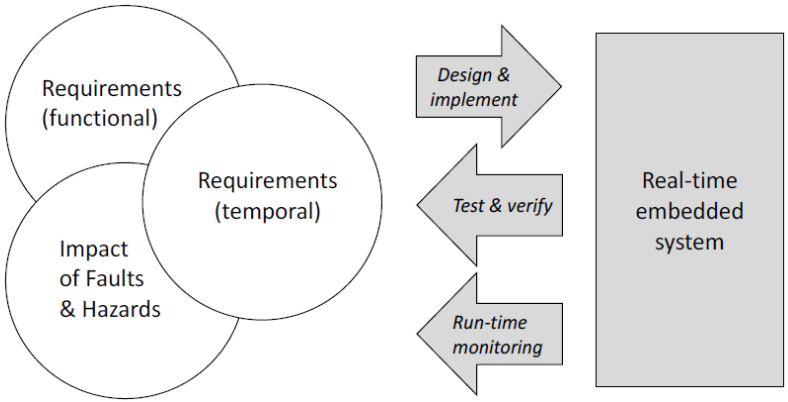
Sistemas gobernados por tiempo (TDS) Conceptos

Agenda.

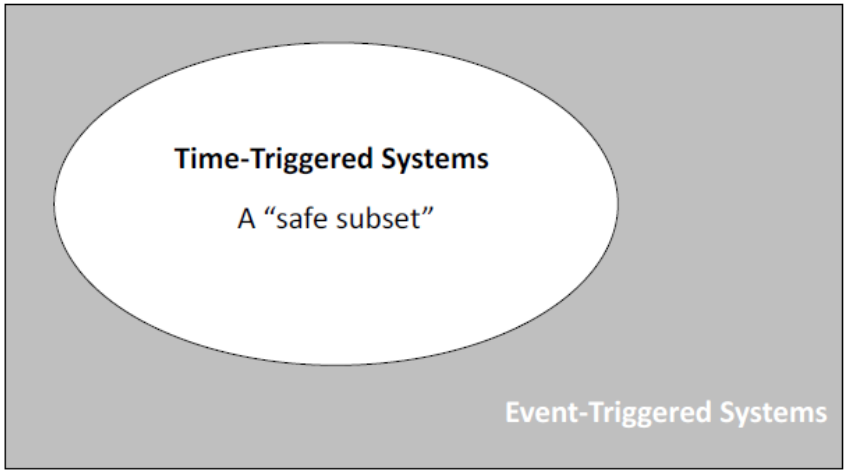
- Sistemas embebidos Confiables.
- Arquitectura TDS vs. EDS.
- Ticks de Sistema.
- Planificador TDS.
- Modos de sistema.
- Chequeos en tiempo de ejecución.
- Modelo de tareas cíclico.
- Hiperperíodo.
- Planificación estática.
- Factor de utilización.

Sistemas embebidos confiables



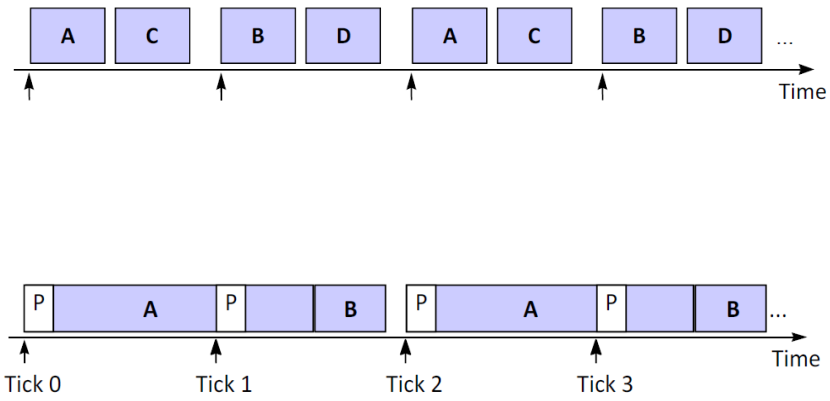
- La ingeniería de un sistema confiable requiere la definición, verificación y validación de ***requerimientos funcionales y temporales y el impacto de las fallas del sistema.***
- En este tipo de sistemas es de vital importancia el ***procesamiento determinístico*** que implica que se pueda garantizar que una actividad particular del sistema siempre se completará dentro del mismo intervalo de tiempo.
- Un sistema confiable que no cumple los requerimientos (por ejemplo que responda más lentamente de lo especificado) ***no es útil para esa función.***

Arquitectura TDS.



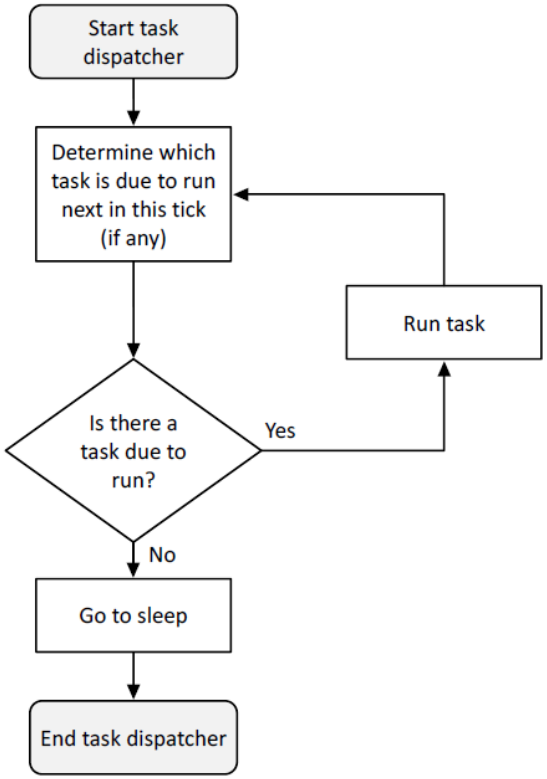
- Los sistemas gobernados por tiempo sólo utilizan una única interrupción (de temporización) para ejecutar un planificador.
- El planificador suele ser estático y cooperativo (las tareas están definidas por el/la desarrollador/a y no se interrumpen entre ellas)
- Debido a las restricciones impuestas por los sistemas TDS ***es posible modelar el comportamiento del sistema de manera precisa y determinar si se cumplen los requerimientos temporales***.

Sistemas TDS



- Un sistema TDS ejecuta una lista de tareas predeterminadas.
- El sistema TDS se va a regir por el “tick de sistema” que se implementará con un timer del microcontrolador. Valores usuales de este “tick” pueden ser:
  - 1ms
  - 100µs
  - 25ms.
- La elección del tick de sistema va a ser una relación de compromiso entre el tiempo de respuesta y el tiempo que tarda el planificador en cambiar de una tarea a otra.

Planificador TDS Cooperativo.



Modos del sistema principales

Modo normal

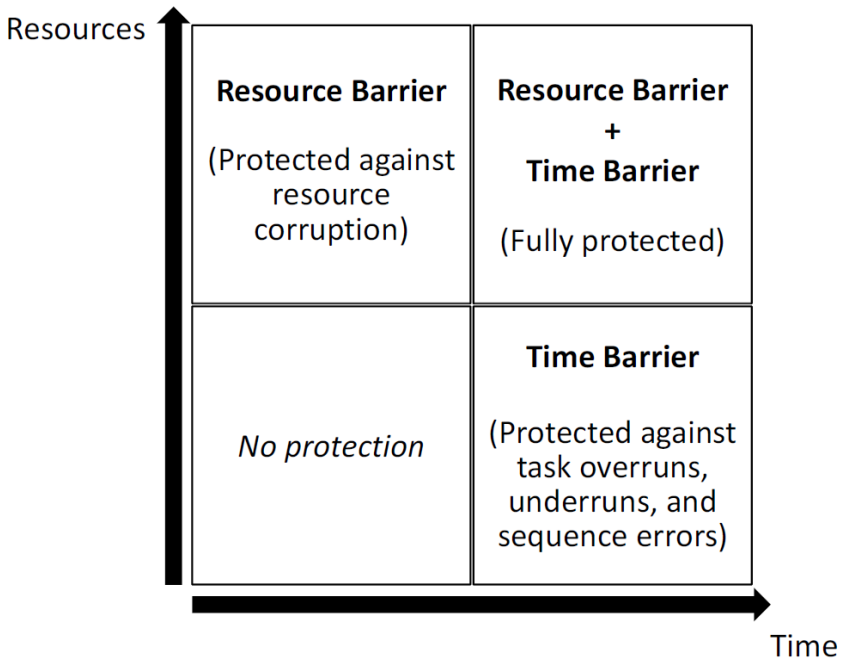
- Es el modo por defecto donde el sistema está completamente operacional y se alcanzan todos los requerimientos esperados sin que el sistema cause daños o riesgos.

Modo de falla controlada.

- Es el modo en el que el sistema detecta alguna condición anómala y hay que apagar el sistema llevandolo a una condición de que minimice el daño o no lo produzca.

Chequeos en tiempo de ejecución

- Construir un sistema manejado por tiempo involucra:
  - Modelar el sistema como una lista de tareas y su verificación.
  - Seleccionar el tipo de planificador para la aplicación (vamos a plantear cooperativo)
  - Generar chequeos en tiempo real para comprobar que el hardware funcione de acuerdo a requerimientos

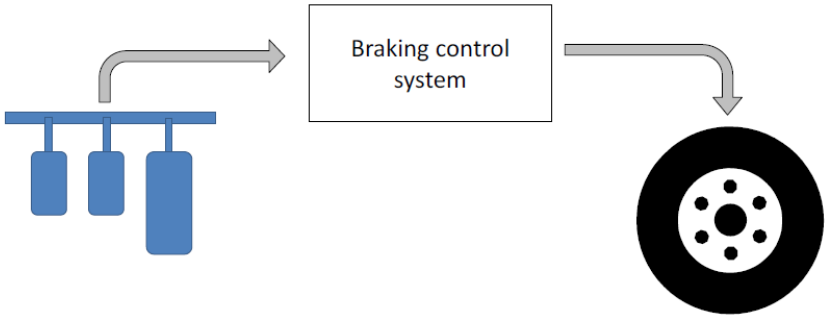


Chequeos en tiempo de ejecución.

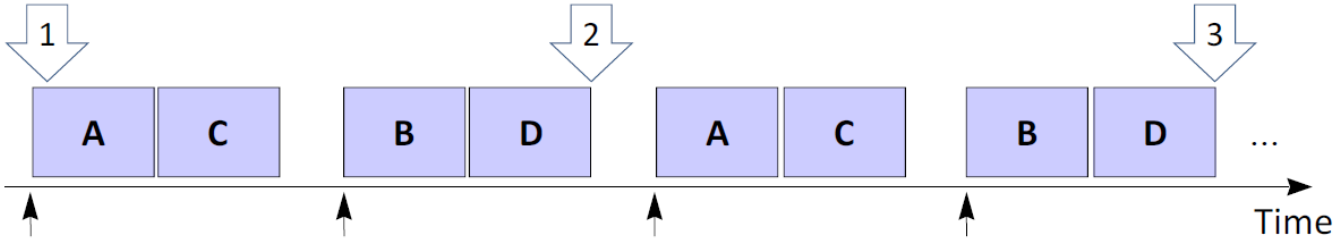
- Los chequeos en tiempo de ejecución son fundamentales para los sistemas de alta disponibilidad o confiabilidad.
  - ***Chequeos en el arranque.*** (***P***ower ***O***n ***S***elf ***T***est). El fin de este chequeo es verificar la correcta configuración del sistema.
  - ***Temporizador de Watchdog.*** Es un temporizador que al alcanzar su conteo máximo resetea al procesador. Se usa reseteándolo periódicamente. En caso de que haya algún problema este temporizador reinicia el sistema.
  - ***Monitor de tareas***. El monitor es un temporizador (puede ser uno de uso general) que verifica que las tareas se ejecuten al menos un tiempo mínimo (BCET) y menos de un tiempo máximo estipulado (WCET).

Tiempo de respuesta

- Independientemente de la arquitectura interna del sistema todos los sistemas en tiempo real necesitan responder a eventos.
- Una consideración muy importante es el tiempo que tarda el sistema desde que sucede un evento hasta que responde al mismo.



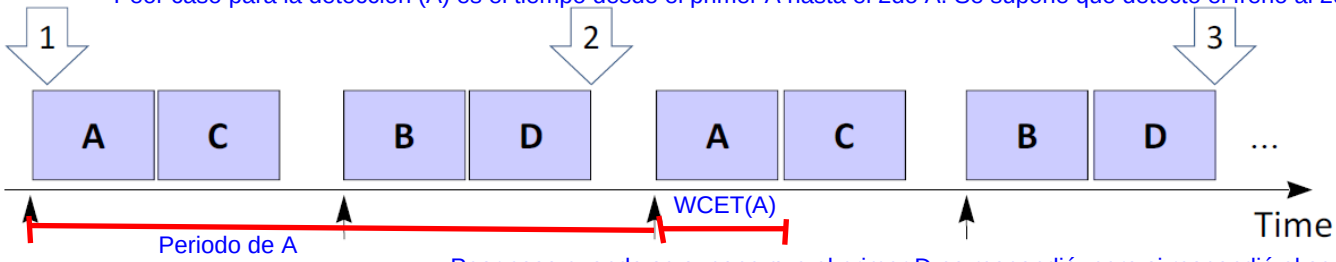
Tiempo de respuesta



- Para este ejemplo, vamos a considerar que la tarea “A” monitorea el pedal y la tarea “D” actúa sobre el pedal de freno.
- Suponemos que en 1 se detecta que se presionó el pedal y que 2 ya se actuó el freno.
- Generalmente se suele tomar el caso de que el evento se detecta en el período siguiente.

Tiempo de respuesta

Peor caso para la deteccion (A) es el tiempo desde el primer A hasta el 2do A. Se supone que detectó el freno al 2do A

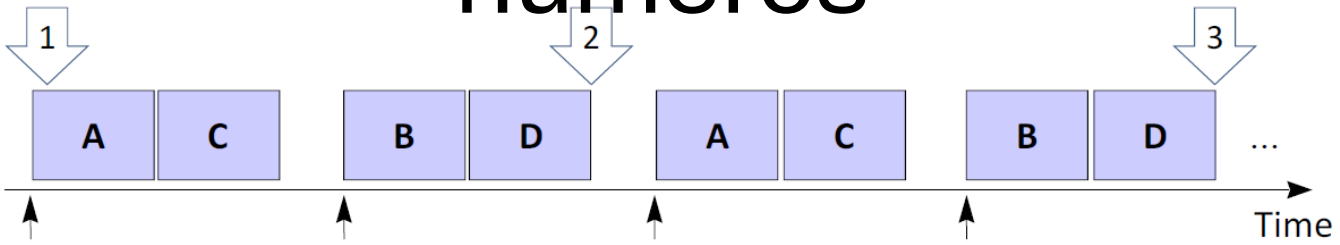


Peor caso cuando se supone que el primer D no respondió, pero si respondió el segundo tick de D

- Para sistemas estrictamente periódicos, se va a considerar que el tiempo de que va a tardar la tarea “A” en detectar un evento es:
  - Periodo(A) + WCET(A). WCET(A) es el tiempo maximo que puede demorar la tarea A
- Para este sistema y suponiendo que la de la figura es la lista completa de tareas:
  - Periodo(A)+WCET(A)+Tick + WCET(B)+WCET(D)

si el tiempo de respuesta ttr = 50ms es mayor que el (periodo(A)+WCET(A)+tick+WCET(b)+WCET(D)) está todo OK

Tiempo de respuesta. Poniendo números



- Si suponemos que se requiere un tiempo de respuesta de 50ms con los siguientes datos:
  - Tick de sistema 10ms
  - WCET(A) = 4ms
  - WCET(B) = 4ms
  - WCET(D) = 5ms
- ¿Cumple este sistema con el tiempo requerido?

Modelando la carga de CPU

Task	Period (ms)	WCET (ms)	CPU load (%)
A	5	2	40%
B	10	4	40%
C	20	3	15%
			95%

- La carga de CPU es un parámetro fundamental para cualquier embebido, ya que describe el uso del tiempo de sistema que es un recurso que va a determinar fuertemente la selección de un procesador.
- La carga de CPU que impone una tarea será el cociente entre el peor tiempo de ejecución de cada tarea (WCET) y el período de la misma.
- La suma de la carga de todas las tareas es la carga del sistema que para poder ser planificable debe ser menor al 100%.

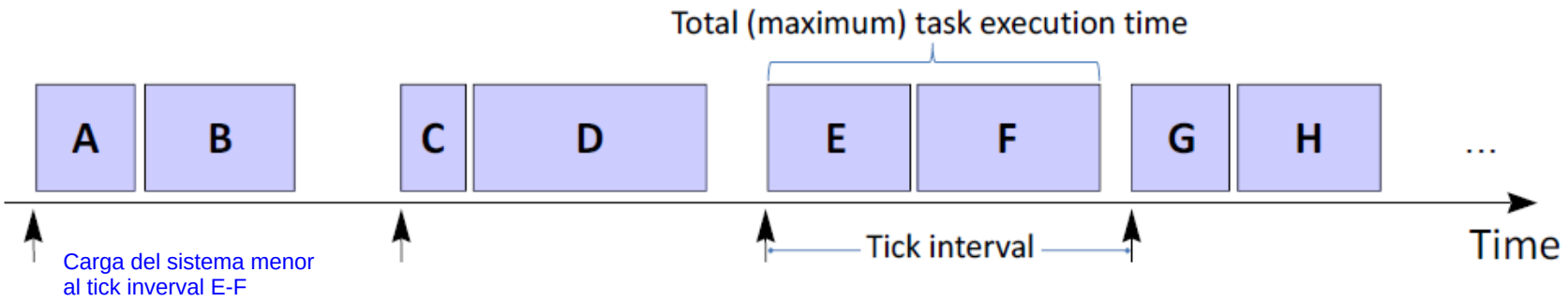
Carga que ocaciona la tarea al CPU = Peor tiempo de ejecucion de la tarea / Periodo de la tarea

Tick = maximo comun divisor de todos los periodos de las tareas

Siempre se trata de dejar un 10% libre del CPU, por si las moscas

- Este simple chequeo, lo podemos considerar un chequeo “preliminar”, pero no da detalle por tick de sistema.

Carga de CPU



- Para tener más información de la carga de CPU es necesario analizar cada tick del hiperperíodo.
- Carga de $CPU_{(por\ tick)}$= WCET/tiempo de Tick

Carga de CPU. Ejemplo.

la lista de tareas se repite cada 400ms! 20ticks * 20ms c/u

		10ms		
17ms	[Tick 0]	[Tick 7]	[Tick 12]	17ms
	Task A	Task C	Task A	
	Task B	Task C	12ms Task B	
	Task C	[Tick 8]	Task C	
	[Tick 1]	Task A	[Tick 13]	
3ms	[Tick 2]	Task B	5ms [Tick 14]	13ms
	Task B	[Tick 9]	Task B	
5ms	[Tick 3]	Task C	18ms Task C	
	Task C	[Tick 10]	Task C	
12ms	[Tick 4]	Task B	[Tick 15]	
	Task A	Task C	[Tick 16]	5ms
	Task B	Task C	Task A	
3ms	[Tick 5]	Task C	Task B	
	Task C	[Tick 11]	[Tick 17]	5ms
3ms	[Tick 6]		Task C	
	Task B		[Tick 18]	3ms
			Task B	
			[Tick 19]	

No habria que dejar Ticks Vacios. Hay que distribuir bien las tareas. Sin pasarme de carga del CPU.

En la lista tengo 5 task A que duran 9ms por lo tanto 5*9ms/400ms es el porcentaje de la tarea A en el total de los 400ms --->11.25%
Tarea B ---> 6%
Tarea C ---> 16.25%

Nosotros vamos a tener que diseñar el Scheduler.

- Dada la lista de tareas de la izquierda, determine la máxima carga de CPU y la carga media de CPU.
- Suponga:
  - $WCET_A$ = 9ms
  - $WCET_B$ = 3ms
  - $WCET_C$ = 5ms
  - Tick = 20ms
  - Carga del planificador 2%

+ 2% de la carga del plaificador

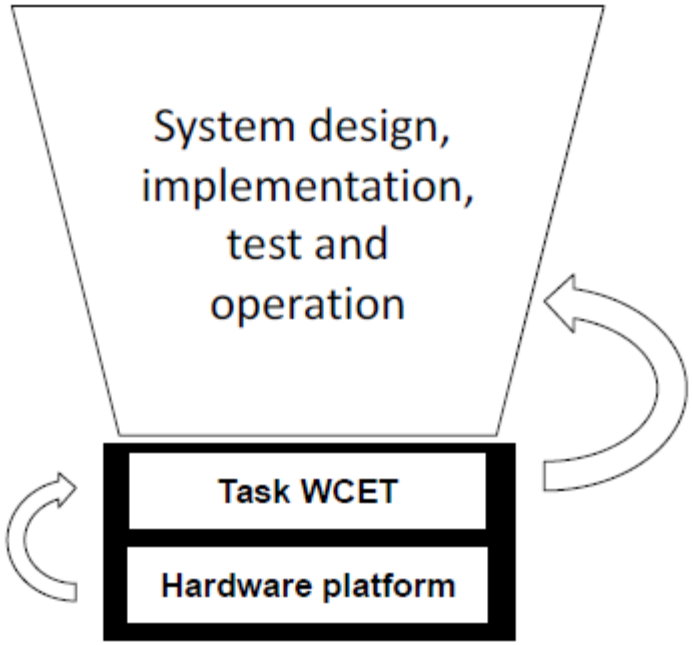
18ms / 20ms = 92% del uso del procesador
Pico de Carga...Lo que hay que cumpli es la carga Media
La peor combinacion serian 3 veces la tarea A?? se pasa del tick de 20ms

20ms es el 100% del CPU.

La suma da la carga media ---> 33.5%
Ésto es lo que tengo que cumplir.

¿Cómo conocer el WCET y BCET?

- El peor caso de tiempo de ejecución (WCET) de cada tarea es fundamental para el modelado de un sistema.
- Conocer el mejor caso de tiempo de ejecución (BCET) también ayuda a monitorear el sistema. Un BCET de 0 probablemente indique que una tarea no esté haciendo nada.
- Los tiempos de ejecución es complicado modelarlos (ya que implica un conocimiento detallado del compilador y el hardware) por lo que para nuestra aproximación los mediremos



DWT

- Los Cortex M3 tienen una unidad llamada Data Watchpoint y Trace que tiene hardware dedicado para el “profiling” o análisis de rendimento del sistema implementado.
- La unidad DWT tiene cuatro comparadores y el contador de ciclos de programa CYCCNT que es un contador ascendente de 32bits que se incrementa cada ciclo de reloj. Se puede leer, borrar, prender, apagar, cuando se te cante. Es parte del Core.
- El CYCCNT se puede leer y escribir en cualquier momento y se habilita a través del registro DWT_CTRL

```
//Inicializo el Contador de Ciclos de reloj
DWT->CTRL |=DWT_CTRL_CYCCNTENA_Msk;
DWT->CYCCNT=0;
```

```
init  main (){
int wcet = 0; int et;
inicializacion HW ();
dwt->CTRL |=dwt_ctrl_cyccntena_msk;
while(1){
dwt->cyccnt = 0;
if(!== contador){
contador = valor;
invertirpin();
}
et = dwt->cyccnt;
it(wcet<et)=et; //para saber el peor caso
--wfi();
}
return
}
```

N

wcft=N/SystemCoreClock

Manos a la obra

- Tomar el ejemplo del led parpadeante https://gitlab.frba.utn.edu.ar/digitales2/blinking.git que duerme el procesador y modificarlo para medir el WCET y la carga del procesador.

- Una vez calculado, cambiar la interrupción del Systick a 100 µs, luego 10 µs y recalcular.

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

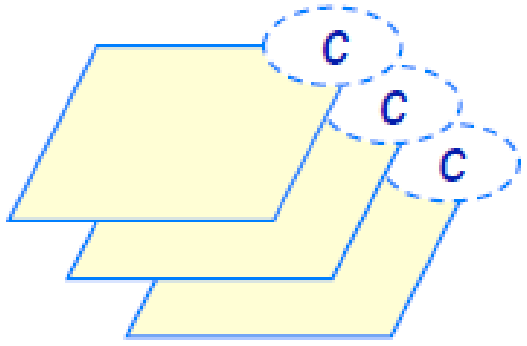
Comenzar con modelos sencillos: tareas periódicas independientes

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

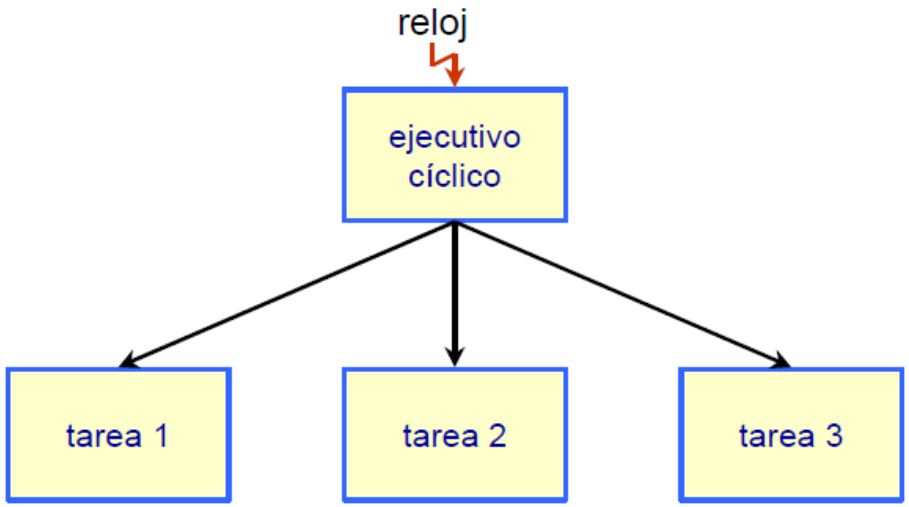
Inicialmente consideramos que no hay comunicación entre tareas (tareas independientes)



Arquitectura Sincrónica/Asincrónica

Las tareas se ejecutan según un **plan de ejecución** fijo (realizado por el diseñador)

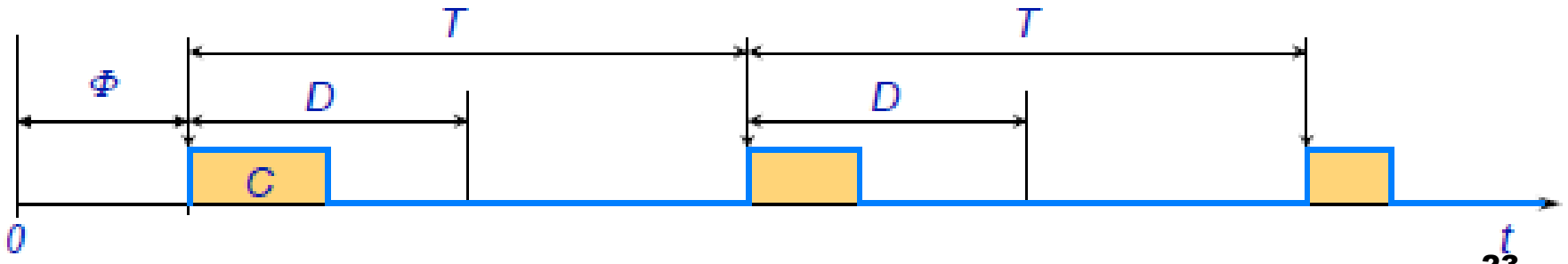
El sistema operativo se reemplaza por un **ejecutivo cíclico**



Parámetros Temporales

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

Φ	es la fase
T	es el período de activación de la tarea (Minimum time between process releases - process period)
C	es su tiempo de procesamiento en el peor caso (Worst-case computation time -WCET- of the process)
D	es el plazo de respuesta relativo a la activación (Deadline of the process)



Elegir el tick de sistema

Task ID	Period (ms)	WCET (µs)	Offset (ticks)
A	1	200	0
B	2	300	0
C	2	200	1
D	2	150	1
E	3	200	11
F	5	100	13
G	7	50	25

- Dada una lista de tareas el tick de sistema lo va a dar el mayor divisor común entre los períodos de las diferentes tareas.
- En la tabla de la izquierda es de 1ms.

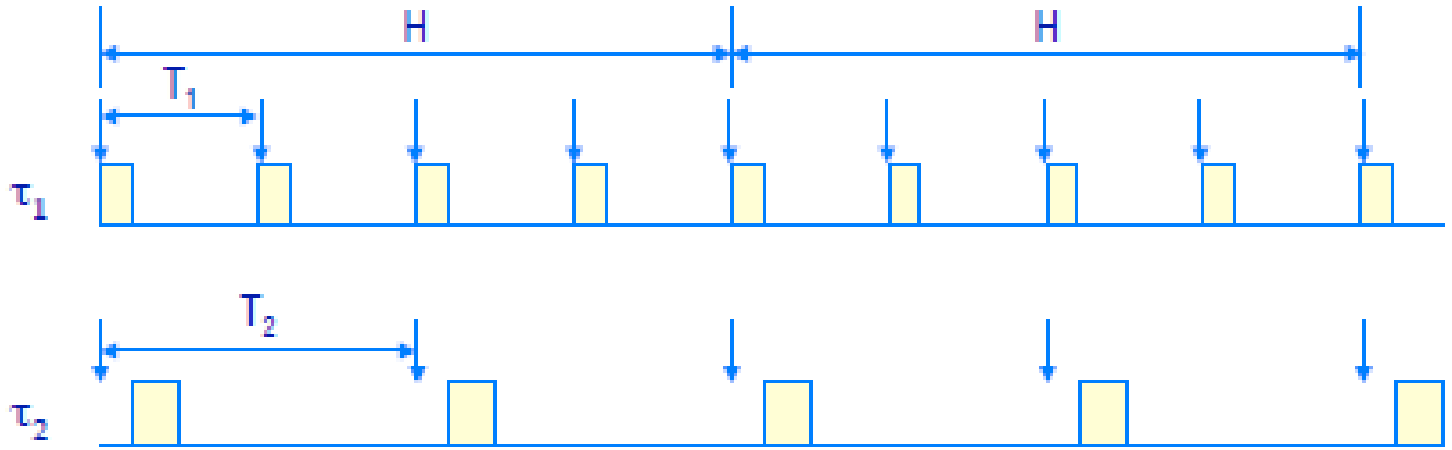
no todas las tareas se ejecutan en todos los ticks

Hiperperíodo : Cuando se vuelve a repetir la lista completa (los 20 ticks de unas diapos. anteriores)

En un sistema formado únicamente por tareas periódicas con períodos **Ti, i = 1 .. N**, el comportamiento global se repite con un período:

H = mcm (Ti)

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:

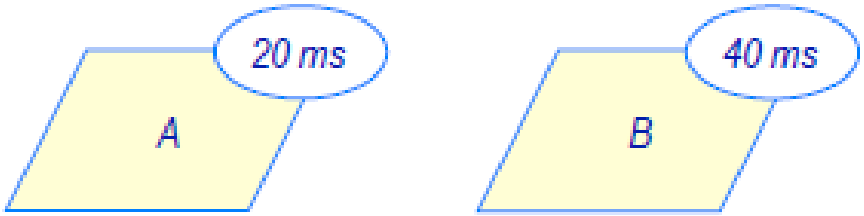
TM = mcm (**Ti**) ciclo principal (hiperperíodo del sistema)

El ciclo principal se divide en ciclos secundarios, con período:

TS (**TM** = K **TS**)

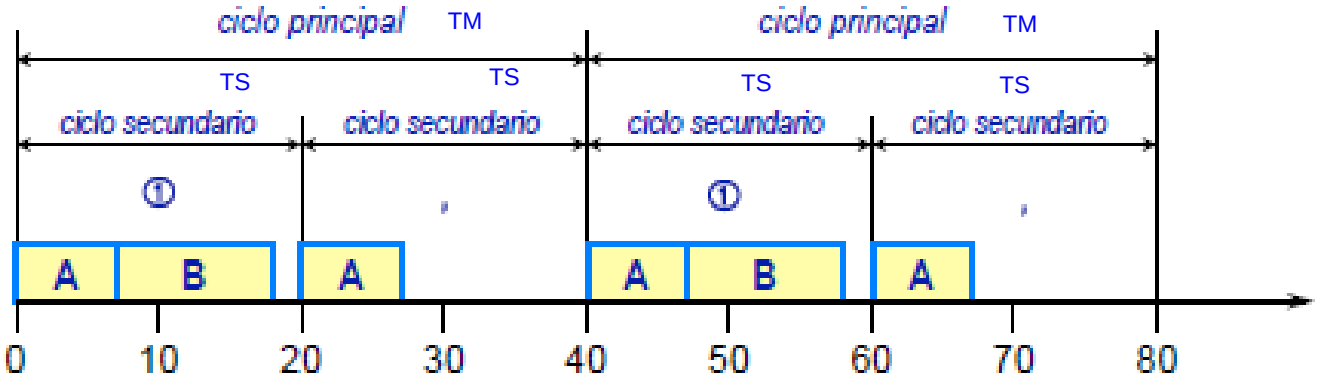
En cada ciclo secundario se ejecutan las actividades correspondientes a determinadas tareas

Planificación Estática: Ejemplo 1



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

hiperperiodo, cada 40ms deben realizarse todas las tareas.



Planificación cíclica estática

- La principal ventaja de los sistemas gobernados por tiempo es que se puede determinar la planificación de las tarea del sistema de manera determinística. Para todas las condiciones del sistema.

WCET

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

Periodo de la tarea

TM -> hiperperiodo $T_M = mcm(T_i)$

TS -> es el tick secundario $T_S = mcd(T_i)$

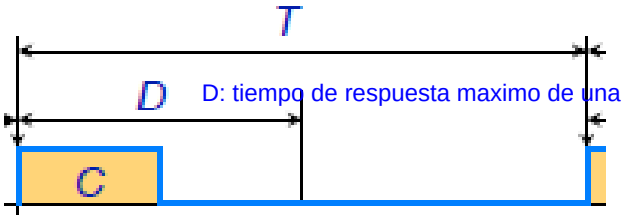
- Para generar la planificación del sistema se necesita:
  - El período de activación de cada tarea (asumiendo un plazo de terminación igual al período)
  - El peor tiempo de ejecución de cada tarea.
- Se calcula el factor de carga de CPU (U), debe ser menor a uno para que el sistema sea planificable.
- El tick de sistema se va a calcular como el ***máximo común divisor de todos los períodos de las tareas***.
- El hiperperíodo se va calcular como ***el mínimo común múltiplo de todos los períodos de las tareas***.
- Se arma la lista de períodos secundarios donde se debe respetar el factor de carga menor a uno para cada ciclo secundario.

Planificación cíclica estática. Ejemplo

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3	80	**40**	80

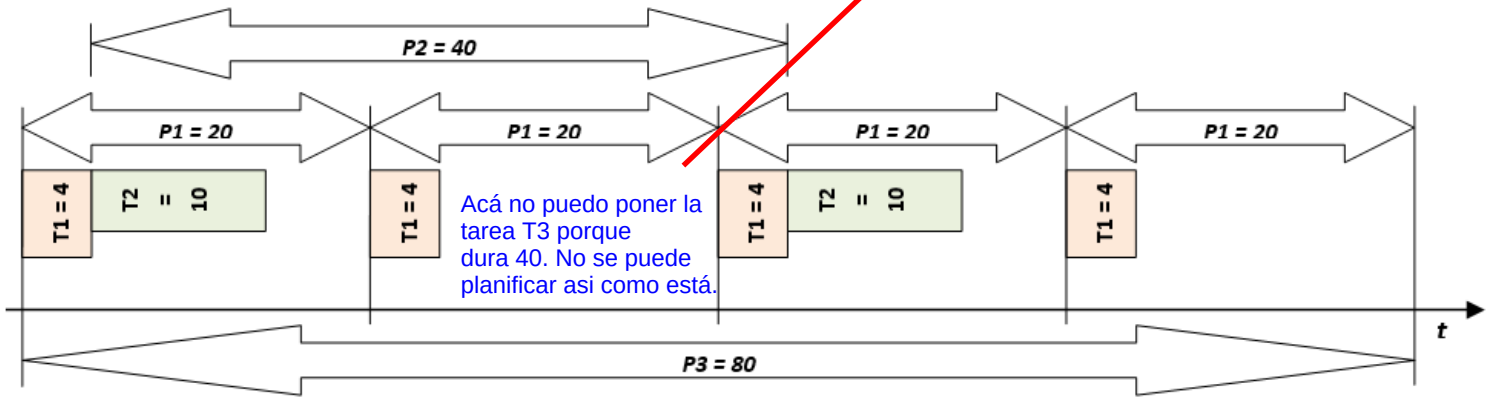
Sumatoria de la diapo anterior

- U = 0,95 tiene que ser <= 1
- $T_M = mcm(T_i) = 80$
- $T_S = mcd(T_i) = 20$
- Para poder planificar todas las tramas se necesita en tick de 20ms pero el peor tiempo de tarea 3 es de 40ms.
- ***No se puede planificar el sistema***



D: tiempo de respuesta maximo de una tarea.

La solucion es Segmentar la tarea
La divido en 2 de 5 y 2 de 15



Acá no puedo poner la tarea T3 porque dura 40. No se puede planificar asi como está.

el hiperperiodo tiene 4 ticks secundarios

Segmentación de Tareas/Problemas

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)

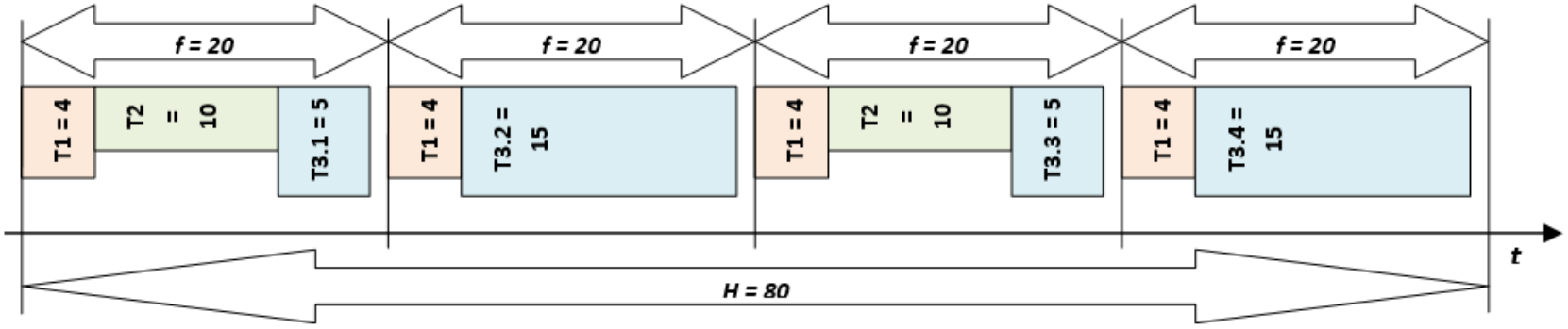
A veces es difícil ajustar el tiempo de cómputo de los segmentos

Si hay recursos compartidos, cada sección crítica debe estar incluida en un solo segmento

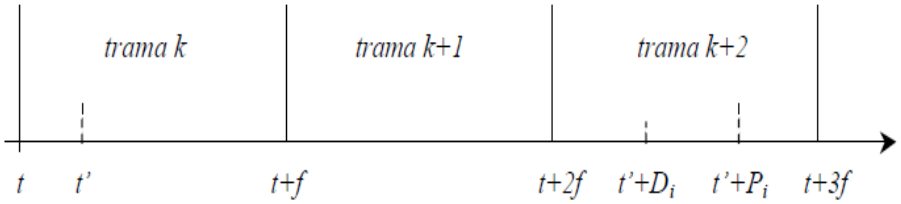
Si se modifica una sola tarea hay que rehacer la planificación completa (y posiblemente volver a segmentar de otra manera)

Planificación de la tarea

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3.1	**80**	**5**	80
T3.2	**80**	**15**	80
T3.3	**80**	**5**	80
T3.4	**80**	**15**	80



Planificación cíclica estática (2)



Osea, Osea, Osea, Osea

1. $T_s \geq máx(C_i)$
2. $U = \sum_1^N \left(\frac{C_i}{T_i}\right) \leq 1$
3. $T_M = mcm(T_i)$
4. $\exists i: \frac{P_i}{T_s} - \left\lfloor \frac{P_i}{T_s} \right\rfloor = 0$
5. $\forall i: 2T_s - mcd(T_s, P_i) \leq D_i$

- La ecuación 1 intenta que toda tarea se pueda terminar en un único período secundario.
- La ecuación 2 determina el factor de carga de la CPU al igual que en la planificación anterior.
- La ecuación 3 nos determina el hiperperíodo.
- La ecuación 4 nos dice que ***debe existir al menos una tarea que tenga período que sea múltiplo entero del período secundario***.
- La ecuación 5 determina que una tarea tiene que terminar antes de un nuevo período de activación.

Planificación cíclica estática. Ejemplo

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3	80	40	80

1. $T_s \geq máx(C_i)$
2. $U = \sum_1^N \left(\frac{C_i}{T_i}\right) \leq 1$
3. $T_M = mcm(T_i)$
4. $\exists i: \frac{P_i}{T_s} - \left\lfloor \frac{P_i}{T_s} \right\rfloor = 0$
5. $\forall i: 2T_s - mcd(T_s, P_i) \leq D_i$

1. $T_S \geq 40$ ms
2. $U = 0{,}95$
3. $T_M = 80$ ms
4. Se cumple para T2 y T3
5. Condición de fin
   - T1: 60 ms ≤ 20 ms
   - T2: 40 ms ≤ 40 ms
   - T3: 40 ms ≤ 80 ms
   - ***No se puede planificar***

Planificación cíclica estática. Ejemplo

Tarea	T_i (ms)	C_i (ms)	D_i (ms)
T1	20	4	20
T2	40	10	40
T3.1	**80**	**5**	80
T3.2	**80**	**15**	80
T3.3	**80**	**5**	80
T3.4	**80**	**15**	80

1. $T_s \geq máx(C_i)$
2. $U = \sum_1^N \left(\frac{C_i}{T_i}\right) \leq 1$
3. $T_M = mcm(T_i)$
4. $\exists i: \frac{P_i}{T_s} - \left\lfloor \frac{P_i}{T_s} \right\rfloor = 0$
5. $\forall i: 2T_s - mcd(T_s, P_i) \leq D_i$

1. $T_S \geq 20$ ms
2. $U = 0{,}95$
3. $T_M = 80$ ms
4. Se cumple para todas las tareas
5. Condición de fin
   - T1 : 20 ms < 20 ms
   - T2 : 40 ms < 40 ms
   - T3.1: 20 ms < 80 ms
   - T3.2: 20 ms < 80 ms
   - T3.3: 20 ms < 80 ms
   - T3.4: 20 ms < 80 ms
   - ***Es planificable.***

Tareas Esporádicas

El ejecutivo cíclico sólo permite ejecutar tareas periódicas

Las tareas esporádicas se ejecutan con un **servidor de consulta** (polling server). Se trata de una tarea periódica que consulta si se ha producido el suceso esporádico o no, donde el período depende de la separación mínima entre eventos y del plazo de respuesta

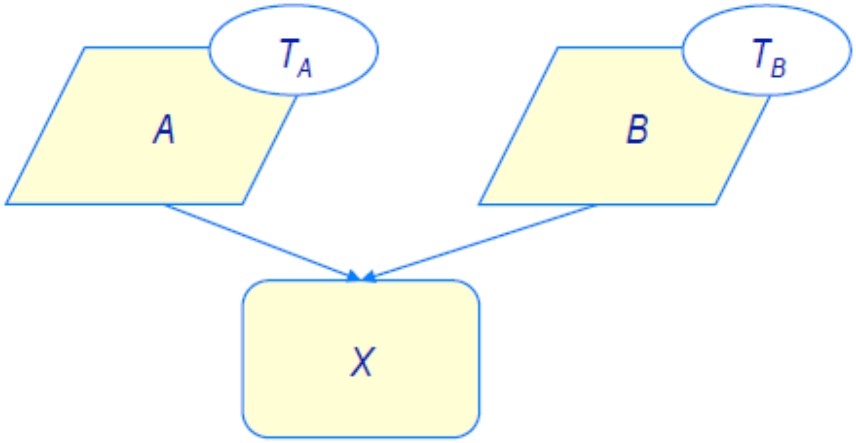
Recursos Compartidos

No hay que hacer nada, cuando se lee o escribe se ejecuta solamente una tarea

Una tarea (o segmento) se ejecuta sin interrupción hasta que termina

No es necesario proteger los recursos compartidos

La exclusión mutua es automática



Construcción del Plan de 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

Plan de Cíclico: 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 una vez construidos

Bibliografía.

- Sistemas Operativos. Diseño e Implementación. Andrew S. Tanenbaum.
- Sistemas de Tiempo Real y Lenguajes de Programación. Alan Burns, Andy Wellings. Tercera Edición. Addison Wesley.

TTS sistem implementation. Buscar en Google.