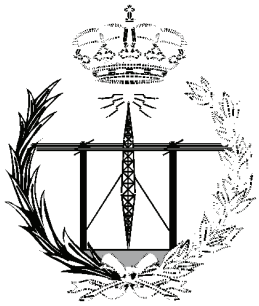
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Carrera

ESTUDIO DE LA ASIGNACIÓN DE PLAZOS EN SISTEMAS DISTRIBUIDOS DE TIEMPO REAL CON PLANIFICACIÓN EDF: PROPUESTA DE UN ALGORITMO HEURÍSTICO

(A Study on deadline assignment in EDF Distributed Real-Time Systems : proposal of a heuristic algorithm)

Para acceder al Titulo de

INGENIERO DE TELECOMUNICACIÓN

Autor: Juan María Rivas Concepción

27 Noviembre - 2008

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: Juan María Rivas Concepción
Director del PFC: José Javier Gutiérrez García
Título: "Estudio de la asignación de plazos en sistemas distribuidos de tiempo real con planificación EDF: propuesta de un algoritmo heurístico "
Title: **"**A Study on deadline assignment in EDF Distributed Real-Time Systems: proposal of a heuristic algorithm **"**

Presentado a examen el día: 27 de noviembre de 2008

para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Drake Moyano, José María
Secretario (Apellidos, Nombre): Gutiérrez García, José Javier
Vocal (Apellidos, Nombre): Zubillaga Rego, Agustín

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del PFC
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Proyecto Fin de Carrera Nº
(a asignar por Secretaría)

A mi padre. Papá, sé que lo estás viendo.

He de expresar mi profundo agradecimiento a todas aquellas personas que me han dado la oportunidad de desarrollarme, tanto intelectual como personalmente. Su influencia es de una importancia capital para poder llegar al punto en el que me encuentro, finalizando una carrera y el proyecto culmen de la misma.

No puedo nombrar a todos, pero si quiero reconocer específicamente el valor a algunos de ellos:

Mi familia, por su apoyo y empuje incondicional, y a mi madre en especial, que seguro que intentará leerlo todo.

José Javier Gutiérrez García, mi director de proyecto, por estar siempre disponible para ayudarme en todos los problemas que me iban surgiendo, y tener la paciencia de repasar toda la memoria para depurarla lo máximo posible.

A José Carlos Palencia por hacer la herramientas de análisis EDF, sin la cual no se podía haber llevado a cabo este proyecto.

Mis compañeros de clase, los cuales me han motivado a hacerlo lo mejor posible, con sus ejemplos de trabajo y excelencia en las distintas áreas de la carrera; y hacer que éstos años de estudio hayan "volado".

Y a todos aquellos que olvido nombrar.

Índice general

Índice de figuras

A Study on deadline assignment in EDF Distributed Real-Time Systems : proposal of a heuristic algorithm

Capítulo 1

Sistemas de Tiempo Real

Los Sistemas de Tiempo real son sistemas en los que existe una gran interacción con el entorno, que cambia con el tiempo. La característica principal de estos sistemas es que es tan importante el obtener un resultado correcto, como el obtenerlo en un determinado espacio de tiempo [2], o dicho de otra forma, poseen unos requisitos temporales, llamados también plazos, que deben cumplir. Típicamente están compuestos de un computador, dispositivos de entrada/salida y un *software* a ejecutar.

Son sistemas extensamente utilizados en una amplia variedad de campos, por ejemplo, en sistemas de control de vuelo en aviones, sistemas electrónicos de ayuda a la conducción (ABS, ESP, etc) en vehículos, sistemas de monitorización, etc.

Todos estos sistemas de tiempo real se implementan tanto en sistemas **monoprocesadores** como en **multiprocesadores**, siendo de especial interés en la actualidad los llamados **sistemas distribuidos**, formados por varios procesadores interconectados por una o varias redes de comunicación.

1.1. Modelo de Sistema de tiempo real

Llamamos *actividad* a cada unidad mínima de trabajo que se planifica y ejecuta en el sistema, y a un grupo de actividades relacionadas entre sí se le llama

transacción. Una actividad se dice que ha sido *activada* cuando está lista para la ejecución. Todas las actividades que se ejecutan en el sistema son activadas mediante un estímulo llamado *evento*. Éstos eventos típicamente pueden llegar al sistema de forma periódica o aperiódica. A las actividades que se ejecutan como consecuencia de la llegada de un evento también se les conoce como respuesta al evento.

Todas las actividades se ejecutan en algún recurso procesador. Estos recursos pueden ser procesadores, redes de comunicación, discos, etc. Cuando la actividad se asocia con un procesador, se le conoce como tarea, y si se asocia con una red, se le conoce como mensaje. Tanto la tarea como el mensaje son realmente lo que se conocen como servidores de planificación, pero para simplificar, se llamará tarea o mensaje a la actividad que se ejecuta en un procesador o red respectivamente.

Los eventos que activan las actividades pueden tener impuestos unos *plazos* que deben cumplir, que típicamente hacen referencia al máximo tiempo del que dispone la actividad para finalizar desde que fue activada. De forma similar, el *tiempo de respuesta* de una actividad se define como la cantidad de tiempo que transcurre desde que la actividad fue activada, hasta que finalizó su ejecución.

No hay que confundir el *tiempo de respuesta* con el *tiempo de ejecución*. El tiempo de ejecución de una actividad hace referencia a la cantidad de tiempo que requiere para su ejecución cuando se ejecuta sola en el procesador, sin otras actividades con las que competir por el recurso. Por lo tanto el valor de este parámetro depende únicamente de la complejidad de la propia actividad, y de la velocidad del hardware (procesador, memoria, entrada/salida, etc).

De acuerdo con la rigidez de los plazos, los sistemas de tiempo real se pueden clasificar en estrictos o no-estrictos. Un plazo no-estricto no necesita ser cumplido siempre, sino que, por ejemplo, se puede estudiar su cumplimiento en promedio. En cambio, un plazo estricto exige siempre su cumplimiento. Si el sistema es incapaz de cumplir un plazo estricto, se considera que ha fallado, hecho que puede conllevar pérdidas económicas importantes, e incluso, de vidas humanas. En este trabajo sólo se tratarán los plazos estrictos.

Para poder prever el comportamiento temporal de un sistema, se hace necesario el disponer de herramientas de análisis teóricas. Como los plazos estrictos han de cumplirse siempre, las herramientas de análisis deben ser capaces de calcular los tiempos de respuesta de peor caso de todas las actividades que componen el sistema. Si dichos tiempos de respuesta de peor caso son menores

o iguales que sus respectivos plazos (en caso de tenerlos), se dice que el sistema es *planificable*. A éste proceso se le conoce como *análisis de planificabilidad*, y es de vital importancia en los sistemas de tiempo real.

1.1.1. Los Sistemas de tiempo real distribuidos

Durante los últimos años los sistemas multiprocesadores y distribuidos están cobrando suma importancia, debido a que el bajo coste de las redes de comunicación permite la interconexión de múltiples dispositivos y de sus controladores en un gran sistema.

La arquitectura típica de un sistema distribuido consta de varios procesadores interconectados por una o varias redes de comunicación. El software del sistema está formado por un conjunto de actividades concurrentes, que se ejecutan en algún procesador o red del sistema.

La comunicación entre tareas situadas en distintos procesadores se realiza mediante la transmisión de mensajes a través de las redes de comunicación. En la figura 1.1 se muestra un esquema sencillo de sistema distribuido.

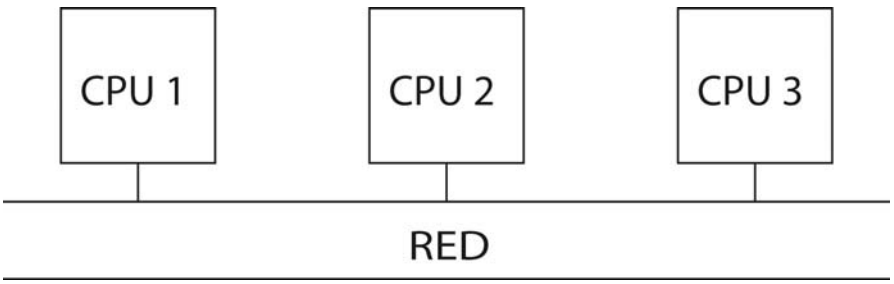


Figura 1.1: Ejemplo de Sistema Distribuido

1.1.2. Modelo Lineal de sistemas de tiempo real

Para realizar el estudio de un sistema de tiempo real dado, es conveniente disponer de un modelo con el que se pueda definir cada sistema de una manera exacta y sencilla, conteniendo las características propias de los sistemas de tiempo real, como los requerimientos temporales, o la imposición de un determinado

orden en la ejecución de las actividades de una respuesta. El *modelo lineal* cumple estos requisitos, y se describirá a continuación.

El modelo lineal es capaz de describir tanto sistemas monoprocesadores, como multiprocesadores y distribuidos. En el modelo se definen unos *eventos externos*, cuya respuesta es una serie de actividades, llamada *transacción*. Los *eventos externos* son producidos por alguna entidad externa, como por ejemplo, un reloj periódico, o algún estímulo producido por el usuario, con carácter aperiódico.

Las actividades contenidas dentro de la transacción se activan entre sí mediante *eventos internos*. Estos *eventos internos* pueden ser producidos por la finalización en la ejecución de alguna tarea (que activan mensajes u otras tareas), o por la finalización en la transmisión de un mensaje (que activan tareas). Suponemos que las tareas y los mensajes están ligados de forma estática a un procesador o una red respectivamente, ya que no existen herramientas que nos permitan predecir el comportamiento de una localización dinámica de las actividades [1].

Llamamos e_i al evento externo que activa a la actividad a_{i1} (primera actividad de la respuesta, que típicamente es una tarea). Llamamos e_{jk} al evento interno que produce la actividad a_{ij} para activar a la actividad a_{ik}. A la serie de actividades que se ejecutan como respuesta al evento externo e_i se la llama transacción i.

A su vez, se define el *Plazo Global* D_{ij}, como el plazo de tiempo asignado a la actividad a_{ij} con respecto a la llegada del evento externo e_i. El *Plazo local* d_{ij} es el plazo que posee la actividad a_{ij} con respecto a su propia activación (llegada del evento interno o externo que activó a a_{ij}).

Por último, se define el *Plazo de principio a fin* ED_{ij} como el plazo global de la última actividad de una respuesta con respecto a la llegada del evento externo e_i.

En la figura 1.2 se muestra un ejemplo sencillo del modelo lineal descrito, en el que se observa una secuencia lineal de actividades, con sus correspondientes plazos.

De manera similar a los plazos, se definen los tiempos de respuesta de peor caso. r_{ij} es el tiempo de respuesta de peor caso de la actividad a_{ij}, desde su propia activación. R_{ij} es el tiempo de respuesta global de peor caso de la actividad a_{ij}, con respecto a la llegada del evento externo e_i, o lo que es lo mismo :

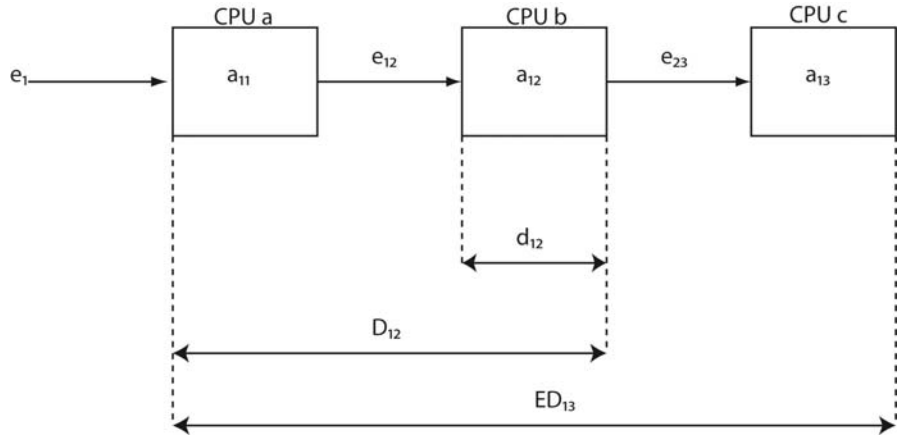


Figura 1.2: Modelo de Sistema Lineal

$$R_{ij} = \sum_{k=1}^{j} r_{ik} \tag{1.1}$$

Llamamos R_i al tiempo de respuesta global de peor caso de la última actividad de la respuesta al evento e_i.

Para sistemas monoprocesadores la transacción típicamente está compuesta por una única actividad. En la figura 1.3 se muestra un ejemplo sencillo de ejecución de una transacción periódica en un sistema monoprocesador, compuesta por una tarea, activada por un evento externo con periodo 4, y con plazo 4.

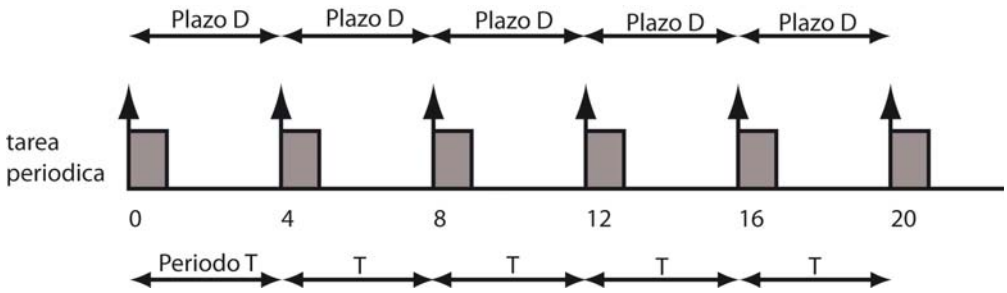


Figura 1.3: Ejemplo Tarea periodica y plazos

En la figura 1.4 se observa como se puede descomponer una típica transacción cliente-servidor definida como un modelo lineal de sistema distribuido. Un evento llega a la CPU 1, que provoca que una tarea se ejecute en ella. Cuando

termina su ejecución, genera un mensaje que se transmite a la CPU 2 mediante la RED. El mensaje, al llegar a su destino, activa una tarea en la CPU 2, y así sucesivamente.

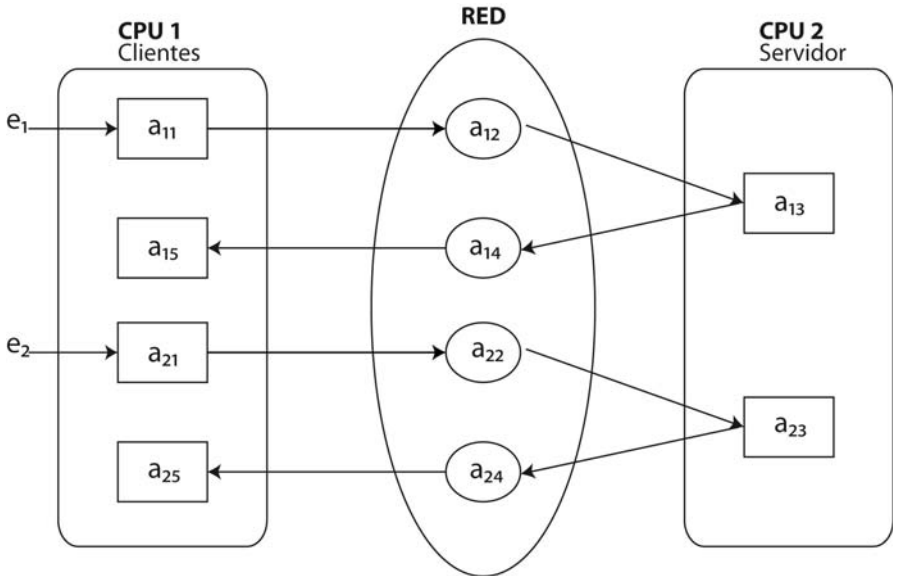


Figura 1.4: Cliente-Servidor como sistema distribuido

Cabe destacar que aunque el modelo lineal definido aquí nos sea suficiente para el desarrollo de este proyecto, presenta una serie de limitaciones, tales como que una actividad solo pueda ser activada por un único evento, y solamente puede generar un evento de salida; o que no se permite la sincronización entre tareas. Existen modelos generales que permiten definir sistemas distribuidos que no tienen éstas limitaciones [3], pero que sobrepasan las necesidades de este proyecto.

1.2. Políticas de Planificación para tiempo real

Cada recurso procesador del sistema va a tener que ejecutar una serie de actividades de forma concurrente. Es por ello necesario la utilización de un algoritmo o política de planificación que seleccione en cada momento qué actividad ejecutar de entre todas las actividades que han sido activadas y están esperando a ser ejecutadas.

Las políticas de planificación para sistemas de tiempo real deben cumplir los siguientes requisitos:

- Predecibilidad, en el sentido de que deben permitir analizar el comportamiento temporal del sistema, con el fin de estudiar la respuesta temporal de peor caso.
- Conseguir utilizaciones altas de los procesadores manteniendo la planificabilidad del sistema.
- En momentos de sobrecarga transitoria, asegurar la planificabilidad de las tareas críticas.

Una técnica de planificación tradicionalmente utilizada es la de los ejecutivos cíclicos [2]. En esta aproximación, a cada una de las tareas o actividades concurrentes que se desean ejecutar se les asigna una rodaja temporal durante la cual se pueden ejecutar. Estas rodajas temporales se ejecutan de forma periódica o cíclica en un orden preestablecido, e intercaladas adecuadamente para garantizar el cumplimiento de todos los requerimientos temporales. Cuando el sistema se hace complejo, esta aproximación se hace intratable, ya que cualquier modificación en el tiempo de ejecución de cada tarea, por pequeño que sea el cambio, requiere el rediseño de todo el sistema. Por este motivo, se definen otras políticas de planificación, basadas en prioridades.

En un sistema planificado por prioridades, cada actividad tiene asignada una prioridad, que no es más que un valor numérico con el que el algoritmo de planificación toma su decisión.

En el momento de la decisión, el algoritmo selecciona para la ejecución a la actividad con mayor prioridad de entre las actividades activas. El concepto de prioridad se puede implementar mediante una cola en la que se sitúan las actividades. Dependiendo de la prioridad asignada a cada actividad, se situará en un lugar u otro de la cola.

El planificador puede ser *no expulsor*. El momento de la decisión ocurre cuando las actividades terminan su ejecución. Cuando una actividad comienza su ejecución en el recurso procesador, no lo abandonará hasta haber finalizado, aunque haya actividades de mayor prioridad esperando. Esto puede provocar un retraso innecesario en las actividades de mayor prioridad.

Para solucionarlo existen los planificadores *expulsores*, en los que se asegura que en todo momento se está ejecutando la actividad con la mayor prioridad, de entre las actividades listas para su ejecución. Para lograrlo, el momento de la decisión ocurre cada vez que una actividad es activada, finaliza su ejecución, o a intervalos regulares gobernados por un reloj (*ticker*)

La prioridad puede ser asignada en tiempo de compilación (*prioridad fija*), o puede ser una *prioridad dinámica* que puede cambiar con el curso de la ejecución, como el caso de la planificación EDF (*Earliest Deadline First*).

1.3. Análisis de Planificabilidad para sistemas monoprocesadores

En un primer lugar se van a describir las herramientas de análisis de planificabilidad para sistemas monoprocesadores, ya que sientan las bases sobre las que se van a desarrollar las técnicas de análisis para los demás tipos de sistemas.

Las herramientas de análisis de planificabilidad para sistemas monoprocesadores nos permiten determinar los tiempos de respuesta de peor caso de las actividades que componen el sistema, a fin de comprobar que es capaz de cumplir todos sus plazos. Por lo tanto, son herramientas de suma importancia en los sistemas de tiempo real estricto. Nos centraremos en algoritmos basados en prioridades, en las que las actividades son independientes, expulsables, activadas mediante eventos externos periódicos, y las transacciones están formadas por una sola actividad.

Para comparar algoritmos de planificación existen diversas figuras de mérito. La bibliografía recoge principalmente dos:

- **Límite de Utilización**. Un algoritmo de planificación puede planificar un sistema de tareas periódicas en un procesador si la utilización total de las tareas es igual o menor que el *Límite de Utilizacion* del algoritmo utilizado. Esta es una condición suficiente pero no necesaria de planificabilidad. Si la utilización es superior al *límite de utilización*, hay que utilizar algún otro método para determinar la planificabilidad. En cualquier caso, nunca se podrá planificar un sistema con una carga superior a 1 (100 %, carga total).

- Un algoritmo de planificación es **óptimo** si siempre es capaz de planificar todo sistema que tiene la capacidad de ser planificado. Dicho de otra forma, si un algoritmo óptimo no es capaz de planificar un sistema, se puede afirmar que dicho sistema no puede ser planificado por ningún algoritmo.

1.3.1. Análisis de Planificabilidad para sistemas monoprocesadores con Prioridades Fijas

Los sistemas de tiempo real con planificador expulsor por prioridades fijas son ampliamente utilizados, ya que existen extensivos trabajos previos [4] que permiten el análisis de dichos sistemas, la respuesta temporal es fácil de entender, el comportamiento bajo momentos de sobrecarga se puede predecir, y son capaces de conseguir utilizaciones que típicamente están en torno al 70-95 %. Además, es soportado en una amplia variedad de lenguajes de programación (como Ada y Java), así como sistemas operativos (estándar POSIX).

Teoría RMA

La teoría RMA, desarrollada por Liu y Layland en 1973 [5], nos provee de un cuerpo teórico con el que analizar ciertos sistemas de tiempo real con planificación por prioridades fijas. Los sistemas sobre los que se puede aplicar deben cumplir los siguientes requisitos:

- Todas las tareas se ejecutan en un solo procesador.
- Todas las tareas son periódicas y no interactúan entre ellas.
- Los plazos de las tareas son iguales a sus periodos.
- Son perfectamente expulsables.
- No se usan interrupciones.
- No se tienen en cuenta los cambios de contexto.
- Las tareas no se autosuspenden.

Si se cumplen éstos requisitos, Liu y Layland demostraron que la asignación óptima de prioridades fijas es la que denominaron *rate monotonic*, en la que las prioridades se asignan de forma inversamente proporcional a los periodos de cada tarea, de forma que las tareas con menores periodos tienen mayor prioridad. Al algoritmo que asigna prioridades *rate monotonic* se le conoce como *Rate Monotonic Scheduling* (RMS).

Se define el *Límite de Utilización* para el algoritmo RM de la siguiente manera:

$$\frac{C_1}{T_1} + ... + \frac{C_n}{T_n} \leq U(n) = n \cdot (2^{\frac{1}{n}} - 1) \tag{1.2}$$

Donde :

C_i Es el tiempo de ejecución de peor caso de la tarea i.

T_i Es el periodo de la tarea i

$U(n)$ Límite de utilización del procesador para n tareas, por debajo del cual se puede asegurar que el sistema va a ser planificable.

Para introducir el segundo test que nos permitirá calcular los tiempos de respuesta de peor caso de cada tarea, antes hay que definir dos conceptos:

Instante Crítico. El tiempo de respuesta de peor caso de cualquier tarea se obtiene cuando todas las tareas del sistema son activadas a la vez. En la figura 1.5 se puede observar un ejemplo de instante crítico. Se llama *periodo de ocupación* al tiempo transcurrido desde el instante crítico, hasta que la tarea de menor prioridad ha finalizado su primera ejecución.

Chequeo del primer plazo. El tiempo de respuesta de peor caso de una tarea se obtiene después de un Instante Crítico. En las activaciones posteriores, el tiempo de respuesta será igual o menor. Por lo tanto, si se cumple el plazo en la primera activación tras el instante crítico, se puede asegurar que se cumplirá siempre, o dicho de otra manera, se podrá asegurar la planificabilidad del sistema.

Estos conceptos dan lugar al *test de tiempos de ejecución*, o *test exacto*, con el que se puede obtener el tiempo de respuesta de peor caso de un conjunto de

Tarea	C	T
1	1	4
2	2	6
3	2	10

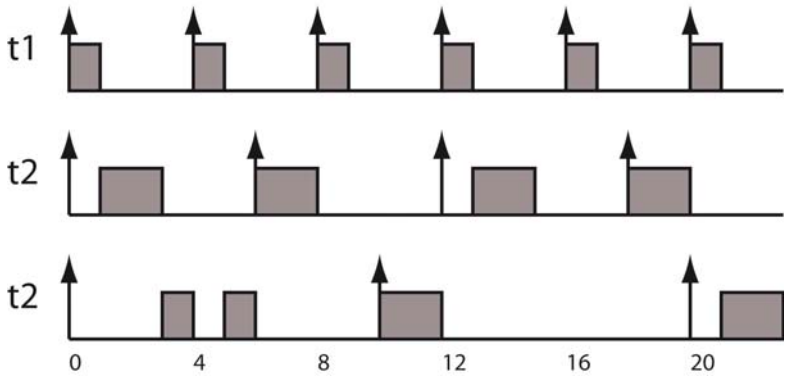


Figura 1.5: Instánte crítico

tareas planificadas con prioridades fijas, bajo las restricciones descritas en esta sección.

Los tiempos de respuesta de cada tarea se obtienen mediante una ecuación iterativa que va calculando el tiempo de ejecución necesario para finalizar todo el trabajo activado hasta el instante de la iteración anterior. La iteración termina cuando dos pasos consecutivos alcanzan el mismo resultado, lo cual indica que no queda más trabajo por realizar a ese nivel de prioridad. La ecuación se puede expresar de la forma siguiente [6] [7] [4]:

$$r_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{r_i^n}{T_j} \right\rceil C_j \tag{1.3}$$

Donde :

r_i. Es el tiempo de respuesta de peor caso de la tarea i. Cuando su valor en dos iteraciones seguidas coincide, las iteraciones finalizan. El valor de arranque es $r_i^0 = C_i$.

$hp(i)$. Es el conjunto de tareas que tienen mayor prioridad que i, o lo que es lo mismo, que pueden expulsar a i.

C_i. Es el tiempo de ejecución de peor caso para la tarea i.

T_j. Es el periodo de la tarea j.

En la ecuación 1.3 se puede observar que en el tiempo de respuesta de una tarea influye su propio tiempo de ejecución C_i, y la interferencia de tareas de mayor prioridad ($\left\lceil \frac{r_i^n}{T_j} \right\rceil C_j$).

Extensiones a la teoría RMA

La teoría RMA es válida en un grupo reducido de sistemas, es por ello interesante extender la teoría para poder abarcar un número de sistemas mayor, y más representativos de la realidad.

Una primera extensión va a ser la incorporación del efecto del cambio de contexto. Para lograrlo, al tiempo de ejecución de una tarea, C_i, se le va a añadir una cantidad $2 * C_s$ que modela el tiempo requerido para el cambio de contexto.

Es común en los sistemas de tiempo real que haya tareas con plazos anteriores al periodo. En estos casos, la asignación *Rate Monotonic* ya no es óptima. Leung y Leyland [8] probaron que la asignación óptima en estos casos es la llamada *Deadline Monotonic*, en la que las prioridades se asignan de acuerdo a los plazos. Una tarea tendrá mayor prioridad si posee un plazo más corto. En estos casos con plazos menores que el periodo, el *test exacto* definido para una asignación RM sigue siendo válido. De hecho, es válido para cualquier asignación de prioridades fijas.

El test del límite de utilización debe ser modificado. Para el análisis de una tarea τ_i, las tareas que la pueden expulsar (que poseen mayor prioridad) se dividen en 2 grupos

H_1 Tareas con periodos mayores a D_i. O lo que es lo mismo, que van a expulsar a τ_i una sola vez durante el tiempo que dura el plazo D_i.

H_n Tareas con periodos menores a D_i.

Para τ_i se define su utilización efectiva como:

$$f_i = \sum_{j \in H_n} \frac{C_j}{T_j} + \sum_{k \in H_1} \frac{C_k}{T_i} + \frac{C_i}{T_i} \tag{1.4}$$

El test del límite de utilización debe aplicarse a cada tarea individualmente. El límite de utilización para la tarea τ_i se calcula de la siguiente manera:

$$\begin{aligned} n &= num(H_n) + 1 \\ \delta_i &= \frac{D_i}{T_i} \leq 1 \\ U(n, \delta_i) &= \begin{cases} n((2\delta_i)^{1/n} - 1) + 1 - \delta_i & , 0,5 \leq \delta_i \leq 1 \\ \delta_i & , 0 \leq \delta_i \leq 0,5 \end{cases} \end{aligned}$$

Existen una multitud adicional de fenómenos que pueden ocurrir en un sistema real que deben ser tenidos en cuenta en las herramientas de análisis, pero que no van a tener especial relevancia en el desarrollo de este proyecto. A continuación se citan los más importantes:

- **Inversión de prioridad**, que es el efecto producido cuando una tarea es retrasada por otra de menor prioridad. Puede ocurrir cuando hay regiones de código no expulsables, colas FIFO, o las tareas compartan resursos. El modelado de la inversión de prioridad de lleva a cabo añadiendo *tiempos de bloqueo* en las ecuaciones de análisis.
- **Eventos aperiódicos**. Puede ocurrir que los eventos que activan las tareas no tengan naturaleza periódica. Si la cantidad de eventos que pueden llegar en un intervalo de tiempo dado es ilimitada, sólo se pueden especificar plazos no estrictos.
- **Plazos superiores al periodo**. El concepto de instante crítico sigue siendo válido, pero no basta con comprobar el primer plazo. La cantidad de plazos a comprobar viene determinado por el *periodo de ocupación de peor caso*. Ni la asignación RM ni DM de prioridades es óptima en este caso. Audsley [9] propuso un algoritmo que es óptimo en la asignación de prioridades para plazos arbitrarios.

1.3.2. Análisis de Planificabilidad para sistemas monoprocesadores con prioridades dinámicas EDF

Hasta ahora se han tratado los sistemas planificados por prioridades fijas, que nos permitían, de una forma relativamente sencilla, obtener altas utilizaciones en los procesadores manteniendo la planificabilidad del sistema. Con una planificación por prioridades dinámicas es posible utilizar los recursos procesadores de una forma más eficiente aún.

Existen varios algoritmos de planificación con prioridades dinámicas, pero durante este trabajo nos centraremos en el algoritmo EDF. En la planificación EDF se tiene en cuenta:

- $t_{i,j}$, que hace referencia al instante de tiempo **absoluto** en el que la tarea i tiene su j-ésima activación.
- $d_{i,j}$, plazo absoluto de la j-esima activación de la tarea i, o lo que es lo mismo, $t_{i,j} + D_i$.

En cada activación de la tarea τ_i se calcula su prioridad de forma inversamente proporcional a su plazo absoluto $d_{i,j}$, y esta prioridad de mantiene constante hasta la siguiente activación de τ_i. En la figura 1.6 se muestra un ejemplo sencillo.

Observando la figura 1.6, se puede ver que es la tarea 1 la que empieza a ejecutarse en primer lugar, ya que su plazo global es el más pequeño (15). Una vez que termina, la tarea 2 se ejecuta, ya que su plazo global (20) está más cercano que el de la tarea 3 (25). En el instante 15 ocurre la segunda activación de la tarea 1. El planificador observa que el plazo global de la tarea 1 es 30, y sólo la tarea 3 está actualmente activa, con un plazo global de 25, por lo que la tarea con más prioridad en ese momento es la 3, continuando su ejecución hasta terminar.

Si se hubiera planificado el ejemplo anterior con prioridades fijas DM, se puede comprobar fácilmente que la tarea 3 hubiera perdido su primer plazo, y por lo tanto, el sistema ya no sería planificable.

Cuando se posee un sistema compuesto por tareas periódicas e independientes ejecutándose sobre un único procesador, con un planificador expulsor, y plazos iguales a los periodos, el algoritmo EDF es óptimo, o dicho de otra manera,

Tarea	C	T	D
1	5	15	15
2	5	20	20
3	6	25	25

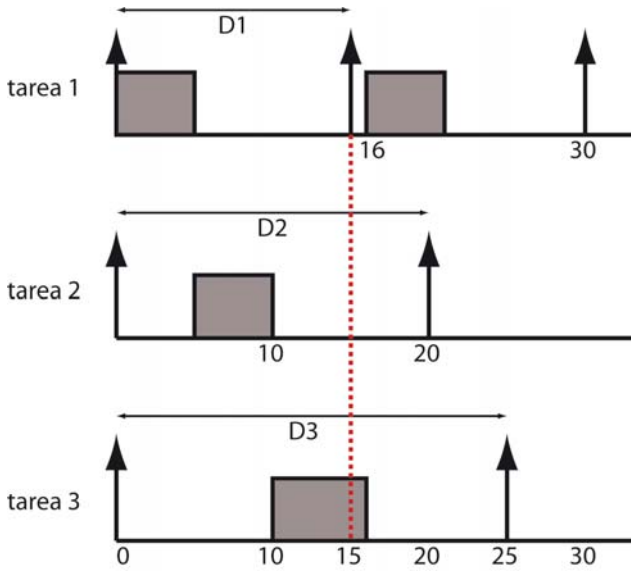


Figura 1.6: Ejemplo EDF

bajo esas limitaciones, si el algoritmo EDF no es capaz de planificar un sistema, no hay otro algoritmo (de prioridades fijas o dinámicas) que sea capaz de hacerlo.

Para determinar si un sistema dado de n tareas periódicas cumplirá todos sus plazos si es planificado con el algoritmo EDF, se puede comprobar la siguiente ecuación, que se corresponde con el *test del límite de utilización* para el algoritmo EDF, que además es exacto.

$$\forall k, \sum_{k=1}^{n} \frac{C_k}{T_k} \leq 1 \tag{1.5}$$

Por lo tanto, si el sistema posee una utilización menor del 100 %, será planificable bajo EDF. Si se diese el caso de que alguna tarea tuviera un plazo menor que su periodo, se define el siguiente test pesimista:

$$\forall k,\ \sum_{k=1}^{n} \frac{C_k}{\min(D_k, T_k)} \leq 1 \tag{1.6}$$

Si el test 1.6 se cumple, se puede asegurar la planificabilidad. Si no se cumple, y la utilización definida en 1.5 se satisface, hay que utilizar un test exacto para determinar la planificabilidad.

Con EDF, en efecto, se pueden conseguir mayores utilizaciones en los procesadores con respecto a los algoritmos con prioridades fijas, manteniendo la planificabilidad, y es por ello por lo que resulta interesante su uso. Desgraciadamente, posee una serie de desventajas que hay que considerar, siendo las principales:

- Mayor complejidad de los sistemas, debido al cálculo de la prioridad en cada activación de las tareas.
- En momentos de sobrecarga transitoria, en los que se puede superar el 100 % de utilización, el sistema se vuelve impredecible. En un sistema con prioridades fijas podemos predecir que las tareas de mayor prioridad van a poder ejecutarse dentro de sus respectivos plazos. En el caso EDF, en cambio, no se puede predecir, y hay que utilizar técnicas adicionales para asegurar la planificabilidad de las tareas mas "prioritarias", aumentando así aún más la complejidad del sistema.

Al igual que en la sección 1.3.1, en la que se trataron sistemas con prioridades fijas, en EDF también existen circunstancias que deben ser tratadas adicionalmente, tales como eventos aperiódicos, compartición de recursos, etc, pero que no están en el ámbito de los objetivos de este trabajo.

1.4. Análisis de planificabilidad para Sistemas Distribuidos

De manera similar a lo que ocurría para sistemas monoprocesadores, las herramientas de análisis para sistemas multiprocesadores nos permiten calcular los tiempos de respuesta de peor caso de cada actividad, y de toda la transacción

en su conjunto, para así poder determinar si se cumplen los plazos establecidos según el modelo lineal. Analizaremos sistemas definidos según el modelo lineal descrito en 1.1.2.

1.4.1. Consideraciones sobre los sistemas de tiempo real distribuidos

La mayoría de las redes de comunicación estándares no se adaptan bien a las comunicaciones de tiempo real, y no soportan una planificación de los mensajes basada en prioridades. Sin embargo, existen algunas redes de comunicación estándares que se han utilizado para llevar a cabo comunicaciones de tiempo real estricto, tales como el token-bus [10], FDDI [11], el bus CAN [12], etc.

Trabajos adicionales [3] desarrollan una herramienta de software basada en la interfaz de colas de mensajes definida en el POSIX.1b [13], que permite la utilización de una planificación de los mensajes por prioridades fijas sobre subsistemas de comunicaciones estándares como el bus VME, o las líneas RS-232/RS-485. Esto permite la utilización de las mismas técnicas RMA sobre las redes. Por lo tanto, se tratarán los mensajes transmitidos en las redes de la misma manera que se trata a las tareas que se ejecutan en un procesador, en donde el tiempo de transmisión del mensaje se modela como el tiempo de ejecución de una tarea en un procesador. Sin embargo, hay que tener en cuenta un pequeño término de bloqueo debido a que los mensajes se dividen en paquetes indivisibles (no expulsables en términos de procesador).

Para el caso de planificación EDF, se hará la misma similitud red-procesador, pero, como se comentará en posteriores capítulos, no existen en la actualidad redes de comunicaciones que puedan utilizar una planificación EDF, siendo éste un campo totalmente abierto a la investigación.

1.4.2. El problema del Jitter

La primera actividad de cada respuesta, al ser activada mediante un evento externo, se puede considerar que tiene una activación periódica. Sin embargo, el instante de activación de las actividades sucesivas de la transacción depende del instante de finalización de la actividad precedente, o lo que es lo mismo, del

tiempo de respuesta global de la actividad precedente, y éste no es típicamente un valor constante.

Por lo tanto, todas las actividades de una transacción dada, salvo la primera, pueden tener activaciones no perfectamente periódicas, también llamadas *activaciones retrasadas*, o *jitter*.

El efecto del *jitter* en sistemas distribuidos influye de manera negativa en la planificabilidad, aumentando los tiempos de respuesta de peor caso, y por lo tanto, disminuyendo las posibilidades de utilización de los recursos. Se puede observar el efecto negativo del *jitter* en el ejemplo sencillo descrito en la figura 1.7.

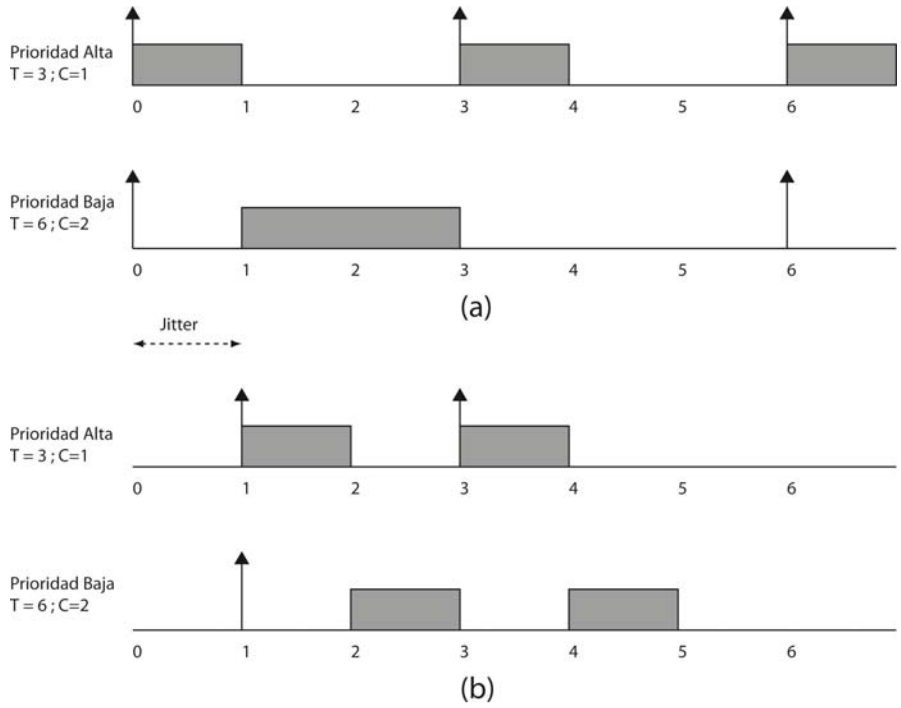


Figura 1.7: (a) Activacion periorica (b) Activación con Jitter

En la figura 1.7 se presentan dos tareas periódicas ejecutándose en un único procesador. Una de las tareas tiene menor prioridad que la otra. Cuando se activan de forma periódica (sin *jitter*), observamos que la tarea de prioridad inferior tiene un tiempo de respuesta igual a 3. Cuando la tarea de prioridad superior retrasa su activación, provoca que la otra tarea sea expulsada una vez, y su tiempo de respuesta aumenta hasta 4.

Definimos el *jitter* J_{ik}, como el *jitter* de peor caso para el evento externo e_i y la actividad a_{ik} . J_{ik} es igual a la variación del tiempo de respuesta de la actividad previa. Como el tiempo de ejecución de una actividad puede ser 0, asumimos que la variación del tiempo de respuesta es igual al tiempo de respuesta global de peor caso de la actividad previa (R_{ik-1}). Si se conociese el tiempo de ejecución de mejor caso de una actividad, el *jitter* se podría generalizar con la siguiente ecuación:

$$J_{ik} = R_{ik-1} - \sum_{l \in A_{ik-1}} b_{il} \tag{1.7}$$

Dónde b_{il} es el tiempo de ejecución de mejor caso de la actividad a_{il}, A_{ik-1} es el grupo de actividades de la respuesta al evento e_i que preceden a la actividad a_{ik}.

Como se puede observar en la ecuación 1.7, el cálculo del *jitter* para las actividades de un recurso dependen de los tiempos de respuesta de otras actividades que pueden estar en otros recursos. A su vez, el cálculo de los tiempos de respuesta dependen del *jitter*.

La solución a este problema de interdependencia es aportada por Tindell [14], y consiste en llevar a cabo el análisis de peor caso de una manera iterativa hasta que se alcance una solución estable para el *jitter* y los tiempos de respuesta. A este tipo de analisis se le llama *Holistic*, y se presentará en la sección 1.4.3.

1.4.3. Análisis para sistemas con prioridades fijas

En la sección 1.3.1 de éste trabajo se introdujo la teoría RMA como base para poder analizar un grupo reducido de sistemas de tiempo real monoprocesadores con prioridades fijas. Mediante extensiones se podían analizar un grupo de sistemas más amplio, pero siempre en el campo de los sistemas monoprocesadores. En esta sección se van a describir las herramientas de análisis para sistemas distribuidos con prioridades fijas.

Holistic

El análisis de peor caso para un sistema distribuido es un campo activo en la investigación sobre sistemas de tiempo real. Una primera aproximación es aquella que considera a cada recurso del sistema (procesadores y redes) como independientes. Esta primera aproximación es inherentemente pesimista, ya que no tiene en cuenta que las relaciones temporales entre diferentes respuestas son interdependientes, pudiendo así construir una situación que es peor que el peor caso real en un sistema distribuido.

Sin embargo, al ser pesimista, éste método nos permite garantizar la planificabilidad de un sistema si los tiempos de respuesta de peor caso calculados son menores o iguales que los plazos.

En el análisis *Holistic* se aplican las técnicas RMA sobre cada recurso, ya que se han considerado independientes. La interdependencia *jitter*-tiempo de respuesta la soluciona Tindell [14] aplicando el cálculo del *jitter* y los tiempos de respuesta de forma iterativa. Las características de monotonicidad del tiempo de respuesta y el *jitter* llevan a las iteraciones a converger a una solución correcta.

Se aplican las técnicas RMA descritas en la introducción sobre cada procesador/red, añadiendo las extensiones para soportar plazos cualesquiera, considerando además los términos de bloqueo debidos a las secciones críticas B_j:

$$W_{ij}^{n+1} = (q+1)C_j + B_j + \sum_{\forall k \in hp(j)} \left\lceil \frac{J_{ik} + w_{ij}^n(q)}{T_k} \right\rceil C_k \tag{1.8}$$

$w_{ij}(q)$ Tiempo de finalización de peor caso de la actividad a_{ij}, al que converge la ecuación cuando $W_{ij}^{n+1}(q) = W_{ij}^n(q)$, en la q-ésima activación. $W_{ij}^0(q) = (q+1)C_j$

$hp(j)$ Conjunto de actividades que pueden expulsar a a_{ij}

B_j Término de bloqueo.

J_{ik} *Jitter* de peor caso de la actividad a_{ik}, definido según la ecuación 1.7

La ecuación 1.8 se aplica se forma sucesiva para valores de $q = [0, 1, 2, ...]$. La

iteración termina cuando se cumple la siguiente desigualdad:

$$w_{ij}(q) \leq (q+1)T_j \tag{1.9}$$

Los tiempos de respuesta locales de peor caso se calculan de la siguiente manera:

$$r_{ij} = \max_{q=0,1,2,\ldots} (w_{ij}(q) - qT_j) \tag{1.10}$$

En una primera iteración, el término de *jitter* J_{ij} se considera 0, y se procede al cálculo de los tiempos de respuesta locales de cada actividad. Con los valores de r_{ij}, se calculan de nuevo los *jitter*, y así de forma sucesiva. Cuando los valores de los tiempo de respuesta y de *jitter* converjan a un valor determinado, se considerará como la solución del análisis.

Análisis mediante Offsets

El análisis *Holistic* es por naturaleza pesimista, en el sentido de que puede obtener resultados de situaciones peores que el peor caso real posible. Dicho de otra manera, nos da una cota superior del tiempo de respuesta de peor caso de las actividades de un sistema.

Si una técnica exacta, o menos pesimista al menos, pudiera ser definida, se podría utilizar la capacidad de computación de estos sistemas de tiempo real de una manera más eficiente. Tindell [15] desarrolló una técnica que permite calcular de forma exacta los tiempos de respuesta de un sistema compuesto por tareas con *offsets* estáticos.

El *offset* de una actividad se define como la cantidad de tiempo que transcurre desde que el evento externo activa la transacción, hasta que la propia actividad es activada. En la figura 1.8 se puede observar gráficamente su significado, para una transacción dada.

Las flechas superiores indican la llegada del evento externo que inicia la transacción de una secuencia de 3 actividades. Adicionalmente se puede incluir el efecto del *Jitter*, modelando la activación de la tarea como si ocurriera en algún momento entre el instante $t_0 + \Phi_{ij}$ y $t_0 + \Phi_{ij} + J_{ij}$, siendo t_0 el instante de activación de la transacción, y J_{ij} el *Jitter* de peor caso de la tarea a_{ij}.

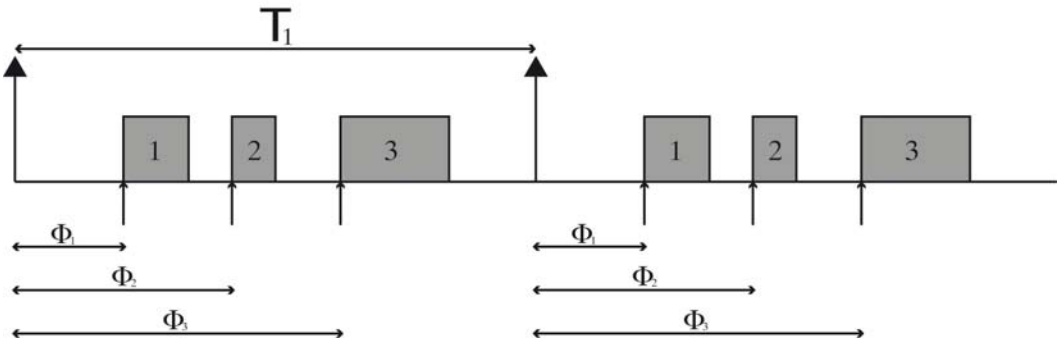


Figura 1.8: Definición de offsets estáticos en una transacción

El análisis mediante *offsets* consiste en estudiar la contribución de cada tarea a_{ij} sobre la respuesta de peor caso de otras tareas de menor prioridad. Para ello se dividen sus activaciones en tres posibles grupos:

- **Set 0**. Activaciones que ocurren antes del instante crítico, y que cuyo *Jitter* máximo no puede retrasarlas hasta el periodo de ocupación. No se tienen en cuenta en el análisis del periodo de ocupación.
- **Set 1**. Activaciones que ocurren antes o en el instante crítico, y cuyo *Jitter* puede retrasarlas hasta el instante crítico.
- **Set 2**. Activaciones que ocurren después del instante crítico

Una vez divididas las activaciones de la tarea a_{ij} en algún grupo, se calcula el *Jitter* que provocaría el peor caso sobre las tareas de menor prioridad. Dicho peor caso ocurre cuando las activaciones del Set 1 se ven retrasadas por una cantidad de *Jitter* que las hacen coincidir todas con el instante crítico, y las activaciones del Set 2 no experimentan ningún retraso. Adicionalmente, en el peor caso, la primera activación del Set 1 se ve retrasada por el *Jitter* máximo J_{ij}.

Mediante estas consideraciones, Tindell define una serie de ecuaciones [15] con las que se pueden calcular de forma exacta los tiempos de respuesta cuando se considera que los *offsets* son constantes. El problema es que el análisis se vuelve computacionalmente intratable, ya que la tarea que genera la contribución de peor caso sobre una transacción dada es desconocida, y por lo tanto, hay que comprobar todas las posibles combinaciones de las tareas.

En el mismo trabajo [15] , Tindell propone una aproximación al análisis exacto, que produce resultados ligeramente más pesimistas, pero en el que el número

de casos a tratar ya no crece exponencialmente con el número de tareas. La aproximación consiste en que, para cada transacción, se define una función que es el máximo de todas las contribuciones de peor caso considerando que cada una de las tareas de la transacción inicia el instante crítico.

Análisis mediante offsets dinámicos

El análisis mediante offsets fue extendido por J.C.Palencia y M.G.Harbour [16], añadiendo la capacidad de los offsets de ser dinámicos, además de eliminar la restricción de que los offsets deban ser menores que los periodos.

El uso de offsets dinámicos es necesario para poder analizar sistemas distribuidos. En estos sistemas, se especifica que el offset varía entre un valor máximo y mínimo, $\Phi_{ij} \in [\Phi_{ij,min}, \Phi_{ij,max}]$. Tal y como se describe en [16], se puede modelar el caso en el que los offsets varíen como un caso especial de offsets estáticos, definiendo un offset estático equivalente Φ'_{ij}, y un Jitter asociado J'_{ij} de la siguiente manera:

$$\begin{aligned} \Phi'_{ij} &= \Phi_{ij,min} \\ J'_{ij} &= J_{ij} + \Phi_{ij,max} - \Phi_{ij,min} \end{aligned}$$

Para el caso de sistemas distribuidos, se comprueba que la equivalencia es la siguiente:

$$\begin{aligned} \Phi'_{ij} &= R^b_{ij-1} \\ J'_{ij} &= R_{ij-1} - R^b_{ij-1} \end{aligned}$$

El problema que aparece ahora es que los offsets dependen de los tiempos de respuesta, y los tiempos de respuesta de los offsets. La solución es aplicar un análisis iterativo, de forma similar a como se resolvía la dependencia Jitter-Tiempos de Respuesta en el análisis Holistic. Se comienza con unos valores iniciales de $J_{ij} = 0$, y $\Phi_{ij} = R^b_{ij-1}$ para cada tarea. Las características de monotonicidad del tiempo de respuesta con el *jitter* nos aseguran la convergencia del algoritmo.

1.4.4. Análisis para sistemas EDF

Existen diversas técnicas de análisis para sistemas distribuidos EDF. Spuri [17] adaptó la técnica *Holistic* de prioridades fijas para el análisis de sistemas EDF. Al igual que en el caso de prioridades fijas, obtiene soluciones pesimistas.

Palencia [18] adaptó el análisis mediante offsets a los sistemas EDF, con el que se obtienen resultados menos pesimistas que con el análisis Holistic. En este análisis, las actividades son planificadas según sus respectivos plazos globales.

Palencia, en un trabajo aún sin publicar, ha adaptado el análisis mediante offsets para sistemas distribuidos EDF descrito en [18], para el caso en el que las actividades se planifican según sus plazos locales. Éste es el análisis que se va a utilizar en este proyecto cuando haya que analizar sistemas distribuidos planificados mediante EDF.

1.5. Técnicas de Asignación de Parámetros de Planificación

Hay dos problemas fundamentales que presentan los sistemas de tiempo real. En primer lugar, la capacidad de determinar la planificabilidad del sistema utilizando algún test de límite de utilización, o calculando los tiempos de respuesta de peor caso de las actividades. El segundo problema es el de encontrar una asignación de prioridades en las actividades que consiga hacer el sistema planificable.

Para algoritmos con prioridades fijas, el objetivo es el de que cada actividad tenga asignado un valor numérico que represente su prioridad, que utilizará el planificador a la hora de tomar su decisión. En la sección 1.3.1 se presentaron algunas técnicas de asignación para sistemas monoprocesadores, como RM y DM.

- *Rate Monotonic*. Se asignan las prioridades de forma inversamente proporcional al periodo. Es óptimo para el caso de que los plazos sean iguales a los periodos.

- *Deadline Monotonic*. Las prioridades se asignan de forma inversamente proporcional al plazo. Es óptimo para el caso de prioridades menores o iguales al periodo.
- *Asignación Óptima de Prioridades*. Audsley [9] propuso un algoritmo de asignación de prioridades que es óptimo para plazos cualesquiera.

En el caso de la utilización de prioridades dinámicas, se introdujo el algoritmo EDF, en el que las prioridades se iban recalculando según el plazo absoluto de cada activación. Por lo tanto, cada tarea debe tener asignado un plazo, que debe conocer el planificador. A éstos plazos se les conoce como *plazos de planificación*, SD_{ij}. El objetivo es, por lo tanto, la asignación de un *plazo de planificación* a cada actividad del sistema. Cuando la respuesta está compuesta de únicamente una actividad, el cálculo del *plazo de planificación* es trivial, no es más que $SD_i = D_i$.

Resulta evidente que las técnicas de asignación definidas hasta ahora no son aplicables a una transacción, y se hace necesaria la definición de nuevos algoritmos de asignación. Para el caso de prioridades fijas, Tindell, Burns y Wellings [14] propusieron la técnica del *templado simulado*, que calcula las prioridades mediante un método de propósito general, que da lugar a un algoritmo lento (referido al tiempo que tarda en alcanzar la solución).

Otra técnica de cálculo y optimización de prioridades fijas es el algoritmo heurístico HOPA [20], basado en el conocimiento de los tiempos de respuesta del sistema, y que comparativamente es más rápido que el *templado simulado*, además de ser capaz de ofrecer mejores soluciones. Este algoritmo se tratará con profundidad en la sección 2.1.

Para sistemas EDF, Liu [1] define algoritmos que reparten el plazo de principio a fin ED_{ij} de la transacción entre sus actividades, de forma que asigna a cada tarea un *plazo de planificación*. Se describirán en el capítulo 2.

1.6. Objetivos del Proyecto

Se han introducido los conceptos básicos que rigen los sistemas de tiempo real, entre ellos, los algoritmos de planificación y las técnicas de análisis. Se ha

presentado además el problema de la asignación de parámetros de planificación, introduciendo algunas técnicas existentes. Los objetivos que se pretenden conseguir en este proyecto son los siguientes:

- Estudio de las técnicas de asignación de parámetros de planificación existentes en la actualidad para sistemas distribuidos. Nos centraremos en dos técnicas de asignación de plazos de planificación para sistemas EDF descritas por Liu [1], y el algoritmo heurístico de asignación de prioridades fijas, HOPA [20].
- Una vez estudiadas las técnicas actuales, se propondrá un algoritmo heurístico de asignación de plazos de planificación para sistemas distribuidos EDF que estará basado en el algoritmo HOPA (que obtiene buenos resultados para prioridades fijas).
- Se llevará a cabo una implementación software de la técnica propuesta y de las dos técnicas de asignación de plazos de planificación estudiadas.
- Se desarrollará un generador automático de ejemplos que nos permitirá aplicar las técnicas de asignación de parámetros de planificación sobre un elevado número de sistemas de tiempo real distribuidos.
- Para concluir, se hará un estudio comparativo entre las tres técnicas de asignación de plazos de planificación y el algoritmo HOPA, con idea de verificar la bondad del algoritmo propuesto.

1.7. Organización del Proyecto

El proyecto se organiza de forma similar a como se han presentado los objetivos de este proyecto.

- En el capítulo 2 se va a llevar a cabo el estudio de las técnicas existentes en la actualidad de asignación de parámetros de planificación para sistemas distribuidos.
- En el capítulo 3 se propondrá un algoritmo heurístico de asignación de plazos de planificación para sistemas distribuidos con planificación EDF.

- En el capítulo 4 se llevará a cabo una implementación software de las técnicas.
- En el capítulo 5 se desarrollará un generador automático de ejemplos.
- En el capítulo 6 se realizará un estudio comparativo entre las técnicas de asignación de parámetros de planificación.
- En el capítulo 7 se mostrarán las conclusiones obtenidas tras la realización de este proyecto, además de aventurar algunas líneas futuras posibles.

Capítulo 2

Asignación de parámetros de planificación en Sistemas de Tiempo Real Distribuidos

El problema de la asignación de parámetros de planificación ya se presentó en la introducción de este trabajo. Para sistemas monoprocesadores, en donde las respuestas suelen estar compuestas de una sola tarea, existen varios algoritmos que eran capaces de asignar de forma óptima (bajo ciertas circunstancias) prioridades fijas a las tareas. Estos algoritmos eran *Rate Monotonic* para el caso de plazos iguales a los periodos; *Deadline Monotonic*, como una generalización de RM cuando los plazos eran menores que los periodos; y la asignación óptima de prioridades propuesta por Audsley, para plazos arbitrarios.

En los sistemas multiprocesadores/distribuidos hemos visto que las respuestas están formadas por varias tareas que se ejecutan de manera sucesiva, llamadas transacciones, en las que los plazos se dan típicamente con respecto a toda la transacción, plazos de principio a fin ED_{ij}, y existen efectos que provocan que las tareas no sean periódicas (*jitter*), aunque la transacción lo sea.

Por estos motivos se hace necesario definir nuevas técnicas de asignación de prioridades para sistemas multiprocesador/distribuidos. Nos centraremos en sistemas descritos según el modelo lineal definido en la sección 1.1.2, con tareas y mensajes asignados de manera estática a un determinado procesador y red respectivamente, en los que las actividades están planificadas por algún algoritmo de planificación por prioridades. A pesar de esta restricción, la búsqueda de una

solución exacta para la asignación de prioridades es un problema NP-completo.

En primer lugar expondremos el problema para sistemas distribuidos planificados con prioridades fijas, presentando el algoritmo de asignación de prioridades fijas HOPA. Más adelante se tratará el problema de la asignación de plazos de planificación para sistemas distribuidos EDF, presentando dos algoritmos sencillos descritos por Liu [1]. En el capítulo 3 se propondrá un algoritmo heurístico de asignación de plazos de planificación.

Para poder comparar los diferentes algoritmos de asignación de prioridades y plazos de planificación que se van a describir y proponer en este trabajo, se necesita algún criterio con el que comparar los resultados, y poder determinar la bondad de cada uno de ellos. En este trabajo se utilizará el *Índice de Planificabilidad*, en el que se tiene en cuenta la "distancia" entre el plazo de principio a fin ED_{ij} y el tiempo de respuesta global de cada transacción.

$$\text{Índice} = \begin{cases} \sum_g \min(0,\, ED_{ij} - R_i) & \text{Cuando no planificable} \\ \sum_g \max(0,\, ED_{ij} - R_i) & \text{Cuando planificable} \end{cases} \tag{2.1}$$

Dónde ED_{ij} es el plazo de principio a fin de la transacción i, R_i el tiempo de respuesta global de la transacción i, y g hace referencia al grupo de transacciones que forman el sistema de tiempo real.

Por lo tanto, valores altos del Índice indican que existe un amplio margen entre los plazos y el tiempo de respuesta. Un valor negativo del Índice señalaría la no planificabilidad del sistema. El uso del *índice de planificabilidad* nos da una idea aproximada de la "holgura" que tiene el sistema con una asignación de parámetros de planificación dada, pero no es una métrica exacta. Puede darse el caso de que exista una asignación de parámetros de planificación con el que se obtenga un *índice de planificabilidad* elevado, pero en el que un aumento de el tiempo de ejecución en alguna actividad, por pequeño que sea este aumento, provoca la no planificabilidad del sistema. Una métrica exacta sería la utilización del *slack*, o sensibilidad del sistema.

Con el cálculo del *slack* obtenemos de forma precisa cuánto tiempo de ejecución se puede añadir a las actividades del sistema para seguir manteniendo la planificabilidad, o cuanto tiempo de ejecución hay que restar para conseguir la planificabilidad en el caso de que el sistema no fuera planificable. El problema del cálculo del *slack* es que hace un uso continuado de las herramientas de análisis para el cálculo de los tiempos de respuesta, por lo que puede ser un pro-

ceso computacionalmente lento y costoso, razón por la cual no se utiliza en este trabajo.

2.1. Algoritmo Heurístico de Asignación de Prioridades fijas

Tindell,Burns y Weilings [14] afrontaron el problema de la asignación de prioridades fijas en sistemas distribuidos mediante la técnica del *templado simulado*. La técnica del templado simulado [19] es una técnica de optimización de propósito general basada en la relación existente entre la mecánica estadística (comportamiento de sistemas con múltiples grados de libertad en equilibrio térmico a una temperatura finita), y la optimización combinatoria o multivariable (encontrar el mínimo de una función dada que depende de muchos parámetros). Al ser de propósito general, no tiene en cuenta ninguna característica en concreto de los sistemas de tiempo real, y por ello se obtienen los resultados de una forma relativamente lenta.

Con el objetivo de mejorar la velocidad y calidad de los resultados obtenidos con la técnica del templado simulado, J.J. Gutiérrez y M.G. Harbour [20] desarrollaron el algoritmo heurístico HOPA para la asignación de prioridades fijas en sistemas distribuidos. Para conseguir mejores soluciones de forma más rápida, tiene en cuenta uno de los factores más determinantes en el comportamiento temporal de los sistemas, el tiempo de respuesta de peor caso de cada tarea.

El algoritmo HOPA se basa en la distribución del plazo de principio a fin ED_{ij} entre las actividades contenidas en la transacción. Una vez que cada actividad tiene asignado un plazo local "artificial", se les asignan prioridades fijas según la convención *deadline monotonic* (mayor prioridad a las actividades con menor plazo). A continuación se lleva a cabo el análisis del sistema con alguna técnica de análisis para sistemas distribuidos, normalmente haciendo uso de offsets para obtener resultados lo menos pesimistas posible. Teniendo en cuenta los resultados del análisis, se calculan nuevos plazos locales, y se repite el proceso hasta que se cumpla algún criterio de parada. El algoritmo puede describirse con el pseudocódigo 2.1.

Para simplificar la descripción del algoritmo HOPA, solo se tendrán en cuenta los plazos de principio a fin ED_{ij}. Para soportar plazos intermedios D_{ij} se

Listing 2.1: Algoritmo HOPA

```
Algoritmo HOPA is
begin
        Distribuye plazos ED entre las actividades
        loop
                Asigna prioridades según deadline monotonic
                Calculo de tiempos de respuesta
                exit when Se cumple algun requisito de parada
                Calcular nuevos plazos locales
        end loop
end HOPA
```

comentarán los cambios necesarios al final de esta sección.

La distribución inicial de los plazos de principio a fin puede ser cualquiera que cumpla :

$$ED_{ij} = \sum_{k} d_k \tag{2.2}$$

Donde k son todas las actividades que componen la transacción producida por el evento e_i, d_k los plazos locales asignados a las actividades, ED_{ij} el plazo de principio-fin de la transacción.

En el algoritmo HOPA, se reparte el plazo de forma proporcional a los tiempos de ejecución de cada actividad:

$$d_{ij} = \frac{ED_{ik} \cdot C_{ij}}{C_i} \tag{2.3}$$

Donde ED_{ij} es el plazo de principio a fin de la respuesta, C_{ij} es el tiempo de ejecución de peor caso de la actividad j para la respuesta al evento externo e_i, y C_i la suma de tiempos de ejecución de toda la respuesta a e_i.

Con los plazos locales asignados a cada actividad, se realiza la asignación de prioridades *deadline monotonic* de manera independiente en cada recurso, es decir, asignando las prioridades fijas de manera inversamente proporcional al plazo local.

Una vez que todas las actividades tienen asignadas una prioridad fija inicial,

podemos analizar el sistema con alguna técnica de las existentes. Si queremos conseguir altas utilizaciones en los recursos, deberemos utilizar una técnica de análisis lo menos pesimista posible, por lo tanto, el uso de la técnica basada en offsets sería una buena elección.

Con el análisis obtenemos los tiempos de respuesta de peor caso para cada actividad, r_{ij}, que nos da una imagen de cómo se está comportando el sistema con la asignación de prioridades actual. Si nuestros requerimientos temporales (descritos con los plazos globales D_{ij} o los plazos de principio a fin ED_{ij}), ya se cumplen con esta primera asignación de prioridades (o algún otro criterio de parada se cumple), se da por finalizado del algoritmo.

En caso contrario, se procede a recalcular las prioridades por otras que intenten hacer el sistema planificable. En primer lugar se vuelven a distribuir los plazos de principio a fin ED_{ij}, pero teniendo en cuenta la "distancia" que separa a cada actividad de la planificabilidad, concepto que llamamos *exceso*. Se contemplan dos definiciones de exceso de una actividad.

Exceso de tiempo de Respuesta:

$$exc(a_{ij}) = \begin{cases} (r_{ij} - d_{ij}) \cdot \frac{R_i}{ED_i} & \text{cuando } d_{ij} \leq T_i \\ (r_{ij} + J_{ij} - d_{ij}) \cdot \frac{R_i}{ED_i} & \text{cuando } d_{ij} > T_i \end{cases}$$

Exceso de tiempo de computación (*slack*):

$$exc(a_{ij}) = \begin{cases} (s_{ij}) \cdot \frac{R_i}{D_i} & \text{cuando } d_{ij} \leq T_i \\ (r_{ij} + J_{ij} - d_{ij}) \cdot \frac{R_i}{D_i} & \text{cuando } d_{ij} > T_i \end{cases}$$

Donde R_i es el tiempo de respuesta global de la última actividad de la respuesta. s_{ij} es el *slack* de la actividad a_{ij}, que se define como la cantidad de tiempo de ejecución (con signo negativo), que debe restarse a la actividad a_{ij} para poder cumplir su plazo local d_{ij}.

El hecho de tener dos definiciones para el concepto de *exceso* es debido a que la primera definición, exceso de tiempo de respuesta, obtiene resultados en mucho menos tiempo, ya que el cálculo del *slack* es un proceso computacionalmente costoso. En cambio, el *slack* obtiene mejores soluciones, ya que es una representación mas veraz del concepto de exceso.

En la implementación del HOPA utilizada, siempre se va a utilizar el exceso según el tiempo de respuesta, por su capacidad de obtener soluciones que en principio son válidas, en una menor cantidad de tiempo.

Se define el exceso de cada recurso PR_k como:

$$exc(PR_k) = \sum_{a_{ij} \in PR_k} exc(a_{ij}) \tag{2.4}$$

O lo que es lo mismo, el sumatorio de los excesos de todas las actividades localizadas en dicho recurso. Adicionalmente se define el máximo exceso de todos los recursos, y el máximo exceso de las actividades de una determinada respuesta:

$$Mex(PR) = \max_{\forall PR_i} |exc(PR_i)| \qquad Mex(e_i) = \max_{j \in A_i} |exc(a_{ij})| \tag{2.5}$$

Donde A_i constituye el grupo de actividades que forman la transacción que ocurre como respuesta al evento externo e_i.

Con estas definiciones se pueden calcular los nuevos plazos locales de la siguiente forma:

$$d_{ij}(nuevo) = d_{ij}(viejo) \left(1 + \frac{exc(PR_k)}{k_R \cdot Mex(PR)}\right) \left(1 + \frac{exc(a_{ij})}{k_a \cdot Mex(e_i)}\right) \tag{2.6}$$

Donde PR_k es el recurso sobre el que está localizada la actividad a_{ij}. k_R y k_a son parámetros que controlan la influencia relativa de los recursos y de las actividades en el cálculo del plazo. En la implementación del algoritmo, los valores oscilan típicamente entre 1.5 y 3.

Por último, es necesario el ajuste proporcional de los plazos locales nuevos, para hacerlos encajar en los plazos de principio a fin.

$$d_{ij} = \frac{d_{ij}(nuevo)}{\sum_{\forall k} d_{ik}} ED_{ij} \tag{2.7}$$

Como criterio de parada, en la implementación utilizada se finaliza el algoritmo en el momento en el que se encuentra una solución (asignación de prioridades fijas en las actividades) que haga el sistema planificable. Si se continúa ejecutando el algoritmo sobre una solución ya válida, es posible encontrar una solución aún mejor, por lo tanto, el algoritmo HOPA tiene la capacidad de optimizar las soluciones que obtiene. Para poder aprovechar esta peculiaridad, se puede definir un numero máximo de sobreiteraciones en las que intentar mejorar la solución.

Si queremos soportar plazos globales intermedios, es necesario modificar las ecuaciones de los excesos ligeramente. En lugar de usar R_i, se debe usar la diferencia entre las respuestas globales (R_{ij}) de las actividades anterior y posterior a la actividad a_{ij} (la actividad sobre la que está calculando el exceso) que tiene asignado el plazo global intermedio D_{ij}. De la misma forma, en lugar de utilizar ED_{ik}, se debe utilizar la diferencia entre los plazos globales correspondientes a ambas actividades.

2.2. Asignación de plazos de planificación en sistemas distribuidos EDF

En esta sección se va a afrontar el problema de la asignación de plazos de planificación en sistemas distribuidos. Según el modelo lineal definido en 1.1.2, partimos de un grupo de transacciones en las que se define un requisito temporal en cada una de ellas, en forma de plazos de principio a fin ED_{ij}.

Tal y como se vio en la sección 1.5, para poder ser planificadas con el algoritmo EDF, cada actividad debe tener asignado un *plazo de planificación*. A continuación se presentan dos algoritmos propuestos por Liu en [1] que asignan los *plazos de planificación* en función del plazo de principio a fin ED_{ij} de la transacción a la que pertenecen.

Cabe destacar que en un sistema distribuido EDF, los *plazos de planificación* no son plazos locales que deba cumplir el sistema de forma estricta, sino que se utilizan como parámetro de planificación. Los plazos que debe cumplir el sistema son los que se definen como plazos de principio a fin ED_{ij}, o los plazos globales D_{ij} en caso de definirlos.

2.2.1. Reparto Proporcional del Plazo

El algoritmo distribuye el plazo de principio a fin ED_{ik} de forma proporcional a los tiempos de ejecución de peor caso de cada actividad de la respuesta:

$$d_{ij} = ED_{ik} \cdot \frac{C_{ij}}{C_i} \tag{2.8}$$

Donde d_{ij} es el plazo de planificación asignado a la actividad a_{ij}, C_{ij} es el tiempo de ejecución de peor caso de la actividad a_{ij}, y C_i es la suma de tiempos de ejecución de peor caso de todas las actividades de la respuesta a e_i.

Se puede observar el funcionamiento del reparto proporcional en la figura 2.1

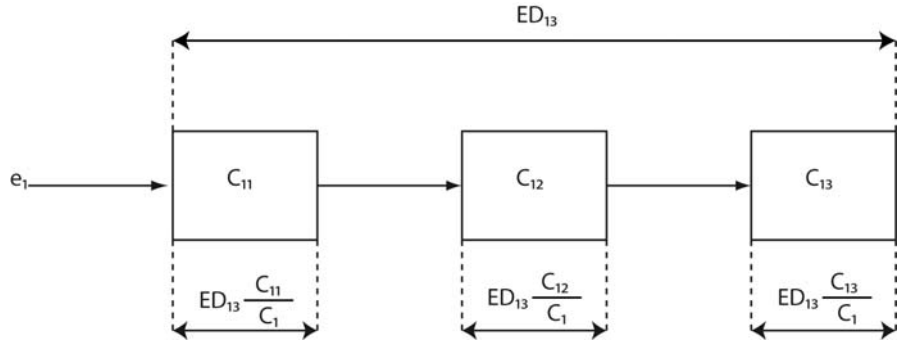


Figura 2.1: Distribución proporcional del plazo de principio a fin

En el caso de que existan actividades con plazos globales, el reparto del plazo se hace de manera similar, tal y como se muestra en la figura 2.2.

2.2.2. Reparto Proporcional Normalizado del Plazo

El algoritmo es similar al reparto proporcional del plazo, pero teniendo en cuenta además la carga en cada procesador. Se define según la siguiente ecuación:

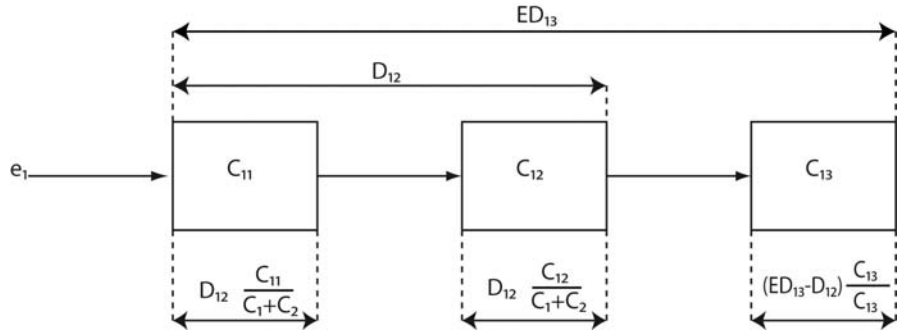


Figura 2.2: Distribución proporcional del plazo

$$d_{ij} = ED_{ik}\frac{C_{ij}U(V_{ij})}{\sum_{l=1}^{n(i)} C_{il}U(V_{il})} \tag{2.9}$$

Donde d_{ij} es el plazo de planificación asignado a la tarea a_{ij}, ED_{ik} es el plazo de principio a fin de la transacción i, C_{ij} el tiempo de ejecución de peor caso de la tarea a_{ij}, $U(V_{ij})$ es la utilización del procesador sobre el que se está ejecutando la tarea a_{ij}, tal y como se define en la ecuación 2.10. El sumatorio $\sum_{l=1}^{n(i)}$ recorre todas las tareas de la transacción que posee a_{ij}.

$$U(V_{ij}) = \frac{C_1}{T_1} + ... + \frac{C_n}{T_n} \tag{2.10}$$

El efecto que se produce en el reparto del plazo de principio a fin es que se asigna un plazo de planificación superior (con respecto al reparto proporcional) en las tareas situadas en procesadores con mayor utilización, a costa de asignar un plazo inferior (con respecto al reparto proporcional) en las tareas situadas en procesadores con menor utilización.

De los algoritmos presentados en [1], el reparto normalizado es el que proponen que obtiene mejores soluciones.

Capítulo 3

Algoritmo Heurístico para la asignación de plazos de planificación en sistemas de tiempo real distribuidos

En el capítulo anterior se presentaron dos algoritmos sencillos de asignación de plazos de planificación, el reparto proporcional y el reparto normalizado. Adicionalmente, se estudió el algoritmo HOPA de asignación de prioridades fijas para sistemas distribuidos. En el capítulo presente se va a proponer un algoritmo heurístico de asignación de plazos de planificación para sistemas distribuidos.

Los algoritmos de reparto proporcional y reparto normalizado calculan los plazos de planificación de las actividades del sistema de una manera rápida y sencilla, pero si la asignación inicial de plazos de planificación que calculan no consigue hacer al sistema planificable, carecen de ”inteligencia” para explorar otras soluciones.

El algoritmo HOPA, en cambio, cuando la primera o sucesivas soluciones calculadas no podían conseguir la planificabilidad del sistema, implementaba mecanismos que permitían la búsqueda de una solución al problema de la asignación de prioridades fijas. Para sacar provecho de esta característica, el algoritmo heurístico de asignación de plazos de planificación que vamos a proponer va a tomar como base el algoritmo HOPA.

Partimos de un grupo de transacciones, definidas según el modelo lineal, en las que se definen como requisitos temporales los plazos de principio a fin ED_{ij}. Como ya se vio en el capítulo 1.5, para poder ser planificadas según EDF, las actividades del sistema deben tener asignados unos plazos de planificación.

Como se puede observar en el listado 2.1, el algoritmo HOPA asigna a cada actividad un plazo local, en primer lugar repartiendo proporcionalmente el plazo de principio a fin, y posteriormente recalculando el plazo local según la ecuación 2.6. Estos plazos locales se pueden utilizar como plazos de planificación de las actividades. Por lo tanto resulta evidente que si en el algoritmo HOPA eliminamos el paso en el que se asignan prioridades fijas *deadline monotonic*, y tomamos los plazos locales como *plazos de planificación* de las actividades, obtenemos un nuevo algoritmo heurístico de asignación de plazos de planificación para sistemas distribuidos. Cabe destacar que los plazos locales que se calculan no son plazos estrictos que se deban cumplir, sino que se utilizan únicamente para ser utilizados como plazos de planificación

El algoritmo heurístico resultante se describe en el listado 3.1. Salvo la asignación *deadline monotonic* de las prioridades, que desaparece, todos los demás pasos se mantienen intactos.

Listing 3.1: Algoritmo heurístico de asignación de plazos de planificación

```
Algoritmo Heuristico EDF is
begin
        Distribuye plazos ED entre las actividades
        loop
                Calculo de tiempos de respuesta
                exit when Se cumple algun requisito de parada
                Calcular nuevos plazos locales
        end loop
end Heuristico EDF
```

Hay que mencionar que en la primera iteración el algoritmo heurístico de asignación de plazos de planificación, y el algoritmo de reparto proporcional, obtienen los mismos plazos de planificación. Por lo tanto, si esta primera asignación consigue hacer planificable al sistema, y por el criterio de parada utilizado se detienen las iteraciones al encontrar la primera solución válida, el algoritmo heurístico propuesto y el algoritmo de reparto proporcional obtienen la misma solución al problema.

Al igual que el algoritmo HOPA, y tal y como se verá en el capítulo de evaluación de las técnicas (capítulo 6), el algoritmo heurístico propuesto en esta sección tiene la capacidad de optimizar una solución ya válida, por lo que se puede definir un parámetro adicional que especifique las sobreiteraciones sobre planificable que se permiten realizar para optimizar la solución.

El algoritmo heurístico de asignación de plazos de planificación propuesto aparece resumido gráficamente en la figura 3.1. En el capítulo 4 se llevará a cabo la implementación software de este algoritmo.

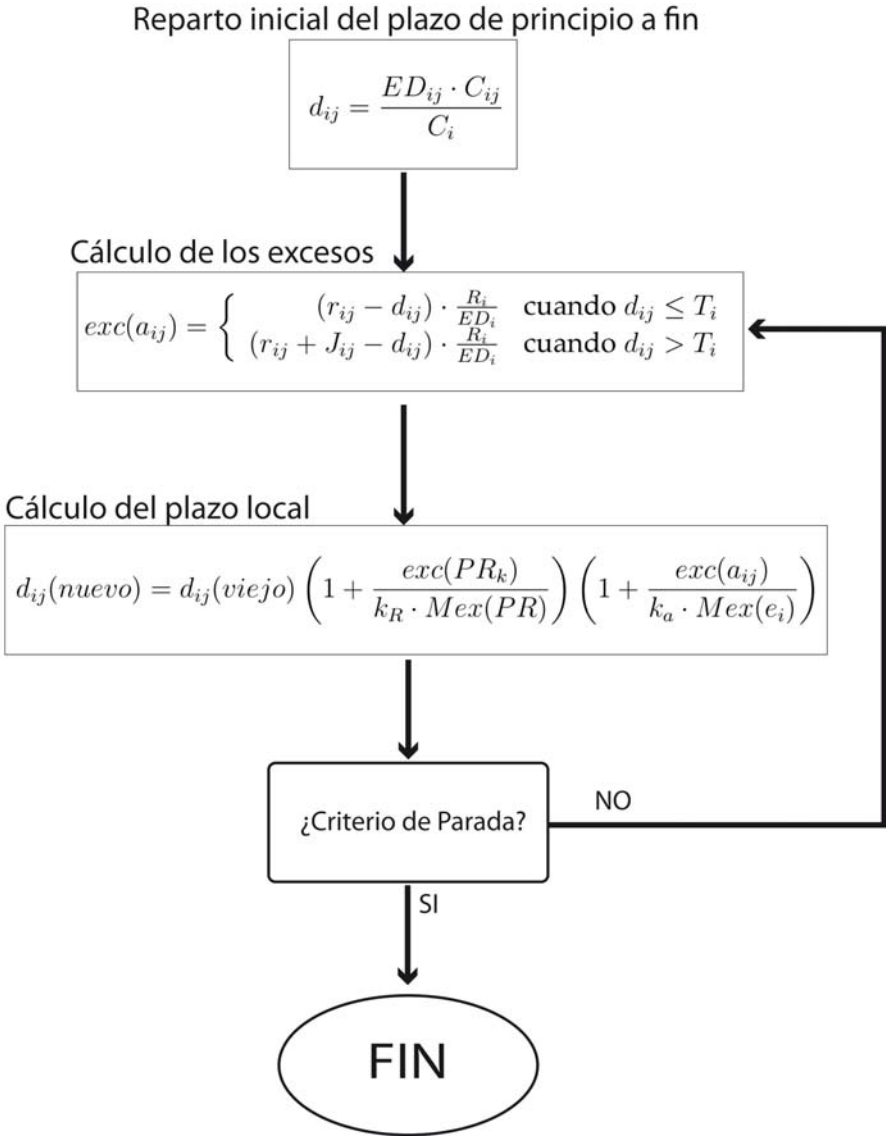


Figura 3.1: Algoritmo heurístico de asignación de plazos de planificación

Capítulo 4

Implementación de las Técnicas de Asignación de Plazos

En capítulos anteriores se han estudiado dos técnicas de asignación de plazos de planificación descritas por Liu [1], y adicionalmente se ha propuesto una tercera técnica, heurística, con el propósito de mejorarlas. Con el objetivo de poder evaluar la capacidad para alcanzar soluciones de estas técnicas, y realizar un estudio comparativo entre ellas, se hace necesario desarrollar una herramienta software que las implemente. Esta herramienta debe tomar un sistema distribuido de tiempo real dado, y afrontar su planificación, utilizando la técnica que se le especifique, además de tener la capacidad de tomar los parámetros característicos de cada algoritmo, y mostrar los resultados de una forma sencilla.

El software a desarrollar se ha integrado en la herramienta de modelado y análisis de sistemas de tiempo real MAST. MAST es un conjunto de herramientas de software libre, programado en ADA, y desarrollado en el grupo de Computadores y Tiempo Real de la Universidad de Cantabria, que describe un modelo con el que definir los sistemas a analizar e integra técnicas de análisis y planificación tanto para sistemas monoprocesador como distribuidos.

En el momento del comienzo de este proyecto, MAST poseía un gran soporte para el análisis y planificación de sistemas con prioridades fijas, integrando las técnicas de análisis *Holistic* y *Offset Based* entre otras, además del algoritmo de asignación de prioridades HOPA. En cambio su soporte para sistemas EDF estaba más limitado, ya que carecía de herramientas tanto de análisis como de asignación de plazos para sistemas distribuidos.

La herramienta de análisis para sistemas distribuidos EDF la integró de forma separada, pero en paralelo con este proyecto, Jose Carlos Palencia, del grupo de Computadores y Tiempo Real. Esta herramienta es una implementación del análisis mediante offsets para sistemas distribuidos EDF, introducida en la sección 1.4.4, para el caso en el que las actividades se planificaban con sus plazos locales. La inclusión de las técnicas de asignación de plazos de planificación será el cometido de este proyecto, y se describirá en esta sección.

4.1. Consideraciones previas para la integración en la herramienta MAST

El objetivo es integrar las técnicas de asignación de plazos de planificación (reparto proporcional,reparto normalizado,algoritmo heurístico) dentro del entorno MAST. La herramienta debe tomar como entrada la descripción del sistema distribuido, además de los parámetros necesarios para llevar a cabo la asignación.

Como se vio en los capítulos 2 y 3, los algoritmos de asignación de plazos, especialmente el heurístico que se propone, se apoya en el análisis continuado del sistema, por lo que podría tener sentido el poder elegir qué técnica de análisis utilizar, aunque actualmente sólo hay disponible una técnica de análisis para sistemas distribuidos EDF. Se deja abierta la opción de la utilización de otras técnicas de análisis mejores que se puedan integrar en el futuro.

Como resultado de la ejecución de nuestra herramienta, se debe obtener en primer lugar si se ha conseguido una asignación de plazos que logre la planificabilidad del sistema, y si se ha conseguido, con qué asignación de plazos de planificación.

En la figura 4.1, extraída del manual de referencia de MAST [21], se puede observar el esquema en el que está dividido MAST. En rojo se indican las herramientas que se van a añadir. El flujo de trabajo que se va a realizar internamente en MAST a la hora de utilizar una herramienta de asignación de plazos se indica con las flechas verdes, y se describe a continuación:

1. Se toma el fichero de entrada en el que se describe en formato MAST el sistema de tiempo real en formato MAST

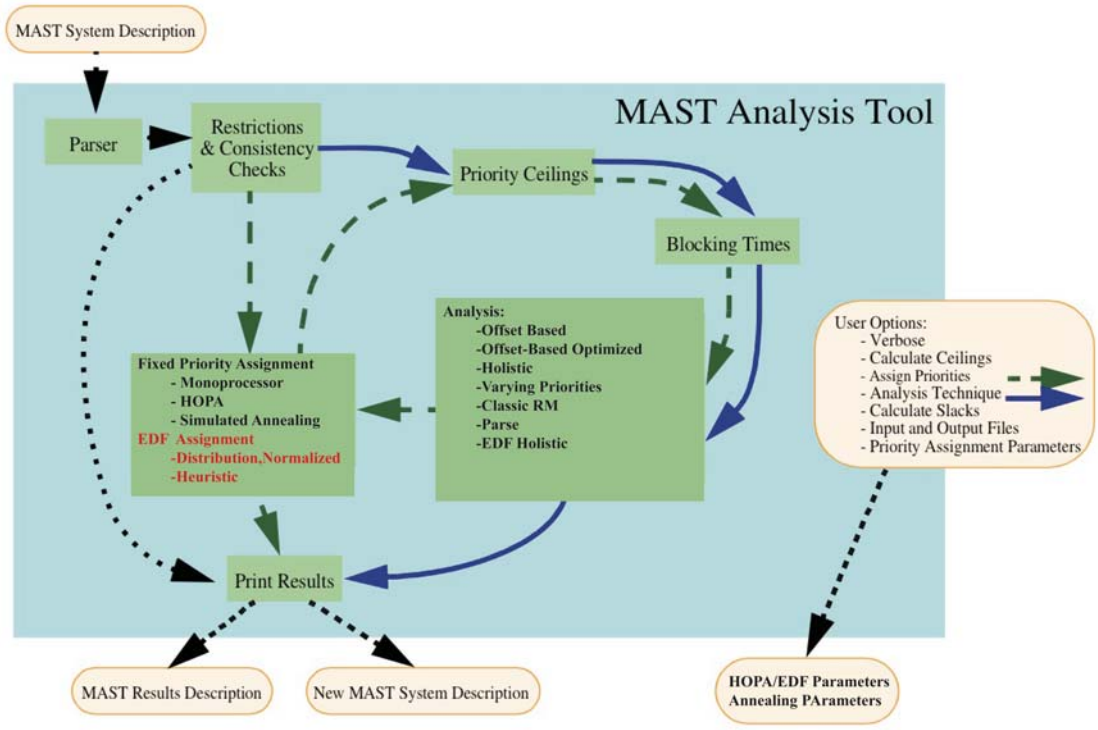


Figura 4.1: Distribución de las herramientas MAST

2. El Parser traduce el fichero de entrada a estructuras Ada, fácilmente manipulables a través de los paquetes Ada incluidos en MAST.

3. Se comprueba si se cumplen una serie de restricciones en el sistema de entrada, como por ejemplo, que haya una carga igual o inferior al 100 % en los procesadores. Una carga por encima del 100 % nunca es planificable.

4. La herramienta de asignación de plazos empieza a ejecutarse sobre el sistema de entrada, haciendo uso de las herramientas de análisis cuando sea necesario. Los pasos de cálculo de techos de prioridad y términos de bloqueo se llevan a cabo cuando el sistema posee recursos compartidos. En este trabajo no se van a estudiar los efectos de los recursos compartidos.

5. Una vez que el algoritmo de asignación de plazos ha finalizado, muestra los resultados obtenidos. En los resultados se especifica si se ha alcanzado una asignación que hace el sistema planificable, y con qué plazos de planificación se ha conseguido. Adicionalmente, para medir la "calidad" del resultado, y poder compararlo con otros resultados obtenidos con otras técnicas de asignación, se mostrará el *índice de planificabilidad* logrado.

El hecho de utilizar una herramienta ya madura como MAST, nos provee de un modelo con el que describir el sistema distribuido, y de las funciones con las que traducir ese modelo de entrada a estructuras Ada con las que trabajar fácilmente. De esta forma sólo nos tenemos que preocupar de la implementación del algoritmo en sí. A continuación se va a describir brevemente como es el modelo con el que MAST define un sistema de tiempo real.

4.1.1. Descripción del sistema distribuido con el modelo MAST

En el modelo MAST de sistemas de tiempo real, la unidad mínima de trabajo es la *actividad*. Cada actividad lleva asociadas una serie de características y relaciones que la definen, tal y como se ve en la figura 4.2, extraída del manual de referencia de MAST [21].

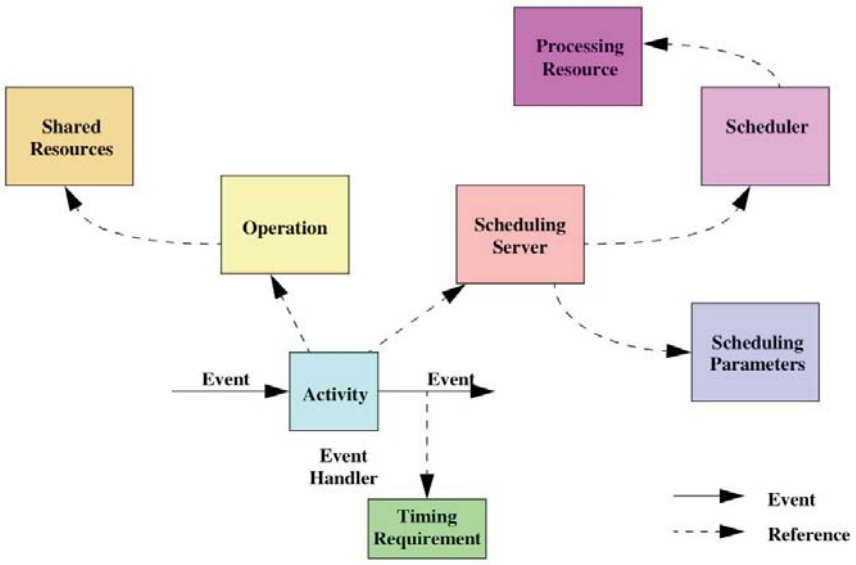


Figura 4.2: Elementos que definen una actividad

Cada uno de los elementos que aparecen en la figura 4.2 deben describirse en el fichero de entrada, y representan:

- **Procesador**(*Processing Resource*). Modelan la capacidad de procesamiento de un componente hardware que ejecuta las actividades. En su descripción se debe especificar si es un Procesador o una Red de transmisión de mensajes.

- **Planificador**(*Scheduler*). Deben especificar a qué procesador van asociados y qué política de planificación utilizan (prioridades fijas, EDF...)
- **Servidor de planificación**(*Scheduling Server*). Se encargan de ejecutar las actividades. En ellos se especifica qué planificador van a utilizar(y por consiguiente, en qué procesador van a ejecutarse). En los *parámetros de planificación* se pueden especificar características adicionales como *Polling Servers* o *servidores esporádicos*, pero no van a ser utilizados en este trabajo.
- **Operaciones**(*Operation*). Modelan el trabajo que realiza la actividad, y se describe con su tiempo de ejecución de peor caso, y los posibles recursos compartidos que utilice. En este trabajo no se describirán recursos compartidos.
- **Actividades**(*Activity*). Representan una instancia de una *operación*, que va a ser ejecutado por un *servidor de planificación*
- **Eventos**(*Events*). Son los encargados de activar los *manejadores de eventos*. Si son externos, pueden ser periódicos o aperiódicos. Sin son periódicos se debe especificar el periodo, y opcionalmente su *jitter de entrada* y *fase*. Los eventos externos en este trabajo serán periódicos, sin *jitter de entrada*. Los eventos internos son generados por los *manejadores de eventos*, y pueden llevar asignados *requisitos temporales* que se describen aparte.
- **Manejadores de Eventos**(*Event Handler*). Cada *manejador de evento* asocia a cada actividad con el *evento* de entrada que la activa, el *evento* de salida que genera cuando finaliza su ejecución, la *operación* que ejecuta la actividad, y que *servidor de planificación* lleva asociado.
- **Requisitos Temporales**(*Timing Requirement*). Van asociados a algún *evento interno*. En este trabajo se utilizará el *plazo global estricto*, que modelará el plazo de principio a fin ED_{ij} definido en el modelo lineal de la sección 1.1.2
- **Recursos Compartidos** (*Shared Resources*). Representan los recursos que son compartidos por diferentes actividades, y que se deben acceder de forma mutuamente exclusiva. En este trabajo no se definirán recursos compartidos.

Finalmente, en el fichero de entrada se deben describir las transacciones que componen el sistema. Como se observa en la figura 4.3, obtenida del manual

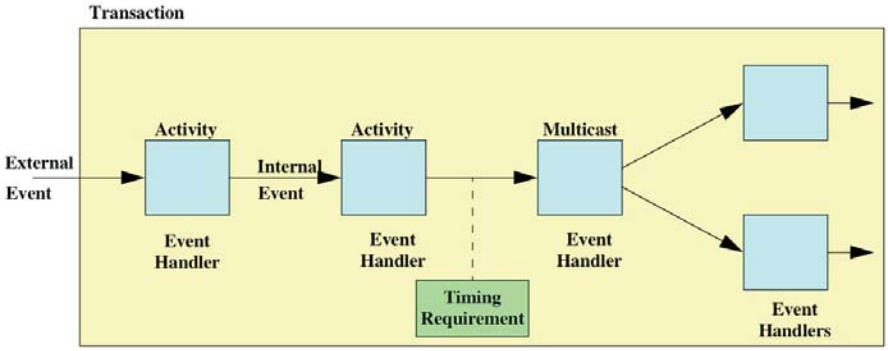


Figura 4.3: Elementos que definen una transacción

de referencia de MAST [21], están formados por *eventos*, *manejadores de eventos* y *requisitos temporales*

Hay que hacer notar una limitación importante de la herramienta MAST, y es que actualmente no permite especificar una red de comunicación planificada según EDF, ya que no existen redes planificadas según EDF en la realidad. Por lo tanto, no se pueden describir sistemas distribuidos EDF, algo realmente necesario en este proyecto. El problema no es tan importante si se tiene en cuenta que, como en el caso de prioridades fijas, las redes de comunicación se pueden modelar como si fueran un procesador más. Por lo tanto, en este trabajo se analizarán sistemas compuestos únicamente por procesadores, que simularán ser sistemas distribuidos, en donde la comunicación entre ellos se supondrá instantánea.

Toda la información que describe el sistema de tiempo real a analizar aparece escrita, con un formato determinado, en un archivo de texto plano. MAST dispone también de una herramienta gráfica con la que generar estos archivos de descripción. El objetivo de este trabajo es comparar algoritmos de asignación de parámetros de planificación, y para ello , como se verá en el capítulo 6, se deben generar multitud de sistemas con distintas cargas de trabajo cada uno. Usar la herramienta gráfica para generarlos sería un trabajo prohibitívamente costoso. Por ello, como otro objetivo en este proyecto, se ha desarrollado una herramienta generadora automática de ejemplos, que se describe en el capítulo 5.

Una vez que se han descrito las peculiaridades a considerar a la hora de integrar las técnicas en MAST, en las siguientes secciones se va a describir cómo se han implementado los 3 algoritmos de planificación, el reparto proporcional, el reparto proporcional normalizado, y el algoritmo heurístico propuesto.

4.2. Integración de los algoritmos de reparto del plazo en la herramienta MAST

Según se vio en el capítulo 3, los algoritmos de reparto proporcional y reparto proporcional normalizado son muy similares, por lo que se describirán sus implementaciones de forma conjunta. El flujo de la ejecución hace uso de múltiples paquetes Ada. Para simplificar la descripción de la implementación, y hacerla independiente del lenguaje utilizado, se describirá a continuación en forma de pseudocódigo:

Listing 4.1: Pseudocódigo Reparto de Plazos

```
Algoritmo Reparto_Plazos is
begin
        Lectura del archivo de entrada
        Comprobar restricciones;
        Traducción del sistema de entrada MAST al modelo lineal;
        Reparto del Plazo de principio a fin en cada transacción;
        Traducción del sistema lineal al modelo MAST;
        Realizar el análisis de planificabilidad sobre el sistema;
        Mostrar Resultados;
end Reparto_Plazos;
```

A continuación se describen brevemente cada una de las operaciones descritas en el pseudocódigo 4.1:

- **Lectura del archivo de entrada y comprobación de restricciones**. El parser de MAST lee el archivo de entrada en el que se describe el sistema, y comprueba que no contenga errores sintácticos. Si no los hay, traduce el archivo a estructuras Ada. El sistema está ahora almacenado en una variable Ada, pero definido con el modelo MAST. Sobre el sistema contenido en la variable Ada se realizan más verificaciones, tales como la comprobación de que la utilización del sistema sea menor que el 100 %, que posea únicamente planificadores EDF, o que la herramienta de análisis especificada sea la correcta para sistemas EDF distribuidos. Si no se cumplen las restricciones, finaliza la ejecución.
- **Traducción**. Se toman las estructuras Ada creadas por el Parser que definen el sistema, y se traducen a otras, mas sencillas, que describen el sistema

según el modelo lineal, más apropiadas para trabajar sobre ellas. La traducción del modelo lineal al modelo MAST se realiza porque la herramienta de análisis debe recibir el sistema en formato MAST.

- **Reparto del Plazo de principio a fin**. Para cada transacción, se toma el plazo de principio a fin, y se reparte entre sus actividades. Éste reparto puede ser de dos tipos posibles, o bien es un Reparto Proporcional, tal y como se ha descrito en 2.2.1; o bien un Reparto Proporcional Normalizado, descrito en 2.2.2. La elección de uno u otro tipo de reparto se especifica en los parámetros de entrada en la herramienta.
- **Análisis de Planificabilidad**. Se realiza el análisis de planificabilidad sobre el sistema, con la asignación de plazos previamente calculada.
- **Mostrar Resultados**. En los resultados, que se pueden mostrar tanto en pantalla, como en un fichero de salida, se muestran los plazos de planificación calculados, los tiempos de respuesta de cada transacción y el índice de planificabilidad obtenido, además de especificar claramente si el sistema es planificable con la asignación de plazos calculada.

4.3. Integración del algoritmo heurístico de asignación de plazos propuesto en MAST

A continuación se va a describir la implementación del algoritmo heurístico propuesto en el capítulo 3. Para comenzar la descripción, vamos a especificar los parámetros de entrada que va a necesitar el algoritmo heurístico:

- **Lista k**. Tal y como se vio en la sección 3, el algoritmo heurístico hace uso de dos parámetros, k_a y k_r. En la implementación del algoritmo, se van a definir como parámetros de entrada dos listas, una con valores de k_a y otra con valores de k_r, teniendo ambas listas la misma longitud. Durante la ejecución del algoritmo se irán tomando parejas (k_a, k_r) con los valores de estas listas. En cada pareja que se forma, ambos valores ocupan la misma posición (índice del array) en sus respectivas listas.
- **Lista de iteraciones**. Es una lista , en la que cada elemento especifica cuantas iteraciones realizar sobre cada pareja (k_a, k_r) posible. Por ejemplo, una lista de iteraciones puede ser (10,20,30), con la que, si la *lista k* mide 3 (hay 3 posibles parejas k_a, k_r), se realizarían $10 \cdot 3 + 20 \cdot 3 + 30 \cdot 3 = 180$ iteraciones.

- **Sobreiteraciones**. En la sección 3 se afirmó que, una vez que se encuentra una asignación de plazos que haga que el sistema sea planificable, es posible encontrar una solución mejor (en términos del índice deplanificabilidad) si se sobreitera sobre esta solución. Para aprovechar esta característica, en la implementación de define un parámetro de entrada llamado *sobreiteraciones* que define el número de iteraciones que ejecutará el algoritmo después de encontrar una solución para la asignación de plazos de planificación. Gracias a este parámetros se puede delimitar el tiempo que emplea el algoritmo optimizando una solución.

Estos tres parámetros (lista k, lista de iteraciones, sobreiteraciones) irán contenidos en un fichero aparte, que leerá la herramienta. En caso de no existir este fichero se tomaran valores por defecto, que son los siguientes: Lista $k_r = (1{,}5, 2, 3)$; lista $k_a = (1{,}5, 2, 3)$; Lista de iteraciones = $(10, 20, 30)$; Sobreiteraciones = 0.

Los 3 parámetros (lista k, lista de iteraciones, sobreiteraciones) van a conformar la implementación del algoritmo, que se presenta en forma de pseudocódigo a continuación:

Listing 4.2: Pseudocódigo Asignación heurística de plazos

```
Algoritmo Heurístico_EDF is
begin
        Lectura del archivo de entrada;
        Carga de los parámetros de planificación;
        Comprobación de restricciones;
        Reparto inicial de los plazos de principio a fin;
        for tamaño [lista de iteraciones]
        loop
                Establece [número de iteraciones];
                for tamaño [lista k]
                loop
                        Establece los valores de (k_a,k_r);
                        for [número de iteraciones]
                        loop
                                Análisis de planificabilidad;
                                exit when Criterio de Parada;
                                Recalcular Plazos de planificación;
                        end loop;
                end loop;
        end loop;
        Realizar último análisis de Planificabilidad;
        Mostrar Resultados;
end Heurístico_EDF;
```

A continuación se describen brevemente cada una de las operaciones descritas en el listado 4.2:

- **Lectura del archivo de entrada**. El parser de MAST lee el archivo de entrada, en donde está descrito el sistema. Si no hay errores sintácticos, traduce el sistema a estructuras Ada, siendo almacenado en una variable.
- **Carga de parámetros de planificación**. Se lee el archivo en el que están situados los parámetros (lista k, lista de iteraciones, sobreiteraciones). Si no existe dicho archivo, se tomarán los siguientes valores por defecto : Lista $k_r = (1{,}5, 2, 3)$; lista $k_a = (1{,}5, 2, 3)$; Lista de iteraciones = $(10, 20, 30)$; Sobreiteraciones = 0.
- **Comprobación de Restricciones**. Sobre las estructuras Ada que devuelve el parser, se comprueban restricciones tales como que la utilización del sistema sea menor del 100 %, o que el sistema posea únicamente planificadores EDF. Si no se cumplen las restricciones, finaliza la ejecución.
- **Reparto inicial de los plazos de principio a fin**. Para cada transacción, se toma el plazo de principio a fin, y se reparte entre sus actividades. En la implementación realizada se hace un reparto proporcional a los tiempos de ejecución de cada actividad.
- **Establece número de iteraciones**. Asigna a la variable [número de iteraciones] el valor de [lista de iteraciones] correspondiente a la iteración actual;
- **Establece valores** k_a, k_r. Se asignan a las variables k_a, k_r los valores de [lista k] correspondientes a la iteración actual;
- **Análisis de Planificabilidad**. Con la asignación de plazos de planificación actual, realiza el cálculo de los tiempos de respuesta de todas las actividades del sistema.
- **Criterio de Parada**. El algoritmo finaliza si se verifica alguna de estas circunstancias:
  - Finalizan todas las iteraciones definidas por [lista k],[lista de iteraciones].
  - Se encuentra una asignación de plazos de planificación que hace que el sistema sea planificable, y no se han definido sobreiteraciones.
  - Se sobrepasa el número preestablecido de sobreiteraciones definidas por [Sobreiteraciones].

- **Recalcular los Plazos de planificación** Cuando no se cumple ningún criterio de parada, se procede a recalcular los plazos de planificación, haciendo uso del algoritmo descrito en el capítulo 3 (cálculo de excesos, cálculo de los nuevos plazos locales), utilizando el par $k_a r, k_r$ actual.

- **Mostrar Resultados**. En los resultados, que se pueden mostrar tanto en pantalla, como en un fichero de salida, se muestran los plazos de planificación calculados, los tiempos de respuesta de cada transacción, el índice de planificabilidad obtenido en cada iteración, además de especificar claramente si se ha conseguido una asignación de plazos de planificación que haga al sistema planificable.

Capítulo 5

Herramienta de generación automática de ejemplos

En este proyecto queremos comparar una serie de algoritmos de planificación, y para ello, se deben generar multitud de sistemas, con diferentes cargas de trabajo, para comprobar cómo se comportan cada uno de los algoritmos según aumenta la carga. Resulta evidente que para un estudio exhaustivo se necesita estudiar el mayor número posible de sistemas. Generar manualmente un archivo que contenga la descripción del sistema de tiempo real a analizar es un proceso costoso en cuanto a la cantidad de tiempo requerido, por lo que el disponer de una herramienta con la que se puedan generar cientos de sistemas de forma automática, simplemente especificando unos parámetros de entrada, resulta altamente atractivo. Es por ello que se creó un generador automático de ejemplos.

Esta herramienta, desarrollada en Ada, toma como parámetros de entrada las siguientes características de un sistema de tiempo real:

- Número de procesadores del sistema.
- Número de transacciones del sistema.
- Número de tareas en cada una de las transacciones.
- En cada transacción, sobre qué procesador va a ejecutarse cada tarea.
- Periodos de cada transacción.

Faltan por definir dos parámetros fundamentales, los plazos de principio a fin de cada transacción, y los tiempos de ejecución de cada tarea, que definen la utilización media de los procesadores. Estos dos parámetros no se especificarán de forma explícita al generador, sino que éste los calculará según convenga.

La herramienta de generación va a crear automáticamente 5 copias exactas del sistema que está generando, con un plazo de principio a fin distinto en cada una de ellas. El objetivo es poder estudiar el algoritmo heurístico propuesto sobre sistemas con diferentes requisitos temporales. Los 5 plazos que se van a generar son los siguientes:

1. Plazo de principio a fin $ED_{ij} = T_i$, siendo T_i el periodo de la transacción i.
2. $ED_{ij} = \frac{RT_i}{2}$, siendo R el número de actividades de la transacción i.
3. $ED_{ij} = RT_i$. Éste es el mas común en los sistemas distribuidos.
4. $ED_{ij} = 2RT_i$
5. $ED_{ij} = random(T_i, 2RT_i)$, un número aleatorio entre el periodo de la transacción, y $2RT_i$.

En el pseudocódigo 5.1 se muestra el funcionamiento básico del generador de ejemplos.

Listing 5.1: Pseudocódigo Generador de ejemplos

```
Algoritmo Generador is
begin
        Toma características del sistema base;
        Se asignan tiempos de ejecución iniciales;
        Se guardan 5 copias EDF y 5 copias prioridades fijas;
        loop
                Se selecciona aleatoriamente un procesador
                if (utilización procesador) < 95% then
                        Tarea aletoria en el procesador;
                        Se aumenta carga en tarea;
                        Se guardan 10 copias (EDF, prio. fijas);
                end if;
                exit when (utilización en todos procs.)>95%
        end loop;
end Generador;
```

A continuación se describen brevemente las operaciones descritas en el pseudocódigo 5.1:

1. Toma las características para el sistema base : número de procesadores del sistema, número de transacciones, número de tareas en cada una de las transacciones, procesador en el que se ejecuta cada tarea, y el periodo de cada transacción.

2. Asigna de forma inicial a todas las tareas un tiempo de ejecución de peor caso igual a 1. Esto provoca que la utilización media de los procesadores sea baja, en torno al 10 % dependiendo de cada sistema. Se guardan 5 copias de este sistema inicial (una para cada tipo de plazo de principio a fin) con planificadores EDF, y otras 5 copias con planificadores de prioridades fijas.

3. Se procede a aumentar la carga en el sistema. Para ello se selecciona aleatoriamente un procesador, de entre los procesadores que tengan una carga por debajo del 95 %. A continuación, se selecciona aleatoriamente una tarea de las que se ejecutan en este procesador, y se aumenta su tiempo de ejecución de peor caso según la siguiente fórmula:

$$C_{\text{nuevo}} = C_{\text{anterior}} + 0{,}01 \cdot \text{porcentaje} \cdot T_i \tag{5.1}$$

   La fórmula provoca que la utilización del procesador seleccionado aumente un *porcentaje* %. Se va a utilizar un *porcentaje*=5. A continuación se guardan 5 copias (una por cada tipo de plazo) del sistema con la nueva utilización media y planificadores EDF, y otras 5 copias con la nueva utilización media y planificadores por prioridades fijas.

4. Se repite el paso 2. Cuando todos los procesadores hayan alcanzado más de un 95 % de carga (consecuentemente, carga media del sistema superior al 95 %), se finaliza la ejecución del generador.

Recapitulando, en la figura 5.1 se resumen los sistemas que se obtienen tras una ejecución del generador.

Cada fichero que se genera sigue el formato descrito por MAST, por lo que nos valen como archivos de entrada. En cada ejecución del generador se crearán en torno a 900 archivos de entrada de una forma prácticamente instantánea, por lo que el ahorro de tiempo y trabajo es notable. Todos estos archivos generados nos permiten ejecutar los algoritmos de planificación de forma reiterada sobre el

Parámetros que definen el sistema base	número de procesadores del sistema, número de transacciones, número de tareas en cada una de las transacciones, procesador en el que se ejecuta cada tarea, y el periodo de cada transacción				
Plazos de principio a fin de cada transacción	EDi=Ti	EDi=R*Ti/2	EDi=R*Ti	EDi=2*R*Ti	Random(Ti, 2*R*Ti)
Tiempos de ejecución de peor caso.	De forma que se obtengan sistemas con utilizaciones medias desde aproximadamente el 10% hasta el 96%.				
Planificadores	Una copia para EDF y otra para prioridades fijas.				
Nº de sistemas generados aproximadamente	180	180	180	180	180

Figura 5.1: Tabla resumen

mismo sistema, pero con diferentes cargas en los recursos, para poder hacernos una idea de hasta qué porcentaje de utilización media en los procesadores es capaz de encontrar una asignación de plazos que haga el sistema planificable.

Capítulo 6

Evaluación y Comparación de las Técnicas de Planificación

En el capítulo 2 se hizo un estudio de tres técnicas de asignación de parámetros de planificación, una técnica heurística de asignación de prioridades fijas (HOPA), y dos técnicas de asignación de parámetros de planificación para sistemas distribuidos EDF (reparto proporcional y normalizado). En el capítulo 3 se propuso un algoritmo heurístico de asignación de parámetros de planificación para sistemas distribuidos EDF.

Con el objetivo de poder hacer un estudio comparativo entre estos algoritmos de asignación de parámetros de planificación, en el capítulo 4 se describió cómo se llevó a cabo la implementación de las técnicas de asignación de plazos de planificación en la herramienta MAST.

El estudio comparativo requiere aplicar estas técnicas de asignación de parámetros de planificación sobre un elevado número de sistemas. Para ello se creó una herramienta generadora de ejemplos, descrita en 5. El objetivo era definir un sistema base, especificando el número de procesadores del sistema, número de transacciones, número de tareas en cada una de las transacciones, procesador en el que se ejecuta cada actividad, y el periodo de cada transacción. Los tiempos de ejecución de peor caso de cada tarea se calculan para que aumente de forma progresiva la utilización media en los procesadores del sistema. Adicionalmente van a hacer 5 copias de todos los archivos generados, una para cada tipo de plazo de principio a fin definido.

El estudio comparativo se llevará a cabo en dos partes. En un primer lugar se compararán entre sí los algoritmos de asignación de plazos de planificación para sistemas EDF (reparto proporcional, reparto normalizado, algoritmo heurístico propuesto). El objetivo es verificar la bondad del algoritmo propuesto comparándolo con las técnicas existentes. A continuación se comparará el algoritmo heurístico de asignación de plazos de planificación para sistemas EDF, con el algoritmo HOPA para sistemas con prioridades fijas. Con ésto se pretende estudiar si, como ocurría con los sistemas monoprocesadores, la planificación EDF también es capaz de alcanzar mayores utilizaciones en sistemas distribuidos o no.

El sistema base sobre el que se realizará el estudio es el mostrado en la figura 6.1. La herramienta de análisis de sistemas EDF distribuidos continúa en desarrollo, y presenta actualmente una serie de limitaciones. Una de ellas es que el sistema no puede poseer un elevado número de tareas. Por esta razón, el sistema base utilizado es relativamente pequeño, aunque perfectamente válido para la consecución de nuestros objetivos. Está compuesto por cuatro procesadores y cinco transacciones. Los cinco tipos de plazos de principio a fin ED_{ij} que tendrán las transacciones se resumen en la figura 6.2

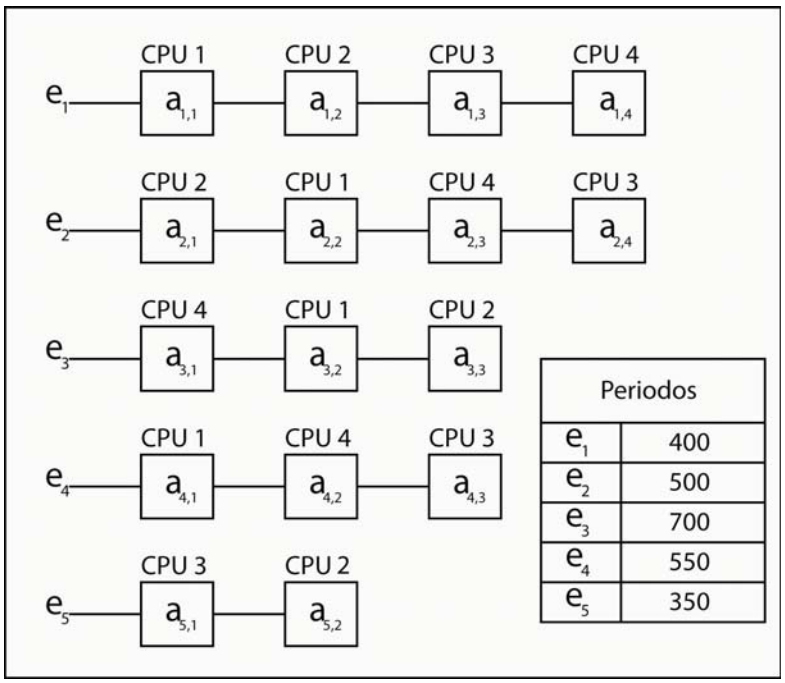


Figura 6.1: Sistema base utilizado en el estudio

En la herramienta MAST no se especifican unidades de tiempo. Cuando se

	transacciones				
Tipos de plazo	1	2	3	4	5
$ED_{ij} = T_i$	400	500	700	550	350
$ED_{ij} = \frac{RT_i}{2}$	800	1000	1050	825	350
$ED_{ij} = RT_i$	1600	2000	2100	1650	700
$ED_{ij} = 2RT_i$	3200	4000	4200	3300	1400
$ED_{ij} = \text{random}(T_i, 2RT_i)$	2088	1367	1219	2764	1297

Figura 6.2: Tabla resumen con los plazos de principio a fin utilizados

especifica que un evento tiene un periodo $T = 500$ por ejemplo, el diseñador debe saber qué unidad de tiempo está utilizando, por ejemplo milisegundos, y ser consecuente en el resto del sistema (periodos, tiempos de ejecución de peor caso, etc).

En *índice de planificabilidad* que se va utilizar en la presentación de los resultados estará normalizado entre [-1,1]. Un valor de *indice*=1 indica que los tiempos de respuesta han sido 0, y un valor del índice igual a -1 indica tiempos de respuesta infinitos. Recordar que un índice de planificabilidad negativo indica la no planificabilidad del sistema. Para simplificar la presentación de los resultados, los valores del *índice de planificabilidad* cuando el sistema no es planificable se asignan a -1.

6.1. Estudio comparativo entre algoritmos de asignación de plazos de planificación

Se compararán los tres algoritmos de asignación de plazos de planificación (reparto proporcional, normalizado y heurístico). Para comprobar la capacidad del algoritmo heurístico de optimizar una solución, el algoritmo heurístico se ejecutará dos veces, una son Sobreiteraciones=0, y otra con Sobreiteraciones=5. Para los parámetros *lista k* y *lista iteraciones* se toman los valores por defecto. A continuación se muestran las gráficas en las que se muestran los resultados, para cada uno de los cinco tipos de plazos de principio a fin posibles.

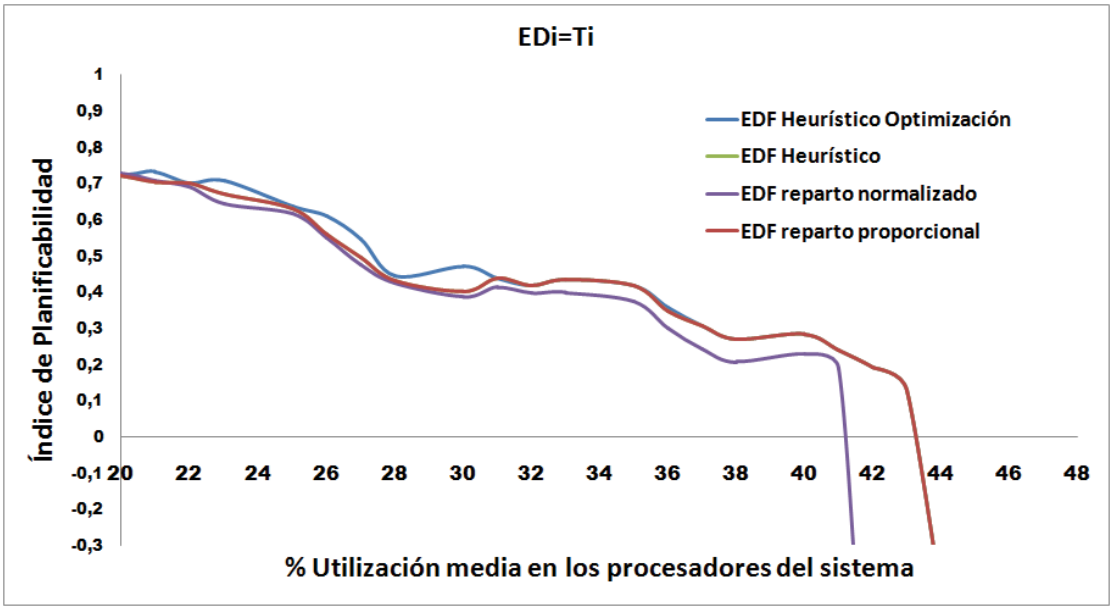


Figura 6.3: Gráfico para el caso $ED_{ij} = T_i$

6.1.1. Caso $ED_{ij} = T_i$

En la figura 6.3 se puede observar que el algoritmo heurístico no consigue hacer planificable el sistema por encima del 43 % de utilización media. El reparto proporcional consigue planificar el sistema hasta la misma carga media. Se observa que en esta gráfica, los índice de planificabilidad obtenidos por el algoritmo heurístico sin optimizar y el reparto proporcional son idénticos. Esto es debido a que el algoritmo heurístico consigue la planificabilidad en su primera iteración, en donde se ha realizado un reparto proporcional. Para los casos por debajo del 32 % de carga media, el algoritmo ha sido capaz de optimizar levemente sus soluciones.

El reparto normalizado tiene un rendimiento inferior, al ser capaz de planificar el sistema solo hasta el 41 %. En cualquier caso, la asignación $ED_{ij} = T_i$ provoca que el sistema esté limitado por sus plazos de principio a fin, no pudiendo conseguir utilizaciones elevadas bajo ningún tipo de asignación de plazos de planificación.

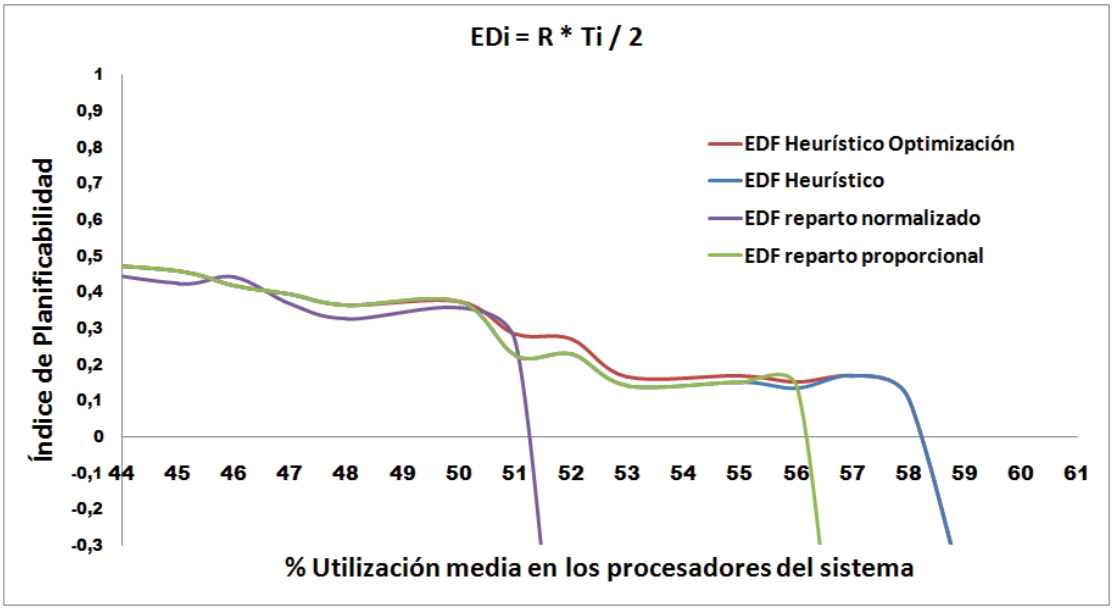


Figura 6.4: Gráfico para el caso $ED_{ij} = \frac{RT_i}{2}$

6.1.2. Caso $ED_{ij} = \frac{RT_i}{2}$

En este caso los plazos de principio a fin ya no son tan restrictivos como antes. En la figura 6.4 se observa como el algoritmo heurístico es capaz de planificar el sistema con un 2 % mas de utilización media con respecto al reparto proporcional, y un 7 % con respecto al reparto normalizado.

Por debajo del 55 % de carga media, el algoritmo heurístico y el reparto proporcional coinciden. Esto es debido a que el algoritmo heurístico encuentra la solución en su primera iteración, que es un reparto proporcional. Sin embargo, cuando se especifican sobreiteraciones, es capaz de optimizar levemente la solución cuando la utilización media está entre el 50 % y el 57 %.

6.1.3. Caso $ED_{ij} = RT_i$

Este caso posee plazos de principio a fin menos restrictivos que en el caso anterior. Observando la figura 6.5 se puede ver que el algoritmo heurístico ha sido capaz de planificar el sistema hasta un 19 % de utilización media por encima que el reparto proporcional, además de conseguir optimizar la solución a partir del

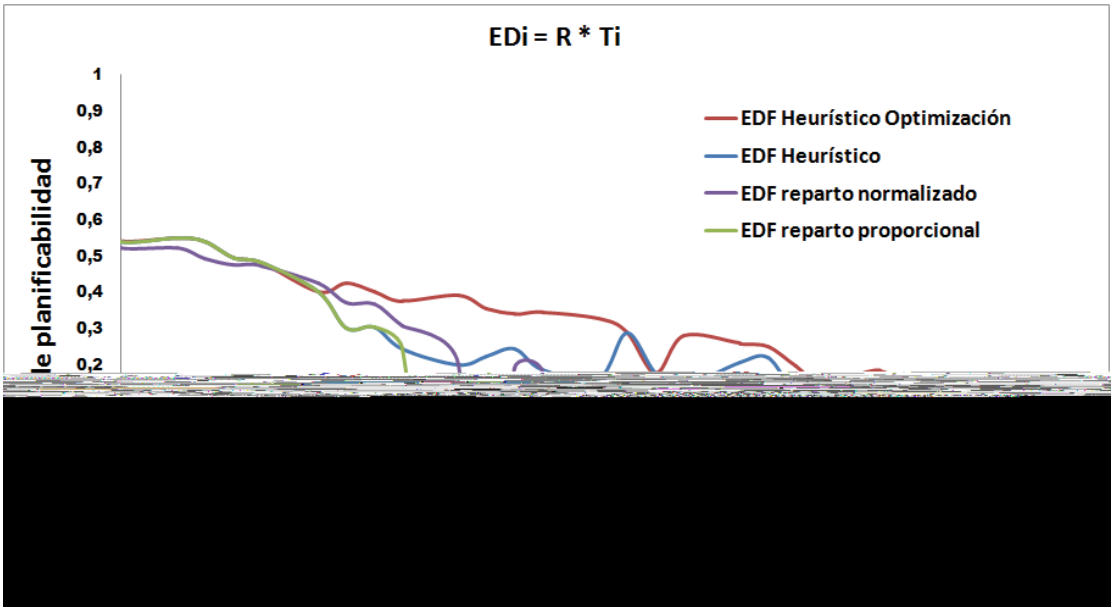


Figura 6.5: Gráfico para el caso $ED_{ij} = RT_i$

65 % de utilización media. El reparto normalizado en este caso se ha comportado mejor que la distribución proporcional. Cabe señalar que en un 71 % de carga media, el reparto normalizado no ha conseguido planificar el sistema, pero sí lo ha hecho en 72 % y 73 %, lo cual podría indicar algún tipo de patología en el sistema con un 71 % de carga media.

6.1.4. Caso $ED_{ij} = 2RT_i$

Éste es el caso menos restrictivo de todos, en el que se observan claramente las diferencias en el comportamiento de los 3 algoritmos. Viendo la figura 6.6 se puede observar como el algoritmo heurístico ha sido capaz de planificar el sistema hasta al menos un 96 % de utilización media. Es posible que hubiera conseguido la planificabilidad por encima de esta carga, pero el generador de ejemplos sólo genera sistemas con carga media de hasta un 96 %.

El reparto proporcional y proporcional normalizado consiguen la planificabilidad hasta que se alcanza un 83 y 85 % de utilización media respectivamente, por lo que el algoritmo heurístico sigue siendo claramente superior.

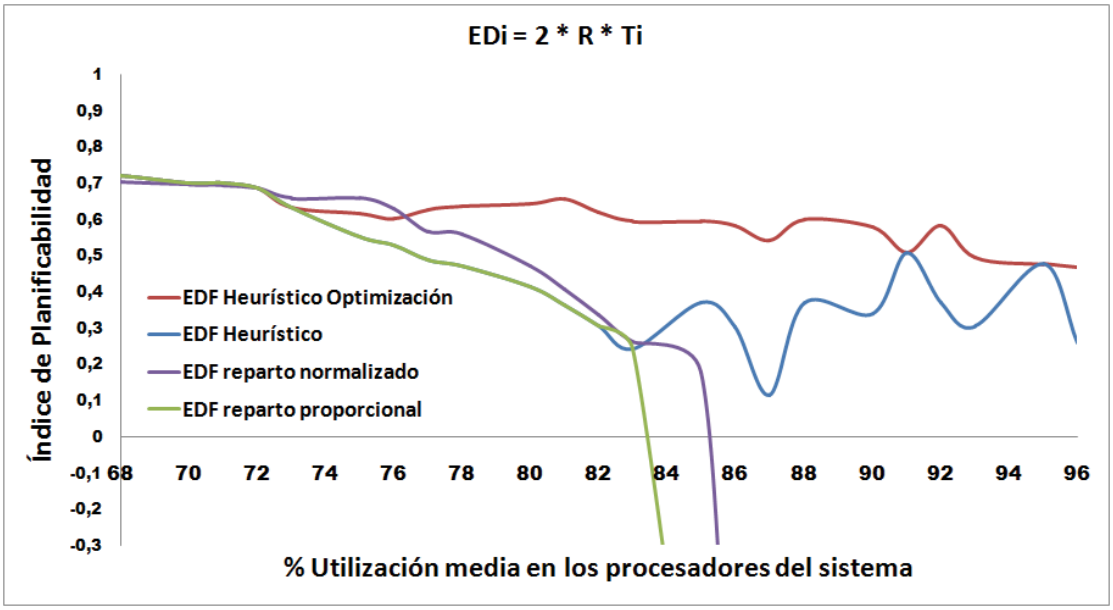


Figura 6.6: Gráfico para el caso $ED_{ij} = 2RT_i$

6.1.5. Caso $ED_{ij} =$ **random**$(T_i, 2RT_i)$

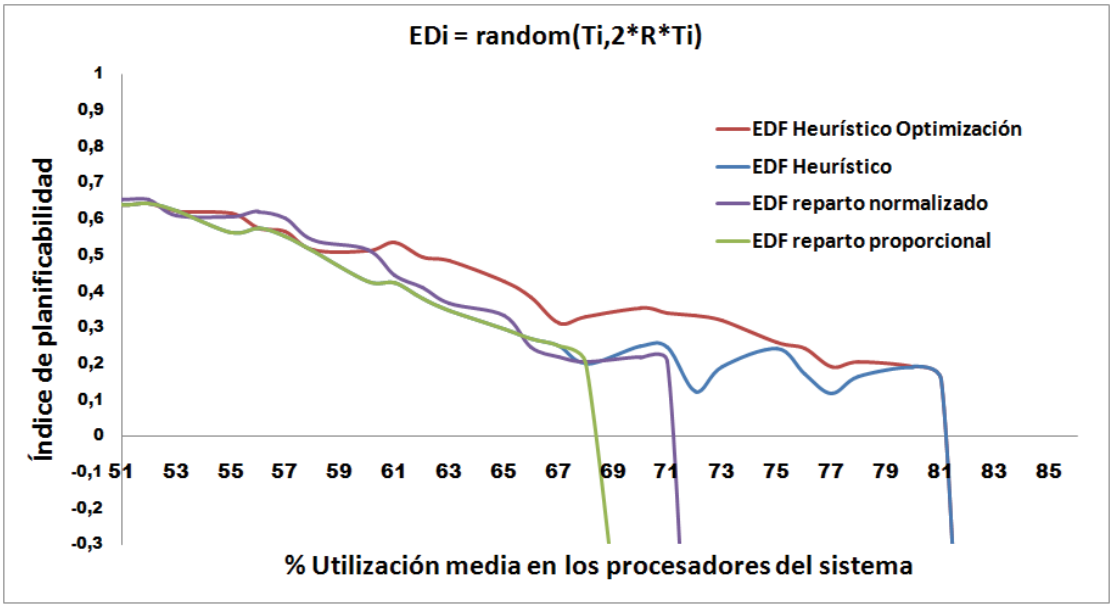


Figura 6.7: Gráfico para el caso $ED_{ij} = \text{random}(T_i, 2RT_i)$

Como los plazos de principio a fin en la realidad no tienen por qué tener una

relación directa con el periodo y la longitud de la transacción, el caso en el que los plazos de principio a fin sean aleatorios son una representación de muchos sistemas reales. El resultado que se observa en la figura 6.7 nos muestra que el algoritmo heurístico ha sido capaz de planificar el sistema con hasta un 10 % más de utilización media que el reparto normalizado, además de haber sido capaz de optimizar la solución notablemente desde prácticamente un 55 % de utilización media.

6.1.6. Análisis de los resultados obtenidos

A la vista de los resultados obtenidos en esta sección, podemos afirmar que el algoritmo heurístico que se ha propuesto obtiene mejores resultados que los repartos proporcionales y normalizados, pudiendo planificar sistemas con mayor carga en sus procesadores. Además, se ha comprobado la capacidad del algoritmo heurístico de optimizar sus soluciones.

6.2. Estudio comparativo entre el algoritmo heurístico de asignación de plazos de planificación y HOPA

En esta sección se va comparar el algoritmo heurístico de asignación de plazos de planificación para sistemas EDF propuesto, con el algoritmo HOPA para sistemas con prioridades fijas. Esta comparación resulta interesante porque los sistemas con prioridades fijas son ampliamente utilizados en los sistemas de tiempo real actuales.

Ambos algoritmos tienen la capacidad de optimizar sus soluciones, pero para simplificar la visualización de los resultados, y ya que el objetivo principal es comprobar con qué tipo de planificación se consiguen mejores utilizaciones, se va a utilizar el parámetro Sobreiteraciones=0.

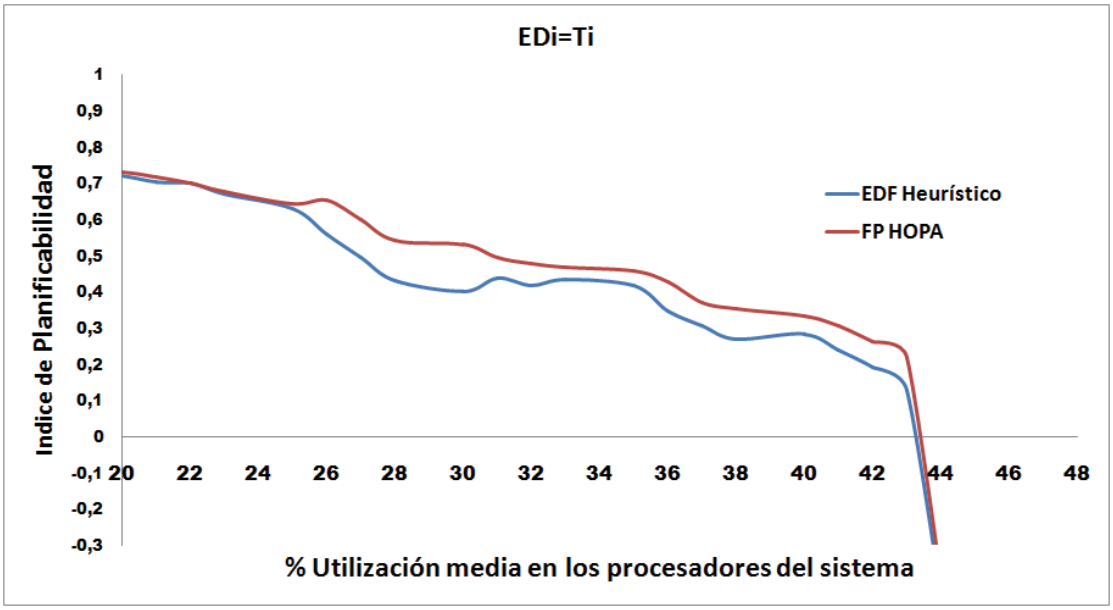


Figura 6.8: Gráfico para el caso $ED_{ij} = T_i$

6.2.1. Caso $ED_{ij} = T_i$

El caso $ED_{ij} = T_i$ es muy restrictivo, hecho que se comprueba viendo la figura 6.8, en donde ninguno de los dos algoritmos ha sido capaz de encontrar una solución por encima del 43 % de utilización media. Sin embargo, el algoritmo HOPA consiguió mejores soluciones (en el sentido del índice de planificabilidad obtenido)

6.2.2. Caso $ED_{ij} = \frac{RT_i}{2}$

En este caso menos restrictivo se empiezan a observar las primeras diferencias. Viendo la figura 6.9, el algoritmo de asignación de plazos consiguió hacer planificable el sistema con hasta un 58 % de utilización media, un 1 % por encima que HOPA.

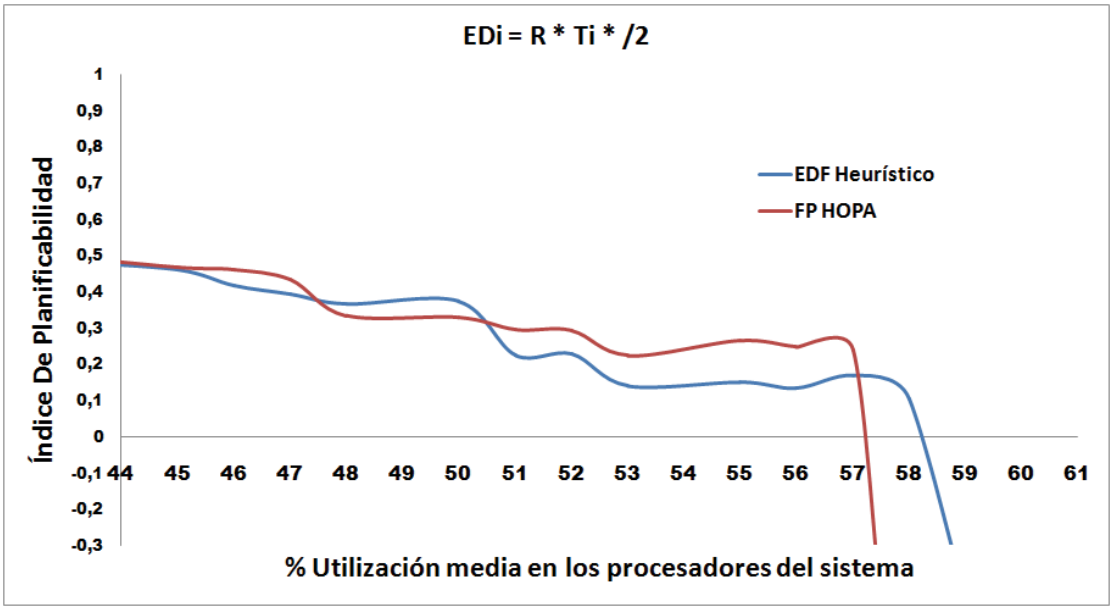


Figura 6.9: Gráfico para el caso $ED_{ij} = \frac{RT_i}{2}$

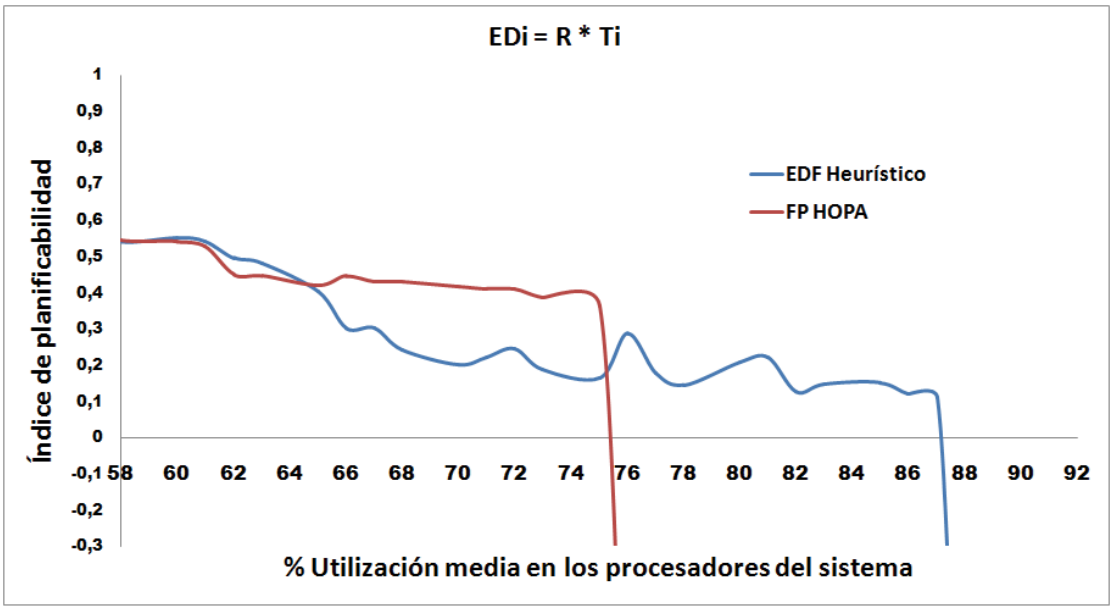


Figura 6.10: Gráfico para el caso $ED_{ij} = RT_i$

6.2.3. Caso $ED_{ij} = RT_i$

En la figura 6.10 se puede ver que asignando plazos de planificación con el algoritmo heurístico se consigue un 12 % más de utilización media que con el

caso de prioridades fijas.

6.2.4. Caso $ED_{ij} = 2RT_i$

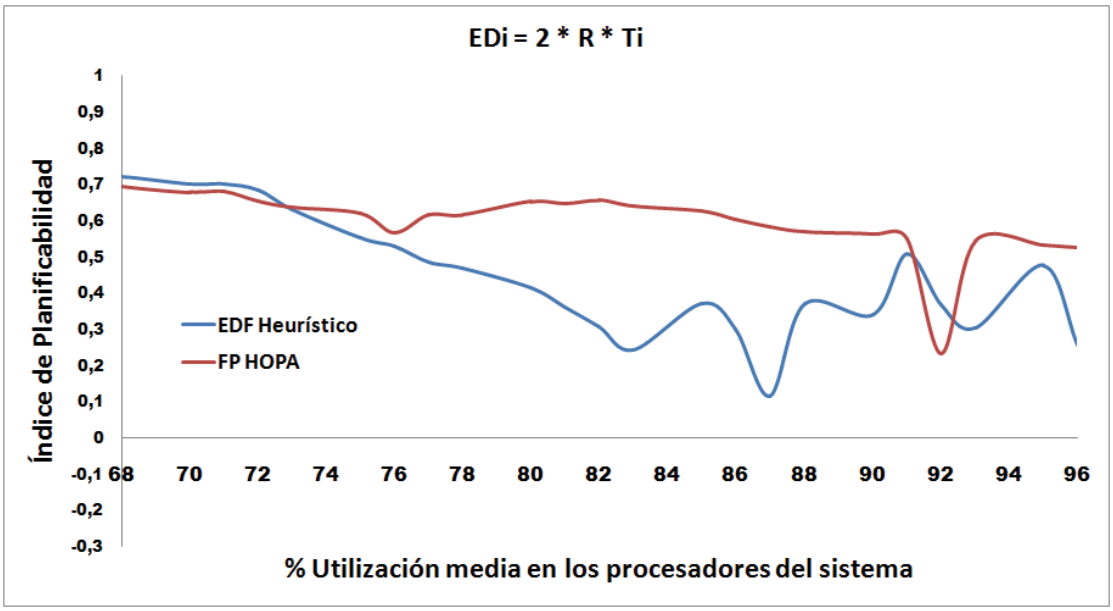


Figura 6.11: Gráfico para el caso $ED_{ij} = 2RT_i$

Este caso es el menos restrictivo en cuanto a plazos de principio a fin, hecho que se comprueba observando la figura 6.11, en donde ambos algoritmo consiguieron la planificabilidad hasta, al menos, un 96 % de utilización media. Se observa claramente que el algoritmo HOPA consiguió generalmente mejores soluciones, en el sentido del índice de planificabilidad.

6.2.5. Caso $ED_{ij} = \mathbf{random}(T_i, 2RT_i)$

Tal y como se ve en la figura 6.12, cuando se asignan plazos de principio a fin de forma aleatoria, el algoritmo heurístico de asignación de plazos ha conseguido planificar el sistema hasta el 81 % de utilización media, lo que representa un 8 % más que el caso de prioridades fijas.

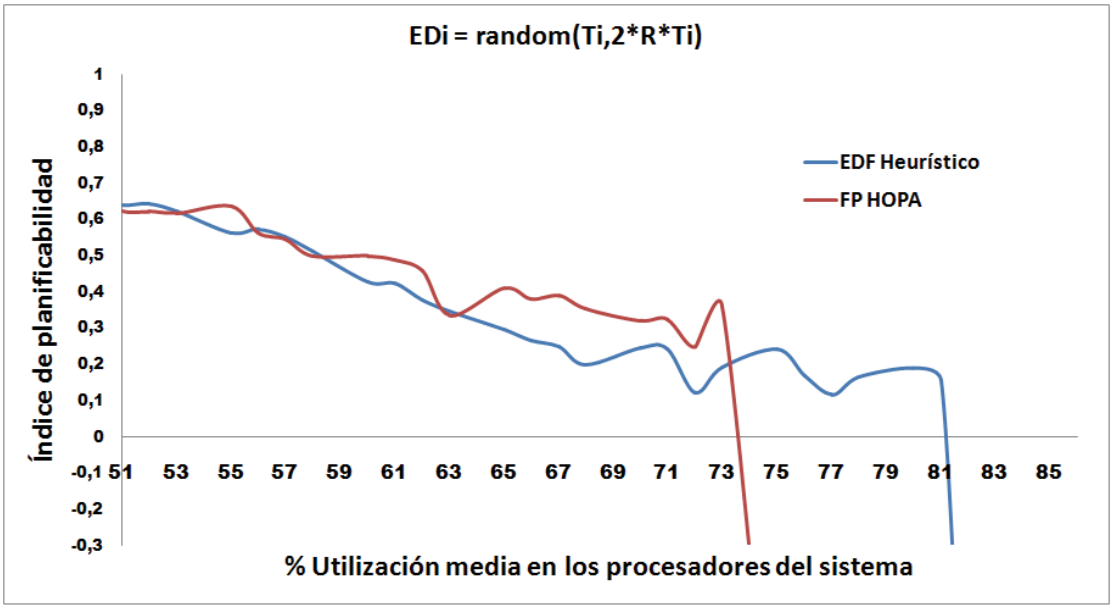


Figura 6.12: Gráfico para el caso $ED_{ij} = \text{random}(T_i, 2RT_i)$

6.2.6. Análisis de los resultados obtenidos

En los sistemas que se han analizado en esta sección, el algoritmo heurístico de asignación de plazos de planificación propuesto ha conseguido obtener mejores utilizaciones que el algoritmo HOPA para prioridades fijas, especialmente cuando los plazos no eran muy restrictivos.

Esta conclusión es importante, ya que en la actualidad la mayoría de los sistemas multiprocesadores de tiempo real están planificados con prioridades fijas, por su sencillez en comparación con la planificación EDF, y porque gracias a algoritmos de asignación de prioridades fijas como HOPA se podían conseguir altas utilizaciones en los procesadores. Mediante el algoritmo de asignación de plazos de planificación propuesto en este trabajo y utilizando sistemas planificados según EDF, podremos conseguir una mayor utilización de los recursos de los sistemas distribuidos de tiempo real.

Capítulo 7

Conclusiones

En la sección 1.6 planteamos los objetivos de este proyecto, que fuimos abordando en capítulos sucesivos. Con el estudio de las técnicas de asignación de parámetros de planificación expuestas en el capítulo 2, y basándonos en el algoritmo HOPA, en el capítulo 3 se propuso un algoritmo heurístico de asignación de plazos de planificación para sistemas distribuidos EDF.

Con el objetivo de poder evaluar y comparar los algoritmos de asignación de parámetros de planificación, se desarrolló una herramienta software que implementaba los algoritmos de asignación de plazos de planificación estudiados y propuestos. Esta herramienta se integró satisfactoriamente en el entorno MAST. Además, para poder analizar el funcionamiento de estos algoritmos sobre un número suficientemente elevado de casos, en el capítulo 5 se desarrolló una herramienta generadora automática de ejemplos.

De la comparación de los resultados que se exponen en el capítulo 6 se pueden extraer las siguientes conclusiones:

- Cuando se utilizan planificadores EDF, la técnica de asignación heurística de plazos de planificación propuesta en el capítulo 3 es capaz de conseguir mayores utilizaciones (en torno al 10 % dependiendo del caso) en los recursos procesadores que las técnicas de reparto proporcional y normalizado. Esta diferencia es tanto mayor cuanto mayores sean los plazos de principio a fin con respecto a los periodos de las transacciones.
- De igual manera, el algoritmo heurístico de asignación de plazos de pla-

nificación propuesto es capaz de conseguir mayores utilizaciones que el algoritmo HOPA para prioridades fijas.

- La técnica de asignación de plazos de planificación heurística propuesta tiene la capacidad de optimizar la solución obtenida.

Con estas conclusiones podemos afirmar que utilizando el algoritmo de asignación de plazos de planificación para sistemas distribuidos EDF propuesto, somos capaces de hacer un uso mas eficiente de los recursos procesadores. Esto nos permite, por ejemplo, la ejecución de software de tiempo real mas complejo, sin la necesidad de aumentar la potencia en los recursos procesadores, con el consiguiente ahorro de costes.

7.1. Trabajo futuro

La técnica de análisis de planificabilidad para sistemas distribuidos EDF implementada en la actualidad posee una serie de limitaciones, tales como un límite en el tamaño del sistema a analizar, o la carencia de soporte de sincronización entre tareas. Añadiendo el soporte a estas carencias en la herramientas de análisis EDF se podría estudiar el comportamiento del algoritmo de asignación de plazos de planificación propuesto en un mayor espectro de sistemas de tiempo real.

En este momento no existen redes planificadas según EDF, y por lo tanto, tampoco están implementadas en MAST. Un trabajo futuro podría ser el desarrollo de redes de comunicación planificadas según EDF, y la posterior incorporación de su soporte en MAST.

El algoritmo heurístico basa su funcionamiento en un parámetro que se llama *exceso*, tal y como se describió en 2.1 y 3. Existían dos definiciones para el *exceso*, pero en las implementaciones utilizadas se utiliza únicamente el *exceso de tiempo de respuesta*. El *exceso de tiempo de computación* resulta una forma más exacta de representar el concepto de *exceso*, pero debido a que su cálculo es computacionalmente muy costoso, se desechaba su uso en la implementación. Un trabajo futuro podría ser el uso del *exceso de tiempo de computación* para el cálculo del *exceso*. Para reducir en la medida de lo posible el tiempo de computación requerido para estos cálculos, se podría utilizar algún sistema de alto rendimiento o

cluster de computación, como los que existen en la actualidad en la Universidad de Cantabria.

Otro trabajo futuro sería el de adaptar las técnicas heurísticas de asignación de parámetros de planificación para dar soporte a sistemas mixtos, en los que conviven recursos procesadores planificados según EDF y otros planificados con prioridades fijas.

Bibliografía

[1] LIU J W. S "Real-Time Systems",Prentice Hall,2000

[2] STANKOVIC J.A. y RAMAMRITHAM K.: "Hard Real-Time Systems". IEEE Computer Society, catalog no. EH0276-6, 1988.

[3] J.J. GUTIÉRREZ GARCÍA and M. GONZÁLEZ HARBOUR (Director) "Planificación, análisis y optimización de sistemas distribuidos de tiempo real estricto." Tésis Doctoral, Universidad de Cantabria, 1995

[4] KLEIN M., RALYA T., POLLAK B., OBENZA R. y GONZALEZ HARBOUR M. "A Practitioner's handbook for Real-Time Analisys". Kluwer, Academic Pub., 1993.

[5] LIU C.L. y LAYLAND J.W.: "Scheduling Algorithms for Multi-Programming in a Hard Real-Time Environment". Journal of the Association for Computing Machinery, vol. 20, no. 1, pp. 46-61, January 1973.

[6] JOSEPH M. y PANDYA P.: "Finding Response Times in a Real-Time System". The Computer Journal (British Computing Society) 29, 5, pp. 390- 395, October 1986.

[7] AUDSLEY N.C., BURNS A., RICHARDSON M., TINDELL K. y WELLINGS A.J. "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling". Software Engineering Journal, Vol. 8, No. 5, pp. 284-292, September 1993.

[8] LEUNG J. y WHITEHEAD J. "On Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks". Performance Evaluation, 2, pp. 237-250, 1982.

[9] AUDSLEY N.C. "Optimal Priority Assignment and Feasibility of Static Priority Tasks With Arbitrary Start Times" Department of Computer Science, University of York, Technical Report YCS-164, December 1991.

[10] STROSNIDER J.K., MARCHOK T., LEHOCZKY J.P. "Advanced real-time scheduling using the IEEE 802.5 Token Ring", Proceedings of the IEEE Real-Time Systems Symposium, Huntsville, Alabama, USA, 1988, pp. 42-52.

[11] AGRAWAL G., CHEN B., ZHAO W., and DAVARI S. "Architecture Impact of FDDI Network on Scheduling Hard Real-Time Traffic". Workshop on Architectural Aspects of Real-Time Systems, December 1991.

[12] TINDELL K., BURNS A. y WELLINGS A.J. "Calculating Controller Area Network (CAN) Message Response Times". Proceedings 1994 IFAC workshop on Distributed Computer Control Systems (DCCS), Toledo, Spain, September 1994.

[13] IEEE Std. 1003.1b: "POSIX (Portable Operating System Interface)". IEEE Standard for Information Technology, 1993.

[14] TINDELL K. y CLARK J. "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems". Microprocessing and Microprogramming, Vol. 50, Nos. 2-3, pp. 117-134, April 1994.

[15] TINDELL K. "Adding Time-Offsets To Schedulability Analysis". Department of Computer Science, University of York, Technical Report YCS-221, January 1994.

[16] PALENCIA J.C., GONZÁLEZ HARBOUR M. "Schedulability Analysis for Tasks with Static and Dynamic Offsets" Proceedings of the 19th Real-Time Systems Symposium, IEEE Computer Society Press, pp 26-37, December 1998.

[17] M. Spuri. "Analysis of Deadline Scheduled Real-Time Systems". RR-2772, INRIA, France, 1996

[18] PALENCIA J.C., GONZÁLEZ HARBOUR M. "Offset-Based Response Time Analysis of Distributed Systems Scheduled under EDF" Proceedings of the 15th Euromicro Conference on Real-Time Systems, ECRTS, Porto, Portugal, July, 2003, ISBN:0-7695-1936-9, pp. 3-12.

[19] KIRKPATRICK S., GELATT C.D. y VECCHI M.P. "Optimization by Simulated Annealing". Science, vol. 220, no. 4598, pp. 671-680, May 1983.

[20] GUTIÉRREZ GARCÍA J.J, GONZALEZ HARBOUR M "Optimized Priority Assignment for Tasks and Messages in Distributed Hard Real-Time Systems" Proceedings of 3rd Workshop on Parallel and Distributed Real-Time Systems, IEEE Computer Society Press, pp. 124-132, April 1995.

[21] Dpto. de Electrónica y Computadores "MAST Description" http://mast.unican.es/mast-description.pdf