Universidade Federal de Santa Catarina | www.periodicos.ufsc.br ISSNe 2175-8018
IJIE
IBEROAMERICAN JOURNAL OF INDUSTRIAL ENGINEERING
REVISTA IBEROAMERICANA DE ENGENHARIA INDUSTRIAL
REVISTA IBEROAMERICANA DE INGENIARÍA INDUSTRIAL

INSTANTE CRÍTICO CON JITTER: UN ESTADO POCO PROBABLE

Francisco Ezequiel Páez[1]
José Manuel Urriza[2]
Mariano Ferrari[3]
Ricardo Cayssials[4]
Javier Orozco[5]

RESUMEN: Este trabajo trata sobre la probabilidad de encontrar un instante crítico con los peores casos de *jitter* de instanciación en un Sistema de Tiempo Real. Se analizará mediante simulaciones, que esta condición de planificación acontece muy rara vez, más aún cuando los sistemas poseen numerosas tareas. Consecuentemente, basar el diseño del sistema en esta condición puede provocar implementaciones de hardware sobredimensionadas para el funcionamiento normal del mismo. Por otro lado, esta baja probabilidad introduce un gran pesimismo en los métodos de evaluación de planificabilidad.

Palabras clave: Sistemas de Tiempo Real. Jitter. Planificabilidad.

1 INTRODUCCIÓN

Este trabajo aborda el problema del *jitter* de instanciación en sistemas de tiempo real (STR). Se define como *jitter* de instanciación, al tiempo que le insume al sistema desde el arribo de una tarea, a tenerla lista para ejecutar. Este tiempo es acotado y no afecta al período de la tarea. Por ejemplo, en un Sistema Operativo de Tiempo Real (SOTR), es el tiempo que

[1] Licenciado en Informática, Universidad Nacional de la Patagonia San Juan Bosco, Depto. de Informática, Facultad de Ingeniería, Argentina. E-mail: fpaez@unp.edu.ar.

[2] Dr. en Ingeniería, Universidad Nacional de la Patagonia San Juan Bosco, Depto. de Informática, Facultad de Ingeniería, Argentina. E-mail: josemurriza@unp.edu.ar.

[3] Dr. en Matemática, Universidad Nacional de la Patagonia San Juan Bosco, Depto. de Matemática, Facultad de Ingeniería, Argentina. E-mail: mferrari7@gmail.com.

[4] Dr. en Ingeniería, Universidad Nacional del Sur, Depto. de Ing. Eléctrica y Computadoras, Argentina. E-mail: ricardo.cayssials@uns.edu.ar.

[5] Dr. en Ingeniería, Universidad Nacional del Sur, Depto. de Ing. Eléctrica y Computadoras, Argentina. E-mail: jorozco@uns.edu.ar.

invierte desde que arriba la tarea, hasta tenerla en la cola de listas para ejecutar. En lo que sigue, cada vez que se mencione *jitter* se estará refiriendo al *jitter* de instanciación.

Al considerar el *jitter* en la etapa de desarrollo, los diseñadores utilizan algún test de planificabilidad. Este test, por lo general, se basa en calcular el peor caso de tiempo de respuesta de una tarea, en la peor condición posible del sistema. Si el conjunto de tareas de un sistema dado, pasa este test, el sistema se dice planificable.

El análisis del peor caso de tiempo de respuesta de una tarea fue estudiado en numerosos trabajos (LIU; LAYLAND, 1973; JOSEPH; PANDYA, 1986), y con *jitter* en Rajkumar (1990), Tindell (1993), Audsley et al. (1993), Palencia y Gonzalez Harbour (1998), Redell y Torngren (2002), Richard y Goossens (2006), entre otros. En estos últimos trabajos, se presenta cómo la interferencia de tareas de mayor prioridad con *jitter*, puede resultar en la pérdida de vencimientos para las tareas de menor prioridad.

Sin embargo, estos test son para algunos casos de extremo pesimismo, dando como resultado que se descarten posibles configuraciones o sistemas, que aun siendo planificables, el test falla en su evaluación.

En las siguientes secciones se evaluará la probabilidad de que ocurra el peor caso de tiempo de respuesta considerando el *jitter* de cada tarea. Consecuentemente, se mostrará mediante simulaciones que este caso es muy poco probable y consiguientemente los test de planificabilidad tradicionales, introducen un fuerte pesimismo en la planificabilidad.

El resto del trabajo se organiza de la siguiente manera. A continuación se realiza una breve introducción a los STR. En la sección 2 se introduce el problema que causa el *jitter* de instanciación. En las secciones 3 y 4 se presenta el método y el algoritmo desarrollado para el cálculo del peor instante crítico con *jitter*. En la sección 5, se exhiben las simulaciones realizadas y sus resultados. Finalmente, en la sección 6 se presentan las conclusiones y los trabajos futuros.

2 INTRODUCCIÓN A LOS SISTEMAS DE TIEMPO REAL

Los Sistemas de Tiempo Real (STR) en la actualidad se encuentran en un sinnúmero de dispositivos, sin que por ello se percaten los usuarios. Es posible encontrarlos en dispositivos para la industria en aplicaciones de control, instrumentación para la milicia, aviónica, electrodomésticos, computadoras de automóviles, PDA, teléfonos celulares, etc.

Los diseñadores de estos dispositivos utilizan herramientas que permiten validar que el STR funcione de manera correcta y así garantizar que estos dispositivos cumplen con la misión para la cual fueron concebidos. Es común afirmar en la disciplina que, en los STR los resultados no sólo deben ser correctos desde un punto de vista aritmético-lógico, sino que además deben ser obtenidos antes de un determinado instante denominado vencimiento (STANKOVIC, 1988).

El vencimiento es una constricción de tiempo y es un parámetro que posee cada tarea del STR. Debido a esto, los STR se clasifican en duros si no toleran perdidas, en blandos si toleran algunas pérdidas y en firmes si el número de perdidas está acotado.

En Liu y Layland (1973) se definió uno de los primeros modelos de tareas y a partir de él, fue posible construir modelos de sistemas dinámicos, no-lineales, discretos y determinísticos (URRIZA; CAYSSIALS; OROZCO, 2008), que permiten analizar la evolución temporal de un STR como los presentados en Joseph y Pandya (1986), Audsley et al. (1993).

También, en Liu y Layland (1973) se demostró que el peor estado de carga de un sistema mono-recurso es aquel en el cual todas las tareas periódicas requieren ser atendidas en el mismo instante, y se denomina instante crítico. Cuando el sistema comienza desde este instante y en su evolución temporal cumple con todas las constricciones de tiempo, impuestas a la primera instancia de cada tarea, se dice que el STR es planificable. El mínimo común múltiplo entre los periodos de las tareas se denomina hiperperíodo, y se lo representa como H. Debido a la periodicidad de las tareas, cualquier instante de carga del sistema se repite en el hiperperíodo del sistema.

En este trabajo se considera que las tareas son determinísticas, independientes, apropiables, y que las mismas finalizan antes del arribo de una nueva instancia, o son desechadas. Consecuentemente, no existen instancias anteriores de una misma tarea esperando ser ejecutadas. Además, se considera que el periodo de una tarea se mantiene invariante, a lo largo del hiperperíodo y consecuentemente, se mantiene contante el tiempo inter-arribo.

A continuación, en la Tabla 1 se detallan los parámetros del modelo de tareas utilizado, el cual es similar al presentado en los trabajos de Liu y Layland (1973), Audsley et al. (1993) entre otros.

Tabla 1 – Modelo de tarea

Tiempo de ejecución (C_i)	Máximo tiempo que podría tomar la ejecución de una tarea. Normalmente se define como el *peor caso de tiempo de ejecución* (*WCET*).
Período (T_i)	Intervalo de tiempo entre arribos de una tarea periódica.
Vencimiento (D_i)	Instante de tiempo máximo, relativo a la instanciación de la tarea, en que la misma debe completar su ejecución.
Tiempo de bloqueo (B_i)	Máximo tiempo en que una tarea podría esperar por tareas de menor prioridad con las que comparte recursos.
Jitter de instanciación (J_i)	En su caso más común, es el tiempo insumido por el *Sistema Operativo de Tiempo Real* en tener lista para ejecutar una tarea *i* desde su arribo.
Offset (Of_i)	Máximo desplazamiento del tiempo de arribo de una tarea respecto del instante crítico.

En lo siguiente, para especificar un sistema de *n* tareas, se utilizará la siguiente notación: $S(n) = \{(C_1, T_1, D_1, B_1, J_1, Of_1),\ldots, (C_n, T_n, D_n, B_n, J_n, Of_n)\}$.

3 EL PROBLEMA DEL JITTER DE INSTANCIACIÓN

Cuando las tareas de un *STR* poseen *jitter*, con $J_i^{\max} \geq 0$, puede existir un nuevo instante crítico, al que se denominará *peor instante crítico por jitter* (*PICJ*). Este, por lo general, no se produce con el arribo simultáneo de todas las tareas, sino cuando todos los *jitters* máximos de cada tarea se alinean en un instante (ver Figura 1). En este caso, el planificador debe diagramar la ejecución del sistema, con este nuevo *peor caso de estado de carga* (TINDELL, 1993), teniendo un menor tiempo cada tarea para completar su ejecución antes del vencimiento.

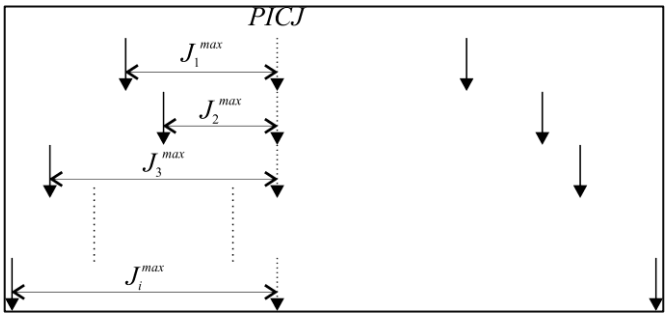
Esto conlleva, a que exista un nuevo *peor caso de tiempo de respuesta*, ahora con *jitter*, para cada tarea y consecuentemente se debe analizar si este nuevo estado permite que el *STR* sea planificable.

A continuación se enuncian las condiciones que llevan a que acontezca el *peor caso de tiempo de respuesta para una τ_i con jitter* (AUDSLEY et al., 1993):

i. La τ_i posee su *peor caso de tiempo de ejecución*, conjuntamente con todas las instancias de las tareas de mayor prioridad hasta que la misma finalice su ejecución[6].
ii. Ocurre un *peor instante crítico por jitter* (*PICJ*).
iii. Todas las siguientes instancias de las tareas de mayor prioridad, posteriores al *PICJ*, posean un *jitter* mínimo o nulo.

[6] Note que pueden existir ante que la tarea τ_i finalice su ejecución, múltiples instancias de tareas de mayor prioridad para su ejecución.

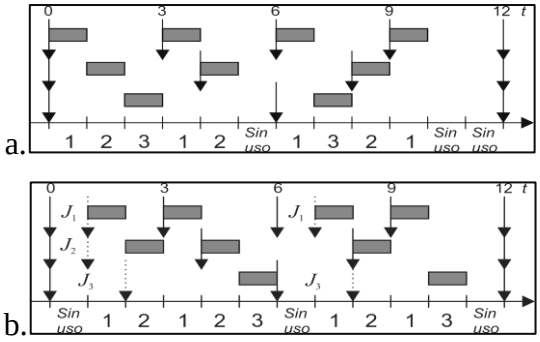
Figura 1 – Ejemplo genérico del *Peor Instante Crítico con Jitter* (*PICJ*)



Si se consideran estas condiciones, a partir del *PICJ*, comienza un intervalo de tiempo en donde una tarea con prioridad *i*, no puede ejecutarse hasta que todas las instancias de las tareas de mayor prioridad que conformaron el *PICJ* o que arribaron posteriormente, sean ejecutadas. De esta manera, el intervalo de tiempo que la τ_i debe esperar es máximo.

Estas condiciones introducen un pesimismo extremo, dado que se tienen que dar los peores casos antes expuestos, para que la tarea sufra su *peor caso de tiempo de respuesta*. A continuación se presenta en un ejemplo. Dado el sistema *S*(3) = {(1,3,3,0,1,0), (1,4,4,0,1,0), (1,6,6,0,1,0)}, en la Figura 2a se expone la ejecución del sistema sin *jitter* y en la Figura 2b la tarea 3 sufriendo el *peor caso de tiempo de respuesta con jitter*.

Figura 2 – (a) Sistema sin *jitter*. (b) Sistema con *jitter* para la tarea 3

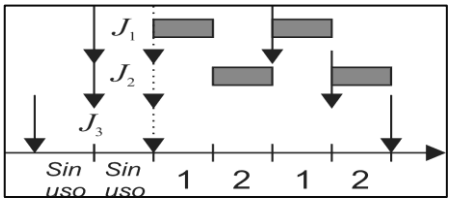


Como se puede apreciar el sistema es planificable para ambos casos. Sin embargo, si se realiza el análisis de planificabilidad introducido por Audsley et al. (1993) con la Ecuación Iterativa 1, resulta que la tarea 3 termina en el instante 7. Pero en el instante 6 ya ha arribado una nueva instancia de la tarea 3 y consecuentemente, según el análisis, la misma ha perdido su vencimiento.

$$w_i^+ = C_i + B_i + \sum_{j=1}^{i-1} \left\lceil \frac{w_i + J_i - Of_i}{T_j} \right\rceil C_j \qquad R_i = w_i + J_i \quad \text{con} \quad R_i \le D_i \tag{1}$$

Esto se debe a que la ecuación considera un *PICJ* que nunca acontece, como se puede deducir de la Figura 3, donde se presenta el peor caso que nunca ocurre y la Figura 2b que es la ejecución real en el hiperperíodo.

Figura 2 – Ejemplo de *PICJ*



Se debe tener en cuenta, que se consideran invariantes los periodos de las tareas, caso contrario el análisis introducido por Audsley et al. (1973) puede acontecer.

En los trabajos de Redell y Torngren (2002), Palencia y Gonzalez Harbour (1998), se identifican el instante crítico que ocasiona un peor caso de tiempo de respuesta para una tarea, en base a los peores casos de *jitter*. Sin embargo, en ninguno de los dos trabajos, se realiza un análisis a fin de conocer la frecuencia con la cual es posible encontrar dicho peor estado de carga con *jitter* en *STR*. No obstante, en Palencia y Gonzalez Harbour (1998), se reconoce la posibilidad de que dicho instante no exista para un *STR* dado,

A continuación, se analizará mediante simulación cuál es la probabilidad de que el *PICJ* ocurra. En las siguientes secciones, los experimentos realizados expondrán que a medida que aumenta el número de tareas, la probabilidad de que exista un *PICJ* decrece aproximadamente de manera exponencial.

4 MÉTODO DE BÚSQUEDA DEL *PICJ*

En esta sección, se presenta un método de búsqueda del *PICJ*. Se comienza analizando el caso particular entre dos tareas de tiempo real, τ_1 y τ_2, con períodos $T_1 \le T_2$ y los peores casos de *jitter* J_1 y J_2. El análisis se realiza sin tener en cuenta el *offset* que pueden poseer las tareas. En la siguiente sección se analizará el caso con *offset*.

Existen dos casos simples que no necesitan de la aplicación de método alguno. El primer caso es que J_1 y J_2 sean idénticos, luego el cálculo del *PICJ* es trivial, ya que este ocurre en el instante $t = J_1 = J_2$. El segundo caso se da si los valores de los períodos de las

tareas son iguales ($T_1 = T_2$), y los valores de *jitter* difieren ($J_1 \neq J_2$), entonces un *PICJ* entre ambas tareas no es posible.

La búsqueda del *PICJ* para el resto de los casos, se puede limitar sólo a aquellos instantes dónde la tarea 1 se instancia con *jitter* máximo, dentro del hiperperíodo generado por el subsistema de dos tareas (τ_1 y τ_2), H_2. Estos instantes conforman el conjunto $\mathbf{M}_1$:

$$\mathbf{M}_1 = \{m \cdot T_1 + J_1 \mid 0 \leq m < H_2 / T_1\}$$

Con cada valor de *m* se obtiene un instante de arribo de τ_1, con su peor caso de *jitter*, dentro de H_2. Luego, el *PICJ* entre ambas tareas existe para algún $t \in \mathbf{M}_1$ cuando se cumple la siguiente igualdad (Ecuación 2):

$$t - T_2 \left\lfloor \frac{t}{T_2} \right\rfloor = J_2 : \quad t \in \mathbf{M}_1 \tag{2}$$

Esta ecuación proviene de igualar J_2 con el resto de dividir $t = m \cdot T_1 + J_1$ por T_2 y, por lo tanto, *m* es solución de la congruencia lineal $m \cdot T_1 = J_2 - J_1 \pmod{T_2}$. Para un *t*, que satisface la anterior igualdad, el *PICJ* entre ambas tareas ocurre en el instante $t = m \cdot T_1 + J_1 = m' \cdot T_2 + J_2$, al que se denominará $picj_1$ del subsistema de dos tareas. Si ningún valor $t \in \mathbf{M}_1$ satisface la igualdad, no existe un *PICJ* para el sistema de 2 tareas. Además, sea *g* el máximo común divisor entre T_1 y T_2, entonces se puede ver que $picj_1$ existe si y sólo si $J_2 - J_1 \pmod{T_2}$ es divisible por *g* y, en caso de existir, es único dentro de H_2.

Se considera ahora una tercera tarea, τ_3. Para que exista un $picj_2$ del subsistema de 3 tareas, es condición necesaria que exista un $picj_1$ del subsistema de 2. Esto se debe a que el *PICJ* de 3 tareas contiene un *PICJ* de 2 tareas, como se puede ver claramente en las Figuras 1 y 3.

Se debe tener en cuenta que el $picj_1$ contiene a las tareas τ_1 y τ_2 con su máximo *jitter*. Por lo tanto, no es necesario tener en cuenta los *jitters* de ambas para el cálculo de los instantes donde es posible encontrar el $picj_2$. Este, si existe, ocurre dentro del siguiente conjunto $\mathbf{M}_2$, que corresponde a aquellos instantes congruentes con $picj_1 \pmod{H_2}$ en uno de los siguientes instantes del conjunto:

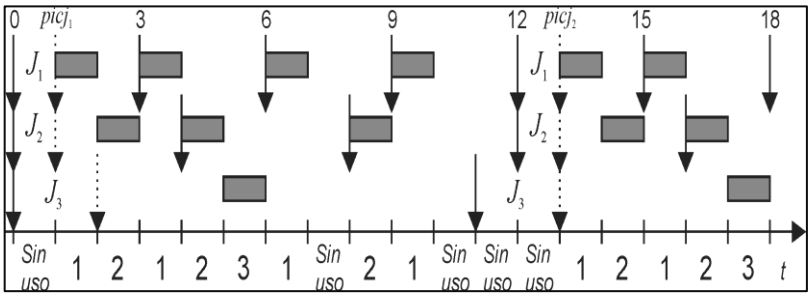
$$\mathbf{M}_2 = \{(m \cdot H_2) + picj_1 \mid 0 \leq m < H_3 / H_2\}$$

Como antes, el $picj_2$ ocurre si se cumple (Ecuación 3):

$$t - T_3\left\lfloor \frac{t}{T_3} \right\rfloor = J_3 \ : \ t \in \mathbf{M}_2 \tag{3}$$

En el ejemplo de la Figura 4 se presenta el sistema $S(3) = \{(1,3,3,0,1,0), (1,4,4,0,1,0), (1,11,11,0,2,0)\}$, donde se puede apreciar como el $picj_2$ ocurre en aquellos instantes congruentes con el $picj_1$. Estos se encuentran en el siguiente hiperperíodo del subsistema de 2 tareas (t = 13).

Figura 3 – Ejemplo de $picj_1$ y $picj_2$



En este momento se puede generalizar el método para n tareas. Para ello se considera que no existe ningún caso trivial, y que $picj_{n-2}$ representa el instante donde ocurre el *PICJ* para el subsistema n-1. De igual manera que para el sistema de 3 tareas, el $picj_{n-1}$ solo existe si también es un $picj_{n-2}$, $picj_{n-3}$,...., $picj_2$, $picj_1$. El *PICJ* para n tareas ocurre si se satisface (Ecuación 4):

$$\begin{gathered} \forall i = 3,...,n \\ \exists\, t \in \mathbf{M}_{i-1} = \left\{ (m \cdot H_{i-1}) + picj_{i-2} : 0 \le m < H_{i-1}/H_{i-2} \right\} \\ \text{tal que } \ t - T_i\left\lfloor \frac{t}{T_i} \right\rfloor = J_i \end{gathered} \tag{4}$$

Para el caso en que dos tareas tengan el mismo valor de *jitter*, y distintos periodos, la igualdad se satisface para m = 0. En caso de que dos tareas τ_{i-1} y τ_i tengan el mismo período la cardinalidad es $|\mathbf{M}_{i-1}| = 1$.

4.1 Extensión del Método con *Offset*

El corrimiento desde el origen (*offset*) puede ocasionar que ocurra o no el *PICJ*. Consecuentemente, se presenta a continuación el análisis extendido para este caso (Ecuación 5).

$$\mathbf{M}_1 = \left\{ m \cdot T_1 + J_1 + Of_1 \mid 0 \le m < H_2 / T_1 \right\}$$

$$t - Of_2 - T_2\left\lfloor \frac{t}{T_2} \right\rfloor = J_2 : \ t \in \mathbf{M}_1 \tag{5}$$

Si se generaliza para *n* tareas resulta (Ecuación 6):

$$\begin{gathered} \forall i = 3,...,n \\ \exists\, t \in \mathbf{M}_{i-1} = \{m \cdot H_{i-1} + picj_{i-2} : 0 \le m < H_{i-1}/H_{i-2}\} \\ \text{tal que } \ t - Of_i - T_i\left\lfloor \frac{t}{T_i} \right\rfloor = J_i \end{gathered} \tag{6}$$

5 ALGORITMO DE BÚSQUEDA

En esta sección se presenta y se describe el algoritmo de búsqueda del *PICJ*. A continuación se presenta el pseudocódigo:

```
function hasPICJ (STR):
 for (j = 2 .. nroTareas) do                    // Sección 1
  if (J1 + Of1 ≠ Jj + Ofj) then break;
 end for;
 picj = Hj-1 + Jj-1 + Ofj-1;
 for (i = j .. nroTareas) do                    // Sección 2
  while (picj < Hi) do
    resto = picj − Ofi − Ti ⌊picj/Ti⌋;
    if (resto == Ji) then continue for;
    picj = picj + Hi-1;
  end while;
  return false;
 end for;
 return true;
end function;
```

La Sección 1 del algoritmo, verifica si las *n* tareas del *STR* cuentan con valores de *offset* y peores casos de *jitter*, tal que $J_1 + Of_1 = J_2 + Of_2 = ... = J_n + Of_n$. De ser así, el cálculo del *PICJ* resulta trivial, y sucede en $t = J_1 + Of_1 = J_2 + Of_2 = ... = J_n + Of_n$. De darse esta condición, el ciclo de la Sección 2 no se ejecuta.

Caso contrario, sea τ_j la primer tarea con un peor caso de *jitter* y *offset* tal que $J_1 + Of_1 = ... = J_{j-1} + Of_{j-1} \neq J_j + Of_j$. Luego, se tiene que $picj = H_{j-1} + J_{j-1} + Of_{j-1}$, ya que para todas las

tareas del subsistema *j*-1 se cumple que $J_1 + Of_1 = ... = J_{j-1} + Of_{j-1}$. Para el caso en que las dos primeras tareas no satisfacen una solución trivial, se tiene que $picj = H_1 + J_1 + Of_1 = T_1 + J_1 + Of_1$.

A continuación, en la Sección 2, el algoritmo analiza el resto de las tareas del *STR* a partir de la τ_j. Por cada τ_i, con $j \le i \le n$, se verifica en cada instante del conjunto $\mathbf{M}_{i-1}$ si existe un *PICJ*. Si dos tareas tuviesen el mismo periodo, debe cumplirse que $J_{i-1} + Of_{i-1} = J_i + Of_i$, para que exista un *PICJ*. Caso contrario, al no existir otro instante que inspeccionar en $\mathbf{M}_{i-1}$, el peor caso no sucede.

Si se analizan todos los instantes en el conjunto $\mathbf{M}_{i-1}$ sin éxito, significa que no existe el $picj_{i-1}$.

6 SIMULACIONES Y RESULTADOS

A continuación se describirán los conjuntos de datos empleados en las simulaciones, y los resultados obtenidos en las mismas.

Para la generación de los *STR* se utilizó el software que se describe en Olguín, Biscayart y Urriza (2012). Se generaron dos conjuntos de *STR*, con valores de períodos distribuidos uniformemente entre 25-100.000 y 25-1.000.000, respectivamente. A su vez, se generaron subconjuntos de 20, 50 y 100 tareas para cada conjunto. Para cada valor de factor de utilización entre 10 y 100, en saltos de 10, se generaron 100.000 *STR*. En total, el número de *STR* generados fueron 6.000.000 (6 millones).

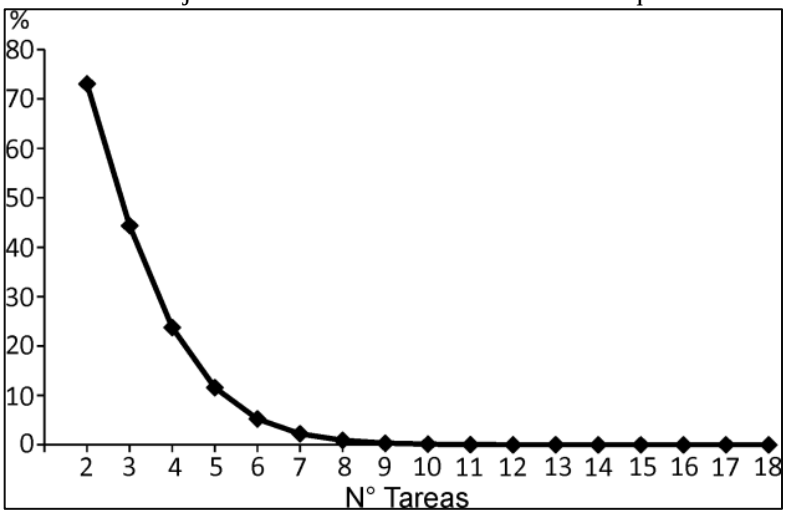
El *jitter* para cada tarea del *STR* tomó valores enteros aleatorios, con una distribución uniforme entre 0% y el 50% del valor del período (T_i) de la tarea generada. Se considero un valor de *offset* igual a cero para todas las tareas de los *STR*.

Por cada *STR* la ejecución del algoritmo lleva registro del número de tareas que van conformando el *PICJ*. El método, para cada sistema, termina cuando se encuentra la primera tarea para la cual no existe un *PICJ* o se analiza el *STR* por completo.

Se encontraron resultados muy similares, al simular todos los sistemas como se puede apreciar en la Tabla 2. Consecuentemente, se optó por presentar solamente un único grafico, con todos los sistemas, en vez de múltiples gráficos discriminados por factor de utilización. Esto se debe a que la presencia del *PICJ* es independiente del factor de utilización. Además, debido a que no se encontraron subsistemas con más de 18 tareas para los que existe un *PICJ*, es posible unificar los gráficos de 20, 50 y 100 tareas.

En la Figura 5, se puede encontrar el resultado global para todos los *STR* simulados. En el eje de abscisas se encuentra el número de tareas que conforman el *PICJ* y en el eje de las ordenadas se indica el porcentaje de *STR* para los cuales un *PICJ* fue hallado.

Figura 5 – Porcentaje de STR con PICJ vs número tareas que lo conforman



De la Figura 5 y de la Tabla 2 se puede observar, que el porcentaje de subsistemas con un *PICJ*, decrece casi de forma exponencial a medida que aumenta el número de tareas.

Tabla 2 – Resultados para *STR* de 20, 50 y 100 tareas

$N° \tau$	20	20 (%)	50	50 (%)	100	100 (%)	Total	Total (%)
2	1.460.790	73.03950	1.461.219	73.06095	1.462.074	73.10370	4.384.082	73.06805
3	888.062	44.40310	887.377	44.36885	886.564	44.32820	2.662.003	44.36671
4	475.052	23.75260	475.281	23.76405	474.338	23.71690	1.424.671	23.74451
5	231.881	11.59405	231.568	11.57840	231.190	11.55950	694.639	11.57731
6	105.161	5.25805	104.806	5.24030	104.580	5.22900	314.547	5.24245
7	45.268	2.26340	44.580	2.22900	44.645	2.23225	134.493	2.24155
8	18.390	0.91950	18.248	0.91240	18.237	0.91185	54.875	0.91458
9	7.142	0.35710	7.084	0.35420	7.087	0.35435	21.313	0.35521
10	2.735	0.13675	2.682	0.13410	2.635	0.13175	8.052	0.13420
11	1.009	0.05045	969	0.04845	1.007	0.05035	2.985	0.04975
12	335	0.01675	345	0.01725	338	0.01690	1.018	0.01697
13	109	0.00545	114	0.00570	110	0.00550	333	0.00550
14	37	0.00185	46	0.00230	40	0.00200	123	0.00250
15	13	0.00065	15	0.00075	18	0.00090	46	0.00077
16	4	0.00020	6	0.00030	8	0.00040	18	0.00030
17	0	0	3	0.00015	2	0.00010	5	0.00008
18	0	0	1	0.00005	0	0	1	0.00002

Se puede apreciar fácilmente que para sistemas de 6 tareas el porcentaje es aproximadamente el 5% y para sistemas de 8 tareas, menor al 1%. Además, para un subsistema de 18 tareas sólo fue posible encontrar un único *PICJ* en 6.000.000 de sistemas.

7 CONCLUSIONES Y TRABAJOS FUTUROS

Este trabajo expone claramente, que considerar para el análisis de planificabilidad el *PICJ*, y a partir de este calcular el *peor caso de tiempo de respuesta*, es extremadamente pesimista con tiempos interarribos constantes. Más aún, si se considera que ocurrirán los *peores casos de tiempos de ejecución* y que a partir del instante *PICJ*, todas las demás instancias de las tareas de mayor prioridad no poseen *jitter*, de manera de retrasar al máximo a la tarea analizada. Sin embargo, es un requisito necesario para el análisis de planificabilidad de los *STR* del tipo duro con tiempos de inter-arribo variables a partir de un mínimo.

Por otro lado, cómo se presentó en la Sección 2, el *STR* puede ser planificable, porque nunca ocurre un *PICJ* dentro del hiperperíodo, aún cuando el análisis de la planificabilidad por los métodos presentados indique lo contrario.

Actualmente se está trabajando en una mejora del algoritmo de búsqueda, a fin de identificar el peor caso de tiempo de respuesta con *jitter* de una tarea. De esta manera se podrán evaluar los casos reales de *PICJ*, en búsqueda de disminuir el pesimismo en los tests de planificabilidad.

CRITICAL INSTANT WITH JITTER: AN UNLIKELY CONDITION

ABSTRACT: In this paper we study the probability of finding a critical instant with the worst case release jitter in a real-time system. Through simulations, is shown that this schedulability condition has an unlikely chance of occurrence, especially when the systems have a large number of tasks. The design of systems based on this condition can lead to exceedingly large hardware implementations for normal operation requirements. Also, taking into account this situation a large pessimism is introduced into schedulability analysis methods.

Keywords: Real-Time Systems. Jitter. Schedulability.

REFERENCIAS

AUDSLEY, N.C.; BURNS, A.; RICHARDSON, M.F.; TINDELL, K.; WELLINGS, A.J. Applying new scheduling theory to static priority preemptive scheduling. **Software Engineering Journal**, v. 8, p. 284, 1993.

JOSEPH, M.; PANDYA, P. Finding response times in real-time system. **The Computer Journal (British Computer Society)**, v. 29, p. 390, 1986.

LIU, C.L.; LAYLAND, J.W. Scheduling algorithms for multiprogramming in a hard real-time environment. **Journal of the ACM**, v. 20, p. 46, 1973.

OLGUÍN, G.; BISCAYART, L.; URRIZA, J.M. Generación de tareas periódicas y aperiódicas para simulación de sistemas de tiempo real. **Journal of Industrial Engineering (IJIE)**, v. 3, p. 53, 2012.

PALENCIA, J.C.; GONZALEZ HARBOUR, M. Schedulability analysis for tasks with static and dynamic offsets. Real-Time Systems Symposium, **Proceedings**…, The 19th IEEE, p. 26, 1998.

RAJKUMAR, R. Real-time synchronization protocols for shared memory multiprocessors. Distributed Computing Systems, **Proceedings**…, 10th International Conference on, p. 116, 1990.

REDELL, O.; TORNGREN, M. Calculating exact worst case response times for static priority scheduled tasks with offsets and jitter. Real-Time and Embedded Technology and Applications Symposium, **Proceedings**…, Eighth IEEE, p. 164, 2002.

RICHARD, P.; GOOSSENS, J. Approximating Response Times of Static-Priority Tasks with Release Jitters. **Proceedings**…, Euromicro Conference on Real-Time Systems. WIP Dresden, Germany, p. 4, 2006.

STANKOVIC, J.A. Misconceptions about real-time computing: a serius problem for next-generations systems. **IEEE Computer,** v. Octubre, p. 10, 1988.

TINDELL, K.W. **Fixed priority scheduling of hard real-time systems**. Doctor of Philosophy, Department of Computer Science, University of York, 1993.

URRIZA, J.M.; CAYSSIALS, R.; OROZCO, J.D. Modelado de Sistemas de Tiempo Real Planificados por RM o DM: Caracterización y Análisis. XXXIV Conferencia Latinoamericana de Informática, **Proceedings**…, CLEI, Santa Fé, Argentina, p. 1435, 2008.

Originais recebidos em: 26/10/2013

Aceito para publicação em: 15/04/2014