Algoritmos que preservan el tiempo del servidor de peticiones aperiódicas. Análisis de planificabilidad.

Daniel Monjas Miguélez

Universidad de Granada

Abstract: El siguiente trabajo realiza una recopilación de información los algoritmos de servidor, Servidor Diferido (*"Deferrable Server"*), Servidor de Intercambio de Prioridad Dinámico (*"Dynamic Priority Exchange Server"*) y Servidor Esporádico Dinámico (*"Dynamic Sporadic Server"*), cuyo objetivo es trabajar con un conjunto híbrido de tareas (periódicas y aperiódicas). En las distintas secciones se realizará una descripción detallada del algoritmo, que incluye a sus desarrolladores, su funcionamiento y un ejemplo básico para ilustrar el funcionamiento que se habrá explicado previamente, así como la explicación escrita del ejemplo.

Además, para cada uno de los tres algoritmos ya mencionados se realizará un análisis de planificabilidad, añadiendo también para el algoritmo de Servidor Diferido un apartado en el que se mostrarán los cálculos para el supremo de utilización del procesador con este algoritmo y para el algoritmo de Intercambio de Prioridad Dinámico se diseña y prueba un mecanismo para la reclamación de recursos libres.

Finalmente, en la sección de conclusiones se realiza una reflexión de todo los explicado en las secciones anteriores.

Keywords: Tiempo Real, análisis de planificabilidad, capacidad aperiódica, fecha límite.

1 Introducción

Muchas aplicaciones de control complejas incluyen tareas que deben ser completadas cumpliendo duras restricciones de tiempo, llamadas fechas límite o fechas de entrega. Si alcanzar una fecha de entrega dada es crítico para el funcionamiento del sistema, y puede causar consecuencias catastróficas, esa fecha de entrega se considera dura o *"hard"*. Si cumplir una restricción de tiempo es deseable, pero si no se cumple esta restricción no ocurre ningún daño serio, entonces esta fecha límite se considera suave o *"soft"*. Sumado a su criticalidad, las tareas que requieren activaciones regulares se denominan periódicas, mientras que aquellas que tienen instantes de llegada irregulares se consideran aperiódicas, aquellas tareas aperiódicas con fechas de entrega duras se denominan esporádicas.

Dado un conjunto de tareas de tiempo real, se dice que una planificación es factible si todas las tareas con restricciones duras son completadas dentro de sus fechas límite. Una tarea crítica con una fecha de entrega dura se dice garantizada en su instante de activación si el sistema es capaz de encontrar una planificación factible para la nueva tarea que acaba de llegar y para todas las tareas previamente garantizadas. Un sistema operativo capaz de garantizar y ejecutar tareas con restricciones de tiempo duras se denomina sistema *"Hard Real-Time (HRT)"*. En una aplicación crítica, la meta de un Sistema de Tiempo Real con Restricciones Duras no es sólo cumplir las fechas límite de todas las tareas duras, sino también minimizar el tiempo medio de respuesta para las actividades con restricciones suaves.

El problema de planificar un conjunto mezclado de tareas periódicas con restricciones de tiempo duras y tareas aperiódicas en un entorno dinámico ha sido ampliamente tratado cuando las tareas periódicas se ejecutan bajo la planificación del algoritmo *"Rate Monotonic"* (RM). Lehoczky, Sha y Strosnider investigaron mecanismos servidor, como el Servidor Diferido, para mejorar la respuesta a peticiones aperiódicas. La idea básica era usar una tarea periódica especiar para servir eficientemente posibles peticiones aperiódicas de ejecución. Sprunt, Sha y Lehoczky describieron un mecanismo de servicio mejor, llamado Servidor Esporádico (versión clásica, no la dinámica que se tratará en este trabajo). Entonces, Lehoczky y Ramos-Thuel encontraron un método de servicio óptimo, llamado *"Slack Stealer"*, el cual se basaba en "robar" todo el tiempo de procesamiento posible de tareas periódicas, sin provocas que estas venzan sus fechas límite. Aunque no es práctico, debido a su alto retraso, el algoritmo proporciona un límite inferior para los tiempos de respuesta, así como la base para algoritmo implementables casi óptimos.

Todos estos métodos asumen que las tareas periódicas han sido planificadas con el algoritmo *"Rate Monotonic"*. Aunque el algoritmo RM es óptimo, es estático y en el caso general no pueden conseguir una utilización total del procesador. En el peor caso, la máxima utilización del procesador obtenida está en torno al 69%, mientras que en el caso medio, para un conjunto de tareas aleatorio, esta utilización máxima ronda el 88%.

Para ciertas aplicaciones que requieren una alta carga de trabajo del procesador, un 69% o un 88% de límite de utilización puede suponer una seria limitación. La utilización del procesador puede aumentarse usando algoritmos de planificación dinámica, tales como *"Earliest Deadline First"* o algoritmo *"Least Slack".* Ambos se han demostrado óptimos y obtienen una completa utilización del procesador, aunque el algoritmo *"Earliest Deadline First"* puede ejecutarse con un menor retraso.

Cuando se trabaja con conjuntos de tareas híbridos (tareas aperiódicas con restricciones suaves y tareas periódicas con restricciones duras), el objetivo principal del *kernel* es garantizar la planificabilidad de todas las tareas críticas en las condiciones del peor-caso y proveer un buen tiempo medio de respuesta para las tareas con restricciones suaves y sin restricciones. Para las tareas aperiódicas planificadas con el algoritmo *"Earliest Deadline First",* se han propuesto test de aceptación para garantizar tareas

esporádicas singulares, o grupos de precedencia para tareas aperiódicas relacionadas. Aunque desde el punto de vista de utilización del procesador es óptimo, estos test de aceptación presentan un retraso demasiado grande para ser prácticos en aplicaciones de mundo real.

En este trabajo se incluyen a parte del Servidor Diferido ya mencionado antes en esta introducción, dos algoritmos con distintos tiempos de retraso debidos a la implementación y distintos rendimientos, estos son el Servidor de Intercambio de Prioridad Dinámico y el Servidor Esporádico Dinámico, los cuales son extensiones de sus versiones clásicas, las cuales trabajaban bajo planificación RM.

2 Descripción de los algoritmos

2.1 Servidor diferido *("Deferred Server" o "Deferrable Server")*

El algoritmo Servidor Diferido es un servicio técnico introducido por Lehoczky, Sha, y Strosnider (1987), el cual es compatible con la planificación RM (*"Rate Monotonic*"). El algoritmo *"Rate Monotonic"* asigna prioridades en relación inversa a los periodos de las tareas, es decir, cuanto menor sea el periodo de la tarea mayor será su prioridad. Este algoritmo se diseñó para mejorar el tiempo medio de respuesta de las peticiones aperiódicas con respecto al servicio de sondeo ("Polling Service"), lo cual se verifica ampliamente, a la vez que sigue cumpliendo las fechas de vencimiento de las tareas periódicas.

Al igual que el Servidor de Sondeo, el algoritmo de Servidor Diferido crea una tarea periódica (normalmente con una alta prioridad, como se verá en el ejemplo) para servir peticiones aperiódicas. Esta tarea periódica tendrá un periodo T_s y capacidad C_s. Este servidor se usa para proporcionar servicio de alta prioridad a las tareas aperiódicas.

A diferencia del Servidor de Sondeo el Servidor Diferido preserva su capacidad si no hay peticiones pendientes para la invocación del servidor. La capacidad se mantiene hasta el final del periodo, de forma que las peticiones aperiódicas pueden ser servidas con la misma prioridad en cualquier momento, siempre que la capacidad no se haya agotado. Al principio de cualquier periodo de servicio, la capacidad es reestablecida a su valor total. El Servidor Diferido se planifica como si fuese una tarea periódica con periodo T_S.

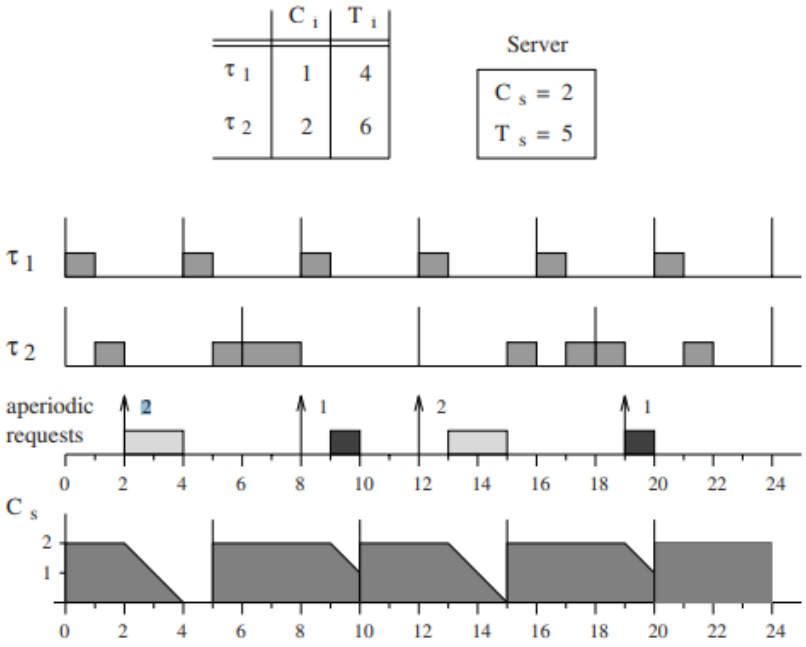
En la figura 2.1.1 se observa un ejemplo de servidor diferido planificado con RM (*Rate Monotonic"*). En esta se puede ver que hasta el instante t=2 no se ve afectada la capacidad del servidor, pues hasta ese momento no hay peticiones aperiódicas. Tras la ejecución de la tarea aperiódica la capacidad del servidor se restaura a su totalidad. También se ve que en los instantes t=8 y t=12, si bien hay peticiones aperiódicas estas deben esperar, pues está en ejecución τ_1 que tiene más prioridad.

Así, el algoritmo de Servidor Diferido proporciona una respuesta a las tareas aperiódicas mucho mejor que la del algoritmo Servidor de Sondeo, ya que preserva la

capacidad hasta que es necesitada. Se pueden obtener menores tiempo de respuesta creando un Servidor Diferido que tenga la mayor prioridad entre todas las tareas periódicas.

Más adelante, Sprunt describe un mejor mecanismo de servicio que el servidor diferido, que se denomina Servidor Esporádico, el cual se ampliará para que trabaje con el algoritmo de planificación *"Earliest Deadline First"*, bajo el nombre de Servidor Esporádico Dinámico.

Fig. 2.1.1. Ejemplo de Servidor Diferido con planificación RM (*"Rate Monotonic"*)



2.2 Servidor de Intercambio de Prioridad Dinámico (*"Dynamic Priority Exchange Server"*)

El Servidor de Intercambio de Prioridad Dinámico (DPE) es una técnica de servicio aperiódica propuesta por Spuri y Buttazzo que puede ser vista como una extensión del Servidor De Intercambio de Prioridad, adaptado para trabajar con un algoritmo de planificación basado en las fechas límite. Este es una extensión del Servidor de Intercambio de Prioridad adaptado para trabajar con el algoritmo *"Earliest Deadline First"*.

La idea principal del algoritmo es dejar al servidor intercambiar su tiempo de ejecución con el tiempo de ejecución de tareas periódicas con una menor prioridad (bajo la

condición de la primera fecha límite primero, esto implica una mayor fecha límite) en caso de que no haya peticiones aperiódicas pendientes. De esta manera, el tiempo de ejecución del servidor es solo intercambiado con tareas periódicas, pero nunca malgastado (a menos que hay tiempos ociosos). Es simplemente preservado, incluso si a un nivel de prioridad menor, y puede ser reclamado más tarde cuando una petición aperiódica entre al sistema.

El algoritmo se define de la siguiente manera:

- El Servidor de Intercambio de Prioridad Dinámico tiene especificados un periodo T_s y una capacidad C_s.

- Al principio de cada periodo, la capacidad aperiódica del servidor se establece en C_s^d, donde d es la fecha límite del actual periodo de servicio.

- Cada fecha límite, d, asociada a las instancias de la *i-ésima* tarea periódica (completa o no) tiene una capacidad aperiódica, $C_{S_i}^d$, inicialmente establecida a 0.

- Las capacidades aperiódicas (aquellas mayores que 0) reciben prioridades conforme a sus fechas límites y al algoritmo EDF ("*Earliest Deadline First"*), como todas las instancias de tareas periódicas (los lazos se rompen en favor de las capacidades, es decir, de las peticiones aperiódicas).

- Siempre que la entidad de mayor prioridad en el sistema sea una capacidad aperiódica de C unidades de tiempo, ocurre lo siguiente:

  - Si hay peticiones aperiódicas en el sistema, estas son servidas hasta que se completen o se agote la capacidad (cada petición consume capacidad igual a su tiempo de ejecución).

  - Si no hay peticiones aperiódicas pendientes, la tarea periódica con la fecha límite más cercana se ejecuta. Una capacidad igual a la longitud de la ejecución se añade a la capacidad aperiódica de la fecha límite de la tarea y se substrae de C (esto es, las fechas límites de la capacidad con mayor prioridad y las tareas periódicas se ejecutan).

  - Si no hay ni peticiones aperiódicas ni instancias de tareas periódicas pendientes, hay un tiempo ocioso, y la capacidad C se consume hasta que, como mucho, se agote.

Para implementar el algoritmo, las únicas operaciones requeridas en caso de intercambio de fechas limites, son la actualización de valores de dos capacidades y la comprobación de si la capacidad que se está ejecutando se ha agotado. Además, la cola de tareas listas puede ser como mucho el doble de largo de lo que sería sin el servidor (hay al menos una capacidad aperiódica para cada instancia de tarea periódica). Desde estas simples observaciones podemos concluir que mientras la implementación de un

Servidor de Intercambio de Prioridad Dinámico es no trivial, el retraso en el tiempo de ejecución no aumenta significativamente el retraso típico en un sistema que utilice un planificador *"Earliest Deadline First"*.

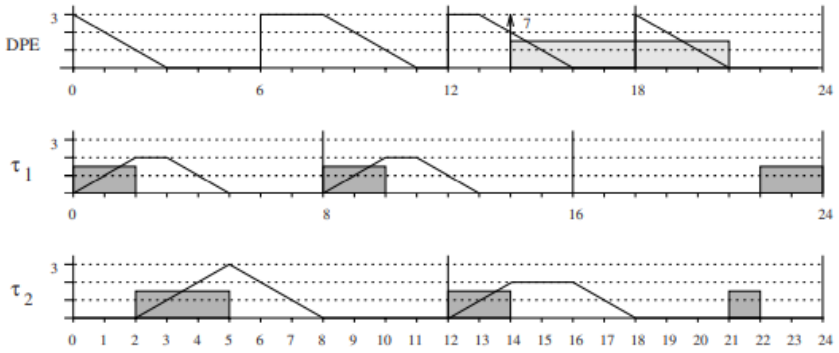
Un ejemplo de planificación producida por el algoritmo de Intercambio de Prioridades Dinámico se muestra en la figura 2.2.1. Dos tareas periódicas, τ_1 y τ_2, con periodos $T_1 = 8$ y $T_2 = 12$ y peores casos de ejecución $C_1 = 2$ y $C_2 = 3$, y un Servidor de Intercambio de Prioridad Dinámico con periodo $T_S = 6$ y capacidad $C_s = 3$, están presentes en el sistema.

En el instante $t = 0$, las capacidades aperiódicas $C_{S_1}^8$ (con fecha límite 8) y $C_{S_2}^{12}$ (con fecha límite 12) se establecen a 0, mientras la capacidad del servidor (con fecha límite 6) se establece a $C_s = C_S^6 = 3$. Como no hay peticiones aperiódicas pendientes, las dos primeras instancias periódicas de τ_1 y τ_2 se ejecutan y C_s es consumido en las primeras tres unidades de tiempo. En el mismo intervalo dos unidades de tiempo son acumuladas en $C_{S_1}^8$ y una unidad en $C_{S_2}^{12}$.

En el instante de tiempo $t = 3$, $C_{S_1}^8$ es la entidad con más prioridad del sistema. De nuevo, como no hay peticiones aperiódicas pendientes, τ_2 sigue ejecutando y las dos unidades de $C_{S_1}^8$ son consumidas y acumuladas en $C_{S_2}^{12}$. En las siguientes tres unidades de tiempo el procesador se encuentra ocioso, y $C_{S_2}^{12}$ se consume completamente. Nota que en el instante de tiempo $t = 6$ la capacidad del servidor $C_s = C_S^{12}$ se establece a 3 y se preserva hasta el instante de tiempo $t = 8$, cuando se convierte en la entidad con más prioridad del sistema (los lazos entre las capacidades aperiódicas se asumen rotos en un orden FIFO). En el instante de tiempo $t = 8$, dos unidades de C_S^{12} son intercambiadas con $C_{S_1}^{16}$, mientras la tercera unidad del servidor es consumida pues el procesador está ocioso.

En el instante de tiempo $t = 14$, una petición aperiódica, J_1, de siete unidades de tiempo de ejecución entra al sistema. Como $C_S^{18} = 2$, las dos primeras unidades de J_1 son servidas con fecha límite 18, mientras que las siguientes dos unidades son servidas con fecha límite 24, usando la capacidad $C_{S_2}^{24}$. Finalmente, las tres últimas unidades son también servidas con fecha límite 24, ya que en el instante de tiempo $t = 18$ la capacidad del servidor C_S^{24} se establece a 3.

Fig. 1.2.1. Ejemplo de Intercambio de Prioridad Dinámico



2.3 Servidor Esporádico (*"Dynamic Sporadic Server"*)

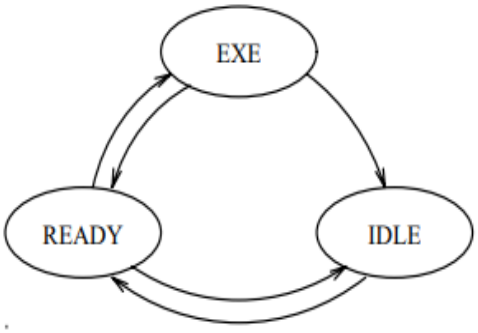
Sprunt, Sha y Lehoczky introdujeron en su libro *"Aperiodic Task Scheduling for Hard-Real-Time Systems"*, un algoritmo eficiente para servir peticiones aperiódicas en un sistema monoprocesador usando un planificador *"Rate Monotonic",* al se le conoce como *Sporadic Server*. Destacar que un algoritmo similar llamado *"Deadline Sporadic Server"* o Servidor Esporádico con Fecha Límite fue independientemente desarrollado por Ghazalie y Baker. Como este algoritmo exhibía propiedades como eficiencia y simplicidad, se ha estudiado como una política similar podía ser extendida para trabajar con un planificador EDF (*"Earliest Deadline First"*).

El Servidor Esporádico Dinámico (DSS) es una estrategia de servicio aperiódica propuesta por Spuri y Buttazo que extiende al Servidor Esporádico para trabajar bajo un planificador dinámico EDF (*"Earliest Deadline First"*). De forma parecida a otros servidores, el Servidor Esporádico Dinámico se caracteriza por un periodo T_s y una capacidad C_s, la cual es preservada para posibles peticiones aperiódicas. A diferencia de otros algoritmos de servidor, la capacidad no se rellena a su valor máximo al principio de cada periodo de servido sino sólo cuando ha sido consumida. Los instantes en los que ocurren los rellenos de la capacidad se eligen cuidadosamente de acuerdo a una regla de relleno, la cual permite al sistema conseguir una utilización total del procesador.

La principal diferencia entre el Servidor Esporádico clásico y la versión dinámica consiste en la forma en la que la prioridad se asigna al servidor. Mientras que el Servidor Esporádico clásico tiene una prioridad elegida de acuerdo al algoritmo RM (*"Rate Monotonic"*), esto es, acorde a su periodo T_s , el Servidor Esporádico Dinámico tiene una prioridad dinámica asignada por medio de una fecha límite adecuada.

Para describir el algoritmo de Servidor Esporádico Dinámico, consideremos los tres estados en los que el servidor, al igual que cualquier otra tarea del sistema, puede estar: OCIOSO (*IDLE)*, LISTO (*READY*) y EJECUTANDO (*EXE*). En la figura 2.3.1 se ilustran estos tres estados y las posibles transiciones entre ellos.

Fig. 2.3.1. Diagrama de transición de estados Servidor Esporádico Dinámico



El algoritmo de Servidor Esporádico se describe en términos de estas tres transiciones:

- Cuando se crea el servidor, su capacidad C_S se inicializa a su valor máximo
- $\boldsymbol{IDLE \rightarrow READY}$**:** o bien una petición aperiódica ha entrado en el sistema y $C_s > 0$, o C_s se ha vuelto mayor que cero (ha ocurrido una renovación de la capacidad). En cualquier caso, el siguiente instante de renovación de la capacidad RT y la actual fecha límite del servidor son establecidos los dos a $t + T_s$, donde t es el instante d etiempo actual.
- $\boldsymbol{READY \rightarrow IDLE}$: el servidor tiene la mayor prioridad en el sistema (es decir, la menor fecha de entrega), pero no hay peticiones aperiódicas que servir.
- $\boldsymbol{READY \rightarrow EXE}$: el servidor tiene la mayor prioridad en el sistema y hay peticiones aperiódicas que servir. Mientras estas peticiones son servidas, una capacidad igual a los instantes de ejecución utilizados por la petición aperiódica es consumida.
- $\boldsymbol{EXE \rightarrow READY}$**:** el servidor es adelantado por una tarea de mayor prioridad.
- $\boldsymbol{EXE \rightarrow IDLE}$: o bien una petición aperiódica ha sido servida y no hay peticiones pendientes, o la capacidad del servidor se ha agotado. Una renovación del tiempo de ejecución del servidor se planifica para que ocurra en el instante de renovación de la capacidad del servidor establecido durante la última transición $IDLE \rightarrow READY$.

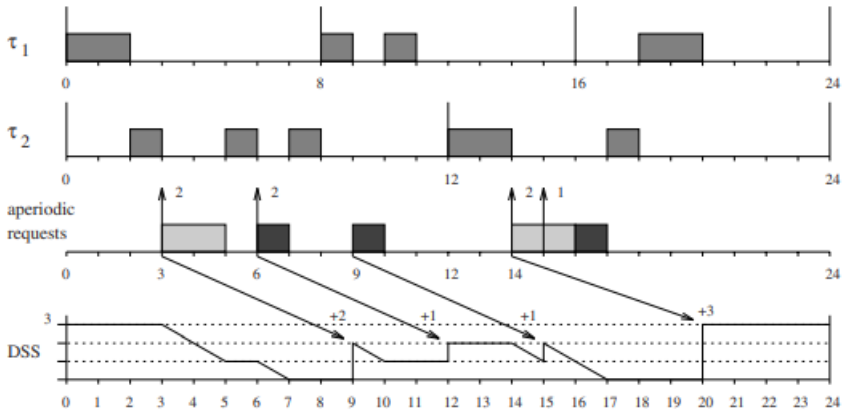
La figura 2.3.2 ilustra una planificación EDF (*"Earliest Deadline First"*) obtenida para un conjunto de instrucciones consistente en dos tareas periódicas con periodos $T_1 = 8$, $T_2 = 12$ y tiempos de ejecución $C_1 = 2$, $C_2 = 3$, y un Servidor Esporádico Dinámico con un periodo $T_s = 6$ y una capacidad $C_s = 3$.

En el instante de tiempo $t = 0$, la capacidad del servidor se inicializa a su valor total $C_s = 3$. Como no hay peticiones aperiódicas pendientes, el procesador es asignado a la tarea τ_1, el cual tiene la fecha límite más temprana. En el instante de tiempo $t = 3$, llega una petición aperiódica con tiempo de ejecución 2, y como $C_s > 0$, el primer instante de relleno y la fecha límite del servidor son establecidas a $RT_1 = d_s = 3 + T_s = 9$. Siendo d_s la fecha límite más temprana, el Servidor Esporádico Dinámico se convierte en la tarea con la mayor prioridad en el sistema y la petición es servida hasta que se complete. En el instante de tiempo $t = 5$, la petición es completada y ninguna otra petición periódica esta pendiente, luego se programa una reposición de dos unidades de tiempo para el instante $RT_1 = 9$.

En el instante $t = 6$, una segunda petición aperiódica llega. Siendo $C_s > 0$, el siguiente instante de relleno y una nueva fecha límite del servidor son establecido, con $RT_2 = d_S = 6 + T_s = 12$. De nuevo, el servidor se convierte en la tarea con más prioridad en el sistema (asumimos que los lazos entre tareas son siempre resueltos a favor del servidor) y la petición recibe servicio inmediato. En este instante la capacidad del servidor es de una solo unidad de tiempo, y esta se ve agotada en el instante de tiempo $t = 7$. Como consecuencia, una renovación de una unidad de tiempo se programa para el instante $RT_2 = 12$, y la tarea aperiódica se ve retrasada hasta el instante de tiempo $t = 9$, cuando C_s vuelve a ser de nuevo mayor que 0. En el instante de tiempo $t = 9$, el siguiente instante de relleno y la nueva fecha límite del servidor son establecidas con $RT_3 = d_s = 9 + T_s = 15$. Una vez más, el Servidor Esporádico Dinámico se convierte la tarea con mayor prioridad del sistema, así, la petición aperiódica recibe servicio inmediato y se completa en el instante de tiempo $t = 10$. Una renovación de una unidad de tiempo se programa para que ocurra en el instante de tiempo $RT_3 = 15$.

Nótese que mientras la capacidad del servidor sea mayor que 0, todas las peticiones aperiódicas pendientes son ejecutadas con la misma fecha límite. En la figura 3 esto ocurre en el instante de tiempo $t = 14$, cuando las dos últimas peticiones aperiódicas son servidas con la misma fecha límite $d_S = 20$.

Fig. 2.3.2. Ejemplo de Servidor Esporádico Dinámico



3 Análisis de planificabilidad de los algoritmos

3.1 Servidor diferido *("Deferred Server")*

Todo análisis de escalabilidad relacionado al algoritmo RM (*"Rate-Monotonic"*) se ha hecho con la asunción implícita de que una tarea periódica no se puede suspender a sí misma, pero debe ejecutarse cuando sea la tarea lista para ejecutar con mayor prioridad. Es fácil ver que el Servidor Diferido viola esta asunción básica. De hecho, la planificación ilustrada en la figura 3.1.1 muestra que el Servidor Diferido no ejecuta en el instante de tiempo $t = 0$, aunque es la tarea lista para ejecutar con mayor prioridad, pero aplaza su ejecución hasta el instante $t = 5$, el cual es la llegada de la primera petición aperiódica.

Si una tarea periódica aplaza su ejecución cuando se podría ejecutar inmediatamente, entonces una tarea de menor prioridad podría vencer su fecha límite incluso si el conjunto de tareas es planificable. La figura 3.1.2 ilustra este fenómeno comprando la ejecución de una tarea periódica con la ejecución de un Servidor Diferido con el mismo periodo y tiempo de ejecución.

El conjunto de tareas periódicas considerado en este ejemplo consiste en dos tareas τ_1 y τ_2, teniendo ambas el mismo tiempo de ejecución ($C_1 = C_2 = 2$) y distintos periodos ($T_1 = 4, T_2 = 5$). Como se muestra en la figura 5.a, las dos tareas son planificables por el algoritmo *"Rate-Monotonic"*. De todas formas, si τ_1 se reemplaza por un Servidor Diferido que tenga el mismo periodo y tiempo de ejecución, la tarea de prioridad más baja, τ_2, puede vencer su fecha de entrega dependiendo en la secuencia de llegada de tareas aperiódicas. La figura 5.b muestra una secuencia de peticiones aperiódicas particular que puede causar que τ_2 venza su fecha de entrega en el instante de tiempo $t = 5$. Esto ocurres porque, en el instante de tiempo $t = 8$, el Servidor Diferido no

ejecuta (como haría una tarea periódica normal) pero preserva su capacidad para futuras peticiones. Esta ejecución aplazada, seguida por servir dos peticiones aperiódicas consecutivas en el intervalo [10,14], previene a la tarea τ_2 de ejecutar durante este intervalo, provocando que se venza su fecha límite.

Fig. 3.1.1. Ejemplo Servidor Diferido con Alta prioridad

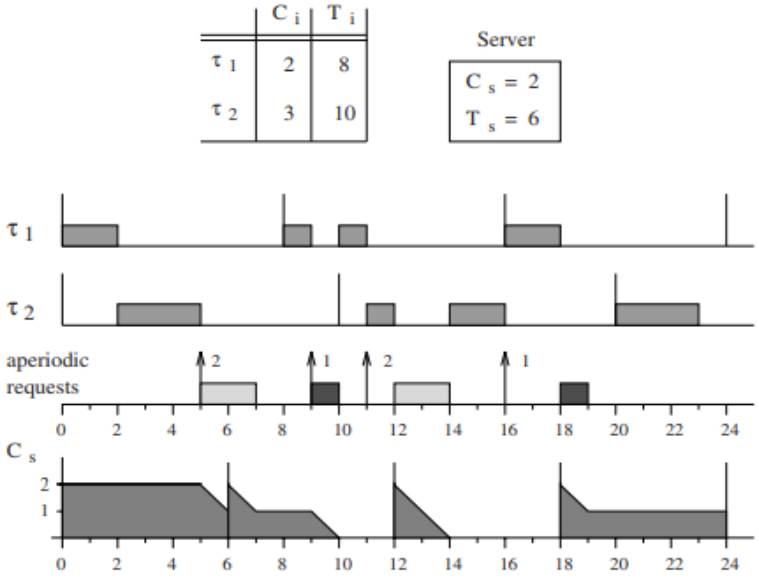
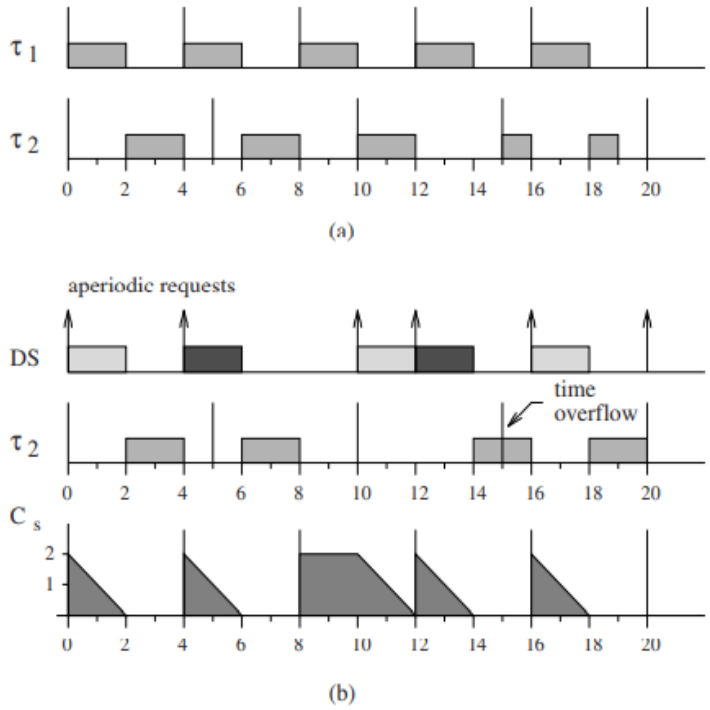


Fig.3.1.2. El Servidor Diferido no es equivalente a una tarea periódica. De hecho, el conjunto periódico $\{\tau_1, \tau_2\}$ es planificable por RM(a); Si reemplazamos τ_1 con DS, τ_2 vence su fecha(b)

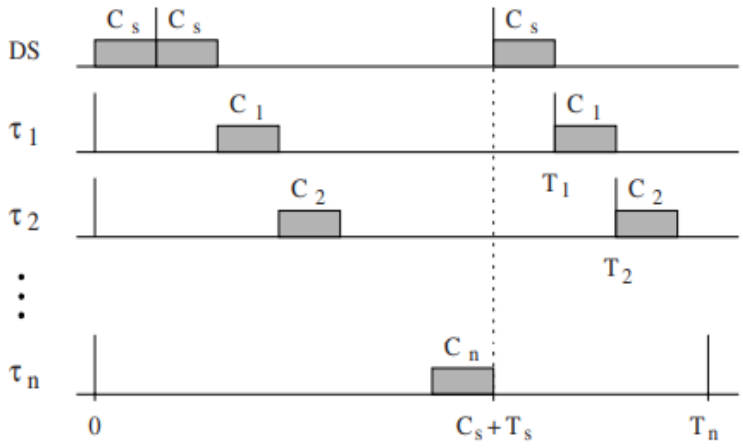


Cálculo del supremo, U_{LUB}, para Servidor Diferido con planificación RM.

Para facilitar el cálculo del límite para un conjunto de tareas periódicas con *n* tareas, primero determinamos las relaciones del tipo peor-caso entre las tareas, y entonces obtenemos el menor límite respecto al modelo del peor caso.

Considérese un conjunto de *n* tareas periódicas, $\tau_1, \ldots, \tau_n$, ordenado por orden creciente de periodos, y un Servidor Diferido con mayor prioridad. La condición del peor-caso para tareas periódicas, al igual que se obtene para el análisis *"Rate-Monotonic",* es tal que $T_1 < T_n < 2T_1$. En presencia de un Servidor Diferido, la obtención del peor caso es más compleja y requiere el análisis de tres casos diferentes, tal y como propusieron Strosnider, Lehoczky, y Sha. Aquí se analizará sólo un caso, el más general, en el que el Servidor Diferido puede ejecutarse tres veces dentro del periodo de la tarea periódica con mayor prioridad. Esto ocurre cuando el Servidor Diferido aplaza su servicio al final de su periodo y también se ejecuta al principio del siguiente periodo. En esta situación, representada en la figura 3.1.3, el completa utilización del procesador se consigue a través de los siguientes parámetros de las tareas:

Fig. 3.1.3. Relaciones entre tareas del tipo peor-caso para un Servidor Diferido



$$\begin{cases} C_s = T_1 - (T_s + C_s) = \frac{T_1 - T_s}{2} \\ C_1 = T_2 - T_1 \\ C_2 = T_3 - T_2 \\ \dots \\ C_{n-1} = T_n - T_{n-1} \\ C_n = T_s - C_s - \sum_{i=1}^{n-1} C_i = \frac{3T_s + T_1 - 2T_n}{2} \end{cases}$$

Por lo tanto, la utilización resultante es,

$$U = \frac{C_s}{T_s} + \frac{C_1}{T_1} + \cdots + \frac{C_n}{T_n}$$

$$= U_s + \frac{T_2 - T_1}{T_1} + \cdots + \frac{T_n - T_{n-1}}{T_{n-1}} + \frac{3T_s + T_1 - 2T_n}{2T_n}$$

$$= U_s + \frac{T_2}{T_1} + \cdots + \frac{T_n}{T(n-1)} + \frac{\left(\frac{3T_s}{2T_1} + \frac{1}{2}\right) T_1}{T_n} - n$$

Definiendo

$$\begin{cases} R_s = \frac{T_1}{T_s} \\ R_i = \frac{T_{i+1}}{T_i} \\ K = \frac{1}{2}\left(\frac{3T_s}{T_1} + 1\right) \end{cases}$$

y nótese que

$$R_1 R_2 \dots R_{n-1} = \frac{T_n}{T_1}$$

el factor de utilización se puede escribir como

$$U = U_s + \sum_{i=1}^{n-1} R_i + \frac{K}{R_1 R_2 \dots R_{n-1}} - n$$

esto es, cuando todos los R_i tienen el mismo valor:

$$R_1 = R_2 = \cdots = R_{n-1} = K^{\frac{1}{n}}$$

Sustituyendo este valor en U obtenemos

$$U_{lub} - U_s = (n-1)K^{\frac{1}{n}} + \frac{K}{K^{1-\frac{1}{n}}} - n =$$

$$= nK^{\frac{1}{n}} - K^{\frac{1}{n}} + K^{\frac{1}{n}} - n = n\left(K^{\frac{1}{n}} - 1\right)$$

esto es,

$$U_{lub} = U_s + n\left(K^{\frac{1}{n}} - 1\right). \qquad (1)$$

Ahora, viendo que

$$U_s = \frac{C_s}{T_s} = \frac{T_1 - T_s}{2T_s} = \frac{R_s - 1}{2}$$

tenemos que

$$R_s = (2U_s + 1).$$

Así, *K* puede escribirse como

$$K = \left(\frac{3}{2R_s} + \frac{1}{2}\right) = \frac{U_s+2}{2U_s+1},$$

y finalmente

$$U_{lub} = U_s + n[\left(\frac{U_s+2}{2U_s+1}\right)^{\frac{1}{n}} - 1] \quad (2)$$

Tomando el límite con $n \to \infty$, obtenemos el límite del peor caso como una función de U_s dada por

$$\lim_{n\to\infty} U_{lub} = U_s + (\frac{U_s+2}{2U_s+1}). \quad (3)$$

Una representación gráfica de la ecuación (3) se muestra en la figura 3.1.4. Para comparar, el límite obtenido para *"Rate-Monotonic"* también se muestra en la representación. Nótese que debido a que $U_s < 0.4$ la presencia del Servidor Diferido empeora el límite RM, mientras para $U_s > 0.4$ se mejora el límite obtenido con RM.

Si desarrollamos la ecuación (3) con respecto a U_s, podemos encontrar el valor mínimo absoluto de U_{lub}:

$$\frac{\partial U_{lub}}{\partial U_s} = 1 + \frac{2U_s+1}{U_s+2}\frac{(2U_s+1)-2(U_s+2)}{(2U_s+1)^2} = \frac{2U_s^2+5U_s-1}{(U_s+2)(2U_s+1)}$$

El valor de U_s que minimiza la expresión anterior es

$$U_s^* = \frac{\sqrt{33}-5}{4} \approx 0.186$$

por lo tanto el mínimo valor de U_{lub} es $U_{lub}^* \approx 0.652$

En resumen, dado un conjunto de *n* tareas periódicas con utilización del procesador U_p y un Servidor Diferido con utilización del procesador U_s, respectivamente, la planificabilidad del conjunto de tareas periódicas está garantizada por RM si

$$U_p \leq n(K^{\frac{1}{n}} - 1). \quad (4)$$

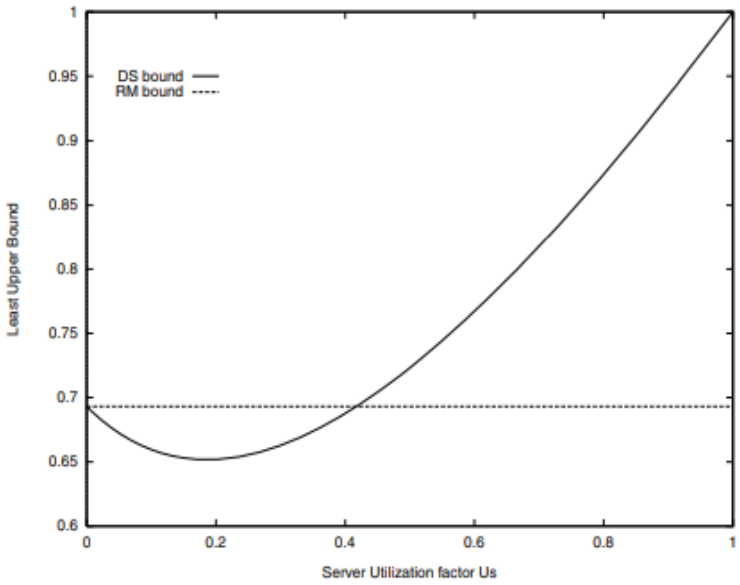
Donde,

$$K = \frac{U_s+2}{2*U_s+1},$$

Usando el límite hiperbólico, el test de garantía para un conjunto de tareas en presencia de un Servidor Diferido se puede realizar como sigue:

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s+2}{2U_s+1} \quad (5)$$

Fig. 3.1.4. Límite de escalabilidad para tareas periódicas y Servidor Diferido como una función dependiente del factor de utilización del servidor U_s



Dimensionando un Servidor Diferido

La utilización máxima, U_s^{max}, para un Servidor Diferido se puede calcular fácilmente con la ecuación (5), la cual se puede escribir, definiendo P como una ecuación (6)

$$P = \prod_{i=1}^{n}(U_i + 1) \quad (6)$$

como,

$$P \leq \frac{U_s+2}{2U_s+1};$$

esto es,

$$U_s \leq \frac{2-P}{2P-1}.$$

Por lo tanto,

$$U_s^{max} = \frac{2-P}{2P-1} \quad (7)$$

Entonces, T_s puede establecer igual al menor periodo T_1, de manera que el Servidor Diferido sea ejecutado con el algoritmo de planificación *"Rate Monotonic"* con su mayor prioridad (asumiendo que los lazos de prioridad son rotos en favor del servidor), y finalmente $C_s = U_s T_s$.

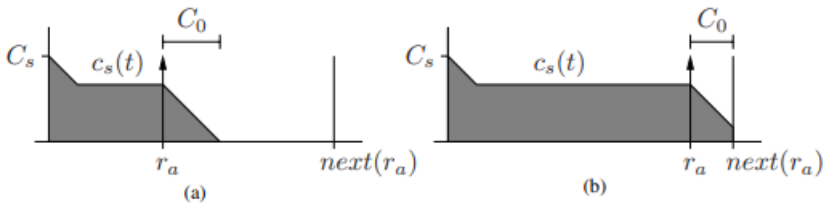
Garantía aperiódica

La garantía de un trabajo aperiódico firmado puede ser realizado estimando el tiempo de respuesta de su peor ejecución en el caso de un Servidor Diferido con la mayor prioridad. Como el Servidor Diferido preserva su tiempo de ejecución, sea $C_S(t)$ el valor de su capacidad en el instante de tiempo t, y sea J_a un trabajo aperiódico con tiempo de computación C_a y fecha límite relativa D_a, que llega en el instante de tiempo $t = r_a$, cuando no hay ninguna otra petición aperiódica pendiente. Entonces, si $next(r_a) = \left\lceil \frac{r_a}{T_s} \right\rceil T_s$ es la siguiente activación del servidor tras r_a unidades de tiempo, los dos casos ilustrados en la figura 3.1.5 pueden ocurrir:

1. Caso (a): $c_s(t) \leq next(r_a) - r_a$. En este caso, la capacidad es completamente descargada dentro del periodo actual y una porción $C_0 - c_s(t)$ de J_a es ejecutado en el periodo del servidor actual.

2. Caso (b): $c_s(t) > next(r_a) - r_a$. En este caso, el periodo termina antes de que la capacidad del servidor se descargue completamente. Así una porción $C_0 = next(r_a) - r_a$ de J_a es ejecutada en el periodo del servidor actual.

Fig.3.1.5. Ejecución de J_a en el primer periodo del servidor



En general, la porción C_0 ejecutada en el periodo del servidor actual es igual a

$$C_0 = min\{c_s(t), next(r_a) - r_a\}.$$

Se define:

$$\begin{cases} \Delta_a = next(r_a) - r_a \\ F_a = \left\lceil \frac{C_a - C_0}{C_s} \right\rceil - 1 \\ \delta_a = C_a - C_0 - F_a C_s \end{cases}$$

Por lo tanto, como se representa en la figura 3.1.6, el tiempo de respuesta R_a de un trabajo J_a puede ser calculado como

$$R_a = \Delta_a + F_a T_s + \delta_a,$$

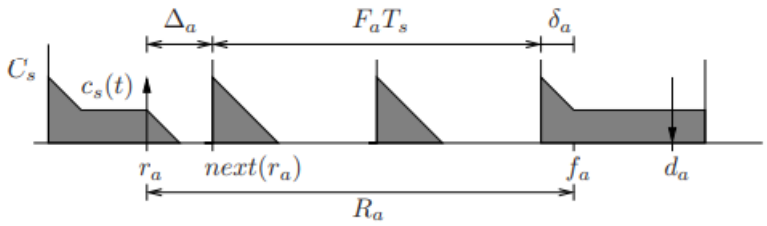
el cual también se puede escribir como:

$$R_a = \Delta_a + C_a - C_0 + F_a(T_s - C_s). \quad (8)$$

Nótese que el término $F_a(T_s - C_s)$ en la ecuación (8) representa el retraso introducido por los F_a intervalos inactivos del servidor, cada uno de tamaño $(T_s - C_s)$.

Entonces la planificabilidad de un trabajo aperiódica se puede garantizar si y sólo si $R_a \leq D_a$.

Fig.3.1.6. Tiempo de respuesta de un trabajo aperiódico planificado por un Servidor Diferido con la mayor prioridad



3.2 Servidor de Intercambio de Prioridad Dinámico (*"Dynamic Priority Exchange Server"*)

En esta subsección se analizará la condición de planificabilidad para un conjunto de tareas periódicas que se planifican junto con un Servidor de Intercambio de Prioridad Dinámico. Intuitivamente, con el algoritmo que se muestra más abajo. El servidor se comporta como otra tarea periódica. La diferencia es que puede intercambiar su tiempo de ejecución con el tiempo de ejecución de tareas con prioridad más baja. Cuando una cierta cantidad de tiempo ha sido intercambiada, una o varias tareas con prioridad más baja se ejecutan con un nivel de prioridad mayor, y su tiempo de ejecución de prioridad más baja se preserva para la posible ejecución de tareas aperiódicas. Este intercambio de tiempo de ejecución no afecta a la planificabilidad, como se muestra a continuación.

Como siempre, definiremos el factor de utilización de las tareas periódicas como

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

y el factor de utilización del servidor como

$$U_s = \frac{C_s}{T_s}$$

Nuestro objetivo es probar que el resultado clásico de Liu y Layland ($U_p + U_s \leq 1$) para un planificador *"Earliest Deadline First"* se puede extender incluyendo el factor de utilización del servidor. Para hacer esto, dado una planificación σ producida usando el algoritmo de Intercambio Dinámico de Prioridad, considérese una planificación σ' construida de la siguiente manera:

- Se sustituye el Servidor de Intercambio de Prioridad Dinámica con una tarea periódica τ_s con un periodo T_s y un tiempo de ejecución en el peor de los casos C_s, de forma que en σ', τ_s ejecuta cuando la capacidad del servidor sea consumida en σ.

- La ejecución de instancias periódicas durante un intercambio de fechas de entrega se pospone hasta que la capacidad decrezca.

- Todas las demás ejecuciones de instancias periódicas se dejan como en σ.

Nótese qué de la definición de algoritmo de Intercambio Dinámico de Prioridad, en cualquier instante, al menos una capacidad aperiódica decrece en σ, de forma que σ' está bien definida. También obsérvese que, en cada planificación factible producida por el algoritmo de Intercambio Dinámico de Prioridad, todas las capacidades aperiódicas se agotan antes del vencimiento de sus respectivas fechas de entregas.

La figura 3.2.1 muestra la planificación σ' obtenida de la planificación σ de la figura 2. Nótese que todas las ejecuciones periódicas correspondientes a incrementos de capacidades aperiódicas han sido movidas a los correspondientes intervalos en los cuales las mismas capacidades decrecen. También nótese que la planificación σ' no depende de las peticiones aperiódicas, sino que depende sólo de las características del servidor y del conjunto de tareas periódicas. Basado en esta observación se puede probar el siguiente teorema:

Teorema 1 (Spuri, Butazzo) *Dado un conjunto de tareas periódicas con utilización del procesador U_p y un Servidor de Intercambio de Prioridad Dinámico con utilización del procesador U_s. El conjunto completo es planificable si y sólo si*

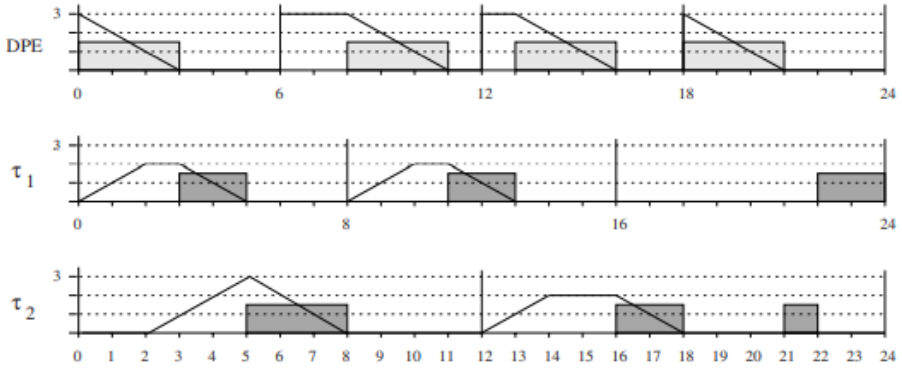
$$U_p + U_s \leq 1$$

Demostración. Para una carga aperiódica, todas las planificaciones producidas por el algoritmo de Intercambio de Prioridad Dinámico tienen una única planificación *"Earliest Deadline First"* correspondiente, construida de acuerdo a la definición ya dada. Es más, el conjunto de tareas en σ' es periódico con una utilización del procesador $U = U_p + U_s$. Por lo tanto, σ' es factible si y sólo si $U_p + U_s \leq 1$. Ahora mostramos que σ es factible si y sólo si σ' es factible.

Obsérvese que en cada planificación σ el tiempo en el que se completa una instancia periódica es siempre menor o igual que el tiempo en el que se completa la correspondiente instancia en la planificación σ'. Por lo tanto, si σ' es factible, entonces σ es

factible; esto es, el conjunto de tareas periódicas es planificable con el algoritmo de Intercambio de Prioridad Dinámico. Por el contrario, observando que σ' es una planificación particular producida por el algoritmo de Intercambio de Prioridad Dinámico, cuando hay suficientes peticiones aperiódicas, si σ es factible, entonces σ' será también factible. Por lo tanto, el teorema se mantiene. **Fin de la Demostración.**

Fig. 3.2.1. Planificabilidad Servidor de Intercambio Dinámico de Prioridad



Reclamando el tiempo libre.

En los sistemas de tiempo real más típicos, la carga del procesador de actividades periódicas, bien estadísticamente o dinámicamente, se garantiza *a-priori*. Esto significa que la máxima carga de trabajo posible alcanzable por tareas periódicas se tiene en cuenta. Cuando esta carga máxima no se alcanza, es decir, los tiempos actuales de ejecución son menores que los valores del peor caso, no siempre es obvio como reclamar el tiempo libre para actividades de tiempo real (una aproximación trivial sería ejecutar las tareas en segundo plano).

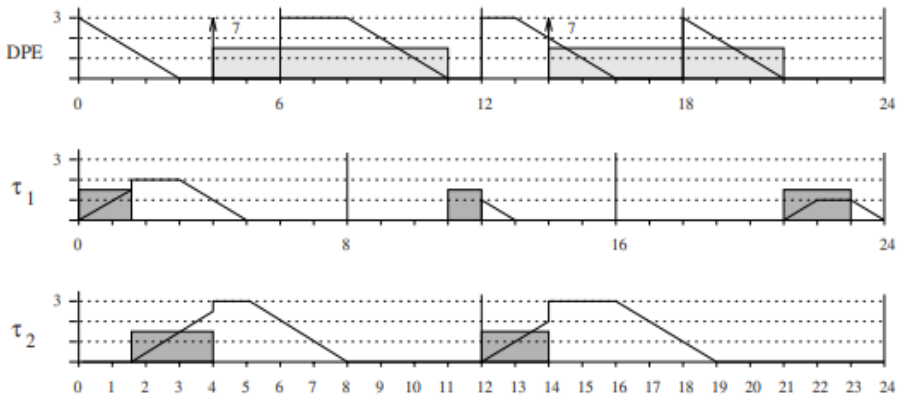
Usando un Servidor de Intercambio de Prioridad Dinámico, el tiempo libre no utilizado por las tareas periódicas se puede reclamar fácilmente para servir peticiones aperiódicas. Cuando una tarea periódica se completa, es suficiente añadir su tiempo libre a la correspondiente capacidad aperiódica. Un ejemplo del mecanismo de reclamación se muestra en la figura 3.2.2.

Como se puede ver por la representación gráfica de la capacidad, en el instante de finalización de las primera dos instancias periódicas, las correspondientes capacidades

aperiódicas ($C_{s_1}^{8}$ y $C_{s_2}^{12}$) son incrementadas con una cantidad igual al tiempo libre que se ha guardado. Gracias a este mecanismo de reclamación, la primera petición aperiódica puede recibir servicio inmediato para las siete unidades de tiempo que requiere, finalizando así en el instante de tiempo $t = 11$. Sin reclamación, esta petición se habría completado en el instante de tiempo $t = 12$.

Nótese que reclamando el tiempo libre de las tareas periódicas como capacidades aperiódicas no afecta a la planificabilidad del sistema. Es suficiente observar que, cuando una tarea periódica tiene tiempo libre, este tiempo ya ha sido *"reservado"* para un nivel de prioridad correspondiente con su fecha de entrega cuando el conjunto de tareas ha sido garantizado. Por lo tanto, el tiempo de espera se puede usar de forma segura si se pide con la misma fecha límite. Pero esto es exactamente lo mismo que añadirlo a la correspondiente capacidad aperiódica.

Fig. 3.2.2. Reclamación de recursos Servidor de Intercambio de Prioridad Dinámico



Complejidad de implementación

El Servidor de Intercambio de Prioridad siempre se ha dicho que es bastante difícil de implementar y que tiene bastante tiempo de sobrecarga (*"overhead"*). Ahora evaluamos la complejidad de la implementación de un Servidor de Intercambio de Prioridad Dinámico en un sistema monoprocesador utilizando *"Earliest Deadline First"* como algoritmo de planificación.

Si asumimos que el planificador y el *"dispatcher"* son capaces de manipular colas con dos tipos de entidades, tareas y capacidades aperiódicas, el servidos sólo incluye unas pocas operaciones más en la reserva de estas capacidades. En particular, en cada

"tic" del sistema puede ser necesario, en caso de intercambio de fecha límite, actualizar los valores de dos capacidades y comprobar si la capacidad en *"ejecución"* se ha agotado. El incremento de la complejidad, respecto a *"Earliest Deadline First",* es por supuesto muy bajo.

Además, la cola de tareas listas puede ser como mucho el doble de largo de lo que sería sin el servidor (hay al menos una capacidad aperiódica por cada instancia de tarea periódica). Esto es, la complejidad de las rutinas que manipulas esta cola puede ser como mucho doblada, si usamos una lista lineal (usando un montón o *"heap"* binario el incremento en complejidad es prácticamente despreciable).

De estas simples observaciones podemos concluir que mientras la implementación del Servidor de Intercambio de Prioridad Dinámico no es trivial, el retraso en el tiempo de ejecución no aumenta significativamente respecto al retraso de un sistema usando un planificador *"Earliest Deadline First"*.

3.3 Servidor Esporádico (*"Dynamic Sporadic Server"*)

Para probar el límite de planificabilidad del Servidor Esporádico Dinámico, primero veremos que el servidor se comporta como una tarea periódica con periodo T_s y tiempo de ejecución C_s.

Dada una tarea periódica τ_i, primero notamos que en cualquier intervalo genérico $[t_1, t_2]$ tal que τ_i es lanzado en el instante t_1, el tiempo de computación planificado por el algoritmo *"Earliest Deadline First"* con fecha límite menor o igual que t_2 es tal que

$$C_i(t_1, t_2) \leq \left\lceil \frac{t_2 - t_1}{T_i} \right\rceil C_i$$

El siguiente lema muestra que esta misma propiedad es cierta para el Servidor Esporádico Dinámico.

Lema 1. *En cada intervalo de tiempo* $[t_1, t_2]$*, tal que* t_1 *es el instante en el cual el Servidor Esporádico Dinámico pasa a estar preparado (esto es, una petición aperiódica llega y no se está sirviendo ninguna otra petición aperiódica), el máximo tiempo aperiódico ejecutado por el Servidor Esporádico Dinámico en* $[t_1, t_2]$ *satisface la siguiente relación:*

$$C_{ape} \leq \left\lceil \frac{t_2 - t_1}{T_s} \right\rceil C_s$$

Demostración. Como las renovaciones (me refiero a la renovación de la capacidad del servidor) son siempre iguales al tiempo consumido, la capacidad del servidor es en

cualquier instante menor o igual que su valor inicial. También, la política de renovación establece que la capacidad consumida no puede ser reclamada después de T_s unidades de tiempo tras el instante en el cual el servidor pasó a estar preparado. Esto significa que, desde el instante de tiempo t_1 en el cual el servidor pasa a estar preparado, como mucho C_s instante pueden ser consumidos en cada intervalo subsecuente de longitud T_s. Por lo tanto, el lema se verifica. **Fin de la demostración.**

Dado que el Servidor Esporádico Dinámico se comporta como una tarea periódica, el siguiente teorema establece que todavía se obtiene una completa utilización del procesador.

Teorema 2 (Spuri, Buttazzo) *Dado un conjunto de n tareas periódicas con una utilización del procesador U_py un Servidor Esporádico Dinámico con utilización del procesador U_s, el conjunto completo es planificable si y sólo si*

$$U_s + U_p \leq 1$$

Demostración. $\Rightarrow$. Asumimos $U_p + U_s \leq 1$ y suponemos que hay un desbordamiento en el instante de tiempo t. El desbordamiento es precedido por un periodo de utilización continua del procesador. Es más, desde un cierto punto t' en adelante ($t' < t$), solo las instancias de tareas listas en el instante t' o después y con fechas de entrega menores o iguales que el instante t se ejecutan (el servidor puede ser una de estas tareas). Sea C el tiempo total de ejecución demandado por estas instancias. Como hay un desbordamiento en el instante t, debemos tener $t - t' < C$. También sabemos que

$$C \leq \sum_{i=1}^{n} \left\lceil \frac{t-t'}{T_i} \right\rceil C_i + C_{ape}$$

$$\leq \sum_{i=1}^{n} \left\lceil \frac{t-t'}{T_i} \right\rceil C_i + \left\lceil \frac{t-t'}{T_s} \right\rceil C_s$$

$$\leq \sum_{i=1}^{n} \frac{t-t'}{T_i} C_i + \frac{t-t'}{T_s} C_s$$

$$\leq (t - t')(U_p + U_s)$$

De aquí, se sigue que

$$U_p + U_s > 1$$

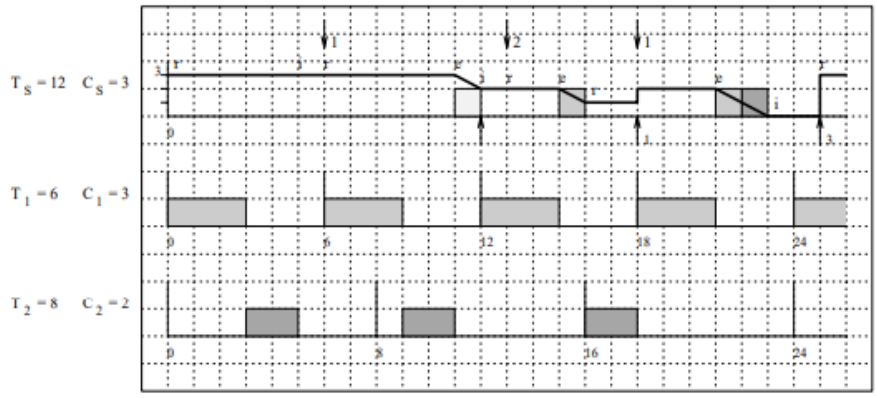
llegando así a una contradicción.

$\Leftarrow$. Como el Servidor Esporádico Dinámico se comporta como una tarea periódica con periodo T_s y tiempo de ejecución C_s, el factor de utilización del servidor es $U_s = \frac{C_s}{T_s}$ y el factor de utilización total del procesador es $U_p + U_s$. Por lo tanto, si el conjunto de

tareas completo es planificable, por el límite de planificabilidad del algoritmo *"Earliest Deadline First"* podemos concluir que $U_p + U_s \leq 1$. **Fin de la demostración.**

Atendiendo a las características y las propiedades del Servidor Esporádico, se puede ver fácilmente que, cuando el servidor tiene un periodo largo, la ejecución de peticiones aperiódicas se puede ver retrasada significantemente. Un ejemplo de esta situación se ilustra en la figura 3.3.1.

Fig. 3.3.1. Ejecuciones Aperiódicas Aplazadas con un Servidor Esporádico Dinámico.



Complejidad de implementación

Para implementar el algoritmo de Servidor Esporádico Dinámico, el *"dispatcher"* y el planificador, además de las entradas usuales correspondientes a tareas, deben ser capaces de coordinar la entrada al servidor en las colas del sistema. Esto sólo implica manejar una cola más (de tareas periódicas) asociadas al servidor.

De forma similar al Servidor de Intercambio de Prioridad Dinámico, el sistema de reservas está implicado en las actualizaciones de la capacidad del servidor. Esto ocurre en dos situaciones diferentes. Primero, durante los servicios de peticiones aperiódicas la capacidad debe ser decrementada en cada *"tic"* del sistema para así comprobar si esta se ha agotado en un determinado instante. Segundo, cuando un instante de renovación de la capacidad del servidor, que ha sido planificado, se alcanza, la capacidad debe ser incrementada con un valor específico.

De nuevo, podemos esperar que la implementación del Servidor Esporádico Dinámico sea bastante directa y el retraso en el tiempo de ejecución sea muy bajo.

4 Conclusiones

En este trabajo se han desarrollado tres algoritmos de planificación para sistemas de tiempo real, uno de ellos (*"Deferrable Server"*) estático y los otros dos (*"Dynamic Priority Exchange Server" y "Dynamic Sporadic Server"*) dinámicos.

Además, respecto a estos algoritmos se ha realizado un análisis de planificabilidad exhaustivo que nos va a facilitar tanto la comprensión como la explicación del funcionamiento de estos. Todo esto acompañado de gráficas y ejemplos visuales que ayudarán a la hora de concretar conceptos.

Se han presentado los fundamentos teóricos del Algoritmo de Servidor Diferido, el cual provee una solución al problema de planificar simultáneamente tareas periódicas como aperiódicas y con sus respectivas restricciones de tiempo, suaves y duras. Para este algoritmo se realizan ciertas comparaciones con el *"Polling Server"* que si bien no se ha tratado en este trabajo si se especifica que a diferencia de este algoritmo de Servidor Diferido preserva su capacidad de procesamiento lo que permite mejorar el tiempo medio de respuesta ampliamente, a la vez que sigue verificando el cumplimiento de las fechas de entrega de cada una de las tareas periódicas. Además, para el algoritmo de Servidor Diferido se han mostrado los cálculos necesarios el supremo de utilización del, así como hemos aplicado estos cálculos para un ejemplo práctico.

En cuanto a los algoritmos dinámicos, los dos algoritmos dinámicos anteriormente nombrados utilizan la muy conocida política de *"Earliest Deadline First"* para lidiar con las tareas periódicas de restricciones suaves y duras. Además, como punto extra para el algoritmo de Intercambio de Prioridad Dinámico se ha diseñado y puesto en práctica con un ejemplo un mecanismo de recuperación de recursos libres.

Respecto al Algoritmo de Intercambio de Prioridad Dinámico se ha observado que se trata de un algoritmo que extiende el algoritmo de Intercambio de Prioridad para que trabaje con el ya mencionado algoritmo *"Earliest Deadline First"*. Su propuesta de planificación nos garantiza el cumplimiento de las fechas límite de las tareas periódicas, lo cual se ha ilustrado con un ejemplo. También se han presentado las operaciones que se deberían realizar en caso de querer implementar este algoritmo y de aquí se ha deducido que la implementación de este algoritmo no es ni mucho menos trivial, de hecho, es más compleja en comparación con otros algoritmos de servidor dinámicos, y además el retraso en el tiempo de ejecución no aumenta significativamente.

En cuento al Algoritmo de Servidor Esporádico, se ha visto que la diferencia respecto a su versión clásica consiste en la forma en la que se asigna la prioridad al servidor, además de que este algoritmo es distinto de los otros ejemplos de servidor, pues lo normal es que la capacidad del servidor se rellenase cuando se cumplía un cierto periodo, pero en este caso, por su implementación rellena únicamente su capacidad cuando esta se ha consumido y en lugar de renovarse entera en cada instante de relleno

añade a la capacidad únicamente la cantidad consumida en el periodo entre que una tarea la consume y se rellena.

Bibliografía

1. J.P.Lehoczky, L.Sha , J.K. Strosnider. "Enhanced aperiodic responsiveness in hard-real time environments". Proceedings of the IEEE Real-Time Systems Symposium (1987)
2. M.Spuri, G.Buttazzo. "Efficient aperiodic service under earliest deadline scheduling". Proceedings of the IEEE Real-Time Systems Symposium (1994)
3. M.Spuri, G.Buttazzo. "Scheduling aperiodic tasks in dynamic priority systems". Journal of Real-Time Systems, 10, 2 (1996)
4. B.Sprunt, L. Sha, J.Lehoczky. "Aperiodic task scheduling for hard-real time systems". Journal of Real-Time Systems, July (1989)
5. Giorgio C. Butazzo. "Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications" (2nd Edition). Springer Verlag (2005)º