UNIVERSIDAD DE CANTABRIA
FACULTAD DE CIENCIAS
DEPARTAMENTO DE
ELECTRÓNICA Y COMPUTADORES

ANÁLISIS DE PLANIFICABILIDAD DE SISTEMAS DISTRIBUIDOS DE TIEMPO REAL BASADOS EN PRIORIDADES FIJAS

TESIS DOCTORAL

José Carlos Palencia Gutiérrez

Santander 1999

UNIVERSIDAD DE CANTABRIA
FACULTAD DE CIENCIAS
DEPARTAMENTO DE
ELECTRÓNICA Y COMPUTADORES

ANÁLISIS DE PLANIFICABILIDAD DE SISTEMAS DISTRIBUIDOS DE TIEMPO REAL BASADOS EN PRIORIDADES FIJAS

MEMORIA
presentada para optar al grado de
DOCTOR EN CIENCIAS FÍSICAS
por
José Carlos Palencia Gutiérrez
Licenciado en Ciencias, Sección Físicas
Especialidad de Electrónica

UNIVERSIDAD DE CANTABRIA
FACULTAD DE CIENCIAS
DEPARTAMENTO DE
ELECTRÓNICA Y COMPUTADORES

ANÁLISIS DE PLANIFICABILIDAD DE SISTEMAS DISTRIBUIDOS DE TIEMPO REAL BASADOS EN PRIORIDADES FIJAS

MEMORIA
presentada para optar al grado de Doctor en Ciencias Físicas por el Licenciado en Ciencias
José Carlos Palencia Gutiérrez

El Director,

Dr. Michael González Harbour
Profesor Titular de Universidad

DECLARO:
Que el presente trabajo ha sido realizado en el Departamento de Electrónica y Computadores de la Universidad de Cantabria, bajo mi dirección y reúne las condiciones exigidas a los trabajos de Doctorado.

Santander, Abril de 1999

Fdo. José Carlos Palencia Gutiérrez

Fdo. Michael González Harbour

A Tino y Margarita,
a Judith y Jezabel,
y a Luis Alberto.

Es curioso como algunas situaciones, aparentemente intrascendentes, pueden cambiar el rumbo de tu vida. Supongo que mi inicial admiración por José María Drake despertó mi interés por trabajar en este grupo. A él quiero dirigir en primer lugar mi agradecimiento más sincero.

También quiero agradecer conjuntamente a José María y a Michael su empeño en que esta tesis haya llegado a término. Especialmente por su interés y esfuerzo personales para conseguir mi sostenimiento económico, en varias situaciones ciertamente difíciles para mí.

Quisiera mostrar públicamente mi agradecimiento a todas aquellas personas que han contribuido a crear a mi alrededor un ambiente agradable. Sin ellos seguramente no hubiera encontrado la inspiración suficiente para superar los momentos difíciles, tanto desde el punto de vista emocional como intelectual. Afortunadamente para mí, debería citar aquí a mucha gente, pero lógicamente no es posible. En primer lugar a mis padres, por enseñarme lo que es realmente importante en la vida, y a mis hermanos. En segundo lugar, a mis amigos y compañeros de trabajo, especialmente a Gustavo y Mercedes, Felipe y Sol, Vicente, Borja, Carmen, Javi, Kakel, Rafa, Romero, Julio, Jorge, ... y un montón de gente más. También quiero citar expresamente a mis amigos y compañeros de carrera, con los que compartí tantas horas de estudio y risas: Pedro, Jose, Manolo, César y Alex. No sería justo si no agradeciera especialmente a Luis Alberto su amistad, interés y sentido común. En último, pero más importante lugar, a Jezabel y a mi hija, Judith, que llenan mi vida de felicidad.

También quisiera expresar mi agradecimiento y admiración hacia aquellos que me han hecho disfrutar con sus trabajos, especialmente a Michael González Harbour, Neil Audsley, Ken Tindell y John Lehoczky.

Por cierto, no se me deberían olvidar Beethoven y Solti. Los tres hemos compartido muchas noches en vela rodeados de ecuaciones.

Gracias a todos.

La presente Tesis Doctoral, ha sido desarrollada en el marco de los siguientes proyectos de investigación:

"Extensión de la teoría RMA al diseño y análisis de sistemas de tiempo real gobernados por eventos".
Proyecto del Plan Nacional de Investigación en Automatización Avanzada y Robótica, 1992-1994, Ref. ROB91-0288.

"Herramientas para el análisis de tiempo real de sistemas distribuidos para automatización industrial".
Proyecto del Plan Nacional de Investigación en Tecnologías Avanzadas de la Producción, 1994-1997, Ref. TAP94-0996.

"Metodología para el análisis y diseño orientado a objetos de sistemas distribuidos de tiempo real estricto".
Proyecto del Plan Nacional de Investigación en Tecnologías Avanzadas de la Producción, 1997-1999.

Indice de contenidos

Resumen

Palabras clave: *Tiempo real, sistemas distribuidos, análisis de planificabilidad, tiempo de respuesta, planificación por prioridades.*

En la tesis doctoral presentada en esta memoria se estudia el análisis de sistemas de tiempo real en los que se debe verificar el cumplimiento de ciertos requerimientos temporales, principalmente referidos a plazos máximos de ejecución. Esta verificación se realiza mediante el análisis de planificabilidad basado en el cálculo de tiempos de respuesta de peor. El estudio se centra en los sistemas multiprocesadores y distribuidos gobernados por eventos y planificados mediante políticas basadas en asignación de prioridades fijas. Dentro de este ámbito no se conocen técnicas exactas de análisis, de manera que parte de la investigación se ha orientado a la búsqueda de técnicas analíticas aproximadas que verifiquen suficientemente el cumplimiento de los requisitos temporales. El trabajo presentado en esta tesis se dirige básicamente a esa búsqueda. Por un lado, se centra en la formalización y optimización de las técnicas de análisis previamente desarrolladas por otros autores, mediante modelos que aproximan el comportamiento real de los sistemas distribuidos de forma que sean analizables, aunque introduciendo con ello cierto pesimismo, que hace que el análisis no resulte exacto, pero sí suficiente. Por otro lado, la tesis se centra en la búsqueda de un nuevo modelo que permita reducir el pesimismo en el análisis. El modelo que se propone es el basado en tareas con offsets dinámicos, mediante el que se consiguen mejoras sustanciales frente a los modelos anteriormente utilizados. También se optimiza el análisis al considerar las relaciones de precedencia en la ejecución de tareas dentro del sistema distribuido. La inclusión de estas relaciones de precedencia cuando deducimos las expresiones analíticas que conducen a la obtención de los tiempos de respuesta hace que se reduzca fuertemente el pesimismo. Este modelo se ha encontrado también adecuado para el análisis de tareas que se suspenden o para el análisis de tareas con prioridades variantes, dentro de sistemas multiprocesadores y distribuidos.

1. Sistemas distribuidos de tiempo real.

1.1. Introducción

Tradicionalmente se ha considerado que un sistema computacional funciona correctamente cuando la solución que obtiene es correcta desde el punto de vista lógico. Esta definición, si bien es válida para sistemas de cálculo convencional, no es suficiente cuando estamos considerando sistemas que interactúan fuertemente con el entorno. Este tipo de sistemas, normalmente dedicados a tareas de monitorización o control, suelen estar formados por un conjunto de recursos tales como sensores, unidades de cómputo y actuadores que deben trabajar de forma coordinada para realizar la labor encomendada. La cooperación entre diferentes recursos obliga, no sólo a que cada operación sea correcta, sino a que se realice en los instantes oportunos. Este tipo de sistemas a los que es preciso imponer restricciones temporales se denominan sistemas de tiempo real [STA88].

Desde el punto de vista de los requerimientos temporales impuestos, un sistema se pueden clasificar en tres categorías diferentes:

- Sistemas de tiempo real estricto: el incumplimiento de alguno de los requerimientos puede tener efectos catastróficos sobre el sistema que controla, por lo que es imprescindible evitar esta situación. El sistema debe cumplir todos los requerimientos especificados.

- Sistemas de tiempo real no estricto: el sistema puede incumplir ocasionalmente alguno de los requerimientos impuestos. Ese incumplimiento produce una disminución en las prestaciones o calidad de la respuesta pero el funcionamiento se puede considerar todavía correcto.

- Sistemas sin requerimientos de tiempo real: no importa el tiempo que se tarda en obtener una respuesta. La rapidez en la respuesta sería deseable simplemente a efectos de mejorar el tiempo de respuesta promedio o la capacidad media de procesado.

Evidentemente pueden considerarse sistemas en los que existan simultáneamente requerimientos temporales estrictos y no estrictos junto con tareas de cálculo convencional.

Normalmente los requerimientos temporales que se imponen a un sistema se refieren al tiempo de respuesta de cada una de las tareas que lo componen. El más usual, y sobre el que se centra gran parte del análisis de tiempo real, es el concepto de plazo de una tarea, considerado como el máximo tiempo necesario para obtener una respuesta. Otros tipos de requerimientos se refieren a la separación entre dos respuestas consecutivas, a la capacidad mínima de procesado, etc. En esta tesis nos centraremos exclusivamente en los plazos de respuesta en sistemas de tiempo real estricto.

Las técnicas utilizadas comúnmente para verificar el cumplimiento de esos requerimientos temporales se pueden clasificar en dos tipos diferentes:

- Técnicas *off-line*: antes de que el sistema de tiempo real esté operando se preveen y analizan los posibles comportamientos, de forma que estimando cotas superiores de los tiempos de respuesta se puede verificar el cumplimiento de los plazos de respuesta o, en caso de que pudiera incumplir alguno, cuantificar el grado de incumplimiento.

- Técnicas *on-line*: en el instante en que una nueva tarea está lista para ejecutar, se realiza el análisis de planificabilidad considerando la nueva tarea junto con el conjunto de tareas ya existente. Si el nuevo sistema formado siguiera siendo planificable, entonces se acepta la tarea para ejecutar; en caso contrario, se rechaza. Estas técnicas presentan el inconveniente de que pueden rechazar tareas de gran importancia, sin que este hecho se pueda detectar hasta el momento de la ejecución.

El análisis desarrollado en esta tesis se enmarca en el primer tipo de técnicas. El hecho de tener que garantizar a priori plazos en la respuesta de un sistema hace necesario imponer ciertos requisitos sobre la especificación e implementación del mismo. En particular, es

preciso que el comportamiento temporal del sistema de tiempo real sea predecible. Esto quiere decir que todos los componentes (tanto *hardware* como *software*) del sistema de tiempo real deben ser conocidos y estar perfectamente especificados a priori.

1.2. Modelo del sistema

La mayoría de los sistemas de control están basados en uno o varios procesadores conectados entre sí a través de un bus o una red de comunicación y que interactúan con el entorno. Esta interacción se realiza mediante sensores que dan información del estado actual del sistema a controlar y mediante actuadores que pueden modificar o actuar sobre algún aspecto del mismo. La implementación del sistema de control determinará como interactuarán los diferentes elementos que lo forman, clasificándose en:

- Sistemas gobernados por eventos: las acciones encargadas del control del sistema se activan con la ocurrencia de eventos, de forma que la llegada de esos eventos es la que caracteriza el flujo del programa de control.

- Sistemas gobernados por tiempo: las acciones de control son activadas por el sistema ejecutivo en instantes de tiempo predeterminados. Estos instantes de activación se producen normalmente a intervalos de tiempo regulares, de forma que el sistema está formado por acciones activadas periódicamente.

- Sistemas gobernados por el paso de mensajes: la activación de las acciones se produce como consecuencia de la llegada de algún mensaje a través del sistema de comunicación. Esta es la forma usual de activación de acciones cooperantes en sistemas distribuidos, en los que una tarea comienza a ejecutar cuando otra tarea (normalmente ejecutando en otro procesador) ha finalizado previamente. Normalmente, la activación de la primera tarea cooperante está gobernada por un evento o por tiempo y la activación de las siguientes mediante paso de mensajes entre los procesadores involucrados.

En esta tesis modelaremos los sistemas de tiempo real suponiendo que su funcionamiento se basa en esta última política de control, ya que las dos primeras serían un caso particular de ese modelo general.

1.2.1 Caracterización de eventos

La sensorización de un sistema físico se identifica con el concepto de evento, que dispara el proceso de control. Desde el punto de vista de cada elemento que forma el sistema de control (en particular los procesadores donde se ejecutan las acciones oportunas) debemos caracterizar la ocurrencia de eventos y la detección de esa ocurrencia. En cuanto a quién es la fuente que origina el evento, clasificaremos los eventos en tres tipos diferentes:

- Eventos temporizados: producidos por temporizadores o relojes propios del sistema de control, de forma que el sistema ejecutivo es el encargado de hacer notar la ocurrencia de esos eventos.

- Eventos externos: son los eventos originados en el sistema físico que se está controlando y que significan el desencadenamiento de una serie de acciones dentro del sistema de control como respuesta a ese evento.

- Eventos internos: son los eventos producidos dentro del propio sistema de control como consecuencia de la necesaria sincronización entre la ejecución de acciones en diferentes procesadores en respuesta, normalmente, a un mismo evento externo. En nuestro modelo identificaremos estos eventos internos con los mensajes entre procesadores.

La sensorización de cada evento, ya sea interno o externo, caracteriza la activación de la correspondiente acción asociada, de forma que ésta se puede producir:

- A requerimiento del sensor: El sensor genera asíncronamente un requerimiento de interrupción a algún procesador del sistema, de manera que éste invoca un manejador de excepciones con el código oportuno para el procesado de ese evento. Es un

mecanismo de sensorización muy eficiente, pero hay que tener en cuenta la dificultad de su programación.

- A requerimiento del procesador: Hay sensores que no son capaces de generar una interrupción o que ni siquiera lo necesitan. En ese caso, el procesador debe requerir, normalmente de forma periódica, el estado del sensor o dispositivo correspondiente. Este método es más sencillo de implementar pero en algunos casos es menos eficiente que el método anterior, ya que debe tenerse en cuenta la incertidumbre introducida por el periodo de muestreo y la capacidad del dispositivo de dar medidas de forma continua o discreta.

Independientemente de cómo se genera, la llegada de un evento dispara la ejecución de un conjunto de actividades en el sistema procesador. Esto significa que debemos caracterizar el patrón de llegadas u ocurrencia de todos los eventos producidos en el sistema. Este patrón de llegadas puede ser de los tipos siguientes:

- Periódico: los eventos se generan a intervalos de tiempo regulares. La duración de ese intervalo se denomina periodo de activación, *T*, y es suficiente para caracterizar la llegada de eventos.

- Esporádico: los eventos se generan a intervalos no regulares de tiempo, pero con tiempo mínimo entre la llegada de dos eventos consecutivos. Dado que verificaremos el plazo máximo de respuesta al evento, nos interesa conocer cuál es ese tiempo mínimo, al que identificaremos como *T*.

- Limitado: los eventos se generan a intervalos no regulares de tiempo, pero tienen garantizada una densidad máxima de ocurrencia, generalmente expresada como máximo número de eventos n_{max} en un intervalo determinado de tiempo *T*. Desde el punto de vista del análisis de peor caso, podemos modelar este patrón de generación en base a eventos esporádicos (cada uno equivalente a n_{max} eventos) con tiempo mínimo entre llegadas equivalente a *T*.

- Ilimitado: los eventos se generan a intervalos no regulares de tiempo, pero no existe un tiempo mínimo entre llegadas ni una densidad máxima de ocurrencia. Virtualmente podrían llegar infinitos eventos simultáneamente, por lo que no se puede garantizar un tiempo de respuesta limitado. Evidentemente ningún sistema es capaz de seguir instantáneamente un numero infinito de eventos, así que realmente se trataría de un patrón limitado de llegadas, aunque con un límite efectivamente muy grande.

1.2.2. Respuesta a un evento

Cada evento que llega al sistema de control desencadena una serie de acciones en respuesta. Esas acciones normalmente se asocian a tareas cuyo código se activa a instancias de la llegada de eventos (externos, internos o temporizados). Para poder garantizar *off-line* un tiempo finito de respuesta es preciso que cada tarea del sistema tenga definido un tiempo máximo de ejecución, *C*, también llamado tiempo de ejecución de peor caso, entendido como el tiempo máximo que tarda en ejecutar el código correspondiente suponiendo que no hay ningún tipo de interferencia (por ejemplo de otras tareas) en dicha ejecución. Aunque no es tarea fácil en absoluto, en esta tesis supondremos que se conocen los tiempos de ejecución de peor caso de todas las acciones que ejecutan en el sistema. Este es un requisito general en todos los sistemas de tiempo real estricto, sea cual sea su método de planificación. Técnicas que estudian la obtención de estos tiempos de ejecución de peor caso se pueden encontrar en [CHA93] [CHA94] [JOU95] [LIM95].

En sistemas de control complejos se suele asociar la respuesta a un evento a la ejecución de varias tareas, incluso en varios procesadores, que coordinan su ejecución para obtener la respuesta total del sistema. Esta asociación de tareas se conoce como cadena de respuesta al evento y debe ser totalmente conocida en sistemas en los que sea preciso obtener una garantía de tiempos de respuesta de peor caso en tiempo de compilación. En esta tesis consideraremos una cadena de respuesta fija para cada posible evento del sistema, tanto desde el punto de vista de las tareas que lo componen como del procesador en que ejecuta cada tarea.

Asimismo, consideraremos un modelo lineal de la cadena de respuesta: esto significa que cada tarea se activa por un único evento (ya sea interno, externo o temporizado) y puede

generar un único evento interno a su finalización. En [GUT95D] se muestran modelos no lineales, más flexibles para los sistemas de tiempo real; sin embargo, nosotros mantendremos el modelo lineal, ya que el análisis para esos modelos no lineales se puede descomponer en análisis de varios modelos lineales [GUT95D].

Un efecto que también debemos considerar a la hora de caracterizar la generación de eventos es el retraso en la activación de una tarea *(release jitter*). Entre el instante en que ocurre el evento y el instante en que el sistema se da por enterado de esa ocurrencia puede haber cierto desfase, especialmente si la sensorización del evento se realiza a requerimiento del procesador. También puede haber cierto error en la generación o en la anotación de ese instante debido a la imprecisión del reloj, especialmente en eventos temporizados. Al igual que antes, en técnicas *off-line* debemos garantizar un valor máximo para esa magnitud, que identificaremos como *J*.

Sin embargo, tal como veremos posteriormente, existe otra fuente importante de retraso en sistemas cuyas tareas tengan relaciones de precedencia, esto es, sistemas en los que cada tarea se activa mediante un evento interno generado a la finalización de una tarea previa en la cadena de respuesta. En este caso, el instante en que se produce la activación de una tarea respecto de la llegada del evento externo puede retrasarse dependiendo de cuál sea el tiempo de respuesta de la tarea precedente. El cálculo del máximo retraso originado por ese motivo es bastante más complicado, y de hecho, es lo que dificulta el análisis para sistemas distribuidos con relaciones de precedencia, respecto al análisis realizado en sistemas en los que no existan restricciones de ese tipo. Esto hace que el término correspondiente a este retraso no lo consideraremos como parte de la especificación del sistema, sino como parte del cálculo de los tiempos de respuesta, tal como veremos en capítulos posteriores de esta tesis.

1.3. Sistemas multiprocesadores y distribuidos

1.3.1 Red de comunicación

Cuando la respuesta a un evento externo se ejecuta en varios procesadores debemos considerar los mecanismos mediante los que se coordina la ejecución de las diferentes tareas involucradas en esa respuesta. Normalmente este mecanismo es el de paso de mensajes:

cuando una tarea finaliza su ejecución envía un mensaje que activa a la tarea siguiente en la cadena de respuesta.

En general, un sistema distribuido está compuesto por múltiples recursos procesadores conectados entre sí por uno o varios recursos de comunicación (buses, redes locales, líneas serie, etc). En sistemas de tiempo real en los que hay que garantizar un plazo máximo de respuesta es preciso que la comunicación se realice de forma fiable y en un tiempo máximo acotado, por lo que la planificación de los mensajes a transmitir debe estar bien establecida.

Aunque la mayoría de las redes de comunicación no se adaptan bien a las comunicaciones de tiempo real y no soportan una planificación de los mensajes basada en prioridades, sí existen algunas redes de comunicación estándares que se han utilizado para llevar a cabo comunicaciones de tiempo real estricto, tales como el *token-bus* [STR88], FDDI [AGR91], el bus CAN [TIN94E], etc. También se han desarrollado algunas redes de comunicación no estándares para hacer una planificación expulsora basada en prioridades. En [GUT95D] se implementan sistemas de comunicación de tiempo real mediante líneas serie (RS-232 y RS-485) multipunto y buses VME basados en la interfaz de colas de mensajes definidas en el estándar POSIX.1b [POS93].

Evidentemente, debemos incluir el impacto que tiene el envío de esos mensajes en la respuesta total del sistema. La cadena de respuesta a un evento se modela entonces como una sucesión de tareas y mensajes que se ejecutan o envían secuencialmente. El sistema de comunicaciones debe ser también predecible, asegurando un tiempo acotado en la transmisión de cada mensaje. Cada mensaje tendrá asociada, por tanto, una longitud máxima identificada también como l_m.

Adicionalmente supondremos que cada mensaje a transmitir se puede partir en paquetes de tamaño fijo, de forma que la comunicación entre dos tareas involucra la transmisión de un cierto número de paquetes. Este proceso de comunicación entre tareas se realiza en las siguientes etapas:

- Generación y encolado del mensaje. La tarea que envía el mensaje debe componer el mensaje a transmitir y, en caso necesario, partirlo en paquetes de tamaño fijo.

Además, cada paquete transmitido por la red necesitará información adicional para el encaminamiento del mensaje hasta el destino. Finalmente, cada paquete debe ser puesto en la cola de transmisión.

- Acceso al dispositivo de comunicación. Una vez encolado cada paquete, deberá esperar a que el dispositivo de comunicación quede libre y listo para realizar la transmisión del paquete.

- Transmisión del mensaje por el enlace físico. Desde el recurso procesador origen al recurso procesador donde esté alojada la tarea destino. El tiempo de transmisión de cada paquete vendrá determinado por su longitud y por la velocidad de transmisión del dispositivo.

- Recepción y composición del mensaje. Finalmente, y una vez que se hayan recibido todos los paquetes correspondientes deberá componerse el mensaje original y notificarse a la tarea destino.

A la hora de analizar el sistema de tiempo real deberá considerarse el efecto de cada una de estas etapas en la respuesta completa. En muchos sistemas se pueden considerar los tiempos de generación y encolado del mensaje como parte de la ejecución de la tarea emisora, de forma que se deberán sumar estos dos tiempos al tiempo de ejecución de peor caso de la tarea origen. En otros sistemas, sin embargo, parte de las operaciones de generación y encolado las ejecuta alguna otra tarea, de prioridad generalmente más alta, de forma que habría que considerar esa tarea adicional en el análisis. De igual forma, el tiempo de recepción y composición del mensaje se añadirán al tiempo de ejecución de peor caso de la tarea destino o, en caso necesario, se considerará otra tarea, si realiza parte de esas operaciones. El impacto del acceso al dispositivo y de la propia transmisión del mensaje es más complejo, ya que debe tenerse en cuenta la influencia de otros mensajes que quieran transmitirse por el mismo medio de comunicación. La sección 2.3.1 de esta tesis muestra en detalle el cálculo de ese impacto.

Cuando se dispone de una red de comunicación apta para tiempo real, el acceso y la transmisión de un mensaje por una red de comunicación se puede analizar de una forma muy

parecida a como se analiza el acceso y ejecución de una tarea en un procesador. Esto hace que, desde el punto de vista del análisis, no distingamos entre la ejecución de una tarea en un procesador o la transmisión de un mensaje por un dispositivo de comunicación. Por ese motivo, especificaremos cada mensaje mediante su longitud máxima l_m o indistintamente, mediante su tiempo de transmisión de peor caso C_m, calculado suponiendo que tuviera el uso exclusivo del dispositivo de comunicación.

A efectos de uniformizar el modelo, nos referiremos a las tareas que se ejecutan en un procesador o a los mensajes transmitidos por una red de comunicación como acciones a realizar dentro del sistema de tiempo real. Asociaremos con un evento interno la generación de un mensaje, de forma que ese evento dispara la ejecución de una acción en el recurso correspondiente (transmisión del mensaje por el dispositivo de comunicación). De igual forma, asociaremos la recepción del mensaje con otro evento interno que dispara la ejecución de la tarea receptora del mensaje. El modelo de sistema gobernado por paso de mensajes consistirá por tanto en una secuencia de acciones (tareas en procesadores o mensajes en redes de comunicación) conectados entre sí (activados) mediante eventos internos y requiriendo cada una de ellas un tiempo de cómputo C, correspondiente al tiempo de ejecución de peor caso (o al tiempo de transmisión de peor caso, si la acción es un mensaje).

Por extensión, se considera como acción periódica aquella que se activa por un patrón periódico de eventos externos y como acción esporádica aquella activada por un patrón esporádico de eventos. Cada acción a_i se especifica, en resumen, mediante los parámetros:

- C_i: tiempo de ejecución de peor caso si se trata de una tarea en un procesador o tiempo de transmisión de peor caso si es un mensaje en una red de comunicación.

- T_i: periodo de activación (o tiempo mínimo entre llegadas) del evento externo cuya llegada originó la cadena de respuesta a la cual pertenece la acción.

- J_i: retraso máximo en la activación de la acción.

1.3.2 Modelo general para transacciones de tiempo real

Un modelo que nos será muy útil en el análisis de los sistemas distribuidos de tiempo real es el de transacciones de tiempo real [TIN94B]. Una transacción es una entidad que agrupa tareas activadas con igual periodo y que deben satisfacer ciertas restricciones en cuanto a sus instantes de activación. Cada transacción Γ_i es activada por una secuencia periódica de eventos externos con periodo T_i y se compone de un conjunto de m_i tareas o mensajes. Cada ocurrencia del evento periódico externo originará una instancia (*job*) de la transacción asociada. Cada tarea[1] se puede activar a partir del momento en que haya transcurrido un intervalo de tiempo (llamado *offset*) desde la llegada del evento externo. La activación se produce desde ese instante con un retraso aleatorio, que puede estar comprendido entre 0 y el retraso máximo, J_i.

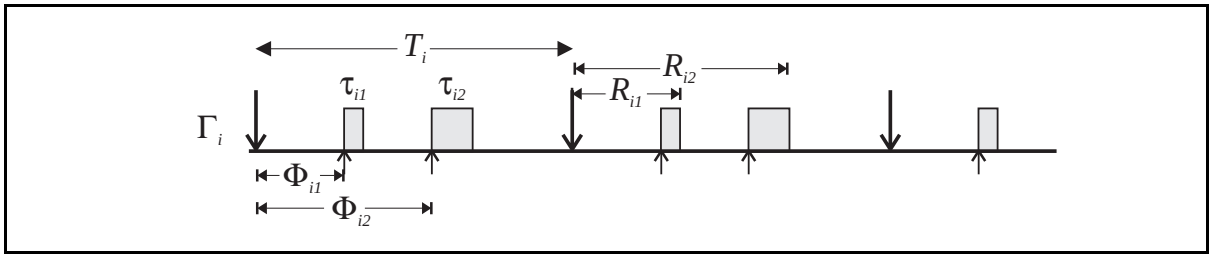


Figura 1-1. Transacción de tiempo real

La Figura 1-1 muestra un ejemplo de transacción, donde el eje horizontal representa el tiempo, y donde suponemos que los retrasos máximos son nulos (J_i=0). Las flechas descendentes representan la llegada de un evento externo asociado a la transacción mientras que las flechas ascendentes representan la activación de las tareas y los rectángulos sombreados la ejecución de las mismas. Nótese que en este modelo no hay relaciones de precedencia explícitas entre las tareas; cada tarea se activa en un instante igual a la llegada del evento externo más el *offset*, independientemente de que las tareas de su misma transacción y menor *offset* hayan finalizado o no.

Identificaremos cada tarea mediante dos subíndices: el primero identifica la transacción a la que pertenece, y el segundo la posición que ocupa dentro del conjunto de tareas de su

[1] A partir de aquí, siempre que hablemos de una tarea nos estaremos refiriendo a una tarea en un procesador o a un mensaje en una red de comunicación

transacción, al ordenarlas en orden creciente de *offset*. De esta forma, con τ_{ij} denotaremos la *j*-ésima tarea perteneciente a la transacción Γ_i, con una fase de activación Φ_{ij} y un tiempo de ejecución de peor caso C_{ij}. Permitiremos además que cada tarea puede sufrir un retraso máximo denotado por J_{ij}. Esto quiere decir que la activación de la tarea puede producirse en cualquier instante dentro del intervalo determinado por los instantes $t_0 + \Phi_{ij}$ y $t_0 + \Phi_{ij} + J_{ij}$, donde t_0 es el instante en que llegó el evento externo. Para cada tarea τ_{ij} definiremos su tiempo de respuesta como la diferencia entre su tiempo de finalización y el instante en que ocurrió el evento externo asociado. El tiempo de respuesta de peor caso será R_{ij}.

Si la fase de activación Φ_{ij} de cada tarea es fija, estaremos hablando de transacciones con *offsets* estáticos. Por el contrario, si la fase de una tarea puede variar en un intervalo $\Phi_{ij,min}$ y $\Phi_{ij,max}$ estaremos hablando de transacciones con *offsets* dinámicos. Tal como se verá en el capítulo 4 de esta tesis, la inclusión en el modelo de *offsets* dinámicos permite modelar y analizar sistemas de tiempo real con relaciones de precedencia, tales como los gobernados por paso de mensajes.

En sistemas multiprocesadores y distribuidos es usual que el sistema pueda ser modelado mediante "transacciones" compuestas por varias tareas, al estilo del modelo computacional descrito anteriormente. Por ejemplo, en un sistema que siga la arquitectura cliente-servidor, una tarea cliente se activa por la llegada de un evento externo y requiere servicios de uno o varios servidores, quizás en diferentes procesadores. Esta tarea cliente puede modelarse mediante una transacción. Cada sección de código entre requerimientos de servicio se modela como una tarea de la transacción y cada porción de ejecución de un servidor en otro procesador se modela también como otra tarea de la misma transacción. Exactamente igual se modelan los mensajes transmitidos entre distintos procesadores. En este modelo, cada tarea τ_{ij} se activa con la finalización de la tarea previa perteneciente a la misma transacción, τ_{ij-1}.

También podemos modelar mediante transacciones con *offsets* dinámicos, sistemas con tareas que se suspenden a sí mismas. Por ejemplo, una tarea puede ejecutar durante algún tiempo, y entonces suspender su ejecución para leer datos de un disco. Supongamos que la duración de la suspensión está comprendida entre S_{min} y S_{max}. Podríamos modelar entonces esa tarea como una transacción compuesta por dos tareas: tarea τ_{i1} correspondiente al código

anterior a la suspensión, y tarea τ_{i2} para el código posterior a la suspensión. El instante de activación para la segunda tarea depende del instante de finalización de la primera tarea más el tiempo que dure la suspensión.

Consecuentemente, modelaremos las actividades que se desarrollan en un sistema distribuido como transacciones, considerada cada una de ellas compuesta por una cadena de acciones, al modo de las tareas en las transacciones originales. Cada acción de una transacción representa una tarea o una sección de una tarea ejecutando en un procesador, o un mensaje transmitido por un canal de comunicación, y se activa cuando se completa la ejecución de la acción previa en la transacción. Esto nos permite modelar la activación de una tarea por la llegada de un mensaje o la transmisión de un mensaje por la finalización de la ejecución de una sección de código

1.4. Planificación de sistemas de tiempo real

Un aspecto fundamental a tratar en el desarrollo de sistemas de tiempo real es el de la compartición de recursos. Dado que la generación de diferentes eventos externos se produce de forma independiente, pueden existir en el sistema de control varias cadenas de respuesta ejecutándose simultáneamente. Esto quiere decir que simultáneamente varias tareas pueden necesitar hacer uso del mismo recurso. Si el recurso permite varios usuarios, tal como las memorias multipuerta, no hay ningún problema; sin embargo, la situación normal es que solamente una de las tareas pueda hacer uso del recurso. Un claro ejemplo de este tipo de recursos de uso exclusivo es el procesador.

La planificación del sistema de tiempo real consiste en la definición de las reglas de uso de cada uno de los recursos disponibles. Un sistema de tiempo real se considera planificable si, en función de la política de planificación elegida, es capaz de satisfacer todos los requisitos temporales impuestos.

Las políticas de planificación deben además considerar los siguientes aspectos:

- Ser predecible, con objeto de asegurar tiempos de ejecución finitos. Para ello, debe ser posible analizar los efectos de las interferencias que el propio planificador tendrá sobre el sistema. Tales efectos deberán ser mensurables.

- Ser capaz de gestionar el uso de diferentes recursos compartidos. Diferentes recursos pueden requerir diferentes políticas de planificación, incluso formando parte del mismo sistema de tiempo real.

- Debe garantizar el tratamiento de eventos tanto periódicos como no periódicos, incluso con patrones de llegada ilimitados.

- Garantizar también la ejecución de tareas sin requerimientos temporales, procurando además que los tiempos de respuesta sean reducidos.

- Alcanzar utilizaciones de recursos altas, sobre todo del procesador.

- Que sea sencillo de implementar en aplicaciones reales. Preferiblemente que esté disponible comercialmente.

El único mecanismo fiable para determinar si la política o políticas de planificación elegidas aseguran el cumplimiento de los plazos consiste en la realización a priori de un test de planificabilidad, ya que la simulación no garantiza que se haya comprobado el comportamiento en cualquier situación [DIJ72] [XU93]. Un posible test de planificabilidad para sistemas de tiempo real consiste en el cálculo de los tiempos de respuesta de peor caso para cada una de las tareas que conforman el sistema. Si los tiempos de respuesta de peor caso de las tareas son siempre menores que los plazos de ejecución asociados a las mismas quiere decir que el sistema verificará en cualquier condición los requerimientos impuestos. Dicho test de planificabilidad debe ser cuanto menos suficiente, esto es, si el test dice que el sistema cumple los requerimientos temporales, entonces inexorablemente los cumple bajo cualquier situación. Un posible test de planificabilidad suficiente pasa por el cálculo de cotas superiores de los tiempos de respuesta de las tareas, mediante los cuales se puede garantizar el correcto funcionamiento del sistema. Un test suficiente puede ser pesimista, en el sentido de que puede considerar como no planificable un sistema que realmente si lo es, como

consecuencia de que los tiempos de respuesta de peor caso se hayan estimado por exceso. Hay tests de planificabilidad exactos para sistemas monoprocesadores; sin embargo, para sistemas multiprocesadores o distribuidos sólo existen tests exactos en ciertos casos, y a menudo resultan impracticables. Para los sistemas distribuidos gobernados por eventos no existen tests de planificabilidad exactos conocidos y todos los test aplicables a ellos son tests suficientes.

Evidentemente, los recursos más importantes y sobre los que se centra la mayor parte del esfuerzo investigador son el procesador y la red de comunicación. Dado que ambos recursos se pueden considerar con un comportamiento similar (tal como hemos señalado en la sección 1.3.1), centraremos el estudio en el procesador, esto es: bajo qué condiciones el procesador es asignado a una tarea (o una red de comunicación a un mensaje).

1.4.1 Planificación estática

El mecanismo más sencillo es la planificación estática, también conocida como ejecutivo cíclico [BAK88]. Esta planificación, para tareas periódicas, se realiza en tiempo de compilación, esto es, una vez conocido el sistema y antes de su ejecución. A cada tarea del sistema se le asignan rodajas temporales durante las cuales puede ejecutar, según una tabla (plan estático) que indica los instantes en los que cada tarea debe ponerse en ejecución y cuando debe finalizar, de forma que se vayan garantizando los plazos de ejecución de todas las tareas. Esta tabla se va siguiendo cíclicamente durante la ejecución del sistema [LOC92] [BUR90].

El funcionamiento del planificador es muy sencillo, ya que sólo tiene que ir leyendo las entradas correspondientes en el plan de ejecución. Desde este punto de vista es además muy eficiente en tiempo de ejecución, ya que la carga que supone sobre la ejecución de las tareas es mínima.

El principal inconveniente de este mecanismo es que el *software* debe partirse en secciones de código tales que puedan ser ejecutadas en una rodaja. Este proceso puede ser bastante costoso y de difícil mantenimiento y no siempre es posible [BUR90].

Otra desventaja es el hecho de que necesitamos elaborar el plan de ejecución, lo que supone construir una tabla en la que figuren todos los posibles estados de ejecución de las tareas. Esto significa que debe hacerse ese plan para un intervalo temporal igual al mínimo común múltiplo de los periodos de todas las tareas, lo cual puede dar como resultado una tabla de tamaño intratable. Esto se puede resolver reduciendo convenientemente los periodos de activación de las tareas, aunque tendría la desventaja de cargar el procesador más de lo que realmente necesita. Además, cualquier cambio en una tarea requiere tener que volver a construir nuevamente toda la tabla y, lo que es peor, a tener que partir las tareas de una forma diferente a como se hizo previamente. No se sigue el principio de independencia entre la estructura lógica del programa y su planificación.

1.4.2 Planificación dinámica por prioridades fijas

Una posibilidad bastante más adecuada es la planificación en tiempo de ejecución o planificación dinámica. En ella, los posibles problemas de contención de recursos se resuelven en el instante en que aparecen, por lo que se elimina la necesidad de establecer un plan de ejecución previo. Evidentemente, esto hace más complejo el funcionamiento del planificador, aunque el sistema resulta más fácilmente mantenible y entendible, al no tener que hacer la partición del código y mantener, por tanto, la independencia entre la estructura lógica del programa y su planificación.

En esta política de planificación se sigue el concepto de tarea como *thread* de ejecución. A cada tarea se le asigna una prioridad y, en función de ella, se resuelven los conflictos de utilización del procesador. Cuando hay varias tareas que quieren ejecutar, el planificador elige de entre todas ellas aquella con prioridad más alta y le asigna el uso del procesador.

Si la prioridad de una tarea no cambia una vez asignada, hablaremos de prioridades fijas. Por contra, si la prioridad puede variar en tiempo de ejecución, en función del estado de operación del sistema, hablaremos de prioridades dinámicas. La asignación dinámica es más eficiente que la asignación estática, desde el punto de vista de que consigue mayor utilización de los recursos. Sin embargo, la implementación del planificador es mucho más

sencilla cuando se utiliza la asignación estática y es el que incorporan la mayoría de los planificadores comerciales existentes.

Dentro de las políticas de planificación por prioridades se puede elegir entre un planificador expulsor o un planificador no expulsor. En un planificador no expulsor, cuando una tarea toma el control del procesador no detiene su ejecución hasta que ésta ha finalizado. En cambio, en un planificador expulsor una tarea cede el uso del procesador (es expulsada) si se activa una tarea de prioridad superior[2]. El planificador no expulsor es más sencillo de implementar, si bien puede producir un retraso significativo para las tareas de mayor prioridad, que es posible evitar con el planificador expulsor.

Aunque recientemente ha resurgido el interés de algunos autores [AUD96] [BAT97] [BAT98] por los planificadores no expulsores, la elección más natural es la de un planificador expulsor y así, el análisis desarrollado en esta tesis se refiere a sistemas con planificadores de este tipo, al menos para el procesador. En todo caso, los efectos de no expulsión son fácilmente modelables en el análisis de una tarea, mediante un término de bloqueo correspondiente al mayor tiempo de ejecución de peor caso de las tareas que tengan asignadas menor prioridad que la tarea bajo análisis (ver sección 2.2.2).

La asignación de prioridades a tareas de forma que se garantice la planificabilidad es todavía un tema abierto, en el que se han hecho muchas aportaciones significativas. La principal aportación vino dada por Liu y Layland [LIU73] y significó el principio de la teoría *RMA* (*Rate Monotonic Analysis*). En este artículo discuten ambos tipos de asignación: la asignación de prioridades estática en función de los periodos de las tareas (*Rate Monotonic*, *RM*, a menor periodo mayor prioridad) o la asignación dinámica en función del tiempo restante hasta los plazos de ejecución de las tareas (*Earliest Deadline First*, *EDF*, el plazo más cercano conlleva la mayor prioridad). Posteriormente el esquema de asignación estática fue mejorado por Leung y Whitehead [LEU82] para tareas con plazos menores al periodo. Esta asignación (*Deadline Monotonic*, *DM*) asigna mayor prioridad a la tarea que tiene menor plazo. Audsley desarrolló un algoritmo de asignación heurístico para el caso de plazos

[2] Realmente una tarea sólo puede ser expulsada por otras tareas de mayor prioridad. Sin embargo, a efectos de análisis de peor caso consideraremos que sí se puede dar este efecto de expulsión entre diferentes tareas con igual prioridad.

superiores al periodo [AUD91B] que obtiene una asignación de prioridades planificable, si acaso existe. Si bien estos esquemas son óptimos para sistemas con tareas periódicas en un sistema monoprocesador no lo son para sistemas con varios procesadores. Esto hace que se hayan desarrollado algunos algoritmos heurísticos para la asignación de prioridades de forma que ésta garantice los requerimientos temporales, como son el templado simulado [TIN92A] o el algoritmo *HOPA* [GUT95A], que obtiene generalmente mejores resultados, si la asignación de tareas a procesadores es conocida.

1.4.3 Planificación dinámica por prioridades dinámicas

Estas políticas de planificación, al igual que las anteriores, se refieren a la asignación de prioridades para la utilización del procesador. Sin embargo, al contrario que en la asignación estática, la prioridad de cada tarea no permanece fija una vez establecida, sino que puede variar en tiempo de ejecución, dependiendo del estado de ejecución de las tareas que requieren el uso del procesador.

El principal algoritmo de este tipo es el EDF (*Earliest Deadline First*) [LIU73], que asigna prioridades en función de los instantes en que se cumplen los plazos de las tareas. Se le asigna la mayor prioridad a la tarea que tiene el plazo más cercano al instante actual. Nótese que la asignación es en función del tiempo restante hasta que cumpla el plazo, que disminuye según el tiempo va transcurriendo. Otro algoritmo utilizado es el LLF (*Least Laxity First*) [AUD90], que asigna prioridades en función de las holguras, entendiendo como holgura la longitud del intervalo entre el instante actual hasta el instante en que se cumple el plazo, menos la cantidad de tiempo de cómputo pendiente de ejecución. Un algoritmo más eficiente es el de asignación al "Mejor Esfuerzo" (*Best-Effort*) [LOC85], que se basa en la asignación a priori de un valor fijo a cada tarea en función de su "utilidad". En tiempo de ejecución se asigna mayor prioridad, y por tanto de elige para ejecutar aquella tarea que tenga una relación menor valor_asignado/tiempo_ejecución_restante. Estos algoritmos son óptimos en sistemas monoprocesadores, pero dejan de serlo en multiprocesadores [AUD90] [RIP96].

Evidentemente, los planificadores por prioridades dinámicas son bastante más complicados de implementar y analizar que los basados en prioridades estáticas, aunque

consiguen a cambio utilizaciones más altas del procesador. Sin embargo, su utilización no se ha extendido lo suficiente, principalmente por las siguientes causas [RIP96]:

- No es estable ante sobrecargas. En caso de que alguna tarea consuma más tiempo del estimado, no hay forma de saber qué tarea perderá su plazo. Esto sucede con los algoritmos EDF y LLF, no así con el *Best-Effort*.

- La mayoría de los sistemas operativos actuales disponen de un planificador basado en prioridades, por lo que sería necesario construir un planificador a medida.

- No se dispone de unas bases teóricas suficientes.

1.5. Exclusión mutua e inversión de prioridad

En el apartado anterior hemos hablado de las políticas de planificación para el uso de los recursos, haciendo especial hincapié en el procesador. Desde luego, el procesador no tiene porqué ser el único recurso compartido en el sistema. Otros dispositivos también muy comunes, como pueden ser un disco duro o estructuras de datos en memoria, son de acceso mutuamente exclusivo y, aunque pueda elegirse una planificación basada en prioridades, es difícil que ese planificador sea expulsor en el uso de ese recurso.

La exclusión mutua puede llevar al efecto de inversión de prioridad en el procesador. Este efecto se produce cuando una tarea de prioridad alta tiene que esperar a que se libere un recurso que está siendo utilizado por otra tarea de prioridad menor. Nótese que este efecto se produce aunque la planificación del procesador sea expulsora, lo cual viola en cierta forma ese principio de expulsión y es una causa muy frecuente de incumplimiento de plazos. Hay muchas fuentes potenciales de inversión de prioridad, por ejemplo:

- Secciones de código no expulsable, de forma que durante su ejecución haya tareas de prioridad superior bloqueadas.

- Peticiones de uso de dispositivos que deban resolverse en orden de llegada (FIFO), de forma que una tarea deba esperar a que se resuelvan las peticiones de tareas de menor prioridad efectuadas con anterioridad.

- La sincronización por recursos compartidos de uso exclusivo, citada anteriormente, y en la que se puede producir un efecto de inversión de prioridad particularmente largo e incluso inversión de prioridad no acotada, que explicaremos a continuación.

La inversión de prioridad (y en especial la no acotada [SHA90A]) reduce notablemente la planificabilidad de los sistemas de tiempo real, por lo que es conveniente evitarla o, cuanto menos, reducirla. El principal esfuerzo en este sentido se ha hecho, dentro de la teoría *RMA*, en la sincronización para recursos compartidos de exclusión mutua.

La sección de código que ejecuta haciendo uso del recurso compartido se conoce como sección crítica. El mecanismo normalmente utilizado para garantizar el acceso mutuamente exclusivo es el de semáforos protegiendo las secciones críticas. Este mecanismo, sin embargo, puede ocasionar retrasos muy grandes cuando se aplica en sistemas de tiempo real. La inversión de prioridad no acotada se produce cuando una tarea de prioridad baja bloquea un semáforo que protege a un recurso compartido con otra tarea de prioridad alta y es expulsada por la ejecución de una tarea de prioridad intermedia. Esa expulsión provoca una inversión de prioridad de duración igual a la ejecución de todas las tareas de prioridad intermedia que puedan expulsar a la tarea de baja prioridad, lo cual puede ser un tiempo excesivamente largo.

El uso de semáforos presenta también otro problema, el bloqueo mutuo, en que dos tareas están interbloqueadas esperando ambas a que la otra libere un recurso que necesita para ejecutar. Las consecuencias de este efecto son catastróficas, puesto que ambas tareas se quedan indefinidamente esperando y pierden sus plazos con toda seguridad.

El objetivo de los protocolos de sincronización de tiempo real es precisamente evitar esas inversiones de prioridad no acotadas, minimizando la duración de las secciones críticas y evitar cualquier tipo de bloqueo destructivo. Los principales protocolos desarrollados para sistemas basados en prioridades fijas son [SHA90A] [RAJ89]:

- Protocolo de no expulsión: no se permite la expulsión durante la ejecución de la secciones críticas. Es equivalente a ejecutar las secciones críticas con prioridad estática igual a la prioridad máxima del sistema. Este protocolo minimiza la duración de las secciones críticas y evita los bloqueos mutuos. Presenta, sin embargo, el inconveniente de que interfiere en la ejecución de todas las tareas, aunque no hagan uso de recursos compartidos, por lo que crea bloqueos innecesarios.

- Protocolo de protección de prioridad o prioridad maximal: también es un protocolo basado en prioridades fijas. La sección crítica se ejecuta a la prioridad techo, que corresponde a la de la tarea de más alta prioridad que pueda hacer uso de ese recurso. Este protocolo es parecido al protocolo de no expulsión, con la diferencia de que aquí sí se permite la ejecución de tareas con prioridad mayor que la máxima del recurso. Con este protocolo pueden producirse secciones críticas con duración indefinida, pero sin que exista inversión de prioridad y además evita bloqueos mutuos, siempre que la tarea no se suspenda durante la sección crítica. También tiene la propiedad de que bloquea solamente una vez a tareas de mayor prioridad. Puede haber bloqueos indirectos sobre tareas de prioridad menor que el techo de prioridad y que no compartan ese recurso.

- Protocolo de herencia básica: Así como en los dos protocolos anteriores una sección crítica siempre ejecuta con una prioridad fija, en este protocolo una sección puede ejecutar a diferentes prioridades. La prioridad que se asigna a la sección crítica es la mayor de las prioridades de las tareas que están bloqueadas por ese recurso. Si durante la ejecución de la sección crítica una tarea de prioridad superior a la de la sección crítica intenta tomar el semáforo se verá bloqueada, pero la sección crítica aumenta su prioridad automáticamente hasta esa prioridad superior. Este protocolo elimina totalmente la inversión de prioridad, aunque produce tiempos de bloqueo de peor caso mayores que los de los protocolos de protección y techo de prioridad.

- Protocolo de techo de prioridad: Este protocolo se introdujo para evitar el problema de los bloqueos mutuos, así como los bloqueos encadenados, que presenta el protocolo de herencia básica, y producidos cuando existen tareas que comparten diferentes recursos simultáneamente, de forma que pueda haber diferentes secciones críticas

encadenadas en una misma tarea. Es el protocolo de herencia básica de prioridad más una regla acerca de la atención a la petición de bloqueo sobre un semáforo libre: una tarea no puede bloquear un semáforo libre a menos que su prioridad sea estrictamente mayor que la del techo del sistema (máximo techo de todos los semáforos actualmente bloqueados por otras tareas). Este protocolo previene el bloqueo mutuo siempre y cumple la propiedad de bloquear como máximo una vez a segmentos de código contiguo. La implementación de este protocolo da lugar a una pequeña sobrecarga debido a que hay que calcular siempre el techo de prioridad del sistema.

- Recientemente se ha desarrollado un protocolo sin bloqueo [CHB98] para la compartición asíncrona de datos en sistemas multiprocesadores. Este protocolo permite compartir datos entre un proceso escritor y múltiples procesos lectores a través de memoria compartida. Utilizando un doble *buffer* de memoria para cada proceso de lectura, el proceso de escritura conoce mediante consenso en qué *buffer* o zona de memoria va a leerse el dato compartido, de forma se elegirá para la escritura del nuevo dato el *buffer* que no se va a utilizar. De esta forma, se mantiene la coherencia de los datos y no se necesita bloquear el dato compartido, por lo que se evita el impacto negativo sobre tareas de menor prioridad.

Todos estos protocolos tienen un comportamiento predecible, en el sentido de que se puede calcular el impacto que tienen sobre el funcionamiento del sistema. Ese impacto se puede modelar como un término de bloqueo que afecta a tareas con prioridad comprendida dentro de un rango de prioridades determinado (en función del protocolo elegido) y que corresponde a la duración máxima de una o varias secciones críticas (dependiendo nuevamente del protocolo). El término de bloqueo se calcula fácilmente para los protocolos basados en prioridades fijas (protocolo de no expulsión y protocolo de protección de prioridad) y para el protocolo de techo de prioridad; el bloqueo sufrido por una tarea τ_i, identificado como B_i, es equivalente a la duración de la sección crítica más larga, con techo de prioridad mayor o igual que la asignada a τ_i, y ejecutada por tareas de prioridad menor que τ_i. El protocolo de herencia de prioridad requiere un cálculo más elaborado, tal como puede verse en [TOR92].

1.6. Planificación de eventos aperiódicos

Otro aspecto en que se ha interesado la teoría *RMA* ha sido en la planificación de tareas aperiódicas. Dentro de este tipo de tareas debemos distinguir entre tareas aperiódicas con o sin requerimientos temporales estrictos. Si el régimen de ocurrencia de eventos está limitado (bien por un tiempo mínimo entre llegadas o bien por una densidad máxima en un intervalo determinado), podemos garantizar un tiempo de respuesta máximo y, por tanto, el cumplimiento de requerimientos temporales estrictos. Otro caso es el de eventos aperiódicos con llegadas ilimitadas (donde no es posible garantizar un tiempo de respuesta máximo) o eventos aperiódicos sin requerimientos de tiempo real estricto. Aunque estas tareas no requieran el cumplimiento de ningún plazo de respuesta conviene que ésta sea lo más reducida posible. Una vez conseguido el cumplimiento de todos los plazos por tareas con requerimientos estrictos, la política de planificación debería favorecer la ejecución de este otro tipo de tareas, de forma que se consigan tiempos de respuesta promedios menores y, por tanto, mayor calidad de respuesta en el sistema.

Tal como comentamos en la sección 1.2.1, la sensorización de eventos (en particular considerando respuestas sin requerimientos temporales) se puede realizar a requerimiento del sensor o a requerimiento del procesador. En caso de que la sensorización se haga a requerimiento del sensor, debe tenerse en cuenta que las interrupciones *hardware* se atienden a prioridad muy alta. Una primera consideración importante es que en la mayoría de los casos, el manejador de interrupciones debería limitarse a "anotar" la ocurrencia del evento, ya que si utilizamos el mismo manejador para obtener la respuesta al evento no podríamos garantizar la planificabilidad del sistema, o la reduciríamos notablemente, si el ritmo de llegada de eventos aperiódicos es muy elevado. Políticas de planificación adecuadas para el tratamiento de estas tareas son la siguientes:

- Servidor de muestreo periódico. La ejecución se realiza mediante una tarea periódica de prioridad muy alta que chequea periódicamente la llegada de eventos y que permite una capacidad máxima de ejecución determinada. El periodo y la capacidad de ejecución se calculan a priori de forma que permita el cumplimiento de plazos a las tareas con requerimientos estrictos.

- Procesado directo en alta prioridad: la ejecución se realiza directamente, y a una prioridad muy alta. Esta política sólo es posible en eventos con régimen limitado de llegadas, puesto que, en caso contrario, se podrían incumplir plazos de tareas periódicas por estar ejecutando continuamente tareas aperiódicas.

- Procesado en baja prioridad: La planificación se basa también en prioridades fijas, pero asignando siempre una prioridad menor que la correspondiente a cualquier tarea con requerimientos temporales estrictos.

- Protocolos de reserva de ancho de banda. Se han propuesto varios planificadores basados en reservar cierta capacidad de ejecución para tareas aperiódicas, al estilo del servidor periódico visto en primer lugar. Uno de ellos es el Servidor Esporádico [SPR89A]. Este planificador reserva una cierta cantidad limitada de tiempo de ejecución para el procesado de eventos aperiódicos a un nivel de prioridad fijo previamente determinado y funciona según las siguientes reglas:

  - Cuando ocurre un evento aperiódico, si hay capacidad de ejecución disponible, la tarea aperiódica ejecuta a la prioridad asignada consumiendo la correspondiente capacidad de ejecución.

  - Cuando la capacidad de ejecución se agota, la tarea puede continuar ejecutando a un nivel de prioridad bajo (*background*).

  - Cada porción de capacidad de ejecución consumida se rellena, es decir, se añade a la capacidad de ejecución actual, después de que haya transcurrido un tiempo igual al período de relleno, contado a partir del instante en el que la porción de capacidad de ejecución consumida estaba lista para ejecutar.

  - Cuando se produce un relleno y el servidor esporádico estaba ejecutando a prioridad de background, su prioridad se eleva al nivel normal.

  Una propiedad muy interesante del Servidor Esporádico es que, desde el punto de vista del análisis de planificabilidad, se comporta como si fuera una tarea periódica

de prioridad igual a la prioridad asignada, periodo de activación igual al periodo de relleno y tiempo de ejecución de peor caso igual a la capacidad máxima de ejecución. Desgraciadamente, no hay muchas implementaciones comerciales de este planificador; aunque existen implementaciones de este servidor que permiten incorporarlo en el nivel de aplicación de una forma cómoda y eficiente [GON91B] [GON97].

- Servidores de holgura. Se basan en el concepto de intervalos de holgura, que corresponden a intervalos temporales en los cuales no hay ninguna tarea con requerimientos temporales estrictos que esté pendiente de ejecución. El algoritmo de extracción de holgura (*Slack Stealing*) [LEH92] se basa en la determinación *off-line* de los intervalos de holgura, de forma que se aprovechan para ejecutar tareas aperiódicas, si hay alguna pendiente. Una tarea de prioridad muy alta se encarga de planificar las tareas aperiódicas en los intervalos previamente determinados. Si una tarea aperiódica está ejecutando dentro de un intervalo de holgura, y no ha finalizado cuando termina dicho periodo, entonces se suspende su ejecución hasta el siguiente intervalo de holgura. Este algoritmo presenta el inconveniente de que debe generarse *off-line* una tabla con los intervalos de holgura, por lo que solamente es aplicable a sistemas en los cuales todas las tareas con requerimientos estrictos sean periódicas y además puede dar lugar a tablas excesivamente grandes, al igual que ocurre en el ejecutivo cíclico.

  Otros servidores retrasan al máximo la ejecución de las tareas periódicas, sin llegar a producir la pérdida de ningún plazo, de forma que el procesador disponga del mayor tiempo posible para atender a tareas aperiódicas. Dentro de este tipo de servidores es muy interesante la planificación por *Prioridades Duales* [DVI94A]. En esta planificación, todas las tareas con requerimientos temporales estrictos comienzan sus ejecución a un nivel de prioridad bajo, de forma que permiten la ejecución de tareas sin requerimientos temporales, a los que se asigna una prioridad intermedia. Cuando una tarea llega al límite de tiempo en que puede perder sus plazos, se le promociona (aumenta) automáticamente hasta su nivel normal de prioridad, de forma que no sobrepase el plazo establecido.Tal como se puede ver en [DVI94A], para realizar el análisis de planificabilidad, es suficiente con considerar el sistema sobrecargado de eventos aperiódicos, de forma que se puede considerar el sistema formado por las

tareas con requerimientos estrictos ejecutando a su nivel de prioridad normal. Esto hace que el análisis de sistemas con planificación por prioridades duales se pueda hacer exactamente igual que si estuviera formado sólo por las tareas con requerimientos temporales estrictos ejecutando siempre a su nivel normal de prioridad. En función de los resultados de ese análisis se asignan después los instantes de promoción de prioridad de las tareas.

1.7. Planificación en el modelo transaccional

Existen ciertas políticas que utilizan el modelo transaccional como modelo para planificar tareas en un sistema de tiempo real. Dentro de este contexto, una política de planificación interesante para reducir o eliminar los efectos de la activación retrasada (*release jitter*) es la basada en el algoritmo de Modificación de Fase [BET92]. En este método la activación de una tarea en respuesta a un evento no se produce de forma dinámica (mediante paso de mensajes), sino que se activa estáticamente un tiempo fijo después de la llegada del evento. Ese instante estático de activación se calcula teniendo en cuenta la respuesta de las tareas previas en la respuesta al evento, de forma que la activación de una tarea se produzca siempre después de que haya terminado la ejecución de las tareas previas.

Tindell [TIN93B] [TIN94B] propone directamente el modelo transaccional como modelo de planificación para tareas periódicas con dependencias temporales en sus patrones de activación. En este modelo, cada transacción se constituye con aquellas tareas que tengan el mismo periodo y restricciones de precedencia. Los *offsets* o fases de activación se conforman en función de los intervalos temporales relativos entre activaciones de las tareas.

La planificación de transacciones en sistemas multiprocesadores o distribuidos conlleva el problema de la sincronización del evento externo con cada fase de activación, ya que una tarea de la transacción puede ejecutar en un procesador diferente al que recibió el evento. Esto requeriría el uso de un reloj común o de relojes fuertemente sincronizados. Tindell propuso una solución alternativa, que consiste en el paso de mensajes indicando la ocurrencia del evento, considerando en cada tarea un posible retraso adicional en su activación igual al tiempo de transmisión de esos mensajes. Otro algoritmo de planificación de transacciones distribuidas que evita el uso de relojes sincronizados es el *Release Guard Protocol* [SUN96],

en el que cada tarea se activa por la llegada de un mensaje enviado por la tarea precedente. Cuando una tarea finaliza la ejecución de su código correspondiente se suspende hasta un instante fijo relativo a su instante de activación, momento en el cual envía el mensaje de activación a la tarea siguiente. Los mismos efectos se consiguen utilizando el servidor esporádico para las comunicaciones [GUT95A].

En los sistemas distribuidos gobernados por el paso de mensajes no es necesaria la sincronización entre diferentes procesadores, puesto que es la llegada de los mensajes lo que dispara la ejecución de las tareas. En esta tesis utilizaremos, a efectos analíticos, el modelo transaccional como aproximación al modelo gobernado por mensajes, incorporando los *offsets* (fases de activación) dinámicos, tal como veremos en el capítulo 4. Es por esta razón el que no tratemos el problema de la sincronización en el modelo transaccional, puesto que la planificación del sistema real se basa en el paso de mensajes, donde no existe tal problema.

Para el análisis, consideraremos cada cadena de respuesta a un evento externo como una transacción que agrupa las acciones desencadenadas por ese evento externo, tanto las tareas en los procesadores como los mensajes en las redes de comunicación. Como se verá en el capítulo 4 de esta tesis, el *offset* o fase de activación de cada acción variará dinámicamente en función del tiempo de respuesta de la acción previa en la cadena de respuesta original.

1.8. Problemática de los sistemas distribuidos

El análisis y planificación de sistemas de tiempo real estricto está bastante bien resuelto para sistemas monoprocesadores en los que el *software* se puede modelar mediante conjuntos de tareas periódicas o aperiódicas simples [KLE93]. Sin embargo, tanto el análisis como la planificación en sistemas multiprocesadores y distribuidos de tiempo real estricto son aún campos abiertos a la investigación en lo que se refiere a la búsqueda de mejores soluciones, y también de soluciones más generales, que se puedan aplicar a un número mayor de sistemas de este tipo.

Como hemos visto en apartados anteriores, no existen algoritmos óptimos de asignación de prioridades en sistemas distribuidos y para ello se han diseñado técnicas

heurísticas de planificación, tales como el templado simulado [TIN92A] o el algoritmo *HOPA* [GUT95A]. Estos algoritmos heurísticos se basan en cálculos sucesivos de tiempos de respuesta y reasignación de prioridades en función de los resultados obtenidos. De esta forma, el problema de la planificación de tareas se puede beneficiar notablemente si se utilizan mejores técnicas de análisis que las disponibles actualmente.

El problema que surge en los sistemas distribuidos, y que hace que no se disponga de una técnicas exacta de análisis, es el efecto de activación retrasada *(release jitter*) o desviación en la activación periódica de las tareas en sistemas en los que cada tarea se activa mediante un evento interno generado a la finalización de la tarea previa en la cadena de respuesta. En este caso, el instante en que se produce la activación de una tarea respecto de la llegada del evento externo puede retrasarse dependiendo de cuál sea el tiempo de respuesta de la tarea precedente. El cálculo del máximo retraso originado por ese motivo es, de hecho, lo que dificulta el análisis para sistemas distribuidos con relaciones de precedencia, puesto que calcularlo significa conocer a priori los tiempos de respuesta de las tarea precedentes en la cadena de respuesta. Esto origina una dependencia cíclica, en la que los tiempos de respuesta dependen de los retrasos máximos, y viceversa.

Tindell y Clark [TIN94F] propusieron un test de planificabilidad suficiente que proporciona cotas superiores de los tiempos de respuesta de las tareas en sistemas distribuidos con prioridades fijas. Para ello, solucionan el problema del *release jitter* mediante un algoritmo iterativo de análisis en el que suponen activaciones independientes de las tareas, considerando que cada una experimenta un retraso en su activación equivalente al tiempo de respuesta de peor caso estimado para la tarea precedente en la cadena de respuesta al evento externo. Esta técnica de análisis es pesimista, puesto que tal independencia de tareas no existe; la aproximación supone que una tarea puede ver interferida su ejecución simultáneamente por todas las tareas del sistema, incluidas las pertenecientes a su misma cadena de respuesta. Si bien es cierta la independencia entre activaciones de diferentes cadenas de respuesta, no lo es la activación de las distintas tareas de cada secuencia, precisamente por las relaciones de precedencia y, por tanto, puede que no interfieran todas las tareas de una misma cadena. Sin embargo, esta aproximación permite obtener cotas superiores de los tiempos de respuesta y permite aplicar un test de planificabilidad suficiente. Evidentemente, esta técnica de análisis es susceptible de mejora si conseguimos reducir el

pesimismo introducido por la aproximación de tareas independientes. Esa reducción es precisamente el objetivo principal de esta tesis.

1.9. Objetivos planteados en el trabajo

Tal como acabamos de señalar, el análisis de sistemas de tiempo real estricto está bastante bien resuelto en sistemas monoprocesadores modelados con tareas periódicas o aperiódicas simples (análisis que veremos en detalle en la sección 2.2). Este análisis, sin embargo, no es tan aceptable cuando se analizan sistemas monoprocesadores con un modelo no tan simple para las tareas, como es el caso de tareas que se suspenden, o cuando se analizan sistemas más complejos, como los sistemas multiprocesadores o distribuidos. En esta tesis trataremos de encontrar tests de planificabilidad aceptables, o que al menos mejoren los actuales, para sistemas de tiempo real con las siguientes características:

- Físicamente se dispone de un conjunto de procesadores de propósito general o especial, conectados entre sí mediante un sistema de comunicación de tiempo real con capacidad para la transmisión de mensajes o datos en tiempos acotados. Asimismo se dispone de un conjunto de dispositivos de entrada/salida conectados al sistema procesador, y de unidades de almacenamiento, tales como discos y dispositivos de memoria. Tales dispositivos deben ser capaces de responder en un tiempo máximo acotado, de forma que puedan utilizarse en aplicaciones con requerimientos temporales estrictos. Un conjunto de sensores y actuadores permiten la interacción desde y hacia el entorno físico bajo el que va a operar el sistema procesador.

- Desde el punto de vista *software*, el sistema consiste en un conjunto de tareas distribuidas por los distintos procesadores disponibles. Estas tareas se pueden sincronizar para utilizar recursos comunes e intercambian mensajes entre sí, ya sea a través de mecanismos locales si ambas están en el mismo procesador, o a través de algún canal físico del sistema de comunicación, si están ubicadas en diferentes procesadores.

- El disparo o activación de las diferentes tareas y mensajes viene determinado por la llegada de eventos externos ocurridos en el medio físico a controlar. El régimen de llegadas de eventos debe ser periódico, esporádico o limitado, si se quieren verificar requerimientos temporales estrictos. Cada evento producido en el entorno ocasiona la ejecución de una secuencia de respuesta, constituida por diferentes tareas que ejecutan secuencialmente en el mismo o distintos procesadores. La activación de la primera tarea en la secuencia se produce por la llegada del evento externo y las subsiguientes tareas se activan mediante mensajes. Estos mensajes a su vez se "activan" o son enviados por las tareas precedentes, en el instante de su finalización.

- La red de interconexión planifica los mensajes mediante prioridades asignadas a cada uno de los mensajes a transmitir. Los mensajes van partidos en paquetes, de forma que su comportamiento es muy parecido al de un planificador expulsor, con un pequeño término de bloqueo. Aunque éste no un objetivo en la tesis, toda la formulación que aparece en esta memoria se puede modificar para contemplar el análisis de otro tipos de planificación de mensajes.

Los requerimientos temporales impuestos al sistema se refieren a tiempos máximos de respuesta y pueden ser de los siguientes tipos:

- Plazo máximo de respuesta local. Tiempo máximo permitido para la ejecución, medido desde el instante en el cuál se produce la activación de la tarea. Relativo a los mensajes, este requerimiento se refiere al tiempo máximo de transmisión y recepción del mensaje.

- Plazo de respuesta global. Que indica el tiempo máximo permitido para la ejecución de una tarea, medido desde el instante en que ocurrió el evento externo a cuya secuencia de respuesta pertenece.

- Plazo de respuesta de principio-a-fin. Definido como el tiempo máximo en que se debe producir la respuesta completa a un evento. Es equivalente al tiempo de respuesta global de la última tarea ejecutada en la secuencia asociada al evento.

En cuanto a la planificación del sistema de tiempo real, el sistema que pretendemos estudiar se rige por la siguiente política:

- La asignación de procesadores o recursos a las tareas se realiza de forma estática, previamente al problema de verificación de los requerimientos temporales estrictos.

- La ejecución de tareas en cada procesador se planifica según el modelo de planificador dinámico expulsor basado en prioridades fijas. Para ello, se puede utilizar cualquier algoritmo de asignación de prioridades. Aunque no se trate directamente, también se pretende que el análisis sea fácilmente modificable para soportar el análisis de sistemas no expulsores.

Dado que nuestro objetivo último es mejorar las técnicas de análisis para ese tipo de sistemas, nos hemos planteado su consecución a partir de las siguientes etapas:

Optimización de la técnica existente. La técnica que obtiene mejores resultados para este tipo de sistemas es actualmente la desarrollada por Tindell y Clark [TIN94F], que se verá con detalle en la sección 2.3.4. Nuestra intención es profundizar en el desarrollo de esta técnica y formalizar e incidir en aquellos aspectos que nos parezcan de más interés para nuestros propósitos. En especial, es nuestra intención despejar todas las posibles dudas técnicas que se hayan podido plantear y que pongan en entredicho la validez de ese análisis. Posteriormente, intentaremos estudiar modificaciones sobre esa técnica que nos permitan mejorar los resultados obtenidos.

Desarrollo de nuevas técnicas de análisis. Como punto de continuación al anterior, pretendemos estudiar posibles aproximaciones que nos lleven a un análisis menos pesimista de los sistemas de tiempo real distribuidos. La técnica de Tindell y Clark está basada en la aproximación de tareas independientes. Nuestra intención es eliminar pesimismo mediante un análisis que tenga en cuenta de alguna forma la dependencia entre tareas pertenecientes a la misma secuencia de respuesta.

A continuación pasamos a desarrollar estos objetivos con mayor nivel de detalle.

1.9.1 Optimización de la técnica existente

Tal como hemos indicado, la técnica desarrollada por Tindell y Clark para sistemas distribuidos nos parece un punto de partida adecuado para nuevas técnicas de análisis. En un artículo [SUN96], Sun y Liu pusieron de manifiesto ciertos defectos en la demostración del método que les hacía pensar que el método desarrollado no obtiene resultados correctos, desde el punto de vista de que el método identificara como planificable un sistema que realmente no lo es. Si bien el análisis es efectivamente correcto, es fundamental en el desarrollo de nuestros objetivos que la validez del método este fuera de toda duda, y de ahí que pretendamos subsanar los posibles defectos en su formalización. Una vez conseguido esto, buscaremos nuevas vías de mejora, principalmente en dos direcciones:

- Obtención de mejores cotas para los tiempos de respuesta locales de peor caso. Pensamos que el análisis basado en tareas independientes puede suministrar mejores soluciones para los tiempos de respuesta locales, mediante pequeñas modificaciones en su formulación. Este calculo nos permitiría verificar mejor el cumplimiento de los requerimientos temporales locales.

- El análisis de Tindell y Clark consiste en suponer activaciones independientes para las tareas, considerando términos de retraso equivalentes a la máxima variación existente en la ejecución de las tareas precedentes en su misma secuencia de respuesta. Esto nos hace pensar en una posible manera de mejorar el análisis, que consistiría en la estimación de los tiempos de respuesta de mejor caso, entendidos como el mínimo tiempo que necesite una tarea para completar su ejecución. Esto redundaría directamente en la reducción de los términos de retraso y, consecuentemente, de los tiempos de respuesta obtenidos en el análisis. Para ello planteamos la búsqueda de una forma de calcular los tiempos de respuesta de mejor caso de las tareas, para luego evaluar las posibles mejoras introducidas por esta causa. Para poder aplicar esta nueva formulación será necesario medir u obtener tiempos de ejecución de mejor caso, pero pensamos que puede ser ventajoso, y en todo caso, supone un pequeño esfuerzo, equivalente a la obtención de los tiempos de ejecución de peor caso que actualmente se exige para el análisis.

1.9.2 Desarrollo de nuevas técnicas de análisis

Como segunda fase de la tesis doctoral pretendemos desarrollar técnicas de análisis basadas en aproximaciones más realistas que las planteadas en el método de Tindell para el comportamiento del sistema de tiempo real distribuido, que supone la independencia en la activación de las diferentes tareas. Un modelo analítico que nos parece muy adecuado es el modelo transaccional, definido en la sección 1.3.2. Este modelo, sin embargo, no es aplicable directamente para el análisis de sistemas distribuidos de tiempo real gobernados por eventos, ya que, tal como lo definió Tindell, los *offsets* de las tareas son fijos y menores que sus correspondientes periodos. Además, este modelo no tiene en cuenta los efectos de precedencia en la ejecución de tareas pertenecientes a una misma transacción. Por ello, pretendemos estudiar y ampliar ese modelo en varias direcciones:

- Extender el análisis de transacciones con *offsets* estáticos para permitir que los *offsets* puedan ser mayores que los periodos y, de esta forma, que se puedan utilizar en sistemas con plazos de ejecución mayores que esos mismos periodos. Parece lógico pensar que si una respuesta se distribuye en varios procesadores, se permita que la respuesta se dilate hasta ese mismo número de periodos.

- Permitir que los *offsets* varíen dinámicamente de una activación a otra. De esa forma, podremos asociar los *offsets* de cada tarea a la ejecución de las tareas previas y así obtener un análisis válido para tareas activadas en sistemas distribuidos con las características descritas al comienzo de esta sección.

- Tener en cuenta las relaciones de precedencia en la ejecución de las tareas pertenecientes a una misma transacción y describir con ello situaciones mas realistas y acordes con el funcionamiento del sistema distribuido.

- También nos parece conveniente considerar en el análisis el perfil de prioridades asignado a las diferentes tareas de cada secuencia. Este estudio permitiría describir de una manera aún más realista las posibles situaciones en las que se obtiene el máximo tiempo de respuesta de cada tarea. Hay cierto tipo de trabajos en este campo que nos hacen pensar que los resultados serían prometedores [GON91A] [SUN95].

Como resumen, el principal objetivo de esta tesis es mejorar el análisis desarrollado hasta ahora para sistemas distribuidos de tiempo real basado en prioridades fijas. Este objetivo se persigue principalmente en dos direcciones diferentes: por un lado, afianzar el análisis desarrollado por Tindell y Clark para sistemas distribuidos basado en la aproximación por tareas independientes e introducir pequeñas mejoras que hagan que el análisis reduzca el pesimismo y sea, por tanto, más efectivo. Por otro lado, desarrollar nuevas aproximaciones al análisis que permitan obtener tests de planificabilidad para sistemas distribuidos basados en modelos más adecuados a las relaciones de precedencia existente en la activación de las tareas dentro del sistema. Estas mejoras nos permitirían analizar sistemas distribuidos de tiempo real eliminando prácticamente todas las fuentes de pesimismo.

Persiguiendo esos objetivos, el trabajo desarrollado en esta tesis doctoral se ha planificado según las estructura descrita a continuación.

1.10. Organización de la memoria de tesis doctoral

Esta memoria está estructurada de la siguiente manera: en el capítulo 2 se hará un breve repaso de las técnicas de análisis existentes para sistemas monoprocesadores y distribuidos de tiempo real basados en prioridades fijas, que nos servirán como base para el análisis desarrollado en capítulos posteriores. En el capítulo 3 demostraremos la validez del método de análisis existente para sistemas distribuidos y describiremos algunas mejoras realizadas sobre ese análisis. El capítulo 4 describe el análisis de transacciones de tiempo real con *offsets* estáticos e introduce y analiza las transacciones con *offsets* dinámicos, como técnica alternativa útil para analizar sistemas distribuidos de tiempo real. En el capítulo 5 describimos las mejoras introducidas en el análisis desarrollado en el capítulo 4 y que nos permiten explotar de una forma más adecuada las relaciones de precedencia en el análisis de sistemas distribuidos. Por último, expondremos las conclusiones de esta tesis doctoral, así como posible trabajo futuro a desarrollar. En los siguientes párrafos describiremos algo más en detalle cada uno de estos puntos:

En el capítulo 2 introducimos las técnicas de análisis empleadas actualmente para sistemas de tiempo real y desarrolladas dentro de la teoría *RMA*. Dado que en esta tesis se desarrollan nuevas técnicas de análisis para sistemas planificados con prioridades fijas, nos

centraremos en las técnicas existentes para este tipo de planificación en sistemas modelados según lo visto en las secciones 1.2 y 1.3. En la sección 2.2 veremos las técnicas existentes para el caso de sistemas monoprocesadores, tanto las basadas en límites de utilización como las basadas en el cálculo de los tiempos de respuesta. Veremos el caso de tareas independientes, y posteriormente se contemplará el caso de tareas que compartan algún recurso. A continuación se mostrará el análisis existente para ese tipo de sistemas cuando los plazos de ejecución de las tareas sean mayores que sus periodos de activación. En la sección 2.3 se verán las técnicas de análisis para sistemas multiprocesadores y distribuidos, tratando en primer lugar el análisis de las redes de comunicación e introduciendo después la problemática de la activación retrasada (*release jitter*) en el análisis de tareas con relaciones de precedencia. Finalmente, describiremos el análisis desarrollado por Tindell y Clark para sistemas distribuidos basado en la aproximación por tareas independientes.

En el capítulo 3 se presentan algunas mejoras que hemos realizado sobre la base del análisis existente para sistemas distribuidos de tiempo real estricto. Comenzaremos demostrando, en la sección 3.2, la validez del análisis desarrollado por Tindell y Clark, sobre la que se han planteado recientemente algunas dudas. Estas demostraciones además sirven como base sobre la que se sustentan las demostraciones incluidas en posteriores capítulos. En la sección 3.4 extenderemos el análisis para efectuar el cálculo de los tiempos de respuesta locales de peor caso de las tareas de una manera más adecuada y que permite estimaciones menos pesimistas que las actuales. También consideraremos en este capítulo (sección 3.5) el cálculo de los tiempos de respuesta de mejor caso, como medio para reducir el retraso estimado en las activaciones de las tareas y reducir, por tanto, los tiempos de respuesta de peor caso obtenidos con las ecuaciones actuales. Todas estas mejoras sobre el cálculo podrán ser utilizadas dentro de las técnicas desarrolladas en capítulos posteriores como forma adicional de reducir el pesimismo.

El capítulo 4 describe el análisis de transacciones de tiempo real con *offsets*. En primer lugar, en la sección 4.2 mostramos y extendemos el análisis para tareas con *offsets* estáticos en varias direcciones: por un lado, eliminamos la restricción de que los *offsets* sean menores que los periodos de las transacciones y además formalizaremos la técnica mediante un conjunto completo de demostraciones. Este desarrollo introduce una nueva notación que nos será útil cuando extendamos nuestra técnica para tener en cuenta las relaciones de precedencia

entre las tareas de una transacción, y que desarrollaremos en el capítulo siguiente. Más importante es la extensión que realizamos en la sección 4.3 para cubrir el caso de tareas en las que los *offsets* puedan variar dinámicamente y que aplicaremos directamente, en la sección 4.4, al análisis de sistemas distribuidos y sistemas con tareas que se suspenden a sí mismas. En la última sección del capítulo hacemos un estudio comparativo que nos permite evaluar las ventajas del nuevo método. Como veremos, la aplicación de esta nueva técnica en sistemas distribuidos permite incrementar de una manera significativa las utilizaciones planificables de los procesadores, en comparación con los resultados del análisis aplicado actualmente a ese tipo de sistemas. Es de notar que estas mejoras no significan ningún cambio en la planificación de los sistemas, que puede ser realizada todavía con prioridades fijas.

En el capítulo 5 revisamos en profundidad el análisis de tareas con *offsets* dinámicos y refinamos su utilización en sistemas con relaciones de precedencia. Esto nos permite mejorar el cálculo de los tiempos de respuesta de peor caso eliminando pesimismo del análisis. En la sección 5.2 mejoraremos el algoritmo teniendo en cuenta que una tarea no puede ser expulsada por tareas a las que precede si estas últimas corresponden al mismo o a posteriores instancias del evento. Asimismo, en esa misma sección resumimos los resultados de simulaciones realizadas para mostrar las mejoras que reporta. En la sección 5.3 mejoramos sustancialmente el análisis explotando las relaciones de precedencia y considerando los diferentes niveles de prioridad asignados a tareas de una misma transacción. Este estudio nos permite eliminar situaciones incompatibles y ajustar así la máxima interferencia que puede sufrir una tarea y, por tanto, reducir la estimación de su tiempo de respuesta de peor caso. En la sección 5.3.1 veremos como afecta este cambio a la interferencia debida a tareas de transacciones diferentes a la de la tarea analizada y en la sección 5.3.2 como afecta a las tareas que tienen relaciones de precedencia con la tarea analizada. En la sección 5.3.3 evaluaremos las mejoras conseguidas, comparando los resultados obtenidos con el nuevo algoritmo diseñado, para diferentes conjuntos de sistemas a analizar. Finalmente, en la última sección del capítulo describimos cómo se puede aplicar el modelo transaccional al caso de sistemas formados por tareas con prioridades de ejecución variantes, siendo especialmente de interés el uso que se puede hacer de la técnica basada en *offsets* dinámicos para analizar este tipo de tareas en sistemas distribuidos.

El capítulo 6 está dedicado a resumir las conclusiones extraídas del presente trabajo, así como a referir las posibles líneas de investigación a seguir en un futuro. La parte final de esta memoria incluye las referencias bibliográficas citadas en los capítulos precedentes.

2. Técnicas existentes de análisis temporal.

2.1. Introducción

En este capítulo vamos a introducir las técnicas de análisis empleadas actualmente para sistemas de tiempo real y desarrolladas dentro de la teoría *RMA*. Dado que en esta tesis se desarrollarán nuevas técnicas de análisis para sistemas planificados con prioridades fijas, nos limitaremos a las técnicas existentes para este tipo de planificación en sistemas modelados según lo visto en las secciones 1.2 y 1.3. En [RIP96] se puede encontrar un resumen de las técnicas de análisis para planificaciones con prioridades dinámicas.

El objetivo de estas técnicas es verificar el cumplimiento de los requerimientos temporales estrictos, centrándose en el cumplimiento de los plazos máximos de respuesta. Esta verificación se lleva a cabo mediante un test de planificabilidad que determina a priori, en tiempo de compilación, si el sistema es planificable. Un test puede cumplir las siguientes condiciones:

- Suficiente: cuando el test dice que un sistema es planificable entonces efectivamente lo es siempre, en cualquier circunstancia y condición.

- Necesario: cuando el test dice que el sistema no es planificable entonces existe alguna condición bajo la cual el sistema no es planificable.

- Exacto: cuando verifica ambas condiciones de suficiencia y necesidad.

En sistemas de tiempo real estricto es imprescindible que nunca se pierdan los plazos de ejecución; esto quiere decir que un test válido para el análisis de planificabilidad debe ser siempre suficiente. Por otra parte, un test que no sea exacto será tanto más aceptable cuanto más se acerque a la condición de necesario.

En la sección 2.2 veremos las técnicas desarrolladas para el caso de sistemas monoprocesadores, tanto las técnicas basadas en límites de utilización como las basadas en el cálculo de los tiempos de respuesta. Analizaremos el caso de tareas independientes, que extenderemos para contemplar el caso de tareas que compartan algún recurso. Posteriormente se mostrará el análisis desarrollado para ese tipo de sistemas cuando los plazos de ejecución de las tareas sean mayores que sus periodos de activación. En la sección 2.3 se verán las técnicas de análisis para sistemas multiprocesadores y distribuidos, tratando en primer lugar el análisis de las redes de comunicación e introduciremos la problemática del *release jitter* en el análisis de tareas con relaciones de precedencia. Finalmente, describiremos el análisis desarrollado por Tindell y Clark para sistemas distribuidos, basado en la aproximación de tareas independientes [TIN94F].

2.2. Sistemas monoprocesadores

En esta sección veremos los principales tests de planificabilidad desarrollados para sistemas de tiempo real implementados en un sistema monoprocesador. Para la exposición de las diferentes técnicas supondremos un conjunto de N tareas periódicas ejecutándose en un único procesador. Cada tarea τ_i estará especificada mediante un tiempo de ejecución de peor caso C_i, un periodo de activación T_i y un plazo de ejecución D_i. Supondremos también las tareas ordenadas en orden creciente de periodos.

Principalmente hay dos familias de tests aplicados a tiempo real, basada cada una en un concepto diferente:

- Límite de utilización. Entendiendo por utilización el porcentaje de ocupación del procesador. Cada tarea periódica τ_i carga el procesador con una utilización dada por:

$$U_i = \frac{C_i}{T_i} \tag{1}$$

- Tiempo de respuesta. Se centran en el cálculo de los tiempos de respuesta de peor caso de todas las tareas del sistema para comparar posteriormente con sus plazos respectivos.

Ambas familias se basan en el concepto de instante crítico para una tarea τ_i, que corresponde al instante en que se produce la activación que tiene como tiempo de respuesta el de peor caso. El siguiente teorema establece ese instante [LIU73]:

Teorema 2-1 (Liu y Layland). El instante crítico para una tarea se produce cuando se activa simultáneamente a la activación de todas las tareas de prioridad superior.□

2.2.1. Tareas independientes

Liu y Layland dedujeron el primer test de planificabilidad dentro de la teoría *RMA* [LIU73]. Este test se aplica al caso de que las *n* tareas sean independientes en su ejecución, con plazos de finalización iguales a sus periodos. Los autores probaron que la asignación de prioridades óptima para este tipo de sistemas era la *rate monotonic* (*RMS*), que asigna mayor prioridad al que tiene menor periodo de activación. Está asignación es óptima en el sentido de que si un sistema con esta asignación no es planificable, entonces no hay ninguna otra asignación de prioridades que haga que el sistema lo sea. En estas condiciones, el sistema será planificable si cumple:

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq U(N) = N\,(2^{\frac{1}{N}} - 1) \qquad (2)$$

donde,

U Es la utilización total del procesador, e igual a la suma de las utilizaciones del conjunto de tareas en el sistema.

$U(N)$ Es el límite de utilización máximo del procesador para N tareas.

Este test condiciona la planificabilidad del sistema a que la utilización total del procesador no sobrepase un valor máximo establecido, dependiente del número total de tareas. Este límite de utilización varía (para $N>1$) entre el 83% y el 69%, lo cual apunta a sistemas planificables para utilizaciones relativamente bajas. Este test es suficiente pero no necesario,

de forma que puede haber conjuntos de tareas que sobrepasen el límite de utilización establecido por el test y que, sin embargo, sean planificables.

Lehoczky, Sha y Ding [LEH89] propusieron un test exacto para este mismo caso. Para ello, valiéndose de la definición de instante crítico dado por el teorema 2-1, definen una utilización de peor caso para una tarea τ_i, en un intervalo de anchura *t*. Esta utilización de peor caso se obtiene como la cantidad total de tiempo de ejecución requerida por tareas de prioridad mayor o igual que la de τ_i, dividido entre la anchura *t*, es decir:

$$U_i(t) = \frac{\sum_{j=1}^{i} \left(\left\lceil \frac{t}{T_j} \right\rceil C_j \right)}{t} \tag{3}$$

donde,

$U_i(t)$ es la utilización de peor caso en el intervalo $(0,t)$.

$\lceil x \rceil$ es la función techo, definida como el menor entero mayor o igual que *x*.

Entonces, el sistema será planificable si verifica:

$$\min_{0<t\le T_i} U_i(t) \le 1 \qquad \forall i = 1..N \tag{4}$$

Según esto, si para cada tarea τ_i existe un instante *t* comprendido entre 0 y el periodo T_i con utilización de peor caso menor o igual que 1, entonces el sistema es planificable. Para verificarlo es suficiente con chequear los instantes correspondientes a puntos de planificación, en los cuales se activa alguna tarea de mayor o igual prioridad que τ_i.

$$P_i = \left\{ k \cdot T_j \mid 1 \le j \le i \;,\; k = 1 \;..\; \left\lceil \frac{T_i}{T_j} \right\rceil \right\} \tag{5}$$

Esto se justifica por el hecho de que el término $\lceil t/T_j \rceil$ que aparece en la expresión (3) sólo puede cambiar de valor en instantes que sean múltiplos de T_j. El test queda en la forma:

$$\min_{t \in P_i} U_i(t) \le 1 \qquad \forall i = 1..N \tag{6}$$

Cuando los plazos de finalización son menores que los periodos, los tests anteriores ya no son válidos. Además, la asignación de prioridades en función de los periodos ya no es óptima: Leung y Whitehead probaron que, en este caso, la asignación óptima es la basada en

los plazos de ejecución (*deadline monotonic, DM*) [LEU82], de forma que le corresponda la mayor prioridad a la tarea que tenga menor plazo de ejecución.

Audsley [AUD91A] desarrolló un test de planificabilidad suficiente, basado también en utilizaciones, para un conjunto de tareas con asignación de prioridades según sus plazos de ejecución. La mayor interferencia en la ejecución de una tarea τ_i se produce en un instante crítico. Para una tarea que cumpla su plazo, esta interferencia está limitada por la expresión:

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{D_i}{T_j} \right\rceil C_j \tag{7}$$

donde,

hp(*i*) es el conjunto de tareas que pueden interferir la ejecución de τ_i, formado por las tareas con prioridad mayor o igual que la de τ_i, (exceptuando la propia τ_i).

El sistema será planificable si cada tarea experimenta una utilización máxima, definida dentro de su plazo, menor del 100 %. Esto es:

$$\frac{C_i}{D_i} + \frac{I_i}{D_i} \leq 1 \qquad \forall i = 1..N \tag{8}$$

Este test no es exacto, ya que supone que las tareas de mayor prioridad del conjunto *hp*(*i*) pueden interrumpir la ejecución de τ_i en cualquier instante dentro de su plazo, sin tener en cuenta que la ejecución de τ_i podría haber finalizado antes de que esa expulsión se hubiera producido.

El análisis exacto se obtiene por extensión del análisis de Lehoczky, Sha y Ding dado en la expresión (4), al caso de asignaciones *deadline monotonic*. Este test se basa igualmente, en una utilización de peor caso, definida ahora mediante:

$$U_i(t) = \frac{\left\lceil \frac{t}{T_i} \right\rceil C_i + \sum_{j \in hp(i)} \left(\left\lceil \frac{t}{T_j} \right\rceil C_j \right)}{t} \tag{9}$$

El sistema será planificable si verifica:

$$\min_{0<t\leq D_i} U_i(t) \leq 1 \qquad \forall i = 1..N \tag{10}$$

Al igual que con el test original, es suficiente verificar la desigualdad en los puntos de planificación producidos dentro del plazo de ejecución de τ_i,, del conjunto Q_i siguiente:

$$Q_i = \left\{ k \cdot T_j \mid j \in hp(i) \ , \ k = 1 \,..\, \left\lfloor \frac{D_i}{T_j} \right\rfloor \right\} \cup \{ D_i \} \tag{11}$$

De esta forma, el test se reduce a

$$\min_{t \in Q_i} U_i(t) \leq 1 \qquad \forall i = 1..N \tag{12}$$

Todos los test que hemos visto hasta ahora se basan en la utilización del procesador y nos permiten estudiar la planificabilidad de sistemas con asignación de prioridades *rate monotonic* o *deadline monotonic*. Un método más potente, en el sentido de que permite estudiar conjuntos de tareas con plazos menores o iguales que el periodo y cualquier asignación arbitraria de prioridades, es el desarrollado por Joseph y Pandya [JOS86]. Este método es equivalente al Algoritmo de Dilación Temporal (*Time Dilation Algorithm*) desarrollado por Harter [HAR84] y que utiliza lógica temporal para obtener tiempos de respuesta.

El método de Joseph y Pandya se basa en el cálculo de los tiempos de respuesta de peor caso de cada tarea. Si R_i es el tiempo de respuesta de peor caso de una tarea τ_i, la máxima interferencia producida a partir del instante crítico, debida a tareas de mayor prioridad viene dada por:

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{13}$$

El tiempo de respuesta de peor caso R_i vendrá dado por la suma de esta interferencia más el tiempo de ejecución de la propia tarea esto es:

$$R_i = C_i + I_i \tag{14}$$

De forma que la planificabilidad del sistema se condiciona a que se cumpla

$$R_i \leq D_i \qquad \forall i = 1..N \tag{15}$$

Desafortunadamente, el cálculo de la interferencia I_i para el cálculo del tiempo de respuesta R_i requiere el conocimiento previo de ese mismo tiempo de respuesta. Sin embargo, Joseph y Pandya demostraron que esa ecuación podía resolverse de forma iterativa, mediante la expresión siguiente, obtenida de sustituir (13) en (14):

$$R_i^{(n+1)} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{(n)}}{T_j} \right\rceil C_j \qquad (16)$$

El método consiste en estimar nuevos valores de R_i a partir de los valores obtenidos en la iteración anterior. La dependencia monótona de la ecuación respecto del término R_i garantiza la convergencia del algoritmo, siempre y cuando la utilización sea menor del 100% [JOS86]. La iteración comienza asignando un valor inicial al tiempo de respuesta $R_i^{(0)} = C_i$ y finaliza cuando se obtienen dos valores consecutivos iguales $R_i^{(n+1)} = R_i^{(n)}$.

2.2.2. Tareas con recursos compartidos

En esta sección contemplaremos el caso de tareas con recursos compartidos. Para ello supondremos que la compartición de recursos se lleva a cabo de acuerdo con alguno de los algoritmos de planificación de recursos discutidos en el capítulo 1 de esta memoria. Para los protocolos de no expulsión, protección de prioridad y techo de prioridad, cada tarea τ_i compartiendo recursos sufrirá un bloqueo en su ejecución, equivalente a la duración de la sección crítica más larga, con techo de prioridad[3] mayor o igual que la asignada a τ_i, y ejecutada por tareas de prioridad menor que τ_i. Identificaremos con B_i ese término de bloqueo.

$$B_i = \max_{\forall j \in lp(i)} s_j^i \qquad (17)$$

donde,

$lp(i)$ es el conjunto de tareas con menor prioridad que la asignada a τ_i y

s_j^i es la duración de la sección crítica con techo de prioridad mayor o igual que la asignada a la tarea τ_i, ejecutada por la tarea τ_j.

La inclusión del término de bloqueo en el análisis no sólo es importante desde el punto de vista de la compartición de recursos. Tal como dijimos en el capítulo anterior, el término de bloqueo nos permite modelar planificadores no expulsores o la planificación de mensajes en las redes de comunicación. En el caso de planificadores no expulsores, el término

[3] En el caso de protocolo de no expulsión, definimos el techo de prioridad como la prioridad más alta dentro del sistema.

de bloqueo que debemos considerar para el análisis de una tarea τ_i equivale al mayor de los tiempos de ejecución de peor caso de tareas de menor prioridad. Esto es:

$$B_i = \max_{\forall j \in lp(i)} C_j \tag{18}$$

Considerando el término de bloqueo, se puede generalizar el test basado en el límite de utilización de Liu y Layland. Un sistema compuesto por un conjunto de *N* tareas será planificable, utilizando asignaciones de prioridad *rate monotonic*, si para cada tarea τ_i se verifica [SHA90A]:

$$U_i = \sum_{j=1}^{i} \frac{C_j}{T_j} + \frac{B_i}{T_i} \leq U(i) = i\,(2^{\frac{1}{i}} - 1) \qquad \forall i = 1..N \tag{19}$$

Es decir, una tarea será planificable si la utilización de tareas con mayor o igual prioridad es menor o igual que el límite de utilización dado por la expresión $U(i)$, que es función del número de tareas con prioridad mayor o igual que τ_i. Esta relación debe verificarse para cada tarea del sistema, por lo que en realidad se trata de un conjunto de desigualdades.

Lehoczky [LEH91] y Klein et al. [KLE93] extendieron este análisis al caso de que las tareas tuvieran plazos iguales al periodo pero que pudiera hacerse cualquier asignación de prioridades fijas, sin tener que limitarse a *RMS*. Este planteamiento es útil, por ejemplo, cuando el sistema trata tareas aperiódicas mediante un servidor periódico, de forma que la prioridad de ese servidor sea alta, aunque el periodo de servicio sea mayor que el de otras tareas periódicas. El test establece el límite de utilización planificable en la forma:

$$\sum_{j \in H(i)} \frac{C_j}{T_j} + \frac{C_i}{T_i} + \frac{B_i}{T_i} + \sum_{j \in S(i)} \frac{C_j}{T_i} \leq n_i\,(2^{\frac{1}{n_i}} - 1) \qquad \forall i = 1..N \tag{20}$$

donde,

$H(i)$ es el conjunto de tareas con mayor prioridad y periodo menor que τ_i,

$S(i)$ es el conjunto de tareas con mayor prioridad que τ_i , pero con periodo mayor, y

n_i es el número de tareas en el conjunto $H(i) + 1$

El método de Sprunt [SPR89A], considerando tareas aperiódicas y asignación de prioridades *rate monotonic*, generaliza el test de Lehoczky, dado en (4). El nuevo test incluye el término de bloqueo, quedando:

$$\min_{t \in P_i} \left(U_i(t) + \frac{B_i}{t} \right) \leq 1 \qquad \forall i = 1..N \tag{21}$$

donde $U_i(t)$ es la utilización de peor caso dada por la ecuación (3) y P_i es el conjunto de puntos de planificación definido en (5).

Por último, Audsley [AUD93] generalizó el método de Joseph y Pandya basado en el cálculo de los tiempos de respuesta de peor caso. Este método es válido para conjuntos de tareas con cualquier asignación fija de tareas y con plazos de finalización menores o iguales que los respectivos periodos. Además contempla el efecto de activación retrasada en tareas periódicas (*release jitter*). La técnica de análisis para una tarea τ_i se basa también en la creación de un instante crítico. Sin embargo, cuando hay activaciones retrasadas, la definición de instante crítico varía ligeramente de la dada anteriormente, y se construye según la forma indicada en el siguiente teorema, suponiendo el protocolo de no expulsión, protección de prioridad o techo de prioridad para el uso de recursos compartidos:

Teorema 2-2 [AUD93]. El instante crítico para una tarea se produce cuando se activa sufriendo el máximo retraso posible, recién bloqueado el recurso compartido que tenga mayor sección crítica, y coincidiendo con la activación, también retrasada, de todas las tareas de mayor o igual prioridad. El resto de activaciones posteriores al instante crítico se producirán sin retraso.□

Considerando la creación de ese instante crítico, el tiempo de respuesta de peor caso de una tarea τ_i será igual a:

$$R_i = B_i + C_i + I_i + J_i \tag{22}$$

donde,

I_i es la máxima interferencia debida a tareas de mayor o igual prioridad que τ_i

J_i es el máximo retraso en la activación de la tarea τ_i

Ahora bien, la interferencia debida a tareas de mayor o igual prioridad sólo puede producirse cuando la tarea τ_i ya se haya activado. Esto quiere decir que la interferencia máxima no es la producida en un intervalo de duración R_i, sino que debemos considerar un intervalo w_i (también llamado tiempo de finalización) que representa el intervalo comprendido entre el instante en que se activa la tarea τ_i y el instante en que finaliza su ejecución. En estas condiciones, la interferencia máxima viene dada por la expresión:

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{w_i + J_i}{T_j} \right\rceil C_j \tag{23}$$

y el instante de finalización mediante

$$w_i = B_i + C_i + I_i \tag{24}$$

A partir de ese tiempo de finalización, el tiempo de respuesta de peor caso, considerado desde el instante en que debería haberse activado la tarea, corresponde a:

$$R_i = w_i + J_i \tag{25}$$

Con estas consideraciones, el conjunto de tareas será planificable si se verifica:

$$R_i \leq D_i \qquad \forall i = 1..N \tag{26}$$

Afortunadamente, también se puede aplicar un método iterativo, equivalente al utilizado por Joseph y Pandya en (16), para la obtención del tiempo de finalización. Sustituyendo la ecuación (23) en la (24) llegamos a la expresión :

$$w_i^{(n+1)} = B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^{(n)} + J_i}{T_j} \right\rceil C_j \tag{27}$$

La iteración comienza asignando al tiempo de finalización un valor inicial $w_i^{(0)} = C_i$ y finaliza cuando se obtienen dos valores consecutivos iguales $w_i^{(n+1)} = w_i^{(n)}$. El tiempo de respuesta de peor caso se obtendría sustituyendo ese valor en la ecuación (25).

2.2.3. Plazos superiores a los periodos

Tratamiento aparte merece el caso de plazos superiores al periodo, especialmente importante en sistemas distribuidos. En estos sistemas es usual que la respuesta a un evento involucre la ejecución sucesiva en varios procesadores, de forma que el primer procesador

pueda tratar un nuevo evento cuando los subsiguientes procesadores siguen respondiendo a eventos anteriores.

El hecho de que en un mismo instante pueda haber varios eventos pendientes complica el análisis, ya que deben tenerse también en cuenta las posibles interferencias en la respuesta a un evento debido a activaciones anteriores del mismo evento. Además, ninguna de las dos asignaciones de prioridades citadas anteriormente, en función de sus periodos o de sus plazos, es óptima. Esta asignación debe ser realizada mediante algún algoritmo heurístico [AUD91B].

Lehoczky [LEH90] demostró que la creación de un instante crítico sigue siendo válido, pero debe extenderse el análisis sobre todas las activaciones que ocurran dentro de su periodo de ocupación (*busy period*) de peor caso. Un periodo de ocupación para una tarea τ_i es un intervalo de tiempo durante el cual el procesador está ocupado ejecutando tareas de mayor o igual prioridad que la de τ_i.

El test propuesto por Lehoczky es una generalización del método utilizado para plazos menores o iguales que el periodo, basado en el límite de utilización. Para ello define una función $W_i(x,t)$, que calcula el máximo tiempo de ejecución requerido por tareas de mayor o igual prioridad que la tarea τ_i junto con x activaciones de τ_i, en un intervalo de duración t, contado desde el instante crítico creado para la ejecución de la tarea τ_i. Es decir:

$$W_i(x,t) = x\ C_i + \sum_{j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j \tag{28}$$

A partir de esa función se obtiene la utilización máxima en ese intervalo como:

$$U_i(x,t) = \frac{W_i(x,t)}{t} \tag{29}$$

El conjunto de tareas será planificable si, para cada activación dentro del periodo de ocupación correspondiente, existe un instante anterior a su plazo de finalización para el que la utilización máxima es menor que 1. Dado que la activación x-ésima de la tarea τ_i se produce en el instante $(x\text{-}1)T_i$, el plazo de finalización será el instante $(x\text{-}1)T_i+D_i$ y, por tanto

$$\min_{0<t\le(x-1)T_i+D_i} U_i(x,t) \le 1 \quad \forall x = 1..N_i \quad , \quad \forall i = 1..N \tag{30}$$

Siendo N_i el número de activaciones de cada tarea τ_i dentro de su periodo de ocupación. Este número se puede calcular teniendo en cuenta que corresponde a la primera activación cuya ejecución finaliza antes de que se produzca la siguiente. La siguiente activación a la *x*-ésima se produce en el instante xT_i, con lo que tenemos:

$$N_i = \min \left(x \mid \min_{0<t\le xT_i} U_i(x,t) \le 1 \right) \tag{31}$$

Por último, Tindell y otros extendieron estos resultados para desarrollar una técnica exacta basada en la obtención de los tiempos de respuesta, que incorpora además el efecto de la ejecución retrasada o la compartición de recursos [TIN92B] [TIN94F].

La técnica se basa en el cálculo de los tiempos de respuesta de peor caso de las activaciones ocurridas dentro del periodo de ocupación. Para ello, calcula el tiempo de finalización de cada activación *q*-ésima (Tindell numera la primera activación con el valor q=0 al contrario de Lehoczky, que lo hace con x=1), según la expresión:

$$w_i(q) = (q+1)\, C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i(q) + J_i}{T_j} \right\rceil C_j \tag{32}$$

Esa ecuación se resuelve con el método iterativo ya descrito anteriormente, con el valor inicial $w_i^{(0)}(q) = (q+1)\, C_i$. A partir del tiempo de finalización $w_i(q)$ se obtiene el tiempo de respuesta, considerando el instante $-J_i + qT_i$ en que se produjo esa activación. El tiempo de respuesta de peor caso de la tarea τ_i corresponderá al mayor de todos los tiempos de respuesta obtenidos:

$$R_i = \max_{q=0,1,2,\ldots} \left(w_i(q) + J_i - q\, T_i \right) \tag{33}$$

Este método, sin embargo, no chequea todas las activaciones producidas dentro del periodo de ocupación, sino que se detiene cuando se verifica la condición

$$w_i(q) \le (q+1)\, T_i \tag{34}$$

En el capítulo 3 incidiremos en este punto y demostraremos su validez.

El conjunto será planificable si todas las tareas tienen un tiempo de respuesta de peor caso menor que su plazo de ejecución:

$$R_i \le D_i \tag{35}$$

Este último test compendia las técnicas desarrolladas para sistemas monoprocesadores con planificación expulsora por prioridades fijas. El análisis es independiente de la política de asignación de prioridades elegida y posibilita además la inclusión de otros aspectos no considerados aquí, como pueden ser interferencias del planificador.

2.3. Sistemas multiprocesadores y distribuidos

La planificación de sistemas multiprocesadores o distribuidos se complica bastante respecto a la planificación de los sistemas monoprocesadores. En primer lugar, la asignación de prioridades en función de los periodos o los plazos de ejecución no es óptima, por lo que se debe realizar de forma heurística [TIN92A] [GUT95A]. Por otra parte, aparecen nuevas fuentes de inversión de prioridad, como la debida a la sincronización por el uso de la memoria compartida, produciendo bloqueos remotos que obligan a mejorar los algoritmos de planificación de recursos compartidos.

2.3.1. Análisis de las redes de comunicación

El análisis en sistemas multiprocesadores o distribuidos también se complica bastante respecto al caso monoprocesador. En primer lugar, es necesario considerar el análisis del sistema de comunicación entre los diferentes procesadores: supuesto que queremos enviar un mensaje de un determinado tamaño a través de una red, debemos conocer el tiempo total que empleará el mensaje en llegar a su destino.

Como ya comentamos en el capítulo 1, supondremos que los mensajes se envían partidos en paquetes de tamaño fijo y que, en la implementación del sistema de tiempo real, se emplean redes de comunicación que planifican la transmisión de los paquetes mediante prioridades asignadas a los mismos. El tiempo empleado en el proceso de generación y partición de los mensajes se puede asignar bien a la tarea que quiere enviar el mensaje o bien a otra tarea independiente, si se encarga de ello. De igual forma, el tiempo del proceso de reconstrucción y consumición del mensaje se puede añadir al tiempo de ejecución de peor caso de la tarea destino del mensaje, o a la tarea encargada de la recepción de mensajes. Sin embargo, el tiempo que el sistema de comunicación emplea en enviar el mensaje no depende

exclusivamente del mensaje a transmitir, sino que debe tener en cuenta la contención debida a otros mensajes que estén utilizando la misma red de comunicación.

Dado que la red no es expulsable, debemos considerar el bloqueo debido a la transmisión de mensajes de menor prioridad. La duración de este bloqueo será igual al tiempo de transmisión de un paquete, y lo llamaremos B_m.

Para calcular el tiempo total que tarda el mensaje en llegar completamente a su destino debemos considerar además la interferencia de mensajes de mayor prioridad que circulen por la misma red. Si los plazos de ejecución son menores o iguales que los correspondientes periodos, el tiempo de respuesta se obtiene de la expresión:

$$R_m = C_m + B_m + \sum_{j \in hp(m)} \left\lceil \frac{R_m}{T_j} \right\rceil C_j \tag{36}$$

donde,

C_m es el tiempo de transmisión de peor caso del mensaje, suponiendo que fuera el único a transmitir por ese canal de comunicación.

Estrictamente hablando, y debido a que no hay expulsión, el último paquete se transmite sin sufrir ninguna interferencia, por lo que se debería considerar en esta última ecuación un tiempo de transmisión de peor caso C_m con un paquete menos y, posteriormente, sumarle el tiempo de transmisión de ese último paquete al valor de R_m obtenido. Hemos preferido no hacerlo así para poder uniformizar el modelo y utilizar las mismas expresiones para la ejecución de tareas o el envío de mensajes, aunque cualquiera de las ecuaciones derivadas en esta tesis se puede adaptar para considerar ese hecho, sin más que tratar ese último paquete en la forma comentada.

Este análisis se puede ampliar para considerar también plazos de respuesta mayores que los periodos y retrasos en la generación del mensaje. Al igual que con las tareas, se obtiene primero el tiempo de finalización de la transmisión:

$$w_m(q) = (q+1)\ C_m + B_m + \sum_{j \in hp(m)} \left\lceil \frac{w_m(q) + J_m}{T_j} \right\rceil C_j \tag{37}$$

Y a partir de él, el tiempo total de respuesta para la transmisión del mensaje:

$$R_m = \max_{q=0,1,2,\ldots} \left(w_m(q) + J_m - q\ T_m \right) \qquad (38)$$

2.3.2. Modelo general para el análisis

Debido a esta similitud entre él cálculo de los tiempos de respuesta en la ejecución de las tareas y en la trasmisión de los mensajes, a partir de ahora nos referiremos indistintamente a ambos como acciones.

El modelo general que manejaremos en esta tesis se muestra en la Figura 2-1. Cada evento e_i generado periódicamente en el entorno desencadenará una serie de acciones a_j, tanto en procesadores como en redes de comunicación, activadas entre sí mediante eventos internos e identificadas según el orden en que deban ejecutar en la respuesta al evento externo, de

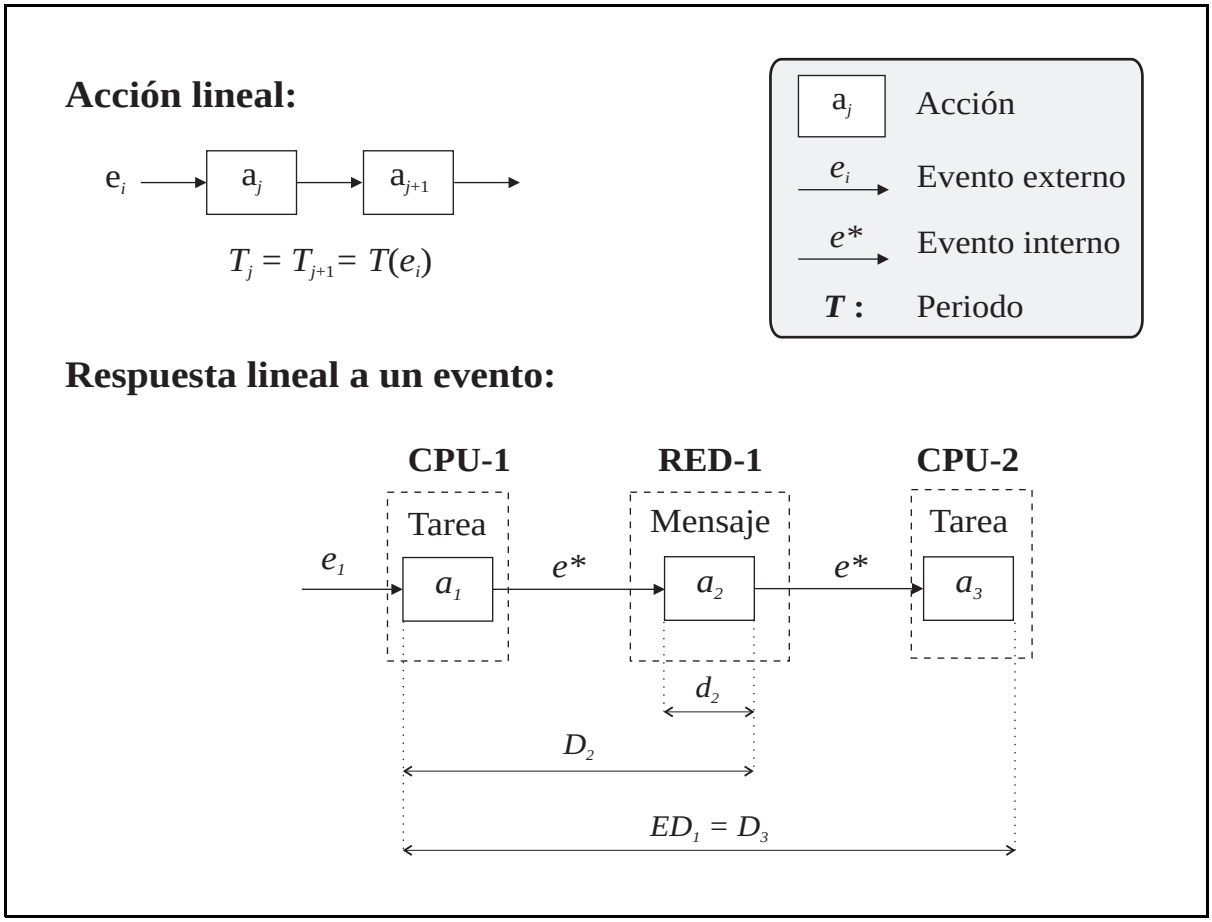


Figura 2-1. Modelo lineal de sistema distribuido gobernado por eventos.

forma que la siguiente acción a una acción a_j corresponde a la a_{j+1}. Cada acción a_j se considera con un tiempo de ejecución o transmisión de peor caso C_j y hereda el periodo T_j del evento externo e_i a cuya secuencia de respuesta pertenece. Asimismo, consideraremos los siguientes requisitos temporales impuestos a las acciones:

Plazos globales: plazos de respuesta relativos a la llegada del evento externo.

Plazos locales: plazos considerados desde el instante real en que se activó la acción en la respuesta al evento.

Plazos de principio-a-fin: plazos asignados a las secuencias de respuesta completas. Coinciden con los plazos globales correspondientes a las últimas acciones en la secuencias de respuesta a eventos externos.

Llamaremos D_j al plazo global de la acción a_j relativo a la llegada del evento e_i, d_j al plazo local de la acción a_j y ED_i al plazo de principio-a-fin para la respuesta al evento e_i, que coincide con el plazo global de la última acción de la cadena de respuesta. En la Figura 2-1 se pueden ver ejemplos de estos plazos.

En capítulos posteriores de esta tesis (capítulos 4 y 5) utilizaremos también el modelo transaccional definido en la sección 1.3.2, en el que modelamos la cadena de respuesta a un evento periódico mediante una transacción periódica formada por las acciones ejecutadas en la cadena, cada una de ellas con su correspondiente fase de activación, obtenida a partir de los tiempos de respuesta de las tareas previas en la cadena (ver capítulo 4). Ambos modelos no son equivalentes, puesto que en una secuencia de respuesta, cada acción no se activa hasta que no haya finalizado la ejecución de la acción previa y, en el modelo transaccional, cada tarea se activa a partir del instante definido por su *offset*, sin precedencia explícita. Si bien este modelo transaccional corresponde a un planificación diferente a la del sistema real, sólo se utiliza a efectos analíticos, como modelo con el que se aproxima iterativamente la planificación por paso de mensajes del sistema real.

2.3.3. La problemática del *jitter*

Además del análisis de las redes de comunicación, hay otro aspecto fundamental que complica el análisis del sistema distribuido. Este aspecto es la precedencia en la activación de las tareas y los mensajes pertenecientes a una misma secuencia de respuesta. Tal como indican Tindell y Clark [TIN94F], esta precedencia significa que una acción a_j se activa en el instante en que finaliza la ejecución de la acción previa en la secuencia de respuesta, a_{j-1}, excepto la primera de la secuencia, que se activa con la llegada del evento externo. Si suponemos que la notificación del evento externo se produce de forma perfectamente periódica con retraso nulo, la primera acción de la cadena de respuesta se activará también de forma perfectamente periódica con retraso J_j=0. Sin embargo, las subsiguientes acciones pueden sufrir un retraso máximo en su activación, equivalente a la máxima variación en el tiempo de respuesta de la acción precedente. Esto es:

$$J_j = R_{j-1} - R_{j-1}^b \tag{39}$$

donde,

R_{j-1} es el tiempo de respuesta global de peor caso correspondiente a la acción previa a a_j en la secuencia de respuesta al evento e_i. Si a_j es la primera tarea de la secuencia de respuesta, entonces se considera R_{j-1}= 0, y

R_{j-1}^b es el tiempo de respuesta global de mejor caso de la acción precedente a a_j en la secuencia de respuesta al evento externo e_i. Tradicionalmente, este tiempo de respuesta de mejor caso se ha considerado arbitrariamente pequeño, de forma que puede ser estimado inferiormente con el valor R_{j-1}^b =0.

Con esa estimación de los términos correspondientes a los retrasos máximos en la activación de cada acción, Tindell y Clark [TIN94F] aplicaron la formulación desarrollada por Tindell para sistemas monoprocesadores [TIN92B] [TIN94D] (ver sección 2.2.3), como si se tratara de tareas independientes. No hay más que modificar la definición del conjunto de acciones que pueden interferir en la ejecución de una acción a_j, considerando sólo las que se encuentren en el mismo recurso, ya sea procesador o red de comunicación.

2.3.4. Análisis bajo la aproximación de tareas independientes

El instante crítico se crea de la misma forma que si las tareas fueran independientes, lo cual no hace sino empeorar la ejecución de la tarea analizada, con lo que se obtendrá una estimación pesimista de los tiempos de respuesta de peor caso. El tiempo de finalización de la activación p-ésima[4] de un acción a_j, después del instante crítico creado, se obtiene a partir de la ecuación siguiente, (resoluble mediante el método iterativo habitual):

$$w_i(p) = p\ C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i(p) + J_j}{T_j} \right\rceil C_j \qquad (40)$$

donde,

$hp(i)$ es el conjunto de acciones de mayor o igual prioridad que la asignada a la acción a_i, y que se encuentren alojadas en el mismo recurso, exceptuando la propia a_i. Realmente, las tareas con igual prioridad que la tarea i no pueden expulsarla una vez haya comenzado a ejecutar; sin embargo, a efectos de análisis de peor caso consideraremos que sí se puede dar este efecto.

B_i es el término de bloqueo correspondiente a la acción a_i debido a la sincronización por el uso de recursos compartidos.

El tiempo de respuesta global se obtiene considerando el instante en que se produjo el evento externo. Teniendo en cuenta la construcción del instante crítico, y que el máximo retraso en la activación de una tarea corresponde al valor dado por R_{i-1}, calculamos el tiempo de respuesta de peor caso mediante la expresión:

$$R_i = \max_{p=1,2,\ldots} \left(w_i(q) + R_{i-1} - (p-1)\ T_i \right) \qquad (41)$$

La iteración sobre p finaliza cuando se verifica:

$$w_i(p) \le p\ T_i \qquad (42)$$

[4] Por conveniencia para las demostraciones que efectuaremos en el capítulo 3 de esta tesis hemos variado el esquema de numeración de los eventos, comenzando por p=1 en lugar del esquema original, que comienza con q=0.

El sistema distribuido de tiempo real será planificable si todas las acciones tienen un tiempo de respuesta de peor caso menor que sus plazos de ejecución:

$$R_i \leq D_i \tag{43}$$

Esta metodología de análisis tiene el problema añadido de que el cálculo del tiempo de respuesta global de una tarea depende de los tiempos de respuesta de las tareas previas en la secuencia de respuesta al evento externo. Los tiempos de respuesta se calculan mediante la ecuación (41), utilizando los tiempos de finalización obtenidos con la ecuación (40); esta ecuación requiere el conocimiento previo de los términos de retraso, calculados en (39), que a su vez, requiere conocimiento previo de los tiempos de respuesta obtenidos con (41).

Tindell y Clark diseñaron un método iterativo que permite obtener los tiempos de respuesta de peor caso de las tareas. Este método, mostrado en la Figura 2-2, consiste en estimar valores para los términos de retraso y en función de ellos obtener, mediante las ecuaciones (40) y (41), valores de los tiempos de respuesta que se puedan usar para estimar nuevos términos de retraso, mediante la ecuación (39). Este proceso estimativo continúa hasta que se obtengan los mismos tiempos de respuesta en dos iteraciones consecutivas. El comportamiento monótono, tanto en el cálculo de los tiempos de respuesta como de los términos de retraso garantiza la convergencia del método.

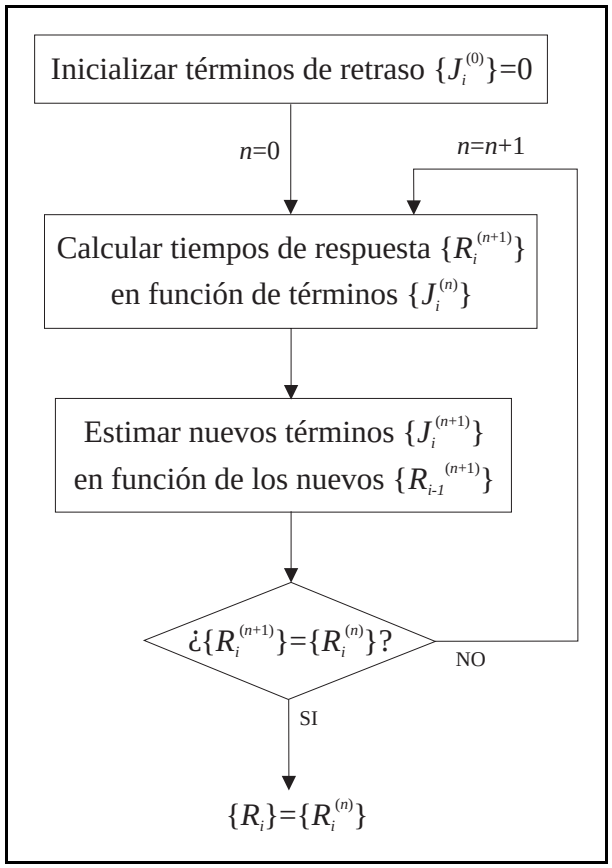


Figura 2-2. Algoritmo recursivo

3. Mejoras sobre el análisis en sistemas distribuidos.

3.1. Introducción

En este capítulo presentaremos algunas mejoras que hemos realizado sobre la base del análisis existente para sistemas distribuidos de tiempo real estricto. Comenzaremos demostrando la validez del análisis desarrollado por Tindell y Clark [TIN94F] y sobre el que se han planteado recientemente algunas dudas. Extenderemos también el análisis para efectuar el cálculo de los tiempos de respuesta locales de peor caso de las tareas de una manera más adecuada y que permita estimaciones menos pesimistas que las actuales. También consideraremos en este capítulo el cálculo de los tiempos de respuesta de mejor caso, como medio para reducir el retraso estimado en la activación de las tareas y reducir, por tanto, los tiempos de respuesta de peor caso obtenidos con las ecuaciones actuales. Como se podrá observar en la sección 3.5.3., la reducción obtenida en los términos de retraso, gracias al cálculo de los tiempos de respuesta de mejor caso, consigue que el análisis modificado por ello obtenga soluciones mejoradas hasta un 5 % respecto al análisis original. La ventaja adicional que presenta este nuevo cálculo de los tiempos de respuesta es que su inclusión no significa una modificación del sistema de tiempo real original, por lo que se obtiene esa ventaja de hasta de un 5% sin coste alguno en la implementación del sistema. El único coste adicional se centra en la medida de los tiempos de ejecución de mejor caso.

3.2 Validación de la técnica existente

Aunque la técnica de análisis desarrollada por Tindell y Clark [TIN94F] para la obtención de los tiempos de respuesta de tareas y mensajes en sistemas distribuidos ha sido ampliamente aceptada y usada, Sun y Liu han mostrado en un artículo reciente [SUN96] que

su demostración no está completa, porque no chequea los tiempos de respuesta de todos los eventos que pueden llegar al sistema durante el intervalo de tiempo que debe ser considerado en el análisis de peor caso, esto es, durante el periodo de ocupación. Sun y Liu concluyen que el análisis de Tindell y Clark es incorrecto y proponen una técnica de análisis que es igual que la técnica original (aunque con diferente terminología), pero que chequea todos los eventos que pueden llegar al sistema durante el periodo de ocupación completo. En este apartado vamos a demostrar que la técnica de análisis original es válida, ya que la respuesta de peor caso puede ocurrir sólo durante el intervalo de tiempo chequeado en las ecuaciones de Tindell y Clark. Esto hace que la demostración de la técnica de análisis esté completa y, por tanto, podamos continuar usándola y confiando en resultados previos obtenidos. Además, la técnica de análisis original es más rápida que la propuesta en [SUN96], puesto que el número de eventos que necesita chequear es menor.

Un concepto importante que se utiliza en el cálculo del tiempo de respuesta de peor caso de una acción es el de *periodo de ocupación de nivel i*, que corresponde a un intervalo continuo de tiempo durante el cual el procesador está ocupado ejecutando acciones de prioridad mayor o igual que *i* [LEH90]. En particular interesa conocer la duración del *periodo máximo de ocupación* o *periodo de ocupación de peor caso*, definido como el mayor de todos los posibles periodos de ocupación. Su comienzo coincide con un instante crítico y finaliza cuando todas las activaciones pendientes de acciones con prioridad mayor o igual que *i* han sido procesadas. Para la construcción de ese máximo periodo de ocupación clasificaremos las activaciones de cada acción a_j en los siguientes conjuntos:

- *Conjunto 0*: Activaciones de a_j ocurridas antes del instante crítico y que no podrían ocurrir en el instante crítico aunque se produjeran con el máximo retraso posible. Este retraso se produce cuando la acción previa, a_{j-1}, ejecuta bajo condiciones de peor caso, es decir, con un tiempo de respuesta igual al de peor caso, R_{j-1}.

- *Conjunto 1*: Activaciones que se producen en el instante crítico o que se pudieran retrasar hasta el instante critico.

- *Conjunto 2*: Activaciones producidas después del instante crítico.

Suponiendo que el uso de recursos compartidos se planifica según el protocolo de no expulsión o protección de prioridad o techo de prioridad, y una vez clasificadas las activaciones de cada acción a_j con prioridad mayor o igual que i en los tres conjuntos anteriores, el periodo de ocupación máximo de nivel i se construye de acuerdo con el siguiente teorema:

Teorema 3-1. El periodo de ocupación de peor caso de nivel i se construye cuando se dan las siguientes dos condiciones a) cada acción a_j de igual o mayor prioridad que i se activa según el siguiente patrón: las activaciones pertenecientes al *Conjunto 1* sufren un retraso tal que coinciden con el instante crítico, y las activaciones pertenecientes al *Conjunto 2* ocurren con retraso nulo, y b) cuando al comienzo del periodo de ocupación se acaba de bloquear el recurso con mayor sección crítica de entre aquellos compartidos por alguna acción de prioridad menor que i y cuyo techo de prioridad sea mayor o igual que el nivel i.

Demostración: Llamaremos t_c al instante crítico en que comienza el periodo de ocupación de peor caso. Según la definición del periodo de ocupación, las activaciones del *Conjunto 0* no pueden afectar a su ejecución, ya que si así fuera, el periodo de ocupación podría comenzar antes y, por tanto, el instante t_c elegido no sería crítico.

Para las activaciones del *Conjunto 1*, debemos considerar los términos de retraso que causen que cada activación ocurra dentro del periodo de ocupación, ya que si ocurrieran antes no podrían contribuir al periodo de ocupación considerado. Pero si el retraso hace que esa activación se produzca después del instante crítico, podría ocurrir fuera del periodo de ocupación. Por ello, la contribución máxima en el periodo de ocupación se asegurará si cada activación se produce en el instante crítico.

Para las activaciones del *Conjunto 2* debemos tener en cuenta que ocurren después del instante crítico, de forma que cuanto más se retrasen más probable es que ocurran fuera del periodo de ocupación. Por tanto, la mayor contribución de estas activaciones se conseguirá cuando las activaciones se produzcan sin retraso, es decir, cuando la acción previa, a_{j-1}, ejecuta bajo condiciones de mejor caso, con un tiempo de respuesta igual al de mejor caso, R^b_{j-1}.□

Mediante este teorema, podemos construir la situación de peor caso que conduce a la obtención del tiempo de respuesta máximo de una acción a_i, tal como demostramos en el siguiente teorema:

Teorema 3-2. El tiempo de respuesta global de peor caso de una acción a_i ocurre dentro del periodo de ocupación de peor caso de nivel igual a la prioridad asignada a a_i.

Demostración: Supongamos que el tiempo de respuesta de peor caso de la acción a_i se produce dentro de un periodo de ocupación concreto iniciado en un instante t_c. Llamemos I al conjunto (posiblemente vacío) de activaciones de la acción a_i ocurridas coincidiendo con el instante t_c, y supongamos que cada una de ellas ha sufrido un retraso comprendido entre cero (activándose un tiempo R^b_{i-1} después de ocurrido el evento) y el máximo permitido (activándose un tiempo R_{i-1} después del evento).

Supongamos ahora que movemos hacia atrás (esto es, adelantamos en el tiempo) la llegada de eventos asociados a la acción a_i y, simultáneamente, incrementamos en la misma cantidad el retraso de todas las activaciones de I, de manera que éstas se siguen produciendo en el mismo instante que antes, mientras dejamos invariables los retrasos del resto de activaciones (no pertenecientes al conjunto I) de forma que esas activaciones se producen más pronto que antes. Bajo esas condiciones, podremos adelantar la llegada de eventos hasta que se produzca alguna de las siguientes situaciones: a) una de las activaciones de I alcanza su máximo retraso, o b) una de las activaciones no pertenecientes al conjunto I alcanza el instante t_c. En el caso b), podemos añadir esa activación al conjunto I y continuar el proceso adelantando la llegada de eventos de forma iterativa hasta que se alcancemos la condición a), bajo la cuál la activación coincidente con el instante t_c experimenta su máximo retraso.

Nótese que durante este proceso, ninguna de las activaciones que pertenecían al periodo de ocupación original se ha adelantado al instante t_c y, por tanto, todas las activaciones del periodo de ocupación original (incluida aquella que originaba la respuesta de peor caso) permanecen en él. Además, al adelantar el instante de ocurrencia del evento, conseguimos aumentar el tiempo de respuesta global de la acción a_i, al estar referido a un instante anterior. Si en esta situación hacemos nulos los retrasos de las

activaciones posteriores al instante t_c (es decir, las no incluidas en el conjunto I), no modificaremos el esquema de ejecución ya que, según la definición de periodo de ocupación, entre el nuevo instante de activación y el anterior sólo podrían estar ejecutando tareas de prioridad igual o mayor que la asignada a la acción a_i y, por tanto, su tiempo de respuesta no variará.

Fijémonos ahora en el resto de tareas del sistema de mayor o igual prioridad que la acción a_i y que interfieren su ejecución, en el mismo periodo de ocupación. Si repetimos el razonamiento anterior, adelantando los eventos hasta la condición límite de retraso máximo para cada conjunto I, pero permaneciendo siempre las acciones en el periodo de ocupación, conseguimos que las activaciones que interferían lo sigan haciendo y, además, podemos conseguir que alguna o algunas activaciones que previamente ocurrían después de finalizado el periodo de ocupación puedan ocurrir ahora dentro de él, incrementando así el tiempo de respuesta para la acción a_i. Si además disminuimos, hasta hacer nulos, los retrasos correspondientes a eventos ocurridos posteriormente al instante t_c, podemos hacer que todavía más activaciones que previamente ocurrían después del periodo de ocupación ocurran ahora dentro, interfiriendo todavía más la ejecución de la acción a_i.

Por último, si forzamos a que, justo en el instante t_c, se haya bloqueado aquel recurso compartido por a_i con tareas de menor prioridad que dé como resultado el máximo tiempo de bloqueo, conseguiremos la máxima interferencia.

Estas condiciones de retrasos y bloqueos son precisamente las definidas en el Teorema 3-1 para la construcción del periodo máximo de ocupación. Esto garantiza que alguna activación de a_i dentro del periodo de ocupación de peor caso de nivel igual a la prioridad asignada a la acción a_i tendrá con toda seguridad un tiempo de respuesta igual al de peor caso.

□

Este teorema ya han sido, en parte, enunciado por Tindell [TIN93D], considerando tiempos de respuesta de mejor caso nulos. En esta tesis aparece para formalizar el caso en que los tiempos de respuesta de mejor caso no nulos [PAL98A], e incluimos su demostración para

hacerla acorde a la de los teoremas enunciados en los capítulos 4 y 5, referentes también a la creación de instantes críticos para la ejecución de una tarea.

Si consideramos origen de tiempos en el instante en que comienza el periodo de ocupación, el evento externo para la primera activación dentro del periodo máximo de ocupación habrá ocurrido en el instante $t = -R_{j-1}$. Por tanto, el evento externo p-ésimo llegará en el instante $t = -R_{j-1}+pT_j$. Esto significa que la activación de a_j correspondiente a ese evento se producirá como muy pronto en $t = -R_{j-1}+pT_j+R^b_{j-1} = pT_j-J_j$ (o en t=0 si el resultado de esto es negativo), para p=1,2,3...; consecuentemente, las primeras $\lceil J_j/T_j \rceil$ activaciones de cada a_j llegan al comienzo del periodo de ocupación y las siguientes llegan periódicamente bajo las condiciones de mejor caso para la ejecución de la acción previa. De esa forma, concentramos más trabajo cerca del instante crítico, originando así el periodo de ocupación más largo posible.

Llamaremos p_b a la última activación de a_i en el periodo de ocupación, de forma que éste será igual a $w_i(p_b)$, es decir, el instante de finalización de esa última activación. Obviamente, p_b corresponderá a la primera activación de la acción a_i cuyo procesado finalice antes de la llegada de una nueva (de otra forma, el periodo de ocupación incluiría esa nueva activación). Por tanto, p_b corresponde al primer índice que verifica:

$$R_i(p_b) \leq T_i + R^b_{i-1} \tag{1}$$

Según vimos en el capítulo 2, el tiempo de respuesta se calcula a partir del tiempo de finalización con la ecuación siguiente,

$$R_i(p_b) = w_i(p_b) - (p_b-1)T_i + R_{i-1} \tag{2}$$

y el máximo retraso en la activación mediante:

$$J_i = R_{i-1} - R^b_{i-1} \tag{3}$$

Si expresamos el tiempo de respuesta en función de ese retraso máximo

$$R_i(p_b) = w_i(p_b) - (p_b-1)T_i + J_i + R^b_{i-1} \tag{4}$$

y sustituimos en la ecuación (1), llegamos a la expresión siguiente:

$$w_i(p_b) - (p_b-1)T_i + J_i + R^b_{i-1} \leq T_i + R^b_{i-1} \tag{5}$$

de donde obtenemos,

$$w_i(p_b) \leq p_bT_i - J_i \tag{6}$$

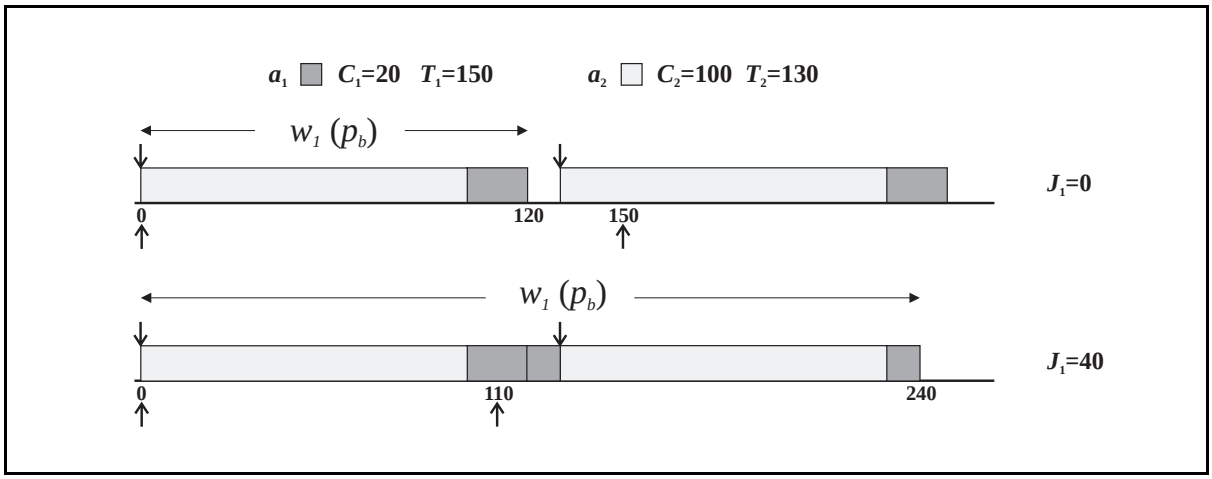


Figura 3-1. Ejemplo que muestra el efecto del retraso sobre el periodo de ocupación

Como se puede comprobar en (6), si aumenta el retraso J_i en la activación de la acción a_i para un determinado periodo de ocupación, la desigualdad puede ya no cumplirse y, por tanto, podemos necesitar calcular un nuevo periodo de ocupación más largo.

En la Figura 3-1 se muestra un ejemplo en el que se puede observar esta influencia de la activación retrasada sobre el periodo de ocupación. El ejemplo está compuesto por dos acciones periódicas a_1 y a_2, cuyos tiempos de ejecución y periodos se indican en la figura. La prioridad de a_2 es mayor que la de a_1. Si las activaciones de la acción a_1 no sufren retraso, el procesado de la primera activación termina en $w_1(1)$=120 y, dado que la siguiente activación se produce en el instante pT_1-J_1=150, no queda ningún evento pendiente y, por tanto, finaliza su periodo de ocupación. Esto es, p_b=1 y $w_1(p_b)$=120.

Sin embargo, si J_1=40, la primera activación se completa igualmente en $w_1(1)$=120 pero la siguiente activación se ha adelantado hasta el instante t=1×150-40=110, por lo que su ejecución estará todavía pendiente de procesado cuando se ha completado el trabajo correspondiente a la primera. Esto significa que el periodo de ocupación debe aumentar hasta incluir, al menos, la ejecución de esta segunda activación. Como se puede ver en la Figura 3-1, ahora p_b=2 y la longitud del periodo de ocupación es $w_1(p_b)$=240, ya que este tiempo es menor que el instante de activación del siguiente evento (correspondiente a p=3), que se producirá en t=2×150-40=260.

En el análisis del tiempo de respuesta de peor caso de un evento, de acuerdo con el análisis de Lehoczky para plazos arbitrarios [LEH90] sobre el cuál se basa el análisis de

Tindell y Clark, se deben chequear los tiempos de respuesta de todas las activaciones pertenecientes al periodo de ocupación. Sin embargo, tal como señalan Sun y Liu en [SUN96], el análisis de Tindell y Clark para una acción a_i no tiene en cuenta el término J_i correspondiente al retraso en las activaciones de a_i y, por tanto, no considera la totalidad del periodo de ocupación. Para una acción a_i con retraso J_i, el análisis finaliza cuando se encuentra el primer p_0 que verifica la condición:

$$w_i(p_0) \leq p_0 T_i \tag{7}$$

Por tanto, puesto que el análisis no chequea todas las activaciones producidas dentro del periodo de ocupación, los resultados del análisis podrían ser incorrectos. Sun y Liu proponen [SUN96] una técnica de análisis que es la misma (aunque con diferente terminología), pero que chequea todas las activaciones del periodo de ocupación, usando el término del retraso en el cálculo de la longitud del periodo de ocupación.

En esta sección vamos a demostrar que la condición de parada usada por Tindell y Clark es suficiente para determinar el tiempo de respuesta global de peor caso y, por tanto, sus resultados siguen siendo válidos aunque la propia acción analizada sufra retraso en sus activaciones.

Sea a_i una acción periódica planificada en un recurso procesador bajo planificación dinámica por prioridades fijas, con tiempo de ejecución de peor caso C_i, periodo T_i, máximo retraso en su activación J_i y término de bloqueo B_i. Supongamos que tenemos en el mismo procesador un conjunto de acciones periódicas a_j, $j \in hp(i)$, con prioridad mayor o igual que la prioridad de a_i, cada una de ellas con tiempo de ejecución de peor caso C_j, periodo T_j y retraso máximo J_j. Sea p_0 el número de activaciones de la acción a_i que se chequean en el análisis de Tindell y Clark.

Teorema 3-3: Dada una activación s posterior a p_0 en el periodo de ocupación de la acción a_i que comienza en un instante crítico ($p_0 < s \leq p_b$), existe una activación $u \in [1, p_0]$ cuyo tiempo de respuesta es mayor o igual que el tiempo de respuesta correspondiente a la activación s.

Demostración: Para cualquier valor $p \in [1,s]$, elijamos p' igual a $s-p$, y fijémonos en el instante $w_i(s)=w_i(p+p')$ —perteneciente al periodo de ocupación— en el cual se completa el procesado de las primeras $p+p'$ activaciones de a_i. Sin pérdida de generalidad, supondremos el origen de tiempos al comienzo del periodo de ocupación. En la Figura 3-2 podemos ver la representación de un ejemplo del periodo de ocupación para la acción a_i, donde cada activación de a_i está representada por una flecha ascendente, y el resto de activaciones, correspondientes a otras acciones, con flechas descendentes. Dado que $w_i(p+p')$ es el tiempo de finalización, en el peor caso, de las primeras $p+p'$ activaciones de la acción a_i, tenemos:

$$w_i(p+p') = (p+p')C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i(p+p') + J_j}{T_j} \right\rceil C_j \qquad (8)$$

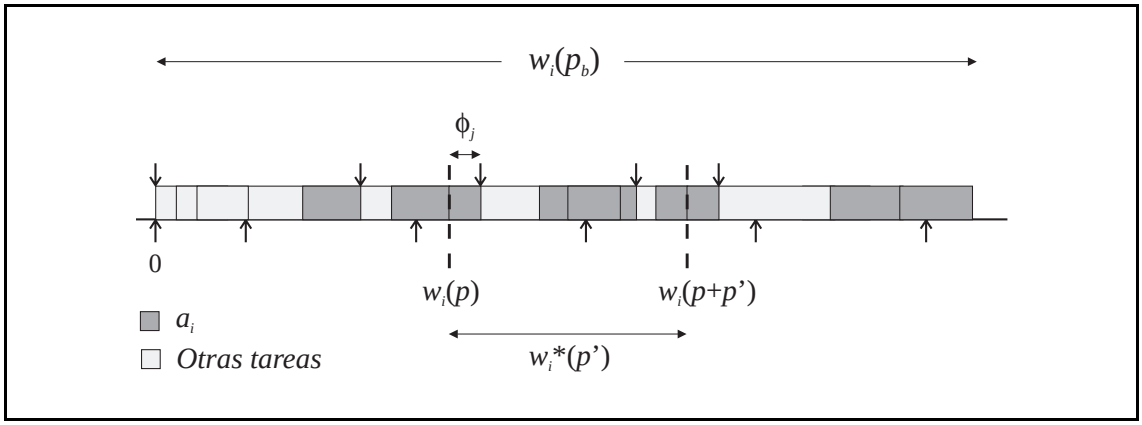


Figura 3-2. Ejemplo de un periodo de ocupación para la acción a_i

Como se puede ver en la Figura 3-2, podemos expresar este tiempo de finalización como:

$$w_i(p+p') = w_i(p) + w_i^*(p') \qquad (9)$$

donde, $w_i(p)$ es el tiempo de finalización en el peor caso de las primeras p activaciones, y $w_i^*(p')$ es el tiempo de procesado de las siguientes p' activaciones, relativo al instante $w_i(p)$. Se puede obtener una expresión para calcular $w_i^*(p')$, similar a la ecuación (8), considerando los siguientes puntos:

- El término de bloqueo B_i sólo afecta a la primera activación de a_i, ya que representa un retraso causado por tareas de menor prioridad que, debido a su nivel de prioridad, no pueden ejecutan durante el periodo de ocupación

considerado. Por tanto, este término ya ha sido tomado en consideración para el cálculo de $w_i(p)$.

- Bajo las condiciones creadas para el periodo de ocupación de peor caso, todas las activaciones correspondientes a acciones a_j que ocurren después del instante crítico que da comienzo al periodo de ocupación son perfectamente periódicas. Por tanto, ninguna de las activaciones de las acciones a_j posteriores al instante $w_i(p)$ sufren retraso, por lo que no es necesario el término J_j.

- Cuando construimos el instante crítico en t=0, todas las acciones a_j se activan al mismo tiempo que a_i. Sin embargo, $w_i(p)$ no es necesariamente un nuevo instante crítico para a_i, y, por tanto, entre $w_i(p)$ y la primera activación de a_j posterior a ese instante $w_i(p)$, puede haber un lapso de tiempo no nulo. Llamaremos a ese tiempo ϕ_j, y le incluiremos en la expresión para el cálculo de $w_i^*(p')$. Puesto que el número de activaciones de a_j que han llegado en el periodo de ocupación hasta el instante $w_i(p)$ viene dado por:

$$\left\lceil \frac{w_i(p) + J_j}{T_j} \right\rceil \tag{10}$$

la siguiente activación de a_j posterior a $w_i(p)$ ocurrirá en el instante

$$\left\lceil \frac{w_i(p) + J_j}{T_j} \right\rceil T_j - J_j \tag{11}$$

y, por tanto, el intervalo ϕ_j será igual a:

$$\phi_j = \left(\left\lceil \frac{w_i(p) + J_j}{T_j} \right\rceil T_j - J_j \right) - w_i(p) \tag{12}$$

Teniendo en cuenta esas consideraciones, podemos calcular $w_i^*(p')$ como:

$$w_i^*(p') = p'C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^*(p') - \phi_j}{T_j} \right\rceil C_j \tag{13}$$

Dado que B_j, J_j y ϕ_j son siempre mayores o iguales que 0, se puede observar que para cualquier valor k, la función $w_i^*(k)$ siempre es menor o igual que $w_i(k)$. En particular para p' se verifica:

$$w_i^*(p') \leq w_i(p') \tag{14}$$

Consideremos ahora el máximo tiempo de respuesta para la acción a_i. De acuerdo con la expresiones (4) y (9), tenemos que:

$$\begin{aligned} R_i(p+p') &= J_i + w_i(p+p') - (p+p'-1)T_i + R_{i-1}^b = \\ &= J_i + w_i(p) + w_i^*(p') - (p-1)T_i - p'T_i + R_{i-1}^b = \\ &= R_i(p) + w_i^*(p') - p'T_i \end{aligned} \tag{15}$$

Teniendo en cuenta la propiedad definida en (14), concluimos la siguiente relación:

$$R_i(p+p') \leq R_i(p) + w_i(p') - p'T_i \qquad \forall\, p,\ p' \mid p+p' \leq p_b \tag{16}$$

Si consideramos p' igual a p_0 (el límite usado en el análisis de Tindell y Clark), tenemos $p=s-p_0$, con lo que podemos reescribir la ecuación (16) en la forma:

$$R_i(s) \leq R_i(s-p_0) + w_i(p_0) - p_0T_i \tag{17}$$

De acuerdo a la definición de p_0, tenemos:

$$w_i(p_0) \leq p_0T_i \quad \Rightarrow \quad R_i(s-p_0) + w_i(p_0) - p_0T_i \leq R_i(s-p_0) \tag{18}$$

que, junto con (17), conduce a la siguiente desigualdad:

$$R_i(s) \leq R_i(s-p_0) + w_i(p_0) - p_0T_i \leq R_i(s-p_0) \tag{19}$$

Por tanto, podemos concluir que $R_i(s-p_0) \geq R_i(s)$ para cualquier valor de $s \in (p_0,p_b]$. Si $s-p_0$ es mayor que p_0, usando el mismo resultado podemos decir que $R_i(s-2p_0) \geq R_i(s-p_0)$ e, iterativamente, para cualquier valor de $s \in (p_0,p_b]$ encontramos un valor $u = s - \lfloor (s-1)/p_0 \rfloor\, p_0$, $u \in [1,p_0]$ que verifica $R_i(u) \geq R_i(s)$.□

Como consecuencia del Teorema 3-3, podemos concluir que el algoritmo de Tindell y Clark obtiene efectivamente cotas superiores de los tiempos de respuesta de las acciones en un sistema distribuido.

3.3. Explorando la reducción del número de casos a comprobar en el test de planificabilidad

La necesidad de iterar la expresión del tiempo de respuesta de peor caso para diferentes valores de *p* viene causada por la presencia de múltiples activaciones de la tarea bajo análisis dentro del periodo de ocupación. La ejecución de activaciones previas de una acción puede retrasar la ejecución de activaciones posteriores de la misma acción. En algunos casos, este efecto de retraso puede incrementarse de una activación a otra, causando que el tiempo de respuesta de peor caso no ocurra necesariamente en la primera activación [LEH90]. Eventualmente, el efecto del retraso decrece hasta un punto en el cual una activación tiene tiempo para completarse antes de que se produzca una nueva activación y, por tanto, finaliza el periodo de ocupación. Esta situación siempre se da en sistemas cuyo factor de utilización sea menor que el 100%, ya que el recurso (es decir, el procesador o la red de comunicación) quedará desocupado en algún momento.

Recientemente Alan Burns ha sugerido que este comportamiento del retraso causado por activaciones previas en el periodo de ocupación podría ser explotado para mejorar el test de planificabilidad, chequeando un número menor de activaciones en el periodo de ocupación. La idea podría ser detener el análisis cuando el tiempo de respuesta de una acción decreciera respecto de la activación anterior en el periodo de ocupación. Hemos estudiado esta posibilidad y encontrado varios contraejemplos en los cuales el tiempo de respuesta crece y decrece varias veces, y en los cuales el tiempo de respuesta de peor caso se obtiene después de que el tiempo de respuesta hubiera decrecido durante varias activaciones consecutivas. A continuación mostramos uno de esos contraejemplos.

Tarea	C_i	T_i	R_i
τ_1	30	100	30
τ_2	10	130	40
τ_3	10	190	50
τ_4	46	85	110

Tabla 3-I. Parámetros del contraejemplo.

Considérese un único procesador con cuatro tareas independientes que, por simplicidad, se activan periódicamente sin retraso. Sus tiempos de ejecución de peor caso y sus periodos se muestran en la Tabla 3-I. Las tareas se han numerado en orden decreciente de prioridad, con la prioridad más alta correspondiente a la tarea con índice menor. La tabla muestra también los tiempos de respuesta de peor caso obtenidos para cada tarea.

p	$w_4(p)$	$(p\text{-}1)T_4$	$R_4(p)$
1	96	0	96
2	182	85	97
3	278	170	108
4	354	255	99
5	450	340	110
6	496	425	71

Tabla 3-II. Análisis de las activaciones de τ_4

Centremos nuestra atención en la tarea de prioridad menor, τ_4. La Tabla 3-II muestra los resultados obtenidos por las ecuaciones (40) y (41) del capítulo anterior, para cada activación de esta tarea dentro del periodo de ocupación. Como se puede ver, el tiempo de respuesta obtenido para cada activación va incrementando hasta la tercera activación, para la cual encontramos un primer máximo local igual a R_4=108. Sin embargo, podemos ver que el tiempo de respuesta de peor caso ocurre en la quinta activación, con un tiempo de respuesta R_4=110. Consecuentemente, el criterio para detener el análisis en cuanto los tiempos de respuesta decrezcan no es válido.

3.4. Cálculo de los tiempos de respuesta locales de peor caso

Las técnicas de análisis que vimos en el capítulo 2 sólo sirven para determinar los tiempos de respuesta globales de peor caso, medidos desde el instante en que se produjo la llegada del evento externo. En esta sección encontraremos cotas superiores a los tiempos de respuesta de peor caso locales, relativos al instante de activación de la acción concreta.

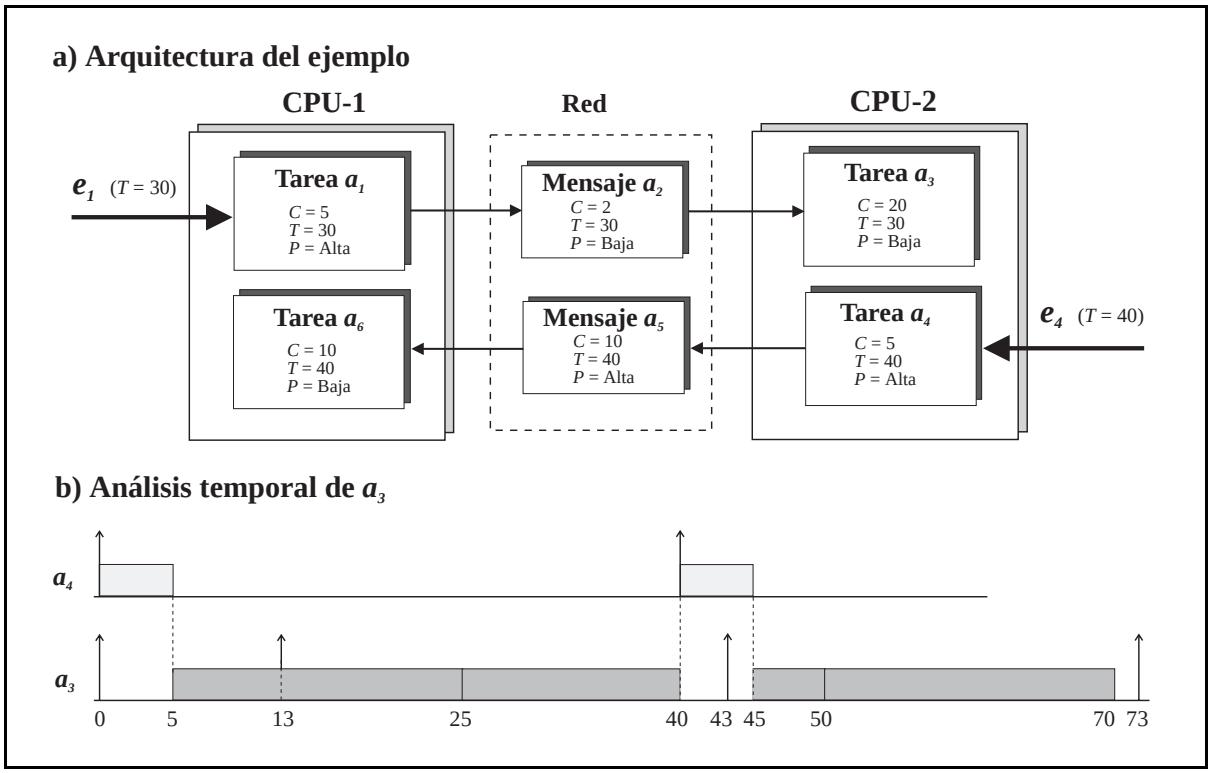


Figura 3-3. Ejemplo para el cálculo de los tiempos de respuesta locales de peor caso

La Figura 3-3 muestra un ejemplo sencillo de sistema distribuido que utilizaremos para ilustrar como encontrar una buena estimación de los tiempos de respuesta locales de peor caso. En la Figura 3-3-a se presenta la arquitectura del sistema, constituido por dos procesadores y una red de comunicaciones de tiempo real. Hay dos eventos externos que llegan al sistema, y que generan dos respuestas distribuidas entre los diferentes recursos. Suponemos para todas las tareas y mensajes tiempos de respuesta de mejor caso nulos. Si aplicamos el análisis de tiempo real sobre cada acción de este sistema obtenemos los términos de retraso y los tiempos de respuesta globales de peor caso que se muestran en la Tabla 3-III.

Acción	J_i	R_i
a_1	0	5
a_2	5	17
a_3	17	42
a_4	0	5
a_5	5	15
a_6	15	30

Tabla 3-III.Análisis temporal del ejemplo

La Figura 3-3-b muestra el diagrama temporal para la acción a_3 después de un instante crítico, que usaremos para estimar su tiempo de respuesta local de peor caso. El peor caso

para una acción a_i ocurre cuando la primera activación sufre el máximo retraso y las siguientes activaciones se producen sin él, de forma que se requiera la mayor cantidad de tiempo de procesado cerca del instante crítico. Para la acción a_3, el peor caso corresponde a la situación en la cual es activada al mismo tiempo que a_4 y la segunda activación de a_3 ocurre en el instante más cercano posible, igual al periodo menos su retraso máximo (t=13). Esta situación produce el periodo de ocupación más largo, con tres activaciones de la tarea a_3. El tiempo de respuesta local de peor caso se determina entre esas tres activaciones del periodo de ocupación: la primera activación tiene un tiempo de respuesta de 25 unidades de tiempo, la segunda 50-13=37 y la tercera 70-43=27, por tanto, el tiempo local de peor caso para la acción a_3 es 37 unidades de tiempo.

A través de este ejemplo vemos ver que es posible estimar mejores cotas del tiempo de respuesta local que las obtenidas si consideramos el tiempo de respuesta global de peor caso como cota superior del tiempo de respuesta local de peor caso (en este ejemplo serían 42 unidades de tiempo).

A partir de los tiempos de finalización y retrasos máximos obtenidos con el análisis para los diferentes valores de p, podemos definir el tiempo de respuesta local de peor caso para la activación p-ésima de la acción a_i de la forma siguiente:

$$r_i(p) = \begin{cases} w_i(p) & si\ p = 1 \\ w_i(p) - (p-1)T_i + J_i & si\ p \geq 2 \end{cases} \tag{20}$$

La justificación para esta función se basa en el criterio que usamos para construir el instante crítico. Los tiempos de respuesta locales se miden respecto al instante en que se produce la activación de la acción analizada, por tanto, la primera activación (producida en el instante t=0) tiene un tiempo de respuesta local igual a $w_i(1)$-0 y el resto de activaciones, puesto que se producen adelantadas una cantidad J_i respecto al periodo, se activan en $(p\text{-}1)T_i$-J_i, por lo que su tiempo de respuesta local será igual a $w_i(p)$-$[(p\text{-}1)T_i\text{-}J_i]$.

Nótese que con la definición anterior, para $p \geq 2$ todavía es aplicable la ecuación (15) a los tiempos de respuesta locales:

$$r_i(p+p') = r_i(p) + w_i^*(p') - p'T_i \tag{21}$$

Por consiguiente, los mismos resultados que aparecían en las ecuaciones (16) a (18) son aplicables también a los tiempos de respuesta locales y podemos llegar, en definitiva, a una desigualdad similar a la (19):

$$r_i(s) \leq r_i(s-p_0) \tag{22}$$

Por tanto, para $p \geq 2$ podemos descartar el cálculo de tiempos de respuesta para activaciones posteriores a la p_0+1 en el periodo de ocupación donde p_0 corresponde al número de activaciones que se deben chequear en el algoritmo de Tindell y Clark definido mediante la condición dada en (7).

Como la ecuación (21) no es aplicable para $p=1$, no podemos utilizar (22) para descartar la activación p_0+1, y por tanto, debemos chequear su tiempo de respuesta, si es que esa activación pertenece al periodo de ocupación. El número de activaciones que debemos chequear para determinar el tiempo de respuesta local de peor caso de una acción a_i será entonces igual a $\min(p_0+1, p_b)$.

Una vez que hemos obtenido los valores de $r_i(p)$ para las activaciones necesarias, podemos calcular el tiempo de respuesta de peor caso como el mayor de todos:

$$r_i = \max\ [r_i(p)] \quad \forall\ p=1,2,3,...,\min(p_0+1, p_b) \tag{23}$$

Para determinar la planificabilidad de cada acción con plazo local asociado, se deben comparar los tiempos de respuesta locales de peor caso (r_i) con los respectivos plazos locales de ejecución (d_i).

p	J_3	$w_3(p)$	$r_3(p)$
1	17	25	25
2		50	37

Tabla 3-IV. Análisis temporal de a_3

La Tabla 3-IV muestra los resultados de aplicar el análisis descrito para calcular el tiempo de respuesta local de peor caso de la acción a_3, en el ejemplo de la Figura 3-3. Se puede ver que $p_0=1$, puesto que $w_3(1) \leq T_3$ y, por tanto, el análisis debe considerar solamente las primeras dos activaciones en el periodo de ocupación. El tiempo de respuesta local de peor caso obtenido es igual a 37 unidades de tiempo, el mismo valor que obtuvimos por inspección del periodo de ocupación en la Figura 3-3-b.

3.5. Análisis de mejor caso para mejorar el análisis de peor caso

Como vimos en el capítulo 2 de esta tesis, las técnicas de análisis que suministra la teoría *RMA* para sistemas distribuidos obtienen cotas superiores de los tiempos de respuesta de las tareas y los mensajes. El cálculo exacto de esos tiempos de respuesta es, por tanto, un tema todavía abierto. En esta sección vamos a mostrar una forma de reducir el pesimismo en el análisis de planificabilidad, reduciendo la estimación de los retrasos en las activaciones de tareas y mensajes del sistema distribuido. Vimos que, en un sistema de estas características, el retraso en la activación de una acción viene determinado por la diferencia entre los tiempos de respuesta en el peor y en el mejor caso de la acción precedente en la secuencia de respuesta al evento externo a la cual pertenece. En el test de planificabilidad desarrollado por Tindell y Clark [TIN94F] se considera el retraso máximo en la activación de una acción igual al tiempo de respuesta de peor caso estimado para la tarea precedente. El algoritmo de análisis empleado (al que llamaremos TRPC, Tiempo de Respuesta de Peor Caso), es el que se describe en la siguiente figura:

```
algoritmo TRPC is
begin
    Inicializar términos de retraso a cero;
    loop
        Calcular tiempos de respuesta de peor caso;
        Estimar nuevos términos de retraso, iguales a los
          tiempos de respuesta de las acciones precedentes;
        exit when no planificable or
            no variación desde la última iteración;
    end loop;
end TRPC;
```

Por supuesto, siempre es posible eliminar completamente el efecto de las activaciones retrasadas, utilizando un algoritmo de planificación adecuado, tal como el servidor esporádico [SPR89A] o el protocolo de modificación de fase [BET92] [SUN96], incluso en la planificación de los mensajes en las redes de comunicación. Sin embargo, estos protocolos de planificación no siempre están disponibles y, para muchas aplicaciones, el coste se reduce fuertemente si se utilizan planificadores convencionales basados en prioridades fijas.

Hasta ahora, hemos supuesto que los tiempos de ejecución de las tareas o la longitud de los mensajes a transmitir pueden ser arbitrariamente pequeños. Esto implica que, bajo un escenario de ejecución de mejor caso, las tareas (o los mensajes) puedan procesarse sin que se produzca expulsión por parte de las tareas de mayor prioridad, de forma que el tiempo de respuesta pudiera ser igual al tiempo de ejecución y, por tanto, arbitrariamente pequeño. Esta suposición es correcta, desde el punto de vista que nos permite obtener cotas superiores válidas de los términos de retraso en cada tarea o mensaje, y, aplicando el algoritmo TRPC, obtener también cotas superiores válidas de los tiempos de respuesta de peor caso de las tareas y los mensajes.

Sin embargo, es bien conocido que en muchos sistemas el tiempo de ejecución de las tareas o la longitud de los mensajes a transmitir (y, por tanto, sus tiempos de transmisión) no son despreciables aún en el mejor caso. Si podemos estimar los tiempos de ejecución y de transmisión de mejor caso (o una cota inferior de los mismos), es posible disminuir el retraso estimado en la activación de cada acción y, consecuentemente, mejorar las estimaciones de los tiempos de respuesta de peor caso de las tareas o mensajes de menor prioridad a las que expulsan.

Para cada acción a_i definiremos el parámetro C_i^b como una cota inferior estimada para el tiempo de ejecución o trasmisión, de forma que a partir de ahora caracterizaremos cada acción mediante sus tiempos de ejecución en el mejor y en el peor caso. Si la ejecución de la acción es determinista ambos tiempos serán iguales, aunque en la mayoría de los casos serán diferentes. Nuestro problema ahora es calcular el tiempo de respuesta en el mejor caso para cada acción a_i o, al menos, una cota inferior del tiempo de respuesta de mejor caso real, R_i^b. Este valor será utilizado para obtener el termino de retraso J_i de cada acción a_i, como la diferencia entre las estimaciones en la respuesta en los casos peor y mejor para la acción precedente en la secuencia de respuesta al evento externo, esto es, $J_i = R_{i-1} - R_{i-1}^b$.

3.5.1. Estimación trivial del mejor caso

Un primer método para encontrar una cota inferior del tiempo de respuesta de mejor caso para una acción a_i consiste en considerar el caso trivial, en el cual la acción no es expulsada por ninguna de las tareas de mayor prioridad, y los recursos que pueda compartir

con otras acciones están a su disposición, de manera que su tiempo de respuesta global relativo a la llegada del evento externo es igual a su tiempo de ejecución de mejor caso, C_i^b, más el tiempo de ejecución de mejor caso de todas las acciones precedentes en la secuencia de respuesta al evento externo:

$$R_i^b = \sum_{\forall j \in A(i)} C_j^b \tag{24}$$

donde,

$A(i)$ es el conjunto de acciones formado por a_i, y las acciones que preceden a la acción a_i en la cadena de acciones que ejecutan como respuesta asociada al evento externo.

El término de retraso se calcula entonces según la expresión

$$J_i = R_{i-1} - R_{i-1}^b = R_{i-1} - \sum_{\forall k \in A(i-1)} C_k^b \tag{25}$$

Como se puede ver, el tiempo de respuesta obtenido con esta primera estimación es constante desde el punto de vista del algoritmo de análisis, ya que se puede obtener directamente de los parámetros que definen las tareas y los mensajes. Esto quiere decir que los términos de retraso tienen la misma tendencia respecto de los tiempos de respuesta, R_i que tenían en el algoritmo TRPC. En este algoritmo, el comportamiento monótono de las estimaciones para los tiempos de respuesta de peor caso (que se hacen cada vez mayores en cada iteración) garantiza la convergencia del algoritmo, bien porque encuentre una solución planificable o bien porque en alguna iteración se sobrepase el plazo asignado a alguna de las acciones, en cuyo caso el algoritmo se trunca. Este mismo criterio puede ser aplicado si sustituimos en el algoritmo original el cálculo de los términos de retraso, según la forma descrita en (25), por lo que la convergencia del nuevo algoritmo también está garantizada.

Nótese que los términos de retraso calculados según la ecuación (25) son siempre menores que los obtenidos cuando consideramos tiempos de respuesta de mejor caso nulos. Esto hace que el nuevo algoritmo sea capaz de encontrar soluciones planificables cuando el algoritmo TRPC original puede no conseguirlo.

3.5.2 Estimación iterativa del mejor caso

La estimación trivial, tal como veremos en la sección siguiente, obtiene buenos resultados en la mejora del test de planificabilidad. Sin embargo, es fácil darse cuenta de que se podrían obtener mejores estimaciones de los tiempos de respuesta de mejor caso si tuviéramos en cuenta los efectos de expulsión por tareas de mayor prioridad que pudieran darse aún en el caso más favorable para la ejecución de la tareas. En la Figura 3-4 ilustramos este efecto de expulsión con un ejemplo. El sistema está compuesto por dos procesadores y tres eventos externos, e_1, e_2 y e_4, que disparan la ejecución de tres secuencias de respuesta. La Figura 3-4-a muestra las tareas y los parámetros temporales que caracterizan a las tareas y a los eventos. A efectos de simplificar el ejemplo, supondremos que los tiempos de transmisión son despreciables frente a los tiempos de ejecución de las tareas, por lo que no consideraremos en el análisis el efecto de la red de comunicación entre los dos procesadores. De acuerdo con nuestro modelo, supondremos que los tres eventos son puramente periódicos, sin retraso en su activación. Esto implica que sólo la tarea τ_3 sufre retraso en su activación, ya que es la única tarea activada por la finalización de otra tarea (τ_2).

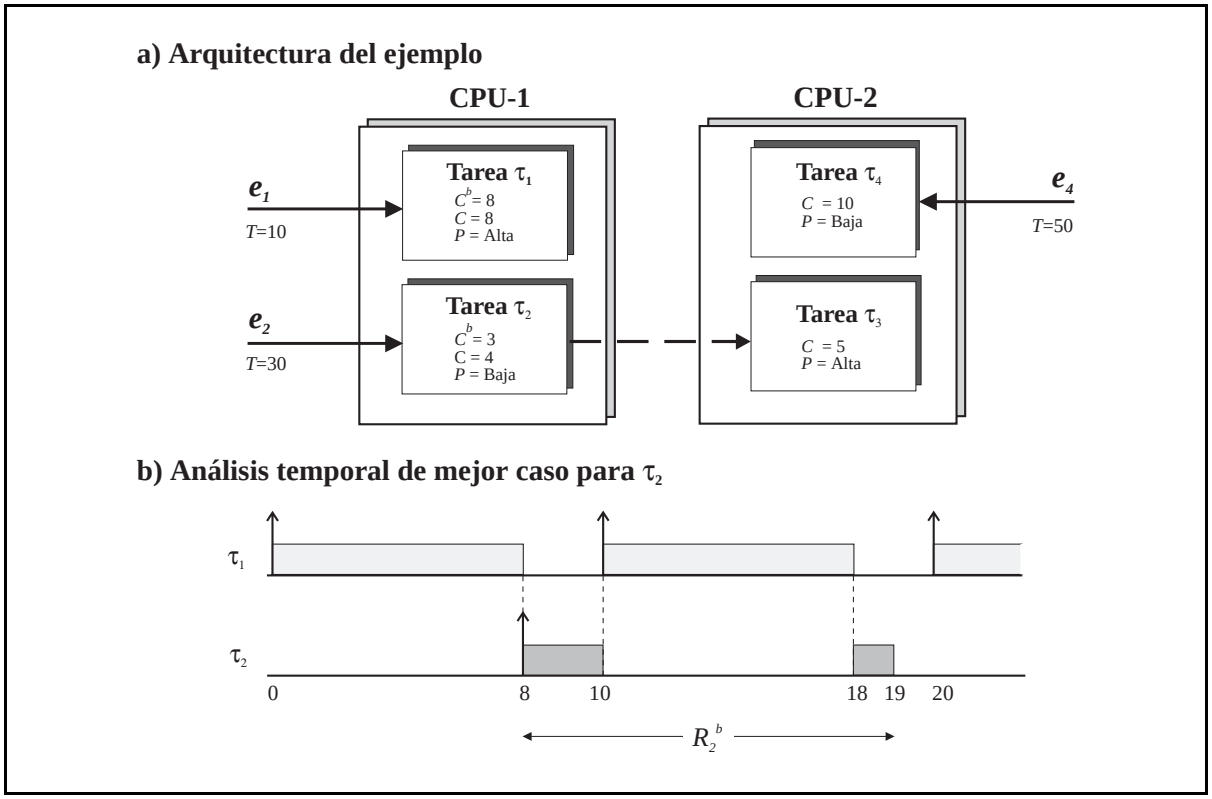


Figura 3-4. Ejemplo que ilustra el efecto de la expulsión sobre la respuesta de mejor caso

La Tabla 3-V muestra los resultados del cálculo de los tiempos de respuesta de peor caso, utilizando el algoritmo TRPC, y la estimación trivial de los tiempos de respuesta de mejor caso.

Tarea	J_i	R_i	R_i^b
τ_1	0	8	-
τ_2	0	20	3
τ_3	0	25	-
τ_4	17	20	-

Tabla 3-V. Resultados del ejemplo.

Si fijamos nuestra atención en la tarea τ_2 podemos ver que la estimación para su tiempo de respuesta de mejor caso es R_2^b=3, lo que causa que τ_3 se pueda activar con un retraso J_3=17. Este retraso origina que la tarea τ_4, que tiene un nivel de prioridad menor, sea expulsada dos veces por τ_3 en un escenario de ejecución de peor caso teniendo, por tanto, un tiempo de respuesta de peor caso R_4=20. Sin embargo, si nos fijamos en la Figura 3-4-b, podemos ver que la estimación realizada para el tiempo de respuesta de mejor caso de τ_2 no es realista, ya que le es imposible ejecutar sin ser expulsada una vez por τ_1. Esto significa que R_4^b=11 y, por tanto, la tarea τ_2 sufrirá un retraso máximo de J_2=20-11=9 unidades. Con este valor de retraso, la tarea τ_4 tendrá menor tiempo de respuesta, igual a R_4=15.

El ejemplo anterior muestra que la estimación trivial de los tiempos de respuesta de mejor caso no es realista, porque no tiene en cuenta los efectos de la expulsión por tareas de prioridad mayor. Necesitamos encontrar la manera de incluir este efecto de expulsión en el análisis, de igual forma que necesitamos incluirlo también en el análisis de peor caso. Obviamente, el instante crítico utilizado como punto de comienzo del periodo de ocupación en el peor caso no se puede usar aquí, y necesitamos, por tanto, encontrar un escenario de mejor caso sobre el que podamos analizar los tiempos de respuesta. Una posible aproximación que nos permite obtener una cota inferior del tiempo de respuesta de mejor caso viene introducida por el siguiente lema.

Lema 3-1: Sean a_i y a_j dos acciones alojadas en el mismo recurso, siendo a_i menos prioritaria que a_j. La menor contribución de la acción a_j al tiempo de respuesta de a_i ocurre cuando a_j se activa de forma que no tiene trabajo pendiente en el instante en que se activa a_i y, además, el resto de activaciones de a_j ocurren lo más retrasadas posible.

Demostración: Supongamos primero que la acción a_i se activara durante la ejecución de una activación de a_j. Si retrasamos el instante de activación de a_i hasta el instante en que finaliza la ejecución pendiente de la acción a_j sólo podemos mejorar el tiempo de respuesta de a_i, ya que la ejecución de a_i y todas las posibles expulsiones debidas a a_j ocurrirán en los mismos instantes, pero el tiempo de respuesta de a_i sería relativo a un instante de activación posterior.

Centrémonos ahora en las activaciones de a_j que se producen después de que se haya activado la acción a_i, y que influyen en el tiempo de respuesta de a_i. Si retrasamos esas activaciones tanto como nos sea posible, estaremos favoreciendo la ejecución de la acción a_i, ya que las expulsiones se producirán más tarde y, por tanto, la acción a_i tendrá más tiempo para ejecutar. Con esto, queda demostrado el lema.

□

Usando el Lema 3-1, podemos extender sus resultados para generar un escenario que nos permita obtener cotas inferiores de los tiempos de respuesta de mejor caso para cada acción a_i cuando ejecuta junto a un conjunto de acciones de prioridad más alta. Este escenario se construye suponiendo la situación descrita en el Lema 3-1 para cada tarea de mayor prioridad, tal como si fueran tareas independientes y no tuvieran ningún tipo de interacción o efectos de expulsión entre ellas previamente a la activación de a_i. Evidentemente, este escenario es ficticio ya que nunca se podría dar una situación así en la práctica porque no tiene en cuenta que las acciones deben cumplir ciertas restricciones en cuanto a sus instantes de activación: sería necesario que todas las acciones completaran la ejecución de sus activaciones pendientes simultáneamente, justo en el instante en que activara la acción a_i. Sin embargo, aunque esta situación sea virtualmente imposible, permite obtener cotas inferiores para el tiempo de respuesta de mejor caso de a_i, que en la realidad sólo podría ser mayor.

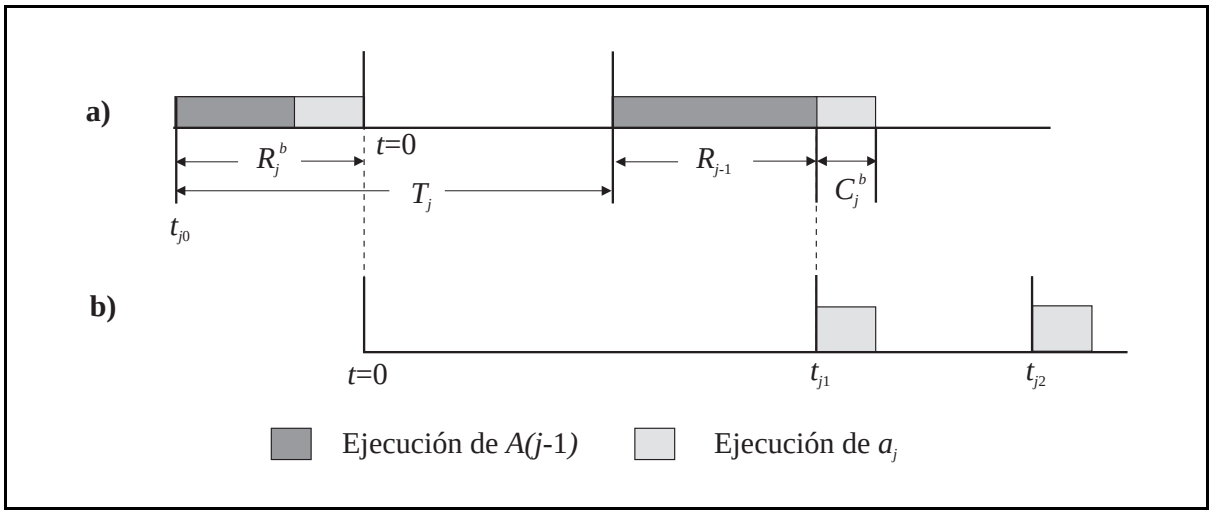


Figura 3-5. Construcción del escenario de mejor caso

A partir de ese escenario obtendremos las ecuaciones que nos permitirán realizar el análisis de mejor caso para una acción a_i. Para ello previamente obtendremos los instantes de activación de las acciones de mayor prioridad, según el esquema definido por el Lema 3-1. Sin pérdida de generalidad, supondremos el origen de tiempos t=0 en el instante en que se produce la activación de la acción a_i. Para cada acción a_j de mayor prioridad, ese origen de tiempos corresponde también al instante en que finaliza la ejecución de sus activaciones pendientes producidas antes de t=0.

Llamemos t_{j0} al instante en que se produjo la llegada del evento externo que dispara la secuencia de respuesta a la cual pertenece la acción a_j, que es la que finaliza en t=0 (ver Figura 3-5-a). En nuestro escenario de mejor caso necesitamos que ese instante t_{j0} se produzca lo más tarde posible, porque de esa forma hacemos que las siguientes activaciones de a_j (que pueden expulsar la ejecución de a_i) se produzcan también lo más tarde posible, ya que el evento es periódico. Para conseguir que t_{j0} sea lo más tarde posible, la acción a_j debería haber encontrado su tiempo de respuesta de mejor caso y, por tanto, el evento haberse producido en t_{j0}=-R_j^b. Fijémonos ahora en t_{j1}, el instante en que se produce la activación de la acción a_j posterior a t=0. Teniendo en cuenta las condiciones del Lema 3-1, debemos buscar el instante más retrasado posible para esa activación, que ocurrirá cuando la acción precedente a_{j-1}, que dispara la ejecución de a_j, encuentre su tiempo de respuesta de peor caso, R_{j-1}. Según esto, t_{j1} será igual al instante en que llegó el correspondiente evento externo, t_{j0}+T_j, más ese tiempo de respuesta R_{j-1}, esto es, t_{j1}=T_j+R_{j-1}-R_j^b, tal como se muestra en la Figura 3-5-a. Por último, las posteriores activaciones de a_j también se activarán en la forma más retrasada posible y,

por tanto, ocurrirán periódicamente, con un periodo T_j, después de instante t_{j1}. Para cada activación de a_j, consideraremos un efecto de expulsión equivalente a su tiempo de ejecución de mejor caso, C_j^b, como se muestra en la Figura 3-5-b.

Basándonos en los instantes de activación de las diferentes acciones en nuestro escenario de mejor caso, podemos derivar las ecuaciones que nos permitirán calcular una cota inferior del tiempo de respuesta en el mejor caso de una determinada acción. Para ello calcularemos una cota inferior de la expulsión de cada a_j sobre a_i, y posteriormente sumaremos los efectos de expulsión de todas las tareas de mayor prioridad.

Teorema 3-4: Una cota inferior para el efecto de expulsión de una acción a_j sobre otra acción de menor prioridad a_i en un intervalo de duración t viene dada por la ecuación:

$$W_j^b(t) = \left\lceil \frac{t-(T_j+R_{j-1}-R_j^b)}{T_j} \right\rceil_0 C_j^b \tag{26}$$

donde la función $\lceil x \rceil_0$ es:

$$\lceil x \rceil_0 = \max\left(0, \lceil x \rceil\right) \tag{27}$$

Demostración: Para probar este teorema obtendremos primero la función $W(t)$, que calcula la cantidad total de trabajo requerido por una secuencia infinita de activaciones de una acción, dentro del intervalo $[0,t)$. Cada activación n está determinada por el instante en el cual ocurre, ϕ_n, y por la cantidad de tiempo de ejecución requerida, c_n. La activación n-ésima deberá tenerse en cuenta en $W(t)$ si se produce dentro de ese intervalo, esto es, si $\phi_n<t$. Por tanto, considerando el conjunto total de activaciones en la secuencia tendremos:

$$W(t) = \sum_{n=0}^{\infty} \theta(t-\phi_n)\ c_n \tag{28}$$

donde, $\theta(x)$ es la función escalón, definida como 1 para $x>0$ y 0 para $x\leq 0$.

Si ahora creamos una secuencia de acciones periódicas con periodo T, fase inicial ϕ_0 y tiempo de ejecución c, tendremos:

$$\phi_n = \phi_0 + nT \quad , \quad c_n = c \qquad \forall\ n \tag{29}$$

con la cual, al aplicar la expresión (28) resulta:

$$W(t) = \sum_{n=0}^{\infty} \theta(t-\phi_0-nT)\ c = c \sum_{n=0}^{\infty} \theta(t-\phi_0-nT) \tag{30}$$

Si ahora tenemos en cuenta que:

$$\sum_{n=0}^{\infty} \theta(x-nT) = \max\left(0,\left\lceil \frac{x}{T} \right\rceil\right) = \left\lceil \frac{x}{T} \right|_0 \tag{31}$$

llegamos a:

$$W(t) = \left\lceil \frac{t-\phi_0}{T} \right|_0 c \tag{32}$$

En nuestro escenario de mejor caso, que crea una situación desfavorable para la expulsión, y, por tanto, una cota inferior de sus efectos, tenemos cada secuencia periódica con fase de activación $\phi_0=T_j+R_{j-1}-R_j^b$, y tiempo de ejecución C_j^b, por tanto, el teorema se verifica.□

Para calcular el tiempo de respuesta de mejor caso para la acción a_i debemos considerar la contribución de la propia ejecución de a_i más los efectos de expulsión de todas las tareas de mayor prioridad. El instante de finalización de la ejecución en el mejor caso se obtiene de la expresión:

$$w_i^b = C_i^b + \sum_{\forall j \in hp(i)} W_j^b(w_i^b) \tag{33}$$

que, sustituyendo por la expresión dada en el teorema 3-4, conduce a:

$$w_i^b = C_i^b + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^b - (T_j+R_{j-1}-R_j^b)}{T_j} \right|_0 C_j^b \tag{34}$$

La ecuación se puede resolver iterativamente, comenzando por un valor inicial $w_i^b=C_i^b$. El valor obtenido es una cota inferior del tiempo de respuesta de a_i, medido desde la activación de esa acción. El tiempo de respuesta global de mejor caso, considerado desde el instante en que llegó el evento externo se obtiene sumando a esa cantidad, el tiempo de respuesta global de mejor caso de la acción precedente, a_{i-1}:

$$R_i^b = w_i^b + R_{i-1}^b \tag{35}$$

En la ecuación (34) podemos ver que la obtención del instante de finalización de mejor caso requiere conocer los tiempos de respuesta en el mejor y en el peor caso de la diferentes acciones. Esta situación es similar a la que ocurría en el análisis del peor caso, donde el cálculo de los tiempos de respuesta de peor caso requería el conocimiento previo de tiempos de respuesta de peor caso. Tindell y Clark resolvieron el problema aplicando el análisis iterativamente, tal como describe el algoritmo TRPC. El comportamiento monótono de los tiempos de respuesta y los términos de retraso, no decreciendo de una iteración a otra, garantizaba la convergencia del algoritmo. El nuevo cálculo de los tiempos de respuesta de mejor caso implica introducir cambios en el algoritmo TRPC. El nuevo algoritmo creado, que llamaremos MTRPC, se muestra en la figura adjunta.

```
algoritmo MTRPC is
begin
    Inicializar tiempos de respuesta de peor y mejor caso;
    loop
        Calcular tiempos de respuesta de peor caso;
        Calcular tiempos de respuesta de mejor caso;
        Estimar nuevos retraso, iguales a los tiempos de
              peor caso menos los de mejor caso;
        exit when no planificable or
              no variación desde la última iteración;
    end loop;
end MTRPC;
```

Para asegurar la convergencia del algoritmo MTRPC, necesitamos verificar que los tiempos de respuesta de peor caso siguen teniendo un comportamiento monótono relativo a los cambios, tanto de los tiempos de respuesta de peor caso como de los de mejor caso, respecto a las iteraciones previas. En la ecuación (34) podemos ver que cuando los tiempos de respuesta de peor caso aumentan, los tiempos de respuesta de mejor caso disminuyen. También se puede ver que cuando los tiempos de respuesta de mejor caso disminuyen, los tiempos de mejor caso obtenidos en la siguiente iteración siguen disminuyendo. Finalmente, cuando los tiempos de mejor caso disminuyen, los términos de retraso aumentan, lo que provoca incrementos en los tiempos de respuesta de peor caso. Esto confirma el comportamiento monótono del algoritmo de análisis MTRPC, siempre y cuando el valor inicial de los tiempos de respuesta de mejor caso fuera mayor que los obtenidos en las sucesivas iteraciones.

Como se puede ver en la ecuación (34) se necesita encontrar un valor inicial para cada término de retraso, R_j^b. Esos valores deberían ser mayores (o como mínimo iguales) que los tiempos de respuesta de mejor caso que vayan a ser obtenidos finalmente por el algoritmo, si queremos garantizar el comportamiento monótono de los tiempos de respuesta y, en definitiva, la convergencia del algoritmo MTRPC. Un valor que verifica esa condición es el tiempo de respuesta de peor caso obtenido a partir de la ecuación definida en la sección 3 del capítulo 2, para valores de retraso nulos, J_j=0. Si consideramos dicha expresión para una acción a_i y para p=1, y la comparamos término a término con la expresión (34), veremos que:

$$\begin{aligned} C_i &\geq C_i^{\,b} \\ B_i &\geq 0 \\ \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i(1)}{T_j} \right\rceil C_j &\geq \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^{\,b} - (T_j + R_{j-1} - R_j^{\,b})}{T_j} \right\rceil_0 C_j^{\,b} \end{aligned} \tag{36}$$

Consecuentemente, como los sucesivos valores estimados para los tiempos de mejor caso son siempre menores que $w_i(1)$ (el tiempo de finalización para p=1), también serán menores que R_i, el mayor tiempo de respuesta de todas las activaciones del periodo de ocupación de peor caso, que es un valor mayor o igual que $w_i(1)$. Este valor (el tiempo de respuesta de peor caso para J_i=0) es precisamente el valor obtenido en la primera iteración del algoritmo MTRPC. De esta forma, aseguramos un valor inicial correcto para calcular los tiempos de respuesta de mejor caso y garantizar la convergencia del algoritmo.

3.5.3. Resultados de simulación

En esta sección mostraremos los resultados de las simulaciones que hemos realizado utilizando las dos técnicas de análisis para el mejor caso descritas en las secciones 3.5.1 y 3.5.2. El propósito de estas simulaciones es cuantificar la mejora obtenida en el análisis de sistemas distribuidos. En la sección 3.5.2 vimos, a través de un ejemplo, cómo se puede mejorar significativamente el tiempo de respuesta de una tarea dada si utilizamos la estimación iterativa del mejor caso, en lugar de la estimación trivial. En ese ejemplo, la mejora en el tiempo de respuesta era del 25%. Sin embargo, para estimar más adecuadamente esas mejoras, debemos aplicar ambos algoritmos de análisis a ejemplos más realistas, bajo diferentes condiciones. Para las simulaciones que se muestran en esta tesis se ha utilizado una herramienta previamente desarrollada [GUT95D] que implementa el algoritmo TRPC y que se ha adaptado para que contemple los nuevos algoritmos desarrollados para el cálculo de los tiempos de respuesta de mejor caso. Esta herramienta es capaz de generar, de forma automática, conjuntos de tareas y mensajes entre diferentes procesadores y estudiar, sobre cada conjunto generado, los límites de utilización planificables, empleando para ello técnicas heurísticas de asignación óptima de prioridades, desarrolladas en [GUT95A].

Hemos realizado simulaciones para diferentes relaciones entre los tiempos de ejecución de peor y de mejor caso de las diferentes acciones, $\Delta=C^b/C$. Nótese que un valor de $\Delta=0$ es equivalente al análisis anterior de peor caso, en el cual se consideraban los tiempos de ejecución de mejor caso despreciables. Por otra parte, un valor de $\Delta=1$ representa un tarea con tiempo de ejecución determinista y es el caso en el que el análisis del mejor caso obtiene mejores resultados. Otro parámetro que influye significativamente en la planificabilidad del sistema es la relación entre los periodos máximo y mínimo de los eventos externos, y que identificaremos con $\delta=T_{max}/T_{min}$.

Las siguientes gráficas muestran los resultados experimentales obtenidos sobre un ejemplo consistente en 8 procesadores con 50 tareas ejecutando en ellos y 3 redes distintas de comunicación, a través de las cuales se transmiten 43 mensajes entre diferentes procesadores. El sistema responde a 7 eventos externos periódicos, cada uno de los cuales tiene un plazo de ejecución de-principio-a-fin proporcional a su respectivo periodo y que tiene en cuenta el número de recursos utilizado por cada secuencia de respuesta (con una media de

10 recursos por secuencia). Hemos elegido este ejemplo porque representa una situación relativamente compleja.

Las primeras cuatro gráficas (Figura 3-6 a Figura 3-9) representan el límite de planificabilidad obtenido cuando la utilización de recursos se incrementa desde un valor inicial pequeño hasta que el sistema se hace no planificable, sin variar las prioridades asignadas inicialmente. Este límite viene dado en términos de la utilización media de los recursos. En la Figura 3-6 podemos ver los límites absolutos medios de utilización planificable para la estimación iterativa del mejor caso, como función de Δ, la relación entre los tiempos de ejecución en el mejor y en el peor caso. Se ha representado el límite de planificabilidad para tres valores diferentes de relaciones entre los periodos, δ. Podemos ver que, tal como se esperaba, la mejora en el nivel de planificabilidad es mayor para valores altos de Δ, obteniendo mejoras superiores al 5% en varios experimentos. La mejora es ligeramente mayor para valores mayores de δ. La Figura 3-7 muestra una comparación entre los incrementos en los límites planificables de utilización para ambos métodos, trivial e iterativo, como función de Δ. El experimento se hizo para un valor de δ=100. Podemos ver que se consigue una mejora significativa para valores de Δ>0.4. La mejora es ligeramente mayor con la estimación iterativa que con la estimación trivial (por encima del 1% de incremento en la utilización). Otros experimentos para valores de δ=10 (Figura 3-8) y δ=1 (Figura 3-9) no muestran mejoras significativas de un método frente a otro.

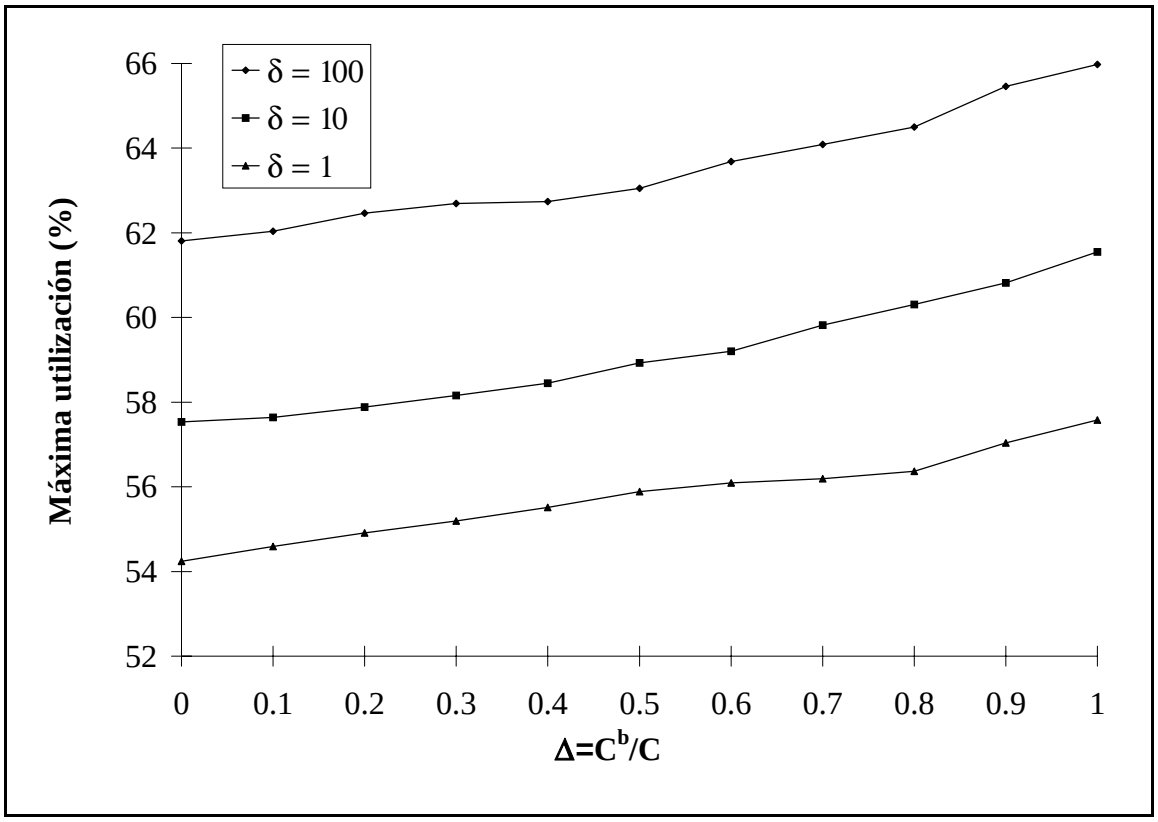


Figura 3-6. Análisis para diferentes valores de δ

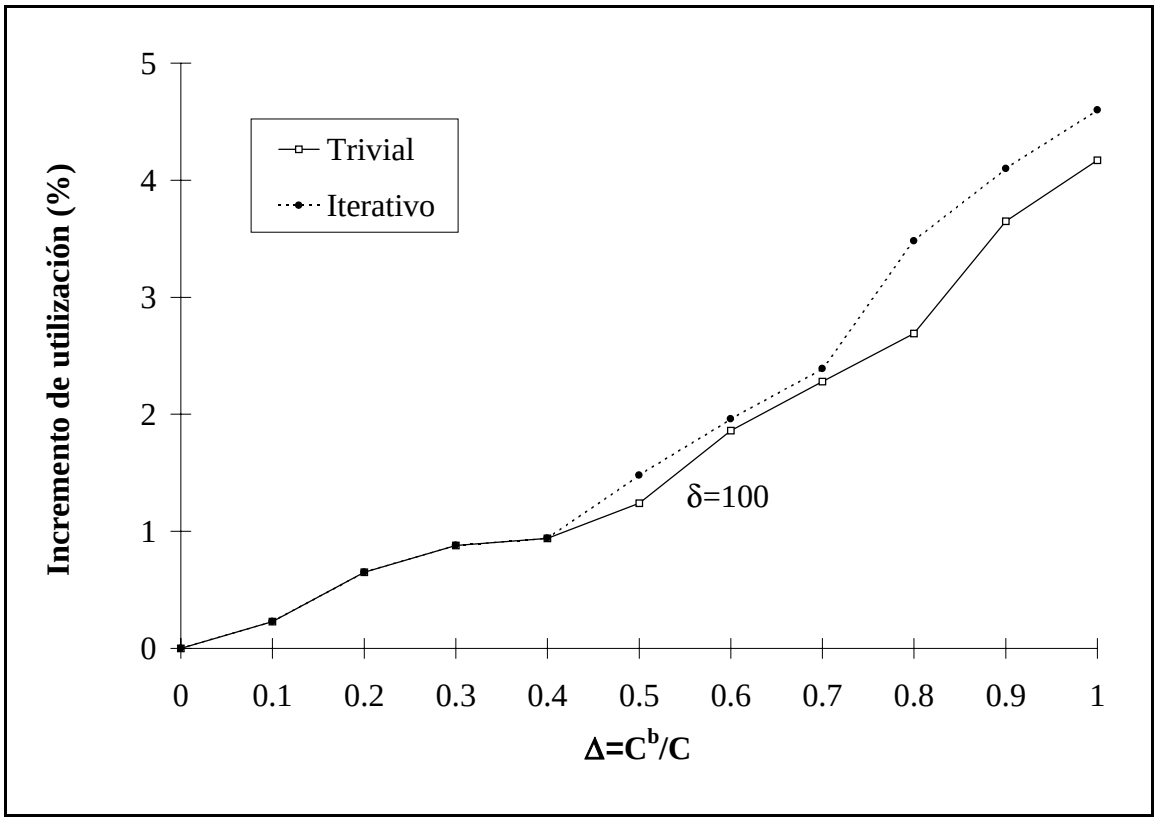


Figura 3-7. Comparación entre las dos estimaciones de mejor caso, con δ=100

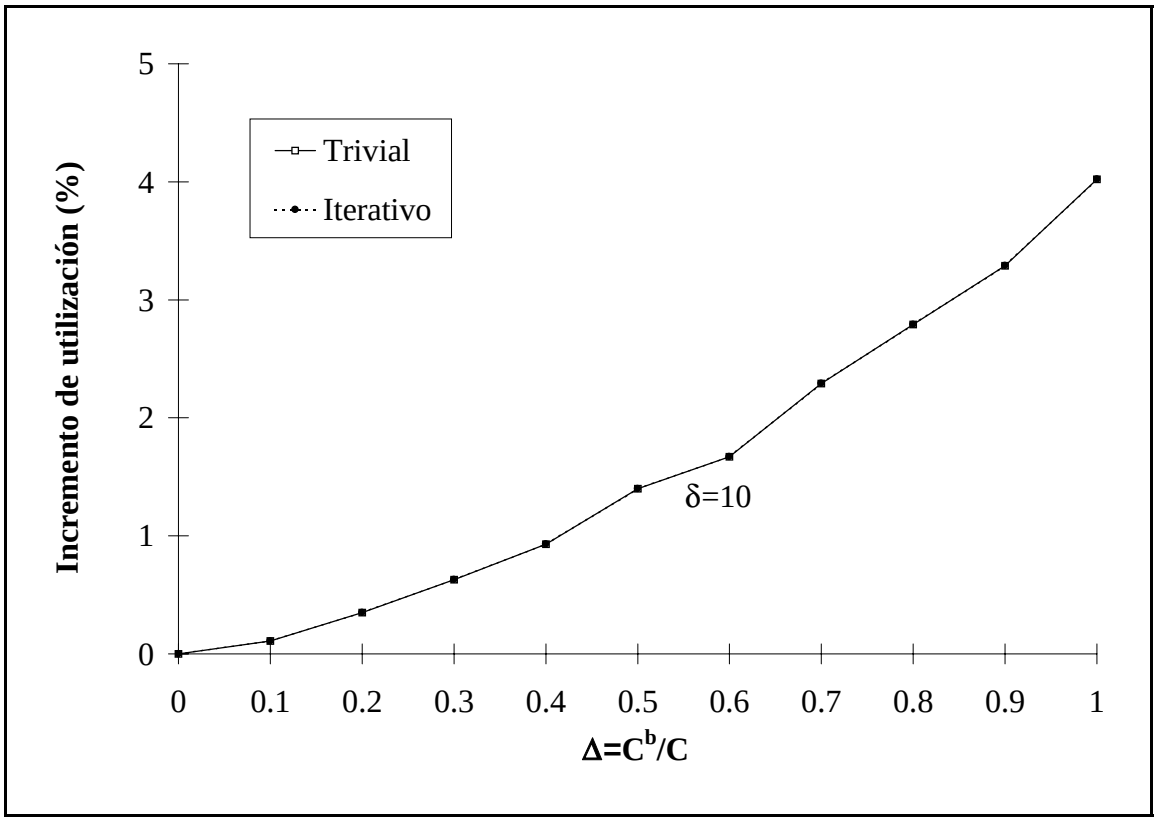


Figura 3-8. Comparación entre las dos estimaciones de mejor caso, con δ=10

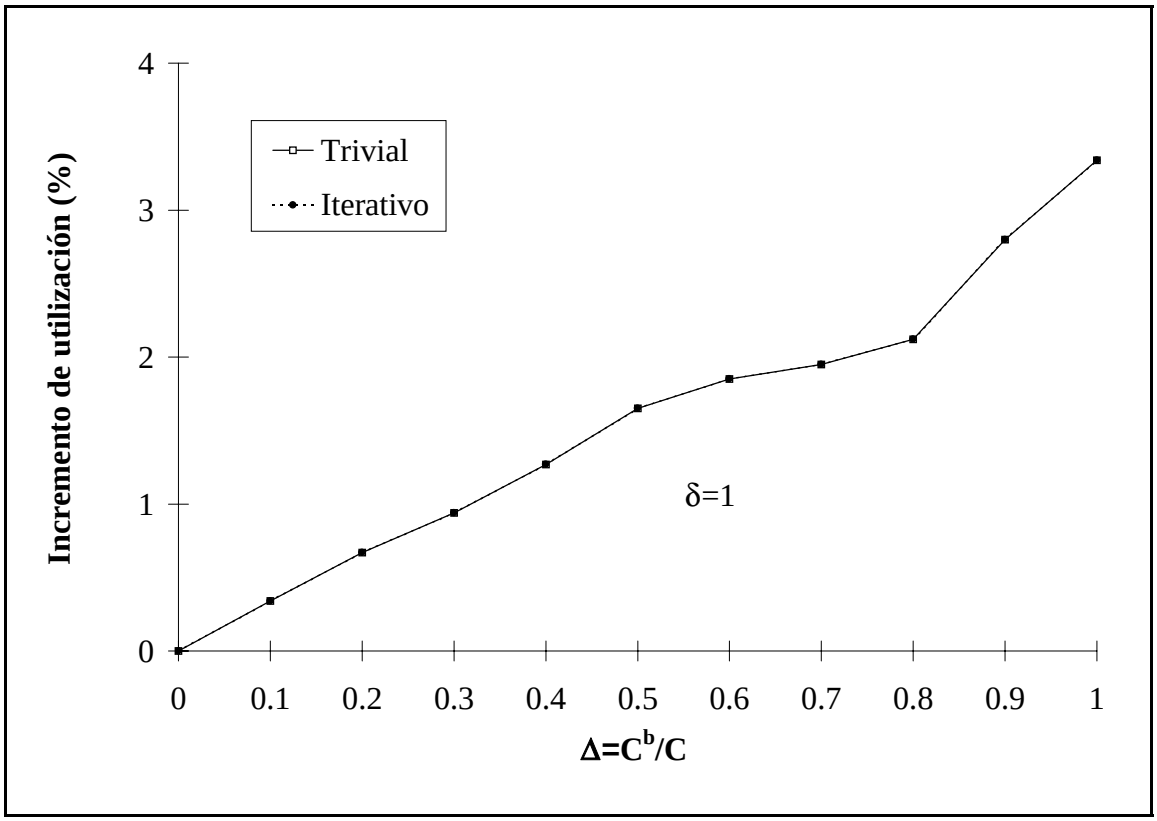


Figura 3-9. Comparación entre las dos estimaciones de mejor caso, con δ=1

Las tres últimas gráficas (Figura 3-10 a Figura 3-12) muestran resultados similares a los obtenidos en las figuras anteriores, pero utilizando la técnica de optimización de prioridades *HOPA* [GUT95A] para determinar los límites planificables de utilización. En este caso, la técnica de optimización también toma ventaja de las nuevas ecuaciones del análisis de planificabilidad para obtener mejores asignaciones de prioridad. La Figura 3-10 muestra los resultados para δ=100, la Figura 3-11 para δ=10, y la Figura 3-12 para δ=1. Podemos ver que los incrementos en la utilización planificable son mayores que cuando no se variaban las prioridades, lo cual muestra que la técnica de optimización de prioridades sale beneficiada con la utilización de las nuevas ecuaciones para el mejor caso. También podemos ver que con la optimización de prioridades disminuye la diferencia entre las dos estimaciones, respecto a cuando las prioridades no cambiaban, pero ahora aparecen ventajas también para valores menores de δ.

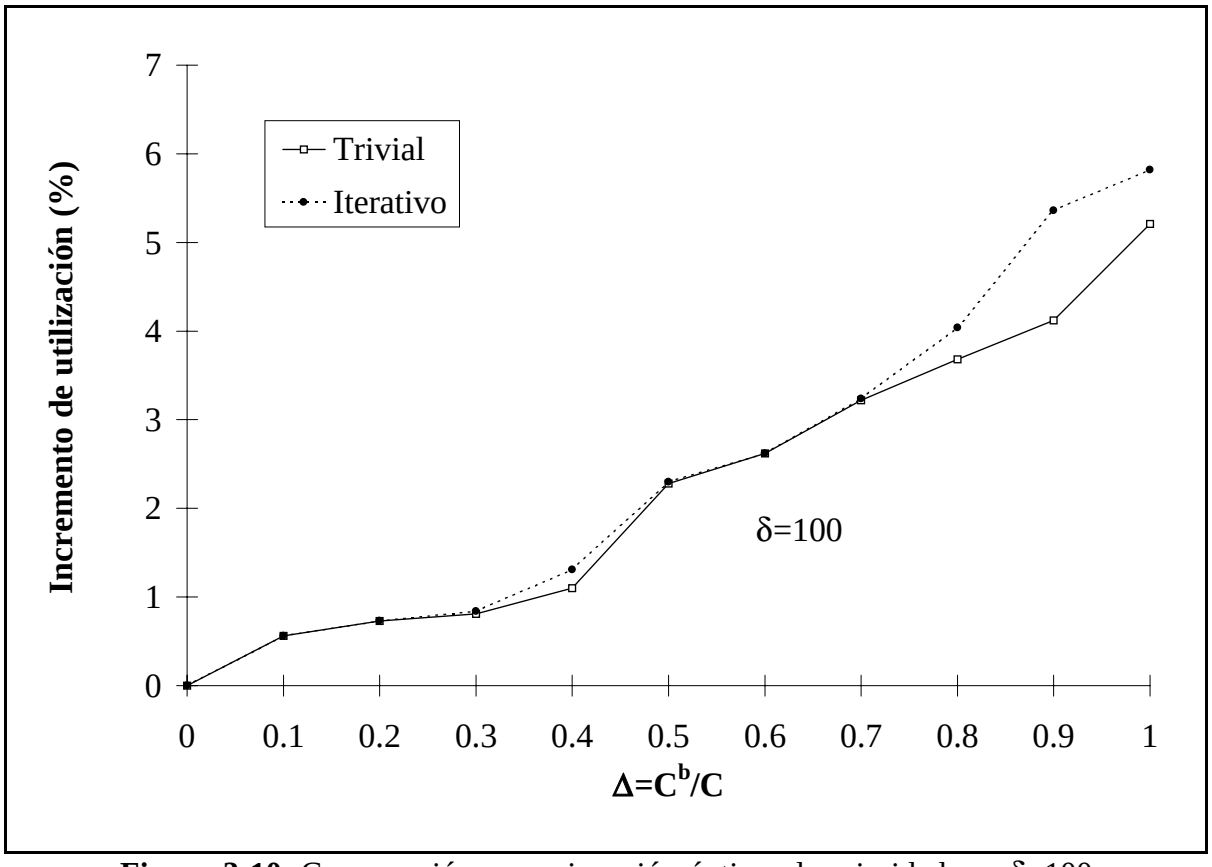


Figura 3-10. Comparación con asignación óptima de prioridades y δ=100

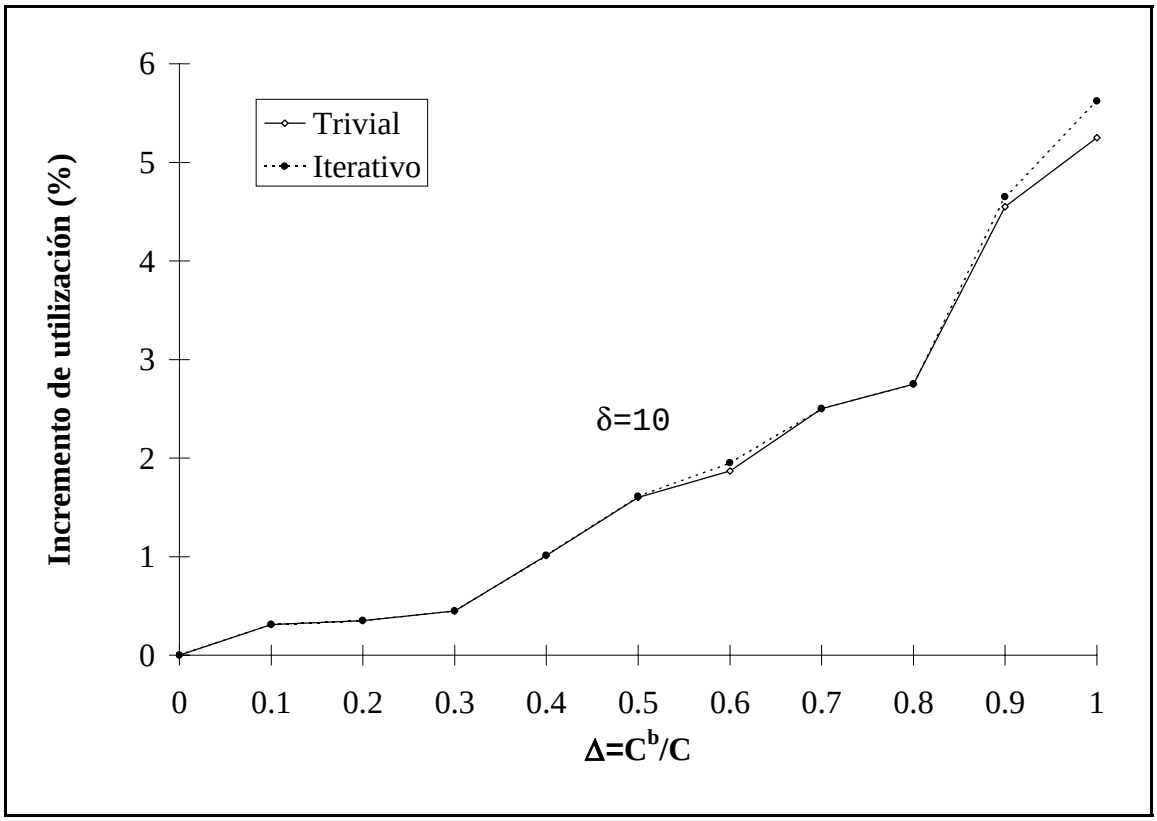


Figura 3-11. Comparación con asignación óptima de prioridades y δ=10

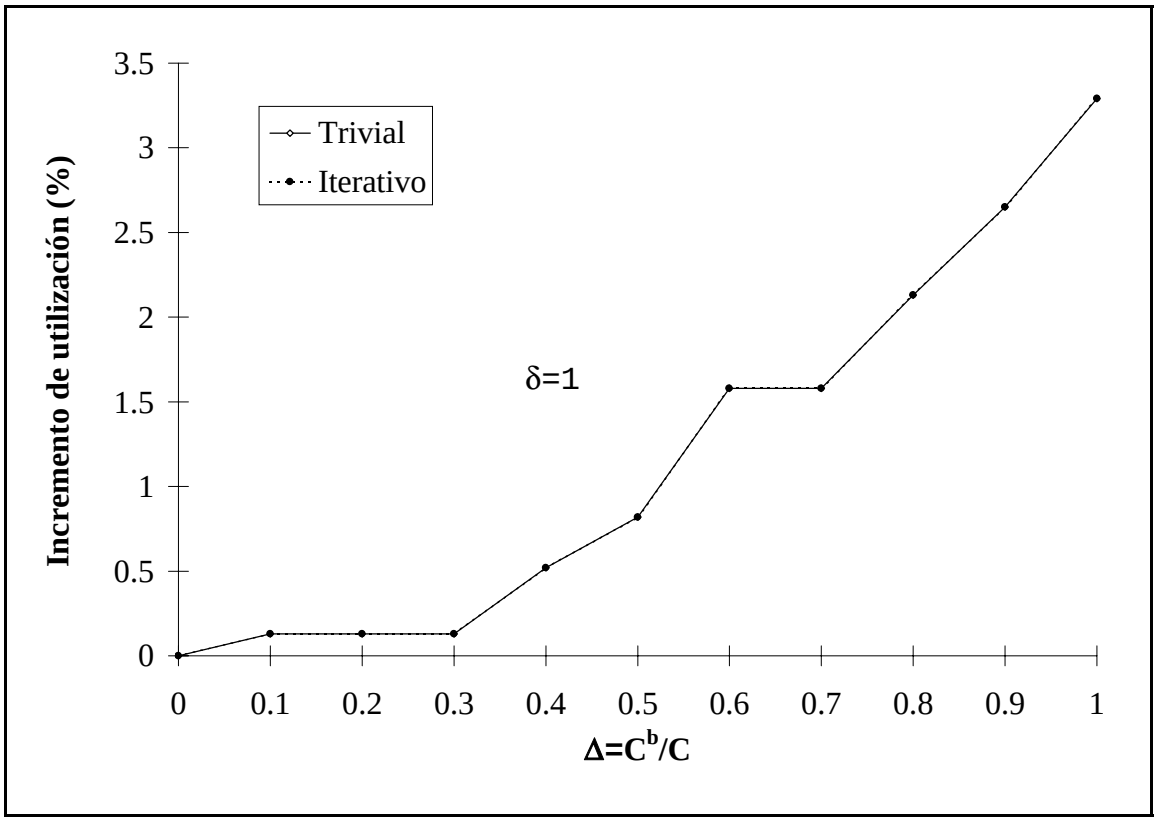


Figura 3-12. Comparación con asignación óptima de prioridades y δ=1

4. Análisis de planificabilidad para tareas con offsets estáticos y dinámicos.

4.1. Introducción

Actualmente, hay dos áreas relacionadas para las que la teoría RMA no suministra soluciones exactas de los tiempos de respuesta: sistemas distribuidos de tiempo real estricto y sistemas en los que las tareas se suspenden a sí mismas. Las técnicas de análisis para esos sistemas se basan en la suposición de que todas las tareas son independientes y conducen, por tanto, a resultados pesimistas [TIN94F] [PAL98A]. Estas técnicas pesimistas permiten garantizar los requerimientos temporales cuando el resultado del análisis así lo dice, pero pueden apuntar a pobres niveles de utilización por el pesimismo existente. Si se pudiera realizar un análisis exacto, o al menos reducir ese pesimismo, se podría hacer un uso más eficiente de la potencia de cálculo disponible en esos sistemas de tiempo real.

Tindell desarrolló en [TIN94B] una técnica para calcular los tiempos de respuesta de peor caso (o una cota superior de los mismos) para conjunto de tareas con *offsets* estáticos En dicho artículo, el sistema está compuesto de transacciones periódicas, cada una de ellas con varias tareas. Cada tarea se libera (*release*) después de transcurrido cierto tiempo (llamado *offset*) desde la llegada del evento que dispara la transacción. En [TIN94B], los *offsets* de las tareas están restringidos a ser menores que el periodo de la transacción, y además, los *offsets* son estáticos. Esto es útil en aquellos sistemas en los que la activación de las tareas se controla de forma precisa para evitar los efectos negativos de la activación retrasada, como pueden ser los sistemas planificados con la técnica de Modificación de Fase [BET92] [SUN96]. Sin embargo, una técnica para calcular los tiempos de respuesta de sistemas formados por tareas con *offsets* podría ser muy útil para obtener soluciones a los problemas

de tareas que se suspenden y sistemas distribuidos, si se permitiera que los *offsets* variaran dinámicamente, esto es, si pudieran cambiar de una activación a otra. Por ejemplo, en sistemas distribuidos una tarea se activa cuando recibe un mensaje indicando que la tarea previa ha finalizado su ejecución; este instante de activación puede variar de un periodo a otro, en función de cómo ejecuten las tareas previas. Además, en sistemas distribuidos es usual que los plazos de las tareas sean mayores que los periodos del evento externo y, por tanto, sería conveniente que los *offsets* de las tareas pudieran ser mayores que sus periodos.

Consecuentemente, en este capítulo extendemos el análisis de Tindell para tareas con *offsets* estáticos en varias direcciones. Por un lado, eliminamos la restricción de que los *offsets* sean menores que los periodos de las transacciones. Además, aunque sus resultados son correctos, existen ciertos defectos en el desarrollo de su análisis; por ejemplo, el Teorema 1 en [TIN94B] no tiene en cuenta el efecto de la activación retrasada y sobre ese teorema se basan desarrollos posteriores de su técnica. Por esas razones, en este capítulo extenderemos la técnica para permitir *offsets* mayores que los periodos de las tareas y además formalizaremos la técnica mediante un conjunto completo de demostraciones. Este desarrollo introduce una nueva notación que nos será útil cuando extendamos nuestra técnica para tener en cuenta las relaciones de precedencia entre las tareas de una transacción, y que desarrollaremos en el capítulo siguiente.

Sun y Liu desarrollaron en [SUN95] una técnica similar al análisis de Tindell con *offsets*, y lo aplicaron al análisis de sistemas multiprocesadores planificados con el protocolo de Modificación de Fase. Esta técnica, sin embargo, se restringía a *offsets* y plazos menores que los periodos y además, no tenía en cuenta el efecto de las activaciones retrasadas. Tanto el análisis de Tindell como el desarrollado en este capítulo permiten plazos superiores al periodo y contemplan las activaciones retrasadas.

Más importante es la extensión que hacemos para cubrir el caso de tareas en las que los *offsets* puedan variar dinámicamente y que aplicaremos directamente al análisis de sistemas distribuidos y sistemas con tareas que se suspenden a sí mismas. Como veremos, la aplicación de esta nueva técnica en sistemas distribuidos permite incrementar de una manera significativa las utilizaciones planificables de los procesadores, en comparación con los resultados del análisis aplicado actualmente a ese tipo de sistemas. Es de notar que estas

mejoras no significan ningún cambio en la planificación de los sistemas, que puede ser realizada todavía con prioridades fijas.

En el próximo capítulo mejoraremos el análisis al explotar las relaciones de precedencia entre las acciones de una misma secuencia de respuesta en sistemas distribuidos.

4.2. Análisis para tareas con *offsets* estáticos

4.2.1 Modelo computacional

El sistema de tiempo real que vamos a considerar para el análisis de tareas con *offsets* estáticos está compuesto por un conjunto de tareas ejecutando en un único procesador y agrupadas en entidades que llamaremos transacciones [TIN94B]. Cada transacción Γ_i es activada por una secuencia periódica de eventos externos con periodo T_i y se compone de un conjunto de m_i tareas. Las fases relativas entre los diferentes eventos externos son arbitrarias. Cada tarea se activa cuando ha transcurrido un intervalo de tiempo (llamado *offset*) desde la llegada del evento externo. En esta sección supondremos que los *offsets* son estáticos, de forma que no cambian de una activación a otra.

La Figura 4-1 muestra un ejemplo de tal sistema, donde el eje horizontal representa el tiempo. Las flechas descendentes representan la llegada de un evento externo asociado a una transacción, mientras que las flechas ascendentes representan la activación de las tareas y los rectángulos sombreados la ejecución de las mismas. Supondremos que cada tarea tiene una prioridad única y que el conjunto de tareas se planifica mediante un planificador expulsor basado en prioridades fijas. Nótese que no hay relaciones de precedencia; cada tarea se activa en un instante igual a la llegada del evento externo más el *offset*, y ejecuta al nivel de prioridad que le fue asignada, independientemente de que las tareas de su misma transacción y menor *offset* hayan finalizado o no.

Identificaremos cada tarea mediante dos subíndices: el primero identifica la transacción a la que pertenece, y el segundo la posición que ocupa dentro del conjunto de tareas de su transacción, al ordenarlas en orden creciente de *offset*. De esta forma, con τ_{ij} denotaremos la *j*-ésima tarea perteneciente a la transacción Γ_i, con una fase de activación Φ_{ij} y un tiempo de

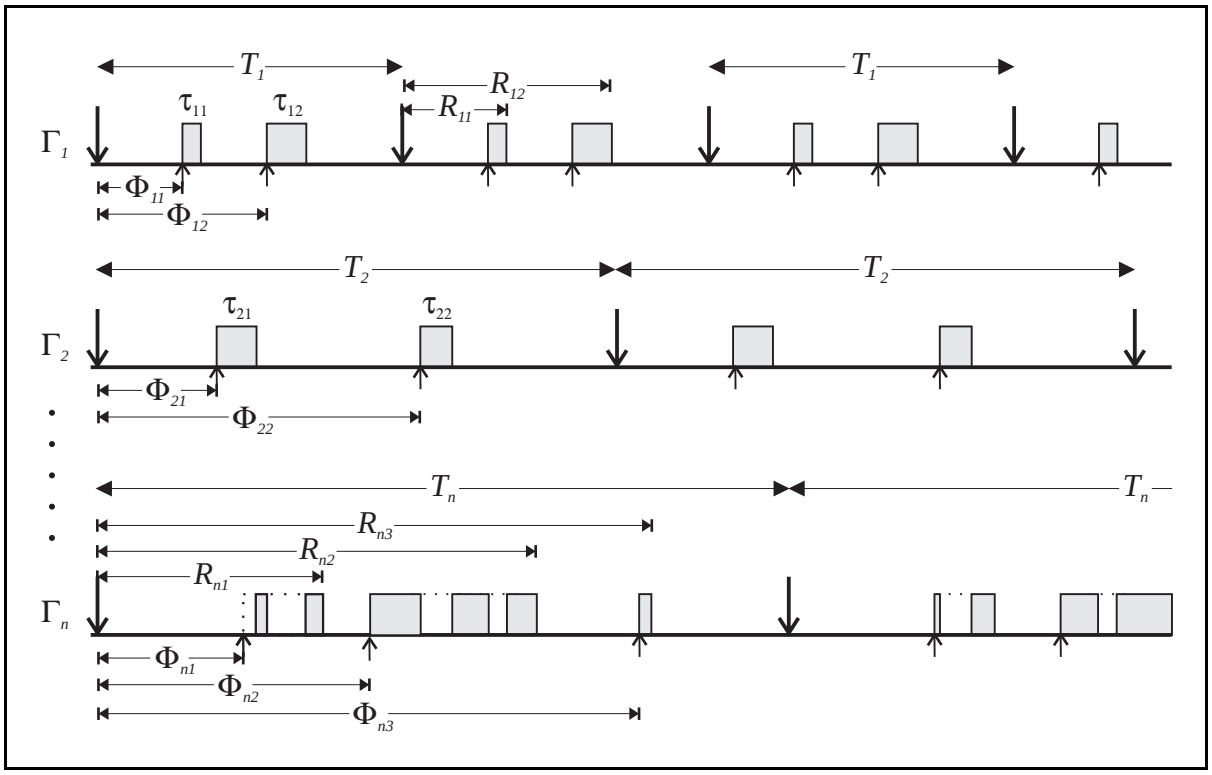


Figura 4-1. Modelo computacional de un sistema de transacciones con *offsets* estáticos

ejecución de peor caso C_{ij}. Permitiremos además que cada tarea pueda sufrir un retraso máximo denotado por J_{ij}. Esto quiere decir que la activación de la tarea puede producirse en cualquier instante dentro del intervalo determinado por los instantes $t_0 + \Phi_{ij}$ y $t_0 + \Phi_{ij} + J_{ij}$, donde t_0 es el instante en que llegó el evento externo.

Permitiremos plazos superiores al periodo, de forma que en un mismo instante puede haber varias instancias de una misma tarea pendientes de ejecución. Permitiremos también que tanto el *offset* Φ_{ij} como el retraso máximo J_{ij} sean mayores que el periodo de su transacción, T_i. Para cada tarea τ_{ij} definiremos su tiempo de respuesta global como la diferencia entre su tiempo de finalización y el instante en que llegó el evento externo asociado. El tiempo de respuesta de peor caso será R_{ij}. Cada tarea puede llevar asociado un plazo global de ejecución, D_{ij}, que es también relativo a la llegada del evento externo. Por tanto, para chequear si el sistema cumplirá sus plazos debemos comprobar si los tiempos de respuesta de peor caso son menores o iguales que los plazos.

Supondremos también que las tareas se sincronizan para utilizar recursos comunes de acceso mutuamente exclusivo. El efecto de ese uso por tareas de menor prioridad, en el análisis de una tarea τ_{ab} está limitado por un término de bloqueo máximo B_{ab}.

4.2.2 Análisis exacto de los tiempos de respuesta

En este apartado obtendremos las expresiones analíticas que nos permitirán calcular los tiempos de respuesta de peor caso de las tareas, y formalizaremos su desarrollo. Aunque el análisis exacto resulta intratable para sistemas grandes, debido al gran número de casos que hay que considerar, nos servirá como base para el desarrollo de una aproximación que obtiene estimaciones superiores de esos tiempos de respuesta que, aunque inexactas, son bastante aproximadas y, sobre todo, hacen tratable el análisis para sistemas con gran número de tareas.

Cuando construyamos el escenario de peor caso para el análisis de una tarea τ_{ab} debemos tener en cuenta que el instante crítico puede no incluir la activación simultanea de todas las tareas de prioridad superior, tal como ocurriría si todas las tareas fueran independientes. La existencia de *offsets* hace imposible la activación simultanea de tareas pertenecientes a la misma transacción.

Cuando se analiza una tarea determinada podemos modificar el *offset* de una tarea de prioridad superior, sumando o restando periodos completos de esa misma tarea, sin que tenga ningún efecto sobre el tiempo de respuesta de la tarea con menor prioridad, puesto que las activaciones pertenecientes a una misma tarea son indistingibles entre sí. Por tanto, a efectos de simplificar el análisis, consideraremos para cada tarea un *offset* reducido, ϕ_{ij}, que es siempre un valor positivo menor que el periodo y que se calcula de la siguiente forma:

$$\phi_{ij} = \Phi_{ij} \bmod T_i \tag{1}$$

donde la función *mod* es la operación módulo definida como:

$$A \bmod B = A - \left\lfloor \frac{A}{B} \right\rfloor B \tag{2}$$

y $\lfloor x \rfloor$ el mayor número entero que es menor o igual que x.

Este resultado se mantiene para todas las tareas de prioridad superior, incluso para las pertenecientes a la misma transacción que la tarea analizada, ya que en el modelo que estamos considerando no existen relaciones de precedencia.

Para derivar la técnica de análisis obtendremos primero la contribución de cada tarea al tiempo de respuesta de peor caso, suponiendo que conocemos cuándo se produce el instante crítico. Posteriormente veremos como calcular ese instante crítico. Centremos nuestra atención en el patrón de activaciones de una tarea τ_{ij}, cuando el desfase entre el instante crítico y la transacción Γ_i es ϕ. Este desfase se mide entre el instante crítico y la última activación de la transacción Γ_i producida antes del instante crítico. Nótese que $0 \leq \phi < T_i$.

A efectos de calcular la contribución de peor caso de la tarea τ_{ij} al tiempo de respuesta de tareas de menor prioridad, podemos clasificar las activaciones de la tarea en los siguientes conjuntos:

- *Conjunto 0*: Activaciones ocurridas antes del instante crítico y que no podrían ocurrir en el instante crítico aunque se produjeran con el máximo retraso posible, J_{ij}.

- *Conjunto 1*: Activaciones que se producen en el instante crítico o que se pudieran retrasar hasta el instante critico

- *Conjunto 2*: Activaciones producidas después del instante crítico.

La Figura 4-2 muestra dos posibles escenarios para el desfase entre el instante crítico y las activaciones de la transacción Γ_i. El escenario 1, en la parte superior de la figura, corresponde al caso de que el desfase sea mayor que la fase de activación de la tarea τ_{ij}, es decir, $\phi \geq \phi_{ij}$. El escenario 2 de la parte inferior corresponde al caso $\phi < \phi_{ij}$. Las líneas punteadas representan el retraso sufrido por cada activación de la tarea. El instante t_0 corresponde, en ambos escenarios, a la primera activación de la transacción Γ_i cuya tarea τ_{ij} se puede retrasar hasta el instante crítico, t_c (las activaciones producidas antes de t_0 necesitarían sufrir un retraso mayor que el máximo permitido J_{ij} para que pudieran ocurrir en el instante t_c). La activación de τ_{ij} correspondiente al evento producido en t_1 también puede ser retrasada de forma que coincida con el instante crítico. Sin embargo, la activación de τ_{ij}

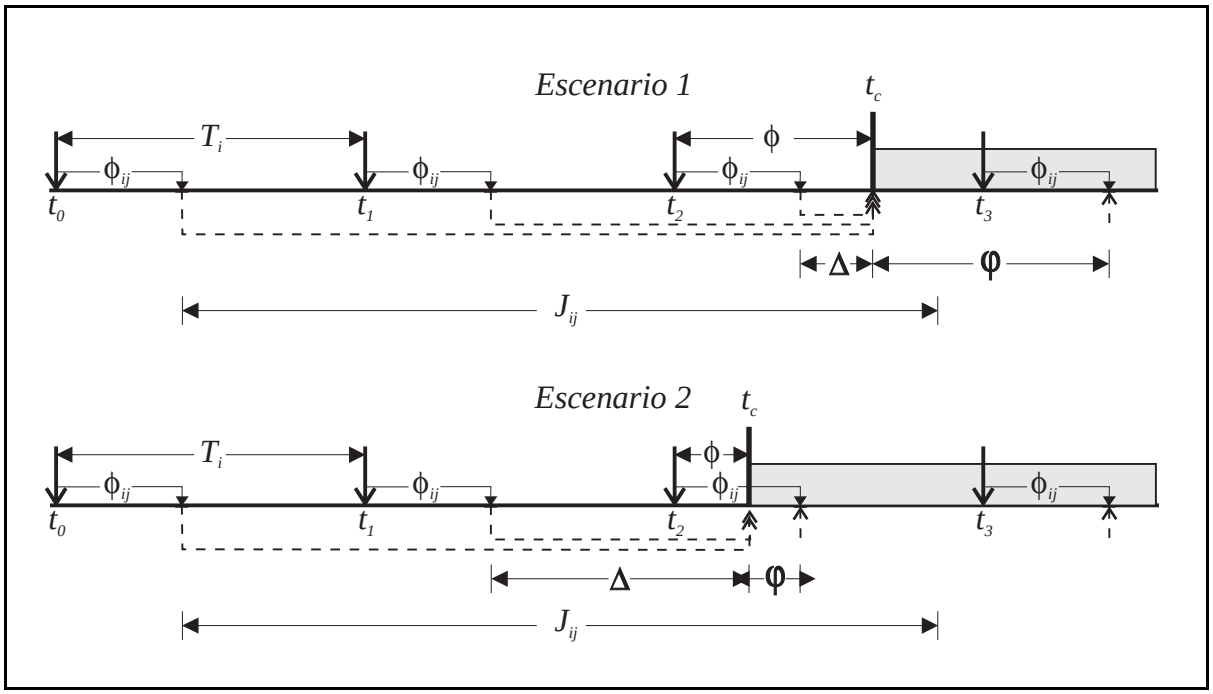


Figura 4-2. Escenarios para calcular la contribución de la tarea τ_{ij} al tiempo de respuesta de tareas de menor prioridad

asociada al evento producido en t_2, puede ser retrasada hasta el instante crítico en el escenario 1, pero no en el escenario 2, puesto que el *offset* ϕ_{ij} es mayor que la fase relativa ϕ entre la llegada del evento y el instante crítico, de forma que esa activación se producirá en cualquier caso después del instante crítico. En el escenario 1, las activaciones de la tarea τ_{ij} correspondientes a eventos producidos en instantes anteriores a t_0 pertenecen al *Conjunto 0*, las correspondientes a los instantes t_0, t_1 y t_2 pertenecen al *Conjunto 1* y las de los eventos posteriores a t_2 al *Conjunto 2*. Para el escenario 2, la activación asociada con el evento que llega en el instante t_2 deberá ser incluida en el *Conjunto 2*.

Una vez clasificadas las activaciones de la tarea τ_{ij} en los tres conjuntos anteriores, el cálculo de los retrasos que conducen a la contribución de peor caso de τ_{ij} sobre la respuesta de tareas de menor prioridad se realiza de acuerdo con el siguiente teorema:

Teorema 4-1. Sea t_c un instante crítico para la ejecución de una tarea τ_{ab} y ϕ el desfase entre el patrón de activaciones de la transacción Γ_i y el instante critico. La contribución de peor caso de la tarea τ_{ij} al tiempo de respuesta de τ_{ab} ocurre cuando las activaciones de τ_{ij} pertenecientes al *Conjunto 1* sufren un retraso tal que coinciden con el instante crítico, y las activaciones pertenecientes al *Conjunto 2* ocurren con retraso nulo.

Demostración: Según la definición del periodo de ocupación, las activaciones del *Conjunto 0* no pueden afectar a su ejecución, ya que si así fuera, el periodo de ocupación podría comenzar antes y, por tanto, el instante t_c elegido no sería crítico.

Para las activaciones del *Conjunto 1*, debemos considerar los términos de retraso que causen que cada activación ocurra dentro del periodo de ocupación. Pero si el retraso hace que esa activación se produzca después del instante crítico, podría ocurrir fuera del periodo de ocupación. Por ello, la contribución máxima en el periodo de ocupación se asegurará si cada activación se produce en el instante crítico.

Para las activaciones del *Conjunto 2* debemos tener en cuenta que ocurren después del instante crítico, de forma que cuanto más se retrasen más probable es que ocurran fuera del periodo de ocupación. Por tanto, la mayor contribución de estas activaciones se conseguirá cuando las activaciones se produzcan sin retraso.□

Bajo las condiciones del Teorema 4-1, calcularemos el número de activaciones de la tarea τ_{ij} incluidas en el *Conjunto 1* y que pueden acumularse en el instante crítico. Llamaremos a ese número n_{ij} (en el ejemplo, corresponde a n_{ij}=3 en el escenario 1 y a n_{ij}=2 en el escenario 2). Para calcular n_{ij} definiremos un nueva magnitud, Δ, como la diferencia temporal existente entre el instante crítico y el instante en que se hubiera producido la última activación del *Conjunto 1* si no hubiera sufrido ningún retraso. En el ejemplo de la Figura 4-2, $\Delta = t_c - t_2 + \phi_{ij}$ para el escenario 1, y $\Delta = t_c - t_1 + \phi_{ij}$ para el escenario 2. Se puede ver que:

$$\Delta = \begin{cases} \phi - \phi_{ij} & si\ \phi \geq \phi_{ij} \\ T_i + \phi - \phi_{ij} & si\ \phi < \phi_{ij} \end{cases} \quad (3)$$

o, de forma equivalente:

$$\Delta = (\phi - \phi_{ij}) \bmod T_i \quad (4)$$

La primera activación de la tarea τ_{ij} en el *Conjunto 1* corresponde al evento que llegó en el instante t_0, puesto que es la primera activación que puede ocurrir en el instante crítico o retrasarse hasta él. Por tanto, es la única activación de la tarea τ_{ij} que verifica simultáneamente las expresiones:

$$t_0 + \phi_{ij} + J_{ij} \geq t_c \quad (5)$$

y:

$$t_0 - T_i + \phi_{ij} + J_{ij} < t_c \tag{6}$$

Si observamos la Figura 4-2, podemos ver que:

$$t_c = t_0 + (n_{ij}-1)T_i + \phi_{ij} + \Delta \tag{7}$$

y, al sustituir en las dos expresiones anteriores, resulta:

$$\begin{aligned} t_0 + \phi_{ij} + J_{ij} &\geq t_0 + (n_{ij}-1)T_i + \phi_{ij} + \Delta \\ t_0 - T_i + \phi_{ij} + J_{ij} &< t_0 + (n_{ij}-1)T_i + \phi_{ij} + \Delta \end{aligned} \tag{8}$$

de donde obtenemos:

$$n_{ij}-1 \leq \frac{J_{ij}-\Delta}{T_i} \qquad y \qquad n_{ij}-1 > \frac{J_{ij}-\Delta}{T_i} - 1 \tag{9}$$

Dado que n_{ij} es un numero entero, las solución de las dos expresiones anteriores es:

$$n_{ij} - 1 = \left\lfloor \frac{J_{ij}-\Delta}{T_i} \right\rfloor \tag{10}$$

o, lo que es lo mismo:

$$n_{ij} = \left\lfloor \frac{J_{ij}-\Delta}{T_i} \right\rfloor + 1 \tag{11}$$

Para determinar los efectos de las activaciones de la tarea τ_{ij} pertenecientes al *Conjunto 2* necesitamos conocer el instante en que se produce la primera de esas activaciones, ya que las siguientes ocurrirán a intervalos periódicos después de ella. El intervalo temporal transcurrido entre el instante crítico y esa primera activación del *Conjunto 2* la llamaremos φ (ver Figura 4-2). Según la definición de Δ tenemos:

$$\varphi = T_i - \Delta \tag{12}$$

Podríamos haber utilizado φ en la ecuación (11), obteniendo:

$$n_{ij} = \left\lfloor \frac{J_{ij}+\varphi}{T_i} \right\rfloor \tag{13}$$

De acuerdo con el Teorema 4-1, la contribución de peor caso de la tarea τ_{ij} al periodo de ocupación de una tarea de menor prioridad es equivalente a n_{ij} activaciones en el instante crítico más una secuencia de activaciones periódicas iniciadas φ unidades de tiempo después

del instante critico. Sin pérdida de generalidad, consideraremos origen de tiempos en el instante crítico. La máxima contribución de la tarea τ_{ij} al tiempo de respuesta de τ_{ab} desde el instante crítico hasta un instante *t* se determina, entonces, a partir de la expresión:

$$W(\tau_{ij},\phi,t) = n_{ij}(\phi)\ C_{ij} + \left\lceil \frac{t-\varphi(\phi)}{T_i} \right\rceil C_{ij} = \left(\left\lfloor \frac{J_{ij}+\varphi(\phi)}{T_i} \right\rfloor + \left\lceil \frac{t-\varphi(\phi)}{T_i} \right\rceil \right) C_{ij} \qquad (14)$$

con

$$\varphi(\phi) = T_i - (\phi - \phi_{ij}) \bmod T_i \qquad (15)$$

La interferencia total de tareas pertenecientes a una transacción Γ_i sobre la ejecución de una tarea τ_{ab} cualquiera se calcula teniendo en cuenta las contribuciones debidas a todas las tareas de la transacción que puedan interferir en la ejecución de τ_{ab}:

$$W(\Gamma_i,\phi,t) = \sum_{\forall j \in hp_i(\tau_{ab})} W(\tau_{ij},\phi,t) \qquad (16)$$

donde,

$hp_i(\tau_{ab})$ es el conjunto de tareas pertenecientes a la transacción Γ_i cuya prioridad sea mayor o igual que la prioridad de la tarea τ_{ab}.

Ahora debemos determinar como calcular ϕ, la fase entre el patrón de activaciones de la transacción Γ_i y el instante crítico. Basaremos ese cálculo en el siguiente teorema:

Teorema 4-2. La máxima interferencia de una transacción Γ_i sobre la ejecución de una tarea τ_{ab} se obtiene cuando la primera activación, dentro del periodo de ocupación, de alguna tarea τ_{ik} en el conjunto $hp_i(\tau_{ab})$ coincide con el instante crítico después de haber experimentado el máximo retraso permitido, J_{ik}.

Demostración: Por definición del periodo de ocupación, justo antes del instante crítico no existe ninguna tarea pendiente de ejecución con mayor prioridad que τ_{ab}. Supongamos ahora que elegimos un instante crítico que no coincida con la activación de alguna tarea de $hp_i(\tau_{ab})$. Fijémonos entonces en la primera activación que se produzca dentro del periodo de ocupación de una tarea perteneciente al conjunto $hp_i(\tau_{ab})$ y sea τ_{ik} esa tarea. Si forzamos la llegada de los eventos de Γ_i a que ocurran más pronto pero manteniendo el mismo patrón de activaciones de todas sus tareas, de forma que esa primera activación de la tarea τ_{ik} coincida con el instante crítico, conseguimos que todas

las activaciones pertenecientes a tareas de $hp_i(\tau_{ab})$ que se producían dentro del periodo de ocupación continúen en él y además favorecemos el que activaciones que se producían después del periodo de ocupación puedan entrar en él, incrementando con ello las interferencias sobre la tarea τ_{ab}. Por tanto, haciendo que la primera activación de τ_{ik} coincida con el instante crítico hacemos que su interferencia sea mayor.

Ahora debemos probar que la contribución de peor caso de la transacción Γ_i se obtiene cuando la activación de la tarea τ_{ik} que coincide con el instante crítico haya sufrido el máximo retraso, igual a J_{ik}. Llamemos I al conjunto de activaciones de tareas de $hp_i(\tau_{ab})$ que inicien el periodo de ocupación coincidiendo con el instante critico, y supongamos que cada una de ellas ha sufrido un retraso de valor j_{il} menor que el máximo permitido J_{il}. Supongamos ahora que movemos hacia atrás (esto es, adelantamos en el tiempo) la llegada de eventos de la transacción Γ_i, y simultáneamente incrementamos en la misma cantidad el retraso de todas las activaciones de I, de forma que todas esa activaciones se siguen produciendo en el mismo instante que antes, mientras dejamos invariables los retrasos del resto de activaciones (no pertenecientes al conjunto I) de forma que esas activaciones se producen más pronto que antes. Bajo esas condiciones, podremos adelantar la llegada de eventos hasta que se produzca alguna de las siguientes situaciones: a) una de las activaciones de I alcanza su máximo retraso, o b) una de las activaciones no pertenecientes al conjunto I alcanza el instante crítico. En el caso b), podemos añadir esa activación al conjunto I y continuar el proceso adelantando la llegada de eventos de forma iterativa hasta que se alcancemos la condición a), bajo la cuál una o más activaciones coincidentes con el instante crítico experimenta su máximo retraso. Nótese que durante este proceso, ninguna de las activaciones que pertenecían al periodo de ocupación original se ha adelantado al instante crítico y, por tanto, todas las activaciones del periodo de ocupación original permanecen en él. Adicionalmente, el adelantamiento de los eventos de Γ_i ha podido provocar que otras activaciones que previamente ocurrían después de finalizado el periodo de ocupación puedan ocurrir ahora dentro de él, incrementando el tiempo de respuesta para la tarea τ_{ab} bajo análisis. Con esto, el teorema queda demostrado.□

Aplicando el Teorema 4-2, y suponiendo que conocemos la tarea τ_{ik} que origina el instante crítico, podemos determinar el desfase entre la activación de la transacción y el instante crítico, en la forma:

$$\phi = (\phi_{ik} + J_{ik}) \bmod T_i \tag{17}$$

Sustituyendo esta expresión en la ecuación (15) obtenemos la fase φ_{ijk} de una tarea τ_{ij} cuando el instante crítico se crea con la tarea τ_{ik}:

$$\varphi_{ijk} = \varphi(\phi)\Big|_{\phi=(\phi_{ik}+J_{ik}) \bmod T_i} = T_i - \left((\phi_{ik} + J_{ik}) \bmod T_i - \phi_{ij} \right) \bmod T_i \tag{18}$$

la cual, si aplicamos las propiedades de la función módulo conduce a

$$\varphi_{ijk} = T_i - \left(\phi_{ik} + J_{ik} - \phi_{ij}\right) \bmod T_i \tag{19}$$

Utilizando este valor, ya podemos obtener la expresión que nos da la contribución al peor caso de la transacción Γ_i cuando el instante crítico se inicia con τ_{ik}. Llamaremos a esta función $W_{ik}(t,\tau_{ab})$, obtenida al sustituir (19) en las ecuaciones (14), (15) y (16)

$$W_{ik}(\tau_{ab},t) = W(\Gamma_i,\phi,t)\Big|_{\phi = (\phi_{ik}+J_{ik}) \bmod T_i} = \sum_{\forall j \in hp_i(\tau_{ab})} \left(\left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + \left\lceil \frac{t - \varphi_{ijk}}{T_i} \right\rceil \right) C_{ij} \tag{20}$$

El cálculo del tiempo de respuesta de peor caso de una tarea τ_{ab} requiere aplicar la función anterior a todas las transacciones del sistema. El problema principal que se nos presenta ahora es que necesitamos encontrar, para cada transacción Γ_i, la tarea τ_{ik} con la que construir el instante crítico. Si queremos obtener un análisis exacto para la tarea τ_{ab} necesitamos chequear todas las posibles variaciones en la tarea elegida para cada transacción y elegir, de entre todas ellas, la variación que origine el mayor tiempo de respuesta para la tarea bajo análisis.

El número de variaciones, y por tanto el número de posibles instantes críticos diferentes que necesitamos chequear, viene determinado por el número de tareas con prioridad mayor o igual que la tarea analizada que existan en cada transacción del sistema. Hay que tener en cuenta también que la propia tarea bajo análisis puede originar el instante crítico para su transacción, por lo que debemos considerarla en las variaciones. El número total de variaciones es, pues:

$$N_v(\tau_{ab}) = \left(N_a(\tau_{ab}) + 1\right) \cdot N_1(\tau_{ab}) \cdot N_2(\tau_{ab}) \ldots = \left(N_a(\tau_{ab}) + 1\right) \cdot \prod_{\forall i \neq a} N_i(\tau_{ab}) \tag{21}$$

donde N_i (τ_{ab}) es el número de tareas perteneciente al conjunto $hp_i(\tau_a)$

Cada una de las $N_v(\tau_{ab})$ variaciones se caracteriza por una tupla v de índices, una por cada transacción. Cada índice $v(i)$ identifica la tarea perteneciente a la transacción Γ_i que inicia el instante crítico.

Por conveniencia, identificaremos las activaciones de la tarea analizada con números consecutivos, ordenados según el instante en que se hubieran producido en caso de no haber experimentado retraso. Además, asignaremos el valor p=1 a la activación de τ_{ab} ocurrida en el intervalo $(0,T_a]$. Esto significa que la activación ocurrida en $(T_a,2T_a]$ le asignaremos el valor p=2, etcétera. Asimismo, la activación que se hubiera producido en el intervalo $(-T_a,0]$ pero que se retrasó hasta el instante crítico le corresponde el valor p=0, la del intervalo $(-2T_a,-T_a]$ el valor p=-1, etcétera. Nótese que las activaciones ocurridas después del instante crítico se identifican con números positivos, mientras las ocurridas antes del instante crítico se identifican con valores de $p \leq 0$.

En cada variación v calcularemos el tiempo de finalización correspondiente a cada una de las activaciones de la tarea τ_{ab}. Este tiempo $w_{ab}^{v}(p)$ se obtiene considerando la ejecución de τ_{ab} junto con la interferencia debida al resto de tareas del sistema:

$$w_{ab}^{v}(p) = B_{ab} + (p - p_{0,ab}^{v} + 1)\ C_{ab} + \sum_{\forall i} W_{i\,v(i)}\left(\tau_{ab}, w_{ab}^{v}(p)\right) \qquad (22)$$

donde $p_{0,ab}^{v}$ corresponde a la numeración de la primera activación de τ_{ab} considerada en el periodo de ocupación y calculada como:

$$p_{0,ab}^{v} = -\left\lfloor \frac{J_{ab} + \varphi_{ab\,v(a)}}{T_a} \right\rfloor + 1 \qquad (23)$$

La solución de la ecuación (22) se obtiene de forma iterativa comenzando con un valor $w_{ab}^{v}(p)$=0. El tiempo de respuesta global se obtiene restando, del tiempo de finalización, el instante en que se produjo la llegada del evento que activó la transacción. Según el esquema de numeración de las activaciones, a la primera activación de la tarea τ_{ab} producida después del instante crítico le corresponde un valor p=1. Esa primera activación se produce, por definición, en el instante $\varphi_{abv(a)}$ y por tanto, la activación p-ésima de la tarea ocurre en el instante $\varphi_{abv(a)} + (p\text{-}1)T_a$. Como la tarea se activa Φ_{ab} unidades de tiempo después de la llegada del evento, el instante de activación de la transacción correspondiente es $\varphi_{abv(a)} + (p\text{-}1)T_a - \Phi_{ab}$.

De esta forma, el tiempo de respuesta global de peor caso de la activación *p*-ésima se calcula como:

$$R_{ab}^{v}(p) = w_{ab}^{v}(p) - \varphi_{ab\,v(a)} - (p-1)T_a + \Phi_{ab} \tag{24}$$

Nótese que en la ecuación anterior hemos usado el término de retraso real, Φ_{ab}, en lugar del término de retraso reducido, ϕ_{ab}, utilizado para calcular la interferencia sobre la ejecución de la tarea bajo análisis por parte de tareas de mayor prioridad.

Este análisis debe repetirse para todas las activaciones ocurridas dentro del periodo de ocupación. La longitud de este periodo de ocupación, que llamaremos L_{ab}^{v}, se puede calcular a partir de la expresión siguiente:

$$L_{ab}^{v} = B_{ab} + \left(\left\lceil \frac{L_{ab}^{v} - \varphi_{ab\,v(a)}}{T_a} \right\rceil - p_{0,ab}^{v} + 1 \right) C_{ab} + \sum_{\forall i} W_{i\,v(i)}\left(\tau_{ab}, L_{ab}^{v}\right) \tag{25}$$

que representa el primer instante después del instante crítico en el que se completan todas las activaciones de τ_{ab} y de todas las tareas de prioridad superior o igual que τ_{ab}. Con la longitud del periodo de ocupación podemos obtener el máximo valor de *p* que deberemos chequear:

$$p_{L,ab}^{v} = \left\lceil \frac{L_{ab}^{v} - \varphi_{ab\,v(a)}}{T_a} \right\rceil \tag{26}$$

También podemos calcular el índice de la última activación a chequear, $p_{L,ab}^{v}$, como el primer valor de $p \geq p_{0,ab}^{v}$ para el que su tiempo de respuesta verifica la siguiente relación:

$$R_{ab}^{v}(p) \leq T_a + \Phi_{ab} \tag{27}$$

si sustituimos (24) en esta expresión y operamos, llegamos a:

$$w_{ab}^{v}(p) \leq pT_a + \varphi_{ab\,v(a)} \tag{28}$$

que es una condición de finalización similar a la utilizada por Tindell y Clark en el algoritmo TRPC (ver capítulo 2). Sin embargo, aquí no podemos reducir el término de la derecha a pT_a (cuya validez demostramos en [PAL97] y que aparece en el capítulo 3 de esta tesis), puesto que en tareas con *offsets* no se verifica la relación en la que se basaba el Teorema 4-1, $w_{ab}^{v}(p+p') \leq w_{ab}^{v}(p) + w_{ab}^{v}(p')$.

Para calcular el tiempo de respuesta de peor caso para la tarea τ_{ab} debemos determinar el máximo de todos los posibles instantes críticos examinados, considerando en cada caso el total de activaciones necesarias. Es decir:

$$R_{ab} = \max_{\forall v} \left[\max_{p=p^v_{0,ab}..p^v_{L,ab}} \left(R^v_{ab}(p) \right) \right] = \max_{\forall v} \left[\max_{p=p^v_{0,ab}..p^v_{L,ab}} \left(w^v_{ab}(p) - \varphi_{ab\,v(a)} - (p-1)T_a + \Phi_{ab} \right) \right] \quad (29)$$

Aplicando el análisis anteriormente descrito a cada tarea del sistema obtenemos los tiempos de respuesta global de peor caso y, comparándolos con su plazos respectivos, podemos verificar el cumplimiento de los requerimientos de tiempo real. Sin embargo, aunque esta técnica de análisis es exacta, representa un algoritmo NP-completo en el cual el número de casos que deben ser chequeados crece exponencialmente con el número de tareas. Esto significa que el algoritmo será intratable para la mayoría de problemas prácticos y no se podrá utilizar. Debido a esto, en la siguiente sección deduciremos una aproximación del análisis que obtiene cotas superiores de los tiempos de respuesta y en el cual la dependencia del número de casos respecto al número de tareas es polinómica.

4.2.3 Análisis aproximado de los tiempos de respuesta

En [TIN94B], Tindell desarrollo un método aproximado que nos permite obtener cotas superiores de los tiempos de respuesta de peor caso en un sistema compuesto por transacciones con *offsets* fijos. Aunque la técnica no es exacta, el número de casos que necesitamos chequear depende polinómicamente del número de tareas, lo que hace que se pueda aplicar en sistemas grandes. Si los tiempos de respuesta obtenidos con este método son menores que los respectivos plazos de ejecución, garantizamos que los requerimientos temporales serán satisfechos.

El análisis se basa en la técnica de análisis desarrollada en [TIN94B] que extendimos en la sección anterior. Allí obtuvimos la ecuación que calcula la contribución de peor caso de una transacción Γ_i al tiempo de respuesta de una tarea τ_{ab} cuando el instante crítico se hace coincidir con la activación de la tarea τ_{ik}:

$$W_{ik}(\tau_{ab},t) = \sum_{\forall j \in hp_i(\tau_{ab})} \left(\left\lfloor \frac{J_{ij}+\varphi_{ijk}}{T_i} \right\rfloor + \left\lceil \frac{t-\varphi_{ijk}}{T_i} \right\rceil \right) C_{ij} \quad (30)$$

El principal problema con la técnica de análisis es que no conocemos con qué tarea τ_{ik} debemos crear el instante crítico que inicia el periodo de ocupación. Esto motiva el que chequeemos todas las posible variaciones. Tindell evitó esto obteniendo una cota superior de la interferencia debida a tareas de la transacción Γ_i en un periodo de ocupación de duración *w*, como el máximo de todas las posibles interferencias causadas considerando cada una de las tareas de Γ_i como la que origina el periodo de ocupación:

$$W_i^*(\tau_{ab},w) = \max_{\forall k \in hp_i(\tau_{ab})} W_{ik}(\tau_{ab},w) \tag{31}$$

Por tanto, usando esta función en el cálculo de los tiempos de respuesta, podemos asegurar que el tiempo obtenido es una cota superior de la contribución de tareas de la transacción Γ_i al tiempo de respuesta y, consecuentemente, no será necesario calcular todas las posibles variaciones para *k*. Empleando una función similar para cada transacción, podemos calcular el tiempo de respuesta global de peor caso, para una tarea particular, chequeando un único caso.

A efectos de introducir menos pesimismo en el análisis, no usaremos dicha función para la transacción a la cual pertenece la tarea bajo análisis sino que utilizaremos la estrategia inicial. Consecuentemente, para el análisis deberemos considerar todos los posibles instantes críticos creados con cada una de las tareas pertenecientes al conjunto $hp_a(\tau_{ab})$, incluida la propia tarea τ_{ab}. El número de posibilidades es pequeño, igual al número de tareas en $hp_a(\tau_{ab})$. El tiempo de respuesta de peor caso se determina mediante :

$$w_{abc}(p) = B_{ab} + (p - p_{0,abc} + 1)\, C_{ab} + W_{ac}\left(\tau_{ab}, w_{abc}(p)\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, w_{abc}(p)\right) \tag{32}$$

Nuevamente podemos emplear el método iterativo para resolver esta ecuación El parámetro $p_{0,abc}$ corresponde a la primera activación que ocurre dentro del periodo de ocupación, y se calcula como:

$$p_{0,abc} = -\left\lfloor \frac{J_{ab} + \varphi_{abc}}{T_a} \right\rfloor + 1 \tag{33}$$

La longitud del periodo de ocupación viene dada por la expresión:

$$L_{abc} = B_{ab} + \left(\left\lceil \frac{L_{abc} - \varphi_{abc}}{T_a} \right\rceil - p_{0,abc} + 1 \right) C_{ab} + W_{ac}\left(\tau_{ab}, L_{abc}\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, L_{abc}\right) \tag{34}$$

y a partir de ella, calculamos la última activación a chequear:

$$p_{L,abc} = \left\lceil \frac{L_{abc} - \varphi_{abc}}{T_a} \right\rceil \tag{35}$$

o, al igual que antes, obteniendo el menor valor de $p \geq p_{0,abc}$ cuyo tiempo de respuesta verifique la relación:

$$R_{abc}(p) \leq T_a + \Phi_{ab} \tag{36}$$

donde el tiempo de respuesta de peor caso se obtiene restando del tiempo de finalización el instante en se produjo la llegada del evento externo asociado:

$$R_{abc}(p) = w_{abc}(p) - \varphi_{abc} - (p-1)T_a + \Phi_{ab} \tag{37}$$

y tomando el peor de los tiempos de respuesta obtenidos:

$$R_{ab} = \max_{\forall c \in hp_a(\tau_{ab}) \cup b} \left[\max_{p=p_{0,abc}..p_{L,abc}} \left(R_{abc}(p) \right) \right] \tag{38}$$

Nótese que este algoritmo requiere inspeccionar solamente un número de posibles instantes críticos igual al número de tareas en la transacción Γ_i con prioridad mayor o igual que la prioridad de τ_{ab}, incluida ella misma. Esto es normalmente un número reducido, con el cual podemos obtener resultados aceptables.

4.2.4. Tareas esporádicas con *offset*

En el análisis de tareas sin *offsets*, el caso de transacciones disparadas por eventos aperiódicos se puede tratar de forma similar a como se tratan los eventos periódicos, siempre y cuando se pueda garantizar un intervalo mínimo T_{min} entre llegadas. Como veremos, este criterio se mantiene para tareas esporádicas con *offsets* menores o iguales que el periodo. Sin embargo, esto no es cierto si el *offset* de alguna tarea de la transacción es mayor que el intervalo mínimo entre llegadas, puesto que entonces el peor caso para la ejecución de una tarea de menor prioridad no tiene porqué suceder cuando el evento esporádico llegue a su máximo ritmo posible.

La Figura 4-3 muestra un ejemplo donde se puede ver claramente este efecto. Las flechas descendentes representan la llegada de eventos, mientras que las flechas ascendentes representan la activación de las tareas, una vez transcurrido un tiempo igual a su *offset* desde la llegada del evento. Un evento esporádico, con tiempo mínimo entre llegadas T_{min}, dispara

la ejecución de una transacción Γ_i en un único procesador. La transacción está constituida por dos tareas: la primera tarea τ_{i1} (coloreada en gris en la figura) tiene *offset* nulo $\Phi_{i1}=0$, y la segunda (sin colorear) un *offset* $\Phi_{i2}=\Phi$ mayor que T_{min}; ambas tareas se activan sin retraso, $J_{i1}=J_{i2}=0$.

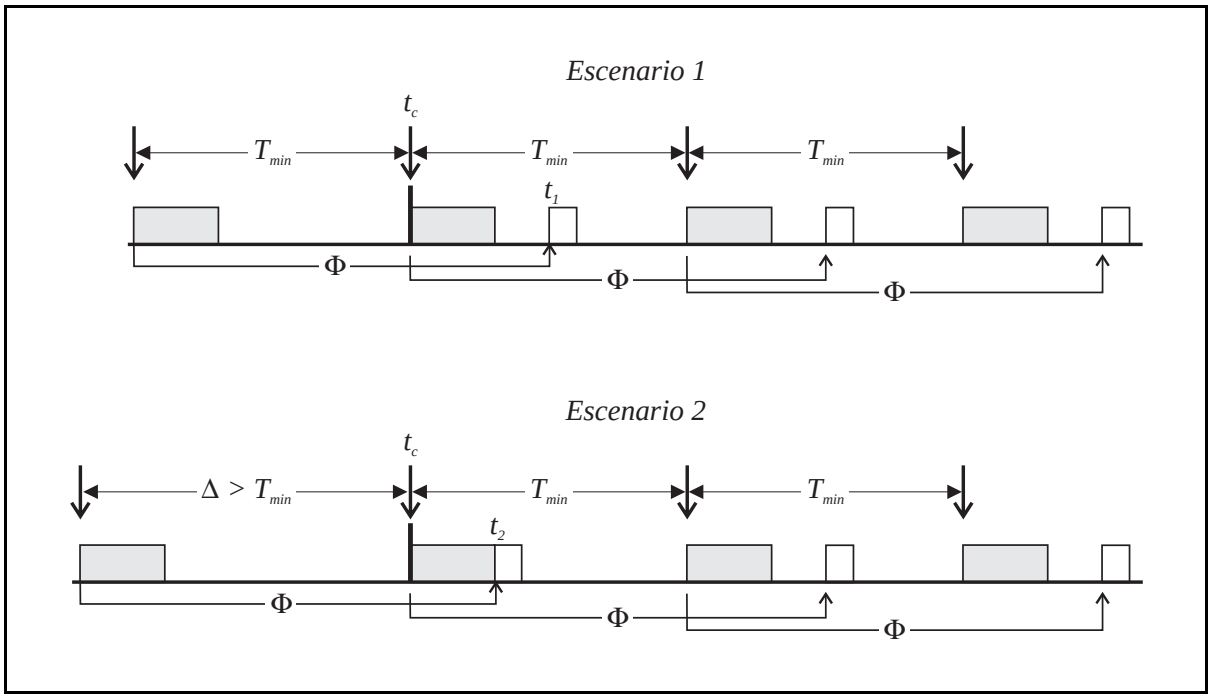


Figura 4-3. Escenarios para una transacción esporádica con *offsets* mayores que T_{min}

El *Escenario 1* en la parte superior muestra el instante crítico t_c construido con la tarea τ_{i1} para el análisis de una tarea de prioridad menor, tal y como establecimos con los Teoremas 4-1 y 4-2, en la sección 4.2.2, si consideramos la ocurrencia de eventos separadas por un intervalo T_{min}. Como se puede ver, la primera activación de la tarea τ_{i1}, dentro del periodo de ocupación, coincide con el instante crítico t_c mientras que la primera activación de τ_{i2} se produce en el instante t_1. Sin embargo, en el *Escenario 2* construimos otro posible instante crítico iniciado con τ_{i1} pero en el que la activación coincidente con el instante crítico se ha producido transcurrido un intervalo $\Delta>T_{min}$ desde la última activación. Como podemos ver, en este caso la primera activación de τ_{i1} dentro del periodo de ocupación también se produce en el instante t_c, pero la tarea τ_{i2} se activa en el instante t_2, anterior al t_1, por lo que este *Escenario 2* representa una situación potencialmente peor para tareas de prioridad inferior que la construida en el *Escenario 1*.

Como se ha podido ver con este ejemplo, las tareas esporádicas con *offsets* mayores que el periodo mínimo pueden compensar el desfase inicial con el instante crítico, mediante activaciones que se produzcan a un ritmo menor que el máximo. Esto hace que el instante crítico no se pueda construir de la misma forma que antes y, por tanto, no se puede aplicar el mismo análisis. Para obtener una técnica válida para transacciones esporádicas nos basaremos en los siguientes dos teoremas:

Teorema 4-3. La contribución de tareas pertenecientes a una transacción esporádica con periodo mínimo entre llegadas T_{min} y *offsets* menores que T_{min} al peor caso de tareas de menor prioridad es máxima cuando los eventos ocurren a su mayor ritmo posible, es decir, cuando las activaciones se producen periódicamente cada T_{min}.

Demostración: Para demostrar este teorema fijémonos en la Figura 4-4; en ella se representan con flechas descendentes las ocurrencias de eventos esporádicos y con flechas ascendentes los *offsets* de cada tarea activada en la transacción. Veamos cuál es la máxima interferencia posible de tareas de la transacción esporádica sobre el periodo de ocupación que comienza en el instante crítico t_c. Para ello, nos fijaremos en la última instancia ocurrida antes o en el instante crítico, en el instante que llamaremos t_1. Esta activación está desfasada una cantidad ϕ respecto de t_c. Llamaremos t_0 y t_2 a los instantes en los que llegan los eventos anterior y posterior al ocurrido en t_1, respectivamente.

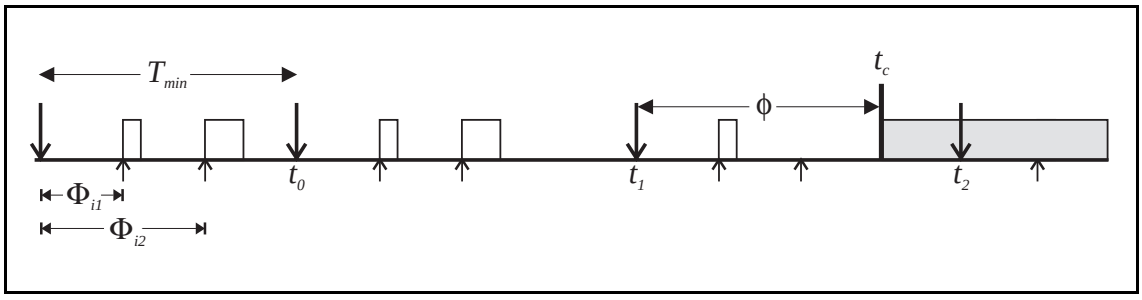


Figura 4-4. Activaciones de una transacción esporádica

Dado que los *offsets* son menores que T_{min}, todas las tareas correspondientes al evento anterior a t_1, ocurrido en el instante t_0 estarían preparadas para ejecutar antes del instante t_1, si su *jitt*er fuese cero. Por tanto, acercando estos instantes lo más posible a t_1 sólo podemos aumentar la posibilidad de que una o varias de estas tareas se puedan retrasar, por efecto del *jitter*, hasta el instante crítico, pasando en ese caso a formar parte del

periodo de ocupación. Por ello, una separación entre t_0 y t_1 igual al mínimo, T_{min}, es la que consigue una contribución mayor por parte de las tareas de la instancia activada en t_0. El mismo razonamiento se puede aplicar a todos los eventos ocurridos antes del instante t_0. Nótese que este razonamiento no sería cierto si algún *offset* fuese mayor que T_{min}, tal como ocurría en el ejemplo mostrado en la Figura 4-3.

Por definición, el instante t_2 siempre es posterior al instante crítico, por lo que si conseguimos que esa activación se produzca lo antes posible, aumentaremos la probabilidad de que una o varias de las tareas asociadas ocurran dentro del periodo de ocupación. Exactamente el mismo razonamiento es aplicable a los eventos posteriores al instante t_2, de forma que la máxima contribución se conseguirá cuando los eventos ocurran cada T_{min}.

□

Este teorema nos permitiría analizar las transacciones esporádicas con *offsets* menores que el intervalo mínimo entre llegadas como si fuera una transacción periódica de periodo T_{min}. El siguiente lema nos permitirá analizar el caso de *offsets* mayores.

Lema 4-1. La contribución de tareas pertenecientes a una transacción esporádica con periodo mínimo entre llegadas T_{min} y *offsets* mayores que T_{min} al peor caso de tareas de menor prioridad nunca puede ser mayor que la producida por un conjunto de transacciones periódicas independientes, construida cada una de ellas con una tarea de la transacción original, manteniendo para ella los mismos valores de tiempo de ejecución y término de retraso, y con periodo igual a T_{min}.

Demostración: La demostración es trivial, puesto que si consideramos la ejecución de cada tarea de la transacción de forma independiente construiremos un posible peor caso sin tener en cuenta las restricciones en las fases relativas de activación de las tareas, por lo que estaremos creando una situación peor de la que realmente se puede dar en el sistema con la transacción original.

□

El Teorema 4-3 y el Lema 4-1 nos permiten derivar una técnica para analizar sistemas donde existan transacciones esporádicas con *offsets* menores y mayores que el período mínimo de activación T_{min}. Está técnica consiste en sustituir, en el sistema bajo análisis, cada transacción esporádica por un conjunto de transacciones periódicas con periodo T_{min}: una formada por todas las tareas con *offsets* menores que T_{min} y una transacción por cada tarea con *offset* mayor que T_{min}(conservando el mismo *offset*, de forma que podamos utilizarlo para calcular su tiempo de respuesta global, relativo al comienzo de la transacción). Los tiempos de respuesta obtenidos para las tareas de este sistema equivalente serán cotas superiores de los tiempos de respuesta de las mismas tareas en el sistema original. La Figura 4-5 muestra un ejemplo de una transacción esporádica formada por tareas con *offsets* mayores y menores que T_{min} y el modelo que manejaremos para el análisis de peor caso.

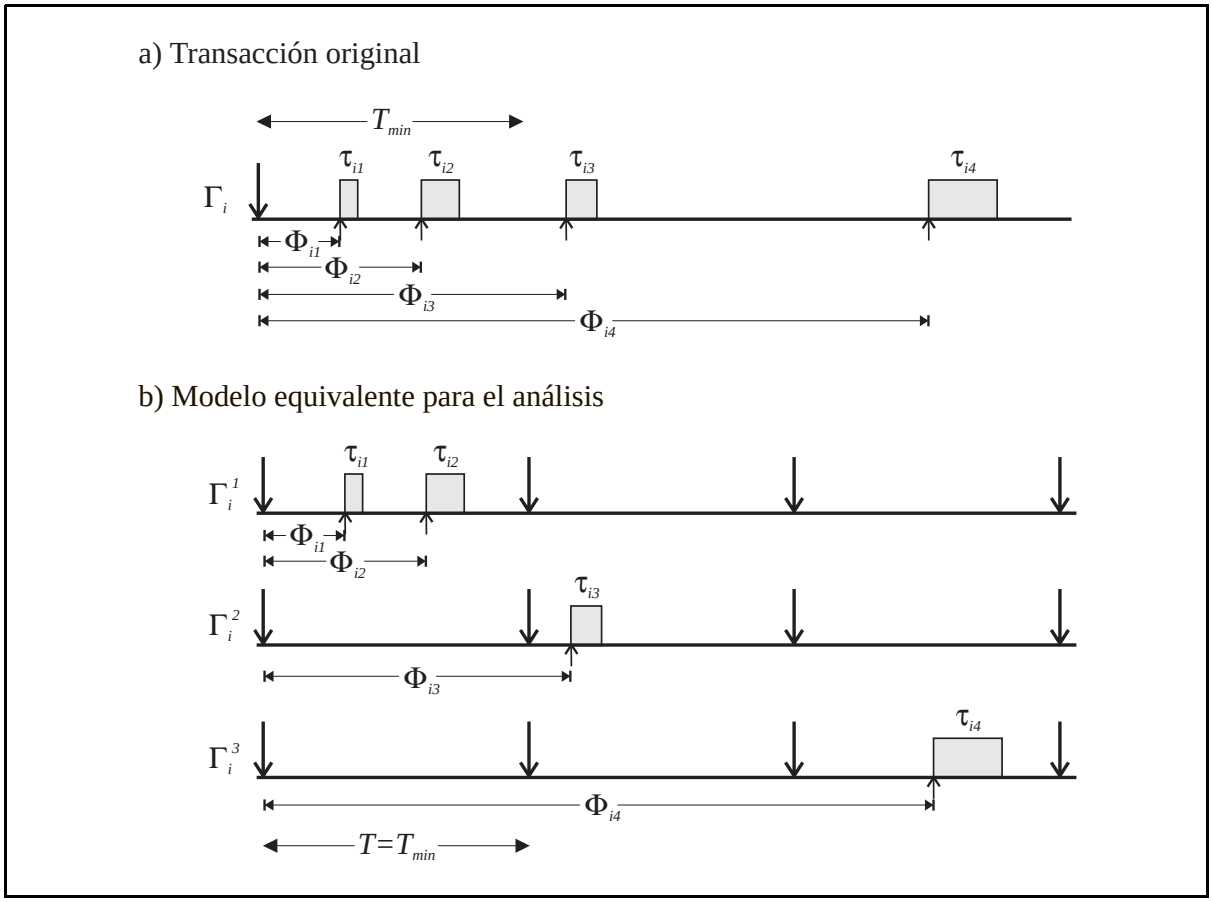


Figura 4-5. Modelo equivalente para el análisis de una transacción esporádica

Para obtener una cota superior de la contribución de tareas de una transacción esporádica Γ_i al peor caso de una tarea τ_{ab} de prioridad inferior debemos tener en cuenta la

contribución de cada uno de los dos tipos de transacciones definidos previamente. El valor desde el instante crítico hasta el instante *t* se obtiene como:

$$W_i^*(\tau_{ab},t) = WS_i^*(\tau_{ab},t) + WP_i^*(\tau_{ab},t)$$

siendo $WS_i^*(\tau_{ab},t)$ la contribución debida a tareas con *offsets* menores que T_{min} y $WP_i^*(\tau_{ab},t)$ la debida a tareas con *offsets* mayores que T_{min}.

Para calcular el valor de $WS_i^*(\tau_{ab},t)$ aplicaremos la expresión obtenida para una transacción periódica con periodo T_{min}. La contribución de peor caso será la mayor de las calculadas con los posibles instantes críticos creados con cada una de las tareas de la transacción, esto es:

$$WS_i^*(\tau_{ab},t) = \max_{\forall k \in S_i(\tau_{ab})} WS_{ik}(\tau_{ab},t) = \max_{\forall k \in S_i(\tau_{ab})} \left[\sum_{\forall j \in S_i(\tau_{ab})} \left(\left\lfloor \frac{J_{ij}+\varphi_{ijk}}{T_{\min}} \right\rfloor + \left\lceil \frac{t-\varphi_{ijk}}{T_{\min}} \right\rceil \right) C_{ij} \right] \qquad (40)$$

donde,

$S_i(\tau_{ab})$ es el conjunto de tareas pertenecientes a la transacción Γ_i con prioridad mayor que la de la tarea τ_{ab} y *offset* menor o igual que T_{min} .

En la contribución debida a tareas con *offsets* mayores que T_{min} debemos considerar una transacción por una de ellas. Si tenemos en cuenta que cada transacción equivalente estará formada por una única tarea, podemos ver que la contribución de la transacción que contiene a la tarea τ_{ik} se calcula mediante:

$$WP_{ik}(\tau_{ab},t) = \left(\left\lfloor \frac{J_{ik}+\varphi_{ikk}}{T_{\min}} \right\rfloor + \left\lceil \frac{t-\varphi_{ikk}}{T_{\min}} \right\rceil \right) C_{ik} \qquad (41)$$

donde φ_{ikk}, se obtiene a partir de la expresión (19), como:

$$\varphi_{ikk} = T_{\min} - (\phi_{ik}+J_{ik}-\phi_{ik}) \bmod T_{\min} = T_{\min} - J_{ik} \bmod T_{\min} = T_{\min} - J_{ik} + \left\lfloor \frac{J_{ik}}{T_{\min}} \right\rfloor T_{\min} \qquad (42)$$

Si sustituimos en la ecuación (41) y operamos:

$$WP_{ik}(\tau_{ab},t) = \left(\left\lfloor \frac{T_{\min} + \left\lfloor \frac{J_{ik}}{T_{\min}} \right\rfloor T_{\min}}{T_{\min}} \right\rfloor + \left\lceil \frac{t - T_{\min} + J_{ik} - \left\lfloor \frac{J_{ik}}{T_{\min}} \right\rfloor T_{\min}}{T_{\min}} \right\rceil \right) C_{ik} = \left\lceil \frac{t+J_{ik}}{T_{\min}} \right\rceil C_{ik} \qquad (43)$$

obtenemos una expresi n similar a la correspondiente a una tarea sin offset Si consideramos todas las tareas con offsets

transacción representa una tarea (o una sección de una tarea) ejecutando en un procesador, o un mensaje transmitido por un canal de comunicación y se activa cuando se completa la ejecución de la acción previa en la transacción. Esto nos permite modelar la activación de una tarea por la llegada de un mensaje o la transmisión de un mensaje por la finalización de la ejecución de una sección de código. La primera acción de la transacción se activa por la llegada del evento periódico externo, que supondremos sin retraso. Usaremos un modelo similar al utilizado en la sección 4.3 para tareas que se suspenden, definiendo un *offset* equivalente, igual a cero para la acción que inicia la transacción e igual al resultado de la siguiente expresión para las acciones que no inician la transacción:

$$\begin{aligned} \Phi'_{ij} &= \Phi_{ij,\min} = R^{b}_{ij-1} \\ J'_{ij} &= J_{ij} + \Phi_{ij,\max} - \Phi_{ij,\min} = R_{ij-1} - R^{b}_{ij-1} \end{aligned} \tag{49}$$

donde $R^{b}_{ij\text{-}1}$ es un cota inferior del tiempo de respuesta de mejor caso de la acción previa en la transacción, $\tau_{ab\text{-}1}$, y $R_{ij\text{-}1}$ una cota superior del tiempo de respuesta de peor caso de la misma acción previa.

Evidentemente, una tarea que ejecute en un procesador determinado no puede ser expulsada por tareas que ejecuten en otro procesador y, por tanto, debemos redefinir el conjunto $hp_i(\tau_{ab})$ de acciones que pueden expulsar la ejecución de una acción τ_{ab} para considerar sólo las que se encuentren en el mismo recurso:

$$hp_i(\tau_{ab}) = \{\, j \in \Gamma_i \mid prioridad(\tau_{ij}) \geq prioridad(\tau_{ab}) \wedge recurso(\tau_{ij}) = recurso(\tau_{ab}) \,\} \tag{50}$$

Empleando esos términos equivalentes de *offset* y retraso podemos utilizar el mismo método iterativo presentado en la sección 4.3, el algoritmo APCOD, pero usando la ecuación (49) para calcular los términos equivalentes de retraso y *offset*, y comenzando con valores iniciales $J_{ij}=0$ y $\phi_{ij}=R^{b}_{ij\text{-}1}$ para cada acción. Como antes, la convergencia del método iterativo está garantizada por la dependencia monótona de los tiempos de respuesta respecto de los términos de retraso.

Para comprender mejor esta técnica, la ilustraremos con un ejemplo sencillo. Considérese el sistema que aparece en la Figura 4-7. Las tareas 1, 3 y 5 son tareas periódicas simples pero la tarea 2 es una tarea periódica que se suspende para requerir servicio de la tarea 4. Justo antes de la suspensión, la tarea 2 transmite un mensaje, m_1, a través de una línea

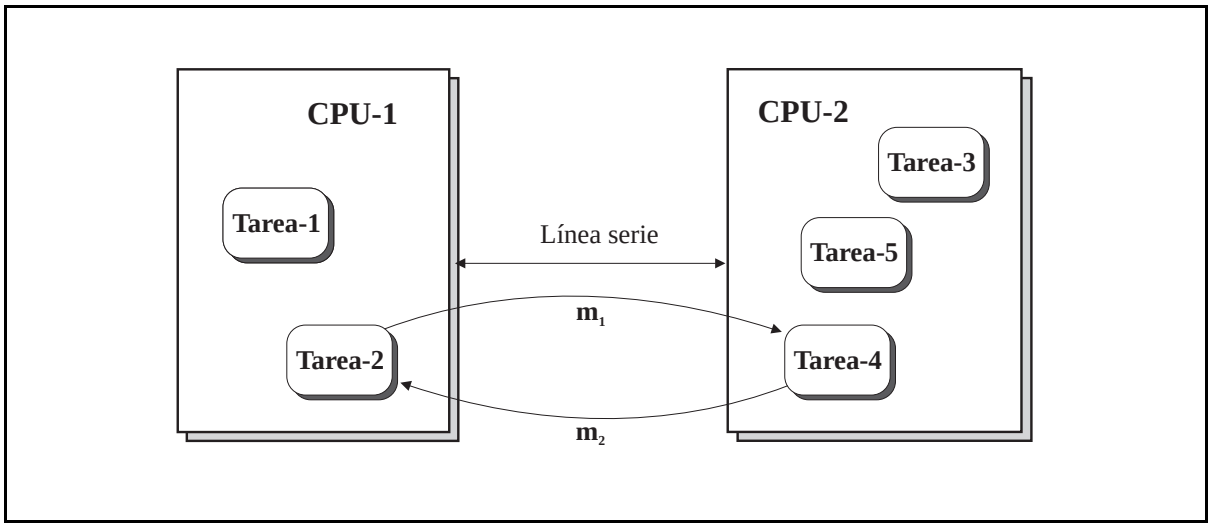


Figura 4-7. Sistema distribuido sencillo

Trans.	Acción	Original	C_i	T_i	D_i	Prio.
Γ_1	τ_{11}	tarea-1	4	20	20	Alta
Γ_2	τ_{21}	tarea-2_1	20	150	150	Baja
	τ_{22}	m_1	25			-
	τ_{23}	tarea-4	15			Media
	τ_{24}	m_2	34			-
	τ_{25}	tarea-2_2	30			Baja
Γ_3	τ_{31}	tarea-3	5	30	30	Alta
Γ_5	τ_{51}	tarea-5	100	200	200	Baja

Tabla 4-I. Parámetros temporales del ejemplo

serie. La tarea 4 se activa por la llegada de ese mensaje al procesador CPU-2 y cuando completa su ejecución envía un mensaje de vuelta, m_2, a través de la misma línea serie. Entonces, la tarea 2 reasume su ejecución hasta que ésta se completa. Las tareas 2 y 4 y sendos mensajes pueden ser modelados como una transacción con cinco acciones. La Tabla 4-I muestra los parámetros de todas las transacciones, así como de las acciones que las componen. Supondremos que la política de planificación en la línea serie es FIFO; esto puede ser fácilmente modelado si suponemos, cuando realicemos el análisis de un mensaje, que el otro mensaje tiene mayor prioridad. Supondremos también que los tiempos de ejecución de cada tarea y los tiempos de transmisión de cada mensaje son fijos, esto es, no hay diferencia entre los tiempos de ejecución o de transmisión en el mejor y en el peor casos. Esto significa que podemos utilizar los tiempos de transmisión o de ejecución como respuesta local de mejor

caso cada acción asociada, de forma que calcularemos los términos de retraso a partir de la expresión:

$$\Phi_{ij} = R^{b}_{ij} = \sum_{k=1..j-1} C_{ik} \quad (51)$$

Si analizamos el sistema descrito utilizando la técnica de análisis desarrollado por Tindell y Clark (ver capítulo 2), en la que se suponen todas las acciones independientes entre sí, obtenemos un tiempo de respuesta para la transacción Γ_2 de 266 unidades de tiempo, que excede el plazo de ejecución asignado de 150. Sin embargo, si aplicamos la técnica de análisis descrita en esta sección, obtenemos un tiempo de respuesta de peor caso para la misma transacción de 145 unidades de tiempo, que hace que la transacción sea planificable (la Tabla 4-II muestra los resultados del análisis para las cinco acciones de la transacción Γ_2). La razón de esta diferencia es que, en el análisis original, la primera porción de la tarea 2 es expulsada una vez por la segunda porción y viceversa. Lo mismo ocurre con los mensajes en la red: cada uno expulsa la ejecución del otro. En la práctica estas expulsiones no son posibles, debido al *offset* de cada una. Esto sí es tenido en cuenta correctamente en el análisis con *offsets* dinámicos desarrollado en esta sección, de forma que se obtienen resultados más realistas.

tarea	Φ_{2j}	J_{2j}	R_{2j}
τ_{21}	0	0	28
τ_{22}	20	8	53
τ_{23}	45	8	73
τ_{24}	60	13	107
τ_{25}	94	13	145

Tabla 4-II. Resultados para la transacción Γ_2

4.5. Comparación con las técnicas existentes

Hemos comparado los resultados del análisis para tareas con *offsets* dinámicos con los resultados obtenidos utilizando la técnica de análisis actual para sistemas distribuidos, que supone las acciones independientes unas de otras. Con este propósito, hemos realizado exhaustivas simulaciones para diferentes conjuntos de tareas con tiempos de ejecución y periodos generados aleatoriamente y las prioridades se han asignado en función de los plazos de ejecución. En esta sección mostramos algunos de los resultados obtenidos.

El primer conjunto de gráficas (Figura 4-8 a Figura 4-10) compara los tiempos de respuesta obtenidos usando la técnica de Tindell y Clark para tareas independientes, R_{indep}, con los tiempos de respuesta obtenidos utilizando el algoritmo APCOD, R_{APCOD}. En cada punto de esas gráficas mostramos la relación promedio R_{indep}/R_{APCOD} de cinco simulaciones. El eje de abcisas representa la utilización del procesador. Cada figura representa los resultados para tres relaciones diferentes entre los máximos y los mínimos periodos de las transiciones, T_{max}/T_{min}. La Figura 4-8 muestra los resultados para conjuntos formados por 10 transacciones y 10 tareas por transacción, ejecutando en un procesador, para el caso de que los tiempos de respuesta de mejor caso sean despreciables y considerados todos como nulos y, por tanto, todas las acciones con *offsets* iguales a 0. Se puede ver que para niveles de utilización normales, en torno al 70%, los tiempos de respuesta con tareas independientes son entre 2.2 y 2.6 veces mayores que los obtenidos con el análisis con *offsets* dinámicos. Los resultados obtenidos suponiendo tiempos de ejecución de mejor caso iguales a los de peor caso son también bastante parecidos.

La Figura 4-9 muestra los resultados para un caso similar, pero ejecutando en cuatro procesadores. Podemos ver que al disminuir el número de tareas ejecutando en el mismo procesador y pertenecientes a la misma transacción, disminuyen también los beneficios del algoritmo APCOD. Sin embargo, esos beneficios son todavía significativos, con tiempos de respuesta entre 1.27 y 1.37 veces mejores para utilizaciones del 70%. La Figura 4-10 muestra los resultados para el mismo caso que la Figura 4-9, excepto que los tiempos de respuesta de mejor caso se consideran iguales a la suma de los tiempos de ejecución de ellas mismas y todas sus predecesoras en la misma transacción. Podemos ver que, en este caso, los resultados son significativamente mejores, con tiempos de respuesta entre 2 y 2.6 veces mejor que en el análisis para tareas independientes y una utilización del 70%.

El segundo conjunto de gráficas (Figura 4-11 y Figura 4-12) compara la utilización máxima para la que un determinado conjunto de tareas es planificable, cuando ésta se calcula con el análisis basado en tareas independientes y cuando se calcula utilizando la técnica basada en *offsets* dinámicos. El análisis con *offsets* dinámicos se realiza para *offsets* nulos, suponiendo los tiempos de ejecución de mejor caso iguales a cero, y para tiempos de ejecución de mejor caso iguales a los de peor caso. La máxima utilización planificable se obtiene analizando un sistema con utilización baja y aumentado progresivamente la utilización

hasta que el sistema deja de cumplir algún plazo. La máxima utilización planificable corresponde al último conjunto de tareas que cumple todos sus plazos. Las simulaciones se han hecho para diferentes relaciones entre los periodos de las tareas, D_i/T_i. La Figura 4-11 muestra los resultados de la simulación de un sistema con 4 procesadores, 5 transacciones y 20 tareas por transacción, distribuidas en los diferentes procesadores, para una relación T_{max}/T_{min}=100. La Figura 4-12 muestra los resultados para un sistema similar, pero con 12 tareas en cada transacción en lugar de 20. Vemos que para valores de D_i/T_i=2 y superiores, conseguimos un incremento en torno al 8% en la utilización planificable en el caso de 12 tareas, y de más del 25% para el caso de 20 tareas. Esto se explica por el hecho de que el análisis con *offsets* se beneficia del incremento de tareas en un mismo procesador. Los resultados son mejores si consideramos tiempos de respuesta de mejor caso mayores que cero, pero es interesante destacar que las mejoras son importantes aún considerando tiempos de respuesta de mejor caso iguales a cero.

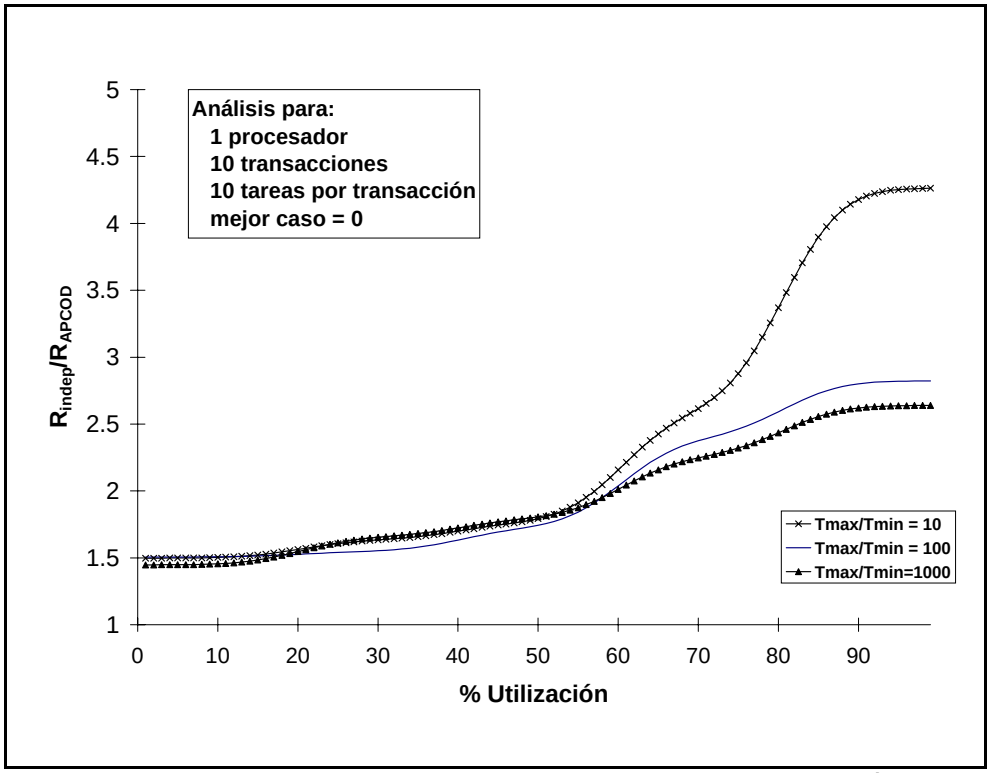


Figura 4-8. Resultados de R_{indep}/R_{APCOD}, para 1 procesador y C_i^b = 0

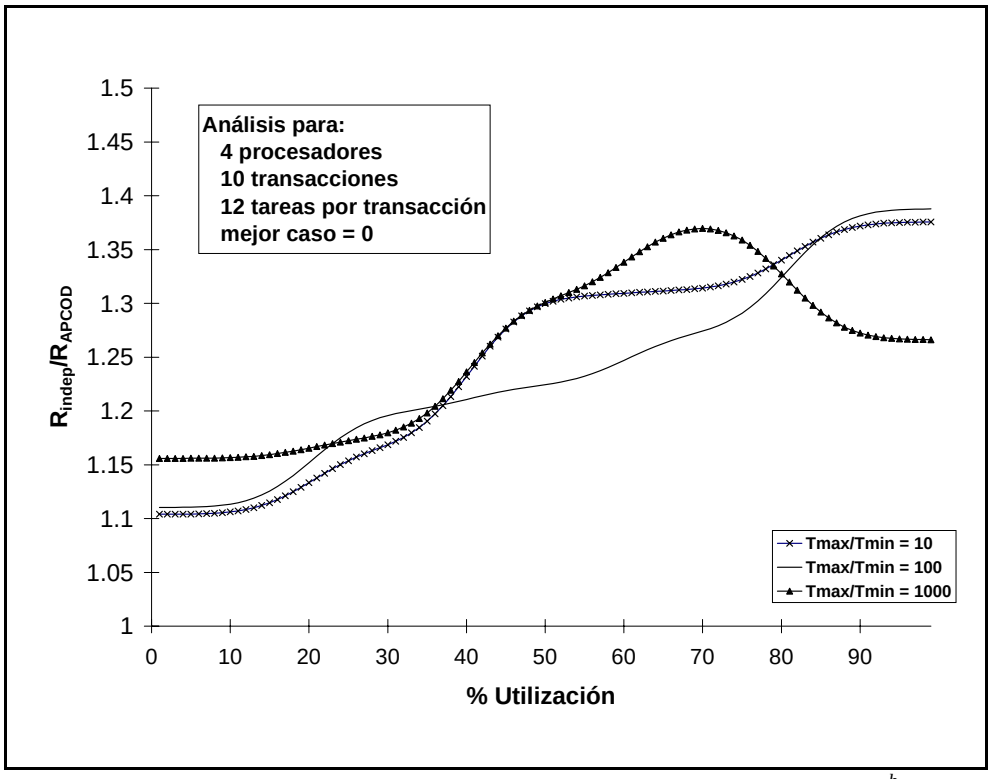


Figura 4-9. Resultados de $R_{\text{indep}}/R_{\text{APCOD}}$, para 4 procesadores y $C_i^b = 0$

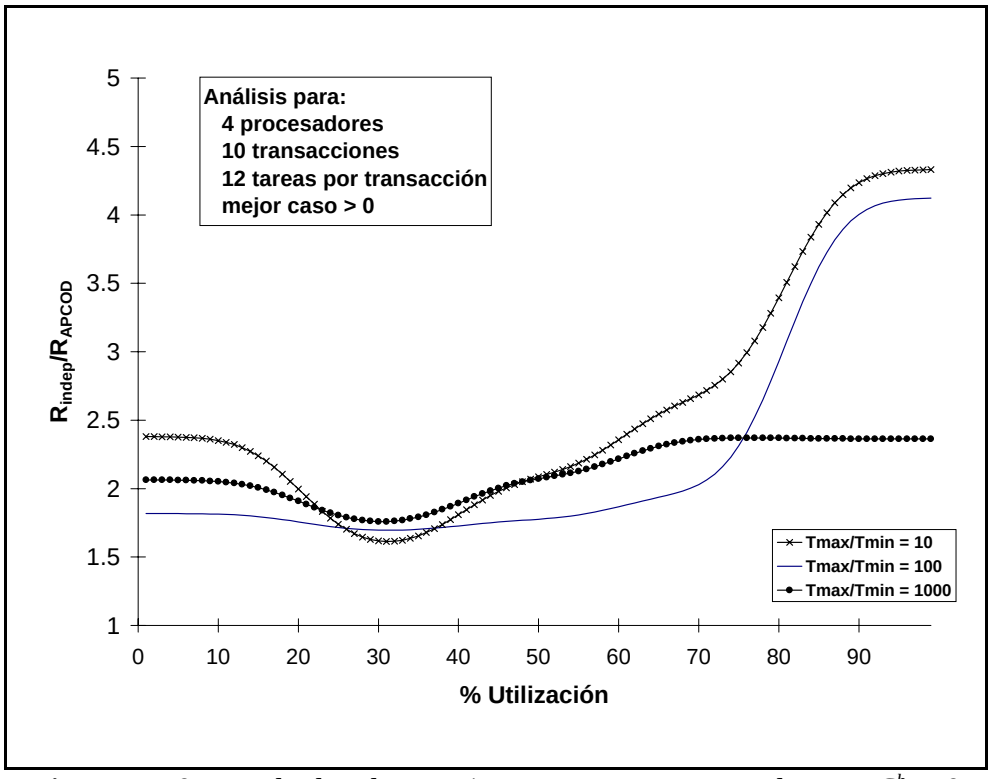


Figura 4-10. Resultados de $R_{\text{indep}}/R_{\text{APCOD}}$, para 4 procesadores y $C_i^b > 0$

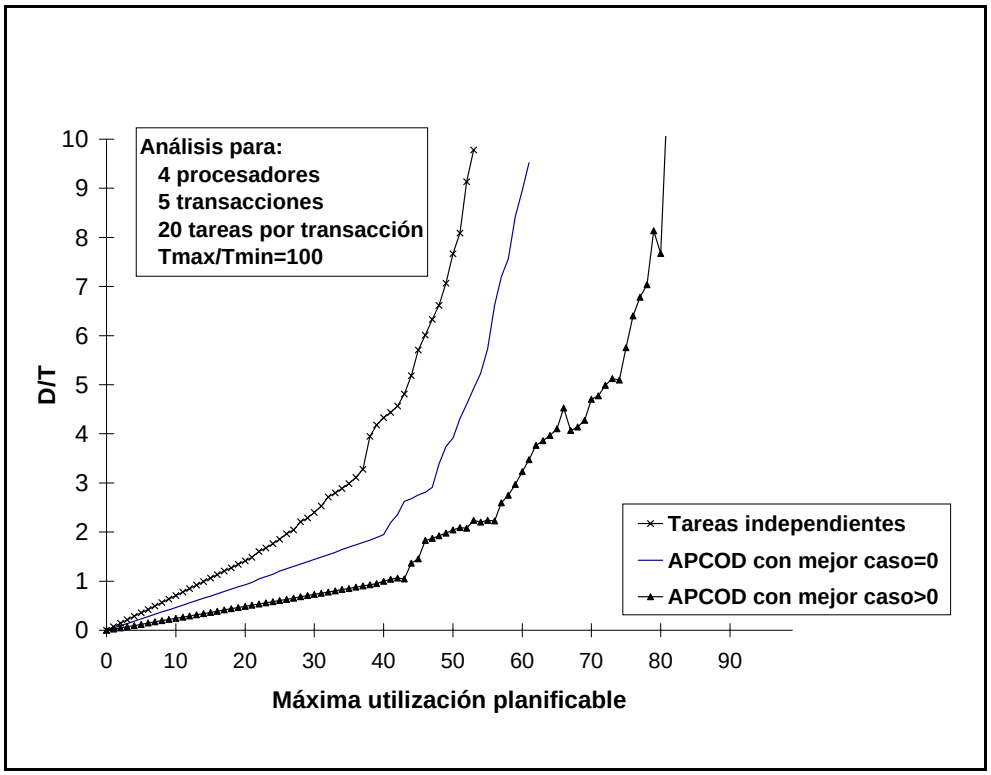


Figura 4-11. Máxima utilización planificable, 20 tareas por transacción

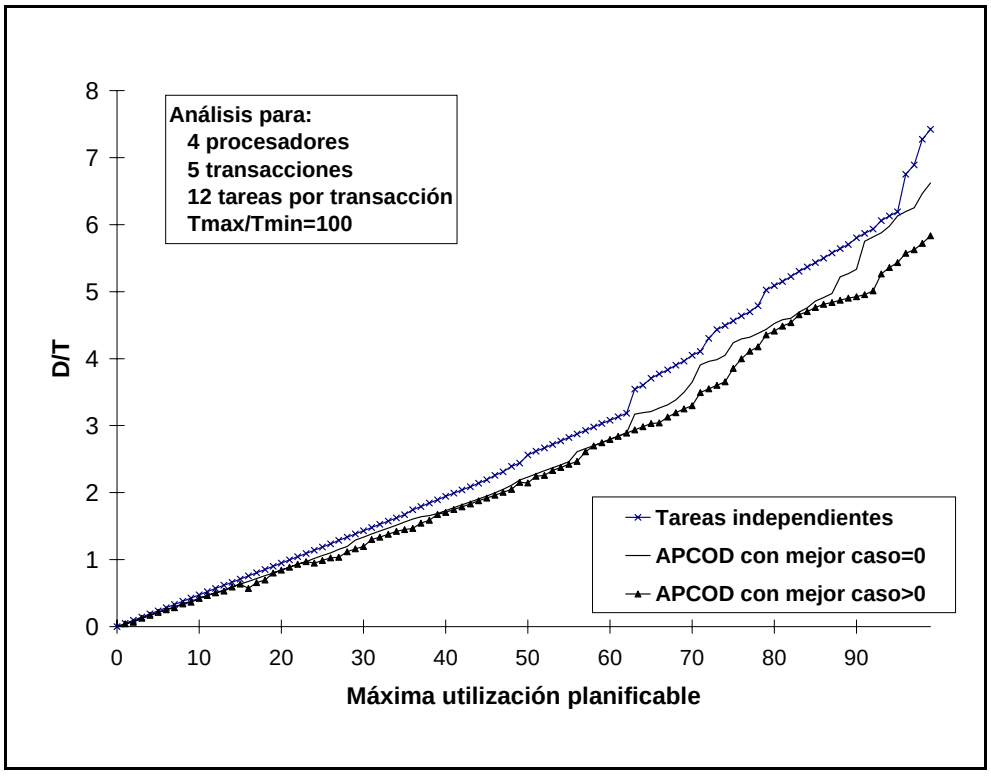


Figura 4-12. Máxima utilización planificable, 12 tareas por transacción

5. Mejoras al análisis de tareas con offsets y relaciones de precedencia.

5.1. Introducción

En el capítulo 4 de esta tesis hemos desarrollado técnicas para el análisis de tareas con *offsets* estáticos y dinámicos. A partir de ellas, diseñamos además un algoritmo para extender el análisis a sistemas con relaciones de precedencia en la activación de las tareas, tales como los sistemas multiprocesadores y distribuidos, o sistemas con tareas que se suspenden. Ese algoritmo recursivo, APCOD, obtenía los términos de retraso y *offset* correspondientes a cada tarea en función de los tiempos de respuesta de peor y de mejor caso de la tarea precedente en la secuencia de respuesta al evento externo y, de esa forma, tenía en cuenta el efecto de esa precedencia. Si bien este algoritmo garantizaba la obtención de cotas superiores de los tiempos de respuesta cuando existe precedencia en la activación de las tareas, no explotaba suficientemente esta característica.

En este capítulo vamos a refinar el cálculo de los tiempos de respuesta de peor caso eliminando pesimismo del análisis. En la sección 5.2 mejoraremos el algoritmo teniendo en cuenta que una tarea no puede ser expulsada por tareas a las que precede si estas últimas corresponden a la misma o a posteriores instancias del evento. Esta mejora supone introducir ligeros cambios en el algoritmo APCOD, ya que sólo debemos modificar la contribución debida a tareas precedidas por la tarea analizada y, como veremos en la sección 5.2.1, en bastantes casos obtiene reducciones apreciables de los tiempos de respuesta de peor caso.

Otra mejora sustancial se consigue si tenemos en cuenta el perfil de prioridades asignado a las tareas de una misma transacción. Por ejemplo, dos tareas de prioridad alta activadas en una misma instancia no pueden interferir simultáneamente la ejecución de otra tarea de prioridad media si en esa misma transacción existe una tarea de prioridad menor

activada entre las dos primeras. En la sección 5.3 se tienen en cuenta estas posibles incompatibilidades para ajustar la máxima interferencia que puede sufrir una tarea y, por tanto, reducir su tiempo de respuesta. En la sección 5.3.1 veremos como afecta este cambio a la interferencia debida a tareas de transacciones diferentes a la de la tarea analizada y en la sección 5.3.2 como afecta a las tareas que tienen relaciones de precedencia con la tarea analizada. En la sección 5.3.3 evaluaremos las mejoras conseguidas, comparando los resultados obtenidos con el nuevo algoritmo y las soluciones dadas por el algoritmo APCOD para diferentes conjuntos de sistemas a analizar.

Finalmente, en la sección 5.4 aplicaremos el análisis desarrollado a sistemas formados por tareas con prioridades de ejecución variante [GON91A] y compararemos los resultados obtenidos con esta nueva formulación y con la técnica original para sistemas monoprocesadores.

5.2. Relaciones de precedencia en la transacción analizada

El análisis del tiempo de respuesta de peor caso de una tarea τ_{ab}, desarrollado en la sección 4.2.3, tiene en cuenta varias componentes para el cálculo del tiempo de finalización:

$$w_{abc}(p) = B_{ab} + (p - p_{0,abc} + 1)\, C_{ab} + W_{ac}\left(\tau_{ab}, w_{abc}(p)\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, w_{abc}(p)\right) \qquad (1)$$

Aparecen, por un lado, el término de bloqueo debido al uso de recursos compartidos, B_{ab}, y la expulsión debida a tareas de otras transacciones distintas de Γ_a, $W_i^*(\tau_{ab}, w_{abc}(p))$. Por otro lado, aparecen las ejecuciones de la propia tarea τ_{ab}, $(p-p_{0,abc}+1)C_{ab}$, y la interferencia debida a tareas pertenecientes a su misma transacción cuando se analiza el instante crítico creado con la tarea τ_{ac}, $W_{ac}(\tau_{ab}, w_{abc}(p))$. En esta sección vamos a obtener una expresión más realista para ese último término, en la que no se tenga en cuenta la expulsión debida a tareas precedidas por τ_{ab} y activadas en la misma o posteriores instancias.

Para ello, vamos a identificar cada activación de una tarea en función del instante en que llegó el evento que desencadenó su ejecución. Nótese que este esquema es ligeramente diferente al utilizado en el capítulo anterior, en el que se identificaba cada tarea en función del instante en que se producía la activación de la tarea. Identificaremos con números positivos consecutivos los eventos producidos después del instante crítico, asignando p'=1 a

las tareas activadas en la transacción Γ_i disparada por el evento que llegue en el intervalo $(0,T_i]$, $p'=2$ a las del evento producido en $(T_i,2T_i]$, etcétera. Asimismo, los eventos ocurridos antes del instante crítico se identificaran con números consecutivos $p'\leq 0$; a las tareas de la transacción disparada en el intervalo $(-T_i,0]$ les corresponde el valor $p'=0$, a las del intervalo $(-2T_i,-T_i]$ el valor $p'=-1$, etcétera. De esta forma, identificamos con el mismo índice activaciones que tienen relaciones de precedencia y correspondientes a la misma instancia.

Vamos a identificar las activaciones de tareas de una transacción Γ_i en un periodo de ocupación determinado. En la figura Figura 5-1 se muestran tres escenarios de activación para la tarea τ_{ij} que nos ayudarán a esa identificación y que muestran el instante crítico t_c desfasado una cantidad ϕ respecto a la llegada de los eventos que disparan la transacción Γ_i. El Escenario 1 corresponde al caso $\phi \geq \Phi_{ij}$ mientras que el Escenario 2 muestra el caso $\phi < \Phi_{ij}$;

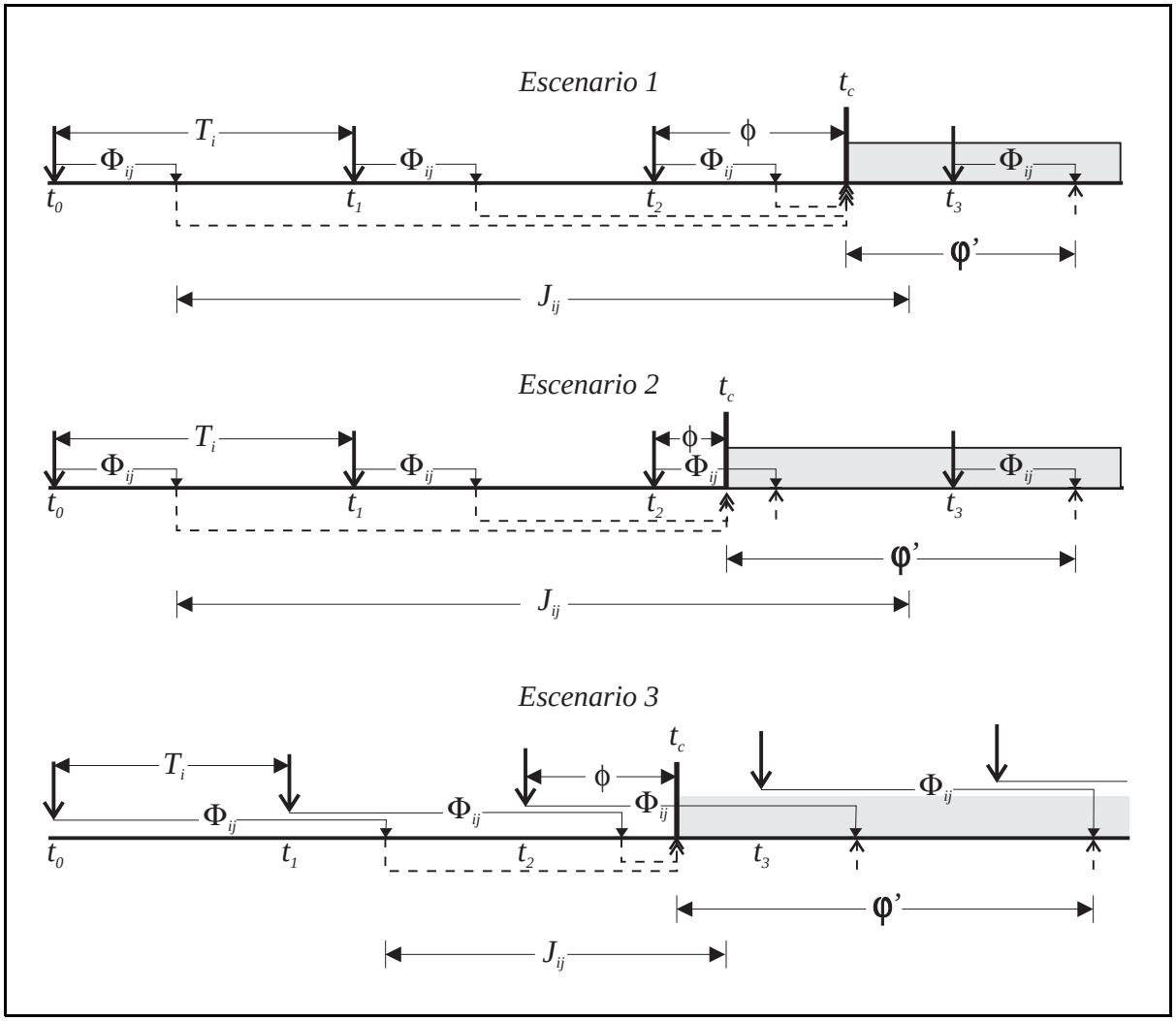


Figura 5-1. Escenarios de ejecución

por último, el Escenario 3 corresponde a una tarea con *offset* mayor que el periodo, $\Phi_{ij}>T_i$, y $\phi<\Phi_{ij}$ (si ϕ fuera mayor que Φ_{ij} nos referiríamos al siguiente evento, con un nuevo $\phi'<\Phi_{ij}$). Nuevamente, con flechas descendentes representamos la llegada de eventos y con flechas ascendentes la activación de las tareas. Las líneas horizontales continuas representan los *offsets* y las discontinuas los retrasos en las activaciones. Calcularemos el número n'_{ij} de eventos ocurridos antes del instante crítico y que todavía tienen pendiente la ejecución de la tarea τ_{ij} en el instante crítico. En la figura, los eventos llegados en los instantes t_0, t_1 y t_2 se encuentran en esa situación, por tanto, en las tres situaciones $n'_{ij}=3$.

Para el cálculo de n'_{ij} definiremos la magnitud φ' como el intervalo transcurrido entre el instante crítico y la activación de τ_{ij} correspondiente a la llegada del primer evento posterior al instante crítico (que se identificará con $p'=1$). Según se puede ver, ese valor es igual a:

$$\varphi' = T_i - \phi + \Phi_{ij} \tag{2}$$

Nótese que en esta expresión usamos el *offset* real, en lugar del reducido definido en el capítulo anterior. Como se puede ver, el valor de φ' puede ser mayor que el periodo, para $\phi<\Phi_{ij}$, incluso varias veces mayor, si el *offset* es suficientemente grande.

En las condiciones en que se crea el instante crítico, t_0 corresponde al instante en que llegó el primer evento que tiene pendiente la ejecución de la tarea τ_{ij} al comienzo del periodo de ocupación. Por tanto, esa primera instancia debe verificar simultáneamente las dos desigualdades siguientes:

$$t_0 + \Phi_{ij} + J_{ij} \geq t_c \tag{3}$$

y

$$t_0 - T_i + \Phi_{ij} + J_{ij} < t_c \tag{4}$$

Fijándonos en la figura, podemos ver que el instante crítico t_c se puede expresar como:

$$t_c = t_0 + n'_{ij}\, T_i + \Phi_{ij} - \varphi' \tag{5}$$

que al sustituir en las dos expresiones anteriores:

$$\begin{aligned} t_0 + \Phi_{ij} + J_{ij} &\geq t_0 + n'_{ij}\, T_i + \Phi_{ij} - \varphi' \\ t_0 - T_i + \Phi_{ij} + J_{ij} &< t_0 + n'_{ij}\, T_i + \Phi_{ij} - \varphi' \end{aligned} \tag{6}$$

conduce a:

$$n'_{ij} \leq \frac{J_{ij}+\varphi'}{T_i} \qquad y \qquad n'_{ij} > \frac{J_{ij}+\varphi'}{T_i} - 1 \tag{7}$$

Dado que n'_{ij} es un numero entero, las única solución de estas dos inecuaciones es:

$$n'_{ij} = \left\lfloor \frac{J_{ij}+\varphi'}{T_i} \right\rfloor \tag{8}$$

Si aplicamos este resultado al instante crítico t_c creado con la tarea τ_{ik}, en el cual el desfase es igual a $\phi = (\Phi_{ik}+J_{ik})\ mod\ T_i$, obtenemos el intervalo φ'_{ijk} transcurrido entre ese instante crítico y la activación de la tarea τ_{ij} correspondiente al primer evento ocurrido después de t_c:

$$\varphi'_{ijk} = T_i - (\Phi_{ik} + J_{ik}) \bmod T_i + \Phi_{ij} \tag{9}$$

que, sustituido en (8), nos da como resultado el valor del número n'_{ijk} de eventos que tienen pendiente la ejecución de la tarea τ_{ij} al comienzo del periodo de ocupación:

$$n'_{ijk} = \left\lfloor \frac{J_{ij}+\varphi'_{ijk}}{T_i} \right\rfloor \tag{10}$$

Siguiendo el esquema de numeración definido previamente, el último de esos eventos tiene como índice p'=0; por tanto, el primero de ellos se identificará con el valor $p'_{0,ijk}$ calculado mediante:

$$p'_{0,ijk} = - n'_{ijk} + 1 = - \left\lfloor \frac{J_{ij}+\varphi'_{ijk}}{T_i} \right\rfloor + 1 \tag{11}$$

La expresión del tiempo de finalización para la activación p' de la tarea τ_{ab} cuando estudiamos el instante crítico creado con τ_{ac} queda entonces en la forma:

$$w_{abc}(p') = B_{ab} + (p' - p'_{0,abc} + 1)\ C_{ab} + W_{ac}\left(\tau_{ab}, w_{abc}(p')\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, w_{abc}(p')\right) \tag{12}$$

y, puesto que la activación p'=1 se produce en el instante $t=\varphi'_{abc}$, el evento correspondiente a la activación p' de la tarea τ_{ab} ocurre en $\varphi'_{abc} + (p'-1)\ T_a - \Phi_{ab}$, y por tanto el tiempo de respuesta global, contado desde ese instante es:

$$R_{abc}(p') = w_{abc}(p') - \varphi'_{abc} - (p' - 1)T_a + \Phi_{ab} \tag{13}$$

Esta formulación es equivalente a la aplicada en el algoritmo APCOD, pero con el nuevo esquema de numeración, relativo al instante en que comienza la instancia (*p'*) a la que pertenece la activación analizada, en lugar de al instante donde se hubiera producido la activación de la tarea (*p*). El nuevo esquema nos permite mejorar el análisis mediante el siguiente teorema.

Teorema 5-1. Sea *p*' el índice de una activación de la tarea τ_{ab} dentro del periodo de ocupación de peor caso creado mediante la tarea τ_{ac}. La máxima interferencia que puede sufrir la ejecución de τ_{ab} por parte de una tarea τ_{aj} a la cual preceda está limitada por:

$$w_{\max} = (p' - p'_{0,ajc})\, C_{aj} \tag{14}$$

siendo $p'_{0,ajc}$ el índice de la primera activación de τ_{aj} dentro del periodo de ocupación.

Demostración. En el nuevo de esquema de numeración, dos tareas pertenecientes a una misma transacción y con igual índice *p*' de activación corresponden a la ejecución de la misma instancia. Si la tarea τ_{ab} precede a la tarea τ_{aj} quiere decir que la activación *p*' de esta última no puede producirse hasta que no haya finalizado la ejecución de la primera y, por tanto, no puede interferir en su ejecución. Por la misma razón, activaciones posteriores de τ_{aj} tampoco podrán interferir la ejecución *p*' de τ_{ab}. Evidentemente, activaciones de τ_{aj} correspondientes a instancias anteriores a *p*' no se ven afectadas por esa precedencia. Por tanto, en el análisis de peor caso de τ_{ab} es suficiente con considerar la interferencia de las activaciones producidas antes de la instancia *p*', para cada tarea τ_{aj} precedida por τ_{ab}. Dado que la primera activación de la tarea τ_{aj} retrasada hasta el instante crítico creado con τ_{ac} corresponde al índice $p'_{0,ajc}$, calculado como:

$$p'_{0,ajc} = -\left\lfloor \frac{J_{aj} + \varphi'_{ajc}}{T_a} \right\rfloor + 1 \tag{15}$$

$$\varphi'_{ajc} = T_a - (\Phi_{ac} + J_{ac}) \bmod T_a + \Phi_{aj} \tag{16}$$

el número máximo de activaciones dentro del periodo de ocupación correspondientes a instancias anteriores a la instancia *p*' es igual a $p'-p'_{0,ajc}$ y por tanto, la interferencia máxima será

$$w_{\max} = (p' - p'_{0,ajc})\, C_{aj} \tag{17}$$

Nótese que $p'-p'_{0,ajc} \geq 0$. Si desarrollamos la expresión (15), sustituyendo φ'_{ajc} por (16) y operamos la función *mod*, podemos llegar a:

$$p'_{0,ajc} = -\left\lfloor \frac{J_{aj}+\Phi_{aj}-J_{ac}-\Phi_{ac}}{T_a} \right\rfloor + \left\lfloor \frac{J_{ac}+\Phi_{ac}}{T_a} \right\rfloor \tag{18}$$

Si la tarea τ_{ab} precede a τ_{aj}, significa que:

$$\Phi_{aj} + J_{aj} > \Phi_{ab} + J_{ab} \tag{19}$$

lo cual conduce a

$$p'_{0,ajc} \leq p'_{0,abc} \quad , \quad \forall j > b \tag{20}$$

dado que el análisis comienza a partir de $p'_{0,abc}$, los valores de p' chequeados siempre son $p' \geq p'_{0,abc}$ por tanto, siempre se cumple $p'-p'_{0,ajc} \geq 0$ para cualquier tarea τ_{aj} con $j>a$.

□

Si no tenemos en cuenta las relaciones de precedencia en las activaciones de las tareas, la interferencia en la ejecución de τ_{ab} por tareas de su misma transacción viene dada por la expresión obtenida en el capítulo 4, esto es:

$$W_{ac}(\tau_{ab},t) = \sum_{\forall j \in hp_a(\tau_{ab})} \left(\left\lfloor \frac{J_{aj}+\varphi_{ajc}}{T_a} \right\rfloor + \left\lceil \frac{t-\varphi_{ajc}}{T_a} \right\rceil \right) C_{aj} \tag{21}$$

con

$$\varphi_{ajc} = T_a - (\Phi_{ac} + J_{ac} - \Phi_{aj}) \bmod T_a \tag{22}$$

Ahora bien, si consideramos el Teorema 5-1 en el análisis de la activación p', podemos limitar la interferencia producida por las tareas precedidas por la tarea τ_{ab}. Consecuentemente, obtendremos una expresión mejor acotada para la contribución al peor caso por tareas de su misma transacción:

$$\begin{aligned} W'_{ac}(\tau_{ab}, t, p') = & \sum_{\substack{\forall j \in hp_a(\tau_{ab}) \\ j<b}} \left(\left\lfloor \frac{J_{aj}+\varphi_{ajc}}{T_a} \right\rfloor + \left\lceil \frac{t-\varphi_{ajc}}{T_a} \right\rceil \right) C_{aj} + \\ & + \sum_{\substack{\forall j \in hp_a(\tau_{ab}) \\ j>b}} \min \left(p' - p'_{0,ajc} \, , \left\lfloor \frac{J_{aj}+\varphi_{ajc}}{T_a} \right\rfloor + \left\lceil \frac{t-\varphi_{ajc}}{T_a} \right\rceil \right) C_{aj} \end{aligned} \tag{23}$$

Que podemos utilizar para obtener el tiempo de finalización de la activación p' a partir de:

$$w_{abc}(p') = B_{ab} + (p' - p'_{0,abc} + 1)\, C_{ab} + W'_{ac}\left(\tau_{ab}, w_{abc}(p'), p'\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, w_{abc}(p')\right) \quad (24)$$

donde W'_{ac} se calcularía con la nueva ecuación (23) y W_i^* igual que en el algoritmo APCOD, mediante la expresión:

$$W_i^*(\tau_{ab}, t) = \max_{\forall k \in hp_i(\tau_{ab})} \left\{ \sum_{\forall j \in hp_i(\tau_{ab})} \left(\left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + \left\lceil \frac{t - \varphi_{ijk}}{T_i} \right\rceil \right) C_{ij} \right\} \quad (25)$$

con

$$\varphi_{ijk} = T_i - (\Phi_{ik} + J_{ik} - \Phi_{ij}) \bmod T_i \quad (26)$$

El rango de valores de p' que hay que chequear en el análisis es el comprendido entre $p'_{0,abc}$ y el primer valor de p' que verifica:

$$R_{abc}(p') \leq T_a + \Phi_{ab} \quad (27)$$

Y finalmente, el tiempo de respuesta global de peor caso se determina a partir de los tiempos de finalización según la expresión:

$$R_{ab} = \max_{\forall c \in hp_a(\tau_{ab}) \cup b} \left[\max_{p = p'_{0,abc} .. p'_{L,abc}} \left(w_{abc}(p') - \varphi'_{abc} - (p' - 1) T_a + \Phi_{ab} \right) \right] \quad (28)$$

Si utilizamos esta nueva formulación, que llamaremos AMPCOE (Análisis Mejorado de Peor Caso con *Offsets* Estáticos) sustituyendo al método APCOE dentro del algoritmo APCOD diseñado en el capítulo anterior, obtendremos mejores estimaciones de los tiempos de respuesta de peor caso en sistemas con relaciones de precedencia en la activación de sus tareas, tales como sistemas distribuidos o tareas que se suspenden. Este nuevo algoritmo, que llamaremos AMPCOD (Análisis Mejorado de Peor Caso con *Offsets* Dinámicos), se muestra en la figura adjunta.

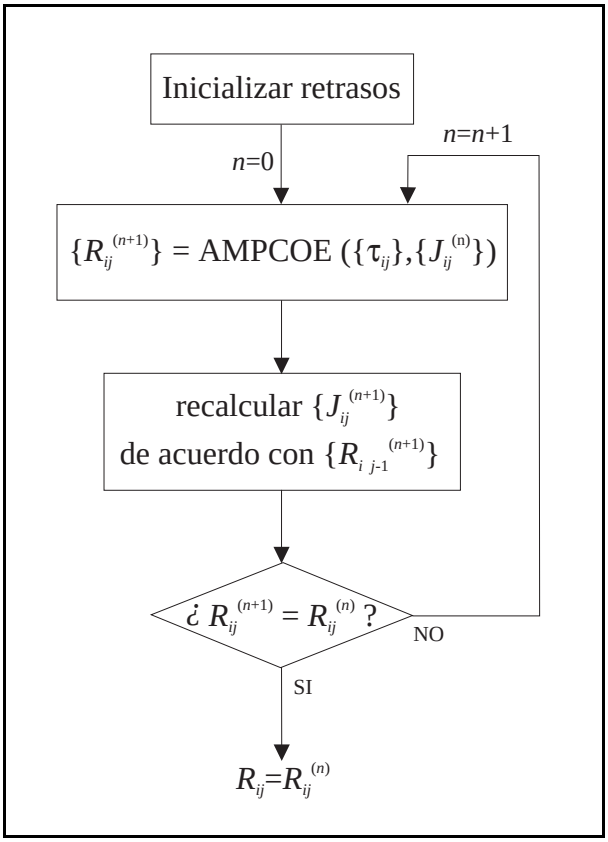


Figura 5-2. Algoritmo AMPCOD

5.2.1. Resultados de simulación

A efectos de evaluar las ventajas del algoritmo AMPCOD hemos comparado los resultados que obtiene frente al algoritmo APCOD, desarrollado en el capítulo anterior. Para ello se han aplicado ambas técnicas sobre conjuntos de tareas generados aleatoriamente en sistemas monoprocesadores y distribuidos. Los tiempos de ejecución y periodos de las tareas se han generado considerando varios parámetros, tales como las relaciones entre los máximos y los mínimos periodos de activación de tareas o la utilización de los recursos. Los plazos de ejecución asociados a las respuestas globales se hacen proporcionales a los periodos de activación y se asignan las prioridades a las tareas en función de esos plazos de ejecución, según el esquema *Deadline Monotonic*, sobre el que se introduce una pequeña componente aleatoria, variando la prioridad asignada a una tarea en cada transacción y procesador del sistema. Tal como veremos, esta variación hace que los sistemas simulados tengan límites de planificación menores que los generados en la sección 4.5, donde comparábamos nuestra técnica basada en *offsets* dinámicos (APCOD) con la técnica basada en tareas independientes de Tindell y Clark. Esta asignación nos permite, sin embargo, explorar las posibilidades del nuevo algoritmo mejorado, AMPCOD, frente al APCOD.

A continuación mostramos algunos de los resultados obtenidos en esas simulaciones. En el primer conjunto de gráficas (Figura 5-3 a Figura 5-5) se comparan los tiempos de respuesta obtenidos con los algoritmos de *offsets* dinámicos R_{APCOD} y R_{AMPCOD}, relativos a los obtenidos utilizando la técnica de Tindell y Clark para tareas independientes, R_{indep}. El eje de abcisas representa la utilización del procesador. Para cada utilización se genera un conjunto de tareas y se comparan los resultados obtenidos, mediante las relaciones $R_{\mathrm{indep}}/R_{\mathrm{APCOD}}$ y $R_{\mathrm{indep}}/R_{\mathrm{APCOD}}$. Cada gráfica representa los resultados para tres relaciones diferentes entre los máximos y los mínimos periodos de las transiciones, $T_{\mathrm{max}}/T_{\mathrm{min}}$ (10, 100 y 1000), representando cada punto el promedio de cinco simulaciones. Es de destacar el hecho de que en la asignación de prioridades elegida y para utilizaciones superiores al 50%, la técnica de Tindell y Clark no siempre converge, de manera que en las gráficas representaremos las utilizaciones hasta ese valor de utilización.

La Figura 5-3 muestra los resultados para sistemas monoprocesadores con 10 transacciones y 10 tareas por transacción, considerando nulos los tiempos de respuesta de

mejor caso de las tareas, y por tanto, con *offsets* iguales a 0. Se puede ver que los beneficios de las técnicas basadas en *offsets* aumentan según disminuye la relación T_{max}/T_{min}. A modo de ejemplo, se puede ver que para utilizaciones en torno al 40% se consiguen tiempos de respuesta R_{APCOD} entre 1.7 y 2.5 veces mejores que los R_{indep} obtenidos con la técnica de tareas independientes y aumenta hasta valores R_{AMPCOD}/R_{indep} entre 1.8 y 3.5 para la nueva técnica AMPCOD. Los resultados obtenidos suponiendo tiempos de ejecución de mejor caso iguales a los de peor caso (*offsets* mayores que cero) son bastante parecidos a estos.

La Figura 5-4 muestra los resultados para un caso similar, pero en un sistema multiprocesador constituido por 10 transacciones con 3 tareas cada una ejecutando en cada uno de los 4 procesadores (lo cual da un total de 12 tareas por transacción). Podemos ver que al disminuir el número de tareas por procesador en cada transacción disminuyen los beneficios de los algoritmos basados en *offsets*, obteniendo tiempos de respuesta entre 1.13 y 1.17 veces mejores para el algoritmo APCOD y entre 1.34 y 1.50 para el algoritmo AMPCOD, siempre para una utilización del 40%. Estos resultados mejoran (ver Figura 5-5) al considerar tiempos de ejecución de mejor caso iguales a los tiempos de ejecución de peor caso, y calcular el *offset* de cada tarea como a la suma de los tiempos de ejecución de ella misma y sus predecesoras en la misma transacción. En la Figura 5-5 podemos comprobar que las mejoras aumentan hasta márgenes entre 1.45 y 1.77 para R_{APCOD} y entre 1.71 y 2.28 para R_{AMPCOD}.

En el segundo conjunto de gráficas (Figura 5-6 y Figura 5-7) se comparan los resultados obtenidos al aplicar las técnicas de análisis para determinar los límites máximos de utilización de un determinado sistema, generado aleatoriamente. Este límite de utilización se calcula analizando el sistema con utilización baja y aumentado proporcionalmente los tiempos de ejecución de cada tarea hasta que el sistema deje de cumplir algún plazo, de forma que la máxima utilización planificable corresponderá al último conjunto de tareas que cumpla todos sus plazos. El análisis se ha realizado además para diferentes relaciones D_i/T_i entre los periodos de activación de las tareas y sus plazos de ejecución. En las gráficas se muestran las máximas utilizaciones encontradas cuando se aplica la técnica basada en tareas independientes y las basadas en tareas con *offsets* (algoritmos APCOD y AMPCOD) para tiempos de ejecución de mejor caso nulos. También se representa el límite de utilización obtenido con el algoritmo AMPCOD suponiendo tiempos de respuesta de mejor caso no nulos y *offsets* iguales a la suma de los tiempos de ejecución de mejor caso de cada tarea y sus precedentes.

La Figura 5-6 muestra los resultados de la simulación de un sistema con 4 procesadores, 5 transacciones y 20 tareas por transacción, distribuidas en los diferentes procesadores, para una relación T_{max}/T_{min}=100. Como se puede apreciar en ella, para relaciones entre los plazos de ejecución y periodos de activación superiores al valor D_i/T_i=3 se obtienen incrementos del orden del 5% en la utilización máxima. Por ejemplo, para un valor D_i/T_i=4 (que podría considerarse el valor lógico para un sistema con 4 procesadores) se consigue incrementar la utilización máxima desde un 15% hasta un 21%, lo cual supone una mejora relativa del 40% del método AMPCOD frente a tareas independientes. En la Figura 5-7 se estudia un sistema similar, pero con 12 tareas en cada transacción en lugar de 20. Vemos que para valores de D_i/T_i=3 y superiores, conseguimos un incremento en torno al 9% en la utilización máxima planificable. Para el caso de D_i/T_i=4 se incrementa este límite aproximadamente desde un 23% hasta un 31%, que significa una mejora relativa en torno al 35%. Nuevamente es interesante destacar que las mejoras son importantes aún considerando tiempos de respuesta de mejor caso iguales a cero. En estos dos ejemplos se consiguen, para D_i/T_i=4, mejoras relativas del 20% (de 15% a 18%) y del 21% (de 23% a 28%), respectivamente, realizando el análisis AMPCOD con *offsets* nulos.

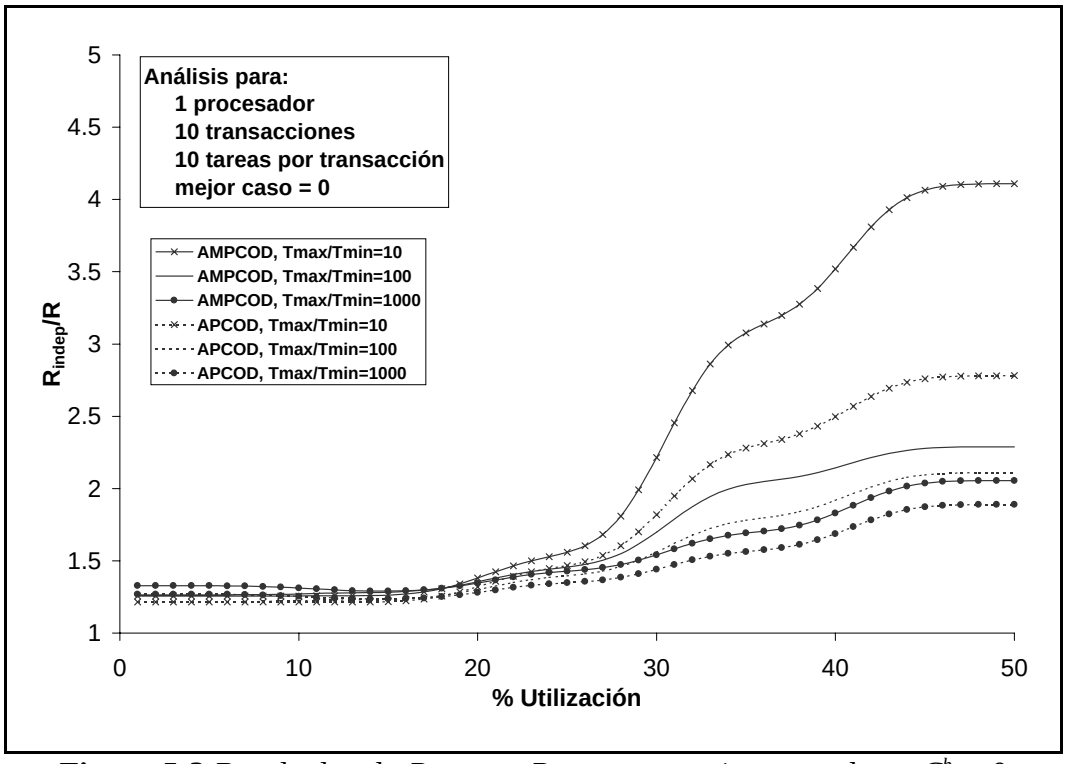


Figura 5-3 Resultados de R_{APCOD} y R_{AMPCOD}, para 1 procesador y C_i^b = 0

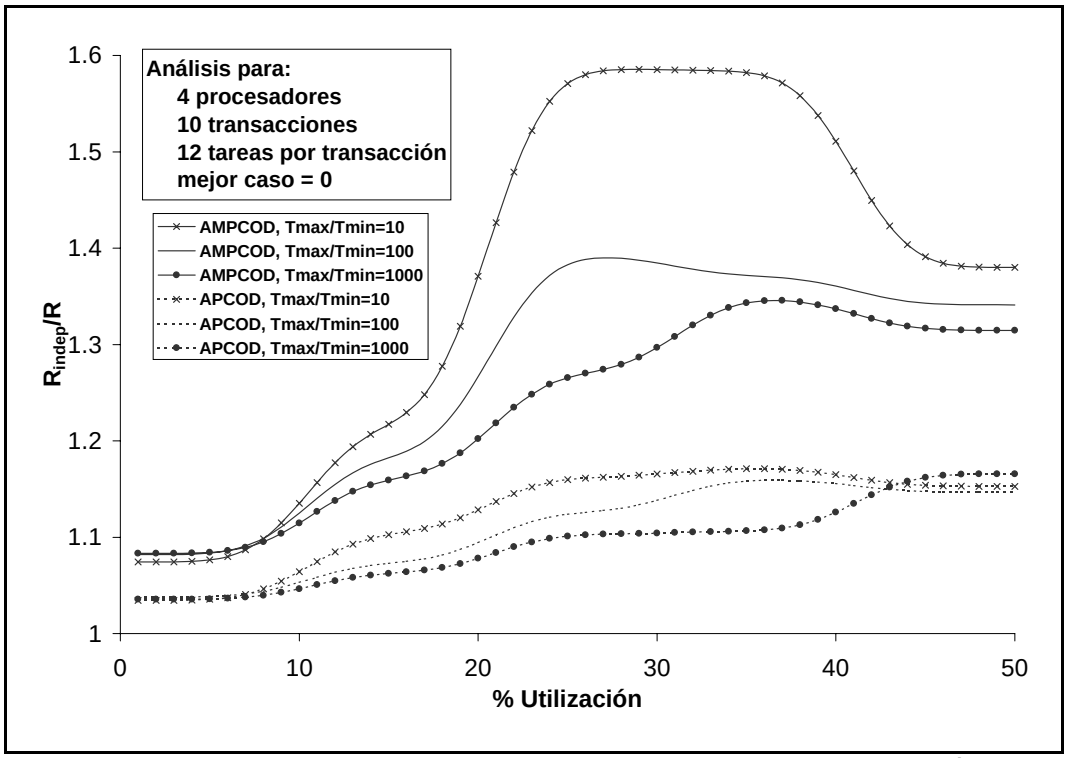


Figura 5-4 Resultados de R_{APCOD} y R_{AMPCOD}, para 4 procesadores y $C_i^b = 0$

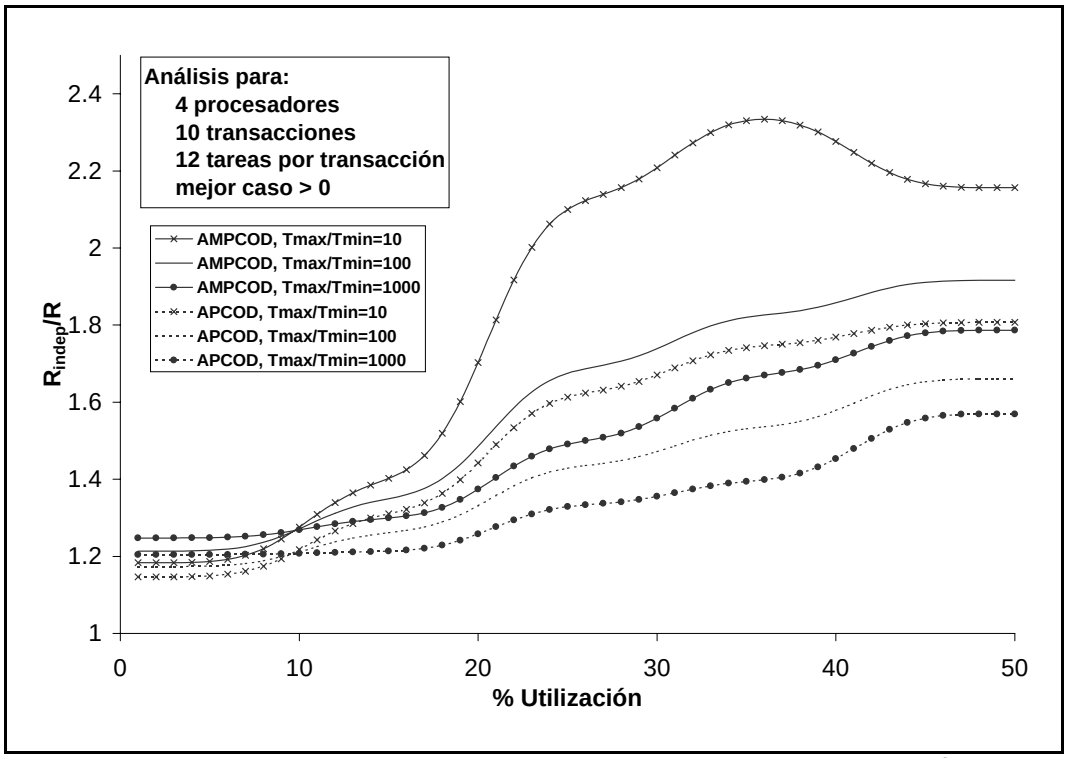


Figura 5-5 Resultados de $R_{\text{APCOD}}/R_{\text{AMPCOD}}$, para 4 procesadores y $C_i^b > 0$

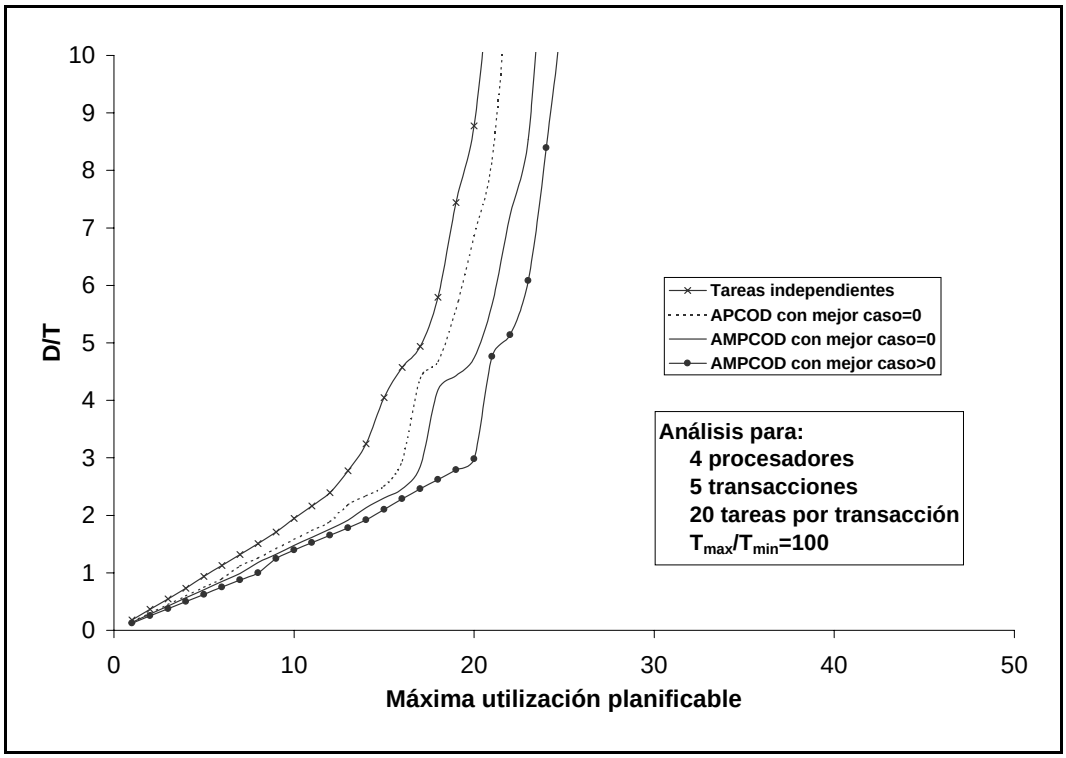


Figura 5-6 Máxima utilización planificable, 20 tareas por transacción

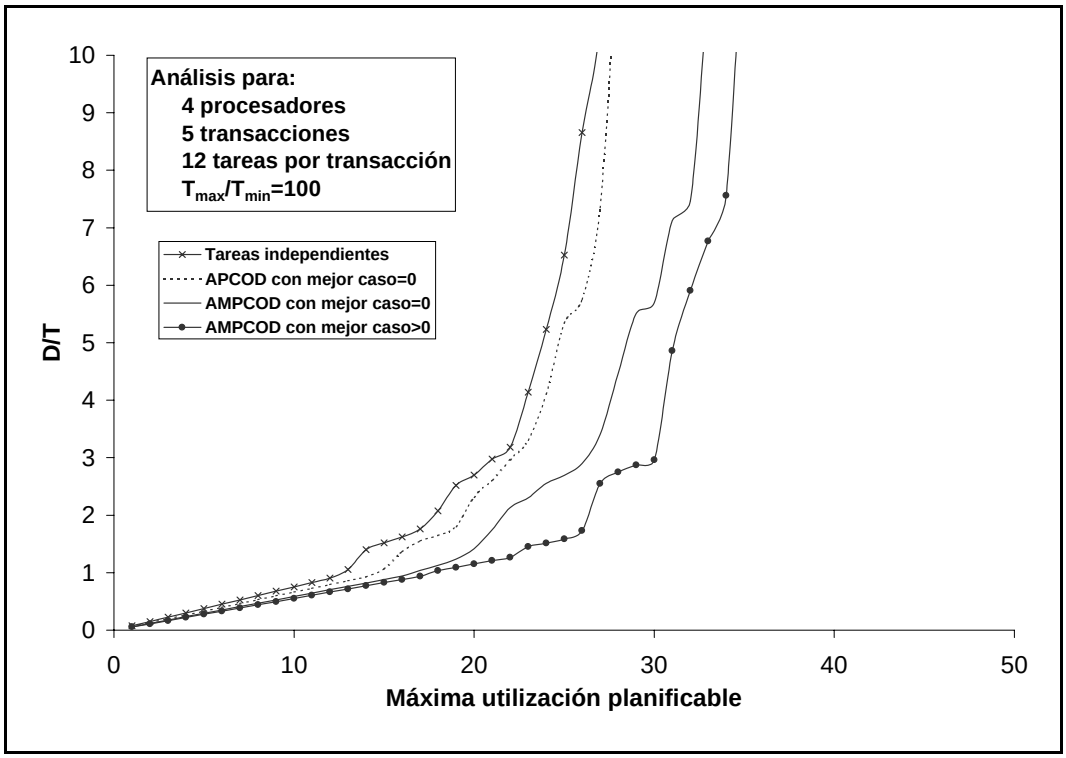


Figura 5-7 Máxima utilización planificable, 12 tareas por transacción

5.3. Perfil de prioridades y precedencia en transacciones

5.3.1. Prioridades y conflictos de activación

Hasta ahora, hemos desarrollado técnicas de análisis que tienen en cuenta las fases de activación de las tareas dentro de las transacciones, para mejorar las estimaciones de los tiempos de respuesta de peor caso en sistemas con relaciones de precedencia. Sin embargo, este análisis es susceptible de mejora si tenemos en cuenta además el perfil de prioridades dentro de una misma transacción, como pone de manifiesto el siguiente ejemplo.

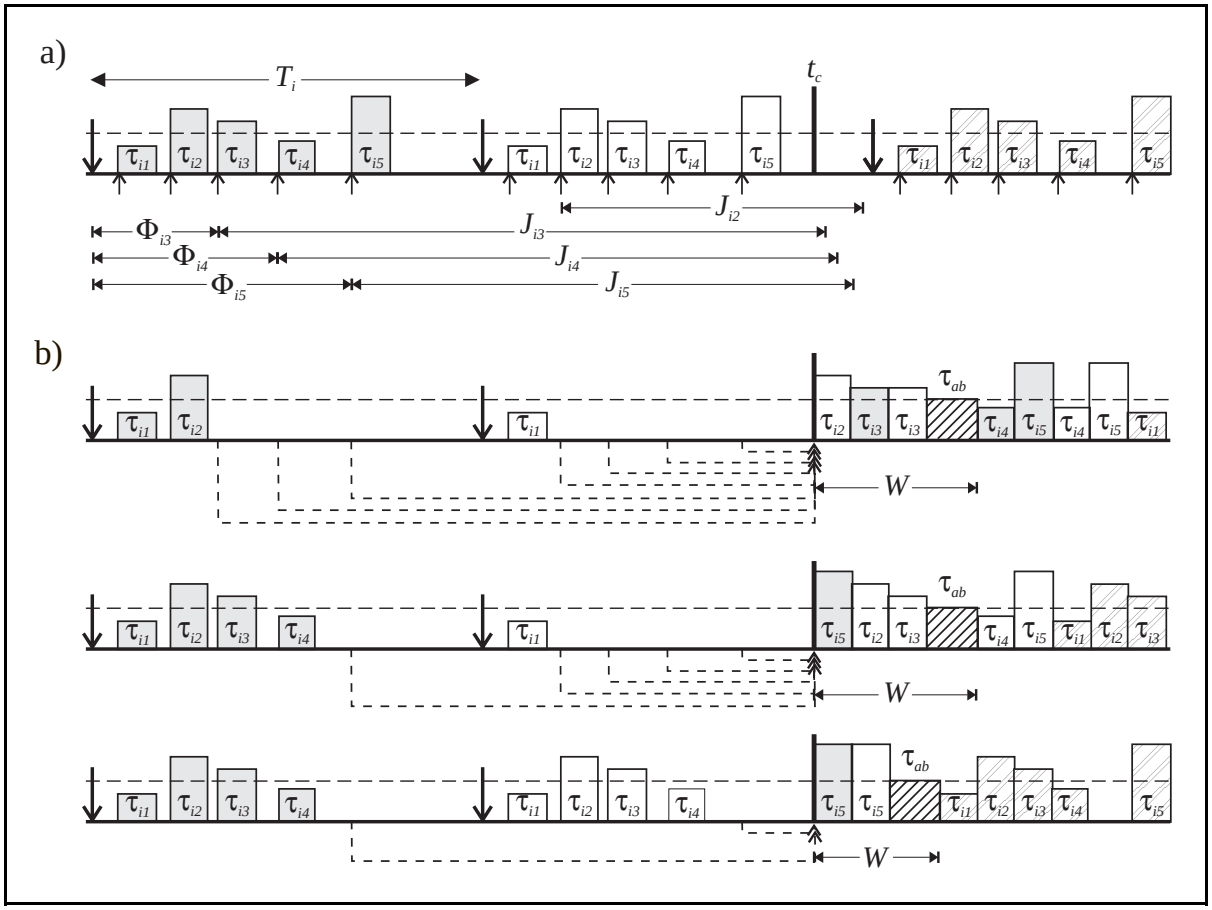


Figura 5-8. Posibles esquemas de ejecución de una transacción

La Figura 5-8-a muestra una transacción periódica Γ_i formada por 5 tareas de diferente prioridad ejecutando en un único procesador. Las flechas descendentes indican la ocurrencia de eventos y las flechas ascendentes los *offsets* de las tareas, Φ_{ij}. Cada recuadro representa la ejecución de una tarea, siendo su altura proporcional a la prioridad asignada a la misma.

Activaciones de tareas en diferentes instancias se representan con distintos rellenos. También se representan mediante líneas horizontales continuas los términos de retraso máximo, J_{ij}, correspondientes a la activación de cada tarea. Vamos a estudiar cuál es la contribución de tareas de Γ_i al peor caso de otra tarea τ_{ab} (cuya prioridad se indica con la línea discontinua horizontal), en el periodo de ocupación que comienza en el instante t_c.

En la Figura 5-8-b se representan tres posibles esquemas de ejecución, en función de los retrasos elegidos para cada activación, indicados en la figura mediante las líneas discontinuas inferiores. El primer esquema representa la ejecución del sistema cuando elegimos los términos de retraso tal y como construimos el peor caso hasta ahora, esto es, haciendo que las activaciones correspondientes a instancias anteriores al instante crítico se retrasen hasta coincidir con él. Si no consideramos precedencia, todas las activaciones de tareas con prioridad mayor que la de τ_{ab} interferirían en su ejecución, de forma que el tiempo de finalización sería, al menos, $W=2C_{i2}+3C_{i3}+3C_{i5}+C_{ab}$. Sin embargo, las relaciones de precedencia obligan a que la ejecución de la tarea τ_{i5} no comience hasta que no haya finalizado la ejecución de la tarea τ_{i4} precedente y, dado que la tarea τ_{i4} tiene menor prioridad que la asignada a τ_{ab} significa que τ_{i4} no puede comenzar a ejecutar hasta que no haya finalizado la ejecución de τ_{ab}, tal como representa la primera de las tres situaciones de la Figura 5-8-b. Esto quiere decir que el tiempo de finalización de la tarea τ_{ab} sería en ese caso igual a $W=C_{i2}+2C_{i3}+C_{ab}$. Esta situación representa un esquema de ejecución más realista, pero dificulta el análisis del peor caso, puesto que no podemos construir el instante crítico de la misma forma que antes. Podríamos elegir, por ejemplo, la fase de activación de τ_{i4} correspondiente a la primera instancia de forma que hubiera finalizado su ejecución antes del instante crítico y que así pudiera ejecutar τ_{i5} dentro del periodo de ocupación. El segundo esquema de la Figura 5-8-a muestra un ejemplo de este caso. La dificultad radica en que, debido a la precedencia, la activación de τ_{i3} perteneciente a la misma instancia también ha tenido que finalizar antes y, por tanto, no puede interferir la ejecución de la tarea τ_{ab}. Esto origina un conflicto a la hora de decidir cual de las dos situaciones debemos considerar para el peor caso, puesto que unas veces puede interesar una frente a otra, dependiendo de los tiempos de ejecución de cada tarea. Exactamente el mismo problema se presenta en la segunda instancia, ya que la ejecución de sus tareas τ_{i2} y τ_{i3} es incompatible con la ejecución de su tarea τ_{i5}, tal como se refleja en los dos últimos esquemas.

Definición 5-1. Dos tareas están en conflicto cuando la ejecución de una es incompatible con la ejecución de la otra, dentro del mismo periodo de ocupación. Sean τ_{ij} y τ_{ik} dos tareas con mayor o igual prioridad que la asignada a otra tarea τ_{ab}, y supongamos que τ_{ij} precede a τ_{ik}, $j<k$. Si en la transacción Γ_i existe una tarea intermedia τ_{il} ($j<l<k$) en el mismo procesador y de menor prioridad que τ_{ab}, entonces en un periodo de ocupación de τ_{ab} no pueden interferir simultáneamente tareas τ_{ij} y τ_{ik} activadas en la misma instancia. En ese caso, se dice que las activaciones de las tareas τ_{ij} y τ_{ik} están en conflicto.

En la primera de las dos instancias de la Figura 5-8, con índice p'=-1, hay un conflicto entre las activaciones de la tareas τ_{i3} y τ_{i5}. En la segunda (p'=0) hay un conflicto entre la activación de las tareas τ_{i2} y τ_{i3} y la activación de la tarea τ_{i5}. Nótese que la ejecución de las tareas τ_{i2} y τ_{i3} es compatible en el mismo periodo de ocupación.

Para calcular la máxima contribución de una transición Γ_i debemos determinar y resolver los posibles conflictos de activación en el periodo de ocupación que comienza en el instante crítico considerado. Resolver un conflicto significa elegir la ejecución de aquellas tareas que den como resultado la mayor interferencia sobre la tarea analizada, τ_{ab}. Es importante hacer notar que las activaciones correspondientes a instancias con índice $p' \geq 1$ se producen después del instante crítico y, por tanto, no hay capacidad para adelantar la fase de activación de sus tareas y resolver así posibles conflictos. Las tareas en ese caso deben ejecutar según el orden de precedencia, comenzando por la primera tarea de la transacción en el mismo procesador. En el ejemplo de la Figura 5-8 se puede observar que la primera tarea de la transacción, τ_{i1}, tiene menor prioridad que τ_{ab} y por lo tanto, ninguna tarea correspondiente a instancias posteriores al instante t_c puede interferir en el periodo de ocupación.

Para poder determinar y resolver los conflictos de activación, debemos caracterizar el perfil de prioridades de cada transacción. Desde el punto de vista del análisis de una tarea τ_{ab} clasificaremos las tareas de una transacción Γ_i en diferentes secciones, que denominaremos secciones *H*. Una sección *H* está compuesta por un conjunto de tareas contiguas (en el procesador donde ejecuta τ_{ab}) con prioridad mayor o igual que la asignada a τ_{ab}.

Definición 5-2. Dos tareas τ_{ij} y τ_{ik} de una transacción Γ_i pertenecen a la misma sección *H*, para el análisis de una tarea τ_{ab}, si ambas ejecutan en el mismo procesador que τ_{ab} con mayor o igual prioridad que la asignada a τ_{ab} y no existe ninguna tarea intermedia de la misma transacción ejecutando en el mismo procesador con prioridad menor que la de τ_{ab}. Identificaremos con $H_{ij}(\tau_{ab})$ a la sección a la cuál pertenece la tarea τ_{ij} y formada por el conjunto de tareas de la transacción Γ_i que verifique:

$$H_{ij}(\tau_{ab}) = \left\{ l \in \Gamma_i \,\middle|\, \left(\nexists \left[j \le x \le l \vee l \le x \le j \right] \mid proc(\tau_{ix}) = proc(\tau_{ab}) \wedge prio(\tau_{ix}) < prio(\tau_{ab}) \right) \right\} \quad (29)$$

Las tareas τ_{ij} y τ_{ik} pertenecerán a la misma sección si se cumple $H_{ij}(\tau_{ab})=H_{ik}(\tau_{ab})$. En el ejemplo de la Figura 5-8, desde el punto de vista de la ejecución de la tarea τ_{ab}, la transacción Γ_i está formada por dos secciones *H* diferentes, que son: la sección $H_{i2}(\tau_{ab})=H_{i3}(\tau_{ab})=\{\tau_{i2},\tau_{i3}\}$ y la sección $H_{i5}(\tau_{ab})=\{\tau_{i5}\}$. Nótese que según las definiciones 1 y 2, dos tareas estarán en conflicto si y sólo si ejecutan en el mismo procesador y pertenecen a distintas secciones *H*. Si en un periodo de ocupación determinado existe una instancia con varias tareas de diferentes secciones en conflicto, deberemos elegir para el peor caso aquellas tareas pertenecientes a una misma sección *H* cuya suma de tiempos de ejecución sea mayor. En el ejemplo de la Figura 5-8 tenemos dos instancias con tareas que se pueden retrasar hasta el instante t_c. La primera instancia tiene en conflicto las activaciones de las tareas τ_{i3} y τ_{i5}, pertenecientes a diferentes secciones, por tanto, elegiremos para el peor caso la que tenga mayor tiempo de ejecución, igual a $Max(C_{i3},C_{i5})$. La segunda instancia tiene la activación de τ_{i2} y τ_{i3} en conflicto con la activación de τ_{i5}, puesto que las dos primeras pertenecen a la misma sección *H*, diferente de la de τ_{i5}, de forma que para el peor caso elegiremos $Max(C_{i2}+C_{i3},C_{i5})$. La contribución total máxima al periodo de ocupación será la debida a ambas instancias, resultando una cota:

$$W = Max(C_{i3},C_{i5}) + Max(C_{i2}+C_{i3},C_{i5}) + C_{ab} \quad (30)$$

Se puede comprobar que este resultado es el mismo que se obtiene de calcular el máximo de los tres esquemas de ejecución considerados en la Figura 5-8-b. Recuérdese que si aplicáramos las técnicas descritas en el Capítulo 4, sin considerar las relaciones de precedencia, obtendríamos al menos un valor $W=2C_{i2}+3C_{i3}+3C_{i5}+C_{ab}$.

Para el cálculo del tiempo de respuesta de peor caso de una tarea τ_{ab} obtenemos las contribuciones de peor caso de cada transacción en el sistema. Tal como dedujimos en el

capítulo 4, obtenemos la contribución de peor caso de una transacción Γ_i estudiando todos los posibles instantes críticos creados con tareas de mayor o igual prioridad que τ_{ab}.

Vamos a centrarnos en el estudio del instante crítico creado con una tarea τ_{ik}. Para ello, distinguiremos entre las instancias iniciadas antes o después del instante crítico. Todas las instancias posteriores al instante crítico, con índices $p' \geq 1$, deben comenzar a ejecutar en orden de precedencia a partir de la primera tarea; por tanto, sólo pueden contribuir al periodo de ocupación las tareas pertenecientes a la primera sección *H*, que identificaremos como:

$$MP_i(\tau_{ab}) = \left\{ l \in \Gamma_i \;\middle|\; (\nexists\, x<l \mid proc(\tau_{ix})=proc(\tau_{ab}) \wedge prio(\tau_{ix})<prio(\tau_{ab})) \right\} \tag{31}$$

y, puesto que, tal como vimos en la sección anterior, la activación correspondiente a $p'=1$ se produce en el instante φ'_{ijk} calculado con (9), la expresión para esta interferencia es:

$$W_{ik}(\tau_{ab},t)\,\Big|_{p'>0} = \sum_{\forall j \in MP_i(\tau_{ab})} \left\lceil \frac{t-\varphi'_{ijk}}{T_i} \right\rceil_0 C_{ij} \tag{32}$$

al poder ser $\varphi'_{ijk} > T_i$ y no poder ser la contribución al tiempo de respuesta negativa, debemos considerar el máximo con 0, y por eso $\lceil x \rceil_0$.

Para la contribución al peor caso de instancias anteriores al instante crítico creado con τ_{ik} debemos encontrar y resolver los posibles conflictos de ejecución entre tareas de Γ_i. Para ello, vamos a construir la que llamaremos tabla de conflictos en el periodo de ocupación de anchura *t* para la tarea τ_{ab}. Cada fila de esta tabla representa un vector de las tareas pertenecientes a la transacción Γ_i, mientras que cada columna representa diferentes instancias de la transacción (con valores $p' \leq 0$). Cada celda (j,p') puede tomar dos valores diferentes: 0 si la activación p' de la tarea τ_{ij} no se ha producido dentro del intervalo $[0,t)$ o C_{ij}, el tiempo de ejecución de peor caso, si la activación p' de τ_{ij} se ha producido dentro del periodo de ocupación de anchura *t*.

Recuérdese que, en el instante crítico creado con la tarea τ_{ik}, tenemos un total de $n'_{ijk}=\lceil (J_{ij}+\varphi'_{ijk})/T_i \rceil$ instancias de la transacción Γ_i que tienen pendiente la ejecución de su correspondiente tarea τ_{ij}, con índices comprendidos entre $p'=p'_{0,ijk}=-n'_{ijk}+1$ y $p'=0$. De ellos, $n_{ijk}=\lceil (J_{ij}+\varphi_{ijk})/T_i \rceil$ activaciones de la tarea τ_{ij} se acumulan en $t=0$ (ver sección 4.2.2.) y el resto, $n'_{ijk}-n_{ijk}$, se activan después del instante crítico en instantes $t_{act}(p') = \varphi'_{ijk} + (p'-1)T_i$.

Si tenemos en cuenta la expresión (20), el número de instancias que tienen pendiente la ejecución de una tarea dada aumenta desde la primera tarea de la transacción hasta la última, de forma que debemos construir una tabla de como máximo N columnas y n'_{iNk} filas, siendo N el índice de la última tarea en el conjunto $hp_i(\tau_{ab})$ y n'_{iNk} el número de activaciones pendientes de esa última tarea. A continuación se muestra el algoritmo de creación de la tabla de conflictos para la transacción Γ_i, correspondiente al instante crítico creado con la tarea τ_{ik} para el análisis de una tarea τ_{ab}

```
Procedure CREA_TABLA_CONFLICTOS(τab, t, Γi,τik, out Tabla) is
begin
        Inicializa_Tabla
        For cada τij ∈ hpi(τab) loop
                For p' in p'0,ijk .. 0 loop
                        if t ≥ φ'ijk - (p'-1)Ti then Tabla(j,p') := Cij;
                        else Tabla(j,p') := 0
                end loop;
        end loop;
end CREA_TABLA_CONFLICTOS;
```

En la elaboración de la tabla podemos reducir el número de conflictos, ya que cuando estudiamos el instante crítico creado con la tarea τ_{ik} estamos eliminando implícitamente la posibilidad de algunos conflictos. Aquellas tareas precedidas por τ_{ik} que entren en conflicto con ella no pueden ejecutar, puesto que ya hemos elegido la ejecución de τ_{ik}, al menos para el evento con el que se crea el instante crítico, con índice igual a $p'_{0,ikk}$. Esto quiere decir que para las instancias con índice mayor o igual que $p'_{0,ikk}$ podemos eliminar las activaciones de tareas precedidas por τ_{ik} que no pertenezcan a su misma sección H, quedando el algoritmo:

```
Procedure CREA_TABLA_CONFLICTOS(τab, t, Γi,τik, out Tabla) is
begin
        Inicializa_Tabla;
        For cada τij ∈ hpi(τab) loop
                For p' in p'0,ijk .. 0 loop
                        if t ≥ φ'ijk - (p'-1)Ti then Tabla(j,p') := Cij;
                        else Tabla(j,p') := 0;
                        if p'≥p'0,ikk and j>k and Hij(τab)≠Hik(τab) then Tabla(j,p') := 0;
                end loop;
        end loop;
end CREA_TABLA_CONFLICTOS;
```

Para resolver los conflictos de activación debemos recorrer la tabla por filas (instancias), obteniendo para cada una el máximo tiempo de ejecución requerido al procesador por una sección *H*. El tiempo de ejecución requerido por cada sección *H* en la instancia *p*' se calcula sumando, en la fila *p*', las columnas correspondientes a las tareas pertenecientes a esa sección. La interferencia total se obtendrá sumando los valores obtenidos para cada fila.

```
Function RESUELVE_CONFLICTOS(τab, t, Γi,τik) return tiempo is
begin
        CREA_TABLA_CONFLICTOS(τab, t, Γi,τik, Tabla);
        Total:=0;
        For p' in p'0,iNk .. 0 loop
                maxima_seccion:=0; suma:=0;
                For cada τij en Γi loop
                        if prioridad(τij)<prioridad(τab) and procesador(τij)=procesador(τab) then
                                if suma>maxima_seccion then maxima_seccion:=suma;
                                suma:=0;
                        else suma:=suma+Tabla(j,p');
                        endif;
                Total:=Total+maxima_seccion;
        end loop;
        return Total;
end RESUELVE_CONFLICTOS;
```

Realmente, la tabla de conflictos no es necesaria, ya que podemos integrar los dos últimos algoritmos en uno único que obtiene los términos que intervienen en cada conflicto y simultáneamente va calculando la secciones máxima para cada activación. Mantendremos el concepto de tabla de conflictos para facilitar la comprensión de la técnica de análisis. El algoritmo integrado se muestra a continuación:

```
Function RESUELVE_CONFLICTOS(τab, t, Γi,τik) return tiempo is
begin
    Total:=0;
    For p' in p'0,iNk .. 0 loop
        maxima_seccion:=0; suma:=0;
        For cada τij ∈ Γi loop
            if procesador(τij)=procesador(τab) then
                if prioridad(τij)≥prioridad(τab) then
                    Celda:=0;
                    if p' ≥p'0,ijk and t ≥ φ'ijk - (p'-1)Ti then Celda:=Cij;
                    if p'≥p'0,ikk and j>k and Hij(τab)≠Hij(τab) then Celda:=0;
                    suma:=suma+Celda;
                else
                    if suma>maxima_seccion then maxima_seccion:=suma;
                    suma:=0;
                end if;
            end if;
        end loop;
        Total:=Total+maxima_seccion;
    end loop;
    return Total;
end RESUELVE_CONFLICTOS;
```

Considerando la resolución de conflictos en instancias anteriores al instante crítico con $p'\leq 0$ y la ecuación (32) para $p'>0$, obtenemos la interferencia debida a tareas de la transacción Γ_i en el periodo de ocupación de anchura t que comienza en el instante crítico creado con τ_{ik}

$$W_{ik}(\tau_{ab},t) = RESUELVE\_CONFLICTOS(\tau_{ab},t,\Gamma_i,\tau_{ik}) + \sum_{\forall j \in MP_i(\tau_{ab})} \left\lceil \frac{t-\varphi'_{ijk}}{T_i} \right\rceil_0 C_{ij} \qquad (33)$$

A partir de esta ecuación podemos derivar la función $W_i^*(\tau_{ab},t)$, que obtiene una cota superior de la interferencia máxima debida a tareas de la transacción Γ_i

$$W_i^*(\tau_{ab},t) = \max_{\forall k \in hp_i(\tau_{ab})} W_{ik}(\tau_{ab},t) \qquad (34)$$

A continuación ilustraremos esta técnica aplicándola al ejemplo mostrado en la Figura 5-8. Analizaremos las posibles situaciones de peor caso construidas con cada una de las tareas de prioridad mayor que τ_{ab}. En las Figuras 5-4 a 5-6 se muestran cada uno de los tres escenarios construidos, según el instante crítico se haya creado con τ_{i2}, τ_{i3} o τ_{i5}.

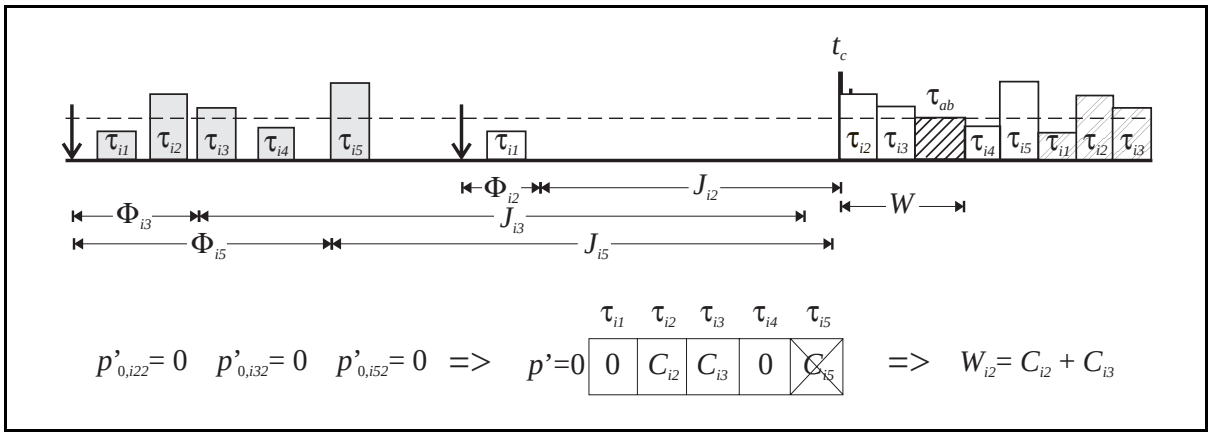


Figura 5-9. Instante crítico creado con la tarea τ_{i2}, para el ejemplo en la Figura 5-8

En la Figura 5-9 se estudia el caso del instante crítico creado cuando activamos la tarea τ_{i2} sufriendo su máximo retraso posible. Como se puede observar, en el instante t_c estaría pendiente la ejecución de las tareas τ_{i2}, τ_{i3} y τ_{i5} correspondientes a la instancia $p'=0$ y así creamos la tabla de conflictos, mostrada en la parte inferior de la figura. Sin embargo, de la tabla podemos eliminar la celda correspondiente a la tarea τ_{i5}, puesto que el instante crítico creado sería incompatible con su ejecución, tal como indicamos en el algoritmo de creación de la tabla de conflictos. De hecho, no existe ningún conflicto de activación, ya que el único escenario posible es el que se muestra en la figura. La resolución de los conflictos se muestra a la derecha de la tabla: dado que las tareas τ_{i2} y τ_{i3} pertenecen a la misma sección H, el resultado en la única instancia ($p'=0$) de la tabla es igual a $C_{i2}+C_{i3}$. El conjunto de tareas MP_i está vacío, puesto que la primera tarea de la transacción ejecuta en el mismo procesador y tiene prioridad menor que la asignada a τ_{ab}, y la contribución de instancias posteriores al instante crítico, con índices $p'>0$, será nula. La posible contribución al peor caso de τ_{ab} en el periodo de ocupación estudiado será en definitiva $W_{i2}=C_{i2}+C_{i3}$.

La Figura 5-10 muestra el instante crítico construido con τ_{i3}. En este caso, la instancia $p'=0$ tiene pendiente la ejecución de la tareas $\tau_{i2,}$ τ_{i3} y τ_{i5}, pero también están pendientes las ejecuciones de τ_{i3} y τ_{i5} correspondientes a la instancia anterior ($p'=-1$). La tabla tendrá por tanto dos filas, tal como se muestra en la figura, y en la cual se eliminan las ejecuciones incompatibles con el instante crítico. Para cada uno de esas dos instancias se calcula la sección máxima y se suman. La cota estimada en este caso para la contribución al periodo de ocupación es igual a $W_{i3}=C_{i3}+C_{i2}+C_{i3}$.

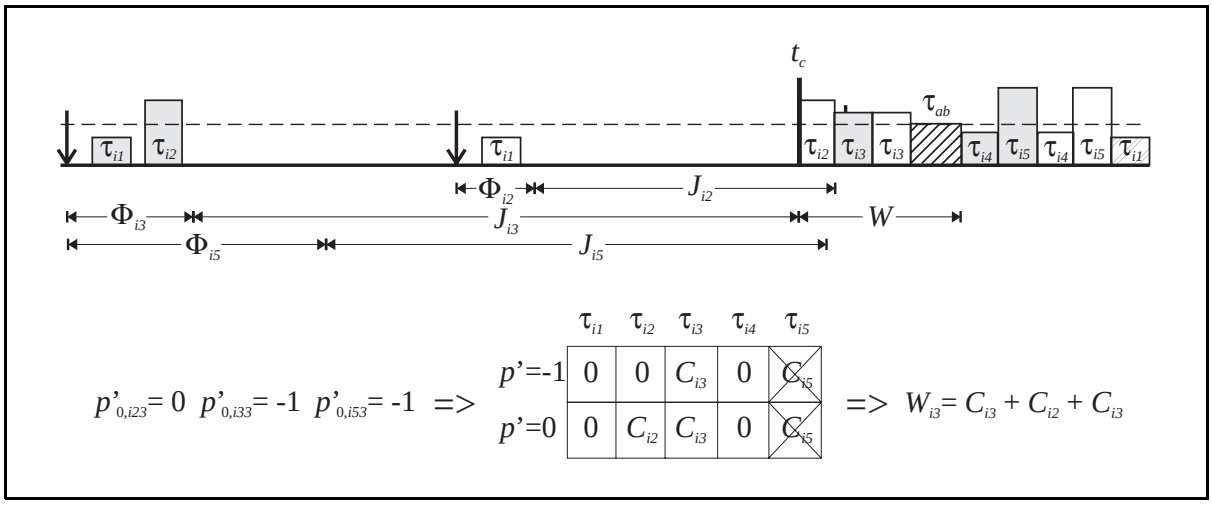


Figura 5-10. Instante crítico creado con la tarea τ_{i3}, para el ejemplo en la Figura 5-8

Por último, la Figura 5-11 muestra los dos posibles escenarios que se pueden presentar cuando el instante crítico se crea con la tarea τ_{i5}. En el instante t_c hay pendientes de ejecución una instancia de las tareas τ_{i2} y τ_{i3} y dos instancias de la tarea τ_{i5}. Nótese que en este caso, al aplicar la reducción como en los casos anteriores no eliminamos ninguna activación de la tabla. Además ahora nos aparece un conflicto en la tabla (en los dos casos anteriores no había ninguno efectivo). La instancia p'=-1 contribuye con C_{i5}, pero en la instancia p'=0 tenemos activaciones de dos secciones diferentes que pueden contribuir al periodo de ocupación. La resolución de este conflicto obtiene una contribución, para esta instancia, igual al máximo de

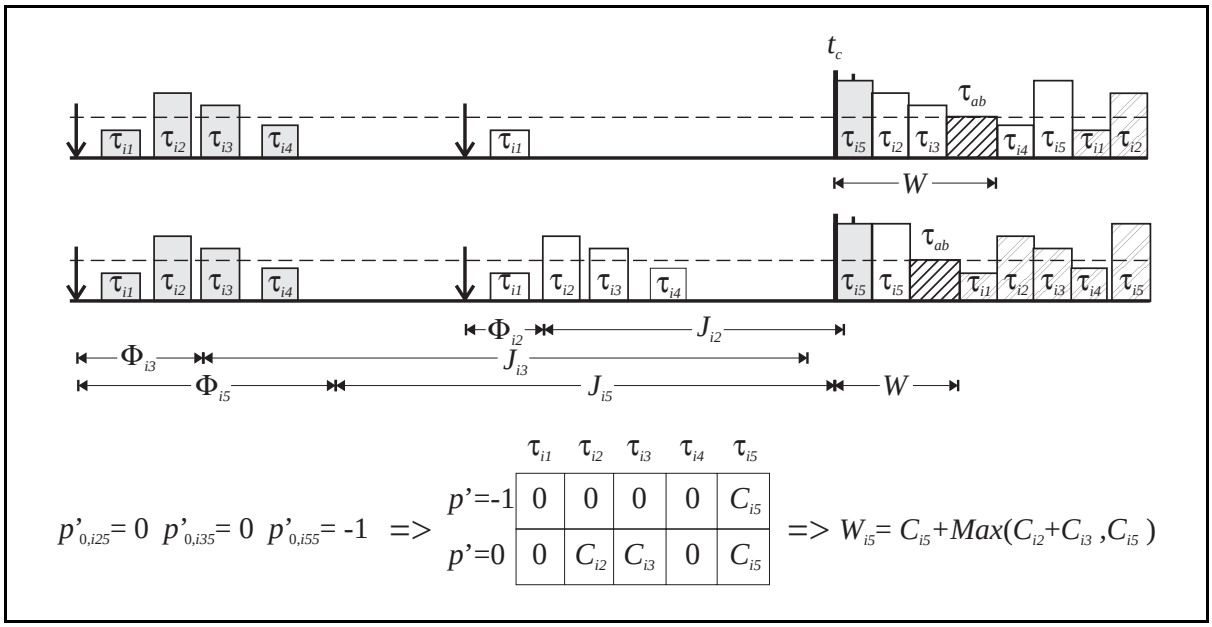


Figura 5-11. Instante crítico creado con la tarea τ_{i5}, para el ejemplo en la Figura 5-8

la contribución de las dos secciones, $Max\ (\ C_{i2}+C_{i3}\ ,\ C_{i5}\ )$. La contribución total en el instante crítico creado con la tarea τ_{i5} resulta $W_{i5} = C_{i5} + Max\ (\ C_{i2}+C_{i3}\ ,\ C_{i5}\ )$.

La contribución máxima de tareas de la transacción Γ_i al tiempo de respuesta de peor caso de la tarea τ_{ab} se puede acotar entonces con la expresión:

$$W_i^* = Max\ (W_{i2}, W_{i3}, W_{i5}) = Max\ \left(\ C_{i2}+2C_{i3}\ ,\ C_{i5}+Max(C_{i2}+C_{i3}, C_{i5})\ \right) \qquad (35)$$

se puede comprobar que esta expresión es equivalente a la obtenida en la ecuación (30), obtenida por inspección del ejemplo de la Figura 5-8.

En sistemas de tiempo real con relaciones de precedencia, en los que una tarea se activa al finalizar la anterior, es posible reducir el pesimismo en el análisis, reduciendo además el número de casos a analizar, si tenemos en cuenta el siguiente lema:

Lema 5-1. Sean τ_{ij} y τ_{ij+1} dos tareas consecutivas en la misma transacción Γ_i, de forma que la tarea τ_{ij+1} se activa en cuanto finaliza la ejecución de la tarea τ_{ij}. Si ambas tareas están ubicadas en el mismo procesador no puede existir un periodo de ocupación de nivel *pr* = mínimo(prio(τ_{ij}) , prio(τ_{ij+1})) o inferior cuyo comienzo coincida con la activación de la tarea τ_{ij+1}.

Demostración. Supongamos que existiera un periodo de ocupación de nivel *pr* o inferior iniciado en un instante *t* en el cuál se active la tarea τ_{ij+1}. Puesto que la tarea τ_{ij+1} se activa inmediatamente después de la tarea τ_{ij}, justo en el instante *t* habrá finalizado la ejecución de esa tarea τ_{ij} y, dado que ejecutan en el mismo procesador, significa que el comienzo del periodo de ocupación no puede ser el instante *t*, sino que se remonta, al menos, hasta el instante de activación de la tarea τ_{ij}. Nótese que este razonamiento no sería válido si la tarea τ_{ij+1} tuviera mayor prioridad que la tarea τ_{ij} y estuviéramos estudiando un periodo de ocupación de nivel intermedio.
□

Este lema nos permite reducir el número de casos a contemplar para el análisis de una tarea τ_{ab}. Si en una transacción Γ_i dos tareas consecutivas ejecutan en el mismo procesador que τ_{ab} y tiene asignadas prioridades mayores o iguales que la asignada a τ_{ab}, entonces es suficiente con analizar el periodo de ocupación que comience en el posible instante crítico creado con

la primera de las tareas. Extendiendo este razonamiento, sólo se necesita analizar los posibles instantes críticos creados con las tareas pertenecientes al conjunto:

$$XP_i(\tau_{ab}) = \left\{ l \in \Gamma_i \;\middle|\; \begin{array}{c} proc(\tau_{il})=proc(\tau_{ab}) \wedge prio(\tau_{il})\geq prio(\tau_{ab}) \\ \wedge \\ proc(\tau_{il-1})\neq proc(\tau_{ab}) \vee prio(\tau_{il-1})<prio(\tau_{ab}) \end{array} \right\} \quad (36)$$

esto es, el conjunto de tareas de la transacción Γ_i que ejecutan en el mismo procesador que la tarea τ_{ab} con prioridad mayor o igual que la asignada a τ_{ab} y cuya predecesora, si existe, no se encuentre en esas mismas condiciones. La expresión para obtener una cota superior de la interferencia máxima debida a tareas de la transacción Γ_i queda en la forma:

$$W_i^*(\tau_{ab},t) = \max_{\forall k \in XP_i(\tau_{ab})} W_{ik}(\tau_{ab},t) \quad (37)$$

donde $W_{ik}(\tau_{ab},t)$ corresponde a la ecuación (33). Puesto que el conjunto $XP_i(\tau_{ab})$ puede contener menos tareas que el conjunto $hp_i(\tau_{ab})$ definido previamente, eliminamos el análisis de periodos de ocupación no posibles, reduciendo, por tanto, el pesimismo introducido en el análisis y el número de casos a analizar, haciendo el algoritmo más rápido.

5.3.2. Aplicación a la transacción analizada

Vamos a aplicar las consideraciones sobre el perfil de prioridades en la transacción a la cual pertenece la tarea analizada, τ_{ab}. La metodología de análisis estima cotas superiores de los tiempos respuesta de una tarea estudiando los posibles instantes críticos creados con cada una de las tareas de la transacción Γ_a con prioridad mayor o igual que la de la tarea bajo análisis, incluida ella misma. En esta sección derivaremos la expresión para la contribución de tareas de Γ_a al peor caso de una tarea τ_{ab}, en el periodo de ocupación que comienza en el instante de máximo retraso en la activación de una tarea τ_{ac}.

En la Figura 5-12-a se muestra un ejemplo de una transacción Γ_a con cuatro tareas ejecutando en un procesador (supondremos el resto de tareas ejecutando en otros procesadores). Nuevamente, las flechas descendentes indican la ocurrencia de eventos y las flechas ascendentes los *offsets* de las tareas. Cada recuadro representa la ejecución de una tarea, siendo su altura proporcional a la prioridad asignada a la misma. Activaciones de tareas en diferentes instancias se representan con distintos rellenos y con líneas continuas los

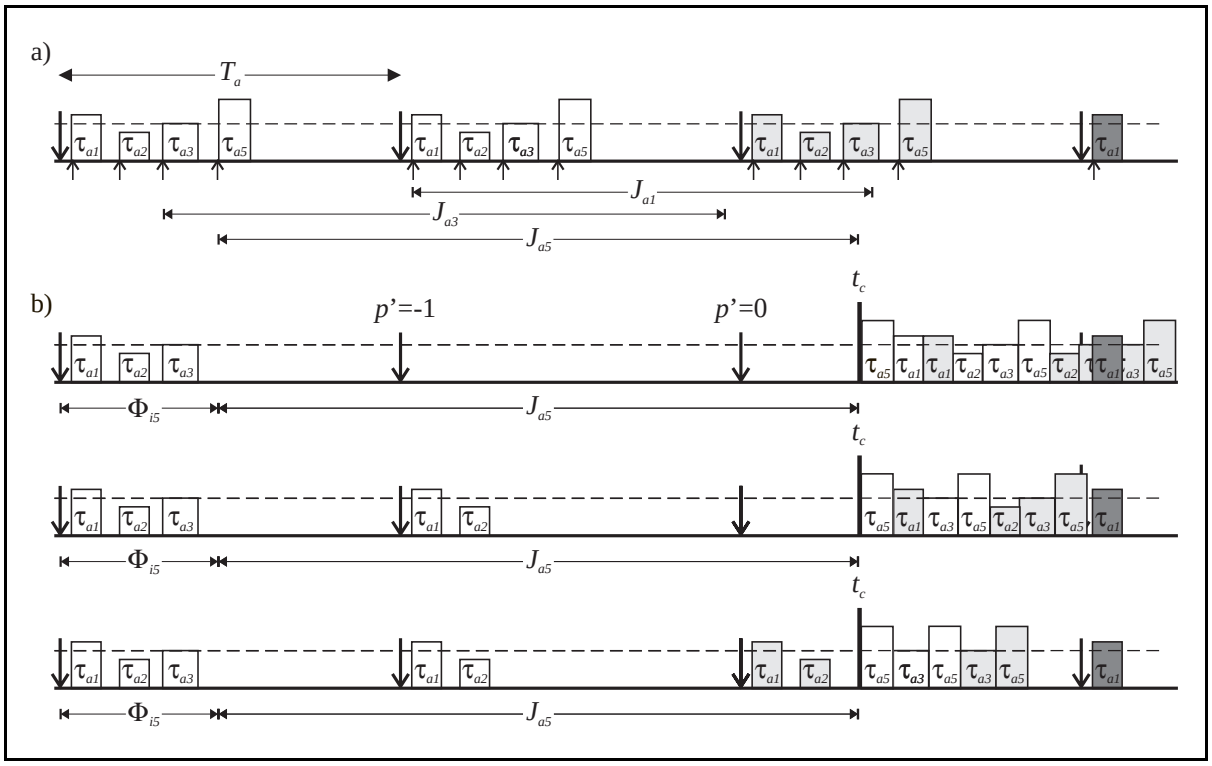


Figura 5-12. Instantes críticos creados con τ_{a5} en la transacción analizada

retrasos máximos de activación. Vamos a considerar el instante crítico creado mediante la tarea τ_{a5} para el análisis de la tarea τ_{a3}. En esas condiciones, la transacción tiene dos secciones *H* diferentes: una formada por la tarea τ_{a1} y otra formada por las tareas τ_{a3} y τ_{a5}. Como se puede apreciar, en ese instante crítico hay tres instancias que tienen pendiente la ejecución de alguna de sus tareas, por lo que debemos tener en cuenta los conflictos de activación entre tareas de diferentes secciones *H*. En la parte inferior se representan los tres escenarios de ejecución posibles en función de las tareas elegidas para ejecutar. El primer escenario representa la ejecución cuando se retrasan hasta el instante crítico las dos activaciones pendientes de la tarea τ_{a1}, con índices p'=-1 y p'=0. El segundo representa la ejecución cuando se retrasa sólo el correspondiente a p'=0 y el tercero cuando se no se retrasa ninguno de los dos (o no se retrasan lo suficiente para que ocurran después del instante crítico).

Si bien los tres escenarios de ejecución comienzan con un posible instante crítico creado con τ_{a5}, no todos son válidos para el análisis de la tarea τ_{a3}. Fijémonos en el primer escenario creado, en el que retrasan las activaciones pendiente de la tarea τ_{a1} para que ejecuten dentro del periodo de ocupación. Dado que las tareas τ_{a1} y τ_{a3} pertenecen a

diferentes secciones *H*, el periodo de ocupación correspondiente no contiene ninguna ejecución de la tarea τ_{a3} y, por tanto, no es útil para su análisis. Los otros dos escenarios sin embargo sí contienen ejecuciones de la tarea τ_{a3} y deben ser analizados.

Esto significa que podemos reducir el número de posibles casos a considerar en el análisis. Tal y como dedujimos en la sección anterior, cuando consideramos el instante crítico creado con una tarea τ_{ac} debemos tener en cuenta que ya se ha forzado la ejecución de esa tarea τ_{ac} al menos para la instancia con el que se creó el instante crítico. Para el análisis de la tarea τ_{ab} estamos forzando además otra situación: que la activación analizada esté en el periodo de ocupación construido. Fijémonos nuevamente en el ejemplo de la Figura 5-12. El periodo de ocupación representado en el tercer escenario contiene la ejecución de la tarea τ_{a3} correspondientes a las instancias p'=-1 y p'=0. Sin embargo, el segundo escenario contiene solamente la ejecución de τ_{a3} perteneciente a la instancia p'=-1, puesto que para la instancia p'=0 se eligió la ejecución de la tarea τ_{a1} en el periodo de ocupación. Esto quiere decir que en el análisis de la instancia p'=-1 debemos considerar los dos escenarios, pero en el análisis correspondiente a la instancia p'=0 es suficiente con analizar el último escenario, puesto que los otros son incompatibles con su ejecución en el periodo de ocupación.

Por tanto, podemos ampliar las reglas aplicables para la reducción de la tabla de conflictos en el análisis de la tarea τ_{ab} correspondiente a la instancia p'_{ab} en el periodo de ocupación creado con la tarea τ_{ac}. Por un lado, la debida a la creación del instante crítico, tal como hicimos en la sección 5.3.1. Por otro lado, los conflictos existentes en la instancia p'_{ab} y anteriores se habrán tenido que resolver de forma compatible con la ejecución, dentro del periodo de ocupación considerado, de la activación p'_{ab} de τ_{ab}. En el ejemplo, el análisis de la activación p'_{a3} =0 sólo es necesario si en las instancias p'=-1 y p'=0 no se ha elegido para ejecutar la tarea τ_{a1}. Estas reglas de reducción son:

1ª regla de reducción (debida a la creación del instante crítico). En la tabla de conflictos se pueden eliminar las activaciones correspondientes a instancias $p'_{0,acc}$ y posteriores que entren en conflicto con la tarea τ_{ac}. Es decir, se pueden anular las celdas (j,p') que verifiquen la condición:

$$p' \geq p'_{0,acc} \quad \wedge \quad j > c \quad \wedge \quad H_{aj}(\tau_{ab}) \neq H_{ac}(\tau_{ab}) \tag{38}$$

2ª regla de reducción (debida a la ejecución de la instancia p'_{ab} de τ_{ab} en el periodo de ocupación). Se pueden eliminar los conflictos en instancias p'_{ab} y anteriores, incompatibles con la ejecución de τ_{ab}. Esto es:

$$p' \le p'_{ab} \quad \wedge \quad j<b \quad \wedge \quad H_{aj}(\tau_{ab}) \neq H_{ab}(\tau_{ab}) \tag{39}$$

Adicionalmente, podemos considerar una regla de reducción más en el análisis de la activación p'_{ab}, que ya aplicamos en el teorema 5-1 de la sección 5.2. Las tareas precedidas por τ_{ab} no pueden interferir su ejecución si corresponden a instancias posteriores a la analizada. Es decir:

3ª regla de reducción (debido al análisis de la instancia p'_{ab} de τ_{ab}). De la tabla de conflictos se pueden eliminar las celdas correspondientes a tareas precedidas por τ_{ab} en la misma o posteriores instancias. Tampoco se tienen que considerar las activaciones posteriores de la propia tarea τ_{ab}. Por tanto, se pueden anular las celdas (j,p') que cumplan:

$$(p' \ge p'_{ab} \quad \wedge \quad j>b) \quad \vee \quad (p' > p'_{ab} \quad \wedge \quad j=b) \tag{40}$$

Con todas estas consideraciones, la tabla de conflictos en la transacción Γ_a para el análisis de la instancia p'_{ab} de τ_{ab} se puede crear según el algoritmo descrito a continuación. El algoritmo es similar al utilizado para crear la tabla de conflictos en la sección anterior, pero añadiendo las nuevas reglas de reducción. El proceso de resolución de los conflictos es similar al de la función RESUELVE_CONFLICTOS descrita previamente en la sección 5.3.1, pero utilizando este último algoritmo para crear la tabla.

```
Procedure CREA_TABLA_CONFLICTOS_EN_Γa(τab, p'ab, t, Γi,τac, out Tabla) is
begin
        Inicializa_Tabla;
        For cada τaj ∈ hpa(τab) loop
                For p' in p'0,ajc .. 0 loop
                        if t ≥ φ'ajc - (p'-1)Ta then Tabla(j,p') := Caj;
                        else Tabla(j,p') := 0;
                        if p'≥p'0,acc and j>c and Haj(τab)≠Hac(τab) then Tabla(j,p') := 0;
                        if p'≤p'ab and j<b and Haj(τab)≠Hab(τab) then Tabla(j,p') := 0;
                        if (p'≥p'ab and j>b) or (p'>p'ab and j=b) then Tabla(j,p') := 0;
                end loop;
        end loop;
end CREA_TABLA_CONFLICTOS_EN_Γa;
```

Tampoco en este caso es necesaria la tabla de conflictos, si integramos los procesos de creación y resolución de conflictos en el algoritmo siguiente:

```
Function RESUELVE_CONFLICTOS_Γa(τab, p'ab, t, Γi,τik) return tiempo is
begin
        Total:=0;
        For p' in p'0,aNc .. 0 loop
                maxima_seccion:=0; suma:=0;
                For cada τaj ∈ Γa loop
                        if procesador(τaj)=procesador(τab) then
                                if prioridad(τaj)≥prioridad(τab) then
                                        Celda:=0;
                                        if p'≥p'0,ajc and t ≥ φ'ajc - (p'-1)Ti then Celda:= Caj;
                                        if p'≥p'0,acc and j>c and Haj(τab)≠Hac(τab) then Celda := 0;
                                        if p'≤p'ab and j<b and Haj(τab)≠Hab(τab) then Celda := 0;
                                        if (p'≥p'ab and j>b) or (p'>p'ab and j=b) then Celda := 0;
                                        suma:=suma+Celda;
                                else
                                        if suma>maxima_seccion then maxima_seccion:=suma;
                                        suma:=0;
                                end if;
                        end if;
                end loop;
                Total:=Total+maxima_seccion;
        end loop;
        return Total;
end RESUELVE_CONFLICTOS_Γa;
```

Con la resolución de los conflictos de activación obtenemos la interferencia debida a instancias activadas antes del instante crítico. Otro aspecto que nos falta todavía por considerar es el de las expulsiones debidas a tareas activadas en instancias posteriores al instante crítico.

Para obtener la expresión de esa interferencia debemos tener en cuenta dos efectos, debidos a las relaciones de precedencia en las tareas de la transacción Γ_a y que en parte ya hemos tenido en cuenta para reducir la tabla de conflictos:

- Instancias posteriores al instante crítico sólo pueden interferir en el periodo de ocupación con tareas pertenecientes al primera sección *H*, incluidas en el conjunto MP_a.

- En el análisis de la activación p'_{ab} de una tarea τ_{ab} no puede haber interferencias debidas a tareas precedidas por ella activadas en instancias igual o posteriores al p'_{ab}.

La activación de una tarea τ_{aj}, correspondiente a la primera instancia posterior al instante crítico (con índice p'=1), se produce por definición en el instante φ'_{ajc}. Por tanto, una tarea τ_{aj} del conjunto MP_a se activará periódicamente cada T_a a partir del instante φ'_{ajc}.

Diferenciaremos la contribución de tareas pertenecientes al conjunto MP_a según precedan o sean precedidas por la tarea analizada. Las tareas que precedan a τ_{ab} no tienen ninguna restricción en cuanto a sus posibles expulsiones, de forma que su contribución al periodo de ocupación de anchura t vendrá dada por la expresión:

$$\sum_{\substack{\forall j \in MP_a(\tau_{ab}) \\ j<b}} \left\lceil \frac{t-\varphi'_{ajc}}{T_a} \right\rceil_0 C_{aj} \tag{41}$$

Las tareas precedidas por τ_{ab} sí tienen un límite en cuanto a sus posibles expulsiones, puesto que sólo pueden expulsar la ejecución de su activación p'_{ab} si corresponden a instancias anteriores a la instancia p'_{ab}, así que su contribución será:

$$\sum_{\substack{\forall j \in MP_a(\tau_{ab}) \\ j>b}} \min\left(p'_{ab}-1 \ , \left\lceil \frac{t-\varphi'_{ajc}}{T_a} \right\rceil_0 \right) C_{aj} \tag{42}$$

Por último, la contribución al periodo de ocupación de la propia tarea analizada. En la tabla de conflictos ya se tuvieron en cuenta las activaciones de instancias anteriores al instante crítico, por tanto, sólo nos resta considerar las activadas en instancias posteriores al instante crítico (con índices p'>0) hasta la activación analizada p'_{ab}, esto es,

$$p'_{ab}\ C_{ab} \tag{43}$$

Nótese que estas dos últimas expresiones sólo tienen sentido para el análisis de las activaciones de τ_{ab} correspondientes a instancias posteriores al instante crítico y por tanto, para valores negativos de p'_{ab} deben ser iguales a 0. En resumen, la contribución al peor caso de tareas activadas por eventos ocurridos después del instante crítico, incluida la propia tarea τ_{ab}, se puede expresar de la siguiente forma:

$$W_{ac}(\tau_{ab},t)\big|_{p'_{ab}>0} = \sum_{\substack{\forall j \in MP_a(\tau_{ab}) \\ j<b}} \left\lceil \frac{t-\varphi'_{ajc}}{T_a} \right\rceil_0 C_{aj} + \max\left(0, p'_{ab}C_{ab} + \sum_{\substack{\forall j \in MP_a(\tau_{ab}) \\ j>b}} \min\left(p'_{ab}-1, \left\lceil \frac{t-\varphi'_{ajc}}{T_a} \right\rceil_0\right) C_{aj}\right) \quad (44)$$

De manera que la contribución total de tareas de la transacción Γ_a al periodo de ocupación para la activación p'_{ab} de la tarea τ_{ab} se calcula como:

$$W_{ac}(\tau_{ab},p'_{ab},t) = RESUELVE\_CONFLICTOS\_EN\_\Gamma_a(\tau_{ab},p'_{ab},t,\Gamma_a,\tau_{ik}) + W_{ac}(\tau_{ab},p'_{ab},t)\big|_{p'_{ab}>0} \quad (45)$$

Con esta expresión se completa la nueva técnica de análisis, en la que consideramos las relaciones de precedencia y los perfiles de prioridad en las transacciones. En esta técnica, el tiempo de finalización de una activación p'_{ab} se calcula según la expresión:

$$w_{abc}(p'_{ab}) = B_{ab} + W_{ac}\left(\tau_{ab}, p'_{ab}, w_{abc}(p'_{ab})\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, w_{abc}(p'_{ab})\right) \quad (46)$$

donde $W_i^*(\tau_{ab},w_{abc}(p'_{ab}))$ corresponde a la ecuación (37). Y, a partir de él, el tiempo de respuesta global de peor caso se determina con la expresión:

$$R_{abc}(p'_{ab}) = w_{abc}(p'_{ab}) - \varphi'_{abc} - (p'_{ab}-1)T_a + \Phi_{ab} \quad (47)$$

En principio, habría que iterar este análisis para todas las activaciones ocurridas en el periodo de ocupación L_{abc}, calculado mediante la ecuación:

$$L_{abc} = B_{ab} + W_{ac}\left(\tau_{ab}, L_{abc}\right) + \sum_{\forall i \neq a} W_i^*\left(\tau_{ab}, L_{abc}\right) \quad (48)$$

donde $W_{ac}(\tau_{ab},L_{abc})$ se obtiene de la expresión (33). Por tanto habría que analizar las activaciones con índices entre $p'_{0,abc}$ y $p'_{L,abc}$, obtenido a partir de ese periodo de ocupación como:

$$p'_{L,abc} = \left\lceil \frac{L_{abc}-\varphi'_{abc}}{T_a} \right\rceil_0 \quad (49)$$

Sin embargo, la existencia de conflictos de ejecución hace que también se pueda restringir el número total de activaciones a chequear en el análisis: por ejemplo, si la primera tarea de la transacción Γ_a tiene prioridad menor que la de la tarea τ_{ab} bajo análisis, quiere decir que no tenemos que chequear las activaciones con valores $p'_{ab}>0$, puesto que no pueden pertenecer al periodo de ocupación. Vamos a generalizar estos límites de chequeo: el primer

valor a chequear siempre corresponde a $p'_{0,abc}$. En cuanto al último valor de chequeo, $p'_{L,abc}$ debemos tener en cuenta los siguientes casos:

- Si la tarea τ_{ab} pertenece al conjunto MP_a, las activaciones correspondientes a instancias posteriores al instante crítico pueden ejecutar en el periodo de ocupación creado con la tarea τ_{ac}. Por tanto, analizaremos todas las activaciones ocurridas dentro del periodo de ocupación y calcularemos $p'_{L,abc}$ a partir de la expresión:

$$p'_{L,abc} = \left\lceil \frac{L_{abc} - \varphi'_{abc}}{T_a} \right\rceil_0 \qquad si \quad b \in MP_a \tag{50}$$

- Si la tarea τ_{ab} no pertenece a la primera sección MP_a no puede haber, dentro del periodo de ocupación, ninguna ejecución de tareas correspondientes a instancias posteriores al instante crítico, con índices $p'_{ab}>0$. Luego es suficiente con iterar el análisis sólo para valores $p'_{ab}<=0$. Hay otra posible simplificación en el número de activaciones a analizar, derivada de la regla 1 de reducción de conflictos: si la tarea con la que se crea el instante crítico, τ_{ac}, precede a la tarea analizada y ambas pertenecen a distintas secciones *H*, no pueden haber ejecuciones de la tarea τ_{ab} correspondientes a instancias igual o posteriores al que creó el instante crítico $p'_{0,acc}$. En ese caso, el último índice a analizar sería el $p'_{0,acc}$-1. Por tanto:

$$\begin{aligned} p'_{L,abc} &= p'_{0,acc} - 1 \qquad si \quad (c<b) \wedge (H_{ac}(\tau_{ab}) \neq H_{ab}(\tau_{ab})) \\ p'_{L,abc} &= 0 \qquad\qquad\quad en\ cualquier\ otro\ caso \end{aligned} \tag{51}$$

Evidentemente, siempre podemos simplificar el proceso chequeando todas las activaciones tal como hacíamos antes, esto es, usando la condición definida en (49). El análisis es correcto, desde el punto de vista que obtiene cotas superiores de los tiempos de respuesta, pero considera situaciones innecesarias o imposibles en la realidad, luego resultaría un análisis más pesimista. En definitiva, el tiempo de respuesta se obtendría de iterar desde el índice $p'_{0,abc}$ hasta el $p'_{L,abc}$, definido en las expresiones (50) y (51).

$$R_{ab} = \max_{\forall c \in XP_a(\tau_{ab})} \left[\max_{p=p'_{0,abc}..p'_{L,abc}} \left(w_{abc}(p'_{ab}) - \varphi'_{abc} - (p'_{ab}-1)T_a + \Phi_{ab} \right) \right] \tag{52}$$

Vamos a ver como se aplica esta técnica en el ejemplo mostrado en la Figura 5-12, para el análisis de la tarea τ_{a3} en el posible instante crítico creado con la tarea τ_{a5}. En la Figura 5-13-a se muestra la tabla de conflictos correspondiente a las instancias anteriores al

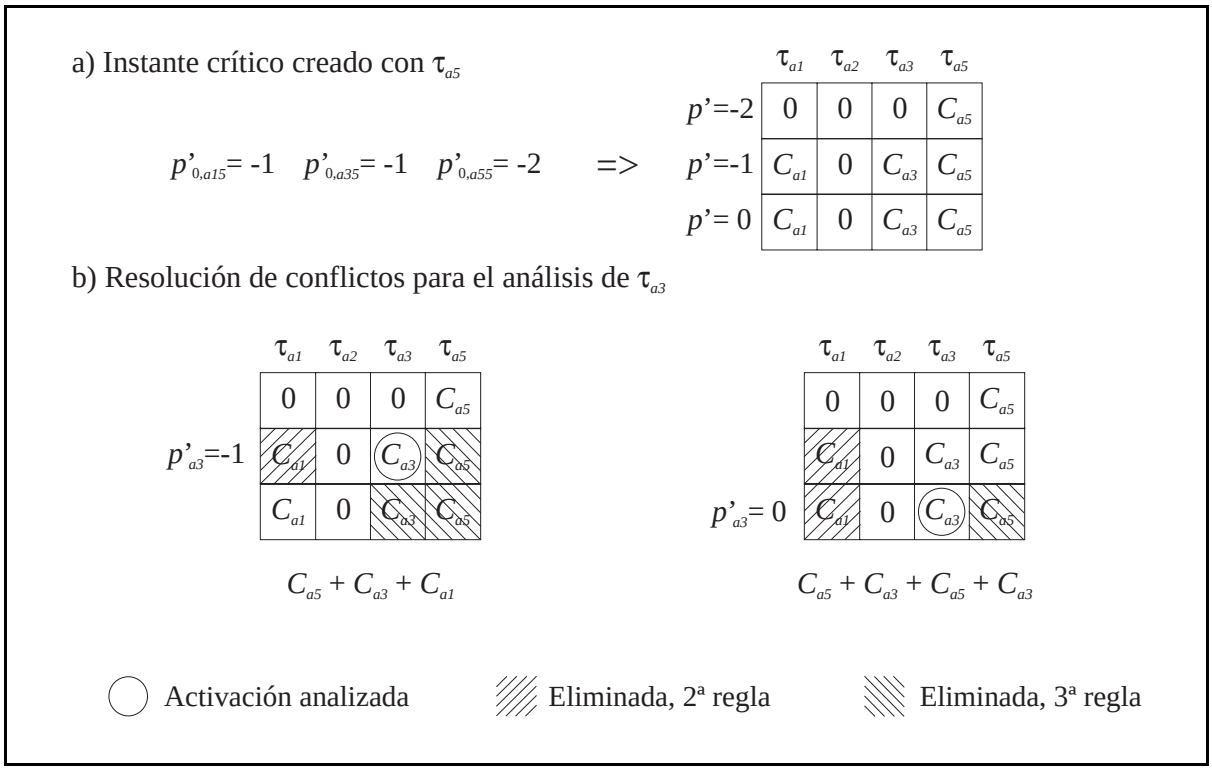


Figura 5-13. Análisis aplicado a la tarea τ_{a3} del ejemplo de la Figura 5-12

instante crítico. Como se puede ver, hay dos activaciones pendientes de la tarea τ_{a3}, correspondientes a los índices p'=-1 y p'=0. La Figura 5-13-b muestra las tablas de conflictos generadas para el análisis de cada una de esas dos activaciones, resultantes de aplicar sobre la tabla de conflictos original las reglas de reducción vistas anteriormente. Asimismo, en la parte inferior de cada tabla se muestra el resultado obtenido en la resolución de dichos conflictos.

Si aplicamos los criterios definidos para el número de activaciones a analizar, podremos ver que únicamente tendremos que realizar el análisis para valores comprendidos entre $p'_{0,a35}$=-1 hasta $p'_{L,a35}$=0, puesto que la tarea τ_{a3} no pertenece al conjunto $MP_a(\tau_a)$, formado únicamente por la tarea τ_{a1}.

La contribución al peor caso de τ_{a3}, en su primera activación analizada, con p'_{ab}=-1 es:

$$W_{a5}(\tau_{a3}, -1, t) = C_{a5} + C_{a3} + C_{a1} + \left\lceil \frac{t - \varphi'_{a15}}{T_a} \right\rceil_0 C_{a1} \qquad (53)$$

que usaremos para calcular el tiempo de finalización $w_{a35}(-1)$ y, a partir de él, el tiempo de respuesta global correspondiente,

$$R_{a35}(-1) = w_{a35}(-1) - \varphi'_{a35} + 2T_a + \Phi_{a3} \tag{54}$$

Para la segunda activación, con $p'_{ab}=0$, la contribución al peor caso se calcula mediante:

$$W_{a5}(\tau_{a3}, 0, t) = C_{a5} + C_{a3} + C_{a5} + C_{a3} + \left\lceil \frac{t - \varphi'_{a15}}{T_a} \right\rceil_0 C_{a1} \tag{55}$$

con la que obtendremos $w_{a35}(0)$ y, a su vez, el tiempo de respuesta según:

$$R_{a35}(0) = w_{a35}(0) - \varphi'_{a35} + T_a + \Phi_{a3} \tag{56}$$

De esta forma, el tiempo de respuesta de la tarea τ_{a3}, en cualquier periodo de ocupación que comience con un instante crítico creado con la tarea τ_{a5} está acotado por el valor:

$$R_{a35} = Max\left(R_{a35}(-1), R_{a35}(0)\right) \tag{57}$$

Integrando esta nueva formulación, que llamaremos APCOEPP (Análisis de Peor Caso con *Offsets* Estáticos y Perfiles de Prioridad) en el algoritmo iterativo para *offsets* dinámicos, obtendremos mejores estimaciones de los tiempos de respuesta de peor caso en sistemas con relaciones de precedencia en la activación de sus tareas, que las obtenidas con los algoritmos APCOD Y AMPCOD. La Figura 5-14 muestra el nuevo algoritmo, que llamaremos APCODPP (Análisis de Peor Caso con *Offsets* Dinámicos y Perfiles de Prioridad).

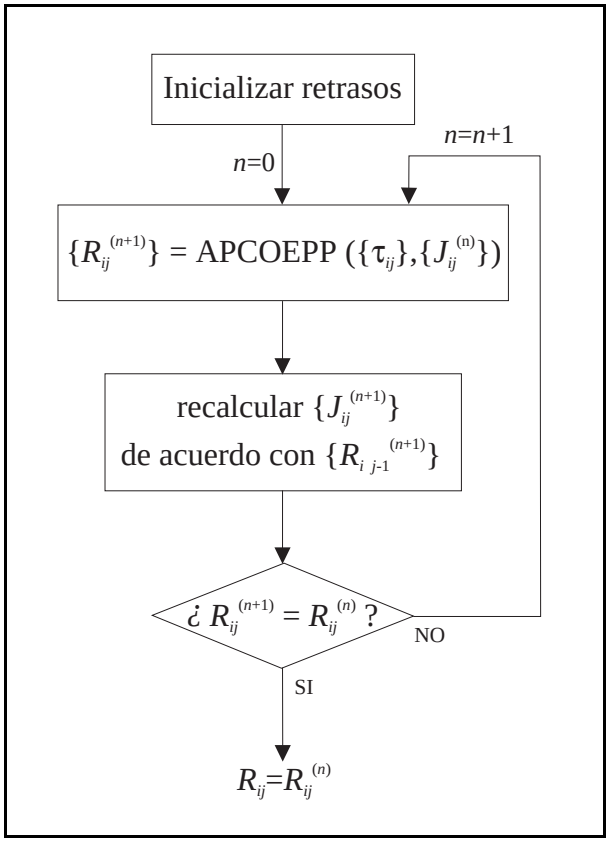


Figura 5-14. Algoritmo APCODPP

En la siguiente sección veremos resultados de simulación de las mejoras introducidas por este algoritmo APCODPP.

5.3.3 Comparación de resultados

En esta sección vamos a cuantificar las mejoras conseguidas mediante la aplicación del algoritmo APCODPP, basado en tareas con *offsets* y perfiles de prioridad. Tal como hemos hecho con las técnicas desarrolladas previamente en esta tesis, generaremos aleatoriamente diferentes conjuntos de tareas, ejecutando en uno o varios procesadores, y compararemos los tiempos de respuesta calculados con cada algoritmo. Para ello se eligen los periodos de activación de las transacciones según una distribución exponencial en el rango (T_{min}, T_{max}) y en función de esos periodos se asocian los plazos globales de ejecución. Cada tarea llevará asociado un tiempo aleatorio de ejecución de peor caso calculado teniendo en cuenta su periodo y la utilización del procesador donde ejecute. De igual manera que en simulaciones anteriores, realizaremos los análisis suponiendo, bien tiempos de respuesta de mejor caso nulos, o bien tiempos de respuesta de mejor caso iguales a la suma de los tiempos de ejecución de la propia tarea y de sus precedentes en la misma transacción.

En cuanto a la asignación de prioridades a las tareas, vamos a estudiar los dos tipos de asignación utilizadas hasta ahora en esta tesis. En la primera, asignaremos prioridades en función de los plazos de ejecución de las tareas, según el esquema *Deadline Monotonic*, con prioridades en orden decreciente para tareas en una misma transacción, tal y como hicimos en las simulaciones descritas en la sección 4.5 de esta memoria. El segundo esquema de asignación que estudiaremos es similar a este, pero hacemos variar aleatoriamente la prioridad asignada a una tarea por transacción y procesador. Esta variación se justifica por el hecho de que la asignación *Deadline Monotonic* es óptima en sistemas monoprocesadores, pero no en sistemas multiprocesadores y distribuidos, donde la asignación óptima de prioridades se busca mediante algún tipo de algoritmo heurístico (ver algoritmo *HOPA* en [GUT95A]). Para este segundo método de asignación de prioridades se representarán utilizaciones hasta el 50%, puesto que para utilizaciones superiores empieza a no converger la técnica de Tindell y Clark. Para el primer esquema, sin embargo, sí se pueden obtener valores acotados de R_{indep} prácticamente hasta el 100% de utilización.

El tipo de experimentos realizados son de la misma familia que los realizados para estudiar las ventajas de los algoritmos APCOD y AMPCOD. Las primeras gráficas muestran los resultados de la simulación cuando se comparan los tiempos de respuesta obtenidos con

el algoritmo mejorado basado en *offsets* dinámicos, R_{AMPCOD}, y con el algoritmo bajo estudio, basado en *offsets* y perfiles de prioridad, $R_{APCODPP}$. Ambos tiempos de respuesta se calcularán relativos al obtenido mediante la técnica de Tindell y Clark basado en tareas independientes, R_{indep}. Las tres primeras (Figura 5-15 a Figura 5-17) corresponden a la asignación de prioridades en función de los plazos de ejecución más la componente aleatoria, y las tres siguientes (Figura 5-18 a Figura 5-20) para asignación estrictamente en función de los plazos de ejecución de las transacciones y en orden decreciente dentro de cada transacción. El eje de abcisas representa la utilización del procesador. Para cada utilización se genera un conjunto de tareas y se comparan los resultados obtenidos, mediante las relaciones R_{indep}/R_{AMPCOD} y $R_{indep}/R_{APCODPP}$. Cada gráfica representa los resultados para tres relaciones diferentes entre los máximos y los mínimos periodos de las transiciones, T_{max}/T_{min} (10, 100 y 1000), representando cada punto el promedio de cinco simulaciones.

La Figura 5-15 muestra los resultados para conjuntos de 10 transacciones y 10 tareas por transacción ejecutando en un único procesador, considerando nulos los tiempos de respuesta de mejor caso de las tareas, y por tanto, con *offsets* iguales a 0. Se puede ver que los beneficios de las técnicas basadas en *offsets* tienden a aumentar al disminuir la relación T_{max}/T_{min}, aunque para valores bajos de utilización prácticamente no se diferencian. Las mejoras obtenidas con los métodos basados en *offsets* aumentan rápidamente con la utilización del procesador. Por ejemplo, para una utilización cercana al 40% se obtiene una relación R_{AMPCOD}/R_{indep} que mejora el análisis entre 1.8 y 3.5 veces, aumentando hasta una mejora de 6 a 12.5 veces en los tiempos de respuesta obtenidos con el algoritmo APCODPP, a pesar de que consideramos *offsets* nulos. En la Figura 5-16 se muestran las mejoras promedio para sistemas con 10 transacciones con 3 tareas ejecutando en cada uno de los 4 procesadores considerados, con tiempos de mejor caso nulos. La mejora, para el 40% oscila entre $R_{APCODPP}/R_{indep}= 1.65$ y $R_{APCODPP}/R_{indep}= 2.23$, aunque aumenta hasta valores comprendidos entre 2 y 3 al considerar tiempos de mejor caso (y por tanto *offsets*) no nulos, como se puede comprobar en la Figura 5-17.

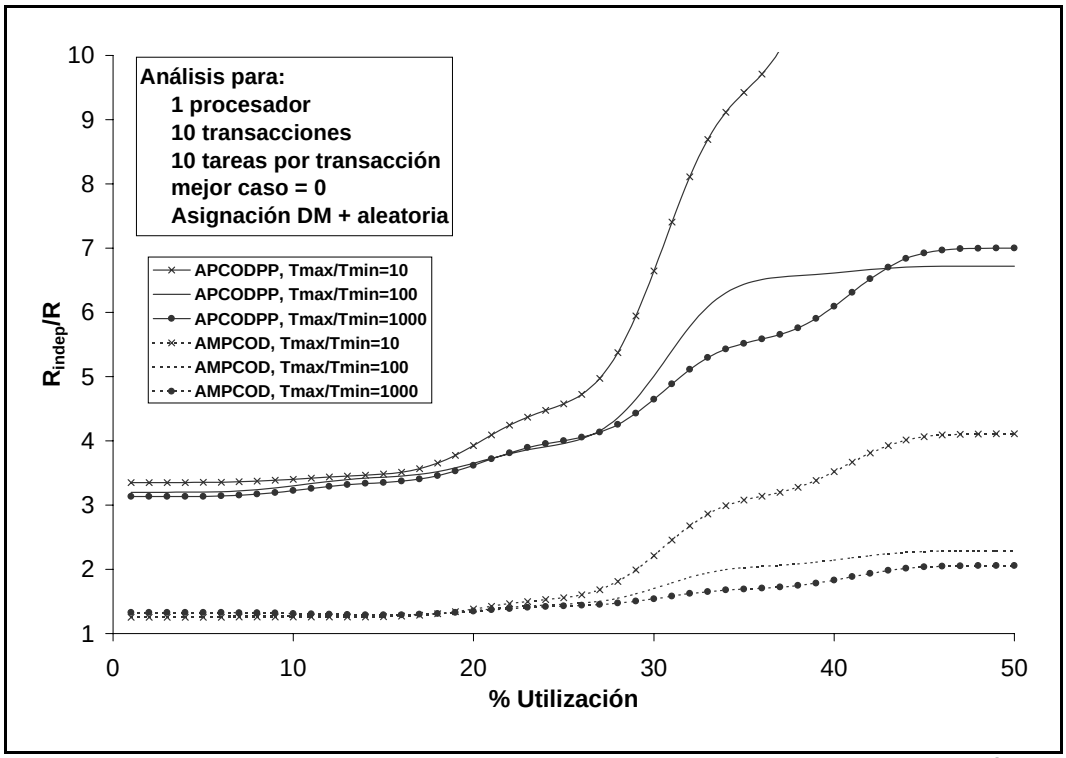


Figura 5-15 Comparación entre R_{AMPCOD} y R_{APCODPP}, para 1 procesador y $C_i^b = 0$

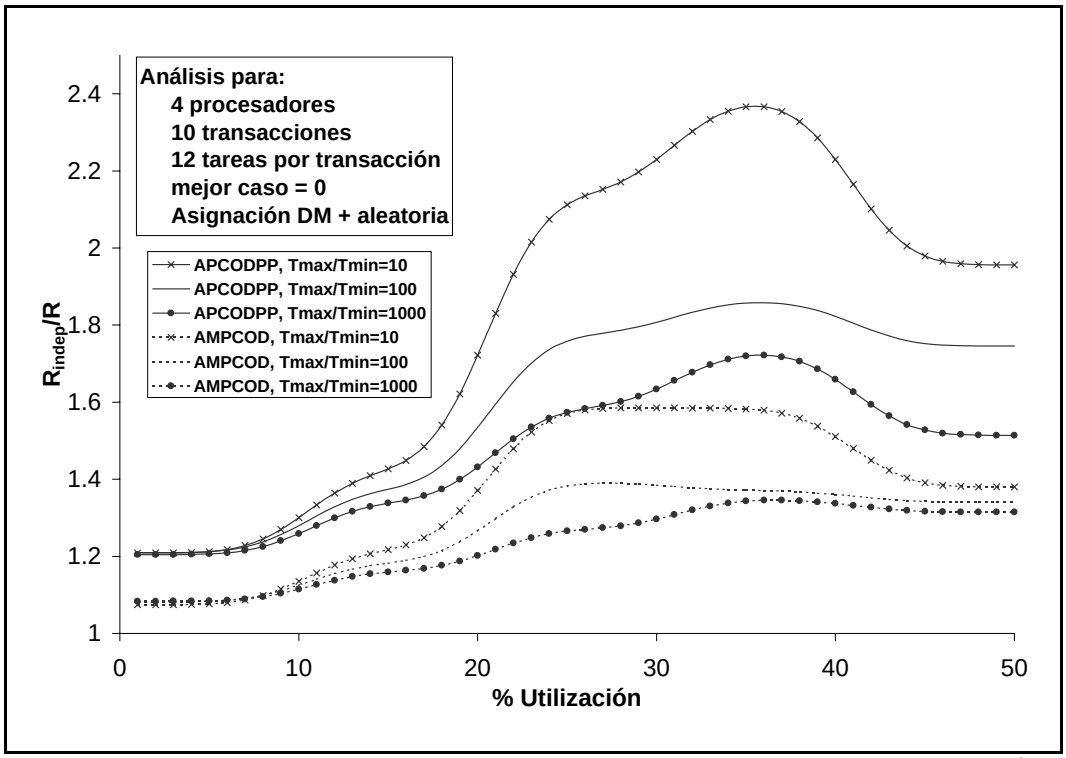


Figura 5-16 Comparación entre R_{AMPCOD} y R_{APCODPP}, para 4 procesadores y C_i^b=0

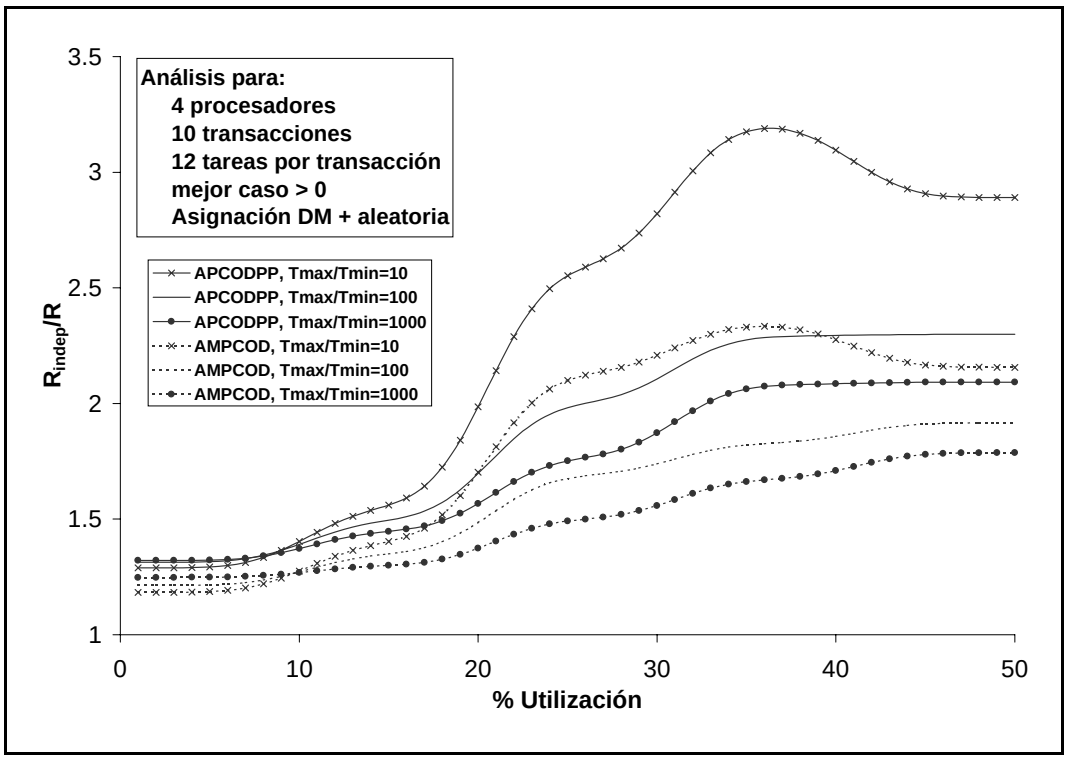


Figura 5-17 Comparación entre R_{AMPCOD} y R_{APCODPP}, para 4 procesadores y $C_i^b>0$

Cuando estudiamos sistema generados de igual forma que los anteriores, pero con asignación de prioridades *Deadline Monotonic* se obtienen los resultados reflejados en las siguientes gráficas. La Figura 5-18 muestra los resultados del análisis aplicado al sistema monoprocesador con 10 transacciones y 10 tareas por transacción. Como se pude ver, la técnica APCODPP ya parte de mejoras cercanas a $R_{\text{indep}}/R_{\text{APCODPP}}=5$, y crece según aumentamos la utilización del procesador. Nuevamente, los beneficios disminuyen al distribuir las tareas en 4 procesadores (Figura 5-19) rondando valores cercanos a $R_{\text{indep}}/R_{\text{APCODPP}}=1.4$ para utilizaciones medias del 70% y aumentando hasta aproximadamente 2.5 si consideramos tiempos de respuesta de mejor caso no nulos (ver Figura 5-20).

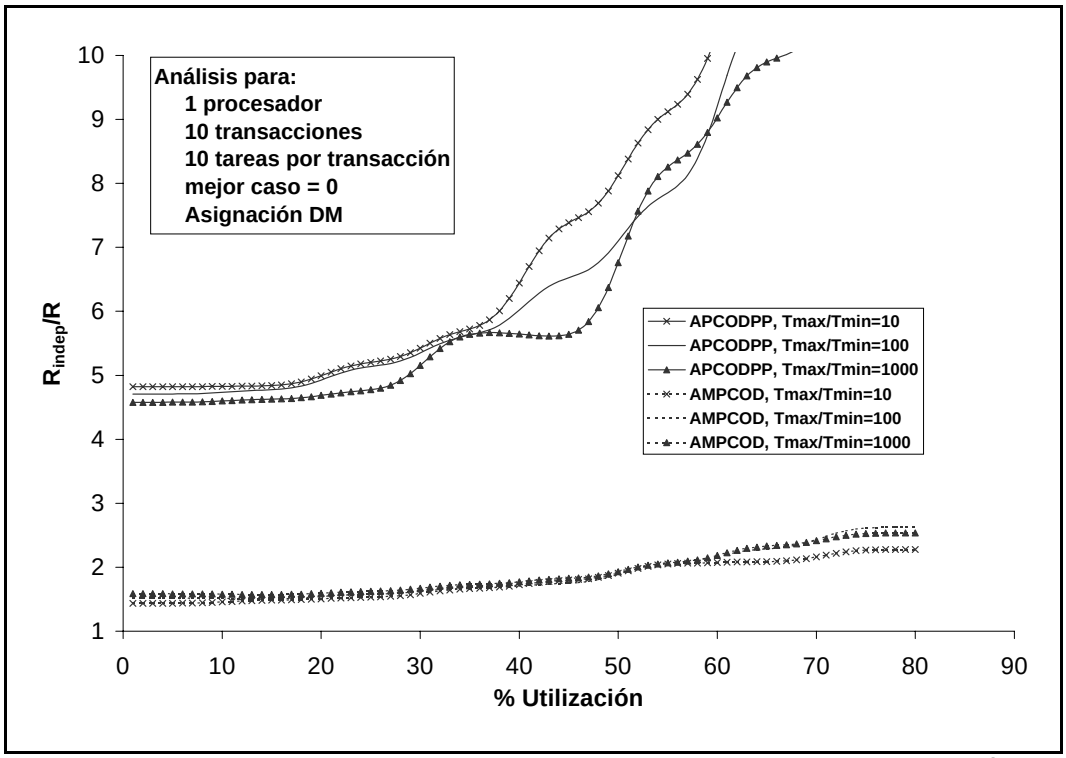


Figura 5-18 Comparación entre R_{AMPCOD} y R_{APCODPP}, para 1 procesador y $C_i^b = 0$

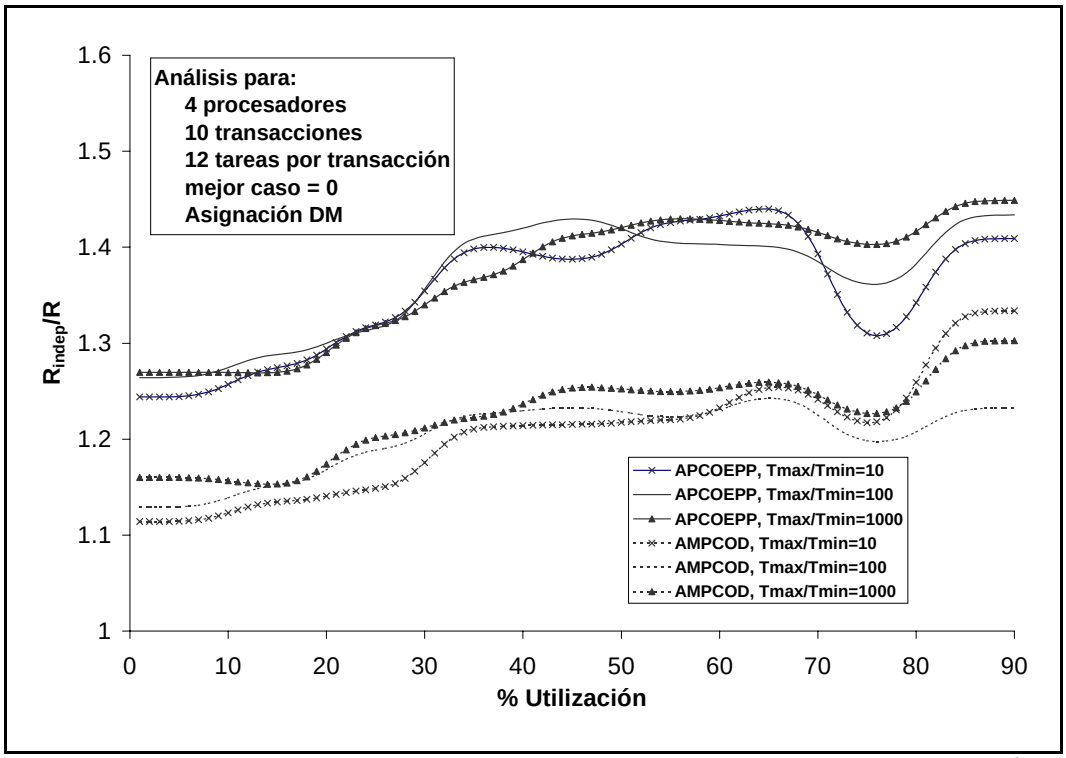


Figura 5-19 Comparación entre R_{AMPCOD} y R_{APCODPP}, para 4 procesadores y $C_i^b=0$

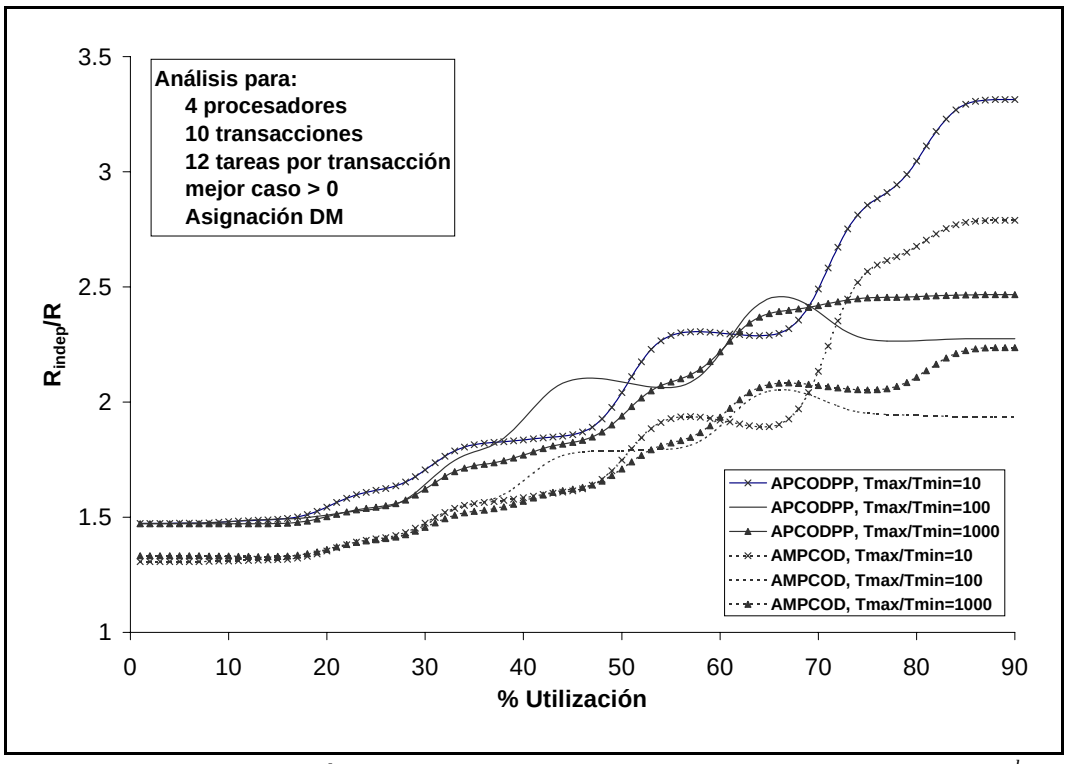


Figura 5-20 Comparación entre R_{AMPCOD} y R_{APCODPP}, para 4 procesadores y $C_i^b>0$

El segundo tipo de experimentos simulados (Figura 5-21 a Figura 5-25) compara los límites máximos de utilización de un determinado sistema generado aleatoriamente. Al igual que en secciones anteriores, el límite de utilización se calcula analizando el sistema partiendo de una utilización baja y aumentado proporcionalmente los tiempos de ejecución de cada tarea (manteniendo el resto de parámetros del sistema) hasta que alguna tarea deje de cumplir algún plazo, de forma que la máxima utilización planificable corresponde al último conjunto de tareas planificable. El análisis se ha realizado además para diferentes relaciones D_i/T_i entre los periodos de activación de las tareas y sus plazos de ejecución, y con una relación $T_{\text{max}}/T_{\text{min}}$=100 para los periodos de activación. Las gráficas muestran el límite de utilización obtenido con cada una de las técnicas desarrolladas en esta tesis, además de la técnica de Tindell y Clark utilizada como referencia. Los análisis se han realizando con tiempos de respuesta de mejor caso nulos e incluyendo además el caso de tiempos de mejor caso no nulos para el algoritmo APCODPP. La Figura 5-21 muestra los resultados de la simulación de un sistema con 4 procesadores, 5 transacciones distribuidas en ellos con 20 tareas por transacción. Como se puede apreciar en ella, para valores D_i/T_i=3 y superiores se obtienen incrementos en torno al 8% en la utilización máxima. Para el caso D_i/T_i=4, que podríamos

considerar como normal, se obtiene un incremento absoluto de 9% (mejora relativa del 60%) con el algoritmo APCODPP. En la Figura 5-22 se estudia un sistema similar, pero con 12 tareas por transacción en lugar de 20. En este caso, el incremento es cercano al 12% a partir de D_i/T_i=1.5, con un incremento de un 10% en el límite de utilización planificable en D_i/T_i=4, que supone una mejora relativa del 43% para el algoritmo APCODPP frente al basado en tareas independientes.

Se han realizado simulaciones similares para asignación de prioridades *Deadline Monotonic* sin variación. Las utilizaciones máximas derivadas de su análisis se muestran en las gráficas 5-23 a 5-25. La Figura 5-23 contiene el resultado del análisis sobre un sistema con 4 procesadores y 5 transacciones de 20 tareas cada una. Los incrementos absolutos conseguidos en la utilización máxima son del orden del 20%, significando una mejora en el análisis del 65% para D_i/T_i=4. Las últimas dos gráficas (Figura 5-24 y Figura 5-25) muestran los resultados sobre sistemas con 5 transacciones de 3 tareas en cada procesador y asignación de prioridades *Deadline Monotonic,* con mejoras significativas (58% y 60% en D_i/T_i=4) para los algoritmos desarrollados en esta tesis.

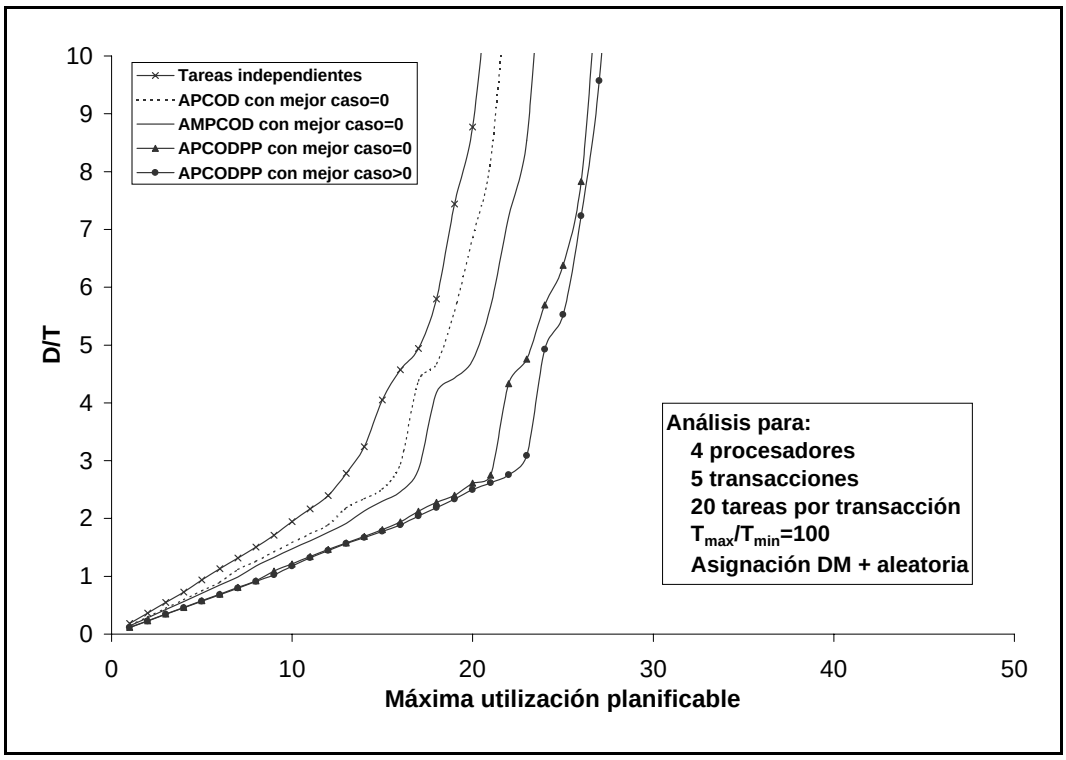


Figura 5-21 Máxima utilización planificable, 20 tareas por transacción

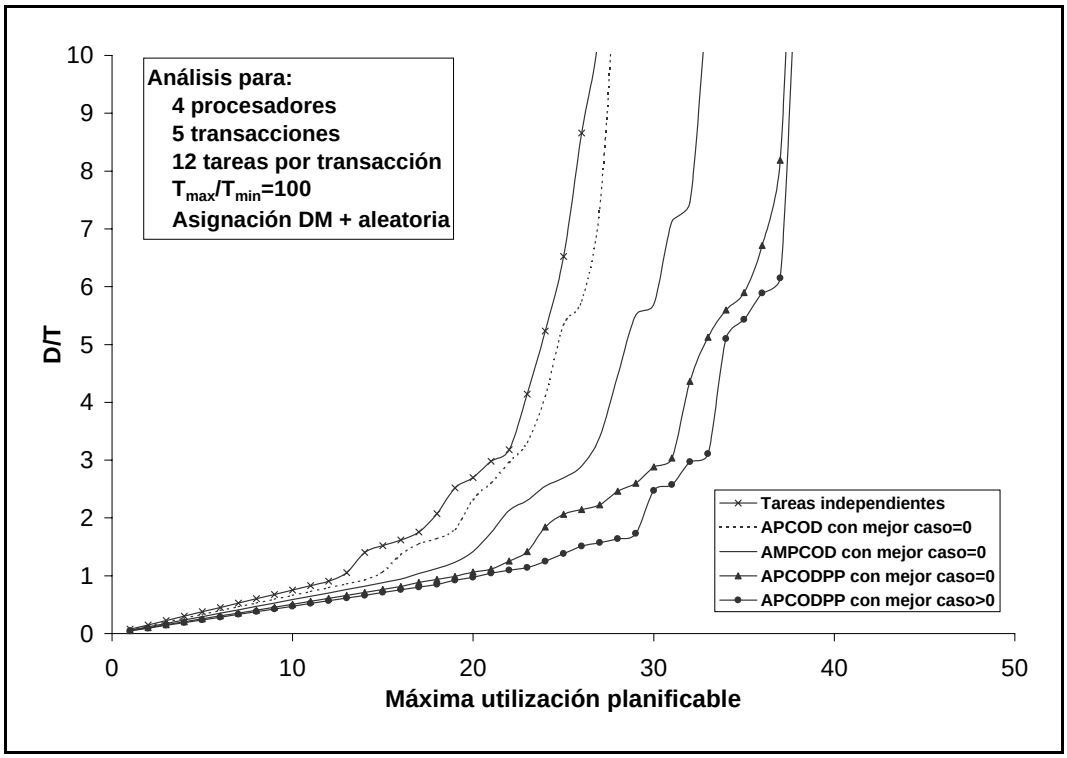


Figura 5-22 Máxima utilización planificable, 12 tareas por transacción

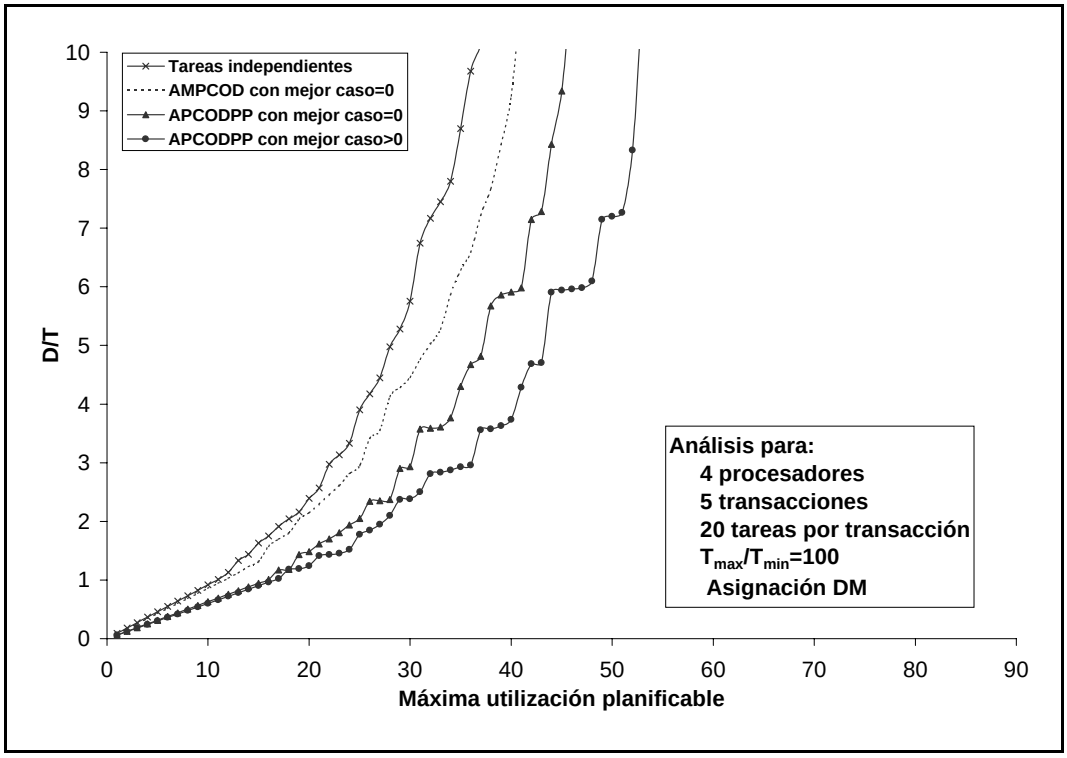


Figura 5-23 Máxima utilización planificable, 20 tareas por transacción

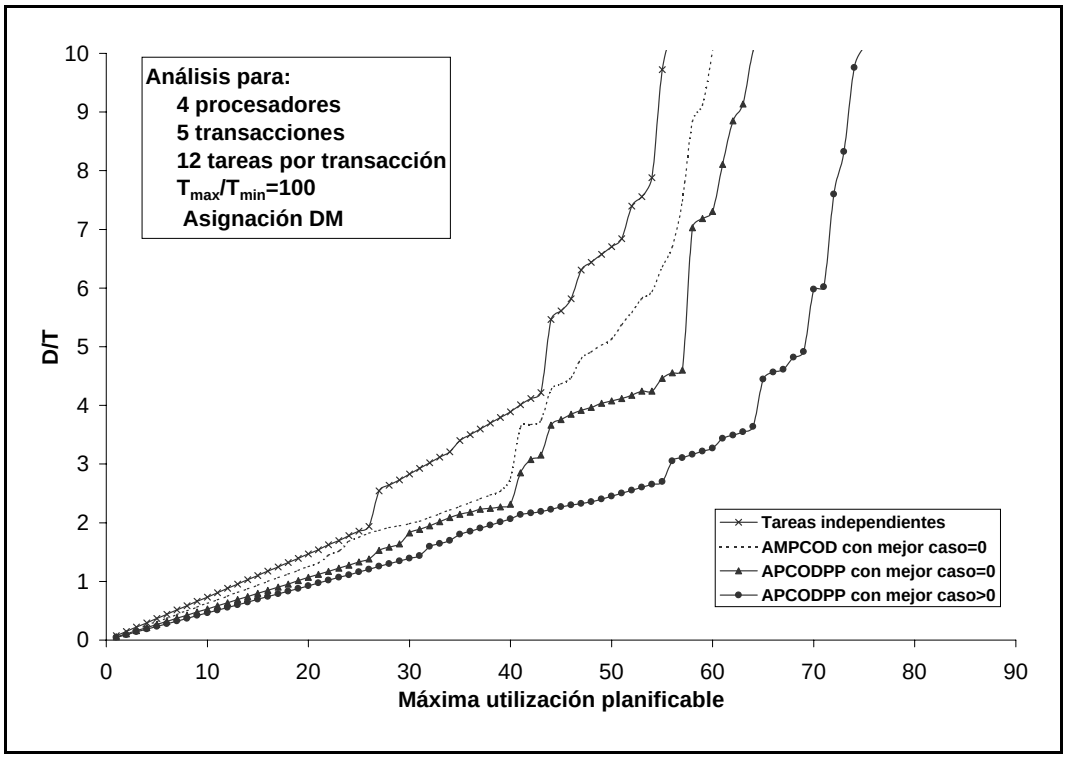


Figura 5-24 Máxima utilización planificable, 12 tareas por transacción

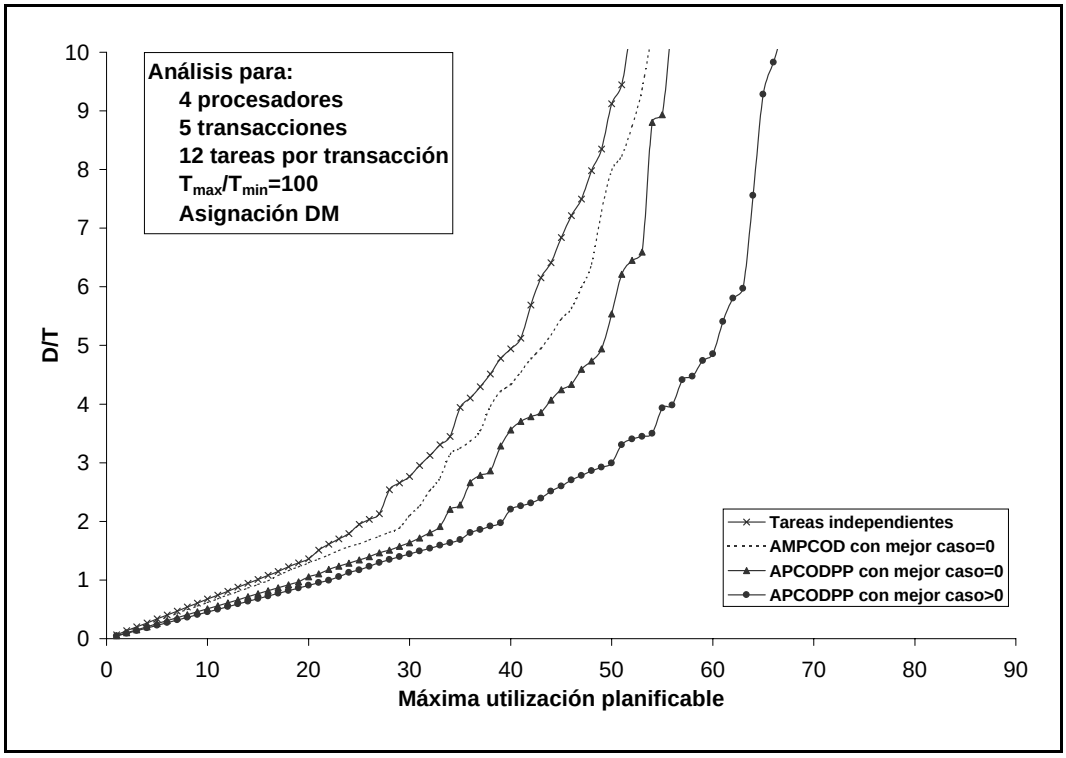


Figura 5-25 Máxima utilización planificable, 12 tareas por transacción

5.4. Tareas con prioridades variantes

Un modelo interesante para tareas en sistemas de tiempo real planificados mediante prioridades fijas es el de tareas periódicas con prioridades de ejecución variantes [GON91A] [GON94]. Una tarea con prioridad de ejecución variante es una tarea que ejecuta en un único procesador y que está compuesta por secciones de código que ejecutan a diferentes niveles de prioridad. Su planificación está basada en prioridades fijas, puesto que una determinada sección de código siempre ejecuta al mismo nivel de prioridad; sin embargo, no encaja con el modelo tradicional de tarea (manejado en esta tesis), en el que la prioridad asignada a una tarea es única. El modelo computacional es el de una tarea τ_i compuesta por diferentes subtareas τ_{ij} que ejecutan de forma consecutiva, caracterizadas cada una por su tiempo de ejecución de peor caso, C_{ij}, y por su prioridad, P_{ij}.

Desde el punto de vista del modelo general contemplado en esta tesis, un sistema gobernado por eventos está compuesto por un conjunto de tareas, agrupadas en diferentes secuencias de ejecución. Cada tarea genera un evento interno a su finalización, que dispara la ejecución de la tarea a la que precede en la secuencia de respuesta al evento externo. Para su análisis, hemos propuesto en esta tesis el modelo transaccional, donde asociamos a cada tarea *offsets* y términos de retraso en función de la ejecución de las tareas precedentes en la cadena de respuesta al evento. Ese mismo modelo transaccional es válido para tareas con prioridades variantes: cada tarea variante τ_i se puede modelar mediante una transacción τ'_i disparada por el mismo evento y constituida por tantas tareas τ'_{ij} como subtareas τ_{ij} tenga la tarea variante original. Cada tarea de la transacción vendrá definida por los mismos parámetros que la correspondiente subtarea original, con tiempo de ejecución de peor caso $C'_{ij}=C_{ij}$ y prioridad $Prio(\tau'_{ij})=P_{ij}$. Los términos de retraso y *offset* de cada tarea τ'_{ij} en el modelo transaccional vendrán determinados por los tiempos de respuesta de las subtareas precedentes a la subtarea τ_{ij} en la tarea variante original.

La única diferencia entre el modelo computacional manejado por González Harbour, Klein y Lehoczky, y el modelo transaccional es el concerniente al *Supuesto 3* (ver [GON91A], [GON94]). En él se establece que una instancia de una tarea no puede comenzar a ejecutar a no ser que haya finalizado la ejecución de instancias anteriores de la misma tarea. Tal como indican los autores, este supuesto no se puede mantener en tareas controladas por

interrupciones, en las que la primera subtarea se encarga de atender a la interrupción ejecutando a un nivel de prioridad normalmente muy alto. En tal caso, cualquier subtarea posterior en la tarea puede ser expulsada por esa primera subtarea, violando ese principio. Exactamente lo mismo puede ocurrir en los sistemas gobernados por eventos, donde las diferentes tareas que lo componen se activan por la llegada de eventos (ya sea internos o externos). Este efecto de expulsión es ciertamente posible en tareas con plazos posteriores a los periodos de activación. Aunque las técnicas desarrolladas en [GON91A] se han desarrollado para el *Supuesto 3*, pueden ser modificadas para eliminar esa limitación.

El hecho de que las tareas variantes se puedan modelar mediante transacciones con *offsets* dinámicos permite utilizar directamente las técnicas de análisis desarrolladas en esta tesis. Las cotas superiores estimadas para los tiempos de respuesta globales de las transacciones serán, a su vez, cotas superiores de los tiempos de respuesta de las tareas variantes a las cuales modelan. El análisis deducido en [GON91A] para tareas variantes ejecutando en un sistema monoprocesador es más preciso que el análisis basado en transacciones, puesto que no parte de la independencia en las interferencias debidas a diferentes transacciones (tareas variantes), limitando más las situaciones reales en las que puede producirse el periodo máximo de ocupación para el análisis de una subtarea. A cambio, nuestras técnicas de análisis se pueden aplicar a tareas con prioridades variantes ejecutando en sistemas multiprocesadores y distribuidos, lo cual les da un valor añadido.

A título orientativo, vamos a mostrar los resultados que se obtienen al aplicar las diferentes técnicas a un ejemplo particular. Para ello, vamos a utilizar el usado en [GON91A] para ilustrar su técnica y que está derivado de una aplicación real en un sistema robotizado de tiempo real. Después de varias consideraciones acerca de la arquitectura *hardware* y *software* del sistema real llegan a definir un modelo constituido por 5 tareas con secciones de diferente prioridad, cuyos parámetros se describen en la Tabla 5-I. Para poder comparar las técnicas de análisis vamos a hacer, sobre este ejemplo, medidas similares a las realizadas en secciones previas de esta tesis. Por un lado, obtendremos las relaciones entre los tiempos de respuesta calculados con las técnicas basadas en *offsets* y con la técnica para tareas variantes, referidos a los tiempos de respuesta calculados con el método basado en tareas independientes. Para ello, en función de la utilización que queramos estudiar modificaremos

proporcionalmente los tiempos de ejecución dados en la Tabla 5-I y aplicaremos el análisis en el conjunto de tareas resultante.

τ_i	T_i	C_{i1}	C_{i2}	C_{i3}	P_{i1}	P_{i2}	P_{i3}
τ_1	40	1	5	---	10	7	---
τ_2	100	10	5	5	4	8	4
τ_3	50	8	12	---	5	8	---
τ_4	200	10	20	3	9	2	3
τ_5	400	2	12	10	3	1	6

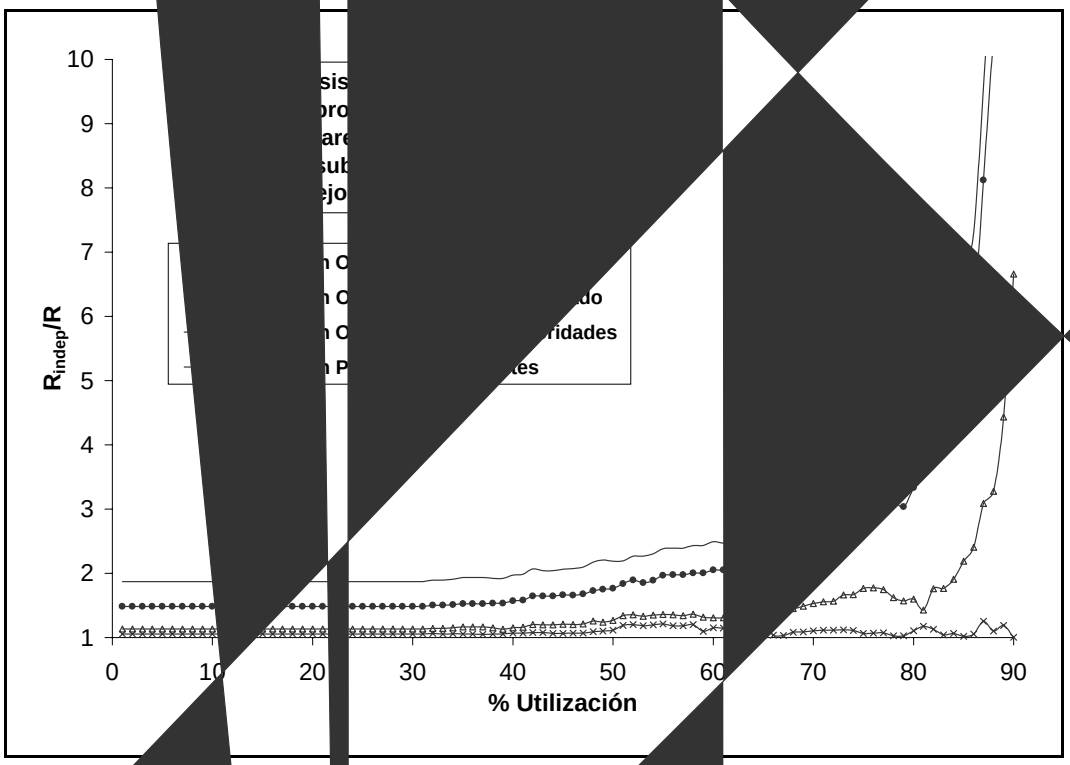
Tabla 5-I Parámetros del ejemplo

La Figura 5-26 muestra los resultados de ese análisis. Como se puede observar, los tiempos de respuesta resultantes del análisis con *offsets* dinámicos, APCOD, corresponden a unos resultados pobres, bastante parecidos a los obtenidos con tareas independientes. El método con *offsets* mejorado, AMPCOD, obtiene mejores resultados, aunque son aún bastante peores que los obtenidos con el método de [GON91A]. Sin embargo, el método completo, basado en *offsets* dinámicos y perfiles de prioridad, aunque también obtiene peores resultados que el método para tareas variantes (como era de esperar), es capaz de calcular soluciones próximas a las obtenidas con ese método.

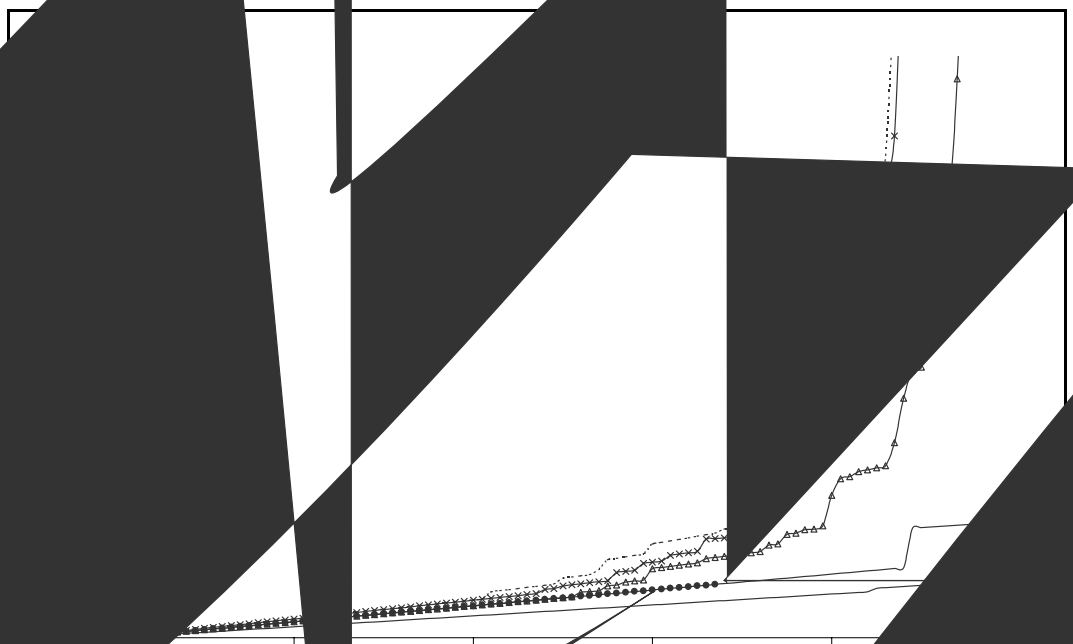
El otro tipo de medidas se refiere a la máxima utilización planificable que se puede conseguir si vamos aumentando progresivamente los tiempos de ejecución hasta que el sistema incumpla alguno de sus plazos de ejecución, proporcionales a los periodos de las tareas. Estudiando este límite para distintos valores de D_i/T_i se obtienen los resultados que se muestran en la figura Figura 5-27. De forma consecuente a los resultados del experimento anterior, el método basado en *offsets* dinámicos ofrece resultados similares a los del método basado en tareas independientes, no pudiendo obtener en ningún caso conjuntos planificables para utilizaciones superiores aproximadamente al 85%. Con el método mejorado se aumentaría este límite hasta algo más del 90%, si tuviéramos relaciones D_i/T_i mayores que 4. Nótese que la técnica de [GON91A] es capaz de obtener conjuntos planificables virtualmente hasta el 100% de utilización (suponiendo que no se viola el *Supuesto 3*). De hecho, para una utilización del 100% se obtiene un valor máximo D_i/T_i=1.271. Si nos fijamos en la curva relativa al método basado en *offsets* dinámicos y perfiles de prioridad podemos comprobar que

esta técnica también consigue conjuntos de tareas planificables hasta utilizaciones del 100%, aunque en este caso se obtiene un valor máximo D_i/T_i=2.426.

Estos resultados nos indican que el método basado en *offsets* dinámicos y perfiles de prioridad es un método bastante adecuado también para estudiar sistemas constituidos por tareas con prioridades de ejecución variantes. Con el valor añadido de que es válido también para sistemas multiprocesadores y distribuidos, para los cuales no hay una técnica específica desarrollada.



Figur Tiempos de respuesta en el eje on prioridades variantes



6. Conclusiones

Como conclusión al trabajo desarrollado en esta Tesis Doctoral, vamos a hacer un rápido resumen de su contenido indicando los logros conseguidos. Para ello, en primer lugar vamos a revisar los objetivos que nos habíamos marcado en el capítulo 1 de esta memoria, así como su grado de cumplimiento. Posteriormente, a modo de resumen, destacaremos las principales aportaciones del trabajo en el entorno de los sistemas de tiempo real. Finalmente, citaremos algunas posibles líneas de investigación hacia las que se podría encaminar en un futuro el trabajo iniciado en esta Tesis.

6.1. Revisión de objetivos

El objetivo principal de esta Tesis era mejorar el análisis de planificabilidad en sistemas distribuidos de tiempo real estricto gobernados por eventos. El modelo lineal que consideramos para este tipo de sistemas contempla los siguientes aspectos:

- Un conjunto de procesadores conectados entre sí por un sistema de comunicación con tiempos de transmisión acotados. A través de un conjunto de sensores se detectan los correspondientes eventos que disparan las acciones oportunas para realizar el control del sistema físico.

- El *software* del sistema está constituido por un conjunto de tareas distribuidas en los distintos procesadores, y que ejecutan según una secuencia de respuesta previamente determinada para cada posible evento.

- El planificador, tanto de las tareas en los procesadores como mensajes en las redes de comunicación, es de tipo expulsor basado en prioridades dinámicas. Los mensajes se parten en paquetes de tamaño fijo.

- Los requerimientos temporales impuestos al sistema pueden ser de tipo local o global para las tareas, o plazos de principio-a-fin para cada secuencia de respuesta a un evento. Los plazos de ejecución pueden ser mayores que los respectivos periodos de activación (o mínimo tiempo entre llegadas, en caso de eventos aperiódicos).

En el capítulo 1 hemos descrito la problemática que se presenta a la hora de verificar el cumplimiento de los requerimientos temporales en sistemas distribuidos de ese tipo, que básicamente consiste en los efectos de activación retrasada debidos a las relaciones de precedencia en la ejecución de acciones pertenecientes a una misma secuencia de respuesta. En el capítulo 2 dimos un repaso a las principales técnicas de análisis existentes. Dentro de ellas, hicimos especial hincapié en el método de Tindell y Clark para sistemas distribuidos, basado en la aproximación de tareas independientes.

El primer objetivo que nos planteamos fue el de introducir mejorar en la técnica de análisis de Tindell y Clark. Previamente a esta optimización nos centramos en demostrar de forma inequívoca la validez del método. Mediante el Teorema 3 en el capítulo 3 hemos demostrado que la condición de finalización impuesta por Tindell y Clark para el test de planificabilidad es correcta, en contra de la tesis sostenida por Sun y Liu. Como consecuencia de esto, concluimos que los tiempos de respuesta de peor caso estimados mediante la técnica basada en tareas independientes son cotas superiores válidas para los tiempos de respuesta máximos correspondientes a la ejecución de las tareas.

A partir de ese punto hemos realizado mejoras tomando como base la técnica de análisis de Tindell y Clark. Esas mejoras han ido en dos direcciones:

- Por un lado, hemos desarrollado una formulación, basándonos en la existente, para determinar de una forma más precisa los tiempos de respuesta locales de peor caso. Con ello conseguimos mejorar las estimaciones actuales, que toman los tiempos de respuesta globales como cotas superiores de los tiempos de respuesta locales.

- Por otro lado, hemos conseguido mejorar el cálculo de los tiempos de respuesta de peor caso incorporando a la técnica el cálculo de los tiempos de respuesta de mejor caso. Hasta ahora se habían estimado tiempos nulos como cotas inferiores de los

tiempos de respuesta globales de mejor caso. En el capítulo 3 hemos demostrado que una estimación más adecuada de estos tiempos de respuesta puede contribuir a mejorar la estimación de los tiempos de respuesta de peor caso. El cálculo de los tiempos de respuesta de mejor caso se puede abordar con cualquiera de las dos técnicas derivadas para ello: la estimación trivial o la estimación iterativa. La estimación iterativa obtiene soluciones ligeramente mejores que la trivial, a costa de una mayor complejidad algorítmica.

Incorporando al algoritmo de Tindell y Clark la nueva formulación desarrollada para los tiempos de respuesta de mejor caso conseguimos introducir mejoras de hasta un 5% en el nivel de planificabilidad de los sistemas de tiempo real simulados, con una complejidad algorítmica similar a la del método original. Es interesante destacar que esas ventajas se obtienen con coste cero para la implementación del sistema. El único requisito necesario es medir u obtener tiempos de ejecución de mejor caso de las tareas, de forma equivalente a la obtención de los tiempos de ejecución de peor caso.

El segundo objetivo planteado consistía en el desarrollo de nuevas técnicas de análisis para los sistemas distribuidos. Este objetivo se ha conseguido con las técnicas descritas en los capítulos 4 y 5 de esta memoria. El modelo basado en tareas independientes es válido, pero tiene en cuenta las relaciones de precedencia a través de términos de retraso, que no explotan la topología de las secuencias de respuesta a los eventos. En esta Tesis hemos utilizado un modelo bastante más adecuado, puesto que expresa directamente la pertenencia de tareas a una misma secuencia de respuesta. Este modelo es el modelo transaccional, que agrupa en una misma entidad las tareas pertenecientes a una misma secuencia de respuesta. El modelo de transacciones original, ideado también por Tindell, no es válido directamente para sistemas distribuidos, así que hemos tenido que extender su concepción y análisis en varios sentidos, tal como aparece descrito en el capítulo 4 de esta Tesis. Estos aspectos son:

- Hemos extendido el análisis para considerar el caso de *offsets* mayores que los periodos de activación de las transacciones. Esta extensión es conveniente, dada la distribución de las secuencias de respuesta entre diferentes procesadores y la posibilidad, por tanto, de plazos de ejecución mayores que los periodos de activación. Esta extensión permite modelar de una manera más lógica el hecho de que

simultáneamente pueda haber tareas pertenecientes a diferentes instancias del mismo evento, ejecutando en el mismo o diferentes procesadores del sistema.

- También hemos desarrollado la técnica que nos permite analizar transacciones aperiódicas, para eventos con llegadas esporádicas o limitadas. Tradicionalmente, las técnicas de análisis desarrolladas para tareas periódicas eran igualmente válidas para tareas esporádicas sin más que considerar un periodo igual al intervalo mínimo entre llegadas. Sin embargo, esto no es cierto para tareas esporádicas con *offsets* mayores que el intervalo mínimo entre llegadas. En la sección 4.2.4 hemos demostrado que cuando existen tareas de este tipo no se puede construir el peor caso de igual forma a como se hace con transacciones periódicas. Debido a ello, hemos desarrollado una aproximación que nos permite analizar transacciones aperiódicas con *offsets* mayores que el periodo mínimo entre llegadas.

- El desarrollo de la formulación nos ha servido también para introducir en las ecuaciones una notación diferente, que nos ha sido muy útil al explotar en el análisis las relaciones de precedencia en la activación de las tareas.

- Hemos extendido también el modelo transaccional para permitir que los *offsets* varíen dinámicamente de una activación a otra. De esta forma, hemos podido asociar *offsets* a una tarea en función de las situaciones de ejecución de tareas previas en la transacción.

Basándonos en esa nueva formulación, hemos utilizado el modelo transaccional con *offsets* dinámicos como modelo analítico aproximado para sistemas distribuidos de tiempo real, como alternativa al modelo basado en tareas independientes de Tindell y Clark. En este modelo asociamos a cada tarea de la secuencia de respuesta un *offset* equivalente al tiempo de respuesta de mejor caso de la tarea previa. También asociamos a cada tarea un término de retraso equivalente a la máxima fluctuación que puede experimentar el tiempo de respuesta de la tarea previa, igual a la diferencia entre sus tiempos de respuesta de peor y de mejor caso. Hemos implementado un algoritmo recursivo, que llamamos APCOD (Análisis de Peor Caso con *Offsets* Dinámicos), que obtiene cotas superiores de los tiempos de respuesta globales de peor caso en sistemas multiprocesadores.

En las simulaciones realizadas se ha puesto de manifiesto la ventaja de utilizar la nueva técnica que hemos desarrollado frente a las ya existente. Como se pude ver en la sección 4.5, en los ejemplos analizados se han encontrado incrementos de hasta el 25% en la máxima utilización planificable de los sistemas. También se han encontrado soluciones de los tiempos de respuesta en esos ejemplos que mejoran del orden de 2.5 veces los tiempos estimados con la técnica basada en tareas independientes. Es interesante destacar que las ventajas obtenidas son muy importantes aún considerando tiempos de respuesta de mejor caso nulos. Según esto, ni siquiera es necesario estimar o calcular tiempos de ejecución de mejor caso para mejorar los resultados del análisis.

El modelo transaccional con *offsets* dinámicos se ha demostrado también muy útil para el análisis de tareas que se suspenden, tal como hemos puesto de manifiesto en la sección 4.3. El análisis aplicado hasta ahora calculaba los tiempos de respuesta suponiendo bien secciones de código con ejecución independiente antes y después de la suspensión, o bien considerando el tiempo de suspensión como tiempo de ejecución. El cálculo con *offsets* permite asociar la suspensión a la sección de código posterior a la suspensión y de esta forma, eliminar prácticamente todo el pesimismo del análisis.

Dentro del objetivo de mejorar el análisis para sistemas distribuidos, en el capítulo 5 hemos optimizado el método basado en tareas con *offsets* dinámicos al considerar algunas propiedades debidas a las relaciones de precedencia. En primer lugar, hemos explotado el hecho de que las relaciones de precedencia impiden a una tarea interferir la ejecución de otra tarea perteneciente a su misma secuencia si esta última precede a la anterior y pertenece a la misma o previas instancias del evento. Este hecho supone una modificación mínima sobre el algoritmo de *offsets* dinámicos previamente diseñado y a cambio obtiene grandes ventajas, tal como puede verse en la sección 5.2.1.

Una revisión más profunda es la que realizamos en la sección 5.3. En ella, se explotan las características inherentes a la precedencia, al considerar los perfiles de prioridad asignados a las distintas secuencias de respuesta dentro del sistema. Esto permite definir y estudiar de una forma mucho más precisa las posibles situaciones de peor caso en el funcionamiento real del sistema. La revisión viene dada por la definición de *Conflictos de activación*. Según ella, dos tareas pertenecientes a una misma secuencia y activadas en la misma instancia del evento

no pueden interferir simultáneamente la ejecución de una tercera tarea si entre ellas existe una tarea con menor prioridad. La aplicación de este hecho, que es simple en concepto, supone la reformulación de las ecuaciones derivadas para el análisis de tareas con *offsets*, aunque de una complejidad similar a la que ya tenían. El nuevo análisis desarrollado es siempre más preciso (en ocasiones bastante), y además más eficiente que el anterior, puesto que elimina el estudio de ciertas situaciones incompatibles, a través de la definición de ciertas *reglas de reducción*. La simulaciones realizadas para comprobar las mejoras introducidas muestran resultados notables, llegando a obtener mejoras relativas del 60% en cuanto a la máxima utilización planificable y mejoras de hasta un factor de 10 en los tiempos de respuesta encontrados con este último método frente a los obtenidos con el método de Tindell y Clark, en condiciones que podríamos considerar normales en los sistemas de tiempo real estudiados.

La determinación del tiempo de respuesta con el método basado en tareas independientes se basa en aplicar iterativamente ecuaciones — ver ecuación (40) del capítulo 2 — cuya complejidad algorítmica es de tipo $O(m \times n)$, donde m es el número de eventos y n el número máximo de tareas en las secuencias de respuesta a eventos. Con los métodos basados en *offsets*, los algoritmos iterativos aplican ecuaciones que conllevan una complejidad algorítmica $O(m \times n^2)$ — ver ecuación (32) del capítulo 4 y ecuación (46) del capítulo 5. A cambio, las soluciones encontradas con nuestros método son *siempre* mejores (en el peor caso iguales) que las estimadas con el método de Tindell. Esta complejidad añadida no supone gran incremento en el tiempo de computo requerido para analizar los sistemas de tiempo real. Por ejemplo, las gráficas presentadas en los capítulos 4 y 5 requieren el análisis de varios cientos o miles de sistemas simulados, y para ello se ha necesitado desde varios minutos a pocas horas, dependiendo de los parámetros de simulación empleados.

También es destacable el uso que se puede hacer del análisis basado en tareas con *offsets* dinámicos y perfiles de prioridad para el cálculo de tiempos de respuesta de tareas con prioridades de ejecución variantes. El método es menos preciso que el desarrollado por Gonzalez Harbour, Klein y Lehoczky para el caso de sistemas monoprocesadores, aunque es capaz de dar soluciones muy aproximadas, incluso cuando los restantes métodos no son capaces de obtener tiempos de respuesta acotados (como se refleja en los resultados mostrados en la sección 5.3.3). Sin embargo, nuestra técnica se revela muy recomendable para analizar

directamente tareas con prioridades de ejecución variantes en sistemas multiprocesadores, donde no se puede aplicar el método original.

6.2. Contribuciones de este trabajo

El aspecto más importante del trabajo presentado en esta Tesis Doctoral es el desarrollo de nuevas técnicas de análisis para sistemas distribuidos de tiempo real estricto con planificación basada en prioridades fijas. Como resumen de las contribuciones destacables, dentro de este trabajo, al análisis de los sistemas de tiempo real, podemos citar las siguientes:

- Se ha demostrado la validez del método basado en tareas independientes, desarrollado por Tindell y Clark, para calcular los tiempos de respuesta globales de peor caso de tareas que ejecutan en sistemas multiprocesadores y distribuidos. Esta validación es ciertamente importante, puesto que muchas aplicaciones reales ya se han desarrollado utilizando esa técnica como medio de verificación de los requerimientos temporales.

- Se ha introducido el cálculo de los tiempos de respuesta de mejor caso como un paso recomendable en el cálculo de los tiempos de respuesta de peor caso.

- Se han desarrollado técnicas que permiten la acotación más precisa de los tiempos de respuesta locales

- Se han extendido las técnicas de análisis para transacciones con *offsets* estáticos al caso de tareas con *offsets* mayores que los correspondientes periodos de activación. También se han extendido para tareas esporádicas con *offsets*, especialmente para el caso de *offsets* mayores que el tiempo mínimo entre llegadas. Estas técnicas son directamente aplicables en sistemas planificados mediante la Técnica de Modificación de Fase [BET92], con servidores esporádicos [GUT95A], o mediante el algoritmo de Prioridades Duales [DVI94A].

- Hemos extendido el modelo y análisis de tareas con *offsets* estáticos para permitir el caso de *offsets* que varíen dinámicamente. A partir de esa formulación hemos desarrollado un método para realizar el análisis de sistemas multiprocesadores y

distribuidos con unos resultados que mejoran notablemente los resultados obtenidos hasta ahora.

- Nuestras técnicas de análisis permiten analizar sistemas en los que existan tareas que se suspenden, de una manera más precisa que las técnicas existentes hasta la fecha.

- Hemos optimizado nuestra técnica basada en tareas con *offsets* dinámicos, para explotar las relaciones de precedencia, así como los perfiles de prioridad asignados a las tareas de las diferentes transacciones. Estas últimas consideraciones nos permite eliminar gran parte del pesimismo existente en las técnicas de análisis, de forma que el nuevo método implementado mejora notablemente las ya notables mejoras del método anterior. Es de resaltar el hecho de que nuestra técnica mejora *siempre* y en cualquier caso las soluciones obtenidas mediante cualquiera de las otras técnicas desarrolladas.

- Las técnicas de análisis desarrolladas en esta tesis pueden aplicarse directamente para analizar sistemas de tiempo real con tareas con prioridades de ejecución variantes. Especialmente es interesante el caso de sistemas multiprocesadores y distribuidos, donde la técnica original no tiene aplicación.

- Las técnicas son fácilmente modificables para analizar sistemas con planificadores no expulsores.

6.3. Trabajo futuro

Evidentemente, cualquier técnica aproximada es susceptible de mejora. En este caso, las técnicas desarrolladas en esta tesis eliminan gran parte del pesimismo existente en las técnicas previas de análisis, pero no todo. Una posible línea de investigación futura podría ser la de identificar y eliminar alguna (o la totalidad) de las fuentes de pesimismo que aún tiene el análisis. Las principales fuentes de pesimismo que aún persisten en el análisis son debidas a que se trata la ejecución de diferentes secuencias de forma independiente y, principalmente, a que la técnica se basa en el análisis por separado de cada recurso. Esa técnica debería

contemplar seguramente situaciones más complejas y globalizadoras del comportamiento total del sistema, o al menos, mejores aproximaciones que contemplen esos aspectos.

Otra línea de trabajo es la de incorporar las técnicas desarrolladas a herramientas de análisis y planificación previamente desarrolladas. Es nuestra intención integrar la nueva formulación para el cálculo de tiempos de respuesta al algoritmo *HOPA* [GUT95D] de planificación de sistemas distribuidos de tiempo real. Pensamos que es en algoritmos de este estilo donde nuestras técnicas de análisis pueden tomar mayor ventaja frente a las ya existentes.

También puede ser interesante extender nuestra técnica de análisis a sistemas de tiempo real basados en prioridades dinámicas. Aunque la formulación y ejecución sean diferentes en este tipo de sistemas, la filosofía (agrupamiento de tareas en transacciones, precedencia en una misma secuencia de respuesta, etc.) continua siendo válida y con posibilidades de éxito.

Una línea con muchas probabilidades de éxito sería la de utilizar el modelo transaccional al *modelo generalizado multimarco* (*multiframe*) de Mok y Baruah [MOK96]. Este modelo, en su versión mas generalizada, es equivalente al modelo transaccional con *offsets* dinámicos, de forma que podemos aplicar nuestra técnica de análisis directamente al análisis de ese tipo de sistemas. En su versión original, el modelo es más sencillo, y nos parece que en ese caso, se podría obtener una versión optimizada y mas eficiente de nuestros algoritmos de análisis. Además, nuestra técnica ya es válida en sistemas multiprocesadores y distribuidos, de forma que se podría utilizar para contemplar el análisis de tareas multiframe en sistemas multiprocesadores o distribuidos.

Bibliografía

[ADA83] "Reference Manual for the ADA Programming Language". ANSI/MIL-STD 1815 A.

[ADA93] "Ada 9X Rationale". Ada 9X Project, Department of Defense, USA, Draft 4.0, September 1993.

[ADA95] International Standard ISO/IEC 8652:1995(E): "Information technology - Programming languajes - Ada. Ada Reference Manual".1995.

[AGR91] AGRAWAL G., CHEN B., ZHAO W., and DAVARI S.: "Architecture Impact of FDDI Network on Scheduling Hard Real-Time Traffic". Workshop on Architectural Aspects of Real-Time Systems, December 1991.

[ALM90] AL-MOUHAMED M.A.: "Lower Bound on the Number of Processors and Time for Scheduling Precedence Graphs With Communication Costs". IEEE Transactions on Software Engineering, vol. 16, no. 12, pp.1390-1401, December 1990.

[ARA94] ARAS C.M., KUROSE J.F., REEVES D.S. y SCHULZRINNE H.: "Real-Time Communications in Packet-Switched Networks". Proceedings of the IEEE, Vol. 82, No. 1, January 1994.

[AUD90] AUDSLEY N.C. y BURNS A.: "Real-Time System Scheduling". Department of Computer Science, University of York, Technical Report YCS_134.

[AUD91A] AUDSLEY N.C.: "Deadline Monotonic Scheduling". Department of Computer Science, University of York, Technical Report YCS_146, September 1991.

[AUD91B] 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.

[AUD91C] AUDSLEY N.C., BURNS A., RICHARDSON M.F. y WELLINGS A.J.: "Hard Real-Time Scheduling: The Deadline Monotonic Approach". Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software, Atlanta, May 1991.

[AUD93] AUDSLEY N.C., BURNS A., RICHARDSON M., TINDELL K. y WELLINGS A.J.: "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling". Software Engineering Journal, Vol. 8, No. 5, pp. 284-292, September 1993.

[AUD94A] AUDSLEY N., BURNS A., RICHARDSON M. t WELLINGS A.: "STRESS: A Simulator for Hard Real-Time Systems". Software Practice and Experience, July 1994.

[AUD94B] AUDSLEY N.C., BURNS A., DAVIS R.I. y WELLINGS A.J.: "Integrating Best Effort and Fixed Priority Scheduling". Proceedings of the 1994 Workshop on Real-Time Programming, Lake Constance, Germany, June 1994.

[AUD96] AUDSLEY N.C., I.J.BATE y BURNS A: "Putting Fixed Priority Scheduling into Engineering Practice for Safety Critical Applications". Proceedings of the Real-Time Technology and Applications Symposium, pp. 2-10. IEEE Technical Comittee on Real-Time Systems, 1996

[BAI93] BAILEY C.M., FYFE E., VARDANEGA T. y WELLINGS A.J.: "The Use of Preemptive Priority-Based Scheduling for Space Applications". Proceedings Real-Time Systems Symposium, December 1993.

[BAI94] BAILEY C.M., BURNS A., WELLINGS A.J. y FORSYTH C. H.: "A Performance Analysis of a Hard Real-Time System". Department of Computer Science, University of York, Technical Report YCS_224, February 1994.

[BAK86] BAKER T.P. y SCALLON G.M.: "An Architecture for Real-Time Software Systems". IEEE Software, vol 3, no. 3, pp. 50-59, May 1986.

[BAK88] BAKER T.P., SHAW A.: "The Cyclic Executive Model and Ada". Proceedings of the IEEE Real-Time Systems Symposium, December 1988.

[BAK90] BAKER T.P.: "A Stack-Based Resource Allocation Policy for Realtime Processes". IEEE, CH2933-0, pp. 191-200, 1990.

[BAK91A] BAKER T.P. y RICCARDI G.A.: "Ada Tasking: From Semantics to Efficient Implementation". IEEE Software, pp. 34-46, March 1991.

[BAK91B] BAKER T.P.: "Stack-Based Scheduling of Realtime Processes". The Journal of Real-Time Systems, Kluwer Academic Publishers, 3, pp. 67-99, 1991.

[BAT97] BATE I., BURNS, A.: "Schedulability Analysis of Fixed Priority Real-Time Systems with Offsets". Proceedings of the 9th Euromicro Workshop on Real-Time Systems. Toledo, Spain, pp. 153-160. June 1997.

[BAT98] BATE I., BURNS, A.: "Investigation Of the Pessimism in Distributed Systems Timing Analysis". Proceedings of the 10th Euromicro Workshop on Real-Time Systems. Toledo, Spain, pp. 107-114. July 1998.

[BER97] BERNAT G. Y BURNS A.: "Combining (n m)-hard deadlines and dual priority scheduling". Proccedings of the 18th. Real-Time Sytems Symposium, pp. 46-57, 1997.

[BET92] BETTATI R. y LIU J.W.S.: "End-to-End Scheduling to Meet Deadlines in Distributed Systems". 12th International Conference on Distributed Systems, pp. 452-459. Japan, June 1992.

[BLA91] BLAKE B.A. y SCHWAN K.: "Experimental Evaluation of Real-Time Scheduler for a Multiprocessor System". IEEE Transaction on Software Engineering, vol. 17, no. 1, pp. 34-44, January 1991.

[BOO87] BOOCH G.: "Software Components with Ada. Structures, Tools and Subsystems". The Benjamin/Cummings Publishing Company, Inc, 1987.

[BOR89A] BORGER M.W., KLEIN M.H. y VELTRE R.A.: "Real-Time Software Engineering in Ada: Observations and Guidelines". Carnegie Mellon University, Sofware Engineering Institute, Tech. Rept. CMU/SEI-89-TR-22, ESD-89-TR-30, September 1989.

[BOR89B] BORGER M.W. y RAJKUMAR R.: "Implementing Priority Inheritance Algorithms in an Ada Runtime System". Carnegie Mellon University, Sofware Engineering Institute, Tech. Rept. CMU/SEI-89-TR-15, ESD-89-TR-23, April 1989.

[BUR89] BURNS A. y WELLINGS A.J.: "Real-Time Ada: Outstanding Problem Areas". Third International Real Time Ada Workshop, Conference Report. Ada User Vol.10, no. 3, pp.141-142, July 1989.

[BUR90] BURNS A. y WELLING A.J.: "Real-Time Systems and Their Programming Languages". Addison Wesley Publishing Company, 1990.

[BUR91] BURNS A. y WELLINGS A.J.: "Priority Inheritance and Message Passing Communicaton: A Formal Treatment". The Journal of Real-Time Systems, Kluwer Academic Publishers, 3, pp. 19-44, 1991.

[BUR92] BURNS A. y WELLINGS A.J.: "Designing Hard Real-Time Systems". Proceedings 11th Ada Europe conference (1992), Springer-Verlag Lecture Notes in Computer Science 603.

[BUR93A] BURNS A.: "Preemptive Priority Based Scheduling: An Appropriate Engineering Approach". Department of Computer Science, University of York, Technical Report YCS_214, November 1993.

[BUR93B] BURNS A., NICHOLSON M., TINDELL K. y ZHANG N.: "Allocating and Scheduling Hard Real-Time Tasks on a Point-to-Point Distributed System". Proceedings of the Workshop on Parallel and Distributed Real-Time Systems, pp. 11-20, April 13-15, 1993.

[BUR93C] BURNS A. y WELLINGS A.J.: "Implementing Analysable Hard Real-time Sporadic Tasks in Ada 9X". Department of Computer Science, University of York, Technical Report YCS_209, September 1993.

[BUR94A] BURNS A. y WELLINGS A.J.: "HRT-HOOD: A Structured Design Method for Hard Real-Time Systems". Real-Time Systems, 6, pp. 73-114, 1994.

[BUR94B] BURNS A., TINDELL K. y WELLINGS A.J.: "Fixed Priority Scheduling with Deadlines Prior to Completion". Proceedings Sixth Euromicro workshop on Real-Time Systems, Vaesteraas, Sweden, pp. 138-142, June 1994.

[COO83] COOLAHAN J.E. y ROUSSOPOULOS N.: "Timing Requirements for Time-Driven Systems Using Augmented Petri Nets". IEEE Trans. on Software Eng. Vol. SE-9, No. 5, pp. 603-616 , September 1983.

[COO85] COOLAHAN J.E. y ROUSSOPOULOS N.: "A Timed Petri Net Methodology for Specifying Real-Time System Timing Requirements". International Workshop on Timed Petri Nets. IEEE Catalog number 85CH2187-3, pp. 24-31, July 1985.

[CHA93] CHAPMAN R., BURNS A. y WELLINGS A.J.: "Worst-case Timing Analysis of Exception Handling in Ada". Proceedings Ada UK conference, London, 1993.

[CHA94] CHAPMAN R., BURNS A. y WELLINGS A.J.: "Integrated Program Proof and Worst-Case Timing Analysis of SPARK Ada". Proceedings 1994 ACM Workshop on Language, Compiler, and Tool Support for Real-Time Systems, Florida, June 1994.

[CHE90] CHEN M. y LIN K.: "Dynamic Priority Ceilings: A Concurrency Control Protocol for Real-Time Systems". Real-Time Systems, Kluwer Academic Publishers, 2, pp. 325-346, 1990.

[CHB98] CHEN J. y BURNS A.: "Asynchronous Data Sharing in Multiprocessor Real-Time Systems Using Process Consensus". Proceeding of the 10th Euromicro Workshop on Real-Time Systems, Berlin, Germany, June 1998.

[CHT90] CHETTO H., SILLY M. y BOUCHENTOUF T.: "Dynamic Scheduling of Real-Time Tasks under Precedence Constraints". Real-Time Systems, vol. 2, no. 3, pp. 181-194, September 1990.

[DAV86] DAVARI S. y DHALL S.K.: "An on Line Algorithm for Real-Time Tasks Allocation". IEEE, CH2351-5, 1986. Procedente: Real-Time System Symphosium, December 2-4, 1980.

[DIJ72] E.W.DIJISTRA: "Notes on Structured Programming". Structured Programming, O.J.Dahl, E.W.Dijistra y C.A.R. Hoare, Eds, London:Academic Press, 1972.

[DVI93A] DAVIS R.I.: "Scheduling Slack Time in Fixed Priority Pre-emptive Systems". Department of Computer Science, University of York, Technical Report YCS_216, November 1993.

[DVI93B] DAVIS R.I.: "Approximate Slack Stealing Algorithms for Fixed Priority Pre-emptive Systems". Department of Computer Science, University of York, Technical Report YCS_217, November 1993.

[DVI94A] DAVIS R.I.: "Dual Priority Scheduling: A Means of Providing Flexibility in Hard Real-time Systems". Department of Computer Science, University of York, Technical Report YCS_230, May 1994.

[DVI94B] DAVIS R.I.: "Guaranteeing X in Y: On-line Acceptance Tests for Hard Aperiodic Tasks Scheduled by the Slack Stealing Algorithm". Department of Computer Science, University of York, Technical Report YCS_231, May 1994.

[GAF91] GAFFORD J.D.: "Rate Monotonic Scheduling". IEEE Micro, pp. 34-38 y 102, June 1991.

[GEH91] GEHANI N.: "Ada Concurrent Programming". Prentice Hall, 1991.

[GIE89] GIERING III E.W. y BAKER T.P.: "Toward the Deterministic Scheduling of Ada Tasks". IEEE, CH2803-5, 1989.

[GOM84] GOMAA H.: "A Software Design Method for Real-Time Systems". Communications of the ACM, Vol. 27, No. 9, September 1984.

[GOM93] GOMAA H.: "Software Design Methods for Concurrent and Real-Time Systems". Ed. Addison-Wesley, 1993..

[GON91A] GONZALEZ HARBOUR M., KLEIN M.H. y LEHOCZKY J.P.: "Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority". IEEE Real-Time Systems Symposium, 1991.

[GON91B] GONZALEZ HARBOUR M. y SHA L.: "An Application-Level Implementation of the Sporadic Server". Technical Report CMU/SEI-91-TR-26, ESD-91-TR-26, September 1991.

[GON94] GONZALEZ HARBOUR M., KLEIN M.H. y LEHOCZKY J.P.: "Timing Analysis for Fixed-Priority Scheduling of Hard Real-Time Systems". IEEE Transaction on Software Engineering, Vol. 20, no. 1, January 1994.

[GON97] GONZALEZ HARBOUR M., GUTIERREZ GARCIA J.J. y PALENCIA GUTIERREZ J.C.: "Implementing Application-Level Sporadic Server Schedulers in Ada 95". Lecture Notes in Computer Science 1251, Reliable Software Technologies. Ada-Europe'97, pp. 125-136.

[GON98A] GONZALEZ HARBOUR M., GUTIERREZ GARCIA J.J. y PALENCIA GUTIERREZ J.C.: "Implementación en Ada 95 de Servidores Esporádicos en el Nivel de Aplicación". SpAda Boletín n. 35, pp. 19-28, Febrero 1998.

[GON98B] GONZALEZ HARBOUR M., ALDEA RIVAS, M., GUTIERREZ GARCIA J.J. y PALENCIA GUTIERREZ J.C.: "Implementing and Using Execution Time Clocks in Ada Hard Real-time Applications". Lecture Notes in Computer Science 1411, Reliable Software Technologies. Ada-Europe'98, pp. 90-101.

[GOO88] GOODENOUGH J.B. y SHA L.: "The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High Priority Ada Task". Proceeding of the 2th International Workshop on Real-Time Ada Issues, June 1988.

[GRA89] GRANDA M. y DRAKE J.M. (Director): "Métodos para la Evaluación de Sistemas Multiprocesadores y su Aplicación a la Programación de Sistemas de Tiempo Real Mediante Prototipos Rápidos". Tesis Doctoral, Universidad de Cantabria, 1989.

[GUT91] GUTIERREZ GARCIA J.J., GREGORIO MONASTERIO J.A. y GONZALEZ HARBOUR M.: "Sistema de Control de un Robot para la Medida de Excentricidades de Tubos en el Interior de una Central Nuclear". Tesina de Licenciatura, Junio de 1991.

[GUT94] GUTIERREZ GARCIA J.J., GONZALEZ HARBOUR M. y PALENCIA GUTIERREZ J.C.: "Planificación y análisis de sistemas distribuidos de tiempo real estricto". Primeras Jornadas de Sistemas de Tiempo Real, Málaga, Octubre de 1994.

[GUT95A] GUTIERREZ GARCIA J.J. y GONZALEZ HARBOUR M.: "Optimized Priority Assignment for Tasks and Messages in Distributed Real-Time Systems". Proceedings of the 3rd Workshop on Parallel and Distributed Real-Time Systems, Santa Barbara, California, pp. 124-132, April 1995.

[GUT95B] GUTIERREZ GARCIA J.J. y GONZALEZ HARBOUR M.: "Increasing Schedulability in Distributed Hard Real-Time Systems". Proceedings of the 7th Euromicro Workshop on Real-Time Systems, Odense, Denmark, pp. 99-106, June 1995.

[GUT95C] GUTIERREZ GARCIA J.J. y GONZALEZ HARBOUR M.: "Aumentando la Planificabilidad de Sistemas Distribuidos de Tiempo Real Estricto". IV Jornadas de Concurrencia, El Escorial, Junio de 1995.

[GUT95D] GUTIERREZ GARCIA J.J. y GONZALEZ HARBOUR M. (Director): "Planificación, análisis y optimización de sistemas distribuidos de tiempo rel estricto". Tesis Doctoral, Universidad de Cantabria, 1995.

[HAB90] HABAN D. y SHIN K.G.: "Application of Real-Time Monitoring to Scheduling Tasks with Random Execution Times". IEEE Transactions on Software Engineering, vol. 16, no. 12, pp.1374-1389, December 1990.

[HAR84] HARTER P.K: "Response times in level-structured systems". ACM Transactions on Computer Systems, vol. 5, no. 3, pp. 232-248, Aug. 1984.

[JOS86] JOSEPH M. y PANDYA P.: "Finding Response Times in a Real-Time System". The Computer Journal (British Computing Society) 29, 5, pp. 390-395, October 1986.

[JOU95] M. JOURDAN y F. MARANINCHI: "Static Timing Analysis of Real_Time Systems". ACM SIGPLAN Workshop on Languages, Compilers and Tools for Real-Time Systems. California,1995.

[KIR83] KIRKPATRICK S., GELATT C.D. y VECCHI M.P.: "Optimization by Simulated Annealing". Science, vol. 220, no. 4598, pp. 671-680, May 1983.

[KLE90] KLEIN M. y RALYA T.: "An Analysis of Input/Output Paradigms for Real-Time Systems". Software Engineering Institute, Tech. Rept. CMU/SEI-90-TR-19, July 1990.

[KLE93] KLEIN M., RALYA T., POLLAK B., OBENZA R. y GONZALEZ HARBOUR M.: "A Practitioner's handbook for Real-Time Analisys". Kluwer Academic Pub., 1993.

[LAI86] LAIRD J.D., BURTON B.A. y KOPPES M.R.: "Implementation of an Ada Real-Time Executive - A Case Study". Annual National Conference on Ada Technology 1986, pp. 114-124.

[LEH86] LEHOCZKY J.P. y SHA L.: "Performance of Real-Time Bus Scheduling Algorithms". ACM Performance Evaluation Review, Special Issue 14, 1, May 1986.

[LEH87] LEHOCZKY J.P., SHA L. y STROSNIDER J.K.: "Enhanced Aperiodic Scheduling in Hard Real-Time Environments". Proceedings of the IEEE Real-Time Systems Symposium, Los Alamitos, CA, IEEE Computer Society Press, pp. 261-270, 1987.

[LEH89] LEHOCZKY J., SHA L. y DING Y.: "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior". IEEE Real-Time Systems Symp., CS Press, Los Alamitos, CA, pp. 166-171, 1989.

[LEH90] LEHOCZKY J.P.: "Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines". IEEE Real-Time Systems Symposium, 1990.

[LEH91] LEHOCZKY J., SHA L., STROSNIDER, J. y TOKUDA, H.: "Fixed Priority Scheduling Theory for Hard Real-Time Systems". 1-30 van Tilbury y Koob (ed.) Foundations of Real-Time Computing: Scheduling and Resource Management. Boston,MA: Kluwer Academic Publishers, 1991.

[LEH92] LEHOCZKY J., y RAMOS-THUEL, S.: "An optimal algorithm for scheduling soft-aperiodics tasks in fixed preemptive systems". Proc. 13th IEEE Real-Time System Symposium, pages 110-123, Phoenix, Arizona, december 1992.

[LEI82] LEINBAUGH D.W. y YAMINI MR.: "Guaranteed Response Times in a Distributed Hard-Real-Time Environment". IEEE, 1982. Procedente: Real-Time System Symphosium, December 7-9, 1982.

[LEU82] LEUNG J. y WHITEHEAD J.: "On Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks". Performance Evaluation, 2, pp. 237-250, 1982.

[LEV87] LEVESON N.G. y STOLZY J.L.: "Safety Analysis Using Petri Nets". IEEE Trans. on Software Eng. Vol. SE-13, No. 3, pp. 386-397 , March 1987.

[LIM95] LIM S.S., BAE Y.H., JANG G.T., RHEE B.D., MIN S.L., PARK C.Y., SHIN H., PARK K. MOON S.M., KIM C.S.: "An Accurate Worst Case Timing Analysis for RISC processors". IEEE Transactions on Software Engineering, 21(7), pp. 593-604.

[LIU73] 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.

[LIU91] LIU J.W.S., LIN KJ., SHIH WK. y YU A.C.: "Algorithms for Scheduling Imprecise Computations". IEEE Computer, pp. 58-68, May 1991.

[LOC85] LOCKE C.D., JENSEN E.D. y TOKUDA H.: "A Time-Driven Scheduling Model for Real-Time Operating Systems". Proceedings of Real-Time System Symposium, pp 112-122.

[LOC92] LOCKE C.D.: "Software Architecture for Hard Real-Time Applications: Cyclic Executives vs. Fixed Priority Executives". The Journal of Real-Time Systems, 4, pp. 37-53, 1992.

[LVI88] LEVINE G.: "The Control of Priority Inversion in Ada". Ada Letters, Vol. VIII, No. 6, November/December 1988.

[MAN95] MANABE Y. y AOYAGI S.: "A Feasibility Algorithm for Rate Monotonic Scheduling of Periodic Real-Time Tasks". Proceedings Real-Time Technology and Applications Symposium, pp. 212-218.

[MAR89] MARINESCU D. y STANSIFER. R.: "Verification of Timing Correctness of Real-Time Programs". Software Engineering Research Center report no. SERC-TR-57-P, Purdue University, November 1989.

[MCE94] McELHONE C. : "Adapting and Evaluating Algorithms for Dynamic Schedulability Testing". Department of Computer Science, University of York, Technical Report YCS_225, February 1994.

[MED94] MEDIAVILLA M.E., DRAKE J.M. y GRANDA M. (Codirectores): "Modelado y Evaluación de Programas Paralelos con Distintas Estrategias de Planificación de Procesos". Tesis Doctoral, Universidad de Cantabria, 1994.

[MEN93] MENENDEZ DEL LLANO R. y DRAKE J.M. (Director): "Diseño y Evaluación de un Método de Programación en Ada para Sistemas Multicomputadores".Tesis Doctoral, Universidad de Cantabria, 1993.

[MIL91] MILLARD B.R., MILLER D.S. y WU C.: "Support for Ada Intertask Communication in a Message-Based Distributed Operating System". IEEE, CH2859-5, 1991.

[MOK96] MOK A. K. y CHEN D.: " A multiframe model for real-time tasks". Proccedings of the 17th. Real-Time Systems Symposium, Washington, 1996.

[NIE88] NIELSEN K. y SHUMATE K.: "Designing Large Real-Time Systems with Ada". Mc. Graw Hill, 1988.

[NIE90] NIELSEN K.: "Ada in Distributed Real-Time Systems". Intertext Publications, Mc. Graw Hill Book Company, 1990.

[PAR91] PARK C.Y. y SHAW A.C.: "Experiments with a Program Timing Tool Based on Source-Level Timing Schema". IEEE Computer, pp. 48-56, May 1991.

[PAL97] PALENCIA GUTIERREZ J.C., GUTIERREZ GARCIA J.J., GONZALEZ HARBOUR M. : "On the Schedulability Analysis for Distributed Hard Real-Time Systems". Proceeding of the 9th IEEE Euromicro Workshop on Real-Time Systems, Toledo, Spain, June 1997 pp. 136-143.

[PAL98A] PALENCIA GUTIERREZ J.C., GUTIERREZ GARCIA J.J., GONZALEZ HARBOUR M. : "Best-Case Analysis for Improving the Worst-Case Schedulability Test for Distributed Hard Real-Time Systems". Proceeding of the 10th IEEE Euromicro Workshop on Real-Time Systems, Berlin, Germany, June 1998.

[PAL98B] PALENCIA GUTIERREZ J.C., GONZALEZ HARBOUR M. : "Schedulability Analysis for Tasks with Static and Dynamic Offsets". Proceedings of the 18th. IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.

[PLE92] PLEINEVAUX P.: "An Improved Hard Real-Time Scheduling for the IEEE 802.5". The Journal of Real-Time Systems, 4, pp. 99-112, 1992.

[POS93] IEEE Std. 1003.1b: "POSIX (Portable Operating System Interface)". IEEE Standard for Information Technology, 1993.

[PUS89] PUSCHNER P. y KOZA CH.: "Calculating the Maximum Execution Time of Real-Time Programs". The Journal of Real-Time Systems, Kluwer Academic Publishers, 1, pp. 159-176, 1989.

[RAJ88] RAJKUMAR R., SHA L. y LEHOCZKY J.P.: "Real-Time Synchronization Protocols for Multiprocessors". IEEE Real-Time Systems Symposium, CS Press, Los Alamitos, Calif., December 1988.

[RAJ89] RAJKUMAR R: "Task Synchronization in Real-Time Systems". Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University. August 1989.

[RAJ90] RAJKUMAR R.: "Real-Time Synchronization Protocols for Shared Memory Multiprocessors". Proceedings of The 10th International Conference on Distributed Computing Systems, 1990.

[RAM80] RAMAMOROORTHY C.V. y GARY S.H.: "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets". IEEE Trans. on Software Eng. Vol. SE-6, No. 5, pp. 440-449 , September 1980.

[RIP96] RIPOLL I. y CRESPO A.(director): "Planificación con Prioridades Dinámicas en Sistemas de Tiempo Real Crítico". Tesis doctoral, Valencia, 1996.

[RMM90] RAMAMRITHAM K., STANKOVIC J.A. y SHIAH P.: "Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems". IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 2, pp. 184-194, April 1990.

[SAV85] SAVITZKY S.R.: "Real-Time Microprocessor Systems". Van Nostrand Reinhold Company, 1985.

[SCM94] SCHMID U.: "Monitoring Distributed Real-Time Systems". Real-Time Systems, 7, pp. 33-56, 1994.

[SCH94] SCHOLEFIELD D.J.: "Proving Properties of Real-Time Semaphores". Department of Computer Science, University of York, Technical Report SCP_S_94, 1994.

[SCH94] SCHOLEFIELD D.J., ZEDAN H.S.M. y HE J.: "A Specification Oriented Semantics for The Refinement of Real-Time Systems". Department of Computer Science, University of York, Technical Report TCS_SZH, 1994.

[SHA89] SHA L., RAJKUMAR R., LEHOCZKY J.P. y RAMAMRITHAM K.: "Mode Change Protocols for Priority-Driven Preemptive Scheduling". Real-Time Systems, Kluwer Academic Publishers, vol. 1, no. 3, pp. 243-264, December 1989.

[SHA90A] SHA L., RAJKUMAR R. y LEHOCZKY J.P.: "Priority Inheritance Protocols: An Approach to Real-Time Synchronization". IEEE Transaction on Computers, vol. 39, no. 9, pp. 1175-1185, September 1990.

[SHA90B] SHA L. y GOODENOUGH B.: "Real-Time Scheduling Theory and Ada". IEEE Computer, pp. 53-62, April 1990.

[SHA91A] SHA L., RAJKUMAR R. y LEHOCZKY J.P.: "Real-Time Computing with IEEE Futurebus+". IEEE Micro, pp. 30-33 y 95-100, June 1991.

[SHA91B] SHA L., KLEIN M.H. y GOODENOUGH J.B.: "Rate Monotinic Analisys for Real-Time Systems". In Foundations of Real-Time Computing: Scheduling and Resource Management. Van Tilborg, Andre and Koob, Gary M., Ed. Kluwer Academic Piblishers, pp. 129-155, 1991.

[SHA92] SHA L. y SATHAYE S.S.: "Distributed Real-Time System Design using Generalized Rate Monotonic Theory". 2nd ICARCV International Conference, Vol. 3, pp. 5.5.1-7, 1992.

[SHA93] SHA L. y SATHAYE S.S.: "A Systematic Approach to Designing Distributed Real-Time Systems". IEEE Computer, 0018-9162, pp. 68-78, 1993.

[SHT88] SHATZ S.M. y CHENG. W.K.: "A Petri Net Framework for Automated Static Analysis of Ada Tasking Behavior". The Journal of Systems and Software, no. 8, pp. 343-359, 1988.

[SPI90] SPIVEY J.M.: "Specifying a Real-Time Kernel". IEEE Software, pp. 21-28, September 1990.

[SPR88] SPRUNT B., LEHOCZKY J. y SHA L.: "Exploiting Unused Periodic Time For Aperiodic Service Using The Extended Priority Exchange Algorithm". IEEE, pp 251-258, 1988.

[SPR89A] SPRUNT B., SHA L. y LEHOCZKY J.P.: "Aperiodic Task Scheduling for Hard-Real-Time Systems". The Journal of Real-Time Systems, Kluwer Academic Publishers, 1, pp. 27-60, 1989.

[SPR89B] SPRUNT B., SHA L. y LEHOCZKY J.P.: "Scheduling Sporadic and Aperiodic Events in a Hard Real-Time System". Carnegie Mellon University, Sofware Engineering Institute, Tech. Rept. CMU/SEI-89-TR-11, ESD-89-TR-19, April 1989.

[SPR90A] SPRUNT B. y SHA L.: "Implementing Sporadic Servers in Ada". Technical Report CMU/SEI-90-TR-6, 1990.

[SPR90B] SPRUNT B.: "Aperiodic Task Scheduling for Real-Time Systems". Ph.D. Thesis, Carnegie Mellon University, August 1990.

[STA87] STANKOVIC J.A. y RAMAMRITHAM K.: "The Design of the Spring Kernel". IEEE, pp. 371-384, 1987.

[STA88] STANKOVIC J.A. y RAMAMRITHAM K.: "Hard Real-Time Systems". IEEE Computer Society, catalog no. EH0276-6, 1988.

[STA90] STANKOVIC J.A. y RAMAMRITHAM K.: "What is Predictability for Real-Time Systems?". Real-Time Systems, Kluwer Academic Publishers, 2, pp. 247-254, 1990.

[STA91] STANKOVIC J.A. y RAMAMRITHAM K.: "The Spring Kernel: A New Paradigm for Real-Time Systems". IEEE, pp. 62-72, May 1991.

[STO77] STONE H.S.: "Multiprocessor Scheduling with the Aid of Network Flow Algorithms". IEEE Transaction on Software Engineering, vol. SE-3, no. 1, pp. 85-93, January 1977.

[STR88] STROSNIDER J.K., MARCHOK T., LEHOCZKY J.P.: "Advanced real-time scheduling using the IEEE 802.5 Token Ring", Proceedings of the IEEE Real-Time Systems Symposium, Huntsville, Alabama, USA, 1988, pp. 42-52.

[STT85] STOTTS P.D.: "Hierarchical Modeling of Software Systems with Timed Petri Nets". IEEE Catalog number 85CH2187-3, pp. 32-39, july 1985.

[STY94] STOYENKO A.D. y BAKER T.P.: "Real-Time Schedulability-Analizable Mechanisms in Ada9X". Proceedings of the IEEE, Vol. 82, No. 1, January 1994.

[SUN95] SUN J. y LIU J.W.S.: "Bounding the End-to-End Response Time in Multiprocessor Real-Time Systems". Proceedings of the 3rd Workshop on Parallel and Distributed Real-Time Systems, Santa Barbara, California, pp. 91-98, April 1995.

[SUN96] SUN J. y LIU J.W.S.: "Synchronization Protocols in Distributed Real-Time Systems". Proceedings of the 16th International Conference on Distributed Systems, May 1996.

[TIN91] TINDELL K., BURNS A. y WELLINGS A.J.: "Guaranteeing Hard Real Time End-to-End Communications Deadlines". Department of Computer Science, University of York, Technical Report RTRG_91_107, December 1991.

[TIN92A] TINDELL K.W., BURNS A., and WELLINGS A.J., "Allocating Real-Time Tasks. An NP-Hard Problem Made Easy". Real-Time Systems Journal, Vol.4, no.2, May 1992, pp. 145-166.

[TIN92B] TINDELL K.: "An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks". Department of Computer Science, University of York, Technical Report YCS_189, December 1992.

[TIN93A] TINDELL K. y BURNS A.: "Scheduling Hard Real-Time Multi-Media Disk Traffic". Department of Computer Science, University of York, Technical Report YCS_204, 1993.

[TIN93B] TINDELL K.: "Fixed Priority Scheduling for Hard Real-Time Systems". DPhil Thesis, Department of Computer Science, University of York, 1993.

[TIN93C] TINDELL K., BURNS A. y WELLINGS A.J.: "On Limitations of the Analysis Presented in the Paper 'Engineering and Analysis of Fixed Priority Schedulers' by Katcher et al". Department of Computer Science, University of York, Technical Report RTRG_93_101, December 1993.

[TIN93D] TINDELL K.: "An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks". Journal of Real-Time Systems, Vol. 6, No. 2, March 1993.

[TIN94A] TINDELL K. y BURNS A.: "Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Networks". Department of Computer Science, University of York, Technical Report YCS_229, May 1994.

[TIN94B] TINDELL K.: "Adding Time-Offsets To Schedulability Analysis". Department of Computer Science, University of York, Technical Report YCS_221, January 1994.

[TIN94C] TINDELL K., BURNS A. y WELLINGS A.J.: "Analysis of Hard Real-Time Communications". Department of Computer Science, University of York, Technical Report RTS_TBW_Comms, 1994.

[TIN94D] TINDELL K. y BURNS A.: "Guaranteeing Message Latencies on Controller Area Network (CAN)". Proceedings 1st International CAN Conference, Mainz, Germany, September 1994.

[TIN94E] TINDELL K., BURNS A. y WELLINGS A.J.: "Calculating Controller Area Network (CAN) Message Response Times". Proceedings 1994 IFAC workshop on Distributed Computer Control Systems (DCCS), Toledo, Spain, September 1994.

[TIN94F] TINDELL K. y CLARK J.: "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems". Microprocessing & Microprogramming, Vol. 50, Nos. 2-3, pp. 117-134, April 1994.

[TOR92] TORON ANTONS B.: "Diseño e Implementación de una Herramienta para el Análisis de Sistemas de Tiempo Real Estricto". Tesina de Licenciatura, Septiembre de 1992.

[VAL90] VALLEJO F., GREGORIO J.A., GONZALEZ HARBOUR M. y DRAKE J.M.: "Shared Memory Multimicroprocessor Operating System with an Extended Petri Net Model". IEEE Trans. on Parallel and Distributed Systems, Vol. 5, No. 7, July 1994.

[VAL91] VALLEJO F. y GREGORIO J.A.(Director): "Diseño y Evaluación de un Entorno de Programación para Sistemas Multicomputadores Fuertemente Acoplados". Tesis Doctoral, Universidad de Cantabria, 1991.

[VOL85] VOLZ R.A., MUDGE T.N., NAYLOR A.W. y MAYER J.H.: "Some Problems in Distributing Real Time Ada Programs Across Machine". Ada in Use, Proceeding of the Ada International Conference, pp. 72-84, Paris, 1985.

[WAN93] WANG J.P. y STEIDLEY C.: "Task Allocation Model for Minimization of Completion Time in Distributed Computer Systems". SPIE Vol. 2056 Intelligent Robots and Computer Vision XII, pp. 430-436, 1993.

[WED94] WEDDE H.F., KOREL B. y HUIZINGA D.M.: "Formal Timing Analysis for Distributed Real-Time Programs". Real-Times Systems, 7, pp. 57-90, 1994.

[XU90] XU J. y PARNAS D.L.: "Scheduling Processes with Release Times, Deadlines, Precedence, and Exclusion Relations". IEEE Transaction on Software Engineering, vol. 16, no. 3, pp. 360-369, March 1990.

[XU93] XU J. y PARNAS D.L.: "On Satisfying Timing Constraints in Hard Real-Time Systems". IEEE Transaction on Software Engineering, vol. 19, pp. 70-84, 1993

[ZHA94] ZHANG S. y BURNS A.: "A Study of Timing Properties With The Timed Token Protocol". Department of Computer Science, University of York, Technical Report YCS_226, April 1994.