URJC Escuela Técnica Superior de Ingeniería de Telecomunicación

Sistemas Empotrados y de Tiempo Real

Planificación de Tareas

Grado en Ingeniería de Robótica Software

Teoría de la Señal y las Comunicaciones y Sistemas Telemáticos y Computación

Roberto Calvo Palomino
roberto.calvo@urjc.es

Introducción

- Usualmente los sistemas empotrados de tiempo real disponen de **mayor número de tareas** a ejecutar que numero de núcleos o unidades de procesamiento.
- Las tareas tienen que **competir** para obtener acceso a la unidad de procesamiento.
- El **planificador** es el encargado de decidir que tareas y recursos se ejecutan en cada momento.
- Estados de las tareas:
  - Creación
  - Listo
  - Ejecución
  - Terminación

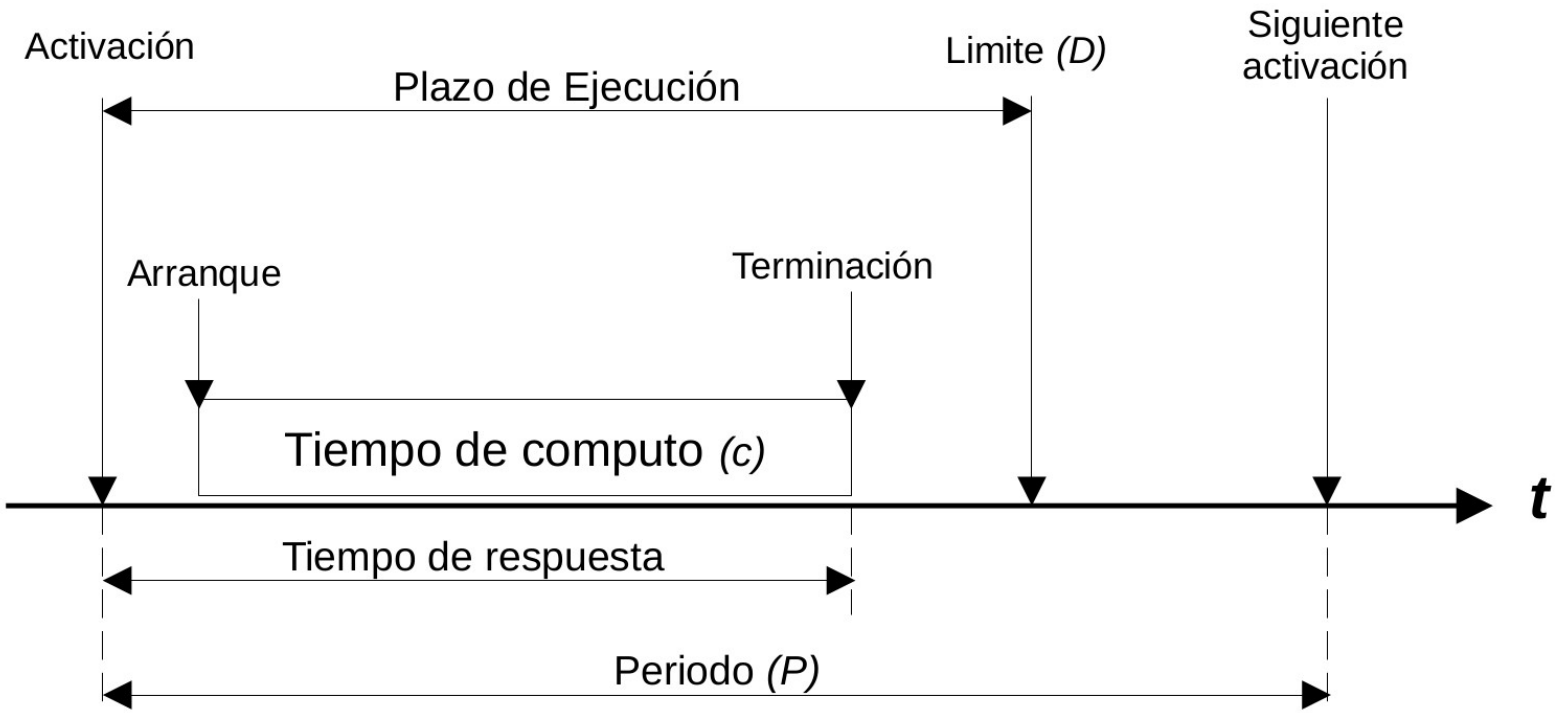
Tareas en Sistemas en Tiempo Real

- Clasificación de los STR según el flujo de ejecución
  - **Sistemas monotarea:**
    - Un único flujo de ejecución
    - Bucle infinito
    - Muestreo de las entradas (sondeo – polling)
    - Sencillos pero poco flexibles
    - Dificil añadir nueva funcionalidad.
  - **Sistemas multitarea:**
    - Compuestos por un conjunto de tareas
    - Procesos concurrentes
    - Controlar recursos
    - Comunicación entre tareas

Tareas en Sistemas en Tiempo Real

- En función de la forma de ejecución, las tareas se clasifican:
  - **Tareas periódicas**: Se activan repetidamente a intervalos de tiempo
    - Período de activación *(p)*
    - Período de ejecución *(d)*
    - Tiempo de cómputo *(c)*

Se debe cumplir que $0 \leq c \leq d < p$



Tareas en Sistemas en Tiempo Real

- **Tareas esporádicas.**
  - Se activan en instantes aleatorios y tienen requisitos temporales críticos.
  - Características:

    - 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).

Tareas en Sistemas en Tiempo Real

- **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.

Tareas en Sistemas en Tiempo Real

- 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.

  ¿ Ejemplos ?

Planificador de Tareas

- Es el sub-sistema encargado de **decidir** qué tareas se ejecutan en el procesador y en qué momento.
  - Cada procesador debe estar asignado como máximo a una sola tarea en cada instante.
  - Cada tarea debe estar asignada como máximo a un solo procesador.
  - La cantidad total de tiempo de procesador asignada a cada tarea debe ser igual a su tiempo máximo de ejecución.
  - Se deben satisfacer todas las restricciones de precedencia y uso de recursos comunes.
  - Dadas ciertas restricciones, el planificador puede no conseguir una correcta planificación de tareas (ya que no existe)

Tipos de Planificadores

- Planificadores **cíclicos**
  - Se define el orden y secuencia de las tareas a mano.
- Planificadores por **prioridades**
  - Cada tarea tiene una prioridad debido a su importancia.
  - La asignación de prioridades puede ser estática o dinámica.
  - Planificadores con prioridades estáticas:
    - Rate Monotonic (RM): Tareas son prioridad alta son planificadas más frecuentemente.
    - Deadline Monotonic (DM). Tareas con el plazo más corto son asignadas con prioridades altas.
  - Planificadores con prioridades dinámicas:
    - Earliest Deadline First (EDF) y el Least Laxity First (LLF)

Planificación Cíclica de Tareas

- Ejecutan de forma iterativa un conjunto de tareas **periódicas** con un solo procesador.
- A partir de los requisitos temporales del conjunto de tareas, se define la secuencia de tareas que deben ejecutarse durante un período fijo de tiempo (**ciclo principal**)
- El plan principal se divide en **planes** secundarios. Secuencia de procesos que se deben ejecutar durante un período de tiempo fijo
- Cada ciclo secundario se divide a su vez en **marcos** dentro de los cuales se ejecuta un solo proceso

Planificación Cíclica de Tareas

- Los métodos de planificación estática basados en una planificación **cíclica** son comúnmente utilizados en sistemas de tiempo real críticos.
- Ventajas:
  - Determinista y predecible
  - No hay sobrecarga por cambios de contexto ni exclusión mutua.
  - La implementación es sencilla y el planificador se ocupa de activar los procesos por turnos.
  - Las tareas se ejecutan según el plan definido por el diseñador/programador.
  - El sistema operativo se reemplaza por una ejecución cíclica
- Desventajas:
  - El diseño de los planes es bastante complejo
  - No son fáciles de mantener y actualizar.

Planificación Cíclica de Tareas

- Sistemas mono-procesador
- Conjunto de tareas estáticas
- Solo admite tareas periódicas
  - Tiempo máximo de ejecución igual al periodo.
- Sistema síncrono: Todas las tareas piden ejecución al mismo tiempo. Todas las tareas se activan en *t=0*
- Hiper-periodo es el mínimo común múltiplo de los periodos de todas las tareas.
- Tareas periódicas no se suspenden por ellas mismas
  - Únicamente al final de la ejecución.

$$tiempo\_computo\,(c) \leq plazo\_ejecucion\,(d) < periodo\_activacion\,(p)$$

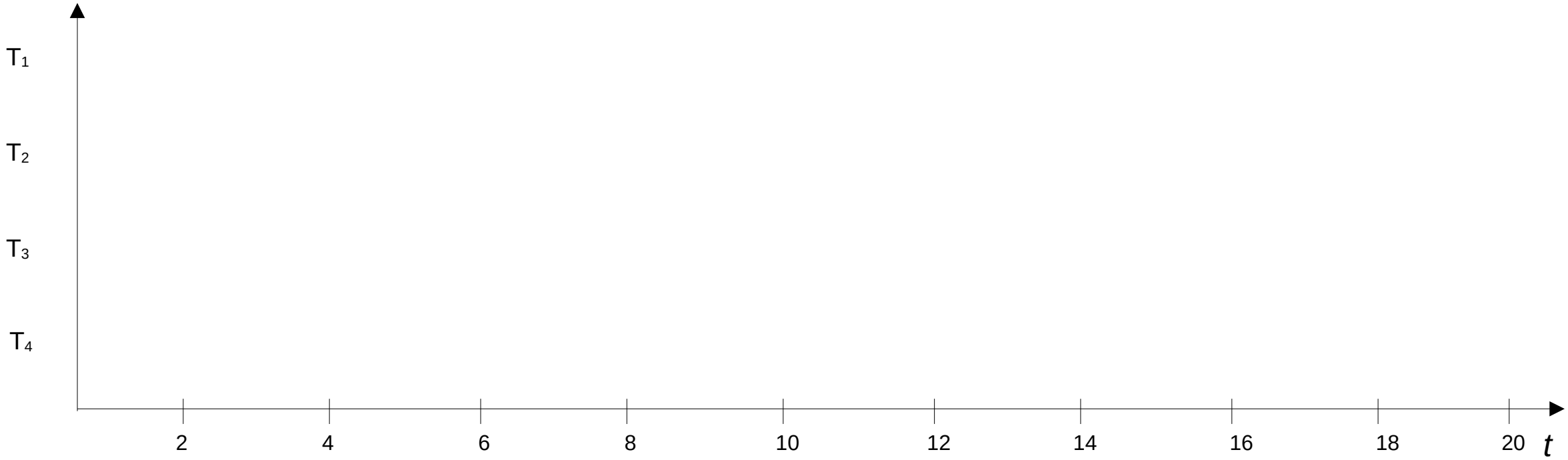
Planificación Cíclica de Tareas

- 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

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

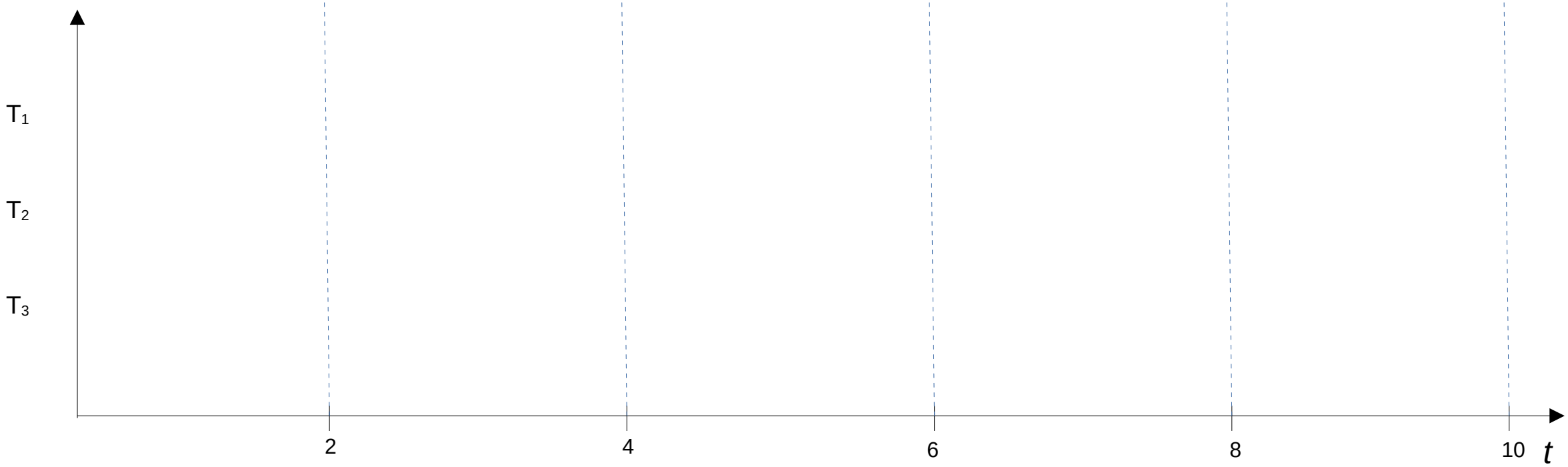
Planificación Cíclica de Tareas

Proceso	p	d	c
T_1	4	4	1
T_2	5	5	1.8
T_3	20	20	1
T_4	20	20	2



Planificación Cíclica de Tareas

Proceso	p	d	c
T_1 (lectura odometria)	2	2	1
T_2 (calculo bloqueo ruedas)	4	4	2
T_3 (accionar abs)	10	10	1

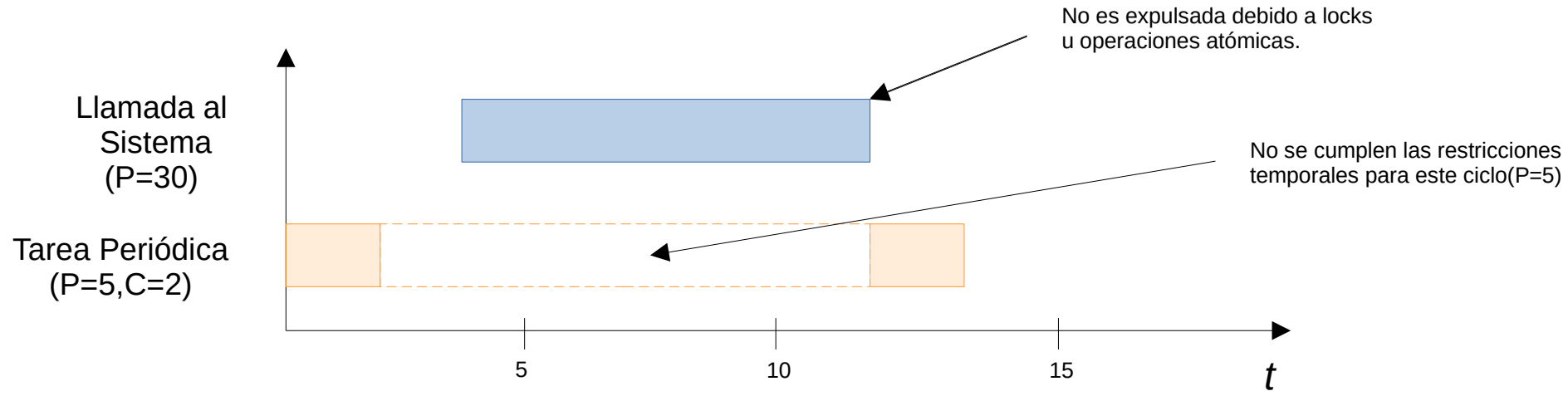


Tareas interrumpibles (PREEMPT)

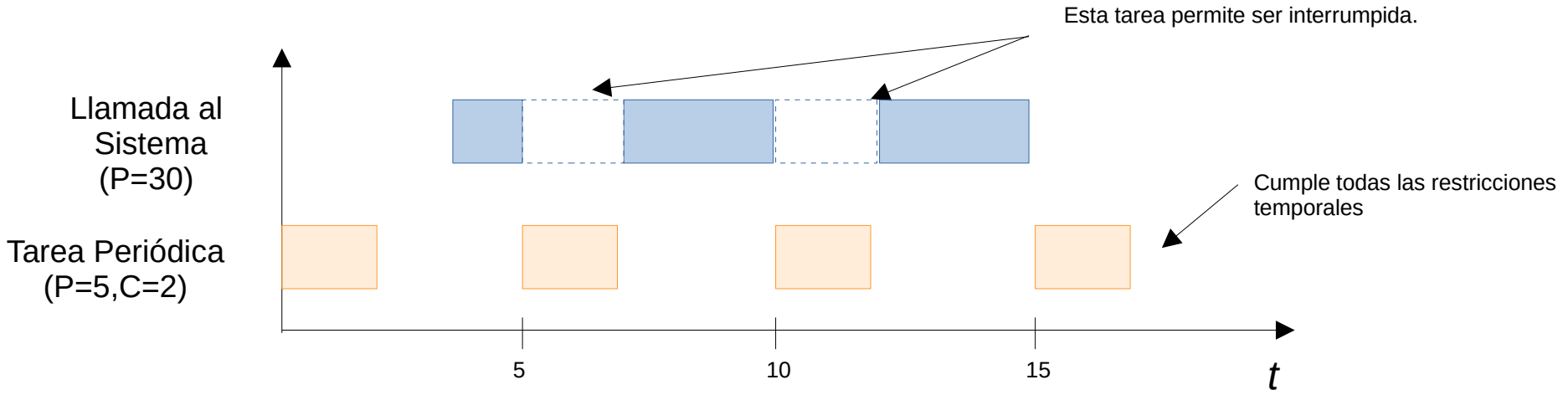
- Un sistema en tiempo real puede permitir que sus tareas puedan ser **interrumpidas** en cualquier momento para que una tarea de mayor prioridad se ejecute.
- La interrupción de una tarea se denomina **PREEMPTION**
- El sistema podría tener tareas que no son interrumpibles.
- Al intercambio de estados en la CPU y memoria, cuando una tarea se interrumpe para dar paso de ejecución a una segunda tarea se denomina ***cambio de contexto***
- Para sistemas Linux tenemos
  - Kernel por defecto (No PREEMPT)
  - PREEMPT y PREEMPT-RT

Tareas interrumpibles (PREEMPT)

- Sistema NO PREEMPT



- Sistema PREEMPT-RT



Planificadores con prioridades estáticas

- En los planificadores por **prioridades** se asigna una prioridad a cada tarea en base a su importancia.
- En tiempo de **ejecución** se ejecutará siempre la tarea de más prioridad que este activa en cada instante.
- En este caso es el planificador quién decide en cada instante que tarea debe ejecutarse.
- El Planificador ***Rate Monotonic (RM)*** es uno de los más usados en sistemas de planificación con prioridades estáticas para tareas periódicas simples.

Rate Monotonic (RM)

- Las **prioridades** (Pr) de las tareas deciden el orden de su ejecución
- A mayor ***Pr*** antes entrará en la CPU a ejecutar, siempre dentro de su periodo de activación.
  (Ej. *P99* entra antes que *P1* en el procesador)
- Todas las tareas son **periódicas** e independientes unas de otras.
- No es un planificador óptimo, **excepto** cuando las tareas periódicas son simples.

$$\sum_{i=1}^{N} \frac{C_i}{P_i} < N\left(2^{1/N} - 1\right)$$

Utilización

- En la política de planificación de Rate Monotonic, todos los deadlines de N tareas periódicas están garantizados si:

$$U \leqslant N\left(2^{1/N}-1\right)$$

  - Condición suficiente pero no necesaria (Liu & Layland, 1973)

- Para calcular la utilización mínima garantizada para *N* tareas

$$U_0 = N\left(2^{1/N}-1\right)$$

Utilización mínima garantizada

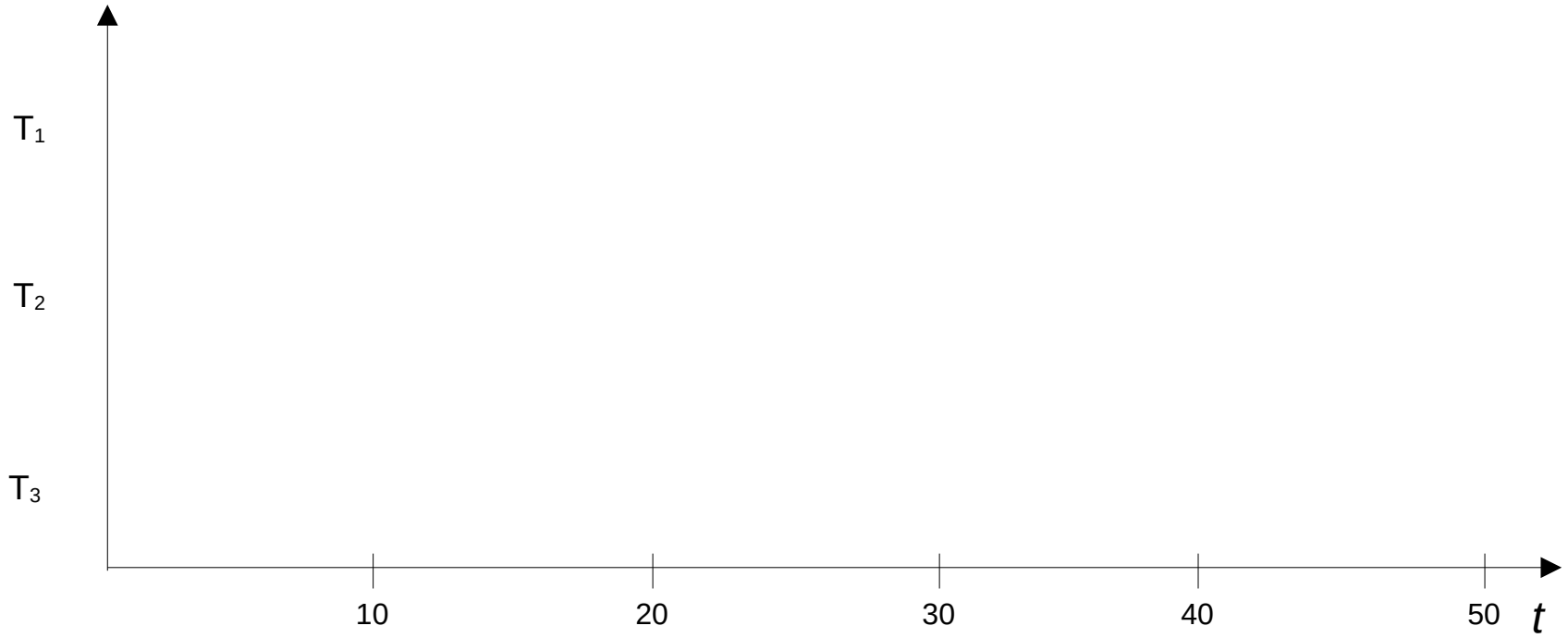
N	$U_0(N)$
1	1,000
2	0,828
3	0,779
4	0,756
5	0,743

- Si $U \leqslant U_0(N)$
  - Deadlines están garantizados.
- Si $U > 1$
  - Deadlines no están garantizados.
- Si $U > U_0(N) \leq 1$
  - No es posible decidir nada mediante este método. Tendríamos que realizar el **cronograma temporal.**

Rate Monotonic (RM)

Tarea	P	C	D	Pr	U
T_1	20	5	20	2	
T_2	30	12	30	1	
T_3	50	10	50	3	

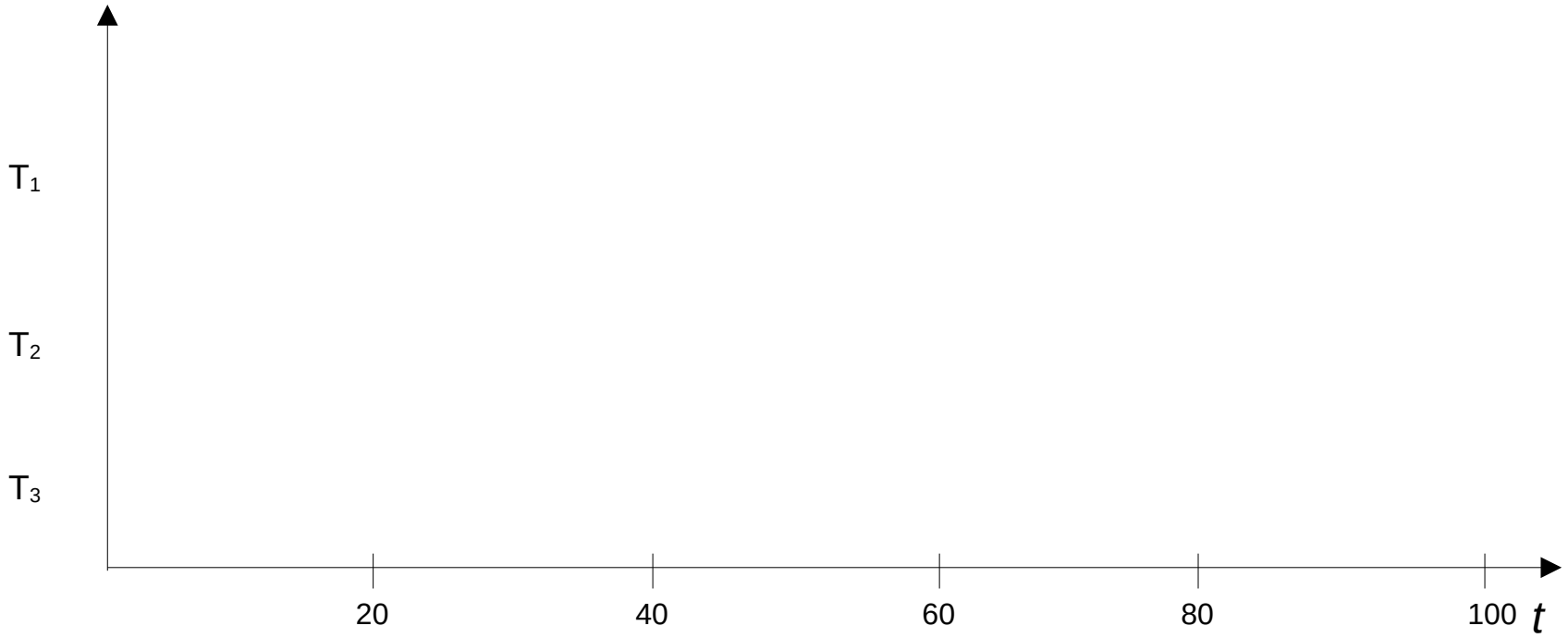
- Calculo de U=
- Calculo de $U_0(N)$ =
- Realizar el cronograma temporal.



Rate Monotonic (RM)

Tarea	P	C	D	Pr	U
T_1	20	5	20	3	
T_2	80	40	80	1	
T_3	40	10	40	2	

- Calculo de U =
- Calculo de $U_0(N)$ =
- Realizar el cronograma temporal.



Sistema Tiempo Real

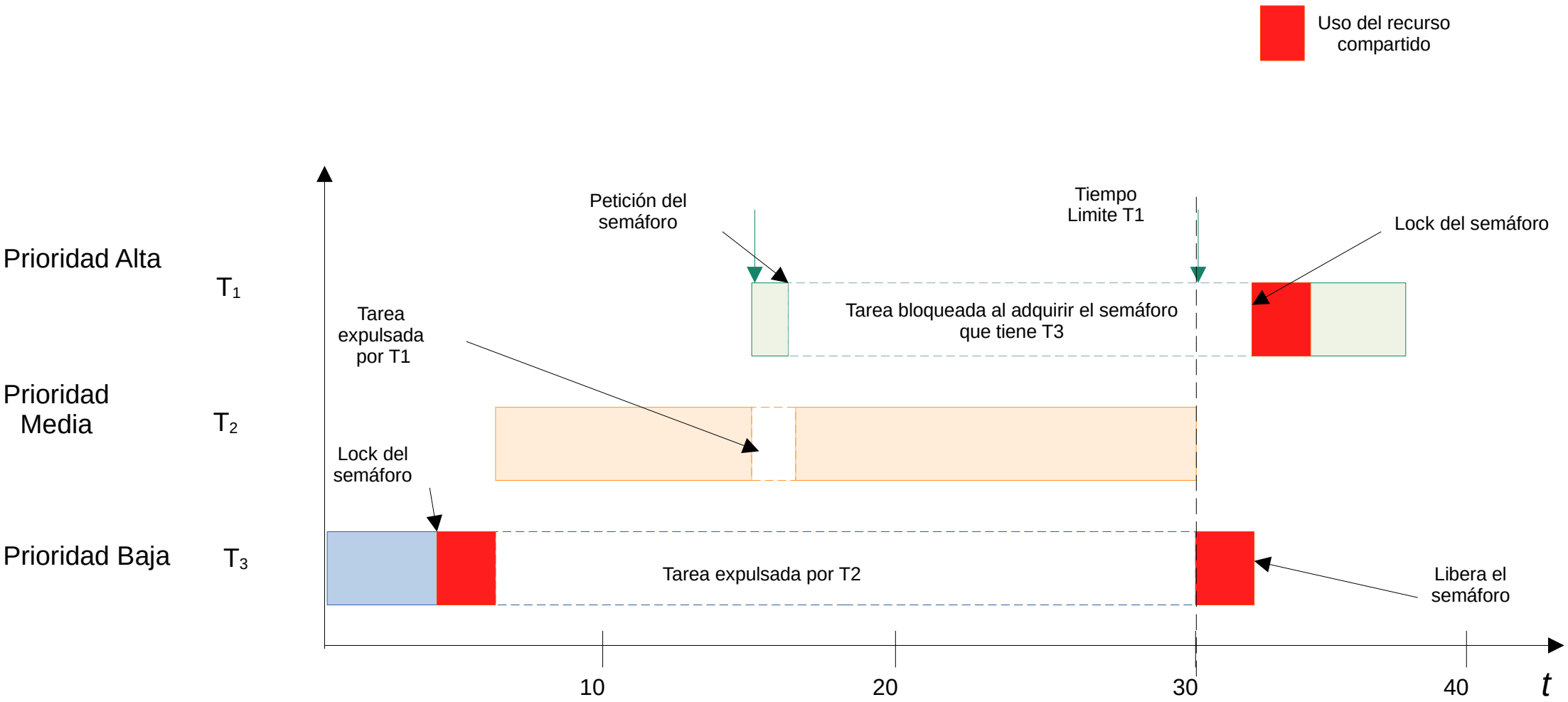
- Tareas
  - [T1] Actuador servo p=20, c=5, pr=2
    - Necesita el valor de la temperatura
  - [T2] Lector sensor temperatura: p=10, c=2, pr=3
  - [T3] Encendido de un led, p=30, c=1, pr=1
    - Se enciende después del actuador

- ¿Podrías además añadir otra tarea al sistema que lea sensor de humedad (p=20, c=3) ?
- U, $U_{0,}$ cronograma temporal.

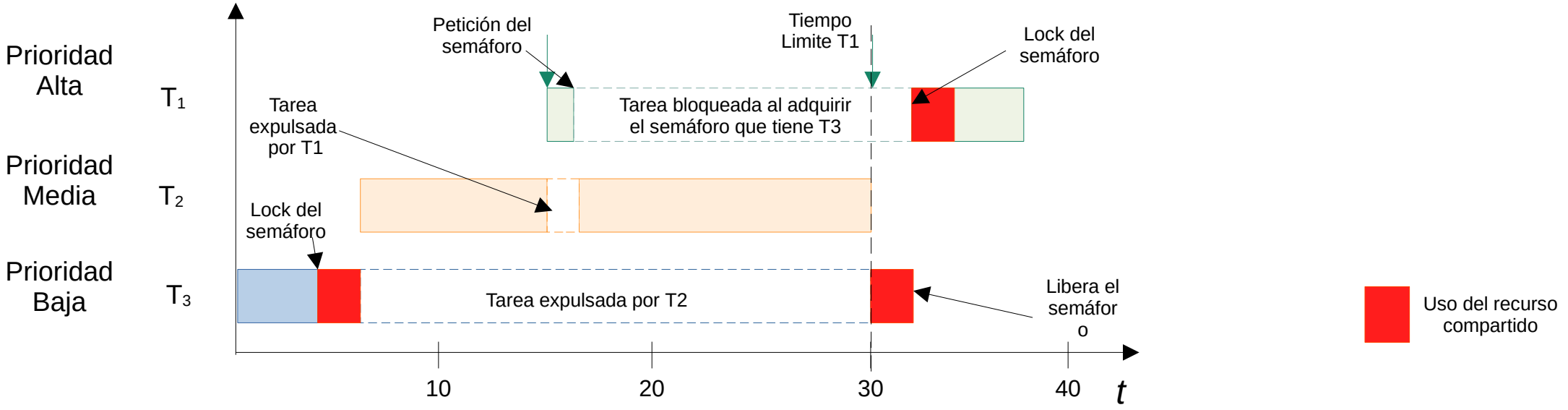
Inversión de Prioridades

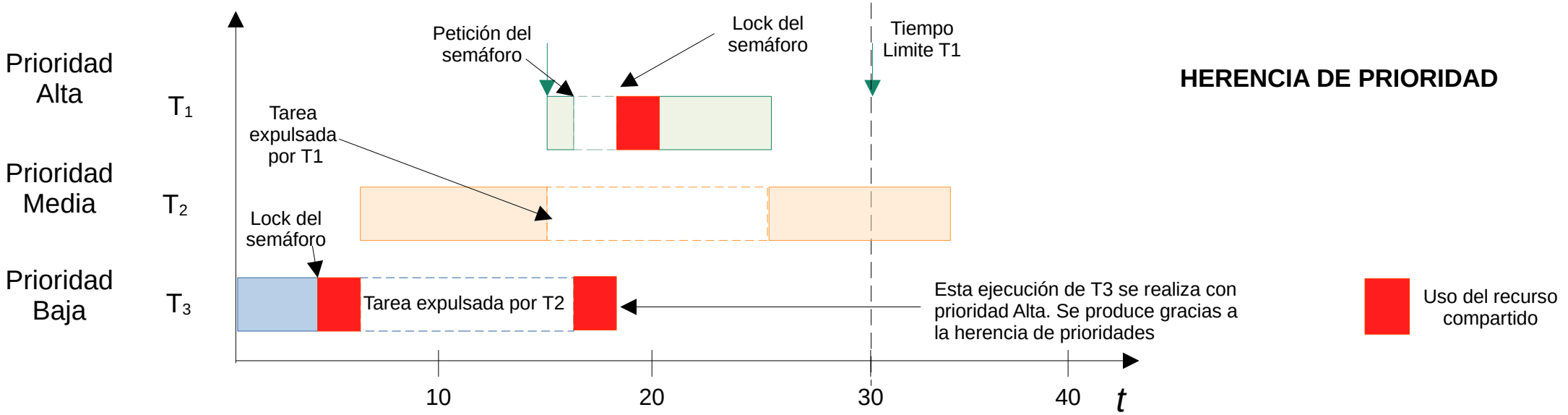
- La **inversión de prioridades** es un problema que se puede presentar en todos los sistemas de multi-programación cuando se usan mecanismos para controlar la exclusión mutua al acceso de recursos compartidos.
- Escenario:
  - T1 con prioridad alta, T2 con prioridad media y T3 con prioridad baja
  - T3 coge un recurso compartido (M), pero es expulsada de la CPU para ejecutar T2
  - T2 es expulsada porque T1 entra en ejecución e intenta coger (M)
  - T1 se queda bloqueada y el planificador pone a ejecutar T2
- Solución:
  - Herencia de Prioridad. El planificador incrementa la prioridad de la tarea que tiene el **cerrojo** (T3) a la máxima prioridad de otra tarea (T1) que está bloqueada esperando la liberación de dicho **cerrojo.**

Inversión de Prioridades



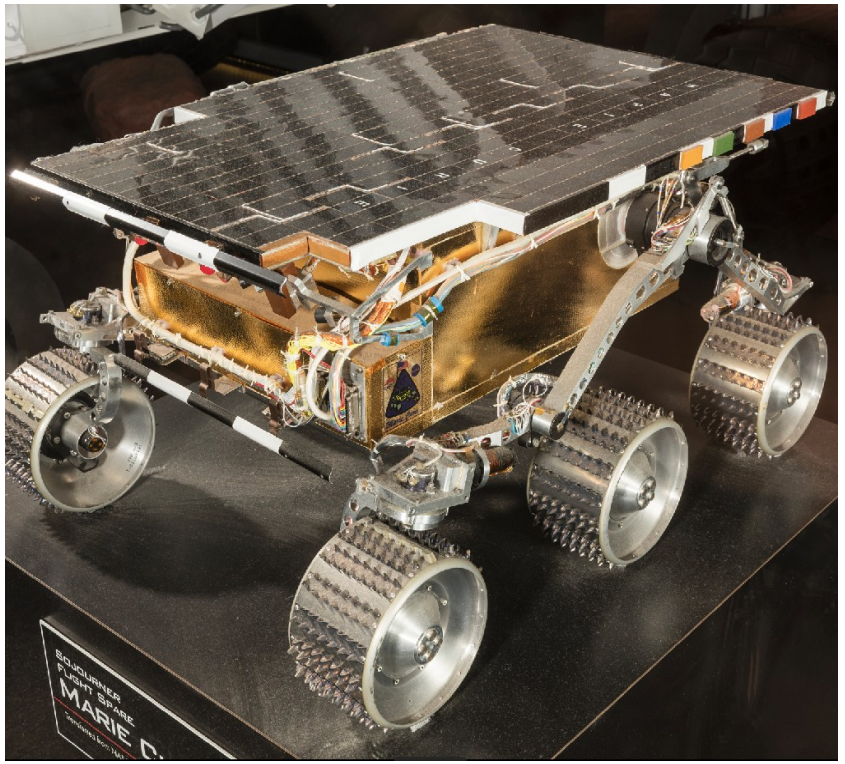
Inversión de Prioridades





Inversión de Prioridad (en Marte!)

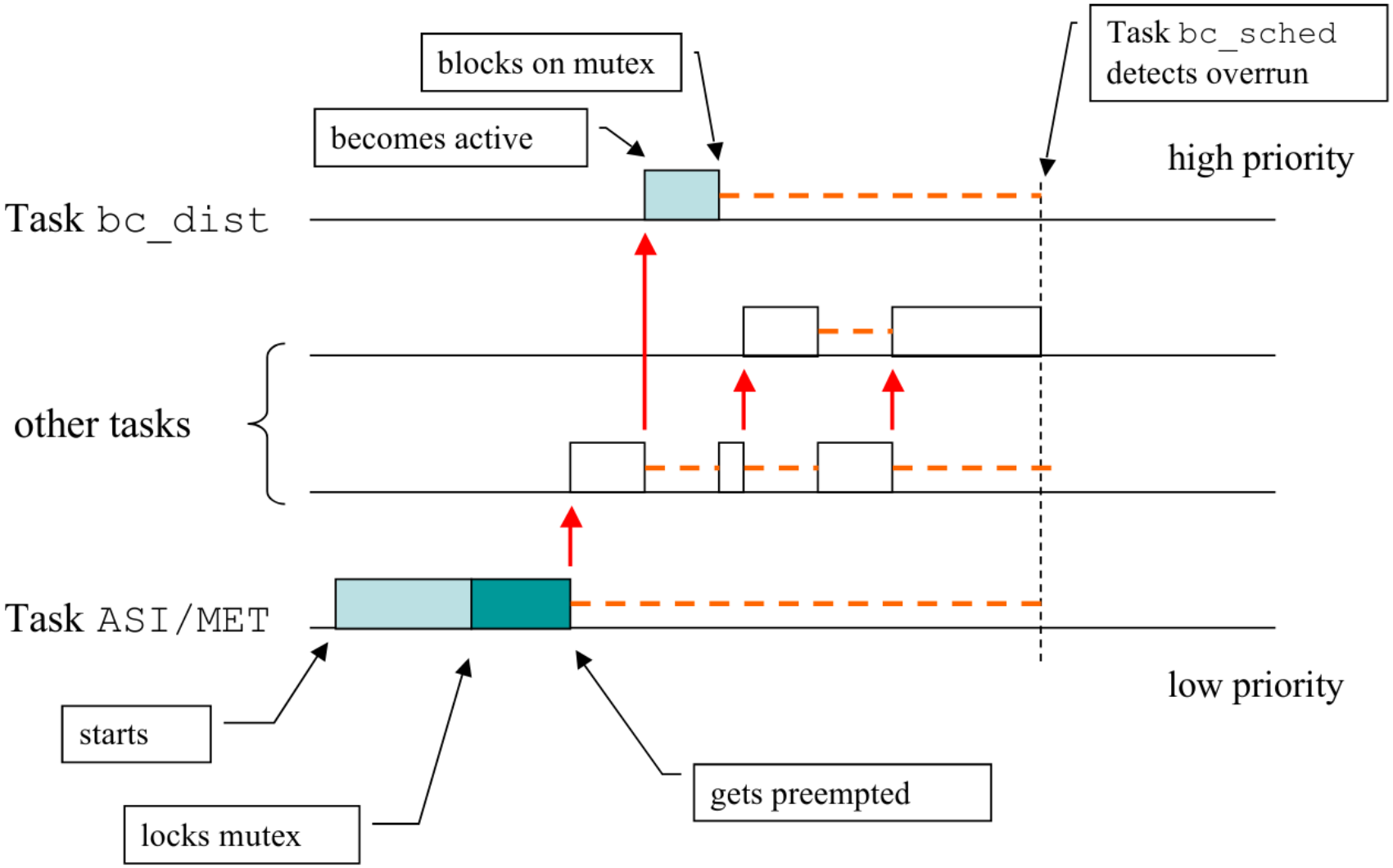
- Mars PathFinder: nava cuya misión fue estudiar Marte desde la superficie (1996)
- Sistema de tiempo real duro VxWorks RTOS
- Tareas con prioridades y “expulsables” (PREEMPT)
- Detectaron continuos reset del sistema después de varios días en Marte.
- El sistema contenía un recurso común “information bus” usado para compartir información entre diferentes componentes.



Inversión de Prioridad (en Marte!)

- 3 de las numerosas tareas a ejecutar por el PathFinder
  - T1: Information Bus Thread (Periodicidad y prioridad Alta)
  - T2: Communication Thread (Periodicidad y prioridad Media)
  - T3: Weather Thead (Periodicidad y prioridad Baja)
- Watchdog que comprueba que la anterior tarea ejecutó por completo en el ciclo previo.
  - Si esta comprobación falla, se considera violación del sistema de tiempo real duro y el sistema se reinicia (tolerante a fallos)
  - En cada reinicio, se perdían datos recolectados de los últimos ciclos.
- Comunicación entre tareas:
  - Memoria compartida era usada para pasar información de la T3 a la T2 via de bus (T1).

Inversión de Prioridad (en Marte!)



http://www.cse.nd.edu/~cpoellab/teaching/cse40463_fall05/lecture11.pdf

Inversión de Prioridad (en Marte!)

- Solución: habilitar la herencia de prioridad.
  - Habilitar un flag en la llamada de adquisición del semáforo.
  - Afecta al rendimiento del sistema
  - Se generó el parche y se envió a Marte por radio.

Bibliografía

- Real-Time Embedded Systems
  (Ivan Cibrario Bertolotti, Gabriele Manduchi)
- Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications (Giorgio C. Buttazzo)
- A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems
  (Mark Klein , Thomas Ralya , Bill Pollak , Ray Obenza , Michael González Harbour)
- Fixed priority pre-emptive scheduling: An historical perspective
  (Neil C. Audsley, Alan Burns, Robert I. Davis, Ken W. Tindell & Andy J. Wellings)
- Priority Inversion on Mars
  https://es.slideshare.net/jserv/priority-inversion-30367388
- L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization.
  In IEEE Transactions on Computers, vol. 39, pp. 1175-1185, Sep. 1990.

URJC Escuela Técnica Superior de Ingeniería de Telecomunicación