Tema 8

Planificación de Tareas Aperiódicas

Objetivos

1. Conocer el problema del *scheduling* en sistemas con tareas periódicas que incluyen **tareas aperiódicas** con distintos niveles de criticidad

2. Conocer y entender el funcionamiento de distintos **algoritmos de planificación** con tareas periódicas y aperiódicas

Índice

Restricciones temporales

- Un STR puede requerir tanto tareas periódicas como aperiódicas
- Las **tareas periódicas** generalmente tienen restricciones de tiempo **hard**
- Las **tareas aperiódicas** son dirigidas por eventos (*event-driven*) y pueden tener restricciones de tiempo:
  - Hard
  - Firm
  - Soft
  - no tiempo real

Garantía de planificabilidad

- Cumplir los plazos de todas las **tareas aperiódicas *hard*** en las condiciones del peor caso
- Maximizar el ratio de **tareas aperiódicas *firm***
- Optimizar tiempos de respuesta medios para **tareas aperiódicas *soft*** o sin restricciones de tiempo real

Índice

Tareas aperiódicas *hard*

- Se asumen como tareas **esporádicas**:
  - Suponemos que en el peor de los casos la tarea es pseudoperiódica (con periodo **T**)
    - El parámetro **T** representa la separación mínima entre dos activaciones consecutivas
  - El plazo de respuesta suele ser **D < T**
- Los análisis de planificabilidad para FPS siguen siendo válidos

Tareas aperiódicas *firm*

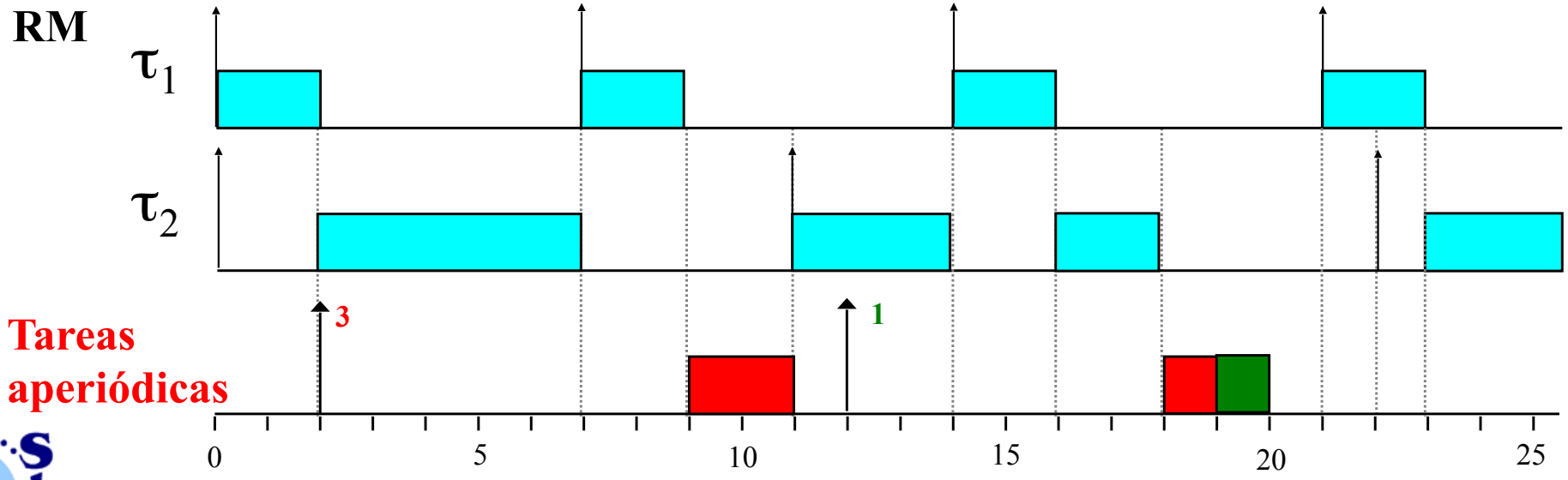
- Cuando no se puede garantizar a priori la tasa de llegada máxima
- Se requiere un **test de aceptación**
  - Verificar si la tarea puede ejecutarse dentro de su plazo, sin incumplir el *deadline* del resto de tareas
  - Si no pasa el test, la tarea es rechazada

Tareas aperiódicas *soft*

- Optimizar el tiempo de respuesta medio
- Planificadores:
  - *Background scheduling*
  - **Servidores** de prioridad fija
    - Servidor por consulta (*Polling Server*)
    - Servidor diferido (*Deferrable Server*)
    - Servidor esporádico (*Sporadic Server*)
  - Algoritmo de extracción de holgura (*Slack Stealing*)

Background scheduling

- Las tareas aperiódicas se ejecutan como procesos con la prioridad más baja
- Es un método muy sencillo, pero los tiempos de respuesta para las peticiones aperiódicas pueden ser muy altos



Índice

Concepto de servidor aperiódico

- Para mejorar los tiempos de respuesta medios de tareas aperiódicas, se usa un **servidor**
- Servidor = tarea periódica que atiende peticiones aperiódicas
  - (T_s, C_s)
  - C_s = capacidad del servidor
- El servidor es planificado con el mismo algoritmo usado para las tareas periódicas
- Una vez activo, sirve las peticiones aperiódicas dentro del límite de su capacidad

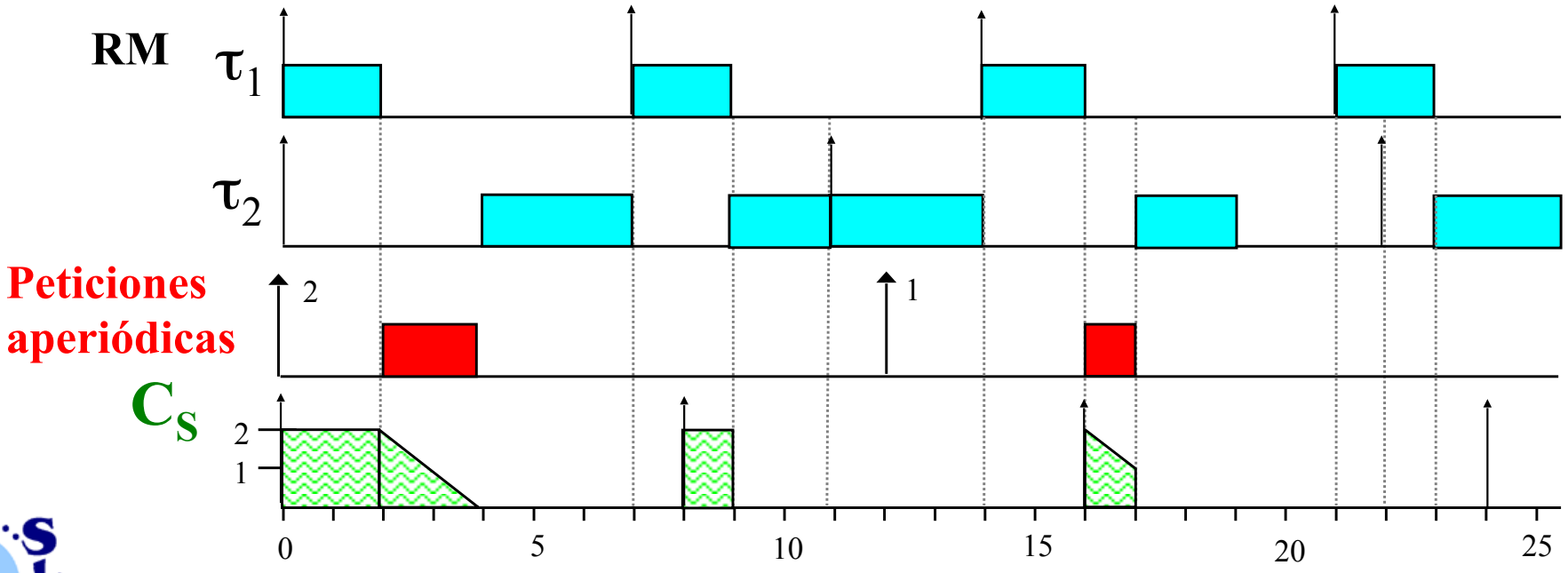
Servidor por consulta (*Polling Server*)

- En cada activación del servidor :
  - Se restablece la capacidad del servidor
  - Cuando es seleccionado por el planificador comprueba si hay peticiones aperiódicas pendientes
    - Si las hay, las puede ejecutar si le queda capacidad
    - Si no las hay, se **suspende** (pierde su capacidad) hasta su próxima activación

Ejemplo de servidor por consulta

	C_i	T_i
τ_1	2	7
τ_2	5	11

	C_s	T_s
servidor	2	8



Análisis de planificabilidad con servidores por consulta

- La planificabilidad de las tareas periódicas puede garantizarse evaluando la interferencia que provoca el servidor
  - En el peor de los casos dicha interferencia es igual que la que produciría una tarea periódica con periodo Ts y tiempo de computación Cs

- Se utiliza el **test de factores de utilización**:

$$U_p + U_S \leq U_{lub}(n+1)$$

Condición suficiente

para **1** servidor:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} + \frac{C_S}{T_S} \leq (n+1) \cdot \left[2^{1/(n+1)} - 1 \right]$$

para **m** servidores:

$$U_P + \sum_{J=1}^{m} \frac{C_{Sj}}{T_{Sj}} \leq U_{lub}(n+m)$$

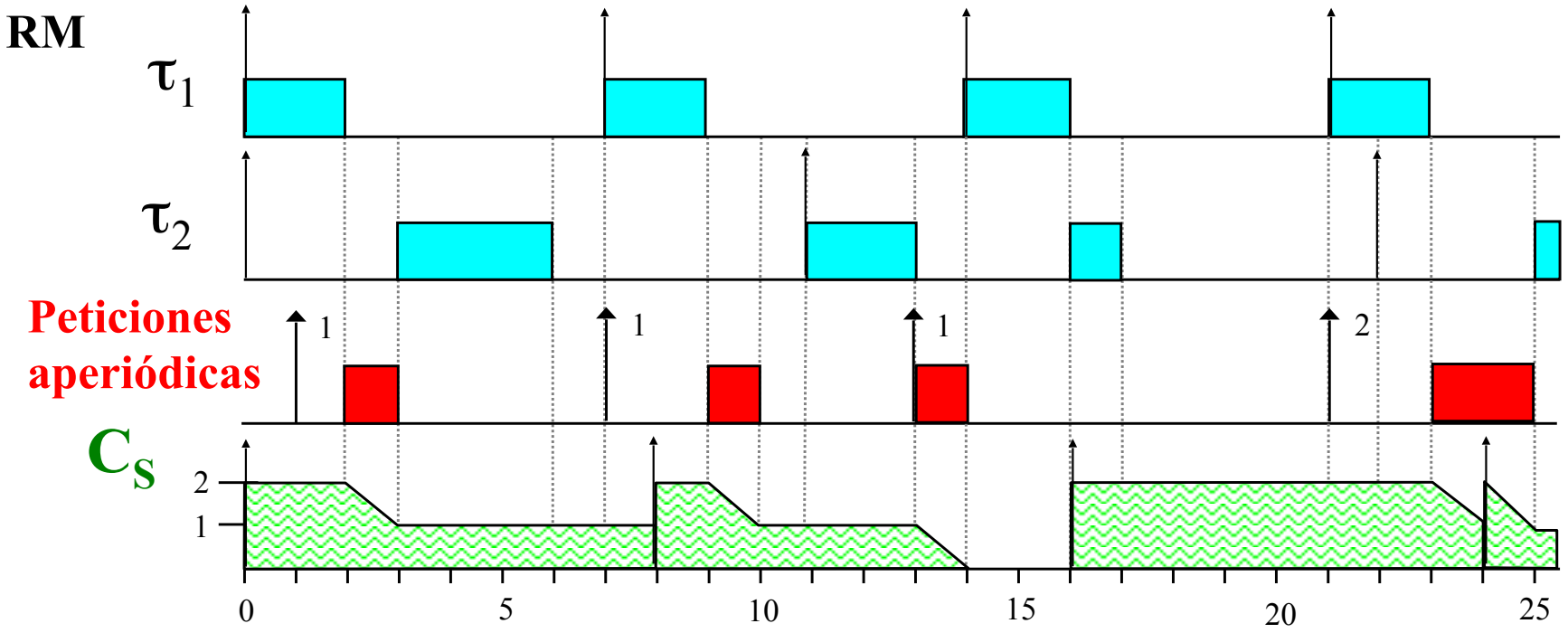
Servidor diferido (*Deferrable Server*)

- En cada activación del servidor :
  - Se restablece la capacidad del servidor
  - Cuando es seleccionado por el planificador comprueba si hay peticiones aperiódicas pendientes
    - Si las hay, las puede ejecutar si le queda capacidad
    - Si no las hay, **conserva** su capacidad hasta su próxima activación (a menos que la necesite antes)

Ejemplo de servidor diferido

	C_i	T_i
τ_1	2	7
τ_2	3	11

	C_s	T_s
servidor	2	8



Análisis de planificabilidad con servidores diferidos

- El servidor diferido necesita de un análisis de planificabilidad propio, pues no cumple una asunción básica del algoritmo RM:
  - Una tarea debe ejecutarse cuando sea la de mayor prioridad lista para ejecución

- La planificabilidad de las tareas periódicas (planificadas mediante RM) se garantiza cuando:

$$U_p \leq \ln\left(\frac{U_s + 2}{2U_s + 1}\right)$$

Condición suficiente

Servidor esporádico (*Sporadic Server*)

- La capacidad del servidor :
  - se conserva cuando no hay peticiones aperiódicas pendientes
  - se repone siguiendo la siguiente regla:
    - **Instante de reposición** (***Replenishment time***) RT:

$$RT = t_a + T_s$$

      - t_a = instante en el que el servidor pasa a estar activo
        - **Servidor activo** = cuando $Prioridad_{ejecución} >= P_s$ **y** le queda capacidad
      - T_s = Periodo del servidor
    - **Cantidad a reponer** (***Replenishment amount***) RA:

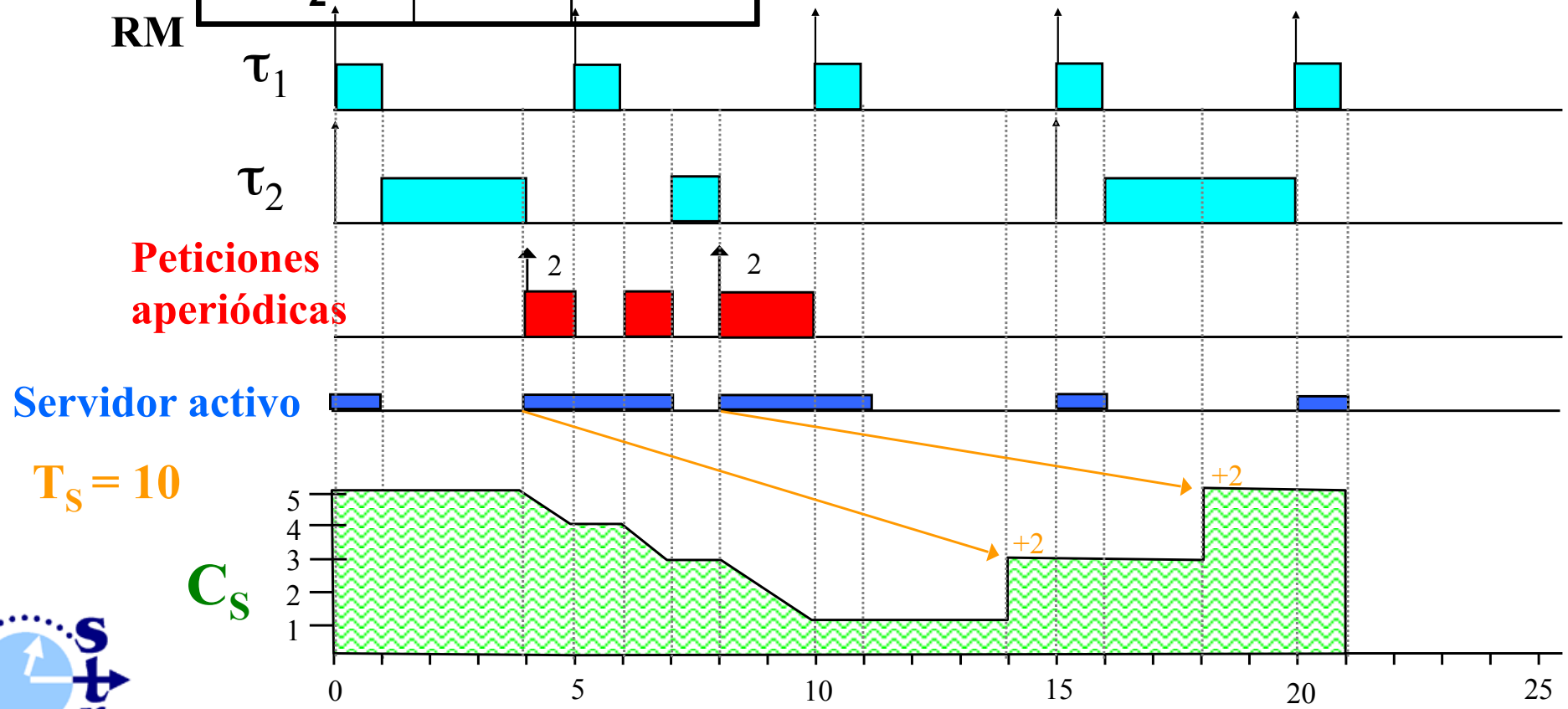
$$RA = \text{capacidad consumida en } [\, t_a \,,\, t_i \,)$$

      - t_i = instante en el que el servidor pasa a estar inactivo (*idle*)
        - **Servidor inactivo** = cuando $Prioridad_{ejecución} < P_s$ **o** no le queda capacidad

Ejemplo de servidor esporádico

	C_i	T_i
τ_1	1	5
τ_2	4	15

	C_s	T_s
servidor	5	10



Análisis de planificabilidad con servidores esporádicos

- La planificabilidad de las tareas periódicas (planificadas mediante RM) se garantiza cuando:

$$U_p \leq \ln\left(\frac{2}{U_s + 1}\right)$$

Condición suficiente

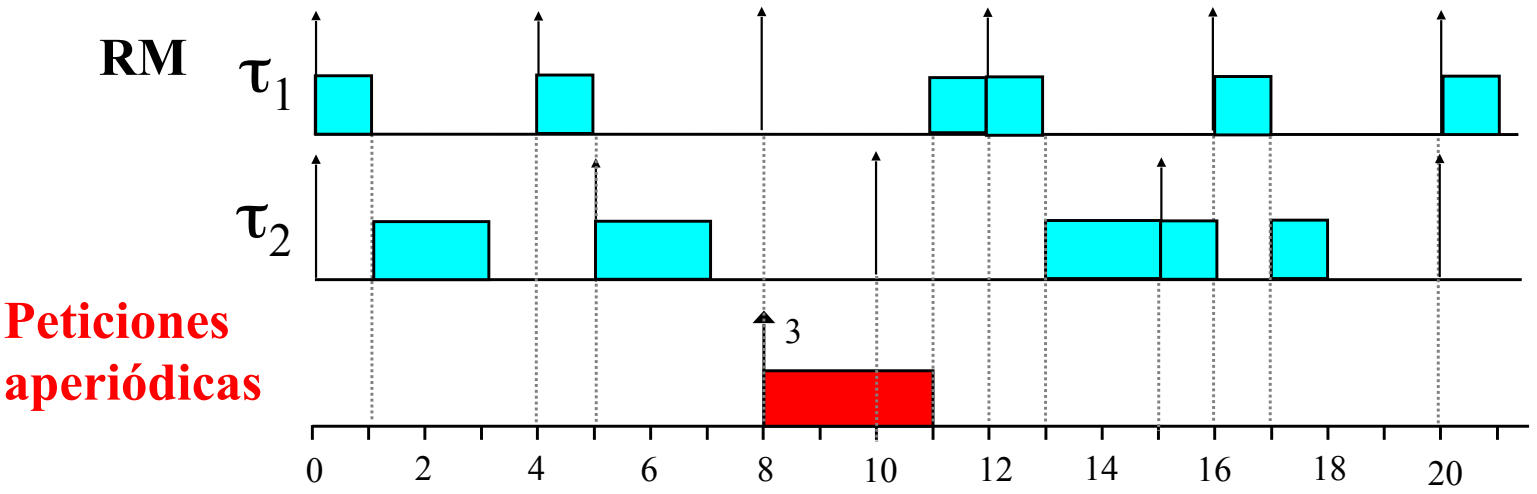
Índice

Extracción de holgura (*Slack stealing*)

- Las tareas periódicas pueden tener un tiempo de holgura (slack time)
- Se puede retrasar la ejecución de una tarea periódica hasta el instante que siga permitiendo cumplir con su *deadline*
- El tiempo de holgura es empleado para ejecutar tareas aperiódicas
- No se trata de un servidor periódico:
  - es una tarea que se activa cuando llega una petición aperiódica
  - "roba" tiempo de las tareas periódicas (sin causar que pierdan su plazo) para dedicárselo a las peticiones aperiódicas

Ejemplo de extracción de holgura

	C_i	T_i
τ_1	1	4
τ_2	2	5

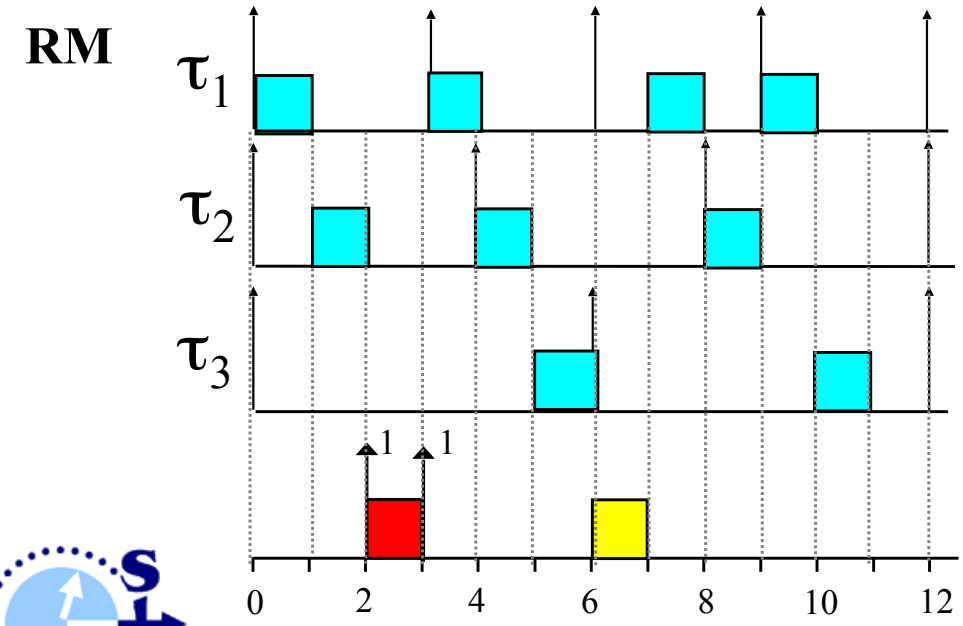


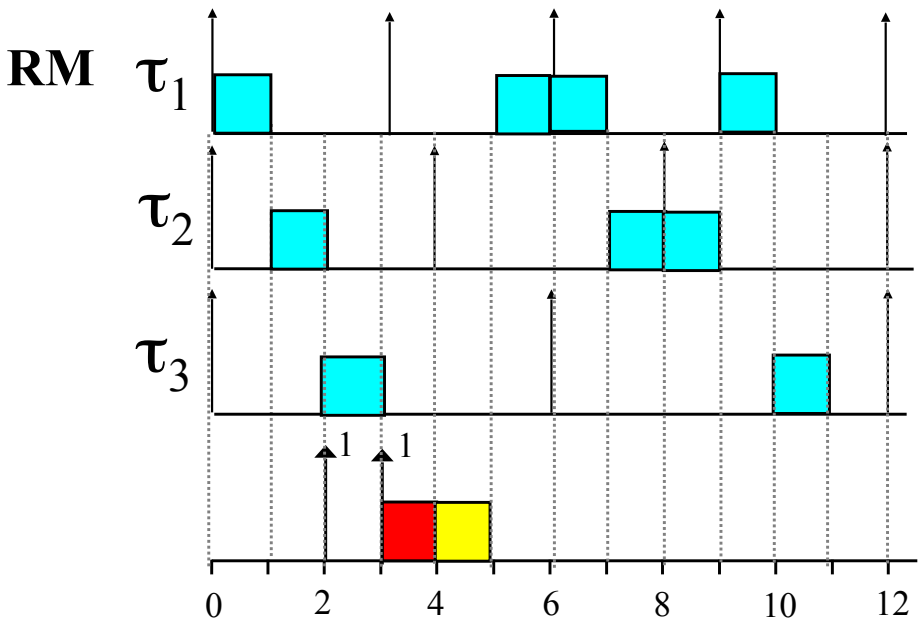
Índice

1. Restricciones temporales de las tareas
2. *Scheduling* de tareas aperiódicas
3. Servidores aperiódicos
4. Algoritmo de extracción de holgura
5. **Planificador óptimo**

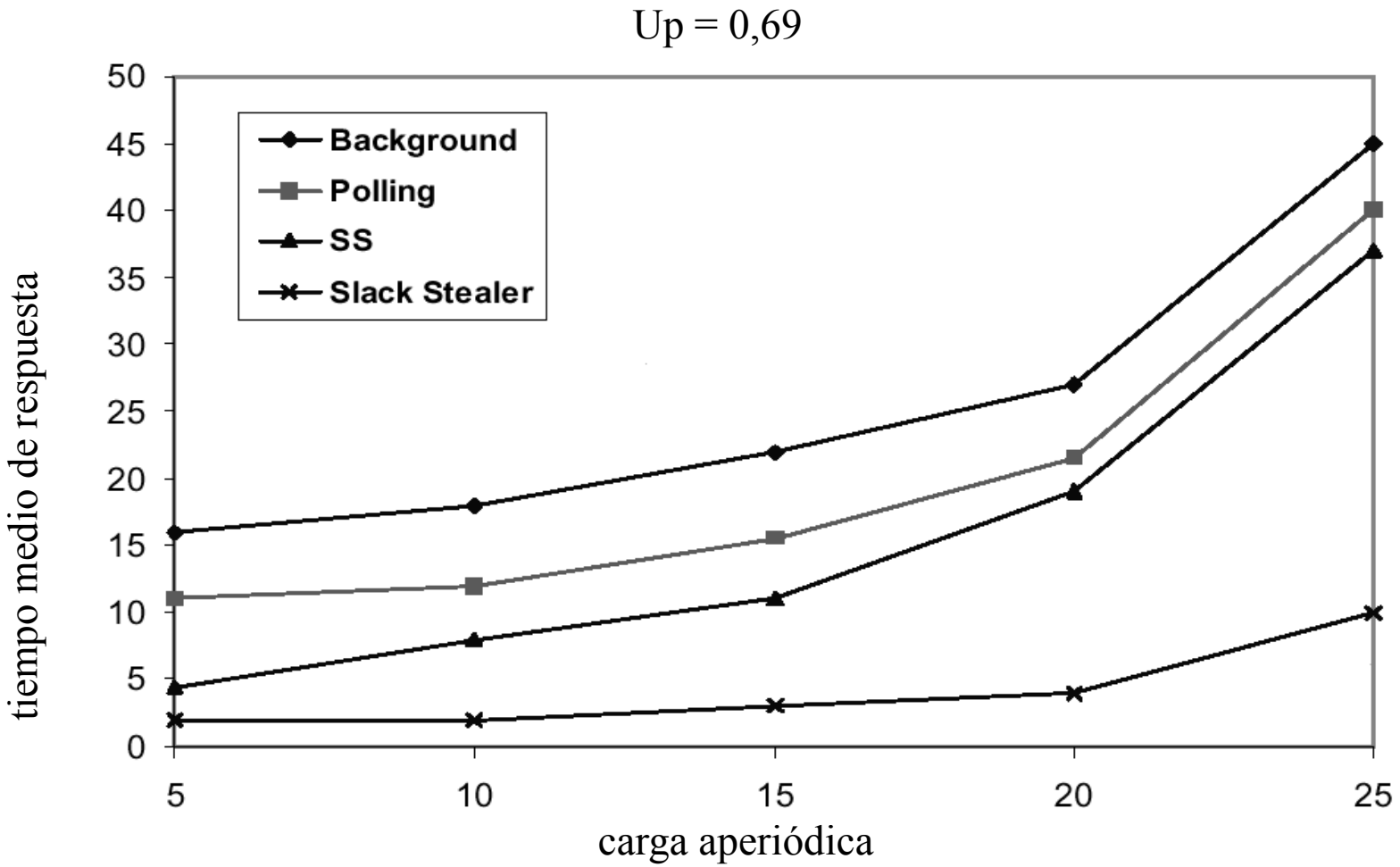
¿Planificador óptimo?

- El algoritmo de extracción de holgura se consideró durante mucho tiempo como el óptimo, ya que aprovecha la holgura para servir las peticiones aperiódicas lo antes posible
  - Sin embargo, hay casos en los que conviene esperar y no usar el primer tiempo de holgura disponible
- **No existe** el planificador aperiódico óptimo





Evaluación de rendimientos



Evaluación comparativa

Planificador	Rendimiento	Complejidad Computacional	Requerimientos de memoria	Complejidad de implementación
Background	Muy pobre	Excelente	Excelente	Excelente
Por Consulta	Pobre	Excelente	Excelente	Excelente
Diferido	Bueno	Excelente	Excelente	Excelente
Esporádico	Bueno	Buena	Bueno	Buena
Extracción holgura	Excelente	Pobre	Pobre	Pobre

Conclusiones

- Muchas aplicaciones de control de tiempo real requieren tanto tareas periódicas (normalmente críticas) como aperiódicas
- Se debe garantizar la planificabilidad en el peor caso de las tareas críticas y proporcionar buenos tiempos de respuesta para tareas no críticas
- Las dos estrategias básicas para ejecutar tareas aperiódicas son en background e interferir en la ejecución de las tareas periódicas
- Sin un conocimiento a priori de las llegadas de peticiones aperiódicas, un algoritmo on-line no sabrá cuándo planificar las peticiones para obtener un tiempo medio de respuesta mínimo

Bibliografía Recomendada

Sistemas de tiempo real y lenguajes de programación (**3ª edición**)
Alan Burns and Andy Wellings
Addison Wesley (2002)

? Apartado 13.8

Hard real-time computing systems (**Second edition**)
Giorgio C. Buttazzo
Kluwer Academic Publishers (2004)

? Capítulo 5 (Apartados: del 1 al 9, excepto el 5)