Programa Oficial de Postgrado en Ciencias, Tecnología y Computación
Máster en Computación
Facultad de Ciencias - Universidad de Cantabria

Algoritmo de asignación de plazos globales en sistemas distribuidos de tiempo real con planificación EDF : comparativa de estrategias de planificación

Juan María Rivas Concepción
rivasjm@unican.es

Director: J.Javier Gutiérrez García
Grupo de Computadores y Tiempo Real
Departamento de Electrónica y Computadores

Santander, Octubre de 2009
Curso 2008/2009

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.

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 en CTR (y en especial a Mónica, ¡gracias por haberlo hecho más ameno!), los cuales me han motivado a hacerlo lo mejor posible, con sus ejemplos de entrega y excelencia en las distintas áreas en las que trabajan.

Y a todos aquellos que olvido nombrar.

Tabla de contenidos

1. Introducción

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 [11], o dicho de otra forma, poseen unos requisitos temporales, llamados también **plazos**, que deben cumplir. Por lo tanto, una característica común y básica a todo sistema de tiempo real es la **predecibilidad** de su comportamiento, para poder asegurar de antemano el cumplimiento de los plazos impuestos.

Los sistemas de tiempo real están compuestos típicamente 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

Para poder determinar la validez (en el sentido del cumplimiento de los plazos) de un sistema de tiempo real hace falta un modelo abstracto que capture las características más relevantes de éste. Con el fin de poder caracterizar una amplia variedad de sistemas de tiempo real, que poseen diversos tipos de carga en computación y comunicación, se definirán con términos generales.

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 un sistema es capaz de cumplir todos sus plazos siempre, se dice que es un **sistema planificable.** 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.

Llamamos **tarea** a cada unidad mínima de trabajo que es planificado y ejecutado por el sistema. Todas las tareas se ejecutan en algún recurso procesador. Estos recursos será principalmente procesadores y redes de comunicación.

Una tarea se dice que ha sido activada cuando está lista para la ejecución. En el modelo que se utilizará en este trabajo, todas las tareas que se ejecutan en el sistema son activadas mediante un estímulo llamado **evento**. Estos eventos típicamente pueden llegar al sistema de forma periódica

o aperiódica. A las tareas que se ejecutan como consecuencia de la llegada de un evento también se les conoce como respuesta al evento.

Los eventos que activan las tareas pueden tener impuestos unos plazos que el sistema debe cumplir para su correcto funcionamiento, que típicamente hacen referencia al máximo tiempo del que dispone la tarea para finalizar desde que fue activada. De forma similar, el tiempo de respuesta de una tarea se define como la cantidad de tiempo que transcurre desde que la tarea fue activada, hasta que finalizó su ejecución. Por lo tanto, si el tiempo de respuesta de una tarea no es mayor que su plazo, se dice que ésta es planificable**.**

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

1.1.1. Modelo lineal para sistemas 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 tareas concurrentes que se ejecutan en algún recurso del sistema. Cada tarea puede comunicarse con otras tareas mediante el envío de mensajes a través de una red de comunicación. En este trabajo no se hará distinción entre la ejecución de una tarea en un procesador o la transmisión de un mensaje por una red de comunicación, en ambos casos se dirá que se está ejecutando una tarea en un **recurso procesador**.

Las tareas se activan mediante la llegada de un evento. Los eventos pueden ser **externos**, como por ejemplo un reloj periódico; o pueden ser **internos**, resultado de la llegada de un mensaje proveniente de otra tarea. Este esquema de activaciones acarrea una relación de precedencia en la ejecución de las tareas. A la sucesión de tareas con una relación de precedencia le llamaremos **transacción.** La primera tarea de la transacción es activada mediante la llegada de un evento externo, y las sucesivas tareas mediante eventos internos.

Llamamos e_i al evento externo que activa a la tarea a_{i1} (primera tarea de la transacción). Al periodo del evento externo e_i lo denotaremos T_i. Llamamos e_{jk} al evento interno que produce la tarea a_{ij} para activar a la tarea a_{ik}. A la serie de tareas 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 tarea a_{ij} con respecto a la llegada del evento externo e_i. El **plazo local** d_{ij} es el plazo que posee la tarea a_{ij} con respecto a su propia activación (llegada del evento interno o externo que activó directamente a a_{ij}).

Por último, se define el **plazo de principio a fin** ED_{ij} como el plazo global de la última tarea de la transacción *i*. En la figura 1.1 se muestran los tipos de plazos sobre un ejemplo concreto

.

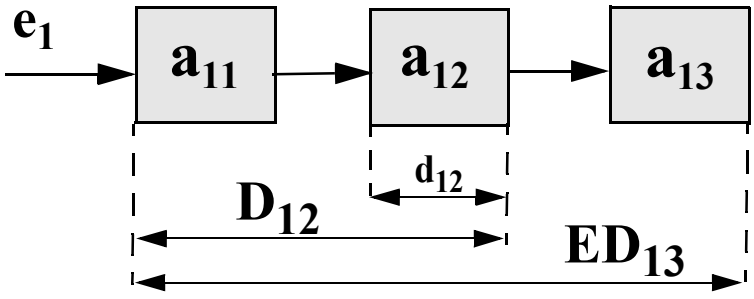

Figura 1.1: Modelo lineal

De manera similar a los plazos, se definen los tiempos de respuesta de peor caso. Así, R_{ij} es el **tiempo de respuesta global** de peor caso de la tarea a_{ij}, con respecto a la llegada del evento externo e_i. Llamamos $\mathbf{R_i}$ al tiempo de respuesta global de peor caso de la última tarea de la respuesta al evento $\mathbf{e_i}$, o dicho de otra forma, el tiempo de respuesta de la transacción *i*. El tiempo de ejecución de peor caso de la tarea a_{ij} se denota con C_{ij}.

Para un recurso procesador determinado, se define su **utilización** como:

$$U = \sum \frac{C_{ij}}{T_i}, \text{ para toda tarea situada en el recurso procesador}$$

La **utilización media** del sistema será la media de las utilizaciones de todos los procesadores que lo forman.

Cabe destacar que aunque el modelo lineal definido aquí nos sea suficiente para el desarrollo de esta Tesis, presenta una serie de limitaciones, tales como que una tarea sólo pueda ser activada por un único evento, y que solamente pueda generar un evento de salida; o que no se permita la sincronización entre tareas. Existen modelos generales que permiten definir sistemas distribuidos que no tienen éstas limitaciones [2], pero que sobrepasan el ámbito de este trabajo.

1.2. Políticas de Planificación

Cada recurso procesador del sistema va a tener que ejecutar una serie de tareas de forma concurrente. Es por ello necesario la utilización de un algoritmo o **política de planificación** que seleccione en cada momento qué tarea ejecutar de entre todas las tareas 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 [11]. En esta aproximación, a cada una de las tareas concurrentes que se desean ejecutar se les asigna una rodaja temporal durante la cual pueden ejecutarse. 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 una modificación en el tiempo de ejecución de cualquier 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 tarea 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 tarea con mayor prioridad de entre las tareas activas. El concepto de prioridad se puede implementar mediante una cola en la que se sitúan las tareas. Dependiendo de la prioridad asignada a cada una, se situarán en un lugar u otro de la cola. Usando esta política, los recursos procesadores nunca quedan inactivos si hay tareas listas para su ejecución [1].

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

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

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 ocurre con la planificación EDF (**Earliest Deadline First**).

1.3. Análisis de Planificabilidad para sistemas monoprocesadores

Las herramientas de análisis de planificabilidad nos permiten determinar la planificabilidad de un sistema de tiempo real, por lo tanto son de vital importancia en el diseño de este tipo de sistemas. En esta sección se describirán brevemente para el caso de sistemas monoprocesadores, tanto planificados por prioridades fijas o dinámicas (EDF)

Una característica importante de un algoritmo de planificación es su capacidad de aportar soluciones óptimas. Una política de planificación se dice que es **óptima** si siempre es capaz de planificar un sistema que tiene la capacidad ser planificado. Dicho de otra forma, si un algoritmo óptimo no es capaz de planificar un sistema, ningún otro algoritmo podrá conseguirlo.

1.3.1. Sistemas con prioridades fijas: Teoría RMA

Los sistemas monoprocesadores de tiempo real con tareas periódicas y planificador expulsor por prioridades fijas son ampliamente utilizados, ya que existen extensivos trabajos previos que

permiten el análisis de dichos sistemas [13], 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).

Un algoritmo de prioridades fijas ampliamente utilizado es el algoritmo *Rate-Monotonic* (RM) [3], en donde a las tareas se les asignan prioridades fijas de acuerdo a sus periodos : las tareas con menores periodos tienen mayor prioridad. Liu y Layland demostraron [3] que la asignación Rate-Monotonic es óptima entre los algoritmos de asignación de prioridades fijas cuando los plazos de las tareas son iguales a sus respectivos periodos.

Otro algoritmo de asignación de prioridades fijas bien conocido es el algoritmo *Deadline-Monotonic* (DM) [3], con el que se asignan las prioridades fijas de acuerdo a los plazos : las tareas con menores plazos tienen mayor prioridad. El algoritmo *Deadline-Monotonic* es óptimo entre los algoritmos de prioridades fijas cuando los plazos de las tareas son iguales o menores que sus respectivos periodos [3].

Tanto para el algoritmo RM como para DM se definen sus límites de utilización [1]. El límite de utilización de una política de planificación se define como la utilización máxima por debajo de la cual se puede asegurar la planificabilidad del sistema. Por lo tanto, cuanto mayor sea este límite, mejor aprovechamiento de los recursos se hará con la política de planificación dada. En cualquier caso, ningún algoritmo puede planificar un sistema con una utilización superior a 1 (100%, carga total).

El no sobrepasar el límite de utilización es una condición suficiente pero no necesaria para determinar la planificabilidad. Si un sistema dado posee una utilización superior al límite de utilización, hay que utilizar otra técnica para determinar la planificabilidad, como por ejemplo, el test exacto**.**

Con el test exacto, o test de tiempos de respuesta, se calculan de forma exacta los tiempos de respuesta de peor caso de todas las tareas que forman parte del sistema [1]. Si éstos son menores que sus respectivos plazos impuestos, se puede determinar que el sistema es planificable.

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 esta Tesis. La teoría RMA se puede extender para tenerlos en cuenta. A continuación se citan los más importantes:

- Efecto del **cambio de contexto**, que es el tiempo que necesita el sistema operativo para expulsar a una tarea y comenzar la ejecución de la siguiente.
- **Inversión de prioridad**, que es el efecto producido cuando una tarea es retrasada por otra de menor prioridad. Puede ocurrir, entre otras razones, cuando hay regiones de código no expulsables, colas FIFO, o las tareas compartan recursos. El modelado de la inversión de prioridad de lleva a cabo añadiendo tiempos de bloqueo en las ecuaciones para el cálculo del tiempo de respuesta.
- **Eventos aperiódicos**. Puede ocurrir que los eventos que activan las tareas no tengan naturaleza periódica. Se pueden tratar mediante técnicas como la del *Polling* periódico, o el servidor esporádico, con las que se puede tratar a la tarea aperiódica como si fuese periódica.
- **Plazos superiores al periodo**. Como los tiempos de respuesta pueden ser mayores que el periodo, puede existir más de una activación pendiente de una tarea. Ni la asignación

RM ni DM de prioridades es óptima en este caso. Audsley [14] propuso un algoritmo que es óptimo en la asignación de prioridades para plazos arbitrarios.

1.3.2. Sistemas con prioridades dinámicas: EDF

Los sistemas planificados por prioridades fijas nos permiten 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. Para explicar el funcionamiento de la planificación EDF se definen los siguientes conceptos:

- $\mathbf{t_{ik}}$, que hace referencia al instante de tiempo absoluto en el que la tarea a_i tiene su k-ésima activación.
- $\mathbf{d_{ik}}$, plazo absoluto de la k-ésima activación de la tarea a_i, o lo que es lo mismo, $t_{ik}+D_i$, siendo D_i el plazo de la tarea en cuestión.

En cada activación de la tarea a_i se calcula su prioridad de forma inversamente proporcional a su plazo absoluto d_{ik}, y esta prioridad de mantiene constante hasta la siguiente activación de a_i.

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. El límite de utilización bajo las condiciones descritas es de 1 (100%).

Si alguna tarea tuviera un plazo menor que su periodo, se define el siguiente test pesimista:

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

Si el test anterior se cumple, se puede asegurar la planificabilidad. Si no se cumple, y la utilización del sistema es menor que el 100%, hay que utilizar un test exacto para determinar la planificabilidad [1].

Aunque la planificación por prioridades fijas goza de más aceptación en la actualidad, el uso de prioridades dinámicas EDF está comenzando a cobrar más relevancia, debido principalmente al mejor aprovechamiento de los recursos de procesamiento que consigue. EDF está ahora disponible en lenguajes tales como Ada 2005 [20] o Java (RTSJ) [21], o en sistemas operativos de tiempo real como S.Ha.R.K [22] , ERIKA [23] u OSEK/VDX (a nivel de aplicación) [24]. Desgraciadamente, la planificación EDF 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 más "prioritarias", aumentando así aún más la complejidad del sistema.

1.4. Análisis de planificabilidad para Sistemas Distribuidos

De manera similar a lo que ocurría para sistemas monoprocesadores, las herramientas de análisis para sistemas distribuidos nos permiten calcular los tiempos de respuesta de peor caso de cada tarea, y de toda la transacción en su conjunto, para así poder determinar si se cumplen los plazos establecidos. Analizaremos sistemas definidos según el modelo lineal descrito en 1.1.1.

Para el caso de planificación bajo prioridades fijas existen trabajos en los que se utilizan algunas redes de comunicación estándares para llevar a cabo comunicaciones de tiempo real estricto, tales como el bus CAN [16], o el protocolo Ethernet para tiempo real RT-EP [15]. Trabajos adicionales [2] desarrollan una herramienta de software basada en la interfaz de colas de mensajes definida en el POSIX.1b [17], 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, los mensajes transmitidos en las redes se tratarán de la misma manera que 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 la planificación con prioridades dinámicas en sistemas distribuidos se hará la misma similitud red-procesador. El uso de la planificación EDF en redes de comunicación no está muy extendido en la actualidad, pero sí ha sido motivo de estudio, tales como su aplicación en redes de propósito general [25] o en el bus CAN [26].

1.4.1. Sistemas distribuidos planificados con prioridades fijas : Técnica Holística

La primera tarea 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 tareas sucesivas de la transacción depende del instante de finalización de la tarea precedente, o lo que es lo mismo, del tiempo de respuesta global de la tarea precedente, y éste no es típicamente un valor constante. Por lo tanto, todas las tareas de una transacción dada, salvo la primera, pueden tener activaciones no perfectamente periódicas, también llamadas **activaciones retrasadas**. A este efecto se le conoce como ***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.

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

$$J_{ik} = R_{ik-1} - \sum_{l \in M_{ik-1}} b_{il}$$

Ecuacion 1.1: Jitter de una tarea

Dónde b_{il} es el tiempo de ejecución de mejor caso de la tarea a_{il}, y M_{ik-1} es el grupo de tareas de la respuesta al evento e_i que preceden a la tarea a_{ik}.

Como se puede observar en la ecuación 1.1, el cálculo del *jitter* para las tareas de un recurso depende de los tiempos de respuesta de otras tareas que pueden estar en otros recursos. A su vez, el cálculo de los tiempos de respuesta dependen del *jitter*.

Una primera aproximación a este problema es aquella que considera a cada recurso del sistema (procesadores y redes) como independientes. Esta visión 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, este 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 impuestos.

En el análisis holístico se aplican las técnicas RMA sobre cada recurso, ya que se han considerado independientes. La interdependencia *jitter*-tiempo de respuesta la soluciona Tindell [4] 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, siempre que la utilización en los procesadores no sea superior al 100%.

1.4.1.1. Análisis mediante *offsets* para sistemas con prioridades fijas

Como ya se ha mencionado, el análisis holístico 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 tareas 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 [5] 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 tarea se define como la cantidad de tiempo que transcurre desde que el evento externo activa la transacción, hasta que la tarea dada es activada.

El problema asociado al análisis propuesto es que se vuelve computacionalmente intratable a medida que crece el número de tareas, 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.

En el mismo trabajo [5] 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.

El análisis mediante offsets fue extendido por J.C. Palencia y M.G. Harbour [6], añadiendo la capacidad a los offsets de ser dinámicos. El uso de offsets dinámicos es necesario para poder analizar sistemas distribuidos, ya que en éstos, el instante de activación de las tareas puede variar de un periodo a otro.

En los resultados expuestos en [6], se observa la relación entre los tiempos de respuesta obtenidos por la técnica holística y la basada en *offsets*. Se puede comprobar que los tiempos de respuesta obtenidos por la técnica holística son en media el doble que los obtenidos por la técnica basada en *offsets*. Con esta comparación se quiere dejar patente la importancia del nivel de pesimismo que puede introducir la técnica de análisis que se use, ya que unos resultados excesivamente pesimistas pueden provocar que no se maximice la utilización de los recursos disponibles en el sistema bajo análisis.

1.4.2. Sistemas distribuidos planificados con prioridades dinámicas EDF

Partiremos de un sistema definido según el modelo lineal. Para el análisis de sistemas distribuidos EDF se distinguirán dos casos : aquellos en los que las tareas se planifican con su plazo local (d_{ij}), y aquellos en los que se planifica con su plazo global (D_{ij}).

En este trabajo se usarán las técnicas holísticas para el cálculo de los tiempos de respuesta para sistemas distribuidos EDF, tanto para plazos locales como globales, a pesar de la existencia de técnicas menos pesimistas basadas en *offsets*. Se ha tomado esta decisión debido a que actualmente no existe ningún trabajo que adapte la técnica de *offsets* para planificación con plazos locales. Para que la comparación entre la planificación con plazos globales y locales que se realizará más adelante sea más justa, en el sentido del pesimismo de los resultados, en ambos casos se aplicará la técnica holística.

1.4.2.1. Planificación con plazos globales

El análisis, propuesto por Spuri [7], y modificado por Palencia [8], se basa en la creación del periodo de ocupación más largo posible. En EDF,el **periodo de ocupación** hace referencia al intervalo de tiempo en el cual el procesador está ocupado procesando ejecuciones pendientes de alguna tarea. El tiempo de respuesta de peor caso de la tarea τ_a se encuentra en un periodo de ocupación en el que el resto de las tareas (y quizás la propia tarea bajo análisis) se activan simultáneamente (en el comienzo del periodo de ocupación).

El primer paso para el cálculo del tiempo de respuesta de peor caso de la tarea τ_a es el cálculo de la contribución de peor caso de una tarea τ_i al periodo de ocupación de longitud *t,* cuando el plazo de τ_a ocurre en el instante D. Llamaremos a esta contribución $W_i(t, D)$:

$$W_i(t, D) = min\left(\left\lceil \frac{t + J_i}{T_i} \right\rceil, \left\lfloor \frac{J_i + D - d_i}{T_i} \right\rfloor + 1\right)_0 C_i$$

La función "$min()_0$" devuelve 0 para valores negativos en su entrada.

Para calcular el tiempo de respuesta de peor caso, necesitamos saber en qué instante del periodo de ocupación ocurre el **instante crítico**. El instante crítico ocurre o bien al comienzo del periodo de ocupación, o en un instante en el que el plazo de la tarea bajo análisis τ_a coincide con el plazo de una tarea τ_i. A este conjunto de instantes lo llamaremos ψ:

$$\psi = \left(\bigcup \{ (p-1)T_i + d_i \} \right) \qquad \forall p = 1 \ldots \left\lceil \frac{L + J_i}{T_i} \right\rceil$$

L representa el periodo de ocupación más largo:

$$L = \sum_{\forall i} \left\lceil \frac{L + J_i}{T_i} \right\rceil \cdot C_i$$

Se observa que *L* aparece en ambos lados de la ecuación. Para resolverlo, se puede comenzar con un valor pequeño para *L*, e ir aplicando iterativamente la fórmula hasta que se obtenga una solución estable para *L*.

Podemos encontrar el instante crítico que provoca el tiempo de respuesta de peor caso probando todos los posibles instantes de ψ. Como puede haber más de una activación de τ_a en el periodo de ocupación, tenemos que analizarlas todas. Si la primera activación de τ_a ocurre en el instante *A* después del comienzo del periodo de ocupación, el tiempo para completar la ejecución *p*-ésima de τ_a, $w_a^A(p)$, se puede calcular de la siguiente forma :

$$w_a^A(p) = pC_a + \sum_{\forall (i \neq a)} W_i(w_a^A(p), D^A(p))$$

Donde $D^A(p)$ es el plazo de la activación *p*-ésima de la tarea τ_a, activada por primera vez en el instante *A*:

$$D^A(p) = A - J_a + (p-1)T_a + d_a$$

El tiempo de respuesta se puede calcular restando el instante de activación con el tiempo para completar la ejecución calculado:,

$$R^A(p) = w^A(p) - A + J_a - (p-1)T_a$$

Para cada *p*, sólo necesitamos comprobar los valores de *A* en el periodo (entre 0 y T_a). Por lo tanto, los valores de ψ a comprobar son :

$$\Psi^* = \{\Psi_x \in \Psi | (p-1)T_a - J_a + d_a \leq \Psi_x \leq pT_a - J_a + d_a\}$$
$$A = \Psi_x - [(p-1)T_a - J_a + d_a]$$

Para calcular el tiempo de respuesta de peor caso de la tarea τ_a hay que calcular el tiempo de respuesta que se obtendría con todos los instantes críticos posibles, y quedarse con el valor máximo obtenido:

$$R_a = max(R_a^A(p)) \quad \forall p = 1 \ldots \left(\left\lceil \frac{L + J_a}{T_a} \right\rceil \right), \quad \forall A \in \Psi_x$$

1.4.2.2. Planificación con plazos locales

En [19] se adaptó el algoritmo descrito en 1.4.2.1. para el caso de sistemas distribuidos EDF planificados por plazos locales. Ambas técnicas se basan en el mismo concepto de periodo de ocupación e instante crítico. La formulación se adapta para tener en cuenta que los plazos de planificación hacen referencia al instante de activación de la propia tarea, y no de la transacción, como ocurría en el caso de utilización de plazos globales.

Esta técnica de cálculo de tiempos de respuesta para plazos locales permite para llevar a cabo la implementación y evaluación de las técnicas de asignación de plazos locales de planificación que se expondrán en la sección 1.5.2.

1.5. Técnicas de Asignación de Parámetros de Planificación

Hay dos problemas fundamentales que presentan los sistemas de tiempo real: el poder determinar la planificabilidad de un sistema dado, y el de encontrar una asignación de parámetros de planificación en las tareas que consiga hacer el sistema planificable.

Para sistemas monoprocesadores con prioridades fijas, el objetivo es que cada tarea tenga asignado un valor numérico que represente su prioridad fija, que utilizará el planificador a la hora de tomar su decisión. En la sección 1.3 se presentaron los algoritmos RM y DM.

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 que se utilizan para planificar se les conoce como **plazos de planificación**, SD_{ij}. Cuando la respuesta está compuesta de únicamente una tarea (caso típico en sistemas monoprocesadores), el cálculo del plazo de planificación es trivial, no es más que $\mathbf{SD_i = d_i}$.

1.5.1. Asignación de prioridades fijas en sistemas distribuidos

Tindell, Burns y Wellings [12] 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).

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 [9] 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 funcionamiento del algoritmo HOPA se puede resumir con el siguiente pseudocódigo:

```
Algoritmo HOPA is
begin
   Distribuye plazos ED entre las tareas
   loop
      Asigna prioridades fijas Deadline-Monotonic
      Calcula tiempos de respuesta
      exit when se cumple algun requisito de parada
      Calcular nuevos plazos locales
   end loop
end HOPA
```

- La distribución inicial de los plazos ED se hace de forma proporcional a los tiempos de ejecución de cada tarea.

$$d_{ij} = \frac{\mathrm{ED}_i \cdot C_{ij}}{\sum_{\forall j} C_{ij}}$$

- Se asignan prioridades fijas *Deadline Monotonic* utilizando los plazos calculados anteriormente
- Se calculan los tiempos de respuesta mediante alguna técnica de análisis, como por ejemplo el análisis holístico o de *offsets* para prioridades fijas.
- El algoritmo HOPA se detiene cuando se cumple alguna de las siguientes situaciones :
  - En dos iteraciones consecutivas se obtiene la misma asignación de prioridades, ya que en ese caso se obtendría la misma asignación indefinidamente.
  - Se consigue una asignación que hace al sistema planificable.
  - Se supera un número máximo de iteraciones preestablecido.
- Utilizando los tiempos de respuesta calculados, y la "distancia" a la planificabilidad que tiene cada tarea, se calculan nuevos plazos locales "artificiales". El resultado de este cálculo se puede controlar mediante dos parámetros, k_r y k_a, que controlan el peso que tiene en el resultado final la "distancia" a la planificabilidad que tiene el procesador y la transacción respectivamente.

Una vez que HOPA encuentra una solución que hace al sistema planificable, el algoritmo tiene la capacidad de optimizar la solución (en el sentido de la holgura de los tiempos de respuesta

con respecto a los plazos). Para aprovechar esta característica, se puede añadir el siguiente criterio de parada a los ya establecidos:

- Se supera el número máximo de sobreiteraciones sobre una solución ya planificable.

1.5.2. Asignación de plazos locales de planificación en sistemas distribuidos

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}. Para poder ser planificadas con el algoritmo EDF, cada tarea debe tener asignado un **plazo local de planificación** (SD_{ij}). Cabe destacar que en un sistema distribuido EDF, los plazos de planificación no son plazos 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 en el modelo lineal : plazos de principio a fin ED_{ij}, los plazos globales D_{ij}, o los plazos locales d_{ij}.

A continuación se presentan tres algoritmos , dos 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, y un tercero basado en HOPA [18][20].

1.5.2.1. Reparto Proporcional del Plazo ED

El algoritmo, al que llamaremos $\mathbf{PD_{local}}$, distribuye el plazo de principio a fin ED_{ik} de forma proporcional a los tiempos de ejecución de peor caso de cada tarea de la respuesta:

$$SD_{ij} = \mathrm{ED}_{ik} \cdot \frac{C_{ij}}{\sum_{\forall j} C_{ij}}$$

Ecuacion 1.2: Asignación proporcional de plazos

1.5.2.2. Reparto Proporcional Normalizado del Plazo ED

El algoritmo, al que llamaremos $\mathbf{NPD_{local}}$, es similar al reparto proporcional del plazo, pero teniendo en cuenta además la carga en cada procesador. Se define según la siguiente ecuación:

$$\mathrm{SD}_{ij} = \mathrm{ED}_{ik} \cdot \frac{C_{ij} \cdot U(V_{ij})}{\sum_{l=1}^{k} C_{il} \cdot U(V_{il})}$$

Ecuacion 1.3: Asignación normalizada de plazos

El sumatorio $\sum_{l=1}^{k} C_{il} \cdot U(V_{il})$ recorre todas las tareas de la transacción en la que reside a_{ij}.

$U(V_{ij})$ es la utilización del procesador sobre el que se está ejecutando la tarea a_{ij}. 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. Los resultados mostrados en [18] muestran que utilizando la asignación de plazos normalizada no siempre se obtienen mejores resultados que con la asignación proporcional.

1.5.2.3. Algoritmo heurístico de asignación de plazos locales de planificación

Los algoritmos de reparto proporcional y normalizado calculan los plazos de las tareas del sistema de una manera relativamente sencilla, pero si la asignación que obtienen no consigue hacer al sistema planificable, no pueden aportar otra solución.

El algoritmo HOPA, en cambio, implementa mecanismos que le permiten la búsqueda de una solución si ésta aún no ha sido encontrada. Para sacar provecho de este característica se adaptó el algoritmo HOPA al problema de la asignación de plazos locales de planificación [18][19]. Al algoritmo resultante lo llamaremos **$\mathbf{HOSDA_{local}}$**.

El esquema de funcionamiento del algoritmo $HOSDA_{local}$ es similar al del algoritmo HOPA, y se puede describir con el siguiente pseudocódigo:

```
Algoritmo Local_HOSDA is
begin
   Distribuye plazos ED entre las tareas
   loop
      Calcula tiempos de respuesta
      exit when se cumple algun requisito de parada
      Calcular nuevos plazos locales de planificación
   end loop
end Local_HOSDA
```

Cada paso del algoritmo es similar al descrito para HOPA:

- El reparto inicial de los plazos ED es similar al realizado en HOPA.
- El cálculo del tiempo de respuesta se hace con alguna técnica de análisis de sistemas ditribuidos EDF planificados con plazos locales.
- Los criterios de parada son:
  - En dos iteraciones consecutivas se obtiene la misma asignación de plazos, ya que en ese caso, en posteriores iteraciones se obtendría la misma asignación indefinidamente.
  - Se consigue una asignación que hace al sistema planificable.
  - Se supera el número máximo de iteraciones preestablecidas.
- Se supera el número máximo de sobreiteraciones sobre una solución ya válida. Esto es así porque el algoritmo $HOSDA_{global}$ también posee la capacidad de optimizar (en el sentido del índice de planificabilidad) una solución ya planificable.
- El cálculo de los nuevos plazos locales se realiza de forma similar a HOPA. La diferencia radica en que en HOPA se utilizaban estos plazos para calcular las nuevas prioridades fijas bajo el criterio *Deadline Monotonic*, y en HOSDA estos plazos son los nuevos plazos de planificación SD_{ij}.

1.6. Objetivos del Proyecto

Se han introducido los conceptos básicos que rigen los sistemas de tiempo real, entre ellos, algunos algoritmos de planificación y técnicas de análisis. Se ha presentado además el problema de la asignación de parámetros de planificación, introduciendo algunas técnicas ya existentes.

Para el caso de la asignación de plazos de planificación para sistemas distribuidos, se hizo la distinción entre sistemas planificados con plazos locales y globales. La diferencia fundamental entre ambos radica en el origen de tiempos sobre el que el plazo de planificación empieza a contar. Utilizando una planificación global, y dada una tarea a_{ij}, el origen se sitúa en el instante en el que llega el evento que activa la transacción, instante al que llamaremos t_0. Si utilizamos plazos locales, el origen está en el instante de activación de la propia tarea, t_0+J_{ij}.

El hecho de planificar con plazos globales nos obliga a que todos los relojes del sistema, situados en cada nodo, estén perfectamente sincronizados, algo de compleja implementación. Ésto es así debido a que cada planificador necesitaría conocer de forma precisa el instante de tiempo en el que la transacción fue activada, y ésto ha podido ocurrir en otro procesador, posiblemente situado en otro nodo. Sin embargo, la planificación con plazos locales no requiere del uso de relojes perfectamente sincronizados.

En este trabajo vamos a adaptar las técnicas de asignación de plazos locales de planificación para la asignación de plazos globales. El objetivo es poder determinar si el coste añadido por la implementación de relojes perfectamente sincronizados se puede justificar con una posible mayor utilización conseguida en los recursos.

Los objetivos que se pretenden conseguir en este trabajo se pueden desglosar de la siguiente manera:

- Estudio de las técnicas de asignación de parámetros de planificación existentes en la actualidad para sistemas distribuidos planificados por plazos locales.
- Una vez estudiadas las técnicas actuales de asignación de plazos locales, se propondrá una modificación para la asignación de plazos globales.
- Se llevará a cabo una implementación software de la técnica propuesta, con el objetivo de poder evaluarla. La implementación se integrará dentro de una herramienta de software ya existente llamada MAST.
- 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 técnicas de asignación de plazos de planificación, para poder estudiar las ventajas e inconvenientes de cada una.

1.7. Organización del Proyecto

Para la consecución de los objetivos planteados en el subapartado anterior, la Tesis estará organizada de la siguiente forma:

- En el capitulo 2, “Asignación de plazos globales de planificación”, se propondrán las modificaciones necesarias a realizar sobre las técnicas de asignación de plazos locales, para la asignación de plazos globales de planificación. Para poder llevar a cabo la

evaluación, se describirá cómo se han implementado las técnicas propuestas en una herramienta software ya existente llamada MAST.

- En el capítulo 3, “Generación de casos de estudio”, se describirá el proceso con el cual se han generado los ejemplos que formarán parte de la evaluación que se realizará en el capítulo 4.
- En el capítulo 4, “Evaluación de las técnicas de asignación”, se llevará a cabo la evaluación comparativa entre los algoritmos de asignación de plazos locales y globales de planificación.
- En el capítulo 5, “Conclusiones”, se describirán las conclusiones que se pueden obtener tras la realización de este proyecto, además de aventurar algunas posibles líneas de trabajo futuras.

2. Asignación de plazos globales de planificación

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; y *Deadline Monotonic*, como una generalización de RM cuando los plazos eran menores que los periodos.

En los sistemas 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 llamados plazos de principio a fin ED_{ij}, y existen efectos que provocan que las tareas no sean perfectamente periódicas (efecto del *jitter*), aunque la transacción lo sea.

Por estos motivos se hace necesario definir nuevas técnicas de asignación de parámetros de planificación para sistemas distribuidos. Nos centraremos en sistemas descritos según el modelo lineal definido en la sección 1.1, con tareas y mensajes asignados de manera estática a un determinado procesador y red respectivamente, en los que las tareas están planificadas por algún algoritmo de planificación por plazos globales. A pesar de esta restricción, la búsqueda de una solución exacta para la asignación de plazos es un problema NP-completo. En la sección 1.5 se introdujeron 4 algoritmos que resolvían este problema para otro tipo de sistemas: HOPA para prioridades fijas, y PD,NPD y HOSDA para la asignación de plazos locales de planificación. En este capítulo afrontaremos el problema de la asignación de plazos globales de planificación.

2.1. Propuesta para la asignación de plazos globales de planificación

En este apartado se propondrán las modificaciones a realizar sobre los algoritmos PD,NPD y HOSDA de asignación de plazos locales para afrontar el problema de la asignación de plazos globales de planificación. De hecho, las modificaciones que se propondrán se pueden aplicar a cualquier algoritmo de asignación de plazos locales de planificación en el que los plazos cumplan:

$$ED_{ik} = \sum_{j=0}^{k} SD_{ij}^{local} \qquad \text{Ecuacion 2.1}$$

Ésto es, que la suma de los plazos locales de planificación de la transacción respete el plazo de principio a fin impuesto. Si esta condición no se cumple, la propuesta para la asignación de plazos globales puede ser aplicada igualmente, aunque con resultados que pueden no ser satisfactorios.

Para la asignación de plazos globales, sobre cada algoritmo de asignación de plazos locales se va a añadir un paso adicional que transformarán los plazos locales calculados (SD^{local}) en otros globales equivalentes (SD^{global}), mediante la siguiente ecuación:

$$SD_{ij}^{global} = \sum_{k=1}^{j} SD_{ik}^{local} \qquad \text{Ecuacion 2.2}$$

2.1.1. Asignación proporcional y normalizada de plazos globales de planificación

La aplicación de la modificación propuesta en la ecuación 2.2 para los algoritmos PD y NPD se resume con el siguiente pseudocódigo:

```
Algoritmo Global_NPD/PD is
begin
   Aplica asignación PD/NPD Local
   Aplicación de la ecuación 2.2 sobre todas las transacciones
end Global_NPD/PD
```

2.1.2. Algoritmo HOSDA para la asignación de plazos globales de planificación

Para adaptar el algoritmo HOSDA de asignación de plazos locales se realizan modificaciones similares, pero con algunas consideraciones adicionales. Las fórmulas para el cálculo de los nuevos plazos se van a mantener intactas desde el algoritmo $HOSDA_{local}$, y éstas necesitan plazos locales como parámetros de entrada, y devuelven nuevos plazos locales de planificación. Por esta razón, en el algoritmo, cada tarea va a tener dos plazos de planificación : un plazo local (SD^{local}), y otro global (SD^{global}). La asignación final que se obtendrá como solución será la última asignación de plazos SD^{global}.

El funcionamiento del algoritmo adaptado se describe en el siguiente pseudocódigo:

```
Algoritmo Global_HOSDA is
begin
   Distribuye plazos ED entre las tareas
   loop
      Aplicación de la ecuación 2.2 sobre todas las transacciones
      Cálculo de tiempos de respuesta
      exit when se cumple algún requisito de parada
      Calcular nuevos plazos locales de planificación
   end loop
end Global_HOSDA
```

- Se distribuyen los plazos ED. Cada tarea posee ahora un SD^{local}. La asignación puede ser cualquiera que respete el plazo de principio a fin (ecuación 2.1), como por ejemplo, la asignación proporcional PD_{local} :

$$SD^{local}_{ij} = \mathrm{ED}_{ik} \cdot \frac{C_{ij}}{\sum_{\forall j} C_{ij}}$$

- Con los plazos SD^{local} recién calculados, y la ecuación 2.2, se calculan los plazos SD^{global} de cada tarea.
- Se calculan los tiempos de respuesta de cada tarea mediante alguna técnica de análisis para sistemas distribuidos planificados con plazos globales, como el descrito en la sección 1.4.1. Por lo tanto, para ello se usan los plazos SD^{global} de cada tarea.
- Los criterios de parada se mantienen intactos con respecto al algoritmo $HOSDA_{local}$, y son los siguientes:
  - En dos iteraciones consecutivas se obtiene la misma asignación de plazos, ya que en ese caso, en posteriores iteraciones se obtendría la misma asignación indefinidamente.
  - Se consigue una asignación que hace al sistema planificable. En tal caso, la asignación que se obtiene como solución es $\mathbf{SD_{ij} = SD^{global}_{ij}}$.
  - Se supera el número máximo de iteraciones preestablecidas.
  - Se supera el número máximo de sobreiteraciones sobre una solución ya válida. Esto es así porque el algoritmo $HOSDA_{global}$ también posee la capacidad de optimizar (en el sentido del índice de planificabilidad) una solución ya planificable.
- En caso de que no se cumpla ningún criterio de parada, se calculan nuevos plazos locales de planificación, haciendo uso de la misma formulación que en $HOSDA_{local}$. El proceso se resume en la sección 2.1.2.1. Con los nuevos plazos locales calculados se vuelve al paso 1, en el que se traducían a plazos globales, y se continúa con la iteración.

2.1.2.1. Asignación de nuevos plazos locales de planificación

El cálculo de los nuevos plazos locales hace uso del concepto al que llamamos "exceso", que expresa la distancia que posee cada tarea hasta la planificabilidad. Para el algoritmo HOPA se daban dos definiciones para el exceso:

- Exceso de tiempo de respuesta : Se basa en las diferencias entre los plazos locales y los tiempos de respuesta locales.
- Exceso de tiempo de ejecución : Se basa en el cálculo del *slack*. El *slack* se define como la cantidad de tiempo de ejecución que hay que sustraer en una tarea dada para que ésta consiga ser planificable. El concepto del *slack* es una mejor representación del exceso, sin embargo, su cálculo es computacionalmente muy costoso.

En este trabajo usaremos el exceso de tiempo de respuesta, ya que su cálculo es más rápido, y con él se obtienen resultados satisfactorios. Para una tarea a_{ij} dada, calculamos su exceso de tiempo de respuesta de la siguiente forma :

$$exc(a_{ij}) = (R_{ij} - SD_{ij}^{local}) \cdot \frac{R_i}{ED_i}$$

Para cada recurso procesador PR_k del sistema definimos su exceso de la siguiente forma:

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

Se definen el exceso máximo en los recursos, y el exceso máximo dentro de las tareas que forman la transacción activada por el evento e_i:

$$Mex(PR) = \max_{\forall k}(|exc(PR_k)|)$$

$$Mex(e_i) = \max_{\forall j}(|exc(a_{ij})|)$$

Con estas definiciones de los excesos ya podemos calcular los nuevos plazos locales:

$$SD_{ij}^{local}(nuevo) = SD_{ij}^{local}(anterior)\left(1+\frac{exc(PR_k)}{k_R \cdot Mex(PR)}\right)\left(1+\frac{exc(a_{ij})}{k_a \cdot Mex(e_i)}\right)$$

Los parámetros k_r y k_a nos sirven para controlar respectivamente la influencia del procesador y la transacción sobre el resultado final. La influencia será mayor cuanto más pequeño sea el parámetro. Un rango de valores típicos para estos parámetros está entre 1.0 y 3.0, con los que en extensivos experimentos se obtienen resultados satisfactorios.

El último paso consiste en adaptar los plazos locales calculados de forma que respeten los plazos de principio a fin de la transacción:

$$SD_{ij}^{local} = SD_{ij}^{local}(nuevo) \cdot \frac{ED_i}{\sum_{\forall k} SD_{ik}^{local}(nuevo)}$$

2.2. Implementación de las técnicas de asignación de plazos globales

Con el objetivo de poder evaluar la capacidad para alcanzar soluciones de las técnicas propuestas, 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, que contiene un conjunto de herramientas de software libre, programadas en Ada, y que ha sido desarrollado en el grupo de Computadores y Tiempo Real de la Universidad de Cantabria. MAST proporciona un marco en el que se define un modelo con el que describir el sistema, las funciones con las que traducir ese modelo de entrada a estructuras Ada, y varias técnicas para el análisis y planificación de sistemas. De ésta añadiremos los algoritmos propuestos para plazos de planificación globales como nuevas técnicas disponibles en MAST, aprovechando completamente el modelo de descripción del sistema.

2.2.1. Consideraciones previas a la integración en MAST

MAST define un flujo general de funcionamiento al que se deben adaptar las nuevas técnicas que se añadan. Dicho flujo se perfila en la figura 2.1, adaptado del que aparece en el manual de referencia de MAST [10].

El flujo para las herramientas de asignación de parámetros de planificación aparece indicado con flechas verdes. Las nuevas herramientas de análisis y asignación aparecen indicadas en rojo.

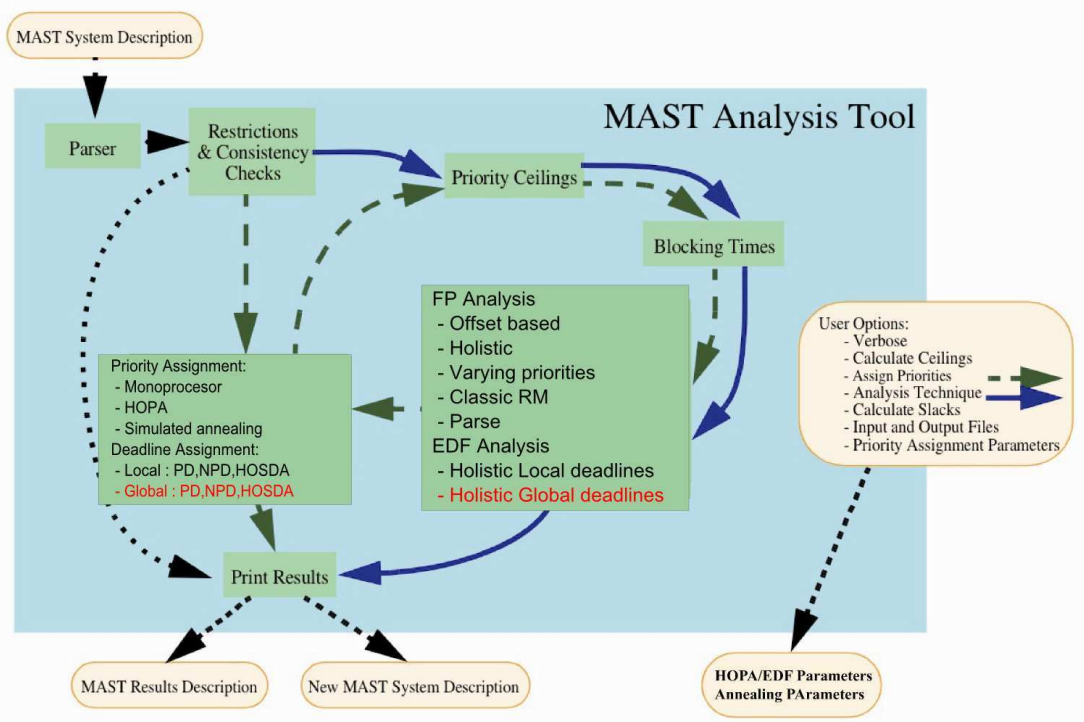

Figura 2.1: Flujo de funcionamiento en MAST

- El sistema descrito en el fichero que sirve de entrada a la herramienta (*Mast System Description)*, es leído por el *Parser* de MAST, que lo traduce a estructuras Ada. La información contenida en este fichero se expondrá en el capítulo 3.
- Se chequean restricciones, tales como que la utilización sea menor que el 100%.
- La herramienta de asignación de parámetros de planificación comienza su ejecución sobre el sistema de entrada, y hará uso de la técnica de análisis especificada.
- Una vez que el algoritmo de asignación finaliza su ejecución, muestra los resultados por pantalla y opcionalmente los guarda en un fichero de salida con un formato de descripción que define MAST (*MAST Results Description*). En estos resultados se especifica, entre otras cosas, si se ha conseguido la planificabilidad del sistema, y en caso afirmativo con qué asignación de parámetros de planificación.

2.2.2. Implementación de los algoritmos PD y NPD

Los algoritmos de asignación proporcional y normalizada son muy similares, por lo que sus implementaciones se describirán conjuntamente. El flujo de la implementación sigue el esquema definido en la figura 2.1, y se describe a continuación:

- El *parser* procesa el fichero de entrada, que está definido en un formato propio de MAST. A continuación la descripción del sistema se guarda en una estructura Ada, sobre la que se comprueban una serie de restricciones, tales como que la utilización no supere el 100% en ningún procesador, o que se vaya a utilizar una herramienta de análisis correcta para el tipo de planificadores que utiliza el sistema. Si alguna no se cumple, se finaliza la ejecución
- Para facilitar la aplicación del algoritmo, se traduce el sistema a un modelo más sencillo, similar al modelo lineal.
- Se aplica la ecuación necesaria para la obtención de una asignación de plazos locales de planificación (ecuación 1.2 para una asignación proporcional, ecuación 1.3 para una normalizada).
- Se transforman los plazos locales en otros globales mediante la ecuación 2.2.
- Se traduce el sistema definido según el modelo simplificado al modelo completo MAST, que es el que requieren las herramientas de análisis.
- Se calculan los tiempos de respuesta mediante alguna técnica de análisis para sistemas distribuidos planificados por plazos globales. Con los tiempos de respuesta se puede determinar la planificabilidad del sistema
- Se muestran los resultados obtenidos. En los resultados figura, entre otros, la utilización en los procesadores, tiempos de respuesta de las tareas, la planificabilidad del sistema y la asignación de plazos obtenida.

2.2.3. Implementación del algoritmo HOSDA Global

El algoritmo heurístico necesita dos parámetros de entrada, k_a y k_r, que necesita para el cálculo de los nuevos plazos. Además, para el criterio de parada, hay que definir un número máximo de iteraciones, y opcionalmente, un número de sobreiteraciones para llevar a cabo la optimización.

- **Parámetros k** : En la implementación se van a agrupar en dos listas, una con los valores de k_a y otra con los valores de k_r. Durante la ejecución del algoritmo se irán formando parejas (k_a,k_r) con los valores de las listas. En cada pareja ambos valores están situados en la misma posición (índice del array) de su lista.
- **Iteraciones** : Se define una lista de valores en el que se indica cuantas iteraciones realizar sobre cada pareja de valores de k posible.
- **Sobreiteraciones :** Opcionalmente se puede definir un número máximo de sobreiteraciones para intentar optimizar una solución.

Todos estos parámetros van a perfilar el funcionamiento de la implementación del algoritmo HOSDA Global, que se resume con el siguiente listado, en el que se han obviado los pasos propios de MAST que se han descrito en la figura 2.1. Por los motivos descritos en la sección 2.1, cada tarea va a poseer dos plazos de planificación, uno local SD^{local}, y otro global SD^{global}.

```
Algoritmo Global_HOSDA is
begin
   ...
   Reparto inicial de plazos ED. Se obtiene SD^local para cada tarea
   Aplicación de ecuación 2.2. Se obtiene SD^global para cada tarea
   for longitud [lista iteraciones] loop
      [iteraciones] = Siguiente valor de [lista iteraciones]
      for longitud [lista k] loop
         [k_a,k_r] = Siguiente pareja de valores contenida en [lista k]
         for [iteraciones] loop
            Herramienta de análisis con los plazos SD^global
            exit when Criterio de Parada
            Recalcular nuevos plazos SD^local
            Ecuación 2.2. Se obtienen nuevos SD^global para cada tarea
         end loop
      end loop
   end loop
   ...
end Global_HOSDA
```

A continuación se describen con más detalle cada paso involucrado en la implementación del algoritmo, junto a los pasos propios de MAST:

- El *parser* de MAST procesa el fichero de entrada. La descripción del sistema se almacena como una estructura Ada.
- Los parámetros que necesita el algoritmo para funcionar (k's, iteraciones y sobreiteraciones) se leen de un fichero de texto externo. En caso de no existir tal fichero, se almacenan los siguientes valores predeterminados :
  - Lista k_a = (1.5,2,3) ; Lista k_r = (1.5,2,3). Éstos son los valores por defecto que ya se usaban en el algoritmo HOPA y $HOSDA_{local}$
  - Lista Iteraciones = (10,20,30)
  - Sobreiteraciones = 0, por defecto no vamos a optimizar las soluciones.
- Se comprueban las restricciones de forma similar a los demás algoritmos
- Se reparte el plazo ED entre cada tarea, de cualquier forma en la que se respeten los plazos de principio a fin. En la implementación se usará el reparto proporcional PD_{local}.

  El resultado es que cada tarea posee un plazo de planificación SD^{local} asignado de forma proporcional a su tiempo de ejecución de peor caso.
- Con los plazos SD^{local} calculados en el paso anterior, y la ecuación 2.2, se calculan los plazos SD^{global} de cada tarea.
- Asigna a la variable [iteraciones] el valor proveniente del *array* [lista iteraciones] correspondiente a la iteración actual.
- Se establece la nueva pareja $[k_a,k_r]$ que se utilizará en las [iteraciones] iteraciones siguientes.
- Haciendo uso de los últimos plazos SD^{global} calculados, y de alguna técnica de análisis para sistemas distribuidos planificados por plazos globales, se calculan los tiempos de

respuesta. Al ser la única herramienta de este tipo implementada, se usará la técnica holística. Para evitar problemas de convergencia que se puedan dar cuando las utilizaciones de los procesadores sea elevada, se añade una condición de parada en el análisis, de forma que éste se detiene cuando el tiempo de respuesta de una transacción supera su plazo impuesto multiplicado por un factor configurable. De esta forma acortamos y acotamos el tiempo requerido para el análisis. El valor de este parámetro se ha prefijado en 2.

- Se detiene la ejecución del algoritmo de asignación de plazos si se cumple algún criterio de parada:
  - En dos iteraciones consecutivas se obtiene la misma asignación de plazos globales SD^{global}, ya que en ese caso, en posteriores iteraciones se obtendría la misma asignación indefinidamente.
  - Se consigue una asignación que hace al sistema planificable. En tal caso, la asignación que se obtiene como solución es $\mathbf{SD_{ij} = SD^{global}_{ij}}$. Este criterio es válido si no se desea optimizar la solución (Sobreiteraciones = 0). Si se desea optimizar, el criterio a utilizar es el siguiente :
    - Se supera el número máximo de sobreiteraciones sobre una solución ya válida para intentar optimizarla.
  - Se supera el número máximo de iteraciones que establecen los parámetros [lista k] y [lista iteraciones].
- Si no se cumple ningún criterio de parada, se calculan los nuevos SD^{local} de cada tarea. Para ello se utiliza la última pareja de k's establecida, los tiempos de respuesta recién calculados, y los últimos plazos SD^{local} (que se han obtenido en la iteración anterior). El proceso se describió en la sección 2.1.2.1.
- Con los plazos SD^{local} recién calculados en el paso anterior, y la ecuación 2.2, se calculan los nuevos plazos SD^{global} de cada tarea. Se comienza la siguiente iteración aplicando de nuevo la técnica de análisis.

3. Generación de casos de estudio

En este trabajo hemos propuesto e implementado tres algoritmos de asignación de plazos globales de planificación. El siguiente objetivo será ponerlos a prueba en una batería de ejemplos. Para ello se ha decidido dividir la evaluación en dos partes.

En el primer caso de estudio se tomará un ejemplo que propone J.C. Palencia en [8] para la comparación de técnicas de análisis. Se ha elegido este ejemplo porque forma parte de un trabajo anterior en el que se aplica una técnica rudimentaria de asignación de plazos globales de planificación en el que se observaban una utilizaciones de los recursos que nos parecieron anormalmente bajas. En él se utiliza además la misma técnica de cálculo de tiempos de respuesta que utilizamos en el presente trabajo.

Para el segundo caso de estudio no queremos ceñirnos a un ejemplo concreto, sino que queremos abarcar un amplio espectro de sistemas con el que poder obtener conclusiones generales sobre los algoritmos bajo estudio. Así, el segundo caso de estudio consistirá en una extensa batería de ejemplos de tests, con sistemas generados aleatoriamente, con el que se abarcarán sistemas con distintos tamaños, plazos y cargas.

En ambos casos de estudio, y especialmente en el segundo, hay que generar un elevado número de ficheros MAST descriptores de sistemas, por lo que se ha desarrollado una herramienta que genera ejemplos de forma automática.

Esta herramienta generadora toma una serie de parámetros de entrada que definirán las características de los sistemas que genere, tales como el número de procesadores del sistema, número de transacciones, plazos, localización de las tareas, etc. Para cada caso de estudio, el generador de ejemplos actuará de manera distinta; se describirá su funcionamiento concreto para cada caso.

Cada fichero generado describe el sistema con un formato propio de la herramienta MAST. Las características básicas de esta descripción se resumen a continuación.

3.1. Descripción del sistema distribuido con el modelo MAST

En esta sección se va a describir brevemente cómo es el modelo con el que MAST describe los sistemas de tiempo real, que está basado en el modelo lineal ya descrito.

En el modelo MAST la unidad mínima planificable es la **actividad**. Cada actividad lleva asociadas una serie de características y relaciones que la definen, tal y como se observa en la figura 3.1, extraída del manual de referencia de MAST [10].

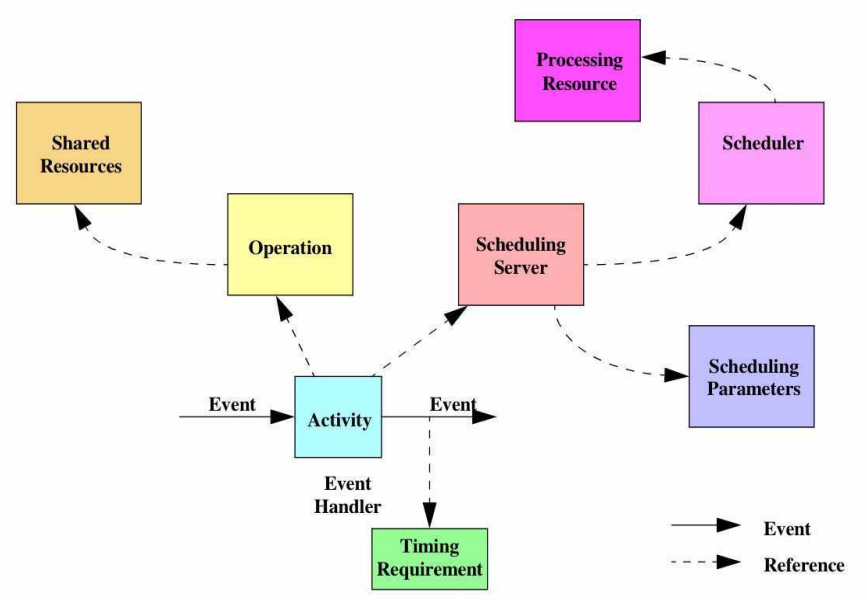

Figura 3.1: Elementos que definen una actividad

- **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).
- **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 la instancia de una **operación** que va a ser ejecutado por un **servidor de planificación** concreto.
- **Manejadores de Eventos** (Event Handler). Cada **manejador de evento** asocia 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 qué **servidor de planificación** lleva asociado.
- **Eventos** (Events). Son los encargados de activar los manejadores de eventos. Si son externos, pueden ser periódicos o aperiódicos. Los eventos internos son generados por los manejadores de eventos, y pueden llevar asignados requisitos temporales que se describen aparte.
- **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.
- **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.

La transacción MAST está formada por eventos, manejadores de eventos y requisitos temporales, tal y como se observa en la figura 3.2, obtenida del manual de referencia de MAST [10]. El manejador *Multicast* que aparece en la imagen es un caso especial de manejador que no va a ser utilizado en este trabajo.

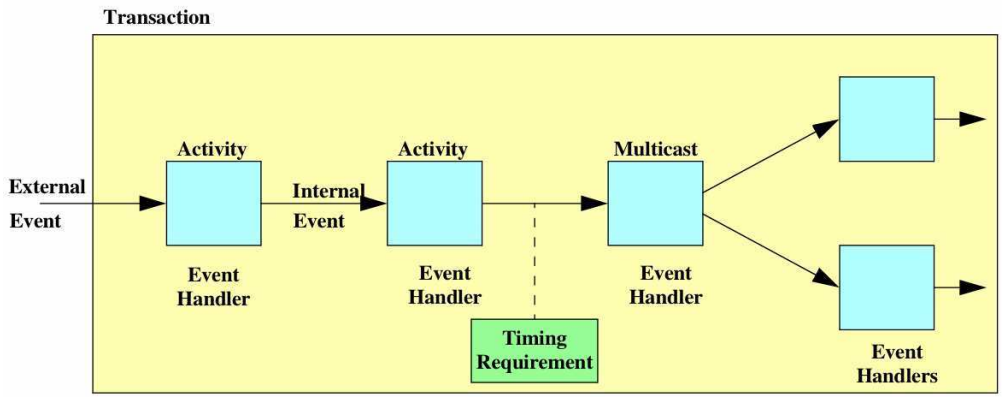

Figura 3.2: Elementos que definen una transacción MAST

Toda la información que describe las actividades y transacciones del sistema de tiempo real a analizar aparece escrita, con un formato determinado, en un archivo de texto plano que MAST es capaz de procesar. Para generar los ficheros descriptores se pueden escribir directamente como un fichero de texto plano, o se puede utilizar una herramienta gráfica de generación de ficheros integrada en MAST.

En este trabajo hemos desarrollado un generador automático de ejemplos que es capaz de generar de forma masiva archivos descriptores MAST de sistemas con una características controlables por el usuario. El modo en el que el generador actúa para la generación de cada caso de estudio se describe a continuación.

3.2. Generación del caso de estudio 1

Para este caso de estudio se ha escogido un ejemplo que propuso J.C. Palencia en [8]. El ejemplo consiste en un sistema compuesto por 100 tareas repartidas en 4 procesadores y 5 transacciones, que se creó con el objetivo de comparar dos técnicas de análisis de sistemas distribuidos planificados por plazos globales : una técnica holística, y otra basada en *offsets*.

En dicho ejemplo se aplicaba una asignación rudimentaria de plazos globales de planificación, en la que los plazos globales de planificación de cada tarea eran :

$$SD_{ij} = 10 \cdot T_i \qquad \text{Ecuacion 3.1}$$

Por lo tanto, todas las tareas en la misma transacción tienen el mismo plazo de planificación. A este algoritmo de asignación lo llamaremos *Ultimate Deadline* (UD) [1]. En la siguiente tabla se resumen las características básicas del sistema que formará parte del caso de estudio 1:

Tabla 3.1: Sistema base para el caso de estudio 1

Nº de procesadores	4
Nº de transacciones	5
Longitud de transacciones	20
T_{max}/T_{min}	10

Los periodos de las transacciones, y los tiempos de ejecución de peor caso de cada tarea, se han calculado de forma que la utilización total del sistema sea del 1%. Para cada transacción, se van a considerar 5 tipos de plazos ED, de forma que la relación ED_i/T_i tenga valores entre 1 y 5. Con ésto se pretende estudiar cómo se comportan los algoritmos a medida que los plazos de la transacción se hacen menos restrictivos.

El objetivo de este estudio es comprobar hasta qué utilización máxima es capaz cada algoritmo de planificar el sistema. Por lo tanto, con el generador de ejemplos tomaremos el sistema base definido en la tabla 3.1, y se le irá aumentando progresivamente la utilización, hasta alcanzar el 100%. Para aumentar la utilización del sistema, el generador multiplica los tiempos de ejecución de cada tarea del sistema base (que posee una utilización del 1%), por la utilización media que quiera obtener. Por ejemplo, si queremos obtener una utilización media del 40%, se multiplicarán todos los tiempos de ejecución del sistema base por 40.

$$U_1 = \sum_{\forall(i,j)} \frac{C_{ij}}{T_i} = 0.01 = 1\%$$

$$\sum_{\forall(i,j)} \frac{K \cdot C_{ij}}{T_i} = K \cdot \sum_{\forall(i,j)} \frac{C_{ij}}{T_i} = K \cdot U_1 = K\,\%$$

Para cada utilización, se aplicarán todos los algoritmos de asignación de parámetros de planificación bajo estudio, y se comprobará si han conseguido hacer planificable el sistema. Para cada algoritmo se anotará la utilización máxima que pudieron planificar.

3.3. Generación del caso de estudio 2

El conjunto de sistemas que forman parte del segundo caso de estudio se dividirán en tres grupos dependiendo del tamaño del sistema : pequeño (Ejemplo Pequeño, EP), intermedio (EI) y grande (EG).

En cada grupo se partirá de un sistema base, sobre el que se generarán aleatoriamente 100 ejemplos diferentes basados en él. En la tabla 3.2 se muestra la información básica que define cada sistema base.

Tabla 3.2: Características de los sistemas base

	EP	EI	EG
Número de procesadores	3	5	8
Número de transacciones	6	8	12

El generador de ejemplos desarrollado tomará la información del sistema base, y generará 100 ejemplos para cada grupo. Cada ejemplo se generará utilizando el siguiente algoritmo:

- Toma el número de procesadores y transacciones que definen el grupo de ejemplos, valores que se muestra en la tabla 3.2.
- La longitud de cada transacción se elige de forma aleatoria entre 1 y el número de procesadores del sistema.
- La localización de cada tarea se elige aleatoriamente, de forma que en cada transacción no haya más de una tarea situada en el mismo procesador.
- Los periodos de cada transacción se eligen de forma aleatoria entre un valor máximo y mínimo preestablecido. El valor máximo y mínimo elegido ha sido 2000 y 100 respectivamente, de forma que la relación T_{max}/T_{min} sea como mucho de 20.
- Todos los tiempos de ejecución de peor caso se inicializan a 1, para así obtener una utilización inicial baja.
- Se aumenta de forma progresiva la utilización del sistema. Para ello se aumenta el tiempo de ejecución de peor caso de una tarea elegida aleatoriamente de acuerdo con la siguiente fórmula:

$$C_{nuevo} = C_{anterior} + 0{,}01 \cdot K \cdot T_i$$

  La aplicación de la fórmula provoca que el procesador en el que reside la tarea en cuestión vea aumentada su utilización en un K %. En todo caso, el generador evita que haya un procesador que alcance una utilización superior al 100%.
- Para cada nueva utilización generada, se guardan 5 copias del sistema, cada una con distintos plazos de principio a fin en las transacciones, siendo T_i el periodo de la transacción, y N_i el número de tareas que forman la transacción.

- $ED_i = T_i$
- $ED_i = N_i T_i/2$
- $ED_i=N_i T_i$
- $ED_i = 2N_i T_i$
- $ED_i=Random(T_i,2N_i T_i)$, el plazo se obtiene aleatoriamente entre los valores T_i y $2N_i T_i$.

Una vez que se han generado todos los ficheros descriptores MAST del caso de estudio, se les aplicarán todos los algoritmos de asignación de parámetros de planificación bajo estudio, con el objetivo de observar las utilizaciones máximas planificables que se obtienen en cada uno.

4. Evaluación de las técnicas de asignación

En este trabajo hemos propuesto e implementado tres algoritmos de asignación de plazos globales de planificación. El siguiente objetivo será el ponerlos a prueba sobre los casos de estudio descritos en el capítulo 3.

Para poder comparar los diferentes algoritmos de asignación de prioridades y plazos de planificación que se han propuesto, se necesita algún criterio con el que comparar los resultados, y poder determinar la bondad de cada uno de ellos. En la realización de este trabajo se han utilizado principalmente dos, el **Índice de Planificabilidad,** y la **Utilización Máxima** alcanzada:

- **Índice de Planificabilidad** : Da una idea de la distancia entre los plazos y los tiempos de respuesta de un sistema. Para cada transacción se define de la siguiente forma:

$$\text{Índice} = \begin{cases} -1, \text{ cuando no planificable} \\ \dfrac{ED_i - R_i}{ED_i}, \text{ cuando planificable} \end{cases}$$

  Por lo tanto, valores altos del índice (cercanos a 1) indican que existe un amplio margen entre los plazos y el tiempo de respuesta. Como estamos trabajando con sistemas de tiempo real estricto, cuando éstos no son planificables, no estamos interesados en conocer el margen que tienen hasta la planificabilidad. Por lo tanto, para diferenciar claramente los sistemas planificables de los que no lo son, a éstos últimos les asignamos un valor del índice de -1. El índice de todo el sistema es el valor medio de todos los índices de planificabilidad de las transacciones que lo forman.

  El índice de planificabilidad es la forma que utiliza MAST de indicarnos si el sistema es planificable.

- **Utilización Máxima** : Para un sistema dado, se va aumentando progresivamente la carga en los procesadores (típicamente aumentando los tiempos de ejecución de las tareas) hasta que el algoritmo de planificación utilizado no consiga planificar el sistema. A la última utilización planificable del sistema la llamaremos utilización máxima.

  Para determinar la utilización máxima observaremos, en los resultados obtenidos por MAST, la última utilización que posee un índice de planificabilidad no negativo.

Para todos los sistemas que forman ambos casos de estudio se han aplicado las siguientes técnicas de asignación de parámetros de planificación: HOPA para prioridades fijas; PD,NPD y HOSDA para la asignación de plazos de planificación tanto globales como locales. Para los

algoritmos heurísticos se han utilizado los valores por defecto para los parámetros configurables que poseen:

- Lista k_a = (1.5,2,3) ; Lista k_r = (1.5,2,3).
- Lista Iteraciones = (10,20,30)
- Sobreiteraciones = 0, no deseamos optimizar.

Como herramientas de análisis, para el caso de la asignación de prioridades fijas con el algoritmo HOPA, se ha utilizado la basada en *offsets* que se propone en [6]. Para el análisis de los sistemas distribuidos EDF se ha utilizado la herramienta holística introducida en la sección 1.4.2., para plazos locales o globales según corresponda.

4.1. Evaluación del caso de estudio 1

Durante las primeras pruebas que se hicieron con los algoritmos de asignación de plazos globales implementadas, se obtuvieron utilizaciones que en principio parecían excesivamente altas. Ésta apreciación era especialmente llamativa si se comparaban con los resultados obtenidos en el trabajo en el que se presentaba la herramienta de análisis que estamos utilizando [8].

Para comprobar que las utilizaciones máximas que estábamos obteniendo eran correctas, aplicamos todos los algoritmos de asignación sobre un ejemplo extraído de [8]. Para poder comparar los resultados, también implementamos la asignación UD que en él se utilizaba. Los resultados que obtuvimos con la asignación UD se correspondieron con los que se mostraban en [8] ,por lo que descartamos un error en la implementación.

El resto de resultados obtenidos con este ejemplo se mostrarán a continuación. En la figura 4.1 se muestran los resultados obtenidos con el algoritmo de asignación UD (ecuación 3.1) y las técnicas de plazos globales propuestas.

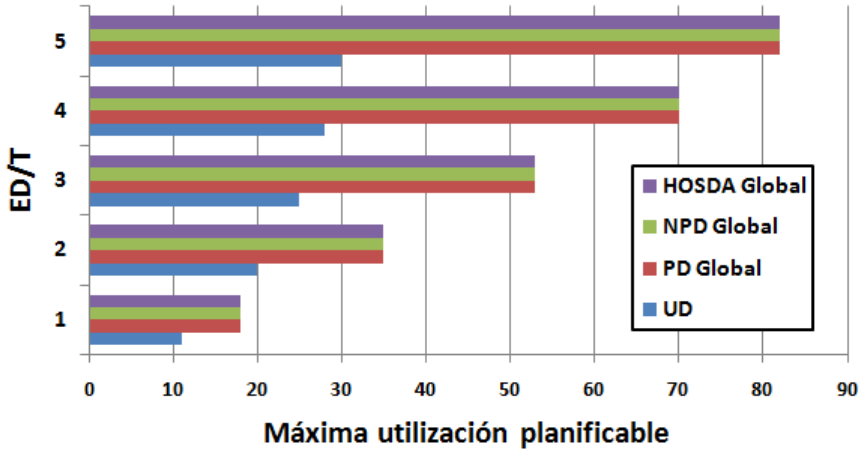

Figura 4.1: Comparación entre UD y los algoritmos de asignación de plazos globales propuestos

Se observa que con la asignación UD se obtienen las peores utilizaciones, especialmente a medida que los plazos de las transacciones se vuelven menos restrictivos. Cuando la relación

ED/T = 5, la asignación UD consiguió una utilización máxima del 30%, mientras que los algoritmos de plazos globales propuestos alcanzaron un 82%.

Se comprueba también que para este ejemplo no hay diferencias en los resultados obtenidos por los algoritmos de asignación de plazos globales. Ésto no quiere decir que los tres algoritmos obtuvieran siempre la misma asignación de plazos, sino que consiguieron siempre la misma utilización máxima. Por lo tanto, para este ejemplo el algoritmo heurístico propuesto no fue capaz de obtener una solución más allá de la asignación proporcional inicial, que es una asignación PD Global.

En la figura 4.2 se muestra la comparación de los resultados obtenidos por los algoritmos de asignación de plazos locales (PD,NPD,HOSDA), y el de asignación de plazos globales PD Global. No se han incluido los otros dos algoritmos de asignación de plazos globales (NPD,HOSDA) porque, como se vio en la figura 4.1, con ellos se obtenían los mismos resultados.

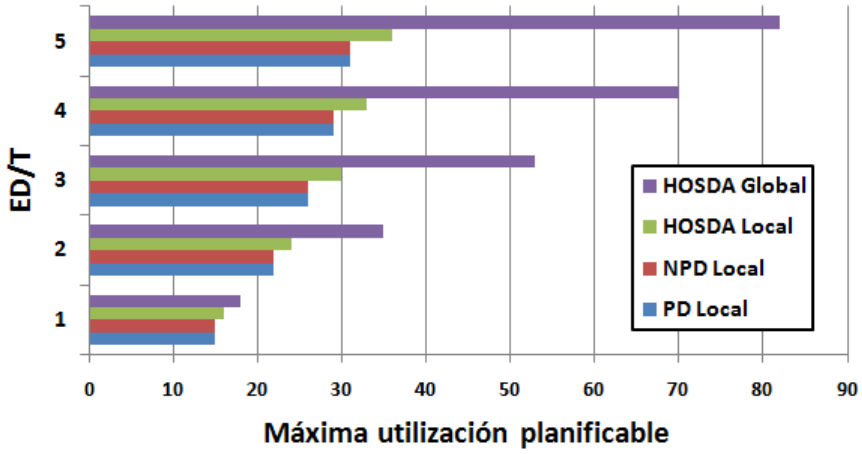

Figura 4.2: Comparación entre algoritmos de plazos locales y globales

Observando los resultados que se muestran en la figura 4.2, se comprueba que a medida que los plazos de las transacciones aumentan, también lo hacen las diferencias entre la planificación con plazos locales y globales, siempre a favor de ésta última. En el caso extremo en el que la relación ED/T es de 5, para este ejemplo, utilizando plazos globales podemos planificar hasta un 50% más de utilización en los recursos que con los plazos locales.

Entre los algoritmos para plazos locales se puede observar que, al contrario que con plazos globales, el algoritmo HOSDA Local es capaz de mejorar los resultados obtenidos por PD Local. Ésto corrobora los resultados obtenidos en [19][18].

4.2. Evaluación del caso de estudio 2

Se va a proceder a la exposición de los resultados obtenidos para el segundo caso de estudio. La evaluación va a venir dividida en 3 partes, una para cada tipo de grupo descrito en la sección 3.3: pequeño (EP), intermedio (EI) y grande (EG).

El objetivo que se pretende conseguir con este estudio es el de observar qué utilización máxima puede planificar cada algoritmo de asignación de parámetros de planificación sobre un extenso conjunto de sistemas, y poder así sacar conclusiones de aplicación general.

4.2.1. Evaluación grupo EP

En esta sección se mostrarán los resultados obtenidos sobre el grupo de sistemas de tamaño pequeño. Para poder sacar conclusiones de carácter general sobre las utilizaciones conseguidas, se mostrará la media de las utilizaciones máximas planificables que obtuvieron los algoritmos en los 100 ejemplos que forman el grupo.

En primer lugar se compararán entre sí los resultados obtenidos por los algoritmos heurísticos, tanto de prioridades fijas como dinámicas (HOPA, HOSDA Local, HOSDA Global):

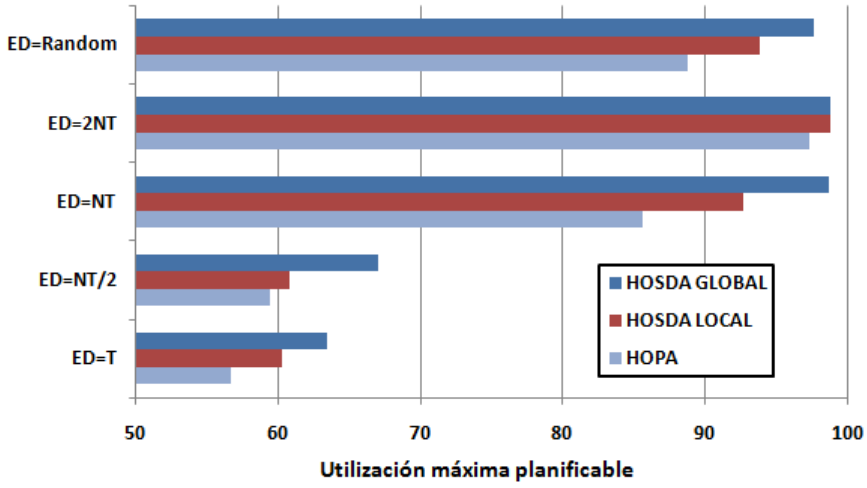

Figura 4.3: Resultados de los algoritmos heurísticos para el Grupo EP

En la figura 4.3 se puede observar que utilizando prioridades dinámicas EDF se obtienen en media mejores utilizaciones que utilizando prioridades fijas obtenidas por el algoritmo HOPA, resultado esperado ya que es una conclusión similar a la que se obtiene en los sistemas monoprocesadores. Entre los sistemas planificados por plazos se observa que utilizando plazos globales (HOSDA Global) se consiguen mejores utilizaciones que utilizando plazos locales (HOSDA Local). Para el caso ED=2NT se observan pocas diferencias entre los tres algoritmos. Ésto es debido a que los plazos de las transacciones son tan poco restrictivos que los tres algoritmos consiguen utilizaciones cercanas al 100%, y no existe margen para la mejora.

En la figura 4.4 se muestran los resultados obtenidos por los algoritmos de asignación de plazos proporcional y normalizada PD/NPD, tanto para plazos globales como locales, y el algoritmo HOSDA Global.

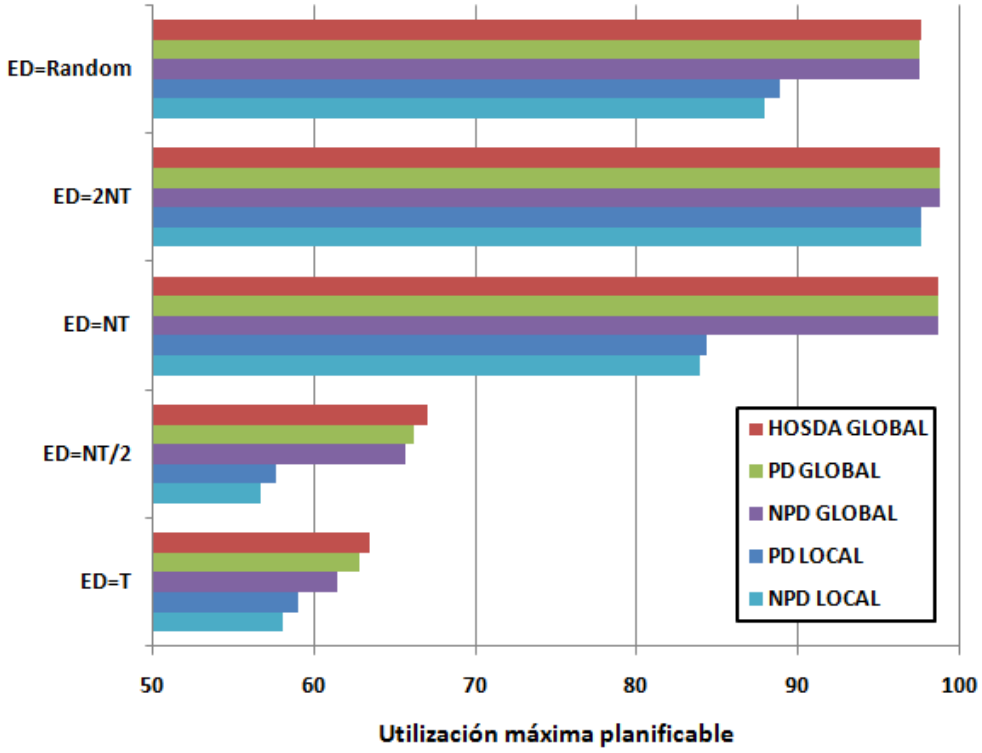

Figura 4.4: Resultados algoritmos PD,NPD,HOSDA Global para el grupo EP

En la gráfica se comprueba que utilizando plazos globales para la planificación se obtuvieron mejores utilizaciones en los recursos del sistema. También se puede observar que apenas existieron diferencias entre los resultados obtenidos por el algoritmo PD Global y HOSDA Global. De ésto se extrae que el algoritmo HOSDA Global consiguió mejorar en algún caso a la asignación inicial PD Global, especialmente con plazos de principio a fin más restrictivos, pero no en un número elevado de casos, ni con una mejora significativa de la utilización.

Adicionalmente se puede apreciar que con la asignación NPD Global se obtienen resultados ligeramente peores que con PD Global cuando los plazos ED son más restrictivos.

Para completar la evaluación, en la figura 4.5 se representan los tiempos de ejecución medios que necesitó cada algoritmo de asignación de plazos globales para obtener y analizar una solución en cada utilización media en los recursos. Se obtuvieron utilizando un reloj de tiempo de ejecución, por lo que en los resultados no se contabiliza el tiempo de ejecución de otras tareas que estuviera ejecutando el sistema operativo que estaba realizando el análisis.

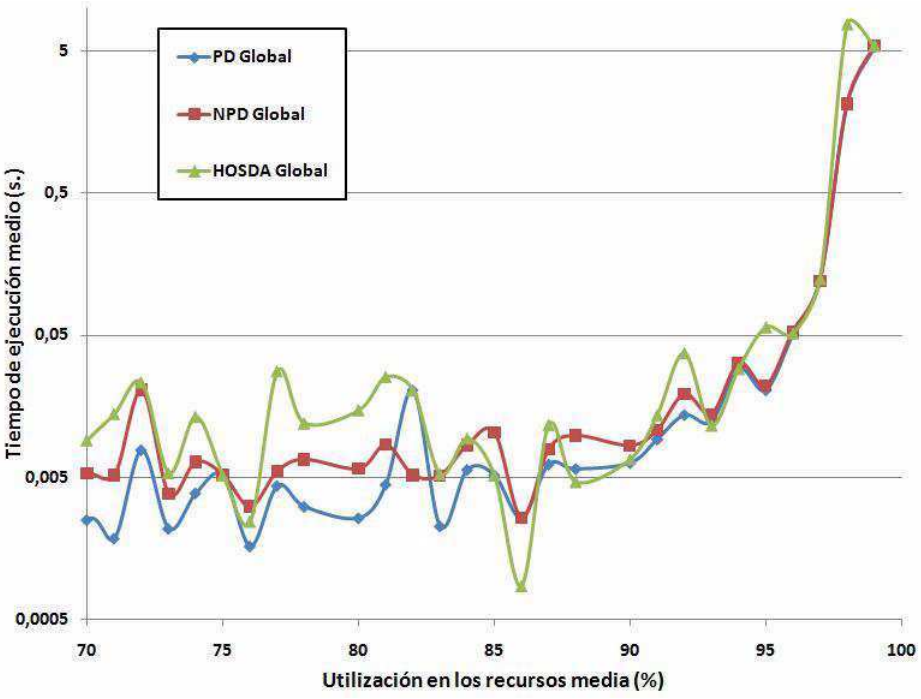

Figura 4.5: Tiempos ejecución grupo EP

Como era de esperar, los tiempos de ejecución aumentan a medida que la utilización en los recursos aumenta. Se observa además que los tiempos no varían drásticamente entre los tres algoritmo analizados, siendo el algoritmo HOSDA Global ligeramente más lento. Ésto es así debido a que el algoritmo HOSDA, en la mayoría de los casos, encontró solución en su primera iteración, por lo que el cómputo total realizado tiene una carga similar a la del algoritmo PD/ NPD Global.

En los casos en los que en la primera iteración del algoritmo HOSDA Global no se encontró solución, y sí lo hizo en posteriores iteraciones, el tiempo de ejecución fue mayor. Cómo ésto ocurrió en un número reducido de casos, las diferencias en el tiempo medio de ejecución no son claramente apreciables.

Cabe destacar que para cualquier algoritmo de asignación que vamos a aplicar, los tiempos de ejecución obtenidos se deben principalmente a la herramienta de análisis utilizada. Ésto es especialmente significativo para el algoritmo HOSDA, ya que potencialmente hará uso en varias ocasiones de la herramienta de análisis hasta encontrar una solución.

4.2.2. Evaluación grupo EI

Al igual que con el grupo de sistemas pequeños, el objetivo es observar la utilización máxima alcanzada por cada algoritmo sobre un extenso grupo de sistemas. Para ello, se mostrará la media de las utilizaciones máximas planificables que obtuvieron los algoritmos en los 100 ejemplos. En primer lugar se compararán los algoritmos heurísticos entre sí, tanto de prioridades fijas como dinámicas (HOPA, HOSDA Local, HOSDA Global), tal y como se muestra en la figura 4.6.

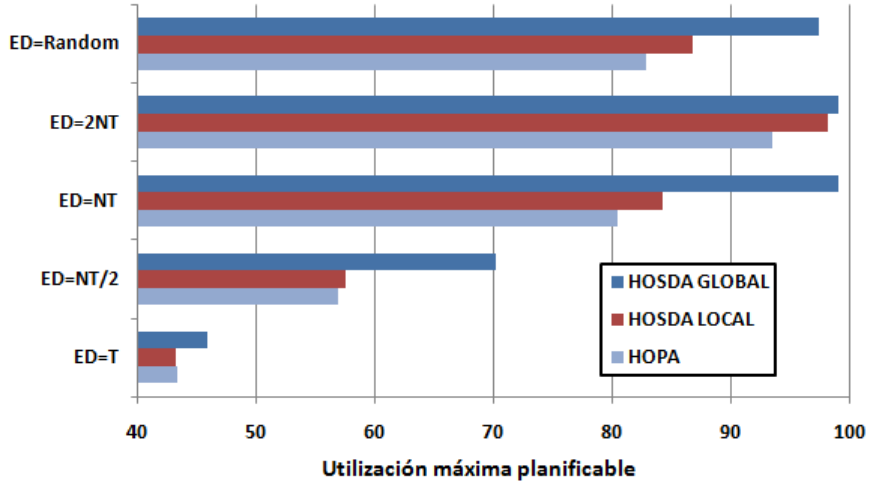

Figura 4.6: Resultados de los algoritmos heurísticos para el Grupo EI

En este caso se observa que las diferencias entre la utilización de plazos locales y globales ha aumentado con respecto al ejemplo de sistemas pequeños. Ésto parece indicar que a medida que el tamaño de los sistemas aumenta, también lo hacen las ventajas del uso de plazos globales. Además, como era de esperar, puede observarse que planificando con prioridades fijas (HOPA) se obtienen menores utilizaciones máximas en los recursos que utilizando planificación EDF.

En la siguiente gráfica, figura 4.7, se muestran los resultados obtenidos por los algoritmos de asignación proporcional y normalizada PD/NPD (tanto para plazos globales como locales), y HOSDA Global.

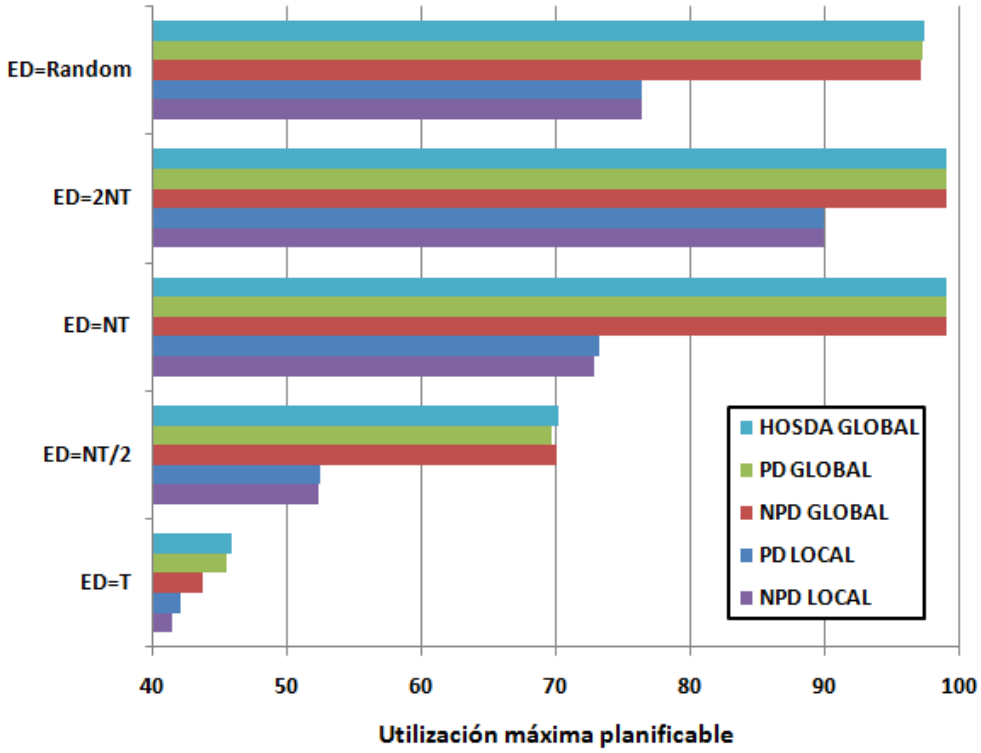

Figura 4.7: Resultados algoritmos PD,NPD,HOSDA Global para el grupo EI

Se puede observar que las diferencias entre la planificación global y local han aumentado con respecto a los sistemas pequeños, por lo que se mantiene la tendencia observada en la figura 4.6. El algoritmo HOSDA Global sólo logró superar levemente a la asignación PD Global cuando los plazos de principio a fin eran más pequeños (ED=T y ED=NT/2).

Se puede apreciar también que para plazos ED=NT/2, NPD Global mejoró levemente los resultados obtenidos por PD Global, llegando incluso a igualar los resultados obtenidos por HOSDA Global.

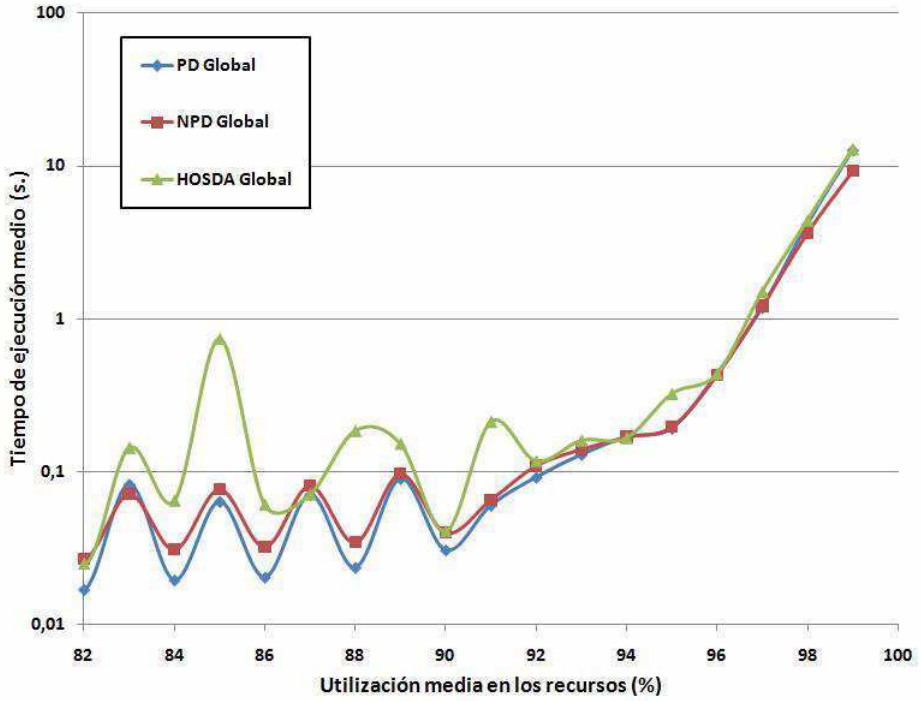

Figura 4.8: Tiempos de ejecución grupo EI

En la figura 4.8 se representan los tiempos de ejecución medios que necesitó cada algoritmo de asignación de plazos globales en obtener y analizar una solución para cada utilización media en los recursos. Los resultados son similares a los obtenidos en la figura 4.5, salvo que esta vez, al ser sistemas con más tareas, los tiempos de ejecución fueron mayores.

4.2.3. Evaluación grupo EG

En esta sección se van a presentar los resultados obtenidos sobre el grupo de sistemas grandes. El objetivo será, como en las secciones anteriores, determinar la utilización máxima planificable que obtiene cada algoritmo sobre un conjunto extenso de ejemplos con características básicas similares.

Como se ha observado en los resultados expuestos hasta ahora, apenas existen diferencias entre las utilizaciones obtenidas por los tres algoritmos de asignación de plazos globales. Adicionalmente, al ser sistemas de tamaño elevado, el tiempo requerido para la aplicación de un algoritmo de asignación de parámetros de planificación sobre el conjunto de 100 ejemplos superaba las semanas.

Por estas razones, para el caso del algoritmo HOSDA Global, se limitó el número de iteraciones a 9 (3 iteraciones por cada pareja de k's), en vez de las 30 iteraciones que son por defecto. Pese

a esta simplificación, los resultados siguen siendo igualmente válidos para llevar a cabo la comparación entre las planificaciones con plazos globales y locales, objetivo principal de este proyecto.

El algoritmo HOPA tampoco fue aplicado por cuestiones de tiempo, además de que con los resultados ya obtenidos se ha comprobado que planificando con prioridades fijas se obtienen peores utilizaciones, algo que se podía intuir de antemano. Por lo tanto, su ejecución para este grupo no aportaría nada nuevo.

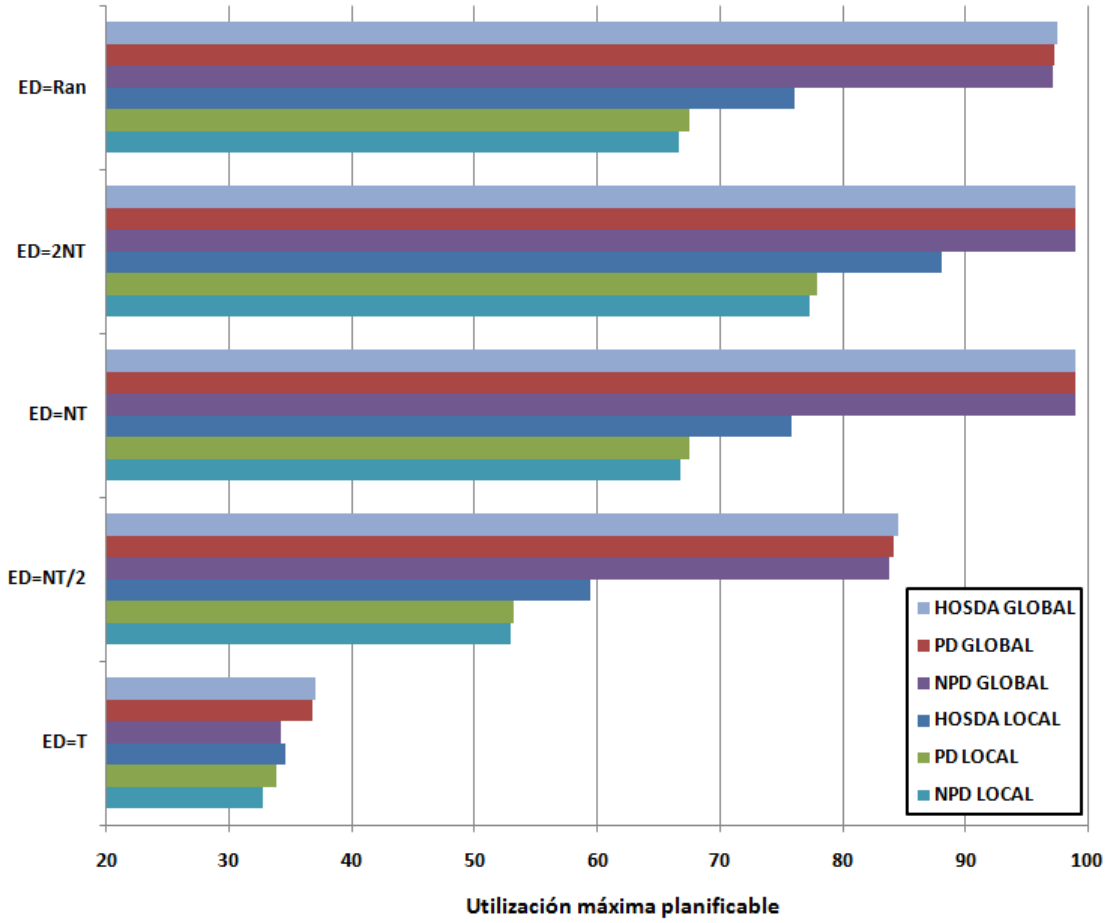

Figura 4.9: Resultados algoritmos PD,NPD,HOSDA Global para el grupo EG

En la figura 4.9 se muestran los resultados obtenidos con los algoritmos aplicados. En ella se observa que las diferencias entre la planificación con plazos globales y locales ha vuelto a aumentar, algo que parece confirmar la apreciación realizada en el ejemplo anterior de que la planificación con plazos globales es más beneficiosa a medida que el tamaño de los sistemas aumenta.

Cabe destacar los resultados obtenidos para ED=T, en el que la asignación NPD Global obtuvo resultados claramente inferiores a las otras asignaciones de plazos globales.

En la figura 4.10 se muestran los tiempos medios de ejecución que necesitaron los algoritmos de asignación de plazos globales para obtener y analizar una solución.

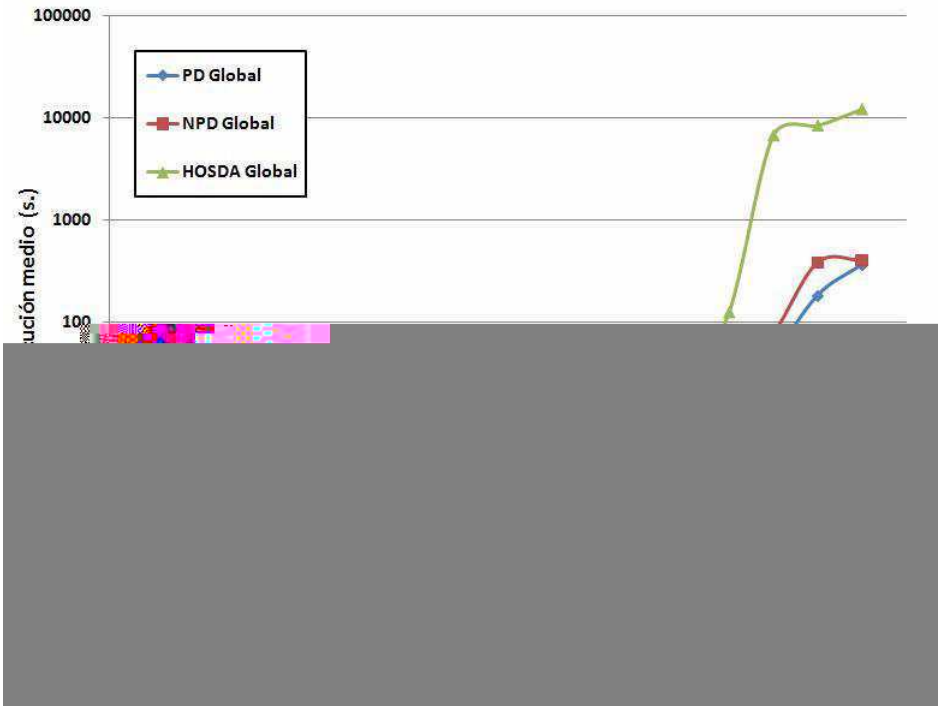

Figura 4.10: Tiempos de ejecución grupo EG

Como era de esperar, los tiempos son mayores que para los sistemas más pequeños. Para este grupo de ejemplos las diferencias entre los tiempos de ejecución del algoritmo HOSDA Global y PD/NPD Global son más apreciables. Estas diferencias se deben fundamentalmente al uso continuado de la herramienta de análisis que ha necesitado llevar a cabo el algoritmo HOSDA para obtener una solución.

4.3. Análisis global de los resultados obtenidos

A la vista de los resultados obtenidos se pueden obtener las siguientes conclusiones:

- Planificando los sistemas utilizando plazos globales se obtienen mejores utilizaciones que utilizando plazos locales. Ésto se debe a que utilizando una planificación global, los planificadores obtienen una mejor ordenación de las tareas que utilizando plazos locales. Esta diferencia es tanto mayor cuanto más largas son las transacciones.

  El fenómeno se ilustra en la figura 4.11. El eje vertical representa la diferencia entre la utilización máxima obtenida por el algoritmo PD Global y PD Local para distintos tipos de plazos ED. En la figura se observa que a medida que el tamaño del sistema aumenta, las diferencias entre las utilizaciones obtenidas también lo hacen. Para el caso ED=T no se aprecian estas diferencias, debido probablemente a que los plazos impuestos son muy restrictivos, y los algoritmos no poseen margen para la mejora.

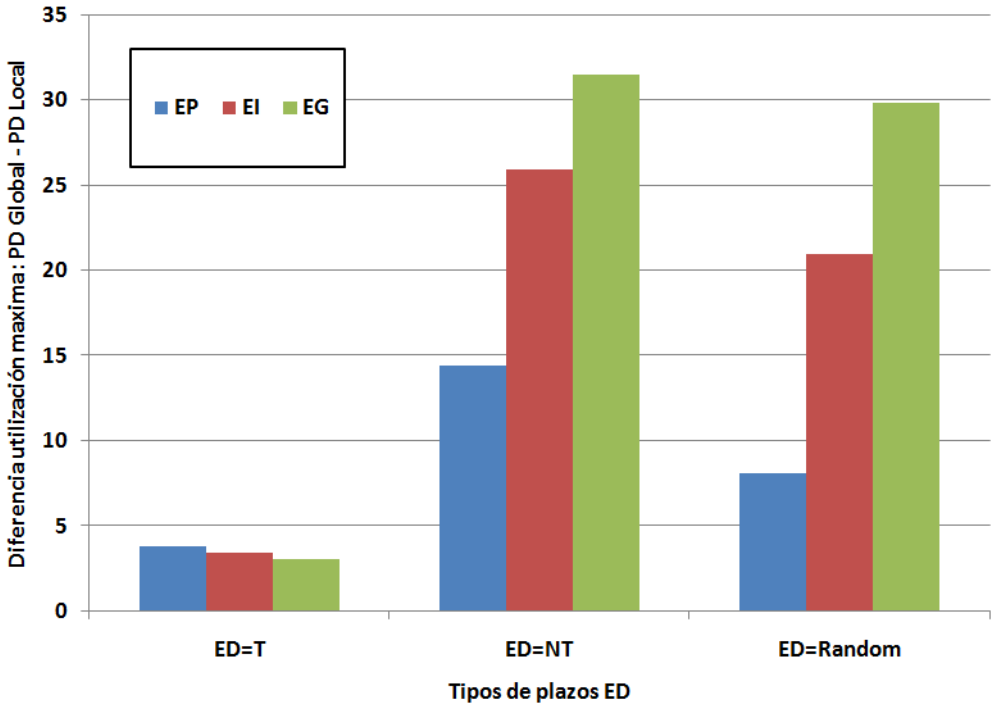

Figura 4.11: Relación tamaño-utilización máxima

- En los sistemas monoprocesadores EDF se cumple que si los plazos de las tareas eran iguales a sus respectivos periodos (D/T=1), se podía alcanzar un 100% de utilización. Ésto era cierto si las tareas eran independientes entre sí. Esto implica, entre otras cosas, que el sistema está formado por transacciones de una única tarea , o lo que es lo mismo, N_i=1 para todo *i*.

  En los resultados experimentales obtenidos se puede observar que, utilizando planificación con plazos globales, cuando la relación ED/T cumple $\frac{ED}{T} \geq N$, se obtienen utilizaciones cercanas al 100%, por lo que en cierta manera, parece una generalización del caso monoprocesador.
- El algoritmo HOSDA para plazos globales apenas consigue mejorar los resultados que obtiene en su asignación inicial, que es una asignación PD Global. Esta apreciación es especialmente válida para los casos en los que los plazos ED son mayores (ED=NT,ED=2NT). Ésto es debido posiblemente a que, partiendo de la asignación PD Global, ya no exista margen de mejora posible. Generalmente, el algoritmo HOSDA Global es mejor que el NPD Global, pero se dan casos concretos en los que ésto no es cierto.
- El algoritmo NPD Global es en media peor que el PD Global, aunque hay casos concretos en los que la asignación NPD supera a PD. En cualquier caso, con ambos se obtienen utilizaciones máximas similares. Por lo tanto el uso de la asignación normalizada sobre la proporcinal solo se justifica en un número reducido de casos.

5. Conclusiones

En el capítulo 1 planteamos los objetivos de este trabajo e introdujimos las técnicas de asignación de parámetros de planificación sobre las que íbamos a trabajar. Así, en el capítulo 2 se propuso una modificación a los algoritmos existentes de asignación de plazos locales de planificación, para la asignación de plazos globales.

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 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 3 se desarrolló una herramienta generadora automática de ejemplos.

De la comparación de los resultados que se exponen en el capítulo 4 se pueden extraer las siguientes conclusiones:

- Si planificamos utilizando plazos globales podemos obtener mejores utilizaciones en los recursos que si lo hiciéramos con plazos locales. De esta forma podemos ejecutar software con más carga computacional sin necesidad de aumentar la potencia en los recursos procesadores, con el consiguiente ahorro de costes.
- Sin embargo, para poder planificar con plazos globales, el sistema distribuido necesita poseer relojes perfectamente sincronizados, de más difícil implementación en este tipo de sistemas. Por lo tanto, siempre que el uso de este tipo de relojes sea posible o factible, la planificación con plazos globales es la recomendada si deseamos obtener altas utilizaciones en los recursos. En caso contrario, si el uso de estos relojes no es posible, habría que decantarse por una planificación con plazos locales, o incluso, con prioridades fijas, aunque con esta última se conseguiría el menor aprovechamiento en los recursos, a cambio de una implementación más sencilla.
- A la vista de los resultados, entre los tres algoritmos de asignación de plazos globales estudiados escogeríamos en primer lugar a HOSDA Global. La asignación PD Global puede ser suficiente en la mayoría de los casos, pero nunca podrá ser mejor que HOSDA Global, ya que ésta realiza un reparto proporcional como primera asignación.
- La asignación NPD Global es en media peor que la HOSDA Global. Sin embargo, pueden darse casos concretos en los que ésto no sea cierto. Por lo tanto, la aplicación del algoritmo NPD Global sólo queda recomendada como segunda opción si el algoritmo HOSDA Global no consiguió planificar el sistema en primer lugar.

5.1. Trabajo Futuro

A continuación nos aventuraremos a apuntar posibles trabajos para el futuro:

- Las técnicas holísticas de análisis de planificabilidad para sistemas distribuidos EDF (para plazos globales y locales) implementadas en la actualidad poseen una serie de limitaciones, tales como un límite en el tamaño del sistema a analizar (debido al uso de *arrays* de tamaño máximo fijo), 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 de los algoritmos de asignación de plazos de planificación en un mayor espectro de sistemas de tiempo real.
- Realizar un estudio del uso de relojes globales en sistemas distribuidos, y comprobar en qué casos es recomendable o factible su uso.
- El algoritmo heurístico basa su funcionamiento en un parámetro que se llama "**exceso"**. Existen dos definiciones para el "**exceso"**, pero en las implementaciones utilizadas se utiliza únicamente el "**exceso de tiempo de respuesta",** debido fundamentalmente a su menor coste computacional. 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 en los cálculos del algoritmo heurístico. 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 podría ser 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 (con plazos locales y/o globales), y otros planificados con prioridades fijas.
- En los sistemas reales puede darse el caso que necesitemos que una tarea posea un plazo de planificación determinado. En el futuro se podrían modificar las técnicas de asignación de plazos de planificación para que puedan actuar en consecuencia en estos casos. Como extensión, también se podrían adaptar las técnicas para el caso en el que las tareas tengan impuestos unos rangos válidos para sus plazos de planificación.
- Durante el transcurso del trabajo se ha apuntado la importancia del nivel de pesimismo en las herramientas de análisis. Para sistemas distribuidos EDF, en la actualidad solo están implementadas las técnicas holísticas. Un trabajo futuro podría ser el desarrollo e implementación de una técnica de análisis basada en *offsets* para sistemas planificados por plazos locales. Para el caso de planificación con plazos globales, la técnica ya está definida en [8], por lo tanto, sólo habría que implementarla en una herramienta de software, como por ejemplo en MAST.

Bibliografía

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

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

[3] 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

[4] 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

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

[6] 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.

[7] SPURI M. "Analysis of Deadline Scheduled Real-Time Systems". RR-772, INRIA, France, 1996

[8] 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. pp. 3-12

[9] GUTIÉRREZ GARCIA J.J, GONZÁLEZ 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.

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

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

[12] TINDELL K.W., BURNS A., WELLINGS A.J., "Allocating Real-Time Tasks. An NP-Hard Problem Made Easy".Real-Time Systems Journal, Vol.4, No.2, May 1992.

[13] KLEIN M., RALYA T., POLLAK B., OBENZA R, GONZALEZ HARBOUR M. "A practitioner's handbook for Real-Time Analysis". Kluwer, Academic Pub, 1993

[14] 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.

[15] MARTÍNEZ J.M.,GONZÁLEZ HARBOUR M. “RT-EP : A Fixed-Priority Real Time Communication Protocol over Standard Ethernet”.4th International Workshop on Real-Time Networks, RTN 05, Palma de Mallorca, Spain, July 2005.

[16] TINDELL K., BURNS A., 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.

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

[18] RIVAS CONCEPCION, J.M., GUTIÉRREZ J.J. (Director) “Estudio de la asignación de plazos en sistemas distribuidos de tiempo real con planificación EDF : propuesta de un algoritmo heurístico”. Proyecto Fin de Carrera, Universidad de Cantabria, 2008.

[19] RIVAS J.M., GUTIÉRREZ J.J., PALENCIA J.C., GONZALEZ HARBOUR M. “Optimized Deadline Assignment for Tasks and Messages in Distributed Real-Time Systems”. Internal Report, Dpto de Electrónica y Computadores, Universidad de Cantabria, 2009.

[20] TUCKER S.,DUFF R.A.,BRUCKARDT L.,PLOEDEREDER E.,LEROY P. “Ada 2005 Reference Manual. Languaje and Standard Libraries. International Standard ESO/IEC 8652:1995(E) with Technical Corrigendum 1 and Ammendment 1”. LNCS 4348, Springer, 2006.

[21] “Real-Time Specification for Java”. http://www.rtsj.org

[22] S.Ha.R.K (Soft Hard Real-time Kernel) . htttp://shark.sssup.it

[23] ERIKA (Embedded Real-time Kernel Architecture). http://erika.sssup.it

[24] DIEDRICHS C.,MARGULL U.,SLOMKA F.,WIRRER G. “An application-based EDF Scheduler for OSEK/VDX”. Proceedings of the conference on Design, Automation and Test in Europe, Munich (Germany), pp.1045-1050,2008

[25] DI NATALE M.,MESCHI A. “Scheduling Messages with Earliest Deadline Techniques”. Real-Time Systems, 20, pp.255-285, Kluwer Academic Publishers, 2001

[26] PEDREIRAS P.,ALMEIDA L. “EDF Message Scheduling on Controller Area Network. Computing & Control Engineering Journal, 13(4), pp. 163-170, 2002.