor obtenido de Ys* y el valor indicado en el computador para la señal de referencia Yr*.
- Un *controlador*, que entre períodos de muestreo, a partir del valor de la señal de error e* y mediante un determinado algoritmo, genera como salida el valor de la señal de control u* que se debe aplicar al actuador.
- Un *conversor digital analógico D/A*, que manteniendo constante durante un tiempo T el último valor calculado de u* produce la señal de control continua u(t) que se aplica al actuador.
- Un actuador, que admite como entrada u(t) y genera como salida la señal de control continua del proceso ua(t).
- Un *proceso*, que recibe la señal ua(t) y produce la señal de salida continua Y(t) que es la variable a controlar.

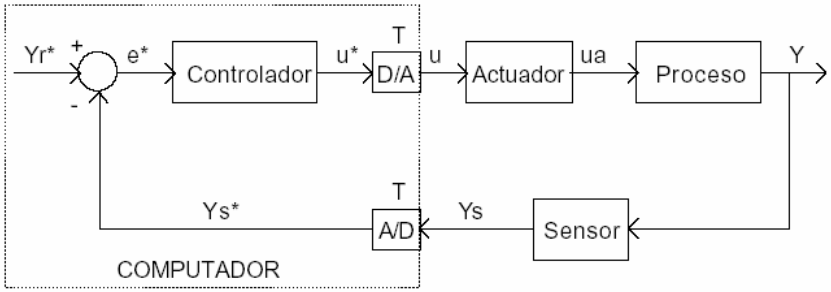
Primer ejemplo de sistemas en Tiempo Real

- La entrada y la salida del computador se actualizan en los mismos instantes de tiempo (*instantes de muestreo*) que se producen a intervalos de tiempo T (*período de muestreo*).

- Un reloj de tiempo real que provoque una interrupción en cada período de muestreo permite tomar muestras de la señal recibida del sensor y mandar la señal de control al actuador en los momentos precisos.

Primer ejemplo de sistemas en Tiempo Real





- El *sensor* corresponde al medidor de nivel.
- El *actuador* está formado por el conjunto depósito acumulador, bomba y servoválvula.
- El *proceso* se encuentra descrito por la relación entre el caudal de líquido que entra en el depósito y la altura alcanzada por el líquido en él.
- La *señal de salida* o *variable a controlar* Y(t) es la altura en centímetros que alcanza el líquido dentro del depósito.
- La *señal de medida* Ys(t) es la medida de Y(t) en voltios realizada por el sensor.
- La *señal de control* u(t) es la tensión en voltios aplicada a la servoválvula para aumentar o reducir la sección de paso del líquido.
- La *señal de control* ua(t) es el caudal de entrada al depósito.

Primer ejemplo de sistemas en Tiempo Real

- Señales en los sistemas en Tiempo Real:

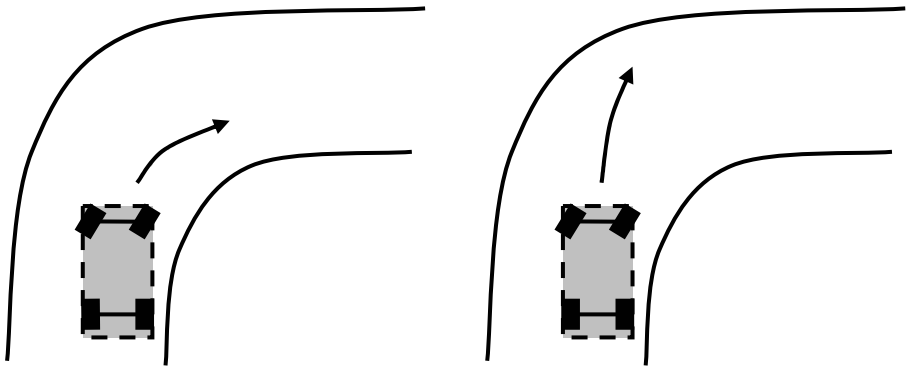
  - *Señales continuas*. Son todas las que intervienen en la planta, desde la señal de control u(t) hasta la señal de medida Ys(t). En señales continuas tiene sentido hablar de su valor en cualquier instante de tiempo.

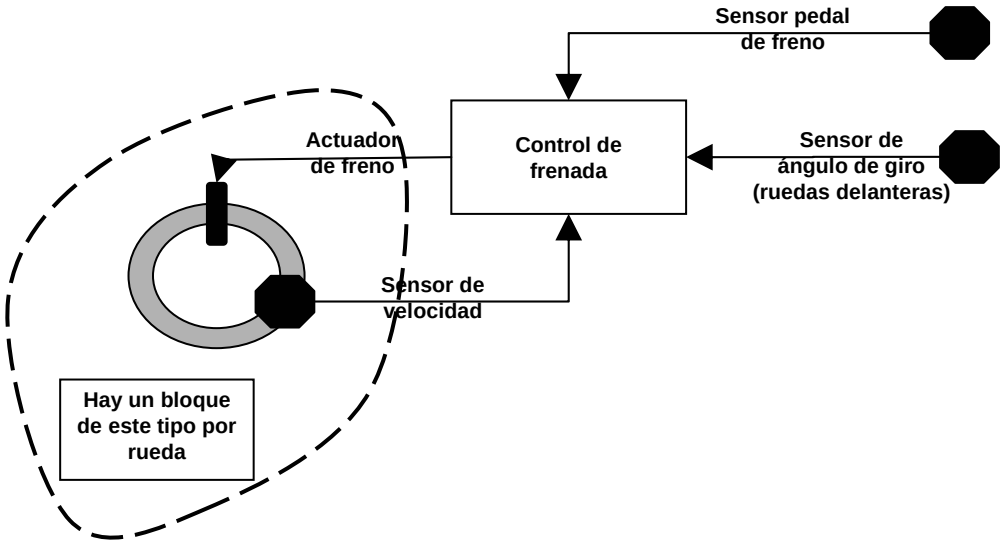
  - *Señales discretas*. Corresponden a la señal que resulta del muestreo de la señal de medida Ys(t) y a la señal que toma los valores de la señal de control u(t) en cada período de muestreo. En señales discretas sólo tiene sentido hablar de su valor en los instantes de muestreo.

  - *Señales digitales*. Son aquellas con las que opera el computador. En cada período de muestreo se digitaliza la señal discreta de medida y se pasa de digital al valor discreto de la señal de control.

Ejemplos de sistemas en Tiempo Real

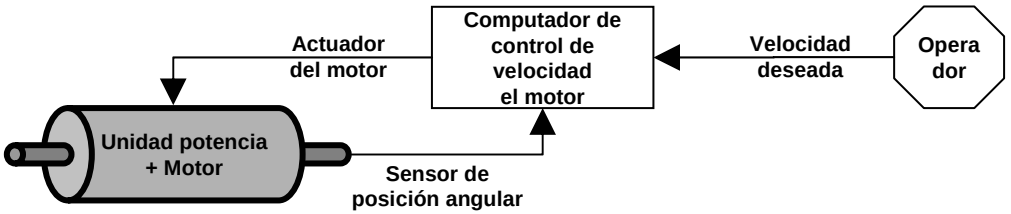
- **ABS**: El sistema en función de la velocidad, el ángulo de giro y el deseo de conductor de frenar, tomará la decisión correcta y todo ello en un tiempo determinado.





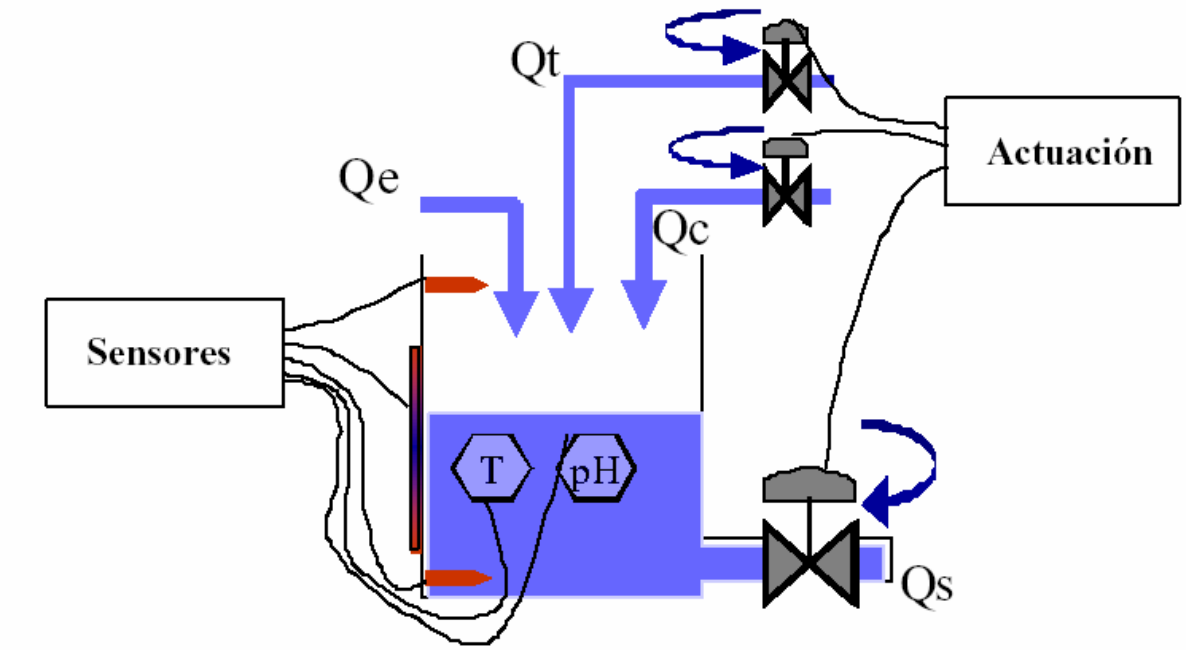
Ejemplos de sistemas en Tiempo Real

- **Control de la velocidad de un motor**: en función de las órdenes del operador y de la velocidad real del motor generará la señal correspondiente para que la velocidad del motor sea la deseada por el operador



Ejemplos de sistemas en Tiempo Real

- **Planta de tratamiento de agua:** que las entradas a un depósito son agua sin tratar, agua con ph conocido y agua a una temperatura conocida y el objetivo es mantener el nivel del depósito de agua y el agua de salida tenga un ph y una temperatura controlada

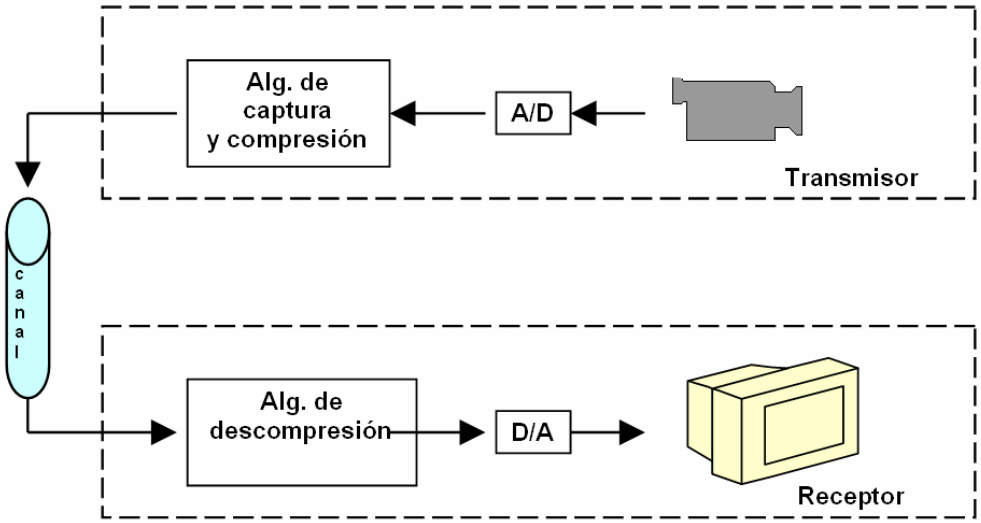


Ejemplos de sistemas en Tiempo Real

- Algunas características:
  - Una actuación incorrecta puede ser peligrosa
  - Los sistemas E/S están orientado a las adquisición de datos de algún proceso físico:
    - Las medidas tienen un tiempo de validez, es decir, deben realizarse medidas continuamente, Ej.: velocidad del motor,
    - Las mediciones pueden ser dirigidas por tiempo (velocidad del motor) o por eventos (pisada del freno).
    - Suelen ser necesarios procesos de acondicionamiento de las mediciones (convertidores A/D y D/A).
    - Los procesos de control pueden ser continuos (cíclicos,) o discretos (eventos), tratamiento de aguas y control de la frenada respectivamente,
  - Necesitan supervisión de la integridad del sistema (datos adquiridos, actuaciones…).

Ejemplos de sistemas en Tiempo Real

- Sistema multimedia: orientadas al tratamiento de imágenes y sonido, es realidad tratamiento de valores numéricos procedentes del muestreo de imágenes y/o sonidos.
- Sistema de transmisión de imágenes:
  - Requerimientos de cálculo:
    - Conversión analógico-digital
    - Compresión y descompresión de la imagen y sonido
  - Requerimientos temporales:
    - En el video por la frecuencia de la captura de la cámara
    - En el sonido por la frecuencia de los sonido (9-20Khz)



Ejemplos de sistemas en Tiempo Real

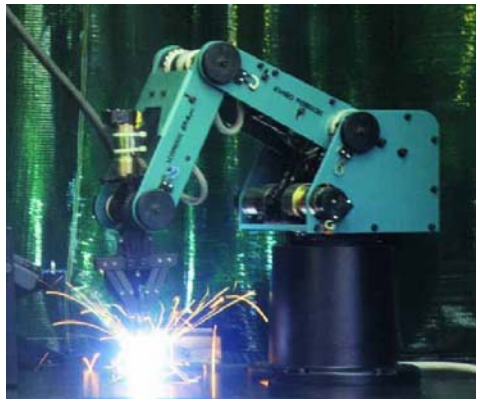
- Más Características:
  - Manejan gran cantidad de datos
    - Siempre son intensivos en datos, es decir, necesitan un gran volumen de información.
    - A veces necesitan gran potencia de computación, para procesar esos datos
  - No producen acción sobre el medio físico. Una actuación incorrecta (lógica o temporalmente) no suele ser peligrosa.

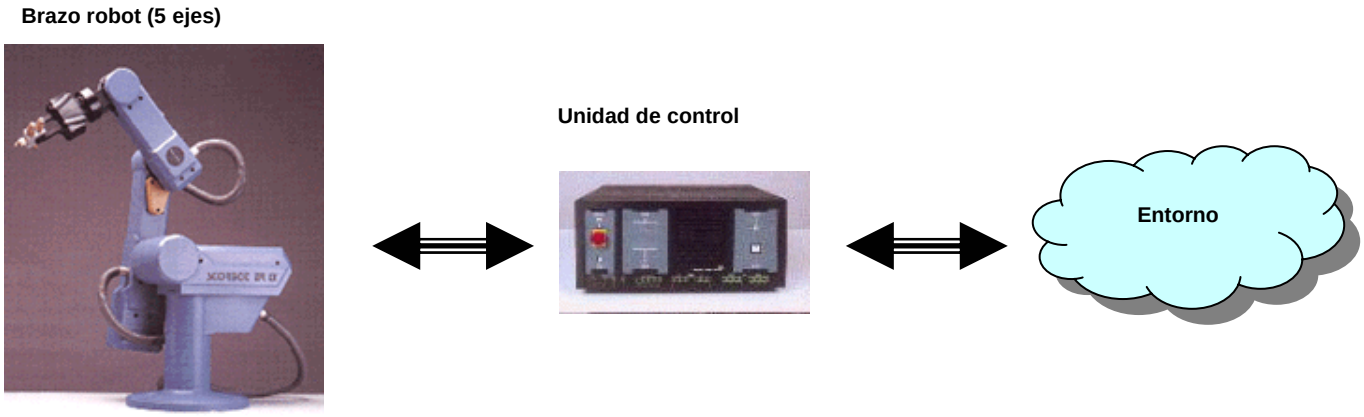
Ejemplos de sistemas en Tiempo Real

- Sistemas empotrados se encuentran en multitud de productos y están integrados en la estructura física de la maquinaria que controlan:
  - Vehículos: Coches, autobuses, trenes, aviones, barcos, grúas,…
  - Maquinaria industrial: Robots, células de fabricación, maquinas/herramienta,…
  - Sistemas de comunicación: Teléfonos móviles, repetidores, rotures, radio,…
  - Electrodomésticos: Elec. línea blanca, televisores, videos, cámaras fotos, alarmas, …
  - Ocio: Consolas videojuego, juguetes, salas de cine, sistemas de sonido, iluminación,…
  - Armamento: Guía de misiles, balística,...

Ejemplos de sistemas en Tiempo Real

- La unidad de control para un SCORBOT ER-VII:
  - CPU: 68020-68881.
  - 512 KB. ROM, 512 KB RAM
  - E/S:
    - Tarjetas de control de motores (6-12 ejes).
    - Unidad de control Brazo robot (5 ejes)
    - 2-10 RS232.
    - 16 entradas, 16 salidas digitales (12-24 voltios)

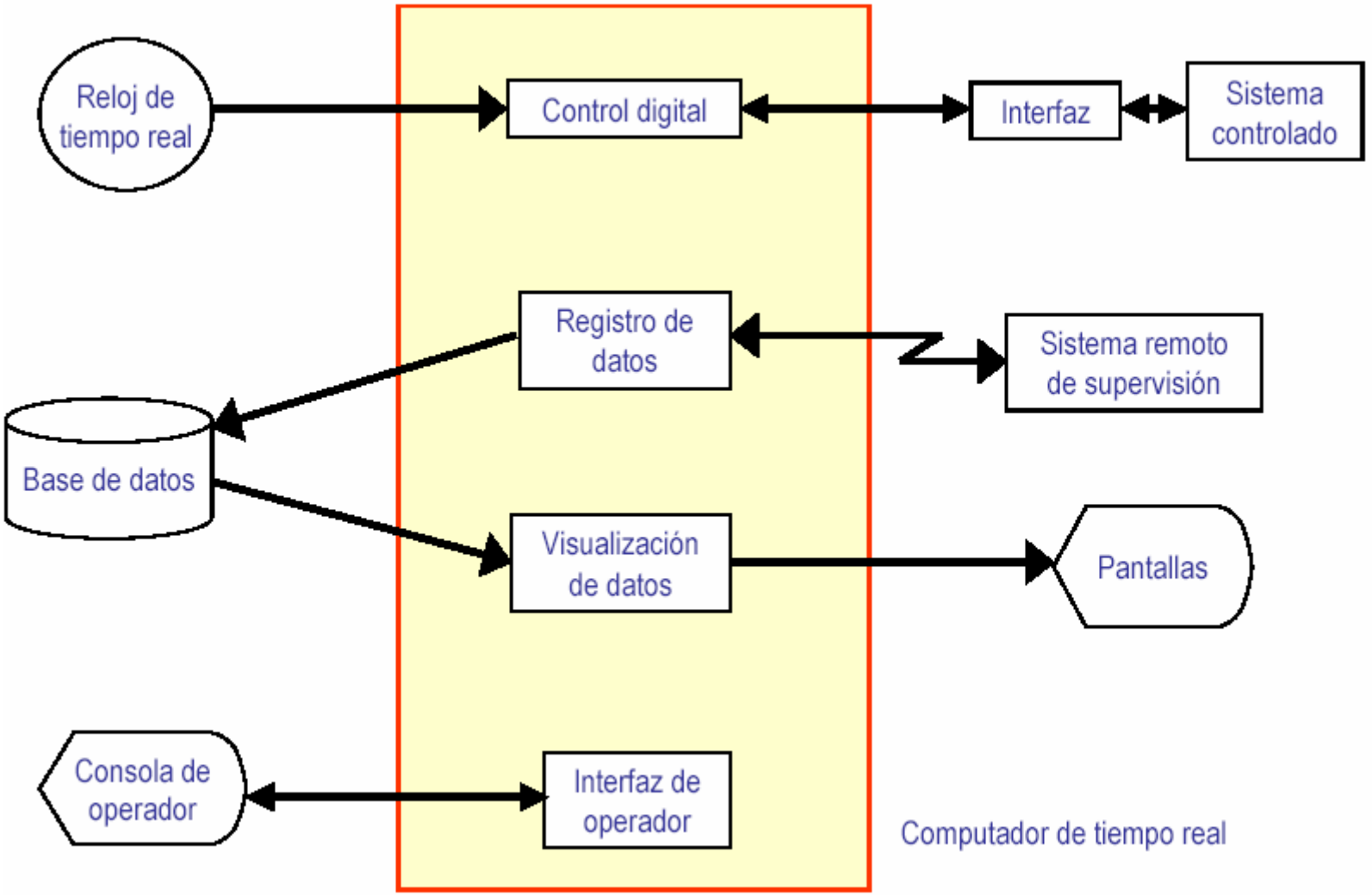




Ejemplos de sistemas en Tiempo Real

- Últimas características:
  - En un sistema con este existen varias tareas realizándose simultáneamente:
    - Interpretación de la lista de acciones a realizar.
    - Cálculo de las trayectorias: → Por qué puntos hay que pasar, qué tipo de movimientos hay que realizar entre esos puntos, a que velocidad,...
    - Traducción trayectoria → movimientos en los motores de los distintos ejes.
    - Control de los motores (5).
    - Supervisión.
    - ....
  - Los sistemas empotrados presentan las siguientes características comunes:
    - Recursos limitados: procesador, memoria, pantalla...
    - Arquitectura orientada al sistema que controla. Sobre todo los sistemas de E/S.
    - Poca o ninguna intervención de operadores humanos.

Estructura general de un STR



Estructura general de un STR

- *Sistema a controlar*. Cualquier sistema que pueda ser controlado.

- *Interfaz con el sistema*. adaptar las señales que desde el sistema se envían al computador y desde el computador se mandan al sistema. Esta formado por conversores analógicos digitales y digitales analógicos, que permiten medir el estado del sistema a controlar e imponer un control sobre la operación a realizar en dicho sistema.

- *Reloj de tiempo real*. un reloj que permita tomar muestras de las señales recibidas de los dispositivos, así como, mandarles determinadas señales en los momentos precisos. El reloj de tiempo real provoca una interrupción en cada período de muestreo.

- *Consola del operador*. Permite al operador humano realizar intervenciones manuales (arranque, parada, modificaciones en el comportamiento del sistema,…).

- *Pantallas*. Se utilizan para enviar información al operador sobre el estado del sistema.

Estructura general de un STR

- *Base de datos*. Los cambios de estado del sistema son guardados en una base de datos que el operador e ingenieros de control pueden interrogar en caso de fallo del sistema o para obtener información con propósito de gestión. Esta información va creciendo y se utiliza para tomar las decisiones que surgen con el funcionamiento del sistema.

- *Sistema de monitorización remoto* . En procesos industriales, la monitorización de la planta es esencial para reducir costos y aumentar la producción. Las decisiones relativas a la producción de una planta pueden repercutir en el rendimiento de otras plantas que dependen de ella, como es el caso de una planta que produce materia prima para otra.

- *Computador*. El software que controla las operaciones del sistema está escrito en módulos que reflejan la naturaleza física del entorno. De forma general, estos módulos son:
  - *Algoritmos de control digital*. Realizan el control del sistema.
  - *Registros de datos*. Permiten guardar los cambios de estado del sistema.
  - *Información de dirección*. Permiten facilitar información sobre el estado del sistema y las operaciones que se realizan a los encargados de la dirección del sistema global.
  - *Interfaz con el operador*. Para interactuar con el operador.

Características de los STR

- Tamaño y complejidad. A menudo problemas relacionados con sistemas en tiempo real se convierten en problemas de gran tamaño y complejidad.

- Manipulación de números reales. Es la manera de representar los valores leídos del mundo real en un computador.

- Seguridad y fiabilidad. Los sistemas en tiempo real suelen estar relacionados con procesos en los que los fallos tienen consecuencias graves, por lo tanto la tolerancia a fallos es un factor de vital importancia en su diseño.

- Concurrencia. A menudo es necesario controlar el acceso a recursos compartidos, ejecutar varias tareas en paralelo, lo que provoca que los STR presente un cierto grado de procesamiento en multitarea.

Características de los STR

- Eficiencia. Esta característica es exigible a todo tipo de sistemas, aun mas en sistemas que pueden ser críticos, como los STR. Con esta característica se pretende asegura que el funcionamiento lógico de sistema es correcto y optimo.

- Dependencia del tiempo. Como ya hemos visto el tiempo es el factor distintivo de los STR, a los que se les exige no solo una corrección lógica, si no que cumplan unos determinados requerimientos temporales. Su comportamiento temporal tiene que ser determinista, y a la hora del diseño, hay que prever el peor de los casos.

- Dispositivos de E/S especiales. La conexión con el exterior esta adaptada a los procesos que se controlan y a menudo condicionan el funcionamiento de sistema. Las interacciones con el exterior pueden ser activas o pasivas, es decir, el sistema debe controlar el acceso al medio físico, o bien el medio físico perturba de alguna manera al sistema de control.

Secuenciamiento de Tareas

- Clasificación de los STR según el flujo de ejecución:
  - ***Sistemas monotarea***. Están compuestos por un único flujo de ejecución. El sistema se compone de un bucle infinito en el que se muestran los dispositivos de entrada a una determinada frecuencia y se generan las salidas correspondientes, es decir, se realiza *polling (sondeo)* sobre los dispositivos a controlar. Su principal ventaja es la sencillez pero son poco flexibles, lo que impide añadir nueva funcionalidad debido a la alta interdependencia entre las tareas a realizar.

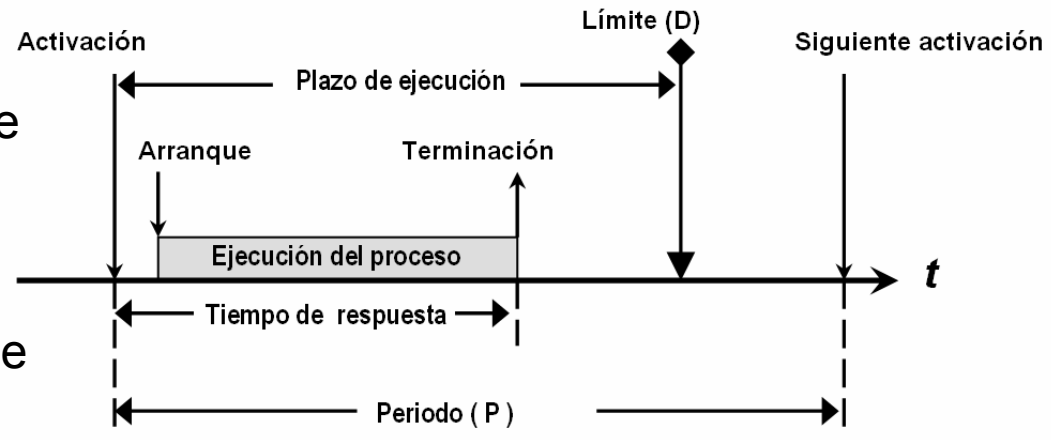
Secuenciamiento de Tareas

– **Sistemas multitarea**. Están compuestos por un conjunto de tareas (asociadas a procesos que se deben controlar) que se ejecutan de forma concurrente para el control del proceso global. Si es necesario el control de nuevos procesos sólo es preciso añadir nuevas tareas. El principal problema es la planificación de las tareas concurrentes de tal forma que se cumplan los requisitos temporales. En este tipo de sistemas es necesario controlar los recursos y la comunicación entre tareas.

Secuenciamiento de Tareas

- En función de la forma de ejecución, las tareas se clasifican en:
  - *tareas periódicas*. Se activan repetidamente a intervalos de tiempo fijo:

•*Período de activación (p)* o tiempo que transcurre entre dos activaciones consecutivas.
•*Plazo de ejecución (d)* o tiempo de respuesta máximo (máximo plazo de tiempo entre la activación y la terminación de forma correcta).
•*Tiempo de cómputo (c)* (tiempo de cómputo máximo en cada activación).
•Se debe cumplir que $0 \leq c \leq d \leq p$



Secuenciamiento de Tareas

- *Tareas esporádicas*. Se activan en instantes aleatorios y tienen requisitos temporales críticos. Las características temporales de estos procesos son:
  - *Separación mínima (p')* (tiempo mínimo que puede transcurrir entre dos activaciones consecutivas).
  - *Plazo de ejecución (d')* o tiempo de respuesta máximo (máximo plazo de tiempo entre la activación y la terminación de forma correcta).
  - *Tiempo de cómputo (c')* (tiempo de cómputo máximo en cada activación).

  Se debe cumplir que $0 \leq c' \leq d' \leq p'$. Si los requisitos temporales son no críticos se denominan *procesos esporádicos*.

- *Tareas aperiódicas*. No poseen requisitos temporales rígidos:
  - *Tiempo de cómputo.* Tiempo de ejecución en el peor de los casos.
  - *Plazo de finalización*. Tiempo máximo que puede transcurrir entre la activación de la tarea y la finalización de la ejecución de ésta.

Secuenciamiento de Tareas

- Las tareas se clasifican, atendiendo a su semántica en:
  - **Críticas**: El fallo de una de estas tareas puede ser catastrófico.
  - **Opcionales** (no críticas): Se pueden utilizar para refinar el resultado dado por una tarea crítica, o para monitorizar el estado del sistema, etc.

- Por *planificación* se entiende la asignación de recursos del sistema (incluido el procesador) a las distintas tareas que los soliciten. Uno de los problemas fundamentales en el diseño de sistemas de tiempo real es la planificación o asignación de recursos del sistema a las tareas de tiempo real de tal forma que se verifiquen sus restricciones temporales.

Secuenciamiento de Tareas

- Los objetivos que persigue la planificación de tareas son:
  - Garantizar la correcta ejecución de todos los procesos críticas.
  - Ofrecer un buen tiempo ejecución de todos los procesos no periódicos.
  - Administrar el uso de recursos compartidos.
  - Posibilidad de recuperación ante fallos software y hardware.
  - Soportar cambios de modo, esto es, cambiar en tiempo de ejecución el conjunto de tareas. Por ejemplo: un cohete tiene que realizar acciones muy distintas durante el lanzamiento, estancia en orbita y regreso; en cada fase, el conjunto de tareas que se tengan que ejecutar ha de ser distinto.

Secuenciamiento de Tareas

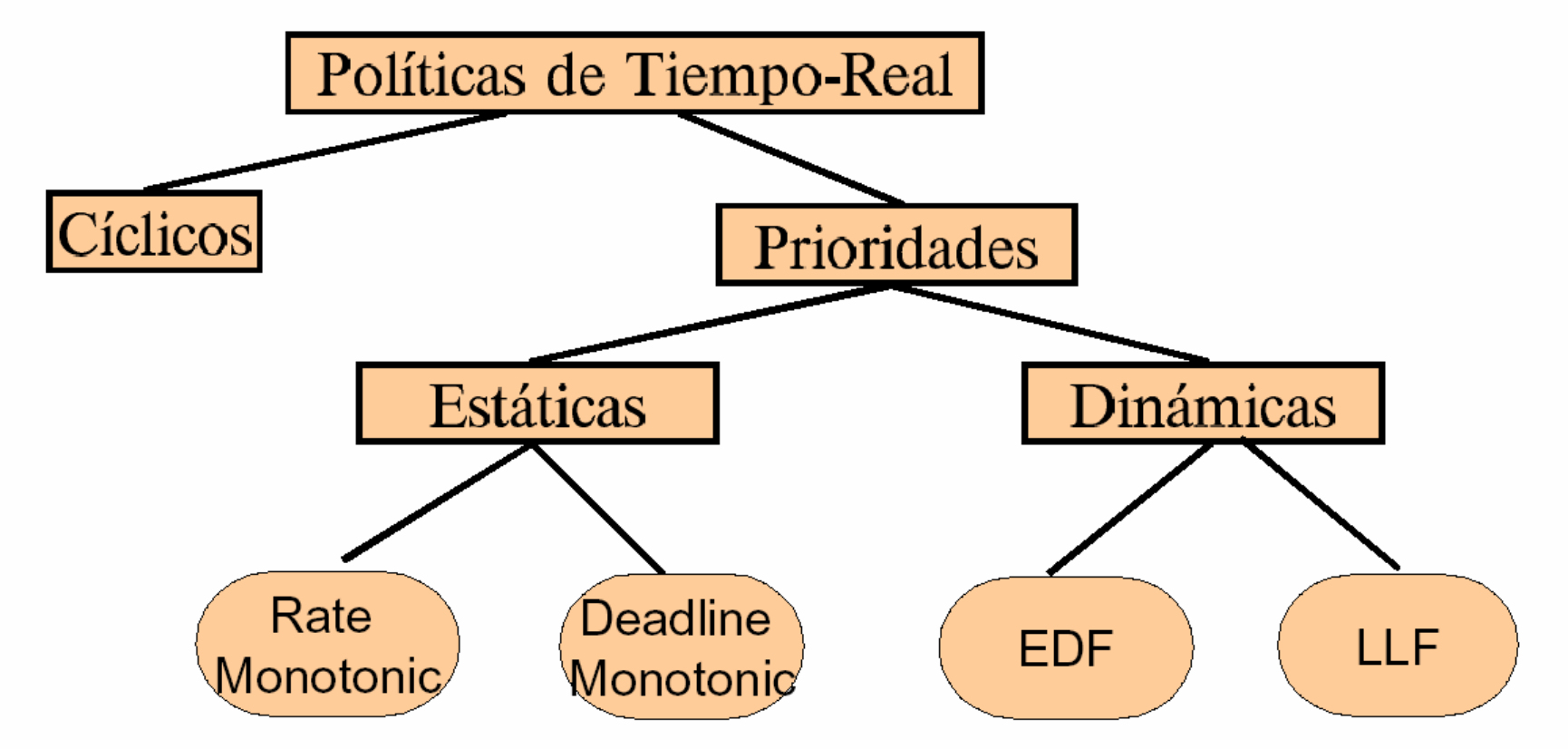
- Inicialmente supondremos las siguientes simplificaciones. Conforme estudiemos con más detalle los algoritmos de planificación, iremos eliminando estas restricciones:
  - Los procesos son independientes.
  - No comparten recursos, ni se comunican entre ellas.
  - Todas las tareas son periódicas.
  - Los tiempos de cambio de contexto son despreciables.

Secuenciamiento de Tareas

- Un *planificador* es un método para asignar recursos (el tiempo de procesador).
- Diremos que un *conjunto de tareas* es *factible* o *planificable* si existe algún planificador que sea capaz de cumplir las restricciones de todas las tareas (en nuestro caso las restricciones temporales: los plazos de ejecución).
- Un *planificador es óptimo* si es capaz de planificar correctamente cualquier conjunto de tareas factible.
- El *factor de utilización* es: $$U = \sum_{i=1}^{k} \frac{C_i}{P_i}$$
- El *hiperperiodo* es el mínimo común múltiplo de los periodos de las tareas.

Secuenciamiento de Tareas

- Clasificación de políticas de planificación:



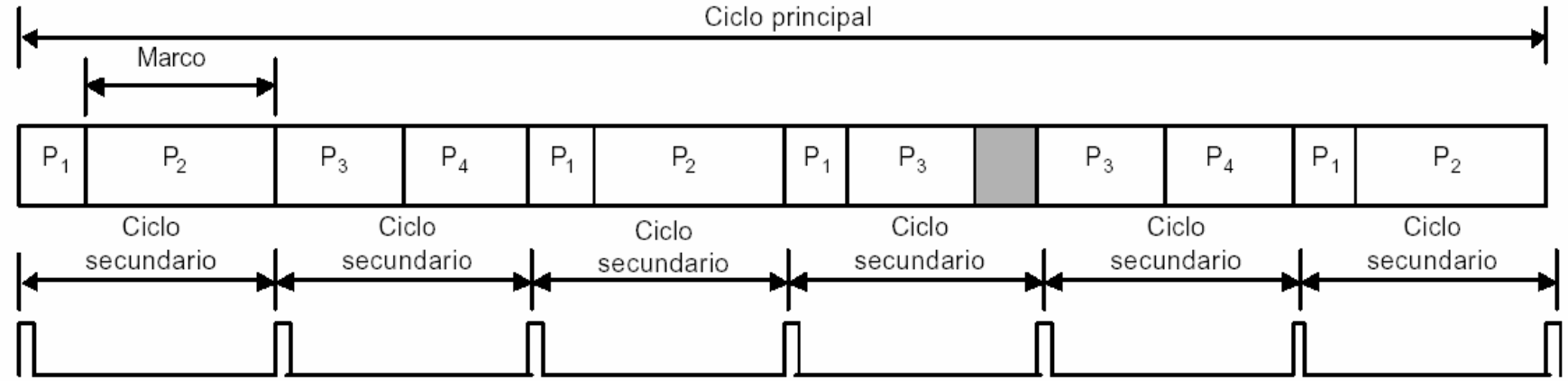
Planificación cíclica

- Ejecutar de forma iterativa un conjunto de tareas periódicas con un solo procesador.
- A partir de los requisitos temporales del conjunto de tareas, se construye un *plan principal* que define la secuencia de tareas que deben ejecutarse durante un período fijo de tiempo (*ciclo principal)*.
- El plan principal se divide en *planes secundarios*, que describen la secuencia de procesos que se deben ejecutar durante un período de tiempo fijo (*ciclo secundario*)
- Cada ciclo secundario se divide a su vez en *marcos* dentro de los cuales se ejecuta un solo proceso de la secuencia del plan secundario correspondiente.
- El instante en que termina un ciclo secundario y empieza el siguiente se sincroniza mediante el reloj de ciclo secundario.

Planificación cíclica

- Ejemplo:

Proceso	p	d	c
P_1	6	6	1
P_2	8	8	3
P_3	8	8	2
P_4	12	12	2



Planificación cíclica

- Una condición necesaria para que sea planificable el conjunto de procesos periódicos {P1,P2,…,Pn} con requisitos temporales representados por (pi,di,ci) es que la utilización del procesador *u* sea menor o igual que uno:

$$U = \sum_{i=1}^{k} \frac{C_i}{P_i} \leq 1$$

- Condición que se verifica para el ejemplo:

$$U = \sum_{i=1}^{k} \frac{C_i}{P_i} = \frac{2}{8} + \frac{3}{8} + \frac{1}{6} + \frac{2}{12} = \frac{23}{24} \leq 1$$

Planificación cíclica

- Para construir un plan de ejecución, se debe calcular la duración del ciclo principal *M* y las duraciones de los ciclos secundarios *mi*.
- La duración del ciclo principal *M* corresponde al mínimo común múltiplo de los períodos de los procesos, M=m.c.m.(pi).

▷ Resultando para el ejemplo M=24.

Planificación cíclica

- Para calcular las duraciones de los ciclos secundarios *mi*, por simplicidad, se considera que todas ellas son iguales a un cierto valor *m* que debe cumplir las siguientes condiciones (si no existe ningún valor de *m* que las verifique, el conjunto de procesos no admite una planificación cíclica):
  - Ser menor o igual que el plazo de ejecución de cualquier proceso, $\forall i(m \le d_i)$
  - Ser mayor o igual que el máximo de los tiempos de computo,
    $$m \ge \max\left(c_i\right)$$
  - Ser divisor de la duración del ciclo principal, $\exists k\left(M = km\right)$
  - Que $\forall i\left(m + \left(m - mcd\left(m, p_i\right)\right) \le d_i\right)$
    condición necesaria y suficiente para que entre el instante de activación de cada proceso y su plazo límite pueda haber un ciclo secundario completo (incluye la primera condición)

Planificación cíclica

- En el plan principal existen:
  - $n_{cs} = \frac{M}{m}$ ciclos secundarios
  - $n_{ei} = \frac{M}{p_i}$ ejecuciones de cada proceso P_i.

- Cada plan secundario está formado por una secuencia *s* de ejecuciones de procesos {Pik,Pjl,…,Prs} (*Pik* representa la ejecución *k* del proceso *i*).

Planificación cíclica

- Al aplicar las condiciones anteriores al ejemplo indicado se llega a que la duración de los ciclos secundarios puede ser m=3 o m=4:

$$\forall i (m \le d_i) \rightarrow m \le 6 \rightarrow m \in \{1,2,3,4,5,6\}$$

$$m \ge \max(c_i) \rightarrow m \ge 3 \rightarrow m \in \{3,4,5,6\}$$

$$\exists k (M = km) \rightarrow m \in \{3,4,6\}$$

$$\forall i \left(m + \left(m - mcd\left(m, p_i\right)\right) \le d_i\right) \rightarrow m \in \{3,4\}$$

Planificación cíclica

- Para m=3 se obtiene n_{cs}=8 y para m=4 resulta ncs=6. Como la complejidad aumenta con el número de ciclos secundarios se elige m=4 con lo que se tienen 6 ciclos secundarios por ciclo principal

- Por otra parte, se obtiene que el número de ejecuciones de cada proceso en el ciclo principal es n_{e1}=4 para el proceso P1, n_{e2}=3 para el proceso P2, n_{e3}=3 para el proceso P3 y n_{e4}=2 para el proceso P4.

Planificación cíclica

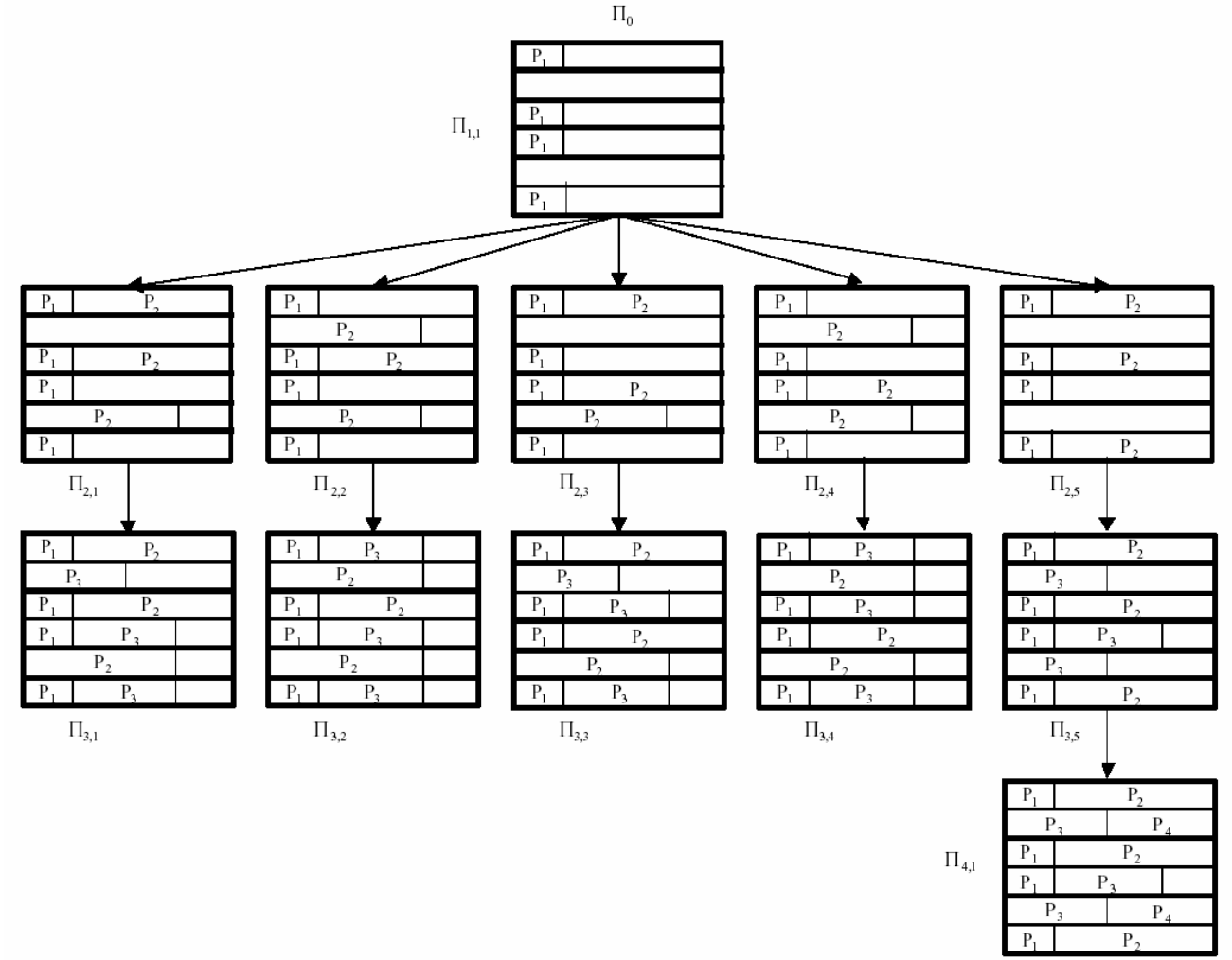
- Obtención de una planificación admisible:
  - Ordenación de las tareas: por plazo de ejecución más corto, por periodo más corto o por tiempo de computo menor.
  - Determinación de posibles ciclos secundarios para cada ejecución de cada tareas:

$$p_i\left(k-1\right) \le m(j-1) \le p_i\left(k-1\right) + d_i - m$$

  - Construir un árbol, probando todas la posibles combinaciones hasta encontrar una que sea admisible.

Planificación cíclica

- Árbol para el ejemplo:



Planificación cíclica

- Los métodos de planificación estática basados en una planificación cíclica son los comúnmente utilizados en sistemas de tiempo real críticos. Las principales ventajas de los métodos de planificación estática son:
  - El método es determinista y predecible.
  - La sobrecarga por cambios de contexto no existe en una planificación cíclica y, en general, es previsible en los planificadores estáticos. Además no es necesaria sobrecarga adicional para realizar exclusión mutua.
  - La realización es muy sencilla y la sobrecarga por planificación es muy pequeña. El planificador se limita a ir activando por turno los procesos según un plan prefijado de ejecución.

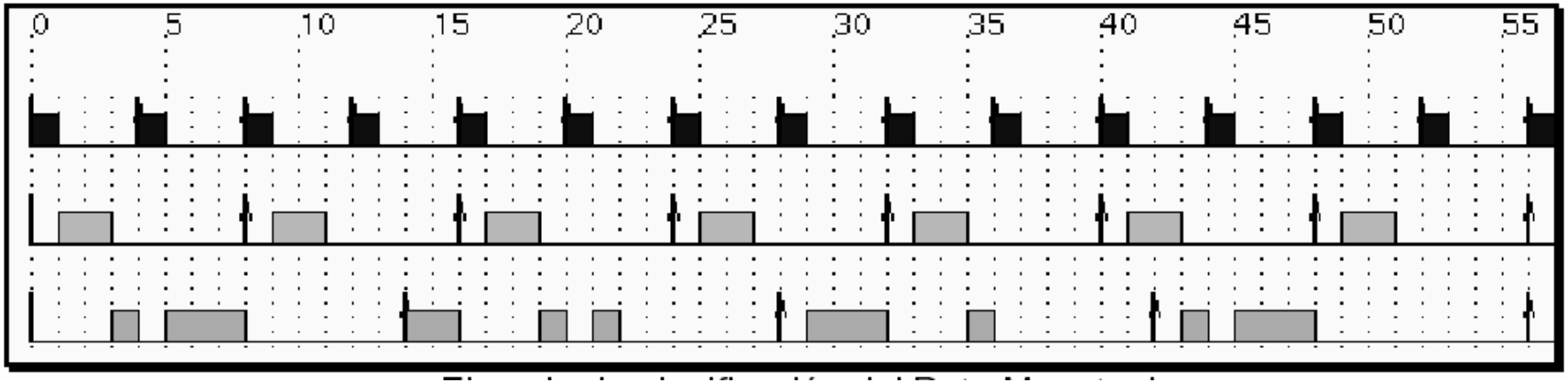
- En cuanto a las desventajas se pueden enumerar las siguientes:
  - El diseño de los planes es muy laborioso.
  - No son fáciles de mantener, ya que un cambio en el conjunto de procesos o en las restricciones temporales obliga a rehacer todo la planificación.
  - Existen grupos de procesos que no admiten una planificación cíclica factible.

Prioridades estáticas (Rate Monotonic)

- Durante la fase de diseño, a cada tarea se le asigna una prioridad inversamente proporcional al plazo de ejecución.
- Durante la fase de ejecución, el planificador selecciona aquel proceso con mayor prioridad.
- Este planificador es expulsivo, una tarea con mas prioridad expulsa a la que este ejecutándose en ese momento.
- A todas las restricciones antes mencionas, hay que añadir que el plazo máximo de ejecución ha de ser igual al periodo de activación de cada tarea

Prioridades estáticas (Rate Monotonic)

- Ejemplo de planificación con RM:
  - Representación con cronograma: tiempo de izquierda a derecha
  - En cada línea una tarea
  - Flecha hacia arriba indica la activación de la tarea



Prioridades estáticas (Rate Monotonic)

- TEST DE GARANTIA DEL FACTOR DE UTILIZACIÓN:

  Un conjunto de n tareas será planificable bajo Rate-Monotonic si se cumple que el factor de utilización del conjunto de tareas es menor que: $n\left(2^{\frac{1}{n}}-1\right)$

Esto es: $$\frac{C_1}{P_1}+\ldots+\frac{C_n}{P_n}=U\le U(n)=n\left(2^{\frac{1}{n}}-1\right)$$

La Tabla 1 representa los valores de U(n):

n	U(n)
1	1.0
2	0.828
3	0.779
4	0.756
5	0.743
∞	0.693

Prioridades estáticas (Rate Monotonic)

- TEST DE GARANTÍA DEL PLAZO DE EJECUCIÓN:

  Un conjunto de n tareas será planificable bajo asignación de prioridades si y sólo si: Cada tarea cumple su plazo de ejecución en el peor caso:

$$\forall 1 \le i \le n . W_i \le d_i$$

  donde Wi representa el instante en el que finaliza la ejecución en el peor caso:

$$W_i = c_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_i}{p_j} \right\rceil c_j$$

  donde hp(i) representa el conjunto de tareas con prioridad mayor que la tarea i.

Prioridades estáticas (Rate Monotonic)

Tal como vemos a ambos lados de la igualdad tenemos el valor Wi, que no se puede despejar. La solución de esta expresión se obtiene de forma iterativa:

$$W_i^0 = c_i$$

$$W_i^1 = c_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_i^0}{p_j} \right\rceil c_j$$

$$W_i^2 = c_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_i^1}{p_j} \right\rceil c_j$$

$$W_i^k = W_i^{k+1}$$

Si en alguna iteración se obtiene un valor de W mayor que el plazo máximo de ejecución de la tarea, entonces esta tarea no será planificable y por tanto el sistema tampoco.

Prioridades estáticas (Rate Monotonic)

- Ejemplo:

  Dado el siguiente conjunto de tareas, comprobar si es o no planificable bajo asignación de prioridades y bajo RM. Dibujar las primeras 50 unidades de tiempo del resultado de la planificación

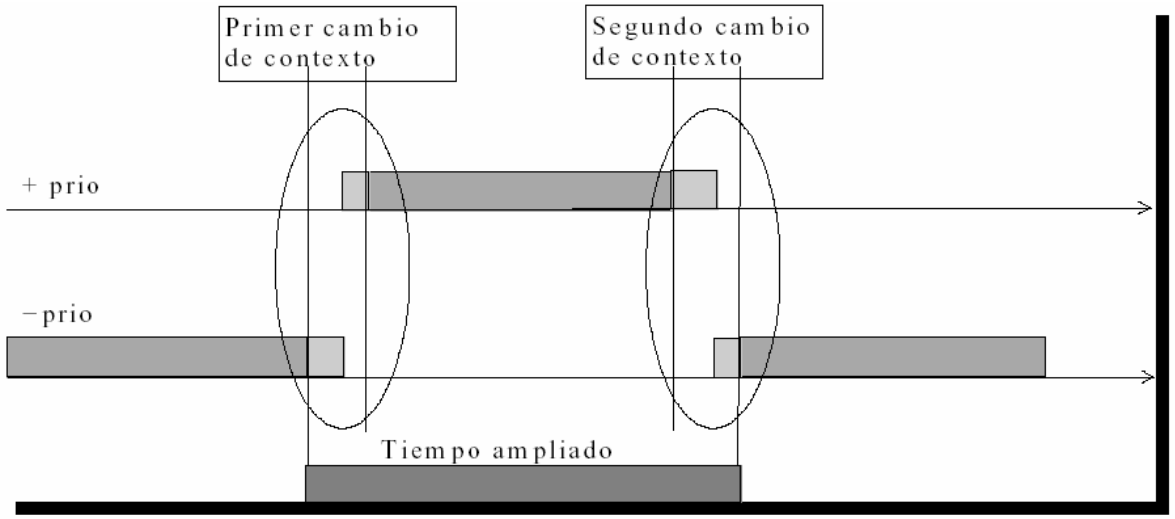
Tarea	C	P
T1	1	4
T2	2	9
T3	4	10

Prioridades estáticas (Deadline Monotonic)

- Idéntico al RM pero eliminando una restricción: las tareas pueden tener un plazo de ejecución menor o igual a su periodo. Las prioridades se asignan de forma inversamente proporcional al plazo máximo de ejecución. → RM es un caso particular del DM en el que todas las tareas tienen plazo de ejecución igual al su periodo.
- El test de garantía del plazo de ejecución es válido para cualquier asignación de prioridades.
- DM es que *es óptimo entre los planificadores basados en prioridades estáticas* (lo que no significa que sea capaz de planificar cualquier conjunto de tareas factible). En otras palabras, si el sistema no es planificable, entonces no existe otra asignación de prioridades que haga al sistema planificable.

Cambios de Contexto

- Hasta ahora no los habíamos considerado
- Procesadores actuales tiene muchos registros y salvar toda esa información en memoria requiere tiempo

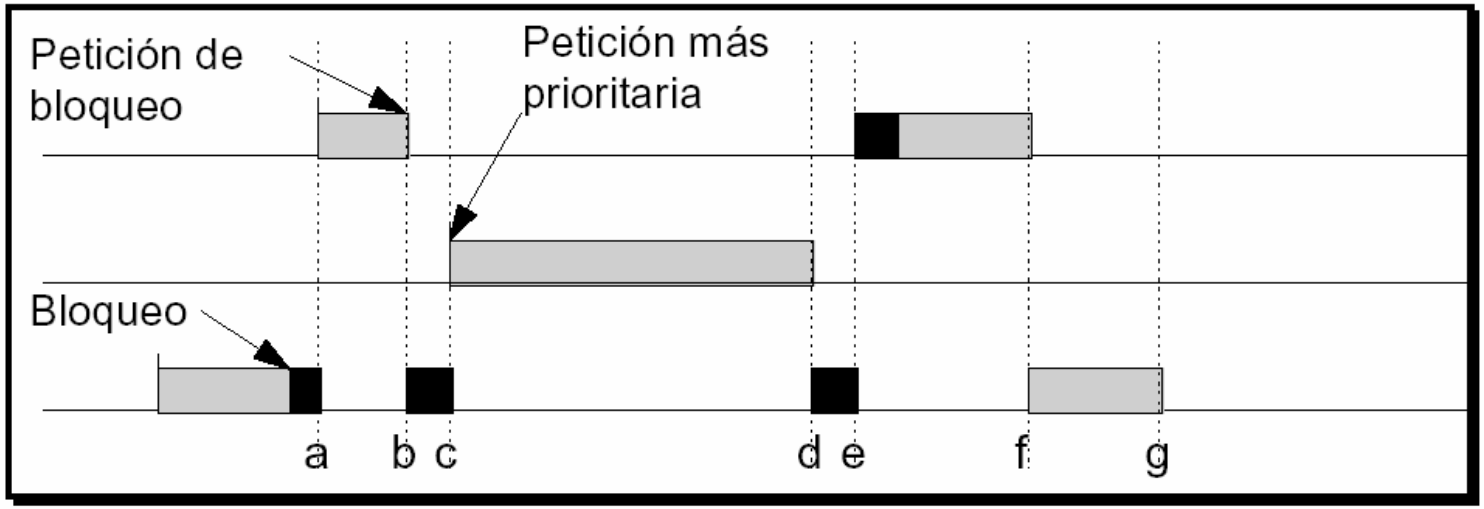


Cambios de contexto

- Los cambios de contexto han sido provocados por la tarea más prioritaria → el tiempo del cambio de contexto se le añade al tiempo de computo de ésta.
- La tarea menos prioritaria también debe tener en cuenta los cambios de contexto ya que en realidad expulsa a la tarea “idle”.
- “Idle” → tarea que no hace nada y que se activa cuando el procesador no tiene nada que hacer → ayuda a simplificar los sistemas operativos.

Uso de recursos

- Por uso de recursos entenderemos tanto la sincronización entre tareas, como la comunicación (bloqueo directo) o el uso de recursos comunes (bloqueo indirecto).
- El uso de recursos introduce el problema de la **inversión de prioridad**. La inversión de prioridad consiste en que una tarea con una prioridad intermedia puede “colarse” a una tarea más prioritaria si ésta esta bloqueada a la espera de un recurso que tiene ocupado otra tarea de baja prioridad.

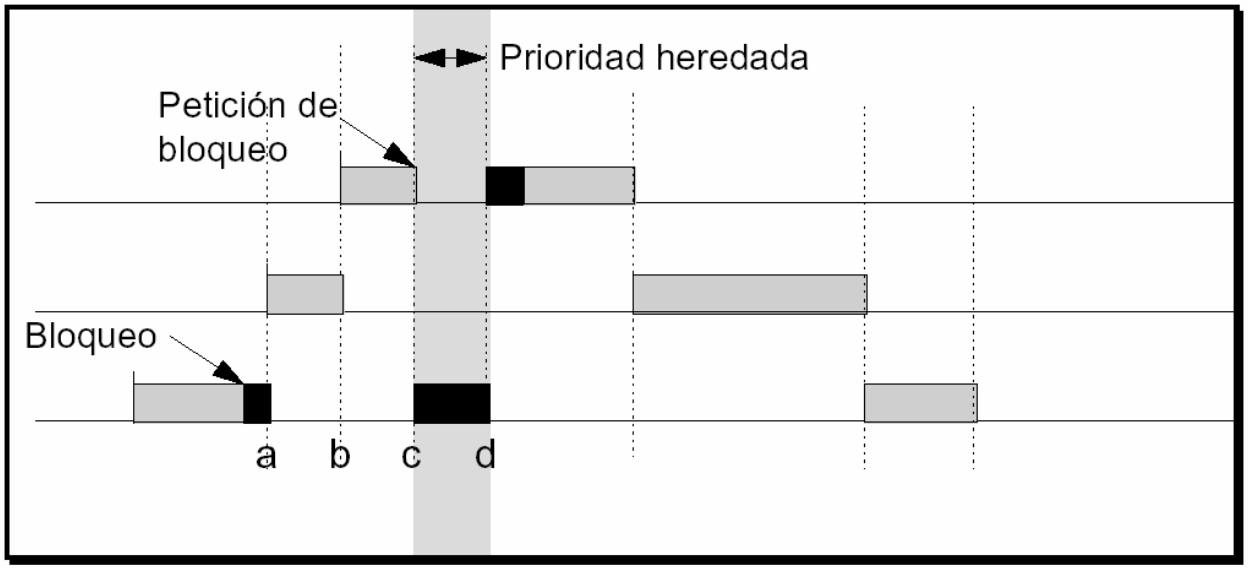


Uso de recursos

- Los métodos propuestos para solucionar este problema se pueden clasificar en tres grandes grupos:
  - EVITAR: Ordenar la ejecución de todas las tareas (off-line) con el fin de evitar que se produzcan estas situaciones. Esto se puede hacer fácilmente si tenemos un planificador cíclico.
  - IGNORAR: Tratamos a todas las secciones críticas como bloques ininterrumpibles; y hacemos que todas las secciones críticas sean lo más breves posible.
  - MINIMIZAR: Modificar la prioridad de las tareas que entren en una sección crítica, normalmente elevándoles la prioridad para que finalicen lo antes posible.
    - Priority Inheritance Protocol (PIP) (herencia de prioridades)
    - Priority Ceiling Protocol (PCP) (techo de prioridades)
    - Semaphore Inheritance (PSP)

Priority inheritance protocol

- Cuando una tarea Th intenta entrar en una sección crítica que está bloqueada (otra tarea Ti está dentro de ésta), la tarea bloqueante Ti hereda la prioridad de la tarea más prioritaria que quiere entrar Th. La herencia de prioridad sólo se produce cuando una tarea menos prioritaria bloquea a una más prioritaria. La entrada en una sección crítica NO implica herencia de prioridad.



Priority inheritance protocol

- Las ventajas que tiene este algoritmo son:
  - Una tarea de alta prioridad sólo puede ser bloqueada por una tarea de baja prioridad si ésta está dentro de una sección crítica.
  - Una tarea menos prioritaria puede bloquear a otra más prioritaria como mucho durante una sola sección crítica.
  - La sección crítica más larga determina el máximo tiempo que una tarea puede bloquear a otras de mayor prioridad.

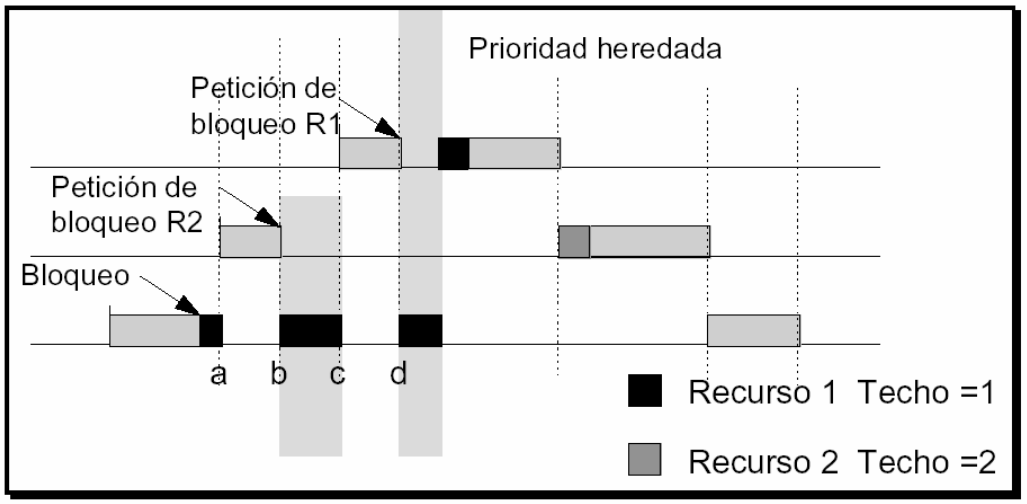
- Problemas:
  - No evita el interbloqueo.
  - Tiempo de bloqueo excesivo.

Priority ceiling protocol

- Está basado en la herencia de prioridades, pero tratando de subsanar los problemas del anterior. Las reglas de este protocolo son:
  - A cada recurso se le asigna una prioridad igual a la de la tarea más prioritaria que lo puede bloquear (techo de prioridad).
  - Una tarea puede bloquear un recurso si su prioridad es estrictamente mayor que el techo de prioridad de todos los recursos que en ese momento estén siendo utilizados.
  - Una tarea mantiene su prioridad mientras no bloquee a otra tarea más prioritaria, en cuyo caso hereda la prioridad máxima de entre las que bloquea.
  - Al abandonar la región crítica recupera su prioridad inicial.

Priority ceiling protocol

- Este protocolo tiene las siguientes propiedades:
  - Impide el interbloqueo.
  - No existen bloqueos encadenados.
  - El máximo tiempo de espera esta acotado por la sección crítica más larga de las tareas menos prioritarias.



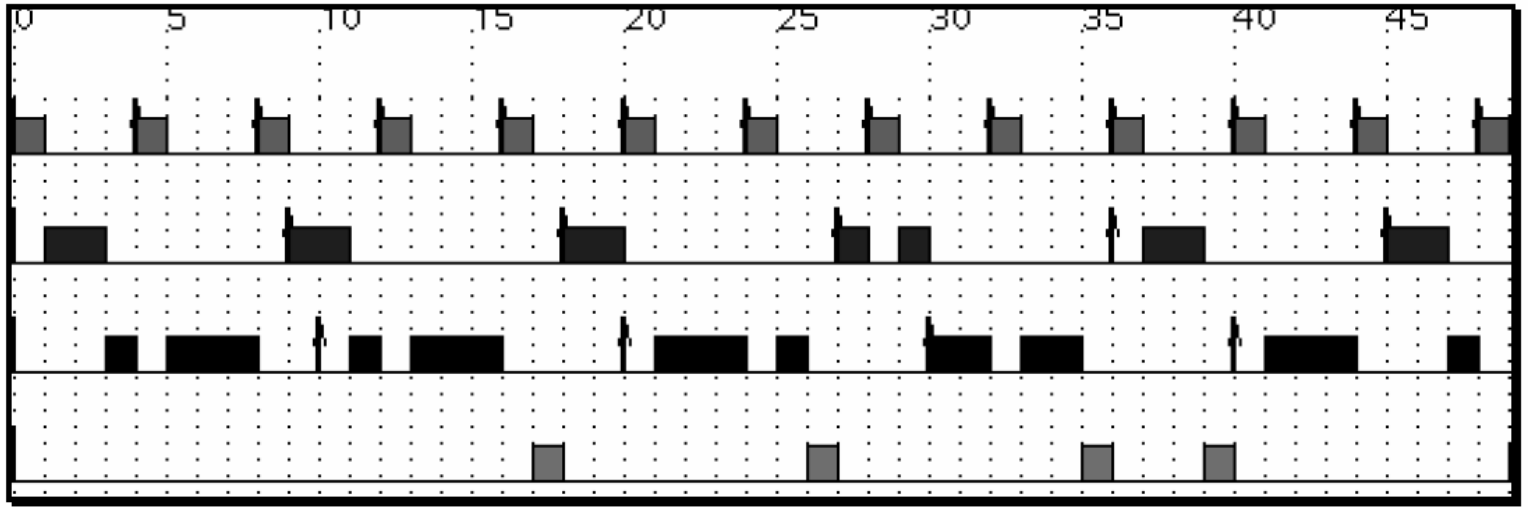
Tareas aperiódicas

- Hasta ahora nos hemos centrado en el estudio de las tareas periódicas.

- El objetivo que se pretende es el conseguir el **menor tiempo de respuesta posible**, evidentemente sin que por ello se pierda ningún plazo de ejecución de ninguna tarea crítica (periódica):
  - Servidor en background (segundo plano)
  - Polling (por consulta)
  - Deferrable server (servidor aplazable)

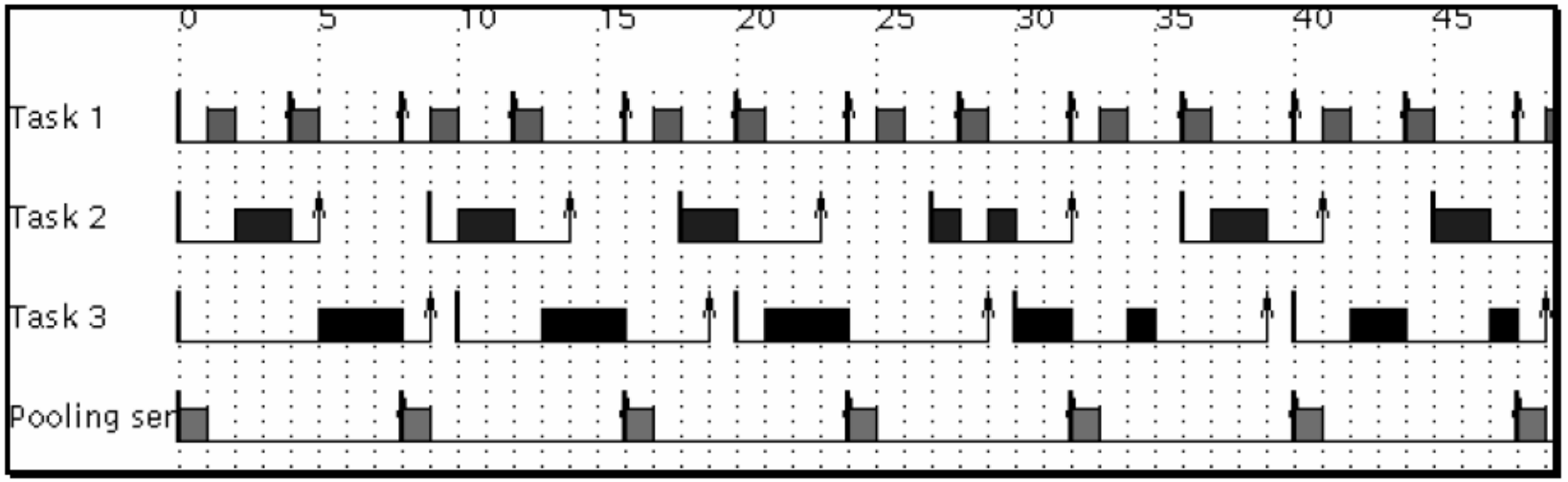
Servidor Background

- Es el más sencillo de implementar. Cuando llega un petición aperiódica no es atendida inmediatamente, sino que se deja suspendida hasta que no haya activaciones de tareas periódicas pendientes de ejecutar, entonces se atiende a todas las peticiones de trabajo aperiódico que hay esperando.
- Sustituir el proceso idle del sistema por el servidor aperiódico.
- Muy sencillo de implementar,
- Ofrece un mal tiempo de respuesta.



Pooling

- Se añade una nueva tarea periódica que consulta si hay trabajo aperiódico pendiente y lo sirve.
- Estará sirviendo trabajo aperiódico mientras le quede tiempo de cómputo o haya trabajo que servir.
- Cuando acaba de servir, **se suspende hasta que nuevamente se active en el siguiente periodo**.
- La tarea que realiza el pooling tiene asignados un periodo, un tiempo de cómputo máximo y una prioridad.
- Se puede utilizar tanto con el RM como con el DM.

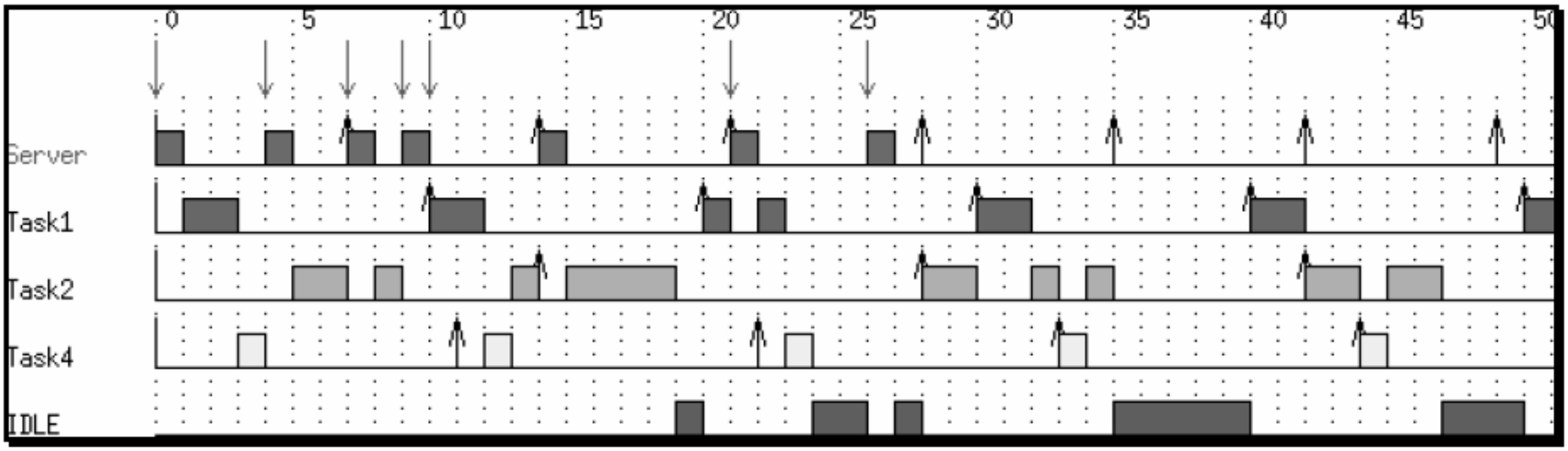


Servidor Aplazable

- Este servidor sólo puede ser empleado juntamente con el RM, pues supone que para todas las tareas Di=Pi.
- El servidor diferido se comporta como una tarea periódica de máxima prioridad:
  - Mientras le quede tiempo de cómputo disponible, el servidor atenderá las peticiones periódicas inmediatamente (pues tiene la máxima prioridad).
  - La capacidad del servidor se repone completamente al inicio de cada periodo.
- Puede ejecutar tareas aperiódicas en cualquier instante (siempre y cuando disponga de tiempo).
- El servidor puede ser considerado como una tarea "normal" a la hora de realizar el test de planificabilidad.
- Se ha tenido que desarrollar un test especifico para el deferrable server. El test nos dice cuál es el factor de utilización máximo que puede utilizar el servidor:

$$DS = \frac{C_{ds}}{p_{ds}} \leq \ln \frac{2U+2}{2U+1}$$

Servidor Aplazable



Comunicación y sincronización entre procesos

- Los *semáforos* suministran sincronización y señalización pero no contienen información
- Los *semáforos con cola* son primitivas de software que ayudan a gestionar el tráfico.
- Suministran un método para dirigir varias colas, por ejemplo, colas de tareas de espera de recursos, acceso a base de datos, o dispositivos, así como colas de recursos y dispositivos.
- Los semáforos coordinan (sincronizan) las tareas en espera con lo que estén esperando, sin dejar que las tareas o recursos interfieran entre si.

Comunicación y sincronización entre procesos

- Los *buzones* se almacenan temporalmente en lugares (bufferes) para enviar mensajes de un proceso a otro. Un proceso produce una información, la pone en el buzón y luego señala a un proceso consumidor que hay una información en el buzón para que la utilice.
  - frecuentemente se utilizan los semáforos para implementar y gestionar buzones de correo
- Un tercer método para la comunicación y sincronización entre procesos es un *sistema de mensajes*. Con un sistema de mensajes, un proceso envía un mensaje a otro.
  - Tal sistema incurre en sobrecarga debido a la transferencia real de la información, pero suministra una mayor flexibilidad y facilidad de uso.

Soporte para tiempo real

- Los sistemas que den soporte para tiempo real deben contemplar los siguientes aspectos:
  - Determinismo
    - Un sistema es determinista si realiza las operaciones en instantes fijos y predeterminados o en intervalos de tiempo predeterminados.
    - El punto hasta el cual un sistema puede satisfacer las solicitudes de manera determinista depende:
      - de la velocidad con que pueda responder a las interrupciones
      - de la capacidad para gestionar todas las peticiones en el tiempo requerido.
    - El retardo máximo que se produce desde la llegada de una interrupción de alta prioridad hasta que comience el servicio de la rutina asociada. En un sistema con soporte en tiempo real este tiempo puede ir desde unos pocos microsegundos a 1 milisegundo.

Soporte para tiempo real

- Sensibilidad
  - Es una característica semejante a la anterior, hace referencia a cuanto tiempo consume un sistema en reconocer una interrupción, es el tiempo preciso para dar servicio a la interrupción después de haberla reconocido. Depende de:
    - La cantidad de tiempo necesaria para iniciar la gestión de la interrupción y empezar la ejecución de la rutina de tratamiento (ISR *Interrupt Service Routine*). Si la ejecución de la ISR requiere un cambio de proceso ese tiempo será mayor.
    - El efecto de anidamiento de las interrupciones. El servicio se retrasará si el sistema debe atender la llegada de otra interrupción más prioritaria.
    - El determinismo y la sensibilidad forman conjuntamente el tiempo de respuesta a sucesos externos.

Soporte para tiempo real

- Control del usuario
  - Es generalmente mucho mayor en un sistema con soporte para tiempo real que en uno de tiempo compartido. En estos últimos un usuario no puede otorgar prioridades a sus procesos, decidir sobre el algoritmo de planificación, qué procesos deben estar siempre residentes en memoria etc.
- Fiabilidad
  - Es normalmente mucho más importante en un sistema con soporte para tiempo real. Un sistema en tiempo real controla sucesos que están teniendo lugar en el entorno y en su propia escala de tiempos, las perdidas o degradaciones en el sistema que los controla pueden tener consecuencias catastróficas.

Soporte para tiempo real

- Tolerancia de fallos
  - Un sistema con soporte para tiempo real debe diseñarse para responder incluso ante varias formas de fallo, se pretende que se pueda conservar la capacidad máxima y los máximos datos posibles en caso de fallo. Opciones como la de volcar el contenido de la memoria a un archivo y abortar el programa ante la aparición de un fallo están totalmente prohibidas.
  - Un sistema con soporte para tiempo real intentará corregir el problema o minimizar sus efectos antes de proseguir con la ejecución.
  - Asociada a la tolerancia a fallos está la estabilidad. Un sistema será estable si en los casos es los que es imposible cumplir todos los plazos de ejecución de las tareas se cumplen al menos los de las más críticas y de mayor prioridad.