Modelado de Sistemas de Tiempo Real Planificados por *RM o DM*: Caracterización y Análisis

José M. Urriza[1], Ricardo Cayssials[1,2], Javier D. Orozco[1,2]

Universidad Nacional del Sur[1]/CONICET[2]
Bahía Blanca – Argentina
{jurriza@criba.edu.ar}

Resumen: Este trabajo aborda el análisis de las funciones que modelan matemáticamente el comportamiento de un Sistema de Tiempo Real (*STR*), planificado por las disciplinas de prioridades fijas *Rate Monotonic* (*RM*) o *Deadline Monotonic* (*DM*). Con este tipo de estudio, es posible razonar los modelos que predicen el comportamiento del *STR* desde otro punto de vista, no tradicional en la disciplina y por lo tanto, se pueden generar nuevos teoremas para reducir el costo computacional (*CC*) de los métodos existentes.

Palabras Claves: *RM*, *DM*, Real Time Systems, *RTS*

1 Introducción

Numerosos trabajos han sido realizados en las últimas décadas modelando el comportamiento de los Sistemas de Tiempo Real (*STR*) planificados por la disciplina de prioridades fijas *RM* o *DM*. El primer trabajo que presenta un modelo matemático que permite predecir el comportamiento del *STR* fue realizado por Joseph y Pandya en 1986 ([1]). Con el modelo en [1] es posible predecir el comportamiento de un *STR* planificado por la disciplina de prioridades fijas *RM* y calcular mediante el método presentado el peor tiempo de respuesta de cada tarea.

En el presente trabajo se realizará un análisis tomando como base el modelo presentado por Joseph en [1], dado que los modelos propuestos en trabajos posteriores hasta la actualidad, en su gran mayoría son similares o modificaciones del mismo.

A continuación se realiza una breve introducción a los *STR*. Se puede definir informalmente a un *STR* como aquél que necesita terminar una tarea o trabajo, antes de un determinado tiempo denominado vencimiento. Éstos se caracterizan por poseer entre sus requerimientos funcionales el tiempo. Por otro lado, es aceptada por la comunidad de la disciplina la definición de Stankovic ([2]) que dice: "*en los STR los resultados no sólo deben ser correctos aritmética y lógicamente sino que además, deben producirse antes de un tiempo determinado denominado vencimiento"*.

Dependiendo del vencimiento de las tareas, los *STR* se pueden clasificar en tres tipos. El primero no permite que ninguna tarea pierda su vencimiento, por lo cual se los denominan *duros* o *críticos*. El segundo permite que se pierda algunos vencimientos, por lo cual se los denominan *blandos*. Por último, la mayor difusión de los *STR* ha requerido tipificar a aquéllos que permiten sólo una determinada cantidad de pérdidas especificadas bajo algún criterio estadístico. Éstos son denominados *firmes*.

En los *STR duros*, la pérdida de un vencimiento en una tarea puede tener consecuencias graves para la integridad del sistema, y posiblemente de su entorno en el caso de que interactúe fuertemente con el mismo, por ejemplo: aviónica, robótica, etc. A fin de garantizar que todas las tareas terminen antes de su vencimiento es necesario realizar un análisis de la *planificabilidad* del sistema. Si el test es exitoso, se dice que el sistema es *planificable* y se puede garantizar el cumplimiento de todas sus *constricciones temporales*.

En [3], se considera la planificabilidad de sistemas mono-recurso y multitarea. Se entiende como mono-recurso, por ejemplo, a un único microprocesador o un medio físico de transmisión, el cual sólo puede ser utilizado por una tarea o mensaje a la vez. Existen dos formas de asignar las tareas al recurso. La primera forma es predeterminada con anterioridad (estática) y la segunda, que es la más utilizada, se basa en fijar una prioridad a cada tarea.

Una *disciplina de prioridades* establece un orden lineal sobre el conjunto de tareas permitiéndole al *planificador* determinar en cada instante de activación, que tarea utilizará el recurso compartido.

Para formalizar el análisis de planificabilidad, resulta necesario modelar al conjunto de tareas en base a sus requerimientos temporales y sus interdependencias. Usualmente se considera que las tareas son periódicas, independientes y apropiables. Una tarea periódica, es aquélla que después de un determinado tiempo solicita nuevamente ejecución. La tarea se dice que es independiente cuando no necesita el resultado de la ejecución de alguna otra tarea, para su propia ejecución. Finalmente se dice que una tarea es apropiable cuando el *planificador*, puede suspender su ejecución y desalojarla del recurso en cualquier momento.

Generalmente los parámetros de cada tarea, bajo este marco de trabajo, son: su tiempo de ejecución, que se nota como C_i, su período T_i y su vencimiento D_i. Por lo tanto un conjunto $\mathbf{S}(n)$ de n tareas se encuentra especificado por $\mathbf{S}(n)=\{(C_1, T_1, D_1), (C_2, T_2, D_2),..., (C_n, T_n, D_n)\}$.

En [3] se demostró que el peor esquema de generación, para un mono-recurso, es aquél en el cual todas las tareas solicitan ser ejecutadas en el mismo instante y se denomina *instante crítico* o *peor estado de carga*. Se demostró también que, si este estado es planificable, el *STR* es planificable para cualquier otro estado, bajo la disciplina de prioridades utilizada.

El factor de utilización (*FU*) de un conjunto de tareas *duras* $\mathbf{S}(n)$ determina el nivel de utilización del recurso.

Este trabajo se divide de la siguiente manera: a continuación se realizara una breve introducción a los *STR*, en la sección 2 se presenta el modelo que describe la evolución del sistema, en la sección 3 se expone un análisis de los *PF* y atractores del

modelo presentado en la sección 2, en la sección 4 se señala una posible mejora a los métodos existentes y finalmente en la sección 5 se presentan las conclusiones y trabajos futuros.

2 Caracterización del Modelo

El modelo en [1] presenta una ecuación trascendente, por lo tanto, la variable se encuentra en ambos lados de la ecuación y no puede ser despejada. A este tipo de ecuación también se la denomina de *Punto Fijo* (*PF*). Para encontrar una solución, si existe, se debe realizar un proceso iterativo. La ecuación que se presentó en [1], con un simple cambio de nomenclatura, fue la siguiente:

$$t^{q+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t^q}{T_j} \right\rceil C_j \quad \rightarrow \quad t = f(t) \tag{1}$$

El método comienza a partir del *instante crítico* ([3]). Las condiciones que detienen al método son: 1) si la tarea analizada es planificable, entonces se encuentra un *PF* tal que $t^q = t^{q+1}$ y $t^{q+1} \leq D_i$, 2) si la tarea no es planificable $t^{q+1} > D_i$, es decir, no se ha encontrado un *PF* antes del vencimiento de la misma en el intervalo $(0, D_i]$ o puede no existir un *PF* debido a que el sistema es sobresaturado.

Todo *PF* es un atractor, por lo tanto comenzado el proceso iterativo, el mismo evoluciona hacia él. Si consideramos un *STR* planificable, el método se detiene en un *PF* que representa el tiempo de respuesta del subsistema **S**(i). En particular como se han considerado los peores tiempo de ejecución (C_i) se dice que es el *peor tiempo de respuesta* de la tarea i o del subsistema **S**(i). Diversos trabajos han sido publicados con modelos equivalentes en [4, 5, 6, 7, 8].

Para analizar el modelo, es necesario presentar un breve resumen sobre la caracterización de los modelos. Si se observa a la Ecuación (1), $f(t)$ es una función monótona creciente y representa a un *sistema dinámico, no-lineal*, *discreto* y *determinístico*, pero en las siguientes secciones se analiza esta afirmación.

2.1 Funciones Monótonas

Una función entre dos conjuntos parcialmente ordenados se dice monótona si preserva el orden, en otras palabras si $x, y \in \mathbb{R}\ \ x \leq y \Rightarrow f(x) \leq f(y)$ (monótona creciente) o $x, y \in \mathbb{R}\ \ x \leq y \Rightarrow f(x) \geq f(y)$ (monótona decreciente).

Una función monótona de $f : \mathbb{R} \rightarrow \mathbb{R}$ con $x \in \mathbb{R}$, posee las siguientes propiedades:

1. $f(x)$ tiene un límite por la izquierda y por la derecha en cualquier punto de su dominio de definición.
2. $f(x)$ Sólo puede tener discontinuidades de salto.
3. $f(x)$ Sólo puede tener una cantidad numerable de discontinuidades.
4. Si $f(x)$ es una función monotónica definida en un intervalo I, entonces $f(x)$ es derivable casi siempre en I, es decir, el conjunto de puntos x en I en donde $f(x)$ no es diferenciable tiene medida de Lebesgue 0 (es un conjunto vacío).

5. Si $f(x)$ es una función monótona definida en un intervalo $[a,b]$, entonces $f(x)$ es Riemann-Integrable.

La función $f(t)$, cumple con las 5 propiedades presentadas anteriormente y por lo tanto es una función monótona creciente. A continuación se realiza el análisis:

1. La función $f(t)$ es monótona creciente, dado que para cualquier $t_1, t_2 \in \mathbb{R}$ con $t_1 < t_2 \Rightarrow f(t_1) \leq f(t_2)$.
2. Existe el límite por izquierda y por derecha para cualquier *t*. La función techo tiene una discontinuidad de salto inmediatamente después de que *t* es múltiplo de T_j. A este tipo de funciones se las denomina *definidas por debajo*.
3. Tiene sólo discontinuidades de salto y son una numerable cantidad, que son las instanciaciones que poseen las tareas del *STR* en el intervalo $(0,t]$.
4. Es derivable casi siempre y el conjunto de instantes en donde $f(t)$ no es diferenciable (salto), tiene medida de Lebesgue 0 (es un conjunto vacío, no existen puntos en el salto).
5. Es una función integrable en todo intervalo que no posea una discontinuidad.

Como se presentará en la sección 3.1 saber que $f(t)$ es monótona permite aplicar el teorema de Kleene de vital importancia.

2.2 Sistemas Dinámicos

Se conoce por *sistema dinámico* a un modelo que describe la evolución temporal de un sistema.

Un *sistema dinámico* se dice *determinístico* si no existe aleatoriedad en la determinación de los estados futuros del sistema y se dice *estocástico* o *aleatorio* si existe una distribución de probabilidad de los posibles consecuentes.

Un *sistema dinámico* se dice *discreto*, si el tiempo se mide en pequeños lapsos y éstos son modelados como relaciones recursivas, como por ejemplo $t^{k+1} = 2\,t^k$, donde *k* denota los pasos discretos de tiempo y *t* es la variable.

Si el tiempo es medido en forma continua, se dice que es un *sistema dinámico continuo* y es expresado como una ecuación diferencial ordinaria; por ejemplo: $dx/dt = ax(1-x)$ donde *x* es la variable que cambia con el tiempo (*t*). La variable *x* es generalmente un número real ($x \in \mathbb{R}$) aunque también puede ser un vector en $\mathbb{R}^n$.

La ecuación presentada en [1], al estar modelada por una relación recursiva, es un *sistema dinámico determinístico y discreto*.

2.2.1 Mapa

En un *sistema dinámico determinístico y discreto*, una función unidimensional se conoce como un mapa ($f:\mathbb{R}\to\mathbb{R}$). El caso más general son las ecuaciones de la forma $t^{k+1} = f(t^k)$. La condición inicial (t^0) de donde parte la iteración de la ecuación, se la denomina *semilla*. La secuencia que se genera al iterar la ecuación, a partir de t^0 de denomina *trayectoria* u *órbita* $t^0, f(t^0), f(f(t^0)), f(f(f(t^0))), \ldots$.

Se dice *telaraña* de un mapa al resultado de iterar *geométricamente* a un mapa (Figura 1).

2.2.2 Sistemas Dinámicos Lineales y No-lineales

Los *sistemas dinámicos* se distinguen entre *lineales* y *no-lineales*. El modelo de un *sistema dinámico es lineal*, si la función posee una relación lineal con su variable (por ejemplo $x^{n+1} = 2x^n$). Además, se debe cumplir el principio de superposición, por el cual, si se conocen dos soluciones, la suma de ellas es también una solución.

En un *sistema dinámico no-lineal*, su comportamiento no puede ser la suma de sus partes. Generalmente son difíciles de modelar y de analizar, además, pueden presentar comportamientos caóticos, por lo cual el sistema tiene soluciones impredecibles.

La variable t en $f(t)$ es de primer orden, la no-linealidad la introduce la función techo ($\lceil\ \rceil$). Es por ello, que el principio de superposición no es válido, es decir la suma de dos soluciones no es solución. Por lo cual, $f(t)$ representa a un *sistema dinámico no-lineal*.

Para concluir esta sección, el modelo propuesto en [1] representa a un *sistema dinámico, no-lineal*, *discreto* y *determinístico.*

3 Puntos Fijos y Atractores

Sea la ecuación $t^{k+1} = f(t^k)$ se denomina *PF* de la ecuación, si existe un t tal que $t = f(t)$, por lo tanto, dada una *semilla* t^0, si la *orbita* cae en un *PF* permanecerá en él en la sucesivas iteraciones dado que, $t^n = t^{n+1}$ y $t^{n+1} = f(t^n)$ para $n \to \infty$.

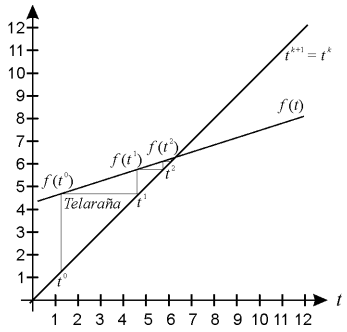

Figura 1 Mapa y telaraña.

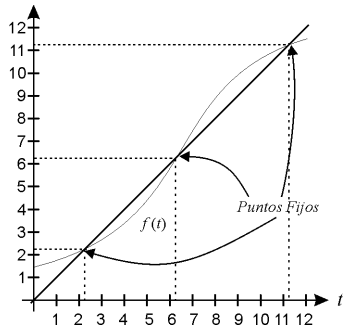

Figura 2 Puntos fijos.

Un *PF* es un tipo de atractor y se denomina generalmente como *PF* atractor. A estos atractores se los llama atractores simples. Existen otros tipos de atractores como ciclo límite, toroide límite, etc. Existe también atractores extraños y se producen cuando la *dinámica* del sistema es *caótica* o cuando poseen *dimensión fractal*.

En un *sistema dinámico*, para cada *atractor* existe una *cuenca de atracción* la cual esta comprendida por el conjunto de condiciones iniciales, por las cuales sus *órbitas* llevan al *atractor*.

3.1 Teorema de Kleene en algoritmos iterativos de Tiempo Real

El teorema de Kleene nos dice que dada una función continua o una función monótona $f: L \to L$ siendo L un conjunto (reticulado completo) parcialmente ordenado con un supremo y un ínfimo, el menor *PF* de una función en una cadena ascendente de Kleene $t^0 \le f(t^0) \le f(f(t^0)) \le f(f(f(t^0))) \le \ldots$ se obtiene de iterar a la función desde el elemento más bajo.

El Teorema de Kleene para que pueda ser aplicado necesita que $t < f(t)$ para encontrar el primer *PF* en la cadena ascendente. Si $t = f(t)$ el sistema se encuentra en un *PF* y si $t > f(t)$ el sistema evoluciona hacia el *PF* anterior como veremos en la siguiente sección.

3.2 *PF* en Sistemas dinámicos no lineales, discretos y determinísticos

Los sistemas dinámicos no lineales, pueden tener un único estado estable (*PF*), múltiples estados estables o no poseer ninguno.

El estudio de la estabilidad que posee un *PF*, está determinado por el cálculo de la primera derivada de la función en el mismo. Como se analizan funciones monótonas crecientes, no se realiza el análisis de los valores negativos de la derivada, dado que nunca ocurren.

La derivada de la recta $t^{k+1} = t^k$ es uno para todo *t*, por lo cual, si la función tiene una derivada menor a uno en el *PF* y también en un entorno $t - \varepsilon < t < t + \varepsilon$ con $\varepsilon > 0$, en funciones continuas suaves significa que la misma cruza la recta $t^{k+1} = t^k$ de $t < f(t)$ a $t > f(t)$. Por lo tanto, en una función monótona creciente, toda iteración *k* debajo de la recta, producirá un valor de la función menor al anterior ($f(t^k) < t^k$), lo cual genera una órbita que retornará hacia el *PF* por donde cruzó y, consecuentemente, se dice que es un *PF* estable. Si la derivada de la función es cero, el *PF* se dice superestable.

Por el contrario, si la función cruza la recta $t^{k+1} = t^k$ de $t > f(t)$ a $t < f(t)$, la derivada de la función en el *PF* es mayor a uno, por lo cual pequeñas variaciones en *t* con $\varepsilon > 0$ se tiene que $f(t^k - \varepsilon) < t^k - \varepsilon < t^{k+1} = f(t^k) < t^k + \varepsilon < f(t^k + \varepsilon)$, por lo tanto pequeñas variaciones, si son negativas causa que la función reduzca su valor, y si son positivas, aumente, así es claramente un *PF* inestable.

En la Figura 3, se presenta un sistema dinámico no lineal, discreto y determinístico, con una función monótona creciente. El mismo posee tres *PFs* de los cuales, dos son estables (1 y 3) y uno inestable (2). Los *PFs* 1 y 3 al ser *PFs* estables poseen una cuenca de atracción. La cuenca de atracción del primer *PF* es de $[0, t_2)$ y la del tercer *PF* es de (t_2, ∞). El *PF* en t_2 es claramente inestable dado que para cualquier $\varepsilon > 0$ queda $f(t_2 - \varepsilon) < t_2 - \varepsilon < t_2 = f(t_2) < t_2 + \varepsilon < f(t_2 + \varepsilon)$, por lo cual la función converge hacia los otros *PFs* o diverge.

En una función monótona creciente se puede presentar otro tipo de *PF*. El mismo ocurre cuando la función del sistema tiene una derivada menor o igual a uno en el punto $f(t)' \le 1$ y luego la derivada para $t + \varepsilon$, con $\varepsilon > 0$, se incrementa $f(t + \varepsilon)' > 1$. En funciones suaves y continuas, la derivada en este tipo de casos es uno, pero en

funciones no suaves y con discontinuidades, como las que se ve en este trabajo, es posible tener que la derivada en el *PF* es 0 y para $\varepsilon > 0$ se encuentra una discontinuidad. En estos casos, la cuenca de atracción en funciones monótonas crecientes termina en el *PF*. Para valores mayores a *t*, la función converge hacia el próximo *PF* o diverge.

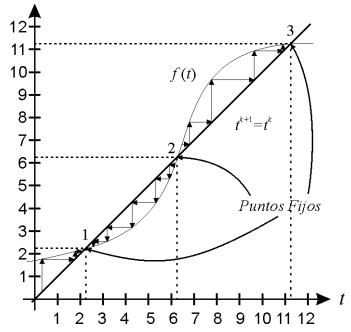

Figura 3 Sistema no lineal con 3 *PFs*.

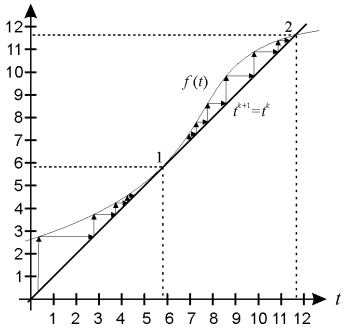

Figura 4 *PF* semiestable.

En la Figura 4 se puede observar que el sistema posee 2 *PFs* (t_1,t_2) para valores de $t > t_1$ la función converge hacia el otro *PF* en t_2. Por lo tanto se comporta como un *PF* estable para $t \le t_1$ e inestable para $t > t_1$ dado que converge hacia el otro *PF*. A este tipo de *PF* se lo define como semiestable.

A diferencia de las funciones suaves, la función techo crea discontinuidades de salto. Debido a estas últimas, la función puede cruzar a la recta $t^{k+1} = t^k$ o sólo tocarla. Esto origina distintos tipos de *PFs* semiestables o superestables en los cuales la derivada es cero.

En ([1]) nada se señala de qué sucede si la función es utilizada después del primer *PF* o si el modelo y el algoritmo sirven para ser utilizados posteriormente. En realidad únicamente determinan el *peor tiempo de respuesta* de una tarea, que es cuando el algoritmo iterativo se detiene en el primer *PF*. Lamentablemente, el modelo y el algoritmo presentados en [1] no cuentan con la posibilidad de extenderlo luego del primer *PF*. Esto se debe a que una vez encontrado el primer *PF*, para poder continuar buscando otros *PFs* se necesita de un mecanismo que salte la cuenca de atracción del *PF* en que se encuentra.

A continuación se presenta un ejemplo. Dado un *STR*, con parámetros **S**(3) = {(1, 4, 4), (2 6, 6), (1 10, 10)} se procede a analizar la planificabilidad de la tarea 3, por el método presentado en [1]. Cabe aclarar que la función representada en las figuras tiene una discontinuidad cada vez que se produce un escalón, por lo cual, cuando la función cruza verticalmente a la recta *t* no existe un *PF*. En la Figura 5 se puede observar esto para *t* =10.

Como se observa en la Figura 5 y de su análisis en la Tabla 1, el *STR* encuentra un *PF* en *t* = 4, el cual es semiestable. Además, para valores de semilla entre $(4,6]$ se obtiene un *PF* en *t* = 5 y para valores de semilla en el intervalo $(6,10]$ que se obtiene un *PF* en *t* = 7.

Si bien el análisis de los *PFs* es correcto, ¿Representan los *PFs* en *t* = 5 y *t* = 7 algún estado del *STR*? No. El problema es, que el modelo ([1]) no considera el tiempo ocioso que el sistema deja libre en el intervalo (5, 6]. Por lo cual, si bien es un *PF* del

sistema, no proporciona directamente ningún dato relevante del *STR*. Aún así en la siguiente sección se verá que esto puede ser útil.

n	t	W
0	0	$\lceil 0/4 \rceil .1 + \lceil 0/6 \rceil .2 + 1 = 1$
1	1	$\lceil 1/4 \rceil .1 + \lceil 1/6 \rceil .2 + 1 = 4$
2	4	$\lceil 4/4 \rceil .1 + \lceil 4/5 \rceil .2 + 1 = 4$

Tabla 1. Ejemplo con el Método de Joseph.

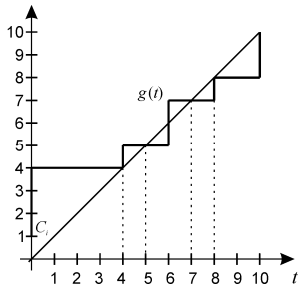

Figura 5 Ejemplos de Puntos fijos.

4 Nuevas Proposiciones

En esta sección se enuncian dos teoremas y un colorario que se desprenden del análisis efectuado en las secciones anteriores que permiten establecer otro tipo de razonamiento de cómo puede ser posible generar, por ejemplo, un test de planificabilidad sin tener la necesidad de efectuar el cálculo del peor tiempo de respuesta para establecer una condición necesaria y suficiente. En lo que sigue se denominará PF_i al instante en donde se encuentra el primer *PF* de la tarea *i*.

En [6, 9], se prueba que la distancia entre dos *PFs* es menor o igual a la distancia entre el instante crítico y el primer *PF*. Resulta entonces que, si cada vez que se comienza el proceso iterativo se realiza desde el instante crítico, el primer *PF* es el más costoso de encontrar un desde el punto de vista del *CC*.

La primera solución para subsanar este inconveniente la realizo Sjödin ([10]). Aunque no se trató al modelo del *STR* desde del punto de vista de la teoría de *PFs*, se estableció que se puede continuar la iteración de la tarea i +1, a partir del instante igual a PF_i , más el tiempo de ejecución de la tarea i +1 ($PF_i + C_{i+1}$).

Como ejemplo de lo señalado en las secciones anteriores, desde la teoría de *PF*, es posible realizar la prueba de la mejora establecida por Sjödin en [10] fácilmente. A continuación se enuncia:

"*En un sistema basado en prioridades fijas, la ejecución del subsistema* **S**(i) *ocupa todo el intervalo* $(0, PF_i]$ *por lo tanto, si el subsistema* **S**(i+1) *posee un PF, este se encontrara luego de que la tarea i +1 sea ejecutada, con lo cual, será en un instante tal que* $PF_{i+1} \geq PF_i + C_{i+1}$ *dentro de* $(0, D_i]$*".*

Dado que el primer *PF* es el más costoso de encontrar, ¿Es necesario encontrarlo para probar la planificabilidad del *STR*? No. En los siguientes teoremas se presenta una nueva forma de analizar la planificabilidad que permite crear un nuevo método.

Teorema 1:

Sea la tarea $i \in$ **S**(i)*, si el proceso iterativo partiendo de una semilla posterior al* PF_i *encuentra un PF dentro del intervalo* $(0, D_i]$*, la tarea i es planificable.*

Prueba:

La existencia de un *PF* posterior al primero, dentro del intervalo $(0, D_i]$ prueba la existencia del primero, por lo cual la tarea *i* es planificable por [1]. □

Como se estableció en la sección 3.2, el encontrar un *PF* posterior al primero no establece con el modelo presentado por Joseph ninguna condición particular del sistema, sin embargo prueba la existencia del primer *PF*.

Teorema 2:

Sea la tarea $i \in \mathbf{S}(i)$*, si existe algún* $t \in (0, D_i]$ *que* $t > f(t)$ *entonces la tarea i es planificable.*

Prueba:

En un *STR* sólo es posible que sea $t > f(t)$ con $t \in (0, D_i]$ si existe un PF_i en un instante anterior $(PF_i < t)$, por lo cual prueba la existencia de PF_i y por [1] la tarea *i* es planificable. □

Colorario 1:

Si $t > f(t)$ *con* $t \in (0, D_i]$ *entonces existe tiempo ocioso en* $(0, D_i]$.

En base a estos dos teoremas y su colorario es posible crear o perfeccionar los métodos que permite determinar la planificabilidad de una tarea sin la necesidad de encontrar el primer *PF* y así reducir el *CC*.

5 Conclusiones y Trabajos Futuros

En este trabajo se presentó la caracterización y análisis del modelo que permite predecir a los *STR* planificados por las disciplinas de prioridades fijas *RM* o *DM,* desde un punto de vista no tradicional para la disciplina y a partir de la teoría de *Punto Fijo*. Este tipo de análisis se agrega como una valiosa herramienta que permite analizar, crear y sustentar nueva teoría de tiempo real de una manera más sencilla. Como se presentó en la sección anterior, el resultado de aplicar este análisis, permite generar nuevos teoremas y colorarios con el fin de reducir el *CC* de los métodos actuales.

En futuros trabajos se aplicará esta herramienta de análisis, para crear y perfeccionar los métodos que: analizan la planificabilidad de los *STR* ([5, 8, 10, 11, 12, 13]) y que establecen en tiempo de ejecución, cuánto tiempo ocioso deja disponible el *STR* (método *Slack Stealing* ([14, 15, 16])), los cuales son de enorme importancia a la hora de tratar *STR* heterogéneos, tan utilizados hoy en dispositivos móviles como teléfonos celulares, Palms, notebooks, etc.

Referencias

[1] M. Joseph and P. Pandya, "Finding Response Times in Real-Time System," *The Computer Journal (British Computer Society),* vol. 29, pp. 390-395, 1986.

[2] J. A. Stankovic, "Misconceptions About Real-Time Computing: A Serius Problem for Next-Generations Systems," *IEEE Computer,* vol. Octubre, pp. 10-19, 1988.

[3] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM,* vol. 20, pp. 46-61, 1973.

[4] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," Department of Statistics, Carnegie-Mellon, Pitsburg, USA, Internal Report 1987.

[5] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior," in *IEEE Real-Time Systems Symposium*, 1989, pp. 166-171.

[6] J. Santos, M. L. Gastaminza, J. D. Orozco, D. Picardi, and O. Alimenti, "Priorities and Protocols in Hard Real-Time LANs," *Computer Communications,* vol. 14, pp. 507-514, 1991.

[7] J. Santos and J. D. Orozco, "Rate Monotonic Scheduling in Hard Real-Time Systems," *Information Processing Letters,* vol. 48, pp. 39-45, 1993.

[8] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings, "Applying New Scheduling Theory to Static Priority Preemptive Scheduling," *Software Engineering Journal,* vol. 8, pp. 284-292, 1993.

[9] R. I. Davis, "Dual Priority Scheduling: A Means of Providing Flexibility in Hard Real-Time Systems," Department of Computer Science, University of York, York, England, Internal Report 1995.

[10] M. Sjödin and H. Hansson, "Improved Response-Time Analysis Calculations," in *IEEE 19th Real-Time Systems Symp.*, 1998, pp. 399-409.

[11] J. M. Urriza, C. Ricardo, and J. D. Orozco, "Test Rápido de Planificabilidad para R.M. o D.M.," in *XXXI Conferencia Latinoamericana de Informática. CLEI*, Colombia, Cali, 2005.

[12] J. P. Lehoczky, "Fixed Priority Scheduling of Periodic Task Sets With Arbitrary Deadline," in *Proceedings 11Th IEEE Real-Time Systems Symposium, Lake Buena Vista, Fl, USA*, 1990, pp. 201-209.

[13] N. C. Audsley, A. Burns, M. F. Richarson, and A. J. Wellings, "Hard Real-Time Scheduling: The Deadline Monotonic Approach," in *Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software*, Atlanta, GA, USA 1991.

[14] J. M. Urriza, J. D. Orozco, and R. Cayssials, "Fast Slack Stealing methods for Embedded Real Time Systems," in *26th IEEE International Real-Time Systems Symposium (RTSS 2005) - Work In Progress Session*, Miami, EEUU, 2005, pp. 12-16.

[15] R. I. Davis, K. W. Tindell, and A. Burns, "Scheduling Slack Time in Fixed-Priority Preemptive Systems," *Proceedings of the Real Time System Symposium,* pp. 222-231, 1993.

[16] J. P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems," in *IEEE Real-Time Systems Symposium*, Phoenix, Arizona, EUA, 1992, pp. 110-123.