Facultad
de
Ciencias

Implementación en MAST de herramientas de cálculo de tiempos de respuesta
(MAST implementation of response time calculation tools)

Trabajo de Fin de Master
para acceder al

MÁSTER EN INGENIERÍA INFORMÁTICA

Autor: Balduino López Arce

Director: J. Javier Gutiérrez García

Co-Director: Michael González Harbour

Septiembre – 2018

Índice de contenido

Índice de contenido 3
Índice de figuras 5
Índice de tablas 5
Agradecimientos 7
Resumen 8
Abstract 9
1. Introducción 10
1.1 Antecedentes y objetivos 10
Metodología 10
1.2 10
1.3 Plan de trabajo 11
2. Tecnologías y herramientas usadas 12
2.1 Tecnologías 12
2.1.1 Ada 12
2.1.2 GNAT 12
2.1.3 MAST 12
2.2 Herramientas 12
2.2.1 GPS 12
2.2.2 Sublime Text 3 12
2.2.3 Microsoft Office 12
2.2.4 Project 2016 13
2.2.5 LaTeX 13
3. Primera técnica: planificación no expulsora en sistemas monoprocesadores 14
3.1 Características de la técnica 14
3.2 Implementación 14
3.3 Ejemplo de uso 18
3.4 Resultados obtenidos 20
4. Segunda técnica: planificación híbrida expulsora y no expulsora en sistemas distribuidos 26
4.1 Características de la técnica 26
4.2 Tareas expulsables y no expulsables en monoprocesador 26
4.3 Tareas expulsables y no expulsables en sistemas distribuidos 26
4.4 Resultados obtenidos 27
5. Conclusiones y trabajos futuros 32
5.1 Conclusiones 32

5.2 Trabajos futuros 32
Anexo I: Planificación del proyecto 34
Anexo II: Guía para nuevas técnicas 35
Bibliografía 37

Índice de figuras

Figura 1: MAST 10

Figura 2: Diagrama de Gantt que recoge el plan de trabajo 11

Figura 3: Ejemplo gráfico del periodo de ocupación e instante crítico. 15

Figura 4: Ejemplo gráfico del efecto del bloqueo por una tarea de menor prioridad. 15

Figura 5: Cálculo del periodo de ocupación para la tarea C del ejemplo 19

Figura 6: Cálculo del tiempo de respuesta de la activación 0 de la tarea C del ejemplo 19

Figura 7: Cálculo del tiempo de respuesta de la activación 1 de la tarea C del ejemplo 20

Figura 8: Desglose de la planificación del ejemplo. Obtenida de la fuente [9] 20

Figura 9: Diagrama de ejecución del peor caso de la tarea A según NPRM 24

Figura 10: Diagrama de ejecución del peor caso de la tarea B según NPRM 24

Figura 11: Diagrama de ejecución del peor caso de la tarea B según CLASSIC_RM 25

Figura 12: Ejemplo para sistema distribuido híbrido. Obtenido de la fuente [10] 29

Índice de tablas

Tabla 1: Sistema de tareas a utilizar como ejemplo 18

Tabla 2: Datos del ejemplo 1 21

Tabla 3: Comparación classic_rm vs nprm para el ejemplo 1 21

Tabla 4: Datos del ejemplo 2 21

Tabla 5: Comparación classic_rm vs nprm para el ejemplo 2 21

Tabla 6: Datos del ejemplo 3 22

Tabla 7: Comparación classic_rm vs nprm para el ejemplo 3 22

Tabla 8: Datos del ejemplo 4 22

Tabla 9: Comparación classic_rm vs nprm para el ejemplo 4 22

Tabla 10: Datos del ejemplo 5 22

Tabla 11: Comparación classic_rm vs nprm para el ejemplo 5 22

Tabla 12: Datos del ejemplo 6 23

Tabla 13: Comparación classic_rm vs nprm para el ejemplo 6 23

Tabla 14: Datos del ejemplo 7 23

Tabla 15: Comparación classic_rm vs nprm para el ejemplo 7 23

Tabla 16: Ejemplo de sistema con todas tareas expulsables 27

Tabla 17: Ejemplo de sistema con todas tareas no expulsables 28

Tabla 18: Ejemplo con tareas expulsables y no expulsables 28

Tabla 19: classic_rm vs nprm para ejemplo completamente expulsable 28

Tabla 20: classic_rm vs nprm para ejemplo completamente no expulsable 28

Tabla 21: classic_rm vs nprm para ejemplo híbrido 28

Tabla 22: Análisis de ambos modelos distribuidos por el algoritmo holistic original. 30

Tabla 23: Análisis de ambos modelos distrubuidos por el algoritmo holistic mejorado 30

Agradecimientos

En primer lugar, quisiera agradecer a mi familia por apoyarme siempre y por impulsarme y haberme ayudado a estudiar esta carrera que tanto me gusta, y también por haber sido mi principal ayuda a lo largo de mi vida.

También me gustaría agradecer a mis amigos por su apoyo y sus ánimos.

También agradecer a todos los profesores que he tenido a lo largo de este máster por los conocimientos, aptitudes ganadas y buenas prácticas transmitidas a lo largo del mismo.

Por último, pero no por ello menos importante, agradecer al grupo de Ingeniería Software y Tiempo Real de la UC, por haberme ofrecido la oportunidad de realizar este TFM, en especial a J. Javier Gutiérrez García y a Michael González Harbour.

Resumen

El conjunto de herramientas MAST desarrollado en el grupo de Ingeniería Software y Tiempo Real de la UC integra un modelo de sistema de tiempo real que permite la aplicación de las técnicas más modernas de análisis de planificabilidad y optimización, tanto sobre sistemas monoprocesadores como multiprocesadores y distribuidos. Como conjunto de herramientas de código abierto y extensible, MAST permite la incorporación de nuevas técnicas de análisis. El objetivo de este trabajo es implementar en MAST algunas de las técnicas que pueden resultar interesantes para contrastar resultados de otras técnicas que se puedan desarrollar en el futuro. En concreto se pretende trabajar con sistemas monoprocesadores no expulsables planificados por prioridades fijas, y con sistemas distribuidos planificados con esa misma técnica. En el primer caso se quiere incorporar una técnica de análisis que hace un cálculo más preciso de los tiempos de respuesta, y en el segundo se ha ampliado el análisis de sistemas distribuidos para que pueda utilizar esa técnica de análisis más precisa.

Palabras clave: MAST, tiempo real, análisis de tiempos de respuesta, prioridades fijas, sistemas distribuidos, planificabilidad, planificación no expulsora

Abstract

The suite of tools called MAST, developed by Software Engineering and Real Time from the UC, integrates a real-time system model that allows the application of schedulability analysis and optimization techniques, both for monoprocessors and for multiprocessors and distributed systems. As an open source and extensible tool suite, MAST allow adding new analysis techniques.

The objective of this work is to implement on MAST some of the techniques that could be interesting to contrast results of others that could be developed in the future. Specifically, we intend to work on non-preemtible monoprocessor system using fixed priorities and with distributed systems using the same technique. First, we want to add an analysis technique that improves the response time calculation by giving more accurate results. Finally, we want to extend the analysis of distributed systems to enable the usage of that more accurate technique.

Keywords: MAST, real-time, response time analysis, fixed priorities, distributed systems, schedulability, non-preemptive scheduling

1. Introducción

A continuación, se expondrá brevemente el contexto en el cual se ha realizado el presente trabajo fin de máster, así como sus objetivos y la metodología utilizada. Asimismo, se presentará el plan de trabajo seguido y el presupuesto.

1.1 Antecedentes y objetivos

Este proyecto se desarrolla como parte de otro más grande, que consiste en el desarrollo de la herramienta de modelado y análisis de sistemas de tiempo real MAST [1]. MAST es una herramienta desarrollada por el Grupo de Ingeniería Software y Tiempo Real de la Universidad de Cantabria, que permite aplicar distintas técnicas de análisis a modelos de sistemas reales.

Figura 1: MAST

El objetivo de este proyecto es implementar una serie de nuevas técnicas que, o bien mejoren los resultados de las técnicas actualmente implementadas para ciertos casos, o bien tengan utilidad para analizar ciertos sistemas, aunque por norma general no sea muy útil. Para ello se han analizado los puntos en los que MAST podría mejorarse, y finalmente se ha decidido centrarse y mejorar el análisis de sistemas con tareas no expulsables, ya que actualmente dicho análisis se realiza de forma pesimista.

En concreto, se van a implementar 2 nuevas técnicas. La primera de ellas servirá para realizar el análisis de sistemas no expulsores basados en prioridades fijas, que en teoría mejorará los tiempos de respuesta calculados por otras técnicas ya implementadas. La segunda técnica consistirá en una ampliación de la misma para dar soporte a sistemas distribuidos en lugar de únicamente monoprocesador, y también para dar soporte a sistemas mixtos con tareas expulsables y no expulsables dentro de un mismo modelo.

Por otro lado, también se ha propuesto la creación de una guía la cual servirá a la hora de introducir nuevas técnicas, para disponer ya de una serie de pasos a seguir y que requiera de menos tiempo en un futuro.

1.2 Metodología

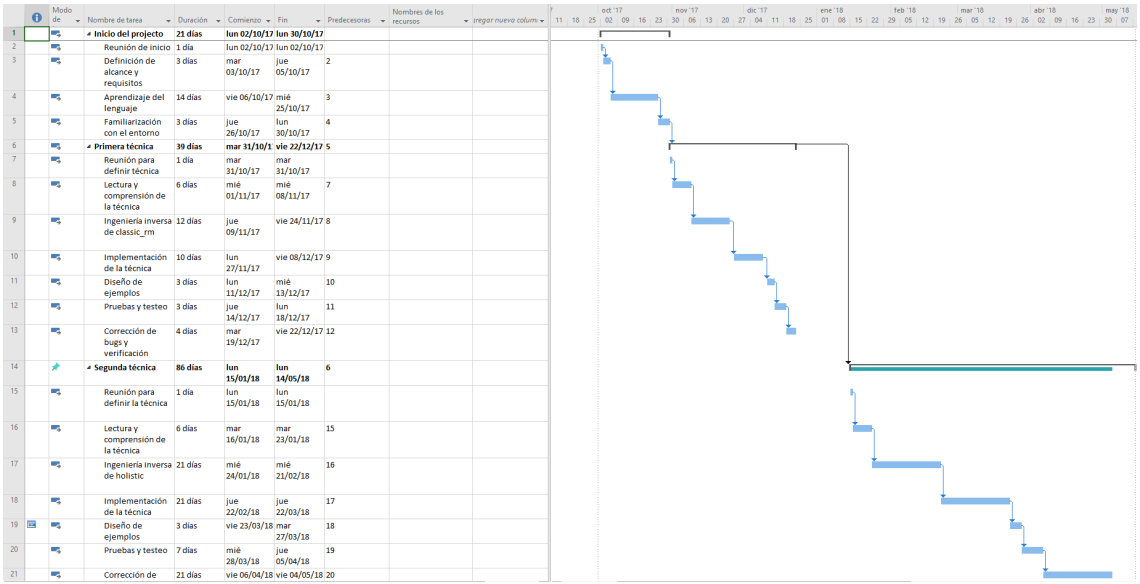
Las actividades contenidas en este proyecto han entrado dentro de un contrato de investigación de la Universidad de Cantabria, en concreto "Sistemas Ciber-Físicos de Criticidad Mixta sobre Plataformas Multinúcleo”.

Las implementaciones de las dos técnicas de análisis han consistido en un procedimiento muy similar para ambos casos. En primer lugar, se tuvo una reunión con el director del proyecto, en la cual se presentó de forma superficial la técnica a implementar, y se proporcionaba la documentación necesaria (artículo en el que se basa la técnica).

Posteriormente, se procedió a entender el funcionamiento de la técnica, y a realizar ejemplos de pequeños sistemas en papel para afianzar su funcionamiento. Una vez asimilada, se procedió a implementar la nueva técnica en MAST. Para probar su correcto funcionamiento, se comprobó en primer lugar que los tiempos de respuesta fueran adecuados, y acto seguido, para los ejemplos concretos analizados en papel anteriormente, se verificó que los resultados obtenidos fueron los mismos. Por último, se presentaron tanto los resultados como el código al director del proyecto para que diera el visto bueno a la nueva técnica implementada.

1.3 Plan de trabajo

Para cada una de las técnicas a implementar, se ha seguido más o menos la misma división de tareas, tal y como se puede ver en la figura 2. Cabe destacar que, dentro del mismo tiempo, se han realizado otras tareas, las cuales no han sido incluidas en el presente documento al no estar englobadas dentro del TFM.



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos	Agregar nueva columna
1		**Inicio del projecto**	**21 días**	**lun 02/10/17**	**lun 30/10/17**			
2		Reunión de inicio	1 día	lun 02/10/17	lun 02/10/17			
3		Definición de alcance y requisitos	3 días	mar 03/10/17	jue 05/10/17	2		
4		Aprendizaje del lenguaje	14 días	vie 06/10/17	mié 25/10/17	3		
5		Familiarización con el entorno	3 días	jue 26/10/17	lun 30/10/17	4		
6		**Primera técnica**	**39 días**	**mar 31/10/1**	**vie 22/12/17**	5		
7		Reunión para definir técnica	1 día	mar 31/10/17	mar 31/10/17			
8		Lectura y comprensión de la técnica	6 días	mié 01/11/17	mié 08/11/17	7		
9		Ingeniería inversa de classic_rm	12 días	jue 09/11/17	vie 24/11/17	8		
10		Implementación de la técnica	10 días	lun 27/11/17	vie 08/12/17	9		
11		Diseño de ejemplos	3 días	lun 11/12/17	mié 13/12/17	10		
12		Pruebas y testeo	3 días	jue 14/12/17	lun 18/12/17	11		
13		Corrección de bugs y verificación	4 días	mar 19/12/17	vie 22/12/17	12		
14		**Segunda técnica**	**86 días**	**lun 15/01/18**	**lun 14/05/18**	6		
15		Reunión para definir la técnica	1 día	lun 15/01/18	lun 15/01/18			
16		Lectura y comprensión de la técnica	6 días	mar 16/01/18	mar 23/01/18	15		
17		Ingeniería inversa de holistic	21 días	mié 24/01/18	mié 21/02/18	16		
18		Implementación de la técnica	21 días	jue 22/02/18	jue 22/03/18	17		
19		Diseño de ejemplos	3 días	vie 23/03/18	mar 27/03/18	18		
20		Pruebas y testeo	7 días	mié 28/03/18	jue 05/04/18	19		
21		Corrección de	21 días	vie 06/04/18	vie 04/05/18	20		

Figura 2: Diagrama de Gantt que recoge el plan de trabajo

Pueden apreciarse dos periodos en los cuales no se ha planificado nada. Dichos periodos corresponden el primero a las vacaciones de navidad, y el segundo simplemente son unos días de vacaciones personales, en los cuales tampoco se avanzó nada en el proyecto.

Ya que la imagen de la figura no puede apreciarse debido a su amplitud, se ha añadido una copia de esta, pero a mayor tamaño en el anexo I.

2. Tecnologías y herramientas usadas

En este capítulo se describirán las tecnologías y herramientas usadas a lo largo de todo el proyecto.

2.1 Tecnologías

2.1.1 Ada

Ada [2] es un lenguaje de programación fuertemente tipado, multipropósito, concurrente y orientado a objetos, aunque este último aspecto no será de utilidad en este proyecto. Ada se ha diseñado con la fiabilidad en mente, así como para evitar errores de programación comunes, gracias a ser un lenguaje fuertemente tipado con gestión segura de memoria y de los errores. Se utiliza principalmente en entornos donde la seguridad y la fiabilidad son factores clave.

2.1.2 GNAT

GNAT [3] es un compilador para el lenguaje de programación Ada. Dispone de una versión *community* bajo licencia GPL, la cual es la licencia de código abierto y software libre de GNU más extendida.

2.1.3 MAST

MAST [1] es una herramienta para el modelado y análisis de sistemas de tiempo real. Dispone de una sintaxis propia para la definición de los modelos, en los cuales se pueden especificar, entre otras cosas, recursos procesadores, planificadores, tareas (con tiempos de ejecución, periodos, plazos (*deadlines*)...). A dichos modelos se pueden aplicar distintas técnicas de análisis para comprobar los tiempos de respuesta u holguras (*slacks*) del sistema y de las tareas (entre otros valores), los cuales nos serán de utilidad a la hora de intentar implantar dichos sistemas en hardware real.

2.2 Herramientas

2.2.1 GPS

GPS [4] es un IDE (*Integrated Development Environment*) de código abierto perteneciente a AdaCore. Es el entorno de desarrollo para el lenguaje de programación Ada que viene por defecto con el compilador GNAT.

2.2.2 Sublime Text 3

Sublime Text 3 [5] es un editor de texto, el cual permite, entre otras cosas, la visualización de código de diferentes lenguajes de forma fácil e intuitiva.

2.2.3 Microsoft Office

Microsoft Office [6] es un conjunto de herramientas ofimáticas que facilitan la creación de informes, gráficos… Para este proyecto, se ha utilizado Microsoft Word para la creación de informes, entre los que se incluye el presente documento, y Microsoft Excel, para el almacenamiento y análisis de los tiempos de respuesta obtenidos con las diferentes técnicas implementadas y poder hacer comparaciones con las técnicas ya existentes.

2.2.4 Project 2016

Microsoft Project [7] es un software desarrollado por Microsoft, que facilita la gestión y administración de proyectos. Se pueden definir tareas, hitos, recursos y asignar tareas a los recursos entre otras funciones.

2.2.5 LaTeX

LaTex [8] es un software libre que dispone de una sintaxis para la creación de documentos en un formato específico. El código que se genera es necesario compilarlo para la creación del documento. Para este proyecto, se ha utilizado LaTeX para la creación de las fórmulas matemáticas.

3. Primera técnica: planificación no expulsora en sistemas monoprocesadores

Esta sección recoge todo el proceso de planificación e implementación de la primera técnica a añadir a MAST, correspondiente a la técnica de análisis para sistemas monoprocesadores no expulsores basados en prioridades fijas.

3.1 Características de la técnica

La primera técnica se aplica en el análisis de sistemas monoprocesadores basados en prioridades fijas y cuyas tareas sean no expulsables.
Un sistema expulsor es aquel en el cual el planificador puede expulsar a una tarea de ejecutar en la CPU en mitad de su tiempo de ejecución para dejar el procesador a otra tarea. Por el contrario, un sistema no expulsor es aquel en el cual cuando una tarea hace uso de la CPU, esta tarea mantiene la CPU hasta que termina con un tiempo de ejecución, no pudiendo ser expulsada en medio para ejecutar otras tareas.
Dicha técnica se ha decidido implementar ya que las técnicas implementadas hasta ahora en MAST eran demasiado pesimistas en algunas situaciones, y los tiempos de análisis podían llega a ser superiores a los reales ya que, por ejemplo, se tienen en cuenta el máximo de expulsiones posibles en lugar de las que realmente pueden sucederle a una tarea. Más adelante, en la sección de resultados, se expondrán algunas de las situaciones en las que la presente técnica supone una gran mejora frente a otras ya implementadas.

La técnica y su formulación se basan en un artículo centrado en el análisis de planificabilidad de sistemas con bus CAN [9]. El bus CAN, es un bus de comunicaciones muy extendido en la industria del automóvil, por ejemplo, que es apropiado para su uso en software de tiempo real debido a que garantiza tiempos de respuesta de peor caso. CAN son las siglas de *Controller Area Network*.

Al estar esta técnica pensada para el bus CAN, han sido necesarias ciertas modificaciones respecto a la formulación original del artículo, las cuales se explicarán en la siguiente sección.

3.2 Implementación

En este apartado se expondrá cómo se ha realizado la implementación de la técnica, así como las modificaciones realizadas a la formulación original del artículo [9] para adaptarla a tareas, y no a mensajes por un bus de comunicaciones.

En primer lugar, esta técnica se diferencia de las otras implementadas en MAST en que esta calcula de antemano el periodo de ocupación del instante crítico, en lugar de iterar hasta que se termine dicho periodo.

En MAST, un modelo para un sistema monoprocesador tiene únicamente una unidad de procesamiento, la cual dispone de los servidores de planificación, a los que se puede asignar prioridades. Dichos servidores de planificación representan tareas que ejecutan un determinado código. Una tarea tiene como atributos su tiempo de ejecución, que es el tiempo que consume de CPU; su periodo, que es cada cuanto tiempo se produce una nueva activación de dicha tarea; y su plazo, que es un límite de tiempo que empieza a contar tras la activación de la misma, y representa el tiempo máximo que debería tardar en responder

dicha tarea. Por último, quiero explicar que estas tareas pueden llegar a tener *jitter* en sistemas con más de un procesador. El *jitter* es un tiempo de retraso en la activación de las tareas que se puede dar por ejemplo si una tarea depende de la ejecución de otra previa posiblemente en otro procesador.

El instante crítico se consigue en sistemas de prioridades fijas activando todas las tareas en el mismo momento, lo cual provoca que todas las tareas experimenten sus peores tiempos, ya que están todas compitiendo por la CPU. En la figura 3 puede verse gráficamente el instante crítico y el periodo de ocupación. En el ejemplo de la figura la tarea A tiene mayor prioridad que la tarea B. En mitad de la ejecución de la tarea B se produce una nueva activación de A, lo que hace que B sea expulsada. Al terminar la nueva ejecución de A, B termina la ejecución de su primera activación.

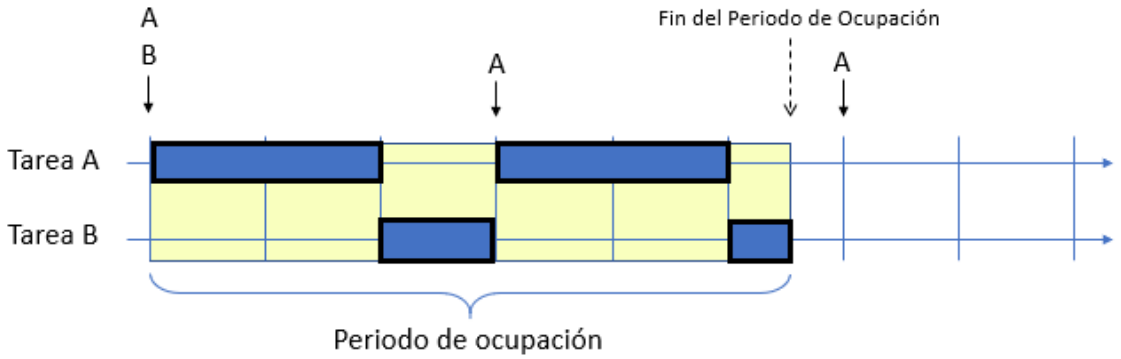


Figura 3: Ejemplo gráfico del periodo de ocupación e instante crítico.

En este caso (tareas no expulsables), para conseguir los bloqueos de tareas de mayor prioridad por las de menor prioridad, la tarea de menor prioridad con mayor tiempo de ejecución respecto a la tarea bajo análisis entrará al planificador inmediatamente antes que el resto de las tareas, (momento 0-, en comparación con el instante crítico, que sería el momento 0). Este efecto puede verse en la figura 4. Al igual que en la figura 3, la tarea A tiene mayor prioridad que la tarea B.

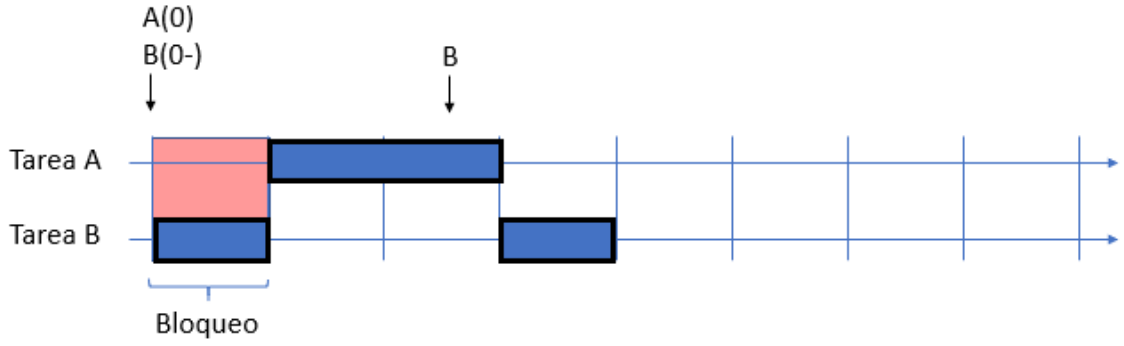


Figura 4: Ejemplo gráfico del efecto del bloqueo por una tarea de menor prioridad.

El periodo de ocupación se define como el periodo de tiempo que el procesador tiene tareas pendientes por ejecutar en la cola de ejecución desde el instante crítico. Dicho periodo de ejecución es directamente dependiente de la tarea bajo análisis, y las tareas de las que depende son las de prioridad igual o superior a ella misma. Una vez que el procesador no

tiene tareas pendientes por ejecutar, porque no se hayan producido aún nuevas activaciones de estas, es cuando se termina el periodo de ocupación.

Al ser un sistema no expulsor, las tareas pueden sufrir un **bloqueo** por otra tarea de menor prioridad que haya tomado el procesador antes que la tarea bajo análisis. Un bloqueo se produce cuando una tarea de menor prioridad toma el procesador antes que una tarea de mayor prioridad, y al no ser expulsable el sistema, la tarea de mayor prioridad no puede ejecutar hasta que la de menor prioridad termine su trabajo. Por ello, el artículo [9] propone una fórmula para calcular dicho bloqueo, la cual se muestra en la ecuación (1).

$$Bm = max_{k \in lp(m)}(Ck) \tag{1}$$

Donde *max* es la función máximo y *lp(m)* es el conjunto de tareas de menor prioridad que la tarea m. En dicha fórmula se contempla que el bloqueo de la tarea bajo análisis (**Bm**), corresponde con el máximo tiempo de ejecución (**C**) de las tareas de menor prioridad (**lp**) que la tarea bajo análisis. Hacemos uso de la función máximo ya que este efecto de bloqueo solo puede producirse por una tarea de prioridad inferior, y para el peor caso nos quedaremos con la más larga de todas las posibles. Al disponer MAST ya de una función que calcula los bloqueos, y comprobar que dichos bloqueos se calculan correctamente para sistemas no expulsables, el cálculo de bloqueo no se ha implementado, sino que se ha reutilizado la función de cálculos de bloqueo ya implementada.

Una vez calculado el bloqueo, el siguiente paso sería calcular el periodo de ocupación (**Tm**) y el número de activaciones de la tarea bajo análisis en dicho periodo de ocupación (**Qm**).

Para el cálculo del periodo de ocupación, el artículo propone las ecuaciones (2) y (3).

$$t_m^0 = C_m \tag{2}$$

$$t_m^{n+1} = B_m + \sum_{\forall k \in hep(m)} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil * C_k \tag{3}$$

Donde *hep(m)* es el conjunto de tareas de igual o mayor prioridad a m, incluyéndose a sí misma y $\lceil x \rceil$ es la función techo, la cual devuelve el menor número entero mayor que x. Dichas fórmulas para el cálculo del periodo de ocupación se han implementado tal cual se plantean en el artículo. Para n=0 se utiliza la ecuación (2), y a partir de entonces se itera con la ecuación (3) hasta que t^n sea igual a t^{n+1}. Una vez se alcanza el criterio de parada anterior, el valor de t^n será el valor del periodo de ocupación.

Para el cálculo del número de activaciones, se utiliza la ecuación (4), también sacada del mismo artículo.

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{4}$$

Tras estas ecuaciones anteriores, podemos proceder al cálculo del tiempo de respuesta de la tarea bajo análisis. Para ello, se proponen en el artículo [9] las ecuaciones (5), (6) y (7).

$$w_m^0(q) = B_m + qC_m \quad (5)$$

$$w_m^0(q) = w_m(q-1) + C_m \quad (6)$$

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \quad (7)$$

Donde *hp(m)* es el conjunto de tareas de mayor prioridad que m y τ_{bit} es el tiempo de transmisión por bit en el bus CAN.

Aunque las ecuaciones 5 y 6 parecen iguales, la ecuación 5 puede usarse en cualquier situación para calcular $w_m^0(q)$, mientras que la ecuación 6 es una manera más eficiente de calcular $w_m^0(q)$ para q>0.

En primer lugar, hay que destacar que el proceso del cálculo de w_m se tiene que iterar para q desde 0 hasta Q_m-1 (es decir, si Q_m=3, el algoritmo iteraría desde 0 hasta 2).

La ecuación (7) ha recibido una serie de modificaciones, las cuales se comentarán a continuación. En primer lugar, el τ_{bit} se ha eliminado, ya que no existen tiempos de transmisión por bus, que era el objetivo original de dicha fórmula, al estar pensado para el bus CAN. También al no estar pensada dicha fórmula para que existan tareas de la misma prioridad, el sumatorio no se aplicará a las tareas de mayor prioridad que la actual, sino que se aplicará a todas las tareas de mayor o igual prioridad que la actual, pero excluyéndose a sí misma. Por último, ha sido necesario modificar la función techo, y sustituirla por la función suelo + 1, ya que el efecto que tenía el τ_{bit} con la función techo es aumentar el valor del numerador levemente, consiguiendo así el mismo valor que la función suelo + 1. Así por ejemplo, si numerador y denominador valen 5 ambos, la función suelo+1 devolvería 2, mientras que la función techo devolvería 1. Al añadir el efecto de τ_{bit} se aumenta el valor del numerador levemente, tal que el numerador en el ejemplo anterior sería, por ejemplo, 5.01, y por lo tanto la función techo de 1.002 sería 2. Tras aplicar todas estas modificaciones, la fórmula original modificada puede verse en la ecuación (8).

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in ehep(m)} \left(\left\lfloor \frac{w_m^n + J_k}{T_k} \right\rfloor + 1 \right) C_k \quad (8)$$

Donde *ehep(m)* es el conjunto de tareas de mayor o igual prioridad que m, excluyendo a la tarea m, y $\lfloor x \rfloor$ es la función suelo, que devuelve el mayor número entero igual o menor que x.

Si nos encontramos en la iteración q=0, para n=0, utilizaremos la ecuación (5), y para valores de n posteriores utilizaremos la ecuación (8). Seguiremos iterando hasta que w^n sea igual a w^{n+1}, de forma similar a como se calculó el periodo de ocupación.

Para valores posteriores de las iteraciones q=1..Q_m-1, podemos utilizar para n=0 la ecuación (6) en lugar de la (5), ya que es un punto de partida más eficiente. Para valores de n posteriores, utilizaremos la ecuación (8), e igual que antes iteraremos hasta que w^n sea igual a w^{n+1}.

Con esto, tenemos el tiempo que la tarea bajo análisis debe esperar antes de poder entrar al procesador a ejecutar para una activación concreta dentro del periodo de ocupación. Para obtener finalmente el tiempo de respuesta de la tarea bajo análisis en esa activación, al resultado anterior debe añadirse el efecto del jitter y el propio tiempo de ejecución, así como quitar el tiempo transcurrido desde el inicio del periodo de ocupación hasta el inicio del periodo de la instancia correspondiente, como puede observarse en la ecuación (9).

$$R_m(q) = J_m + w_m(q) - qT_m + C_m \tag{9}$$

Finalmente, una vez que tenemos todos los tiempos de respuesta de las activaciones de la tarea m en el periodo de ocupación, el peor tiempo de respuesta de la tarea m es el máximo de los tiempos de respuesta de todas las activaciones, tal y como puede verse en la ecuación (10).

$$R_m = max_{q=0..Q_m-1}\left(R_m(q)\right) \tag{10}$$

Aparte de la implementación del propio algoritmo, se han puesto una serie de restricciones a los modelos que son aceptados como entrada válida a esta técnica de análisis, como pueden ser que el modelo sea monoprocesador y que sólo disponga de tareas no expulsables.

También se ha escrito una pequeña guía documentando el proceso de introducción de una nueva técnica en MAST. Esta guía será de utilidad a terceras personas a la hora de introducir nuevas técnicas en un futuro. Esta guía es uno de los resultados del proyecto y puede verse en el anexo II.

3.3 Ejemplo de uso

En este apartado se ejemplificará la técnica con un caso de uso sencillo que está directamente sacado del artículo mencionado anteriormente [9]. Se ha adaptado el ejemplo, de tal forma que se ajuste a las modificaciones explicadas en el apartado anterior. Aparte, se han cambiado los plazos (*deadlines*) para que el sistema sea planificable. Todos los términos de *jitter* son cero.

Se dispone de 3 tareas de distinta prioridad, cuyos tiempos de ejecución, periodos y plazos se muestran en la Tabla 1. No se incluyen unidades, pero tanto periodos como plazos y tiempos de ejecución (WCET) tienen la misma magnitud de tiempo (por ejemplo, milisegundos).

Tarea	Prioridad	Periodo	Plazo	WCET
A	Alta	2.5	2.5	1
B	Media	3.5	3.5	1
C	Baja	3.5	4.0	1

Tabla 1: Sistema de tareas a utilizar como ejemplo

Como ejemplo, se analizará **el tiempo de respuesta de la tarea C**. Para el resto de las tareas se realizaría el análisis de forma similar, pero por no repetir innecesariamente el proceso, sólo se realizará para dicha tarea.

Siguiendo los pasos del algoritmo explicados en el paso anterior, el primer paso sería calcular el tiempo de bloqueo de la tarea C, aplicando la ecuación (1), la cual nos da como resultado 0, ya que no hay tareas de menor prioridad que C.

A continuación, el siguiente paso sería calcular el periodo de ocupación, utilizando las ecuaciones (2) y (3). Como punto de partida, para n=0 tenemos t_C^0=1 (utilizando la ecuación (2)). Desde este punto, utilizando la ecuación (3) tenemos los resultados de 3, 4, 6, 7 y 7. Al ser t_C^4=t_C^5, podemos parar y concluir que el periodo de ocupación del sistema es 7. Un mayor desglose de cómo se han realizado las operaciones puede verse en la figura 5

$$\mathrm{t}_c^0 = \mathrm{C_c} = 1$$

$$\mathrm{t}_c^1 = \mathrm{B_c} + \sum_{k \in hpe(c)} \lceil \frac{t_c^0 + J_\mathrm{k}}{T_\mathrm{k}} \rceil C_\mathrm{k} = 1 + 1 + 1 = 3$$

$$\mathrm{t}_c^2 = \mathrm{B_c} + \sum_{k \in hpe(c)} \lceil \frac{t_c^1 + J_\mathrm{k}}{T_\mathrm{k}} \rceil C_\mathrm{k} = 2 + 1 + 1 = 4$$

$$\mathrm{t}_c^3 = \mathrm{B_c} + \sum_{k \in hpe(c)} \lceil \frac{t_c^2 + J_\mathrm{k}}{T_\mathrm{k}} \rceil C_\mathrm{k} = 2 + 2 + 2 = 6$$

$$\mathrm{t}_c^4 = \mathrm{B_c} + \sum_{k \in hpe(c)} \lceil \frac{t_c^3 + J_\mathrm{k}}{T_\mathrm{k}} \rceil C_\mathrm{k} = 3 + 2 + 2 = 7$$

$$\mathrm{t}_c^5 = \mathrm{B_c} + \sum_{k \in hpe(c)} \lceil \frac{t_c^4 + J_\mathrm{k}}{T_\mathrm{k}} \rceil C_\mathrm{k} = 3 + 2 + 2 = 7$$

Figura 5: Cálculo del periodo de ocupación para la tarea C del ejemplo

Una vez tenemos el periodo de ocupación, utilizando la ecuación (4) obtenemos el número de activaciones de la tarea C en el periodo de ocupación, la cual nos da un resultado de 2.

Por último, sólo resta calcular los tiempos de respuesta de las activaciones 0 y 1 de la tarea C. Para ello utilizaremos las ecuaciones (5), (6) y (8).

Para la activación 0, con la ecuación (5) partimos de un valor de 0. A partir de aquí, iteramos con la ecuación (8), hasta que alcanzamos el factor de parada, lo cual nos da unos resultados de 2 y 2. Al ser el valor de $w_C^1(0) = w_C^2(0)$, podemos concluir que el tiempo de espera para que la tarea C pueda ejecutar en su primera activación es 2, y utilizando la ecuación (9) nos daría un tiempo de respuesta de 3. El desglose de estas operaciones puede verse en la figura 6.

$$\mathrm{w}_c^0(0) = \mathrm{B_c} + \mathrm{qC_c} = 0 + 0{*}1 = 0$$

$$\mathrm{w}_c^1(0) = \mathrm{B_c} + \mathrm{qC_c} + \sum_{k \in hpe(c) \cup c!} \lfloor \frac{w_c^0 + J_\mathrm{k}}{T_\mathrm{k}} \rfloor C_\mathrm{k} + 1 = 0 + 0{*}1 + 1 + 1 = 2$$

$$\mathrm{w}_c^2(0) = \mathrm{B_c} + \mathrm{qC_c} + \sum_{k \in hpe(c) \cup c!} \lfloor \frac{w_c^1 + J_\mathrm{k}}{T_\mathrm{k}} \rfloor C_\mathrm{k} + 1 = 0 + 0{*}1 + 1 + 1 = 2$$

$$\mathrm{R_c}(0) = \mathrm{J_c} + \mathrm{W_c}(0) - \mathrm{qT_c} + \mathrm{C_c} = 0 + 2 - 0 + 1 = 3$$

Figura 6: Cálculo del tiempo de respuesta de la activación 0 de la tarea C del ejemplo

Para la activación 1, a diferencia del caso anterior, podemos partir utilizando la ecuación (6), lo cual nos daría un valor de partida de 3. A partir de aquí, como con el caso anterior, iteramos utilizando la ecuación (8) hasta alcanzar el factor de parada, lo cual nos da unos valores de 4, 5, 6 y 6. Al ser $w_C^3(1) = w_C^4(1)$, podemos concluir que el tiempo de espera para que la tarea C pueda ejecutar su segunda activación es de 6, y utilizando nuevamente la ecuación (9) nos da un tiempo de respuesta de 3.5. Nuevamente, el desglose de estas operaciones puede verse en la figura 7.

$$\mathrm{w}_c^0(1) = \mathrm{w_c}(0) + \mathrm{C_c} = 2 + 1 = 3$$

$$\mathrm{w}_c^1(1) = \mathrm{B_c} + \mathrm{qC_c} + \sum_{k \in hpe(c) \cup c!} \lfloor \frac{w_c^0 + J_\mathrm{k}}{T_\mathrm{k}} \rfloor C_\mathrm{k} + 1 = 0 + 1 + 2 + 1 = 4$$

$$\mathrm{w}_c^2(1) = \mathrm{B_c} + \mathrm{qC_c} + \sum_{k \in hpe(c) \cup c!} \lfloor \frac{w_c^1 + J_\mathrm{k}}{T_\mathrm{k}} \rfloor C_\mathrm{k} + 1 = 0 + 1 + 2 + 2 = 5$$

$$\mathrm{w}_c^3(1) = \mathrm{B_c} + \mathrm{qC_c} + \sum_{k \in hpe(c) \cup c!} \lfloor \frac{w_c^1 + J_\mathrm{k}}{T_\mathrm{k}} \rfloor C_\mathrm{k} + 1 = 0 + 1 + 3 + 2 = 6$$

$$\mathrm{w}_c^4(1) = \mathrm{B_c} + \mathrm{qC_c} + \sum_{k \in hpe(c) \cup c!} \lfloor \frac{w_c^1 + J_\mathrm{k}}{T_\mathrm{k}} \rfloor C_\mathrm{k} + 1 = 0 + 1 + 3 + 2 = 6$$

$$\mathrm{R_c}(1) = \mathrm{J_c} + \mathrm{W_c}(1) - \mathrm{qT_c} + \mathrm{C_c} = 0 + 6 - 3.5 + 1 = 3.5$$

Figura 7: Cálculo del tiempo de respuesta de la activación 1 de la tarea C del ejemplo

Finalmente, sólo resta utilizar la ecuación (10) para obtener el tiempo de respuesta de peor caso de la tarea C, que es el máximo tiempo de respuesta de todas sus iteraciones, en este caso, **el peor tiempo de respuesta de la tarea C es de 3.5**.

Si comprobamos este resultado con el escenario para la tarea C, propuesto también en el artículo [9], podemos comprobar que efectivamente para la primera activación responde en 3, y para la segunda en 3.5, siendo su peor tiempo de respuesta este último, tal y como puede verse en la figura 8.

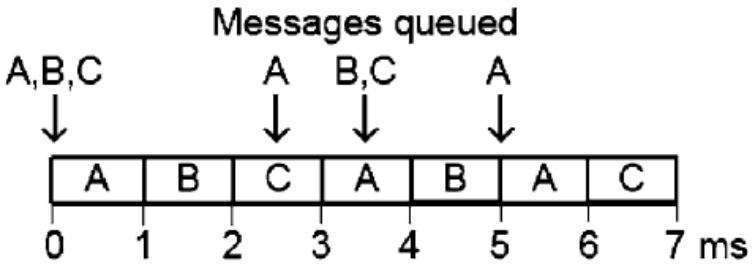


Figura 8: Desglose de la planificación del ejemplo. Obtenida de la fuente [9]

3.4 Resultados obtenidos

Una vez implementada la técnica y comprobado su correcto funcionamiento, se ha diseñado una serie de sistemas de ejemplo con la sintaxis propia de los modelos de MAST. Se han diseñado un total de 7 modelos diferentes, para intentar cubrir bastantes de los posibles escenarios en los que esta técnica podría mejorar respecto a las anteriores implementadas.

El primer ejemplo es el propuesto en el artículo [9], y resuelto en el apartado anterior para la tarea C de este. El segundo ejemplo consiste en un sistema similar al anterior, pero esta vez con 4 tareas de distinta prioridad en lugar de 3. El tercer ejemplo es otro sistema de alta utilización del sistema, similar al primero. El cuarto y quinto ejemplos corresponden a sistemas con media y baja utilización respectivamente. El sexto ejemplo consiste en un modelo de 4 tareas similar al ejemplo 2, pero en esta ocasión existen 2 tareas de la misma prioridad (en este caso máxima prioridad) para ver si el algoritmo realmente es efectivo habiendo varias tareas de la misma prioridad. Por último, el séptimo ejemplo consiste en un sistema con 2 tareas únicamente, una de ellas con poca utilización y un gran periodo, y otra también con poca utilización, pero un periodo mucho menor. Ambas tareas son de idéntica prioridad. Este ejemplo se utilizará para contrastar la aproximación de la nueva técnica respecto a las que ya existían referente a las colas FIFO para las tareas de misma prioridad. Todos los datos de los ejemplos, así como los resultados obtenidos se muestran en las tablas desde la 2 hasta la 15, y cuyos resultados se comentarán a continuación de las mismas. Se comparan la técnica existente actualmente en MAST para sistemas monoprocesadores conocida como *classic_rm* y la técnica propuesta aquí denominada *nprm*. La técnica clásica es capaz de analizar sistemas no expulsables, pero lo hace de manera pesimista:

Ejemplo 1	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	3	1	2,5	2,5
B	2	1	3,5	3,25
C	1	1	3,5	3,5

Tabla 2: Datos del ejemplo 1

Utilización: 97,14%	**A**	**B**	**C**
classic_rm	2	4	5
nprm	2	3	3,5
Mejora	0%	33,33%	42,86%

Tabla 3: Comparación classic_rm vs nprm para el ejemplo 1

Ejemplo 2	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	4	1	5	8,5
B	3	1	6	7,5
C	2	1	5,5	7,25
D	1	1,5	3,5	4,75

Tabla 4: Datos del ejemplo 2

Utilización: 97,71%	**A**	**B**	**C**	**D**
classic_rm	2,5	3,5	4,5	6,5
nprm	2,5	3,5	4,5	4,5
Mejora	0%	0%	0%	44,44%

Tabla 5: Comparación classic_rm vs nprm para el ejemplo 2

Ejemplo 3	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	3	1,5	4	4,5
B	2	2	7	7
C	1	1	3,5	4,25

Tabla 6: Datos del ejemplo 3

Utilización: 94,64%	**A**	**B**	**C**
classic_rm	3,5	6	6
nprm	3,5	4,5	4,5
Mejora	0%	33.33%	33.33%

Tabla 7: Comparación classic_rm vs nprm para el ejemplo 3

Ejemplo 4	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	3	1	6	6,5
B	2	1	5	5
C	1	1	7,5	3,25

Tabla 8: Datos del ejemplo 4

Utilización: 73,33%	**A**	**B**	**C**
classic_rm	3,5	4,5	4,5
nprm	3,5	4,5	4,5
Mejora	0%	0%	0%

Tabla 9: Comparación classic_rm vs nprm para el ejemplo 4

Ejemplo 5	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	3	3	12	9,5
B	2	1	15	8
C	1	3,5	17,5	5,5

Tabla 10: Datos del ejemplo 5

Utilización: 51,67%	**A**	**B**	**C**
classic_rm	6,5	7,5	7,5
nprm	6,5	7,5	7,5
Mejora	0%	0%	0%

Tabla 11: Comparación classic_rm vs nprm para el ejemplo 5

Ejemplo 6	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	3	1	5	5
B	3	2	4	4,5
C	2	1	8	9
D	1	1,5	9	9

Tabla 12: Datos del ejemplo 6

Utilización: 99,17%	**A**	**B**	**C**	**D**
classic_rm	6,5	4,5	11,5	14,5
nprm	4,5	4,5	8,5	8,5
Mejora	44,44%	0%	35,29%	70,59%

Tabla 13: Comparación classic_rm vs nprm para el ejemplo 6

Ejemplo 7	**Prioridad**	**WCET**	**Periodo**	**Plazo**
A	1	4	10	26
B	1	20	100	28

Tabla 14: Datos del ejemplo 7

Utilización: 60%	**A**	**B**
classic_rm	24	36
nprm	24	24
Mejora	0%	50%

Tabla 15: Comparación classic_rm vs nprm para el ejemplo 7

Para los ejemplos del 1 al 3, puede verse una mejoría, sobre todo en las tareas de menor prioridad. Esto se debe a que el algoritmo *classic_rm* considera que las tareas de menos prioridad pueden ser expulsadas más de una vez por una tarea de mayor prioridad, pero al no ser expulsables esto no puede producirse. Este efecto se explicará más detalladamente para el ejemplo 7.

En los ejemplos 4 y 5 puede verse que, con una utilización media o baja, este algoritmo da los mismos resultados que el *classic_rm*. Esto se puede deber a que el periodo de ocupación incluya únicamente 1 activación de cada tarea, y por lo tanto ambos análisis llegan a los mismos resultados.

Para el ejemplo 6, puede verse una clara mejoría en varios aspectos. En primer lugar, las tareas de una misma prioridad tienen el mismo tiempo de respuesta, ya que podrían interrumpirse entre sí en sus peores casos, por lo que por un lado conseguimos afinar más aún los tiempos de respuesta de este tipo de sistemas con tareas de misma prioridad. Por otro lado, el efecto de múltiples expulsiones (las cuales no pueden producirse al estar en un sistema no expulsor) a tareas de menor prioridad se ve reflejado, consiguiendo mejoras de hasta un 70%.

Finalmente, con el ejemplo 7 se pretende demostrar este efecto de las múltiples expulsiones con un ejemplo sencillo. Lo correcto, y lo que calcula la nueva técnica puede verse en las figuras 9 y 10. Para calcular el peor tiempo de A, consideramos que primero se activaría la

tarea B, y ejecutaría sus 20 unidades de tiempo, y posteriormente ejecutaría A (ya que B no se vuelve a activar hasta el instante 100). Con esto se consigue un tiempo de respuesta de la primera activación de 24 unidades de tiempo. La segunda activación se produce en el instante 10, pero en el instante 24 al estar libre ejecuta entre el instante 25 y 28, lo que da un tiempo de respuesta de 18, inferior a los 24 de la primera activación. La tercera activación producida en el instante 20 se ejecutaría entre el 29 y 32, dando un tiempo de respuesta de 12, también inferior a los 24. La cuarta y última activación del periodo de ocupación se produce en el instante 30, y ejecuta entre el 33 y 36, dando un tiempo de respuesta 6, también inferior a los 24. Al terminar el periodo de ocupación, podemos concluir que el mayor tiempo de respuesta de A es 24 unidades de tiempo.

Para la tarea B es más sencillo, ya que únicamente hay 1 activación en el periodo de ocupación. Se activaría en el instante 0, pero antes de su ejecución se ejecutaría A por 4 unidades de tiempo. Al ser no expulsor, aunque A se vuelve a activar en el instante 10, no puede expulsar a B de la CPU, y terminaría su tiempo de ejecución en el instante 24.

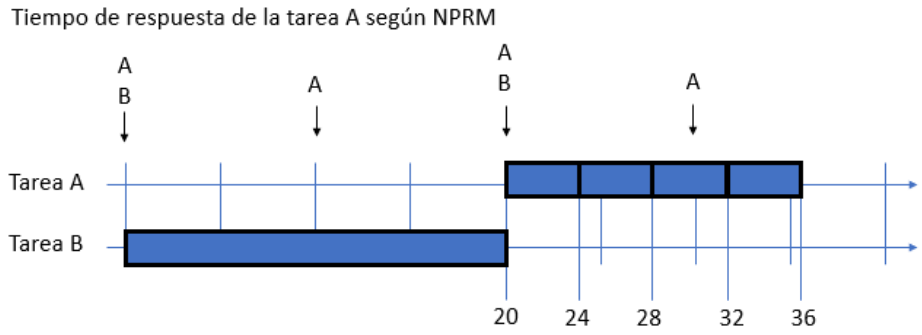


Figura 9: Diagrama de ejecución del peor caso de la tarea A según NPRM

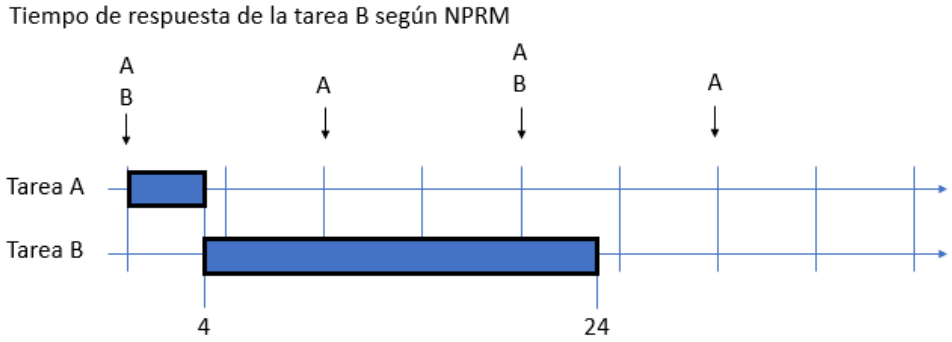


Figura 10: Diagrama de ejecución del peor caso de la tarea B según NPRM

El problema que tiene *classic_rm* se explicará a continuación, y también puede verse gráficamente en la figura 11. Dicho algoritmo calcula el tiempo de respuesta de B como se explica a continuación. Primero, calcula el número activaciones de A en el periodo de ocupación de B, que son 4. El algoritmo *classic_rm* lo que hace es considerar que las 4 activaciones de A ejecutarían antes de B, lo que haría que B tuviera un bloqueo antes de su ejecución de 16 unidades de tiempo, después de los cuales ejecutaría sus 20 unidades, dando un tiempo de respuesta de 36. Esto no es real, ya que en el instante 4 termina de ejecutar A, y quedaría la CPU libre, la cual sería tomada por la tarea B, y al no poder ser expulsada, ejecutaría hasta terminar en el instante 24.

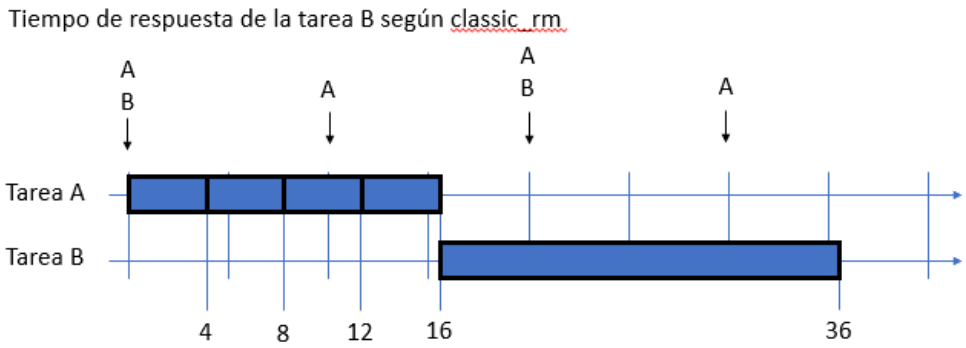


Figura 11: Diagrama de ejecución del peor caso de la tarea B según CLASSIC_RM

En consecuencia, puede verse tras los ejemplos expuestos anteriormente, así como en la descripción del funcionamiento del algoritmo, que esta técnica de análisis proporciona a MAST una mejora en los resultados del análisis de sistemas monoprocesadores con tareas no expulsoras, lo cual permite validar sistemas que con la técnica previa serían dados por no planificables.

4. Segunda técnica: planificación híbrida expulsora y no expulsora en sistemas distribuidos

En este apartado se explicará, similarmente al apartado anterior, el proceso por el cual se ha analizado e implementado la segunda técnica a añadir a MAST.

4.1 Características de la técnica

En primer lugar, se ha decidido modificar el algoritmo para aceptar en la planificación tanto tareas expulsables como no expulsables dentro de un mismo procesador. Para ello, hay que diferenciar entre tareas expulsables y no expulsables y analizarlas de forma diferente según ese criterio. Toda esta técnica continúa bajo un enfoque monoprocesador.

Seguidamente, tras verificar el correcto funcionamiento del apartado anterior, se procederá a implementar dicha técnica de análisis, pero esta vez para sistemas distribuidos. Para ello, se hará uso del ejemplo propuesto por *Di Natale* [10], el cual es ideal para ver si esta técnica produce los efectos buscados, es decir, reducir los tiempos de respuesta de los análisis considerablemente.

4.2 Tareas expulsables y no expulsables en monoprocesador

Para la implementación de esta técnica se ha tomado como base, al igual que en el caso original, el algoritmo *classic_rm*. Se han fusionado *classic_rm* con el algoritmo desarrollado en el punto anterior, y dependiendo de si la tarea es expulsable o no, se utiliza *classic_rm* (para las tareas expulsables) o el algoritmo *nprm* desarrollado, para calcular el tiempo de respuesta de las tareas.
Originalmente, no había forma de diferenciar en MAST si una tarea era o no expulsable, por lo que ha sido necesario modificar la traducción del sistema, añadiendo una nueva variable booleana, la cual se ha denominado *preemptible*. Dicha variable obtiene el valor de *false* si la tarea pertenece a un planificador no expulsable y de *true* en caso contrario. Con esta modificación ya es posible realizar dicha diferenciación y por lo tanto realizar el análisis.

Se ha probado que dicho algoritmo de planificación híbrida funcione correctamente para un sistema con todas las tareas expulsables, y se ha verificado que su resultado era idéntico al de *classic_rm* original, y con un sistema con todas las tareas no expulsables, y se ha verificado nuevamente que su resultado era idéntico al algoritmo desarrollado anteriormente. Por último, se ha probado a mezclar tareas expulsables con no expulsables, y los resultados obtenidos son los esperados. Dichos resultados se detallarán en un apartado posterior, dedicado a los resultados obtenidos.

4.3 Tareas expulsables y no expulsables en sistemas distribuidos

Una vez implementado y probado el sistema híbrido explicado en el apartado anterior, se procederá a la implementación de otro sistema con funcionamiento similar, pero esta vez en lugar de limitarnos a un sistema monoprocesador, se realizará un algoritmo de análisis para sistemas distribuidos.

En contraposición a la implementación anterior basada en *classic_rm*, para el sistema distribuido se utilizará como base el algoritmo *holistic* de MAST [1]. Este algoritmo tiene como base un equivalente al *classic_rm* para analizar los tiempos de respuesta en cada

procesador e itera sobre este algoritmo para ir obteniendo una respuesta final válida para un sistema distribuido con muchos procesadores. El objetivo será, utilizando la nueva variable booleana para diferenciar una tarea expulsable de una no expulsable, cambiar el análisis de tiempo de respuesta de cada procesador individual usando para las tareas no expulsables el algoritmo desarrollado inicialmente en el apartado 3, dejando el análisis de las tareas expulsables al algoritmo equivalente a *classic_rm*.

Con esto, se pretende dotar a MAST [1] de una nueva funcionalidad de la que carecía. Como se explicará en el apartado siguiente, actualmente en MAST para realizar el análisis de sistemas con tareas no expulsables se tiene que crear un artificio del modelo, ya que el algoritmo actual toma las tareas no expulsables como si lo fueran, dando lugar a resultados pesimistas.

En el apartado siguiente se explicará, con el uso del ejemplo propuesto por Di Natale [10] la diferencia en los tiempos de respuesta obtenidos con el algoritmo original frente al algoritmo "mejorado" que se pretende implementar.

4.4 Resultados obtenidos

En el presente apartado se presentarán los resultados obtenidos tras aplicar las 2 técnicas anteriores, ampliaciones de la originalmente propuesta y que fue explicada en el apartado 3 del presente informe.

En primer lugar, se ha probado el correcto funcionamiento de la técnica híbrida para sistemas monoprocesadores. Para ello, se ha probado el algoritmo con 3 sistemas diferentes. El primero de ellos consta de un sistema con únicamente tareas expulsables, y será comparado con el algoritmo *classic_rm*. En este caso los resultados del nuevo algoritmo y el de *classic_rm* deberían ser idénticos ya que todas las tareas del sistema deberían analizarse utilizando el algoritmo de *classic_rm* dentro del algoritmo híbrido. En segundo lugar, se probará con un sistema con todas tareas no expulsables, y su resultado debería ser idéntico al del algoritmo desarrollado en el apartado 3 de este documento, ya que para todas las tareas se utilizará el algoritmo nuevo. Por último, se probará con un sistema híbrido en el cual conviven tareas expulsables y no expulsables. En este caso, el resultado de las tareas expulsables debería coincidir con los tiempos de respuesta de *classic_rm*, mientras que las tareas no expulsables deberían tener tiempos de respuesta iguales a los obtenidos para dicha tarea utilizando el algoritmo del apartado 3. En las tablas 16, 17 y 18 pueden verse las características de los 3 sistemas anteriores respectivamente, mientras que en las tablas 19, 20 y 21 se muestran los tiempos de respuesta obtenidos para estos sistemas, que permiten comprobar lo explicado en el presente párrafo.

Tarea	Prioridad	WCET	Periodo	¿Expulsable?
A	Alta	1	2.5	Si
B	Media	1	3.5	Si
C	Baja	1	3.5	Si

Tabla 16: Ejemplo de sistema con todas tareas expulsables

Tarea	Prioridad	WCET	Periodo	¿Expulsable?
A	Alta	1	2.5	No
B	Media	1	3.5	No
C	Baja	1	3.5	No

Tabla 17: Ejemplo de sistema con todas tareas no expulsables

Tarea	Prioridad	WCET	Periodo	¿Expulsable?
A	Alta	1	2.5	No
B	Media	1	3.5	Si
C	Baja	1	3.5	No

Tabla 18: Ejemplo con tareas expulsables y no expulsables

Tarea	A	B	C
Classic_rm	1	2	5
nprm	1	2	5

Tabla 19: classic_rm vs nprm para ejemplo completamente expulsable

Tarea	A	B	C
Classic_rm	2	4	5
nprm	2	3	3.5

Tabla 20: classic_rm vs nprm para ejemplo completamente no expulsable

Tarea	A	B	C
Classic_rm	2	4	5
nprm	2	4	3.5

Tabla 21: classic_rm vs nprm para ejemplo híbrido

Como puede verse, para un sistema completamente expulsable no se produce ningún cambio en su análisis. Para un sistema completamente no expulsable, se obtienen los mismos resultados que los obtenidos en la tabla 3 de la técnica sólo no expulsable. Así pues, podemos ver que ambos extremos (sistema completamente no expulsable, y sistema completamente expulsable) están correctos. Por último, para el sistema híbrido, se tienen los resultados del análisis no expulsable para las tareas A y C, mientras que la tarea B que sigue siendo expulsable se tiene el tiempo que proporcionaba el análisis *classic_rm* para la misma en un sistema no expulsable, por lo que podemos concluir que, con esta técnica de análisis para monoprocesador, podemos ejecutar todo tipo de sistemas que tengan combinaciones de tareas expulsables y no expulsables. Por lo tanto, esta técnica sustituirá a la anterior propuesta en este trabajo, que únicamente admitía tareas no expulsables, mientras que esta puede hacer ese trabajo y más.

Por otro lado, para el sistema distribuido, se ha tomado como base para el ejemplo el sistema propuesto por Di Natale [10], cuya especificación puede verse en la figura 12. Dicho sistema actualmente no puede modelarse e introducirse para analizarlo en MAST [1] así sin más, sino que es necesario llevar a cabo una transformación para que pueda ser analizado.

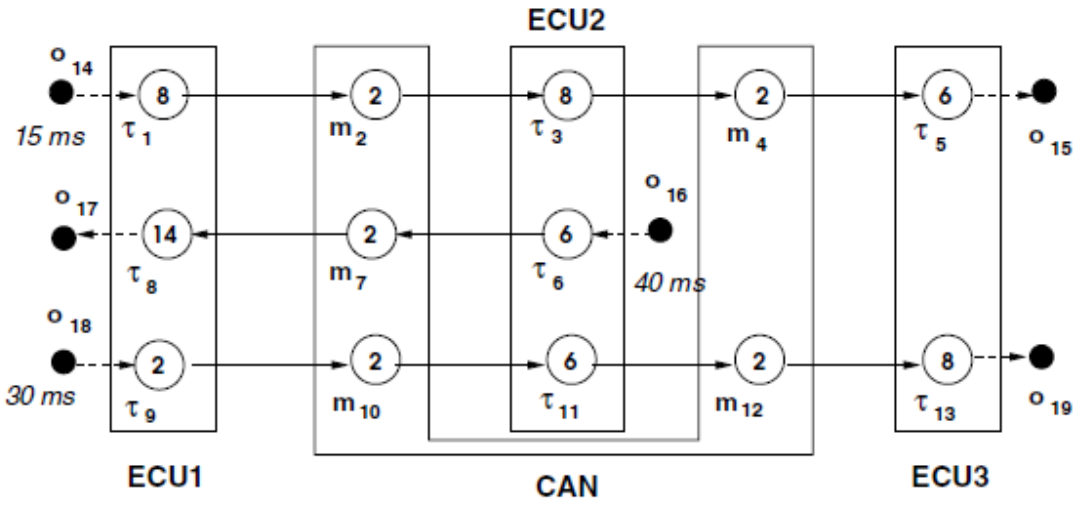


Figura 12: Ejemplo para sistema distribuido híbrido. Obtenido de la fuente [10]

Como puede verse en la figura anterior, en este ejemplo pueden observarse 3 procesadores, interconectados todos entre sí por un bus CAN. Se dispone de 3 transacciones, siendo todas las tareas en los 3 procesadores expulsables, pero las tareas que representan los mensajes del bus CAN no lo son.

Actualmente en MAST, la técnica que analiza sistemas distribuidos no hace caso de que un servidor de planificación sea no expulsable y las tareas asociadas al mismo las trata como expulsables. Sin embargo, para emular el efecto de que una tarea no sea expulsable, se puede hacer uso de un recurso compartido por todas las tareas no expulsables, las cuales al iniciar su ejecución lo toman, y al terminar lo liberan. Más información sobre la emulación de este efecto se puede encontrar en [11]. Con esto, se consigue emular la no expulsión de las mismas, pero el modelo no es fiel a la realidad ya que dicho recurso compartido es ficticio.

Con la nueva técnica de análisis para sistemas distribuidos, se pretende evitar dicha transformación innecesaria en el modelo, para que únicamente etiquetando las tareas como no expulsables, el propio algoritmo de análisis lo tome en cuenta y proceda a realizar su análisis de forma diferente a como si fuera una tarea expulsable.

Para demostrar los resultados, en la tabla 22 se muestran los tiempos de respuesta obtenidos por el análisis *holistic* original de MAST aplicado sobre el sistema original de Di Natale (incorrectos) y el sistema modificado con el recurso compartido. Con objeto de establecer comparaciones también se han añadido a las tablas 22 y 23 una columna adicional, que corresponde al modelo de Di Natale original, pero siendo el bus CAN expulsable también. Acto seguido, en la tabla 23 se analizarán los mismos modelos, pero esta vez utilizando el nuevo algoritmo de análisis.

Tarea	Modelo original	Modelo modificado	Modelo expulsable
τ_1	8	8	8
m2	10	12	10
τ_3	18	20	18
m4	22	26	22
τ_5	28	32	28
τ_6	30	30	30
m7	40	44	40
τ_8	70	74	70
τ_9	162	190	162
m10	178	208	178
τ_{11}	228	266	228
m12	262	302	262
τ_{13}	294	334	294

Tabla 22: Análisis de los modelos distribuidos aplicando el algoritmo holistic original.

Tarea	Modelo original	Modelo modificado	Modelo expulsable
τ_1	8	8	8
m2	12	12	10
τ_3	20	20	18
m4	26	26	22
τ_5	32	32	28
τ_6	30	30	30
m7	44	44	40
τ_8	74	74	70
τ_9	190	190	162
m10	208	208	178
τ_{11}	266	266	228
m12	302	302	262
τ_{13}	334	334	294

Tabla 23: Análisis de los modelos distribuidos aplicando el algoritmo holistic mejorado

Como puede verse, el modelo original en la versión antigua es idéntica a la no expulsable, por lo que está tomando dichas tareas como no expulsables. Con los nuevos cambios, ambos modelos dan el mismo resultado, por lo que ya no es necesario modificar el modelo original y añadir dicho

recurso compartido. También puede verse que el modelo completamente expulsable da resultados idénticos para las dos técnicas de análisis aplicadas.

En conclusión, y tras analizar los resultados anteriores, puede verse que esta nueva técnica al igual que la anterior provee a MAST de nueva funcionalidad, como puede ser el mezclar en un mismo sistema tareas expulsables y no expulsables sin necesidad de adaptar los modelos con la modificación del recurso compartido. Por otro lado, permite aprovechar para la parte no expulsable las mejoras introducidas en la técnica anterior.

5. Conclusiones y trabajos futuros

En la presente sección del documento, se presentarán las conclusiones obtenidas tras la realización del proyecto, así como las tareas pendientes o trabajos que se desearía realizar en un futuro.

5.1 Conclusiones

El presente proyecto tiene como principal finalidad ampliar la funcionalidad de MAST [1] y añadir a dicha herramienta nuevos algoritmos y técnicas de análisis que permitan analizar nuevos sistemas, o bien optimizar los tiempos de respuesta de peor caso en técnicas ya existentes. Para ello, se ha añadido una técnica completamente nueva, y se ha modificado una ya existente para ampliar su funcionalidad. La técnica nueva ha proporcionado unos muy buenos resultados en el análisis de sistemas con tareas no expulsables en sistemas monoprocesadores, mientras que la modificada permite el análisis de sistemas distribuidos con varias tareas, entre las cuales puede haber tareas expulsables y no expulsables, sin necesidad de modificar el modelo añadiendo un recurso compartido ficticio.

El proyecto ha cumplido con las expectativas. Aunque inicialmente se pretendía trabajar también con técnicas de análisis para sistemas bajo EDF, se ha decidido posponer esta parte para profundizar más en lo que se ha desarrollado, debido al buen resultado que han dado estas técnicas para sistemas con tareas expulsables y no expulsables. Esto queda propuesto como trabajo futuro como se comentará posteriormente.

Se han encontrado varias dificultades a lo largo del proyecto, entre la que cabe destacar el realizar ingeniería inversa a la técnica de análisis monoprocesador de MAST "*classic_rm*", y posteriormente a la técnica de análisis distribuido "holistic". Su implementación general es bastante compleja, ya que abarca varios ficheros.

En cuanto a la visión personal, la realización de este proyecto me ha servido para enfrentarme por primera vez a un reto de ingeniería inversa, ya que, para implementar nuevas técnicas, he tenido que basarme en técnicas ya existentes, y modificar sus cálculos para que se adapten a las fórmulas correspondientes. Con esto, también he aprendido la importancia de ir apuntando cada uno de los pasos a seguir para introducir nuevas técnicas, para que la persona que venga detrás de mi tenga más fácil dicho trabajo, y en lugar de tener que hacer ingeniería inversa pueda simplemente mirarse una guía.

Como conclusión de este proyecto, he podido profundizar más en el campo de análisis de tareas de tiempo real, ya que hasta ahora no había profundizado nada dentro de ese campo, fuera de las asignaturas del grado y el máster. Ha sido también una experiencia muy positiva ya que he podido ver y modificar el código fuente de una herramienta de código abierto, y a parte contribuir personalmente en su desarrollo y evolución.

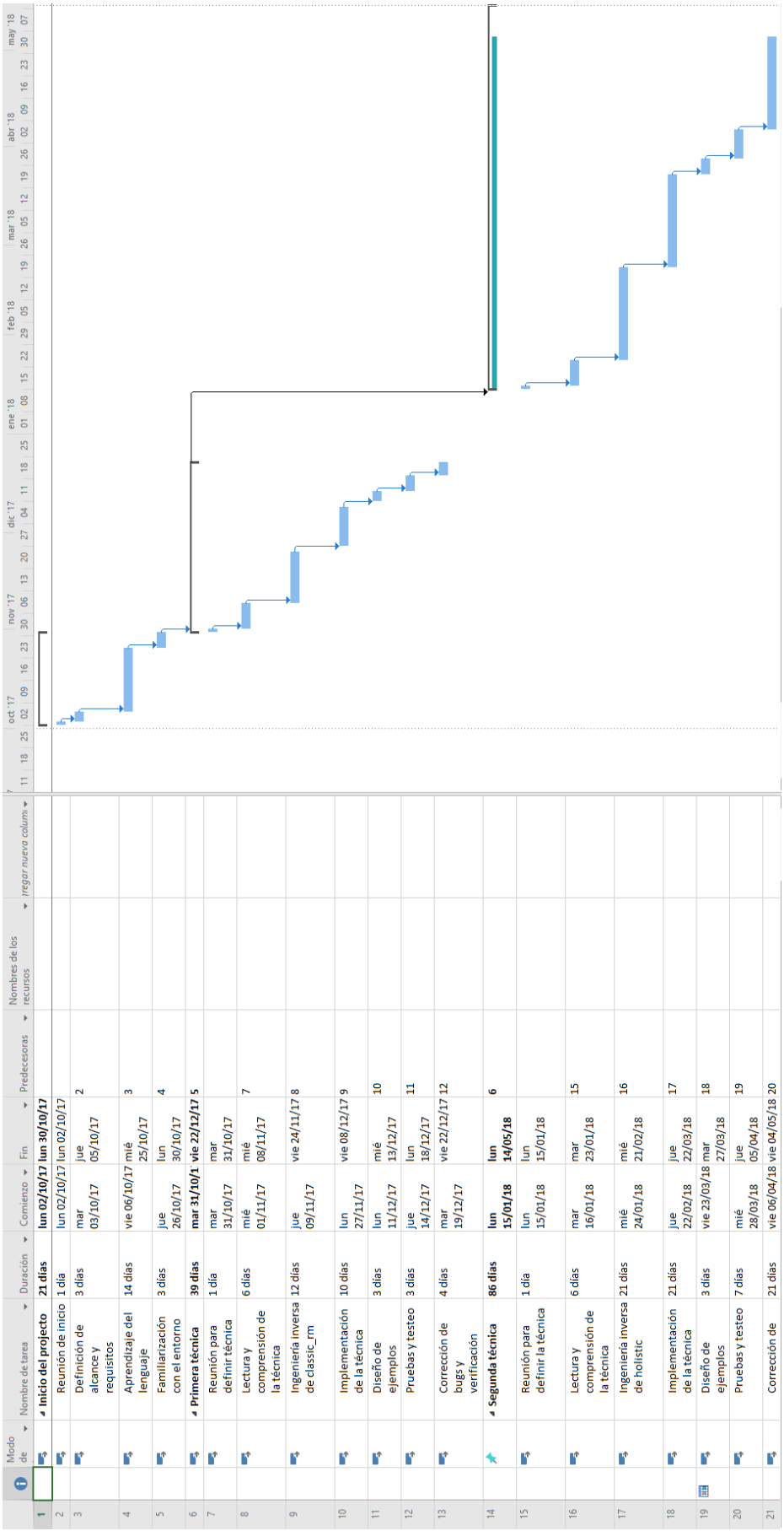
5.2 Trabajos futuros

Como ya se ha mencionado a lo largo de este informe, para el futuro quedaría la implementación de técnicas de análisis basadas en EDF. Dichas técnicas se ha decidido dejarlas fuera del presente TFM debido a que nos pareció más interesante ampliar la primera parte de este y añadir la parte de análisis para sistemas distribuidos, que no simplemente

realizar 2 técnicas de análisis sin relación directa entre sí (la primera para tareas no expulsables, y la segunda para sistemas con planificación EDF).

Por último, también faltaría añadir todos los cambios realizados a una nueva versión estable de MAST, que será publicada en el futuro para su uso, ya que actualmente en el momento de la redacción de este informe, las modificaciones realizadas y las pruebas se han hecho en una versión de testeo. Este trabajo de añadir los cambios a una versión estable de MAST será realizado prácticamente de inmediato una vez terminado el trabajo. Por otro lado, la nueva versión deberá corregir el fallo detectado en sistemas distribuidos en los que no se tiene en cuenta el carácter expulsor o no de las tareas.

Anexo I: Planificación del proyecto



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos	Agregar nueva columna
1		**Inicio del projecto**	**21 días**	**lun 02/10/17**	**lun 30/10/17**			
2		Reunión de inicio	1 día	lun 02/10/17	lun 02/10/17			
3		Definición de alcance y requisitos	3 días	mar 03/10/17	jue 05/10/17	2		
4		Aprendizaje del lenguaje	14 días	vie 06/10/17	mié 25/10/17	3		
5		Familiarización con el entorno	3 días	jue 26/10/17	lun 30/10/17	4		
6		**Primera técnica**	**39 días**	**mar 31/10/17**	**vie 22/12/17**	**5**		
7		Reunión para definir técnica	1 día	mar 31/10/17	mar 31/10/17			
8		Lectura y comprensión de la técnica	6 días	mié 01/11/17	mié 08/11/17	7		
9		Ingeniería inversa de classic_rm	12 días	jue 09/11/17	vie 24/11/17	8		
10		Implementación de la técnica	10 días	lun 27/11/17	vie 08/12/17	9		
11		Diseño de ejemplos	3 días	lun 11/12/17	mié 13/12/17	10		
12		Pruebas y testeo	3 días	jue 14/12/17	lun 18/12/17	11		
13		Corrección de bugs y verificación	4 días	mar 19/12/17	vie 22/12/17	12		
14		**Segunda técnica**	**86 días**	**lun 15/01/18**	**lun 14/05/18**	**6**		
15		Reunión para definir la técnica	1 día	lun 15/01/18	lun 15/01/18			
16		Lectura y comprensión de la técnica	6 días	mar 16/01/18	mar 23/01/18	15		
17		Ingeniería inversa de holistic	21 días	mié 24/01/18	mié 21/02/18	16		
18		Implementación de la técnica	21 días	jue 22/02/18	jue 22/03/18	17		
19		Diseño de ejemplos	3 días	vie 23/03/18	mar 27/03/18	18		
20		Pruebas y testeo	7 días	mié 28/03/18	jue 05/04/18	19		
21		Corrección de	21 días	vie 06/04/18	vie 04/05/18	20		

Anexo II: Guía para nuevas técnicas

-------- ESPAÑOL --------

Añadir una nueva técnica de análisis para sistemas monoprocesadores:

1. En el fichero "mast-monoprocessor_tools.ads":

   -Definir un nuevo procedimiento similar a "RM_Analysis", el cual realizará el análisis de la nueva técnica

2. En el fichero "mast-monoprocessor_tools.adb":

   -Implementar el procedimiento definido en el paso anterior, el cual realizará el análisis de la nueva técnica.

   Aquí contiene toda la lógica de la técnica en sí. (calculos de tiempos de respuesta...)

3. En el fichero "mast-tools.ads":

   -Definir un nuevo procedimiento similar a "Classic_RM_Analysis"

4. En el fichero "mast-tools.adb":

   -Implementar el procedimiento definido en el paso anterior, con las restricciones correspondientes

5. En el fichero "mast-analysis.adb":

   -Añadir un nuevo 'elsif' para el argumento 1, contemplando un nuevo "string" de entrada para la técnica nueva.

   -Añadir el mismo "string" usado anteriormente en el texto mostrado en la excepcion "Wrong_Format", para indicar al usuario la existencia de la nueva técnica

6. Para añadir nuevas restricciones, se puede hacer en los ficheros mast-restrictions.ads y mast-restrictions.adb.

-------- ENGLISH --------

Adding a new analysis technique for monoprocessor systems:

1. In the file "mast-monoprocessor_tools.ads":

   -Define a new procedure similar to "RM_Analysis", which will do the new analysis technique

2. In the file "mast-monoprocessor_tools.adb":

   -Implement the procedure defined in the previous step, which will do the new analysis technique

   Here is where the technique's logic is located (response times calculations...)

3. In the file "mast-tools.ads":

   -Define a new procedure similar to "Classic_RM_Analysis"

4. In the file "mast-tools.adb":

   -Implement the procedure defined in the previous step, with the restrictions

5. In the file "mast-analysis.abd":

   -Add a new 'elsif' for the first paremeter, looking at the new input "string" for the technique

   -Add the same "string" used previously to the help text shown when the exception "Wrong_Format" is thrown, for letting the user to know the existence of the new technique.

6. To add new restrictions, you can define new rules in the files mast-restrictions.ads and mast-restrictions.adb.

Bibliografía

[1] S. e. a. r.-t. g. Unican, «MAST, herramienta de modelado,» [En línea]. Available: https://mast.unican.es/. [Último acceso: 07 Marzo 2018].

[2] AdaCore, «Ada, lenguaje de programación,» [En línea]. Available: https://www.adacore.com/about-ada. [Último acceso: 07 Marzo 2018].

[3] AdaCore, «GNAT, compilador,» [En línea]. Available: https://www.adacore.com/download. [Último acceso: 07 Marzo 2018].

[4] AdaCore, «GPS, IDE de desarrollo,» [En línea]. Available: https://www.adacore.com/community. [Último acceso: 07 Marzo 2018].

[5] Sublime, «Sublime Text 3, editor de texto,» [En línea]. Available: https://www.sublimetext.com/3. [Último acceso: 07 Marzo 2018].

[6] Microsoft, «Microsoft Office, suite de herramientas,» [En línea]. Available: https://products.office.com/es-es/home. [Último acceso: 07 Marzo 2018].

[7] «Microsoft Project,» [En línea]. Available: https://es.wikipedia.org/wiki/Microsoft_Project. [Último acceso: 07 Marzo 2018].

[8] «LaTex Project,» [En línea]. Available: https://www.latex-project.org/. [Último acceso: 15 Marzo 2018].

[9] R. I. Davis, A. Burns y R. J. Bril, «Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised,» Real Time Systems 35 (3), págs. 239-272, 2007.

[10] M. Di Natale, W. Zheng, C. Pinello, P. Giusto y A. Sangiovanni, «Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems,» Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium, Bellevue, Washington, USA, págs. 293-302, 2007.

[11] H. Pérez, J. J. Gutiérrez, M. González Harbour y J. C. Palencia, «The Polling Effect on the Schedulability of Distributed Real-Time Systems,» 21st International Conference on Reliable Software Technologies, Ada-Europe 2016, Pisa (Italy), in Lecture Notes in Computer Science, LNCS Vol. 9695, págs. 179-194, 2016.