AgaPé-TR: una herramienta para simulación de planificación centralizada de procesos en tiempo real

Gabriel A. Wainer
Departamento de Computación
Facultad de Ciencias Exactas y Naturales.
Universidad de Buenos Aires.
Charcas 3960 7mo. 44. (1425). Capital Federal.
gabrielw@dc.uba.ar

Resumen

El problema de planificación centralizada de procesos en tiempo real es muy variado y complejo. La diversidad de soluciones existente complica el panorama de decidir qué algoritmos usar al diseñar cada sistema. En este informe se propone una herramienta (AgaPé-TR) para hacer simulación de estos problemas y facilite la clasificación y comparación de distintas soluciones. La herramienta refleja las características generales del problema, permite obtener métricas significativas, y facilita la inclusión de nuevas soluciones. Siguiendo un modelo sintético de benchmarking, AgaPé-TR permite analizar distintos algoritmos de planificación, y provee una base para probar y comparar la gran variedad de soluciones de planificación existente. A su vez provee facilidades para el análisis de planificabilidad off-line. Las experiencias realizadas nos permiten mostrar la utilidad del modelo implementado, y la facilidad para probar nuevas soluciones.

1. INTRODUCCION

En la actualidad está muy difundido el uso de computadoras para controlar eventos en el mundo real. En dichos sistemas el funcionamiento correcto no sólo depende del resultado de los cálculos, sino también del momento en que son producidos, motivo por el cual se dice que estos sistemas deben actuar en "tiempo real".

Cumplir las restricciones de tiempo de las tareas en estos sistemas suele resultar muy complejo, ya que los requerimientos a considerar son conflictivos. Por un lado, debemos considerar las restricciones de tiempo (las **metas**) de las tareas. Por otro lado, cada tarea necesita acceder a tiempo a los **recursos** necesarios. Las tareas complejas suelen subdividirse en tareas relacionadas por restricciones de **precedencia**. Las tareas deben poder tener acceso **concurrente** a los recursos. En sistemas distribuidos hay que analizar, además, restricciones de **comunicación** (incluyendo la estructura de interconexión entre las tareas y sus requerimientos de tiempo). Cuando se ejecutan simultáneamente múltiples instancias de una tarea, cada una debería tener **ubicación** en procesadores diferentes. Finalmente, de acuerdo a la funcionalidad, cumplir las metas de una tarea puede ser más crítico que otras. La **criticidad** indica este nivel de importancia [Sta93].

Es importante que el entorno de programación provea facilidades para facilitar el desarrollo de estos sistemas. Una función primordial del entorno será establecer el orden de ejecución de las tareas de tal forma que cumplan las restricciones de tiempo del sistema. La diversidad de restricciones hace que este problema (llamado planificación de tareas - scheduling -) sea muy complejo y muchas veces irresoluble. La teoría de planificación en tiempo real se relaciona con cómo cumplir a tiempo estas restricciones de las tareas del sistema, lo que implica establecer un orden (un plan) para ejecutar las tareas.

Este trabajo muestra un intento de facilitar el estudio de la variedad de soluciones de planificación existente, proponiendo el desarrollo de una herramienta de simulación de algoritmos centralizados que sea de fácil extensión y provea todas las facilidades para dicho análisis. La herramienta permitirá hacer estudios de planificabilidad off-line, y a su vez incorporar y estudiar nuevas soluciones de forma muy simple. Finalmente, se mostrará su utilidad analizando algunos algoritmos tradicionales

2. CARACTERISTICAS DEL PROBLEMA

Como fue explicado en la sección anterior, la variedad de restricciones a considerar complica la planificación de tareas en tiempo real. Para facilitar el trabajo, muchos algoritmos utilizan modelos de tareas simplificados que no consideran algunas de las restricciones, y planifican conjuntos de tareas muy acotados. Otros son mas genéricos y adaptables para distintos tipos de cargas, y tratan de encontrar un plan considerando todas las restricciones, para lo cual es necesario analizar las combinaciones de forma exhaustiva. Para acotar la explosión combinatoria se suele reducir la búsqueda por medio de heurísticas.

Existen diversos informes que resumen y clasifican la amplia variedad de algoritmos con ambas aproximaciones, entre los que se incluyen, por ejemplo,

[Che88], [Mer92], [Sta94], [Wai95]. Existen, a su vez, muchas soluciones que mejoran algunos aspectos de otras existentes, con lo cual al diseñador de esta clase de sistemas se le complica la decisión de qué algoritmos usar.

Nuestra propuesta está orientada al desarrollo de una herramienta (llamada AgaPé-TR) que permita hacer simulación de planificación de tareas en tiempo real, así como para poder comparar exhaustivamente distintos algoritmos. De esta forma se permite clasificar la variedad de soluciones existente. Por otro lado se quiere proveer facilidades para hacer análisis de garantía off-line, lo cual facilita el estudio de planificabilidad a los desarrolladores de estos sistemas. Finalmente, se quiere obtener una herramienta genérica de estudio que permita experimentar rápidamente con soluciones nuevas. Este último objetivo es prioritario, ya que de tal forma pueden proponerse y probarse nuevos algoritmos, y el diseñador puede decidir usar una solución conocida, o sus propias mejoras. Esta aproximación es de gran utilidad antes de encarar la implementación de una solución nueva, sobre todo en los casos en los que demostraciones formales de las ventajas del planificador son muy complejas o imposibles de realizar.

Para comparar algoritmos debemos decidir qué métricas utilizar para medir el mérito de cada uno de ellos. Para ello consideremos qué objetivos debe satisfacer un planificador:

- Respuestas **predecibles** ante eventos externos. Una medida de la predictibilidad es la **relación de planificabilidad**, (o porcentaje de éxito) es decir, el número total de tareas garantizadas versus el número de tareas que llegan al sistema.
- Alto grado de **planificabilidad**, (es decir, de utilización de recursos) cumpliendo las restricciones de tiempo de las tareas. Se trata de ejecutar la mayor cantidad posible de tareas con tiempo de respuesta predecible. Una medida relacionada con la planificabilidad es el **factor de carga** del procesador (time-loading), que mide el porcentaje de procesamiento útil que se está haciendo.
- **Estabilidad** ante sobrecargas transitorias. Cuando el sistema está sobrecargado y es imposible cumplir las metas de todas las tareas, aún debemos garantizar el cumplimiento de algunas tareas críticas elegidas. Una medida de la estabilidad de un sistema es la relación de planificabilidad **ponderada** por la criticidad de cada tarea.

Un planificador con estas características reduce la complejidad del desarrollo, aumenta la mantenibilidad y extensibilidad, y por ende, mejora la seguridad del sistema desarrollado (que es menos sensible a fallas de tiempo). En base a esta discusión, a continuación explicaremos algunas características generales de AgaPé-TR.

3. DESCRIPCION DE LA HERRAMIENTA

Un gran número de modelos de planificación existentes consideran la existencia de dos tipos distintos de tareas: las **periódicas** y las **aperiódicas (esporádicas)**. Las tareas periódicas tienen una serie continua de invocaciones regulares (el período de la tarea), y un tiempo máximo de cálculo. Su meta se define como el comienzo del siguiente período. Suelen ser muy comu-

nes porque en general los sistemas de tiempo real tienen alguna forma de muestreo regular. Las aperiódicas tienen una única instancia de activación con un tiempo de comienzo y una meta asociados. También es necesario considerar su peor tiempo de cálculo. Suelen usarse para eventos con tiempo de llegada aleatorio, en especial rutinas de emergencia.

Por ende, como primera medida, se trató que el sistema permitiera simular la ejecución de tareas aperiódicas y periódicas. La entrada al simulador es un conjunto básico de datos sobre las tareas: si son periódicas o aperiódicas, sus períodos (o metas para las aperiódicas), su peor tiempo de ejecución, y su hora de comienzo. También se permite ingresar la criticidad de una tarea.

En base a estos datos se estudia si el conjunto puede ejecutarse en forma predecible, o si existe un estado de sobrecarga. También se calcula el factor de carga para el conjunto de tareas inicial. Estos cálculos se rehacen cada vez que comienza una nueva tarea (periódica o aperiódica) o cuando termina una tarea (esporádica), para permitir análisis on-line de las condiciones de ejecución.

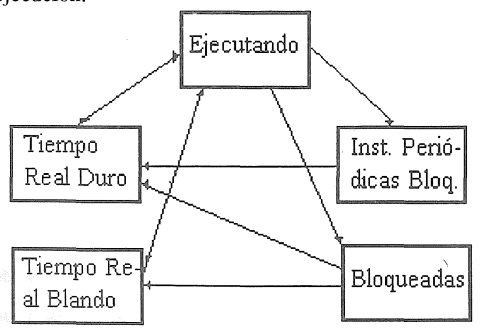


Figura 1. Modelo de tareas simulado.

A continuación se simula la ejecución de las tareas del conjunto siguiendo el modelo de procesos secuenciales, usando la información de cada tarea. Se considera que al finalizar una instancia de ejecución las tareas periódicas se demoran hasta el comienzo de su próximo período en una cola especial manejada por la interrupción del reloj.

Hasta éste momento se han realizado simulaciones de algunos algoritmos de planificación muy difundidos:

- Tasa Monotónica (TM - Rate Monotonic [Liu73], [Leh89]), que está planifica tareas periódicas independientes. Ordena las tareas de forma monótonamente creciente de acuerdo a su tasa de activación (prioridades fijas con remoción).

- Meta Anterior Primero (MAP - Earliest Deadline First [Liu73]), que ordena las tareas por prioridades variables con remoción, dando mayor prioridad a las que tienen meta anterior.

- Menor Flexibilidad Primero (MFP - Least Laxity First [Mok83]). En este algoritmo no se considera sólo la meta de la tarea, sino el tiempo restante de ejecución para cumplir la meta, y se ordenan de forma tal que las tareas con menor flexibilidad ejecuten primero (donde la flexibilidad es el tiempo restante hasta la meta menos el tiempo restante a ejecutar del peor caso).

Por medio de una interfaz gráfica [CDY95], se muestran los tiempos de comienzo y fin de cada instancia, así como remociones y pérdidas de metas. La simulación se ejecuta hasta el mínimo común múltiplo de los

períodos de las tareas periódicas (a menos que se especifique lo contrario), ya que se ha demostrado que a partir de ese instante el plan se repite [Leu80]. Esta primera opción nos permite estudiar la traza de ejecución de un conjunto individual de tareas, y se puede usar para hacer análisis off-line del conjunto de tareas a ejecutar. Una segunda opción analiza los resultados de la traza de ejecución, y calcula distintas métricas, almacenando sus resultados para hacer estudios comparativos.

El objetivo principal es analizar la predictibilidad del sistema, por lo cual estudiamos la tasa de éxito relativa (la relación entre instancias de tareas terminadas a tiempo y número total de instancias). La planificabilidad se estudia analizando el factor de carga. La estabilidad se analiza estudiando la tasa de éxito relativa, ponderada por la criticidad de la tarea. En cada caso se almacenan métricas para todo el conjunto de tareas, y además se discriminan los resultados entre las periódicas y aperiódicas. Se almacena la cantidad de remociones y cambios de contexto (para estudiar el overhead impuesto por el planificador del sistema operativo), y el tiempo ocioso del procesador. Estas mediciones se relacionan, por un lado, con la planificabilidad del sistema, pero por otro, con la posibilidad de ejecutar rutinas alternativas en el caso de desear tolerancia a fallas.

Finalmente, como a cada tarea se le puede asociar una criticidad (un valor real entre 0 y 1, creciente de acuerdo a la criticidad), se puede estudiar la estabilidad del sistema (así como el cumplimiento de las metas de las tareas más críticas).

Llamemos xi a la criticidad de la tarea i, y Gp al porcentaje de éxito relativo ponderado por la criticidad. Sea $\mathbb{P}$ ={instancias que pierden su meta}, y $\mathbb{M}$={total de instancias}. Entonces $Gp = \frac{\sum_{i \in \mathcal{P}} xi}{\sum_{j \in \mathcal{M}} xj}$.

Como $\forall i \in [1,n], xi \in \Re$, $xi \in [0,1]$ y $P \subseteq M \therefore Gp \in \Re$ y $Gp \in [0,1]$. Esto nos permite disponer de una medida que muestra la bondad del algoritmo para atender tareas críticas, expresada como un porcentaje de éxito ponderado. Si $Gp=1$, entonces se están ejecutando todas las tareas a tiempo. Cuanto más se aproxime Gp a 1, mayor la estabilidad del sistema.

La última opción provista por la herramienta es primordial, ya que permite el estudio comparativo de los distintos planificadores implementados. Para ello se generan cargas siguiendo el modelo de pruebas Hartstone [Wei89]. Este modelo provee algunas políticas de chequeo para software de base en tiempo real. Se trata de hacer pruebas de complejidad creciente, tomando requerimientos básicos y estableciendo estrategias para modificarlos. La carga debe ser representativa de sistemas de tiempo real, y proveer cifras de mérito relativas.

Se han realizado algunas modificaciones al modelo propuesto para correr sistemas con alto grado de sobrecarga, ya que la propuesta original no permite estudiar casos de sistemas sobrecargados. Hasta este momento se permite la generación de los siguientes tipos de cargas:

- PA (periódicas armónicas): se ejecutan conjuntos de tareas periódicas con períodos armónicos. Se parte de un sistema con bajo grado de carga, y se

sobrecarga el sistema aumentando el peor caso de tiempo de ejecución de las tareas. Esta prueba permite aumentar la carga total del sistema y chequear la influencia de las sobrecargas sin incrementar el overhead. Una segunda forma de generar sobrecargas es disminuyendo los períodos de las tareas (manteniéndolas armónicas). Esta prueba sirve para chequear la habilidad del sistema para manejar una carga de trabajo incremental. Otra prueba toma una única tarea del conjunto y le incrementa la frecuencia, con lo que aumenta la cantidad de cambios de contexto, y se chequea la habilidad del planificador para manejar granularidad fina de tiempo (si los cambios de contexto toman mucho tiempo, pueden provocar pérdidas de metas). Finalmente, se toma el conjunto original de tareas, y se van agregando nuevas. Esta prueba permite chequear la habilidad del sistema para manejar un gran número de tareas.

- PN (periódicas no armónicas): se repiten las pruebas PA, pero en este caso se ejecutan conjuntos de tareas con períodos no armónicos. Se realizan estas dos pruebas distintas ya que se ha mostrado que los mejores casos de utilización del procesador ocurren en el caso de que las tareas periódicas tengan períodos armónicos.

- AA (aperiódicas armónicas): se toma un conjunto de tareas periódicas armónicas junto con tareas aperiódicas. Las aperiódicas se generan con tiempo de comienzo y meta dentro del tiempo máximo de simulación, para estudiar su influencia en la carga. En un primer caso se sobrecarga el sistema disminuyendo la frecuencia de las periódicas. Esta prueba permitè chequear granularidad fina del tiempo, y estudiar el desempeño de tareas aperiódicas con metas relajadas. En la siguiente prueba se generan sobrecargas agregando nuevas tareas periódicas al conjunto, lo que permite estudiar el comportamiento del sistema con una gran cantidad de tareas y su influencia para ejecutar las tareas esporádicas. La siguiente prueba agrega nuevas tareas aperiódicas para estudiar la influencia de tener un sistema con un conjunto grande de tareas aperiódicas (simulando, por ejemplo, condiciones de emergencia en las que se desencadena la ejecución de un gran número de tareas). A continuación se estudia la influencia de tener aperiódicas con metas breves disminuyendo las metas de las aperiódicas. Finalmente se toma el conjunto original de tareas, y se incrementa el peor caso de ejecución de las aperiódicas, pudiendo chequear la influencia de sobrecargas en las aperiódicas con bajo overhead.

- AN (aperiódicas no armónicas): se repiten las pruebas AA reemplazando las tareas periódicas con otras cuyos períodos no sean armónicos.

Tomando una muestra significativa de cargas se puede estudiar el comportamiento de los distintos algoritmos con respecto a cada una de las métricas consideradas previamente. En todos los casos podemos estudiar tanto el comportamiento de las tareas periódicas como el de las esporádicas y analizar cargas variadas. Las corridas se clasifican de acuerdo a su grado de carga del procesador, comenzando por los sistemas poco cargados. Se estudian los máximos y mínimos, así como la media y el desvío estándar de cada métrica,

disponiendo de diversidad de información para analizar los algoritmos simulados. En las figuras 2 y 3 del apéndice puede verse un diseño general del simulador.

4. ANALISIS DE RESULTADOS

La diversidad de información provista hace difícil el análisis de los resultados, por ende, una segunda función de la interfaz gráfica (ver Figura 4 en el Apéndice) permite seleccionar con facilidad la información a estudiar, permitiendo acotar la visualización de la información y su estudio por medio de gráficos o texto.

AgaPé-TR se ha usado para probar los algoritmos mencionados en la sección 3 (y se está usando para probar nuevas soluciones), y nos ha permitido verificar algunos resultados conocidos (las pruebas que nos permitieron sacar estas conclusiones se pueden ver en el Apéndice II).

- **Tareas periódicas armónicas:**

* Producen un 100% de time-loading cuando se cumplan las cotas.

* El algoritmo TM es el más estable, ya que ejecuta un conjunto de tareas a tiempo (las tareas con períodos más cortos).

* El algoritmo MFP se comporta erráticamente ante sobrecargas. El algoritmo MAP también pero se obtiene mayor garantía que con MFP ante sobrecargas. Esto ocurre porque MFP es un algoritmo altamente dinámico, por ende, en muchos casos pone a ejecutar una tarea con baja criticidad, que tiene menor flexibilidad, y termina tomando tiempo de otras que sí cumplirían sus metas. En cambio, MAP determina el orden de ejecución al comienzo de una instancia (cuyos períodos son múltiplos). Por ende, ante sobrecarga llega a ejecutar más instancias a tiempo.

- **Tareas Periódicas no armónicas:**

* El time-loading no llega al 100% para el algoritmo de Tasa Monotónica.

* Nuevamente, el comportamiento es inestable ante sobrecargas para MFP y MAP. Pero aquí es mejor MFP, ya que no ocurre el fenómeno analizado previamente, ya que las tareas no son armónicas. En muchos casos, MFP considera las tareas menos flexibles, y llega a ejecutarlas a tiempo.

- **Tareas Aperiódicas:**

* Nuevamente, ante sobrecargas, TM es el mejor.

* Esto no es verdad para los casos de carga normal MAP y MFP son los mejores, ya que manejan mejor las cargas esporádicas.

* Al analizar por aislado las tareas aperiódicas sin sobrecarga, MFP se comporta mucho mejor que los otros ya que es el más dinámico para atender estas tareas. TM es el peor para ejecutar esa clase de tareas.

* Ante sobrecargas, el comportamiento depende del tipo de carga. En el caso del benchmark AN4, vemos que MFP es mejor que MAP. Este benchmark va disminuyendo las metas de las tareas aperiódicas, manteniendo el resto de los datos igual. MFP está mejor preparado para manejar este tipo de situaciones. En cambio, en el caso de AA1, donde disminuye la frecuencia de las tareas periódicas, ocurrirá que dichas tareas tendrán menor flexibilidad, calculada dinámicamente. En cambio, como MAP hace los cálculos al comienzo de las instancias, permite hacer un único cálculo de meta, y pierde menos metas de aperiódicas.

* La influencia de estas metas aperiódicas perdidas se ve claramente en el cálculo de la tasa de éxito de todo el conjunto. Nuevamente, en este caso, TM es el más estable (debido a la influencia de las periódicas), pero las aperiódicas influyen en el comportamiento general de la tasa de éxito de MFP y MAP, que siguen un comportamiento similar al recién descripto para las aperiódicas.

- **Overhead:**

* MFP es el algoritmo con mayor overhead. TM y MAP tienen aproximadamente el mismo overhead. En el gráfico de remociones PA1 puede verse un comportamiento interesante. Mientras no hay sobrecarga, la cantidad de remociones de MAP y TM crecen linealmente y con pendiente suave, mientras que MFP lo hace de forma más acelerada. Cuando hay sobrecarga, cae el número de remociones porque la prueba PA1 incrementa el peor caso de ejecución. Esto pro-voca que con el incremento de la sobrecarga haya cada vez más tareas en el conjunto que ni siquiera llegan a ejecutar.
* Esto no ocurre con los cambios de contexto, en el caso de la prueba AN4, debido a que la prueba disminuye las metas de las aperiódicas. Estas ganan en prioridad, y al ser de única instancia aumenta el número de cambios de contexto. No ocurre lo mismo con el número de remociones, que sigue teniendo el mismo comportamiento.

Como vemos, la herramienta nos permite estudiar con mediana simplicidad el comportamiento de los algoritmos implementados. Una gran ventaja de la aproximación usada es la facilidad para la incorporación de nuevos algoritmos de planificación. Se han incorporado algunas soluciones complejas (en relación con las presentadas en este trabajo [Wai96]) en aproximadamente 2 horas hombre. Por otro lado hemos podido detectar que el modelo Hartstone es incompleto para chequear la bondad de algunas soluciones existentes, y estamos estudiando nuevas pruebas. El diseño de la herramienta hace que estas nuevas pruebas (y otras futuras) puedan agregarse con gran facilidad.

5. TRABAJO ACTUAL Y CONCLUSIONES

En este trabajo hemos mostrado las características de una herramienta usada para analizar planificación de procesos en tiempo real. Para ello hemos estudiado las características generales del problema, considerando qué métricas son útiles de analizar, y construído un entorno que permite obtener variedad de resultados siguiendo un modelo consistente con el problema.

Agapé-TR ha mostrado su utilidad para el análisis experimental. La flexibilidad de incorporación de nuevas soluciones permite probarlas con sim-plicidad. En particular, un diseñador interesado en utilizar soluciones que traten de mejorar aspectos de algún algoritmo conocido puede implementarlas en la herramienta y comparar sólo los aspectos que se deseen mejorar, pudiendo, a su vez, analizar efectos colaterales (por ejemplo, incrementos en el overhead o disminución de estabilidad).

AgaPé-TR también sirve para hacer estudio de planificabilidad off-line de trazas de ejecución particulares, para que los interesados puedan estudiar el comportamiento temporal de un conjunto de tareas dado. Ha sido usada con éxito para analizar el comportamiento

de algunos algoritmos tradicionales, y en este momento estamos comparándolos con nuevas soluciones.

En este momento estamos incorporando nuevas pruebas para analizar tareas con restricciones de precedencia, y encarando nuevas modificaciones para estudiar problemas de planificación tolerantes a falla, y analizando las modificaciones necesarias al modelo Hartstone de forma tal de poder probar estos algoritmos y los mencionados previamente.

Un problema importante reside en el tiempo demorado por las simulaciones para obtener una muestra significativa. Para solucionar tales problemas estamos encarando soluciones de simulación paralela y distribuida [Fuj90], aunque la independencia de las funciones del problema implica que se puede obtener aceleración en el procesamiento simplemente corriendo las simulaciones en procesadores aislados.

7. AGRADECIMIENTOS

A los jurados por sus sugerencias, al Lic. Roberto J.G. Bevilacqua, y al grupo de alumnos encargado de implementar la interfaz en el Laboratorio de Sistemas de Tiempo Real [CDY95].

6. REFERENCIAS

[CDY95] CHRISTEN, G.; DOBNIEWSKI, A.; YANKILEVICH, P.; WAINER, G. "Implementación de una interfaz gráfica para AgaPé-TR". *Informe inter-no del Laboratorio de Tiempo Real. FCEN-UBA*. 1995.

[Che88] CHENG, S; STANKOVIC, J.; RAMAMRITHAM, K. "Scheduling Algorithms for Real-time systems: a brief survey". *En "Real-Time Systems", IEEE Press*, 1993. pp. 150-173.

[Fuj90] FUJIMOTO, R. "Parallel Discrete-Event simulation". *Communications of the ACM*. Octubre 1990.

[Leh89] LEHOCZKY, J.P.; SHA, L.; DING, Y. "The Rate Monotonic Scheduling algorithm - exact characterization and average case behavior". *Proceedings IEEE Real-Time Systems Symposium. CS Press, Los Alamitos, Calif.* pp. 166-171. 1986.

[Leu80] LEUNG, J.; MERRIL, M. "A note on preemptive scheduling of periodic real-time tasks". *Information Processing Letters*, 11(3), pp. 115-118. 1980.

[Liu73] LIU, C.; LAYLAND, J. "Scheduling algorithms for multiprogramming in a Hard Real Time System Environment". *Journal of the ACM*, Vol. 20, No. 1, 1973, pp 46-61.

[Mer92] MERCER, C. "An introduction to real-time operating systems: scheduling theory". *Technical Report. Carnegie Mellon University*. 1992.

[Mok83] MOK, A. "Fundamental design problems of distributed systems for the hard-real-time environment". *Ph.D thesis. MIT. Cambridge, Mass.* May 1983.

[Sta93] STANKOVIC, J., RAMAMRITHAM, K. "Hard Real-Time Systems". *IEEE Press*. 1993.

[Sta94] STANKOVIC, J. et al. "Implications of classical scheduling results for Real-Time systems". *CMPSCI Technical Report 93-23. University of Massachussets at Amherst*. January 1994.

[Wai95a] WAINER, G. "Algunos resultados de planificación centralizada en tiempo real". *Anales de las 24 Jornadas Argentinas de Informática e Investigación Operativa*. Agosto de 1995.

[Wai96] WAINER, G. "Two simple strategies to improve the performance of real-time scheduling algorithms". *Informe interno FCEN-UBA. Enviado a ISIC'96*. 1996.

[Wei89] WEIDERMAN, N. "Hartstone: synthetic benchmark requirements for Hard Real-Time applications". Technical Report CMU/SEI-89-TR-23. Carnegie Mellon University. Software Engineering Institute.

APENDICE I

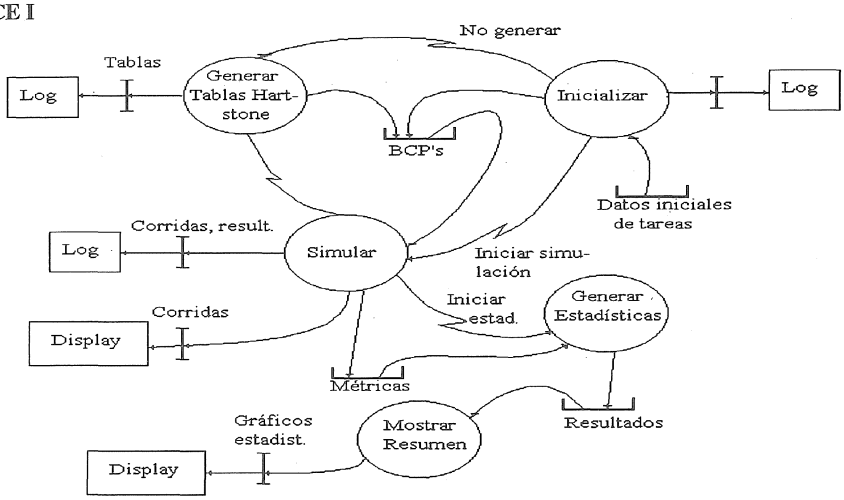


Figura 2. Diseño general del sistema usando MASCOT.

Lugares	Transiciones
P1: Replanificar	T1: Simular ejecución instancia
P2: Listo para simular	T2: Iniciar cálculos
P3: Calcular métricas iniciales	T3: Fin cálculos
P4: Simular ejecución	T4: Comienzo inicialización
P5: Calcular metas perdidas	T5: Fin simulación
P6: Calcular uso de recursos	T6: Fin emisión resultados
P7: Verificar planificabilidad	T7: Fin simulación ejecución
P8: Mostrar resultados de ejecución	T8: Comienzo simulación instancia
P9: Emitir resultados	
P10: Guardar resultados	

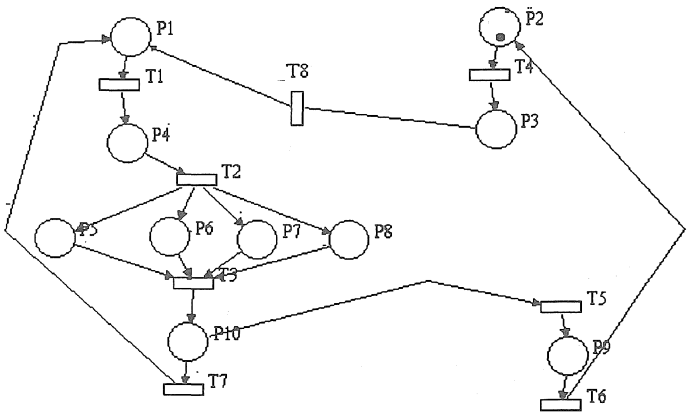


Figura 3. Especificación de rutinas de simulación (burbuja "simular"de figura 2) usando Redes de Petri.

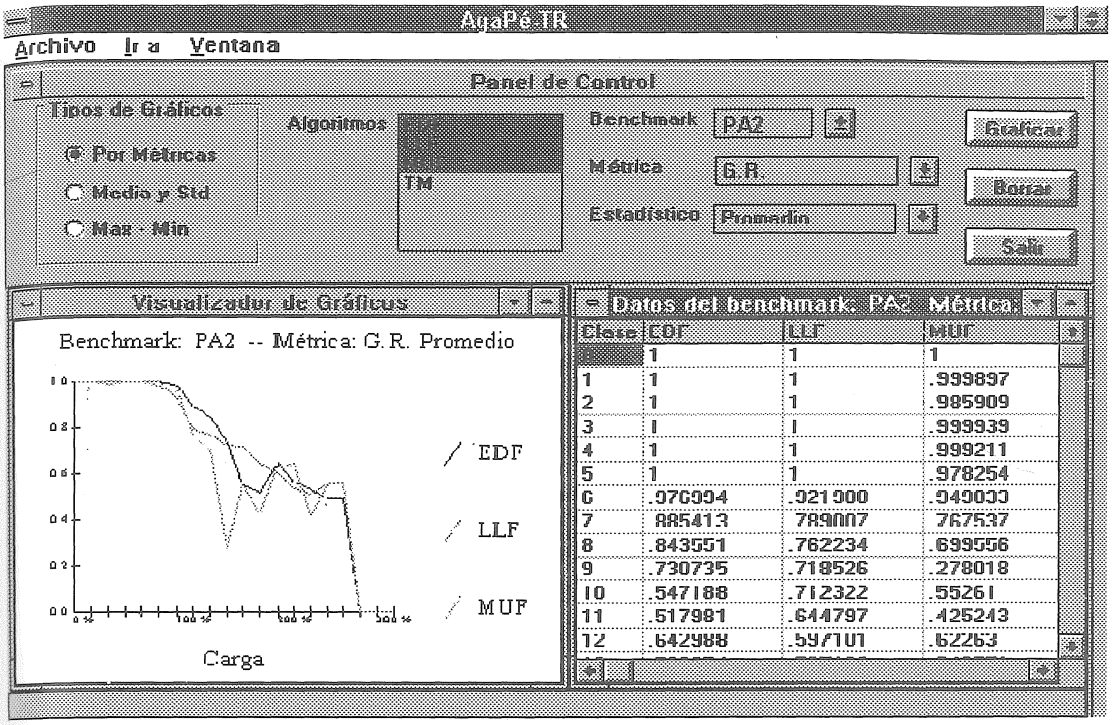
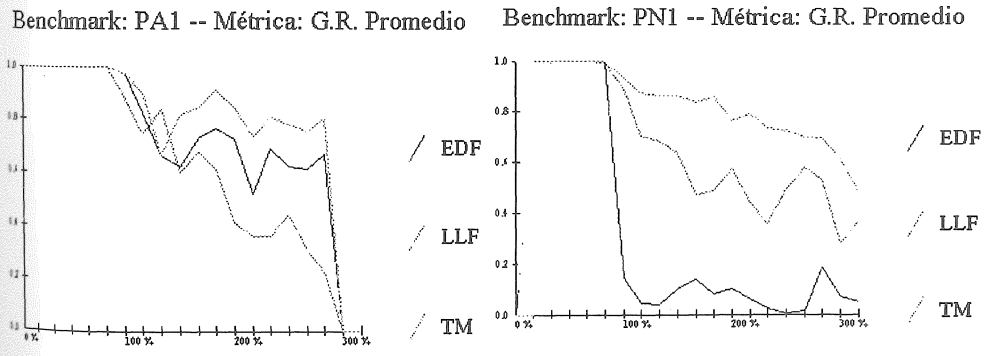


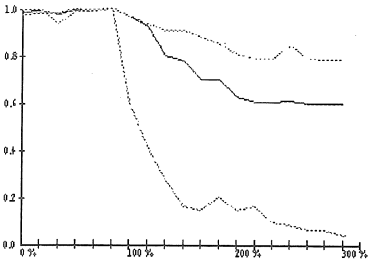
Figura 4. Interfaz gráfica

APENDICE II - RESULTADOS

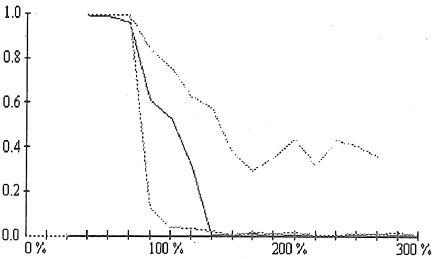
En las siguientes figuras se muestran distintos resultados obtenidos en diversos benchmarks. Se muestran diversas métricas tomadas. El eje X marca el porcentaje de carga del sistema testeado.



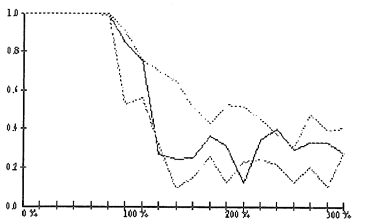
Benchmark: PA4 -- Métrica: G.R. Mínimo



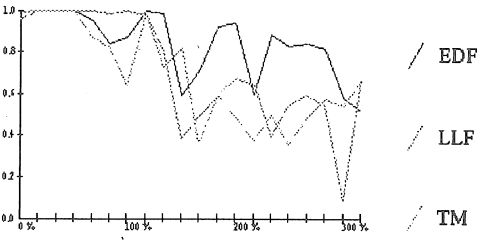
Benchmark: PN3 -- Métrica: G.R. Mínimo



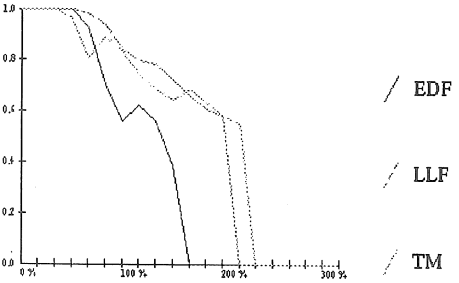
Benchmark: AA1 -- Métrica: G.R. Pond Promedio



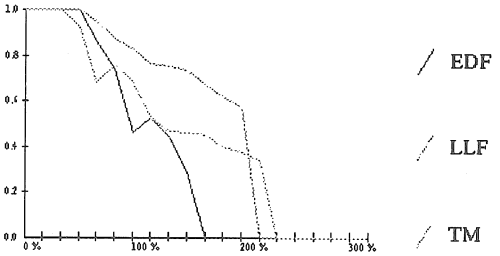
Benchmark: AA1 -- Métrica: G.R. Aper.Pond Promedio



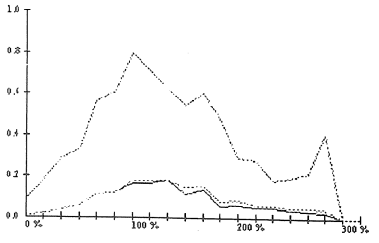
Benchmark: AN4 -- Métrica: G.R. Promedio



Benchmark: AN4 -- Métrica: G.R. Aper Promedio



Benchmark: PA1 -- Métrica: Remociones Promedio



Benchmark: AN4 -- Métrica: C.Contexto Máximo