UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PROPUESTA DE UN MÉTODO DE PLANIFICACIÓN PARA LA RECONFIGURACIÓN EN LÍNEA EN UN SISTEMA DE TIEMPO REAL

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A :

ANGEL GARCÍA ZAVALA

DIRECTORES DE LA TESIS: DR. HÉCTOR BENÍTEZ PÉREZ
DR. MANUEL ROMERO SALCEDO

MÉXICO, D.F. 2003

A mi esposa Irma porque sin su apoyo, ayuda y comprensión no hubiera podido realizar este trabajo y a mi hijo Gustavo por los momentos de felicidad que me ha dado en todo momento.

A mis padres Alfredo y Glafira y a mis hermanos Joel, Verónica y Cecilia por el gran apoyo brindado a mí y a mi familia en todo momento.

A mis familiares: mi abuelita Soledad, mis tías Patricia, Aurora, Cecilia y Olivia, mis primos Paty, Dany, Manolo, Lili, Paco, Poncho y Rosita y a mis tíos Rubén, Gabriel, Adolfo, Ubertino, Juan Manuel por su gran apoyo y confianza.

Agradecimientos:

A mis tutores, los doctores Héctor Benítez Pérez y Manuel Romero Salcedo, mi sincero agradecimiento y respeto por llevar a cabo la dirección de este trabajo, por su apoyo y tiempo dedicado.

A los sinodales Dra. Cristina Verde Rodarte, Dr. Fabián García Nocetti y Dr. Yu Tang Xu por la revisión, sugerencias y correcciones realizadas a la presente tesis.

A Jorge L. Ortega Arjona un agradecimiento especial por el apoyo brindado, enseñanzas y sugerencias realizadas al presente trabajo.

Se agradece el apoyo económico recibido al proyecto del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT No. 106 100).

Se agradece el apoyo económico recibido al Consejo Nacional de Ciencia y Tecnología (CONACyT).

Se agradece también al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas y a la propia Universidad Nacional Autónoma de México que me dieron la oportunidad de realizar los estudios de maestría.

Índice General

Capítulo 1

Introducción

Esta tesis propone un método de reconfiguración en línea para un sistema de cómputo de tiempo real con base a la planificación de tareas. En esta tesis se revisan los temas relacionados con: sistemas de tiempo real, sistemas de cómputo de tiempo real, reconfiguración en línea, planificación y sistemas distribuidos.

"Un sistema de cómputo de tiempo real es aquél en el cual su correcto funcionamiento no depende sólo de los resultados de las operaciones, sino también del instante en el cual tales resultados son producidos" [Kopetz(1997), Krishna y Shin(1997)]. En general, en los sistemas de cómputo de tiempo real sus acciones deben ocurrir en tiempos específicos, con el fin de garantizar la calidad, el determinismo y la consistencia de los datos. Debido a lo anterior, normalmente la variable de interés de un sistema de cómputo de tiempo real es su *respuesta* en un instante de tiempo preestablecido. Un sistema de cómputo de tiempo real se dice que es distribuido si los componentes que lo conforman se ejecutan en computadoras separadas comunicadas a través de un sistema de comunicación de tiempo real.

Así, un sistema de cómputo de tiempo real debe generar cálculos para un intervalo de tiempo determinado por los tiempos máximos (conocidos como *plazos límites*) a través de un programa llamado o, *planificador*. El planificador se encarga de ordenar las diferentes *tareas* con respecto a sus plazos límites. Se entiende por *tarea* a cada acción que realiza cada uno de los elementos del sistema de cómputo de tiempo real.

Particularmente, la teoría de sistemas de cómputo de tiempo real consiste en la manipulación de un conjunto de tareas, las cuales tienen asignado un tiempo de proceso y un periodo en el cual se deben de ejecutar de tal forma que se cumplan sus plazos límites. A esta manipulación se le conoce como *planificación*, que da como resultado un ordenamiento de tareas llamado *plan*.

Un algoritmo de ordenamiento (también llamado planificador) frecuentemente utilizado para el análisis de tareas es el llamado *algoritmo de ordenación por planes*. Este tipo de planificador combina características tanto de *planificadores estáticos* como de *planificadores dinámicos* (sección 2.3), y fue propuesto inicialmente por L. Almeida [Almeida *et al.*(1999)]. El método propuesto en esta tesis utiliza al algoritmo de

ordenación por planes.

Existen situaciones en las cuales los parámetros de las tareas pertenecientes a un sistema de cómputo de tiempo real deben cambiar dinámicamente en el tiempo. Ello se presenta por ejemplo cuando cambia la configuración (su modo de funcionamiento) de los componentes de un sistema de tiempo real. En ese caso es necesario cambiar el periodo de las tareas aumentando o disminuyendo su tiempo de proceso. El problema de la obtención de parámetros de tareas cuando ocurren cambios en el tiempo se conoce como un problema de *reconfiguración* de un sistema de cómputo de tiempo real. Este problema es complejo debido a que se debe de obtener una solución de planificación *"en línea"* que logre garantizar el adecuado funcionamiento de la aplicación[1] .

En esta tesis, se propone un método para encontrar parámetros de tareas adecuados a partir del algoritmo de ordenación por planes para la reconfiguración en línea de un sistema de cómputo de tiempo real distribuido. De esa forma al encontrar parámetros adecuados de las tareas de tiempo real, el método establece una solución adecuada de planificación para la reconfiguración en línea. Este método se divide en dos partes: la primera fuera de línea, se ocupa de generar una tabla de soluciones viables respecto a la planificación de tareas y a los requerimientos de rendimiento del caso de estudio; la segunda parte en línea escoge la configuración más adecuada ya planteada en la tabla de soluciones de acuerdo a los cambios propuestos en la reconfiguración.

1.1 El problema

Un área de interés de los sistemas de cómputo de tiempo real es la relacionada con la planificación. Esta área de conocimiento se ha estudiado extensivamente y se han propuesto muchas soluciones de planificación bajo diferentes restricciones en la política de ordenación Jane W. S. Liu [Liu(2000)]. Por ejemplo, se han dado soluciones en la planificación estática y dinámica, en planificación interrumpible y no interrumpible, etc (sección 2.3).

Sin embargo, una área abierta a la investigación es la planificación de sistemas de cómputo de tiempo real que sean reconfigurables en línea. Existen soluciones de planificadores dinámicos al problema, aunque como se explica en la sección 2.3, tales soluciones son computacionalmente demandantes. La complejidad de encontrar una solución adecuada se debe principalmente a las restricciones temporales en las que un sistema de cómputo debe reaccionar. Ya que éste actúa en tiempos restringidos es necesario asegurar su respuesta a tiempo con base a los plazos límites de cada una de las tareas.

El problema dentro del cual se enmarca la presente tesis es con respecto a la reconfiguración en línea de un sistema de cómputo de tiempo real tomando en cuenta los requerimientos de rendimiento de la aplicación en curso (ver al apéndice B).

[1] Como se recordará, en un sistema de cómputo de tiempo real una de las características principales son las restricciones temporales en las que éste debe actuar en el tiempo.

1.2 Objetivo

El objetivo de esta tesis es proponer un método de manera heurística que obtiene parámetros adecuados de tareas de tiempo real utilizando al algoritmo de ordenación por planes y que da como resultado un conjunto de soluciones adecuadas de planificación, en sistemas de cómputo distribuidos de tiempo real reconfigurables en línea. Este método toma en cuenta los requerimientos de la aplicación y a la planificación (tiempo real).

1.3 Contribución y relevancia

El presente trabajo propone un método con base en el análisis de la planificación de tareas que garantiza la reconfiguración en línea de sistemas de cómputo de tiempo real integrando los requerimientos de rendimiento de la aplicación y de tiempo real. El método propuesto se divide en dos partes: la primera parte fuera de línea se ocupa de generar una tabla de soluciones viables obtenidas de la planificación de tareas y de la respuesta dinámica del caso de estudio; en la segunda parte en línea se escoge la configuración más adecuada ya planteada en la tabla de soluciones de acuerdo a los cambios presentados bajo una reconfiguración en línea del sistema de cómputo.

El método se propone para trabajar con cualquier aplicación de un sistema de cómputo de tiempo real y se considera que los resultados de la planificación pueden ser utilizados para futuros estudios sobre el mismo problema de reconfiguración.

1.4 Estructura de la tesis

La tesis se forma por cinco capítulos, un glosario, y dos apéndices:

- El capítulo 2 expone los conceptos básicos sobre la teoría de los sistemas de cómputo de tiempo real y las áreas relacionadas. También se presenta una introducción teórica sobre los planificadores, su notación, clasificación, y algunos de los tipos existentes.
- El capítulo 3 introduce el método para encontrar una ordenación de tareas como una solución adecuada con base al comportamiento del sistema. El método se divide en dos partes: la primera parte, el procedimiento fuera de línea, se ocupa de generar una tabla de soluciones viables respecto de la planificación de tareas y de la respuesta de un caso de estudio; la segunda parte, el procedimiento en línea, compara el plan propuesto contra las soluciones ya planteadas en la tabla, escogiendo la configuración más adecuada de acuerdo a los cambios propuestos.
- El capítulo 4 presenta el caso de estudio para la evaluación del método. El caso de estudio se basa en una simulación del censado de tres lecturas de una

señal seno obteniendo un resultado a través de un algoritmo de votación. Se describe la implantación del programa que hace la simulación y se muestran los datos obtenidos del funcionamiento del caso de estudio bajo un esquema de reconfiguración.

- El capítulo 5 presenta las conclusiones en torno al trabajo realizado. Asimismo, se mencionan propuestas de investigaciones futuras tomando a esta tesis como base y antecedente.
- El glosario contiene las definiciones de los términos usados en este trabajo, con las correspondientes traducciones al ingles para referencia del lector.
- El apéndice A presenta una introducción sobre el sistema operativo RT-Linux, así como el conjunto de funciones básicas para la implementación de tareas de tiempo real utilizando el estándar POSIX 1003.1, basado en hebras de control.
- El apéndice B presenta el diseño temporal propuesto por M. Saksena [Saksena (1998)] para los sistemas de cómputo de tiempo real.

Capítulo 2

Antecedentes

Este capítulo tiene como objetivo dar una revisión del marco teórico en el cual se ubica al presente proyecto de tesis, presentando una serie de conceptos básicos relacionados con los sistemas de cómputo de tiempo real, así como los algoritmos necesarios para el manejo de un sistema distribuido. Así mismo, se hace una breve revisión del método de diseño a usar en este proyecto. Cabe señalar que se busca establecer la liga de una serie de conceptos que permiten definir al método propuesto en el capítulo 3.

En este capítulo se presentan inicialmente los conceptos básicos de los sistemas de cómputo de tiempo real así como su clasificación de acuerdo a factores internos y externos. En segundo lugar se presentan los planificadores junto con una clasificación que ubica al tipo de planificador empleado. En particular se hace una descripción del algoritmo de ordenación por planes dado que es parte fundamental del método propuesto. Finalmente, se presentan conceptos generales del diseño del sistema de cómputo desde el punto de vista de los aspectos temporales.

2.1 Conceptos básicos

2.1.1 Sistemas de tiempo real

Un sistema de tiempo real es aquel que de acuerdo a restricciones temporales realiza su trabajo de manera determinística [Liu(2000)]. En general, se divide en tres secciones: el operador, el objeto controlado y el sistema de cómputo de tiempo real (Fig. 2.1).

La comunicación entre el sistema de cómputo y el operador se realiza mediante una interfaz, llamada *interfaz hombre-máquina* (Human-Computer Interface), que consiste de dispositivos de entrada y salida. La comunicación entre el sistema de cómputo y el objeto controlado se realiza mediante una *interfaz instrumental*, la cual permite al sistema de cómputo conocer el estado del objeto controlado y actuar sobre él en forma adecuada, a través de dispositivos de entrada y salida.

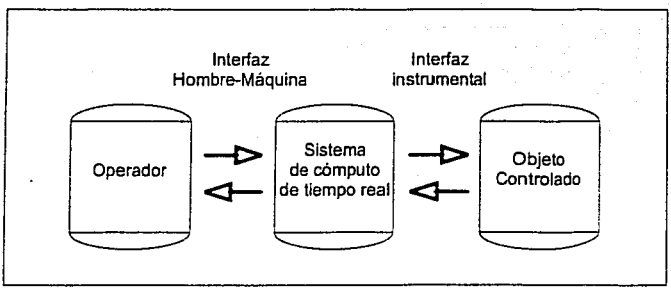


Fig. 2.1: Sistema de tiempo real

"Un sistema de cómputo de tiempo real es un sistema de cómputo en el cual el correcto funcionamiento del sistema no solo depende de los resultados lógicos de las operaciones, sino también depende del instante en el cual estos resultados son producidos" [Kopetz(1997)]. Es decir, un sistema de cómputo de tiempo real[1] presenta un correcto funcionamiento con respecto a su consistencia de datos, y no sólo depende de los resultados de las operaciones. Además, depende del instante en el cual estos resultados son producidos, por lo que las aplicaciones en donde se utilizan los sistemas de cómputo de tiempo real dependen del tiempo en el cual se genera el resultado [Stankovic(1988)]. Su importancia radica en situaciones críticas donde se requiere de determinismo, seguridad[2] y calidad de servicio. Por lo tanto, durante el resto del presente trabajo, se entenderá que un sistema de cómputo de tiempo real es aquél que maneja la interacción de sus elementos en tiempos acotados, de manera determinística, manteniendo su consistencia de datos.

2.1.2 Sistemas de cómputo distribuido de tiempo real

"Un sistema de cómputo distribuido es aquél cuyos componentes, ya sea de hardware o software, están localizados en computadoras en red que se comunican coordinando sus acciones sólo por paso de mensajes" [Colouris *et al.*(2001)]. Por lo tanto, un sistema de cómputo distribuido de tiempo real cubre tanto esta definición, como la propuesta en la sección anterior. En tal caso, su estructura de hardware se muestra como un conjunto de nodos (o computadoras) interconectados por un sistema de comunicaciones de tiempo real (Fig. 2.2). Cada nodo del sistema distribuido de tiempo real tiene dos partes lógicas para su funcionamiento: la interfaz de comunicación y

[1] En toda la tesis la frase "*en tiempo real*" se referirá al tiempo físico. Por lo tanto, todo sistema funciona *en tiempo real*. En cambio la frase "*de tiempo real*" se refiere a la teoría de los sistemas de tiempo real.

[2] Seguridad en el sentido de preservar la vida humana o preservar el ambiente.

el núcleo de la computadora. La interfaz de comunicación se encarga de transmitir la información entre los diferentes nodos. El núcleo de la computadora se encarga de realizar todos los cálculos y planes que requiera una aplicación.

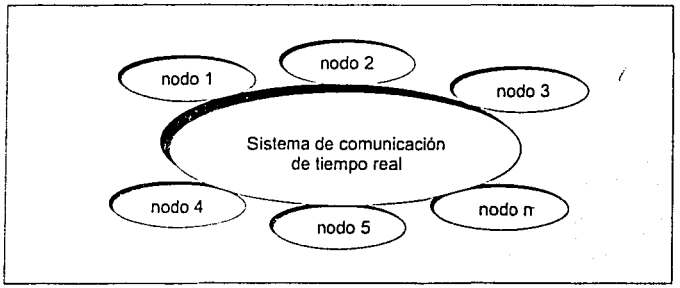


Fig. 2.2: Sistema de cómputo de tiempo real distribuido

El sistema de comunicación de tiempo real constituye la médula espinal de las aplicaciones distribuidas de tiempo real. Su diseño debe considerar las características de determinismo en la transmisión de información en los niveles inferiores del protocolo de comunicación con el fin de lograr un desempeño predecible en los niveles superiores. Si un medio de comunicación exhibe un desempeño impredecible en la transmisión, entonces será muy difícil que una capa superior del protocolo de comunicación pueda proveer comunicación de tiempo real predecible a las aplicaciones.

Un problema típico de los sistemas comerciales de cómputo distribuido son las variaciones temporales debido a comunicaciones asíncronas. Por ejemplo, si el sistema de comunicación (la red) es un canal o *bus*, es posible provocarle una saturación de peticiones de comunicación que puede producir una variación de los retardos de tiempo, afectando el rendimiento de todo el sistema distribuido. Así, con el fin de soportar una completa ejecución en un tiempo establecido de las actividades distribuidas de tiempo real, los sistemas de cómputo distribuido de tiempo real deben asegurar un *límite de retraso* en la entrega de mensajes. Por lo tanto, para tener un sistema de cómputo de tiempo real no sólo deben considerarse las actividades distribuidas dentro de los nodos, sino que también se deben de considerar las actividades de comunicación en el sistema [Livani *et al.*(1999)].

2.1.3 Clasificación de los sistemas de cómputo de tiempo real

De acuerdo con los conceptos básicos, las restricciones temporales son esenciales en los sistemas de cómputo de tiempo real. Es por esto que los sistemas de cómputo de tiempo real se relacionan con aplicaciones donde la calidad de servicio y/o la seguridad son parte crítica de los requerimientos. Así, estos se pueden clasificar bajo diferentes perspectivas dependiendo de las restricciones temporales. Una primera clasificación se hace de acuerdo a *los factores externos e internos de la aplicación.* La Fig. 2.3 muestra tal clasificación [Liu(2000)].

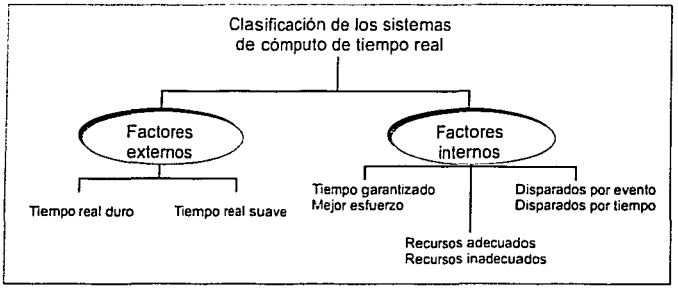


Fig. 2.3: Clasificación de los sistemas de cómputo de tiempo real

Los factores externos al sistema de cómputo incluyen:

- **Tiempo real duro *versus* tiempo real suave**

  Comúnmente, las restricciones temporales se consideran *duras* y *suaves* de acuerdo con la pérdida de los plazos límites y la utilidad de los resultados.

  Respecto a la pérdida de los plazos límites, una restricción temporal (o tiempo límite) es dura si se considera que la pérdida de un *plazo límite* puede provocar una catástrofe. Por el contrario, se dice que es suave si no resulta ningún problema al perderse algunos *plazos límites*.

  Respecto a la utilidad de los resultados, se dice que una restricción temporal (o tiempo límite) es dura si después de haber perdido un *plazo límite* los resultados no tienen ninguna utilidad. En cambio, se dice que es suave si al perderse algún *plazo límite*, los resultados todavía son útiles.

La segunda clasificación considera factores internos del diseño de la aplicación en:

- **Tiempo garantizado *versus* mejor esfuerzo**

  Un sistema tiene una respuesta en *tiempo garantizado* si se diseña tomando en cuenta todos los escenarios posibles y se asegura que responde adecuadamente aun en presencia de saturación de peticiones de uso en los dispositivos. Por otra parte, un sistema tiene una respuesta con el *mejor esfuerzo* si se diseña sin tomar en cuenta todos los escenarios posibles, pero responde adecuadamente en la mayoría de tales escenarios.

- **Recursos adecuados *versus* recursos inadecuados**

  La respuesta garantizada de los sistemas esta basada en el principio de los *recursos adecuados*, es decir, que se cuenta con los recursos disponibles suficientes para manejarse adecuadamente en escenarios de saturación y/o en escenarios de falla. Por otra parte, en el caso en que no se cuente con los recursos suficientes (por ejemplo, por falta de presupuesto, tecnología, etc), se considera que son *recursos inadecuados*.

- **Disparadas por tiempo *versus* disparadas por evento**

  El transcurso del tiempo puede ser representado por una línea recta que viene del pasado y va hacia el futuro. Cualquier suceso que ocurra en tal línea del tiempo es llamado un *evento*. Ahora bien, un *disparo* (o *trigger*) es un evento que causa el comienzo de una tarea. De esta forma, se pueden distinguir dos tipos de sistemas dependiendo de su disparo: *por eventos* y *por tiempo*.

  Un sistema se considera disparado *por evento* cuando todas las comunicaciones y el conjunto de tareas del proceso se empiezan a ejecutar siempre y cuando haya un cambio significativo en el estado del sistema. En cambio, un sistema se considera disparado *por tiempo* si todas las comunicaciones y conjunto de tareas del proceso se ejecutan en puntos de tiempo predeterminados.

Ya que el método supone que las tareas involucradas en el sistema de cómputo de tiempo real son periódicas, se establece entonces que el método es aplicable solo a los sistemas de cómputo de tiempo real disparados por tiempo.

2.1.4 Algunas puntualizaciones sobre los sistemas de cómputo de tiempo real

Para entender mejor a los sistemas de cómputo de tiempo, a continuación se presentan una serie de frecuentes malentendidos que se tienen acerca de estos sistemas [Stankovic(1988)], así se cree que:

- *No hay una teoría para el diseño de los sistemas de tiempo real.* Aunque el diseño de un sistema de tiempo real es particular en cada aplicación, eso no

quiere decir que no se pueda estudiar el problema científicamente. Actualmente, se investiga en métricas de métodos de especificación de sistemas, teorías de semántica en lenguajes de programación de tiempo real, planificación, etc.

- *Los avances logrados en supercómputo toman en cuenta al tiempo real.* En supercómputo se explotan ampliamente los procesos en paralelo que mejoran la velocidad de salida de cómputo. Sin embargo, esta mejora en velocidad no asegura que las restricciones temporales en la ejecución de procesos se cumplan en forma automática. Es por ello que en el diseño de de tales equipos se debe tomar en cuenta las posibles saturaciones de tráfico en su ejecución y comunicación.

- *El cómputo de tiempo real es equivalente al cómputo de rendimiento.* El objetivo del cómputo de rendimiento es minimizar el tiempo medio de respuesta de un conjunto de *tareas*. Sin embargo el objetivo de la computación de tiempo real es asegurar que los plazos individuales requeridos de cada *tarea* se cumplan a tiempo. Más que una respuesta rápida ésta debe ser dada a tiempo.

- *La programación de tiempo real es en código ensamblador con escritura directa a los manejadores de dispositivos.* Aunque es importante la programación y manejo de dispositivos en el proceso de automatización, los sistemas de tiempo real se enfocan en la optimización de algoritmos de programación y en la planificación de tareas que tomen en cuenta los recursos del sistema.

- *Todos los problemas de los sistemas de cómputo de tiempo real han sido resueltos en otras áreas de ciencias de la computación o en la investigación de operaciones.* Algunos de los problemas para obtener sistemas de cómputo de tiempo real ya han sido resueltos en algunas otras áreas de las ciencias de la computación o la teoría de operaciones, por ejemplo en las áreas de los sistemas distribuidos, lenguajes de programación, paralelismo, etc. Sin embargo, existen problemas únicos que no se encuentran en otras áreas y por lo tanto se requiere de una mayor investigación. Por ejemplo, hay soluciones de sistemas que calculan plazos de respuesta en forma estadística. En los sistemas de cómputo de tiempo real estas no son adecuadas, ya que no garantizan una respuesta adecuada en toda situación crítica.

2.2 Planificadores o algoritmos de ordenamiento

El manejo de las restricciones de tiempo real se plantea a partir del manejo en tiempo de las tareas involucradas. Para llevar a cabo dicho manejo, se requiere plantear un algoritmo de ordenamiento o *planificador*, por lo cual se revisan los conceptos básicos inherentes al tema de planificadores.

Un *proceso* es un conjunto de *trabajos* o *tareas*. La unidad básica de un *trabajo* es una *operación*. Una *operación* es una acción elemental que tiene que ser realizada. Las *operaciones* tienen tres atributos principales [Conway *et al.*(1967)]:

1. La representación de la operación de un trabajo particular.
2. La representación de la operación dentro de una máquina en particular.
3. Un número real representando el tiempo de proceso de una operación.

De acuerdo al primer atributo, el conjunto de *operaciones* puede ser dividido en subconjuntos disjuntos, exhaustivos, o mutuamente exclusivos, considerados como *tareas* o *trabajos*. De la misma forma, de acuerdo al segundo atributo, el conjunto de *operaciones* puede ser dividido en subconjuntos exhaustivos y mutuamente exclusivos, asociados a una máquina en particular. De hecho, de manera abstracta una *máquina* es solamente una escala de tiempo con ciertos intervalos disponibles para realizar *tareas*.

A partir de las definiciones anteriores, *planificar* un proceso de trabajos se refiere al hecho de asignar a cada operación una posición específica en la escala de tiempo de una máquina específica. Así, el problema de planificar involucra:

1. Los trabajos y las operaciones a ser procesadas.
2. El número y tipos de las máquinas que se utilizarán en el trabajo.
3. Las políticas que restringen la manera en que las asignaciones pueden ser hechas.
4. El criterio por el cual un plan deberá ser evaluado.

El problema de planificación, que conciernen a la búsqueda de un ordenamiento adecuado sujeto a un número limitado de restricciones, presenta en general una gran complejidad. Planificar n *tareas* en m máquinas tiene un número infinito de soluciones factibles, ya que los tiempos de ocio entre las operaciones pueden variar. A. Mok ha demostrado que el problema de decidir si un conjunto de *tareas* es planificable[3] es *np-completo*, cuando los periodos usan semáforos para forzar la mutua exclusión [Mok(1983)]. La distribución de procesos en el procesador de una computadora es un claro ejemplo de ello. Así, en el campo de la planificación se han propuesto muchas soluciones, desde las derivadas de la investigación de operaciones hasta el uso de la planificación borrosa (*fuzzy*) [Kerr y Slany(1994)].

En conclusión, para los propósitos de esta tesis, el planificador es aquel algoritmo que permitirá definir el tiempo de ejecución de las tareas en el tipo de sistema a estudiar. Por esto último, el método propuesto en esta tesis se desarrolla con base a un planificador específico, llamado *algoritmo de ordenamiento por planes* [Almeida *et al.*(1999)].

[3]Un algoritmo de planificación, u ordenación es un conjunto de reglas que determinan que *tarea* debe ser ejecutada en un momento particular. Un conjunto de *tareas* es *planificable* bajo un algoritmo de ordenación, si al utilizar el algoritmo sobre el conjunto de *tareas* ninguna de ellas pierde sus *plazos límites*.

2.3 Clasificación de planificadores

Los planificadores, como algoritmos, se clasifican de acuerdo al tratamiento que se hace de las tareas que componen los procesos. Muchas veces, los parámetros de algunas tareas se conocen *a priori*, y generalmente se repiten en un plazo fijo (si no es así, muchas veces se les puede asignar uno). Tales tareas son las llamadas *tareas periódicas*, y en general se encargan del funcionamiento principal de un sistema de cómputo de tiempo real.

Por otro lado, se dice que una tarea es *crítica* si el retraso en la obtención del resultado puede producir una situación de peligro o catástrofe. Más aun, H. Kopetz establece que *"en la teoría de los sistemas de cómputo de tiempo real, todas las tareas críticas son periódicas"* [Kopetz(1997)]. Los sistemas de cómputo de tiempo real críticos son aquellos con al menos una *tarea crítica*. La forma en la que se planifican las tareas debe garantizar la correcta ejecución de todas las *tareas críticas* durante todo el tiempo de ejecución del sistema. El problema de garantizar que las tareas se ejecuten en el orden previsto hace necesario un análisis *a priori* para saber si el sistema funcionará adecuadamente. Este tipo de análisis se llama *comprobación o prueba de planificabilidad (test)*. Así, la *comprobación de planificabilidad* es una condición suficiente para que las *tareas críticas* puedan cumplir con sus plazos límites.

Existen tres enfoques generales en la planificación de tiempo real: manejo por reloj, ronda-robin pesada y manejo por prioridades [Liu(2000)]. Debido a que el algoritmo a considerar en esta tesis está basado en prioridades, a continuación sólo se revisa el manejo por prioridades.

En los planificadores que tienen un enfoque manejado por prioridades son varias las políticas de planificación que pueden emplearse al ordenar los *trabajos* o *tareas* [Stankovic *et al.*(1995)]. Sin embargo, existen dos paradigmas bajo los cuales se agrupa a la planificación: *planificación dura y suave* [Kopetz(1997)]. En la planificación dura todas las tareas son críticas. En la planificación suave pueden haber tareas no críticas. En la Fig. 2.4 se presentan las variaciones de la planificación dura.

A partir de esta figura, la planificación dura se divide en estática y dinámica:

- Un planificador es llamado *dinámico* (o en línea) si hace decisiones de planificación en tiempo de ejecución, seleccionando las tareas a ejecutarse. Los planificadores dinámicos son flexibles y adaptables a diferentes escenarios, aunque requieren de una gran demanda de operaciones en el procesador.
- Un planificador se llama *estático* si toma las decisiones de planificación en tiempo de compilación. Produce una tabla o plan fuera de línea que posteriormente, en tiempo de ejecución, debe distribuir las tareas entre los diferentes nodos. Para esto, es necesario tener toda la información antes de ejecutar el plan, es decir, características de las tareas, como podrían ser los tiempos máximos de ejecución, las restricciones de precedencia, los *plazos límite*, etc. Puesto que el plan se calcula fuera de línea, en general su costo en tiempo de ejecución es pequeño.

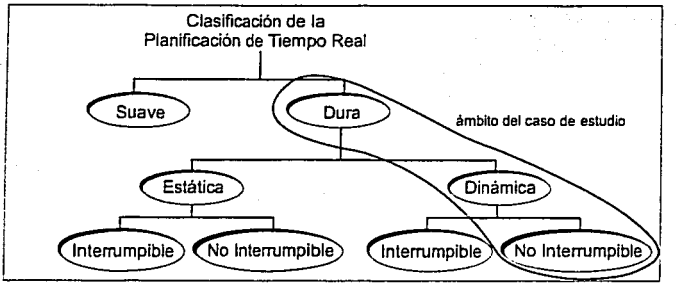


Fig. 2.4: Clasificación de los planificadores de tiempo real

La planificación estática y dinámica pueden ser a la vez en interrumpibles y no-interrumpibles:

- En la planificación *interrumpible*, una tarea en ejecución puede ser interrumpida para ceder a la ejecución de otra tarea con mayor prioridad.
- En la planificación *no-interrumpible*, una tarea no puede ser interrumpida hasta que ésta decida dejar los recursos, lo que normalmente sucede hasta que completa su trabajo.

Con base a esta clasificación, el *algoritmo de ordenamiento por planes* se clasifica como un planificador duro, dinámico, y no-interrumpible.

2.3.1 Algoritmo de frecuencia monótona

Para utilizar el *algoritmo de ordenamiento por planes*, se hace necesario antes revisar un planificador previo que es clásico en la literatura de planificación, este es el *algoritmo de frecuencia monótona* (*rate monotonic algorithm*). Este planificador monotónico, propuesto por Liu y Layland [Liu y Layland(1973)], es interrumpible, y se maneja por prioridades. Las suposiciones que se deben cumplir para que funcione un planificador de tipo monotónico en un solo procesador son:

1. Todas las tareas que tienen *plazos límites* duros, son periódicas, y se ejecutan con intervalos constantes entre ellas.
2. Cada tarea debe ser completada antes que la siguiente petición sea hecha. La *k-ésima tarea*, llamada τ_k, se realiza en el tiempo $k \cdot P_k$[4], y debe concluir antes

[4]Donde k es entero y P_k es el periodo de la tarea τ_k.

del tiempo $(k+1) \cdot P_k$, con lo que se eliminan los problemas de espera entre tareas.

3. Las tareas son independientes, es decir, la realización de una de ellas no depende del inicio o conclusión de otra.

4. El *tiempo de ejecución* (el tiempo empleado por el procesador para ejecutar una tarea sin interrupción) se supone constante.

5. Cualquier tarea no periódica se considera como especial, y se supone que no contiene *plazos límites* críticos.

6. Todas las tareas están listas a ejecutarse antes de comenzar la planificación.

Las suposiciones anteriores permiten caracterizar a cada τ_i mediante dos parámetros: su periodo P_i y su tiempo de proceso C_i con $0 < C_i < P_i$. Así, se define como *factor de utilización* a C_i/P_i, que es la fracción de tiempo de procesador que se invierte en la ejecución de τ_i. Para un conjunto de m tareas, el factor de utilización es:

$$U = \sum_{i=1}^{m} \frac{C_i}{P_i} \tag{2.1}$$

Aunque la utilización del proceso puede mejorarse al incrementar los valores de C_i, o haciendo un decremento en los valores de P_i, los parámetros de las tareas están restringidos a que cada tarea debe cumplir sus plazos límites. Se dice que un conjunto de tareas *utiliza* completamente al procesador si:

- La asignación de prioridades hace planificable dicho conjunto.
- Un incremento en el tiempo de proceso de alguna de las tareas hace que el conjunto no sea planificable.

Se considera que el planificador monotónico es *óptimo*[5] al asignar las prioridades de acuerdo al inverso de los periodos de las tareas y tiene un mínimo de factor utilización de $U = m\left(2^{1/m} - 1\right)$. Este valor es aproximadamente 83% cuando $m = 2$ y $ln\,(2)$ cuando $m \to \infty$. Por lo tanto, cualquier conjunto de m tareas periódicas que tenga un factor de utilización total no mayor a $U = m\left(2^{1/m} - 1\right)$ será planificable, es decir, cada una de las m tareas deberá cumplir con todos sus *plazos límites*.

El planificador monotónico es básico, y ha sido usado extensivamente en la teoría de la planificación. A pesar, de que en el artículo original el planteamiento de la demostración de la prueba de planificación no es correcto, como lo menciona R.

[5]Se dice que un conjunto de tareas es "óptimo" bajo un algoritmo de planificación definido, si el conjunto de tareas es planificable, y si un incremento en el valor del tiempo de ejecución en alguna de las tareas hace que el conjunto ya no sea planificable.

Devillers[6] [Devillers y Goossens(2000)], el resultado final sí lo es, por lo que se sigue utilizando en muchas aplicaciones.

2.3.2 Algoritmo de ordenamiento por planes

Una vez establecido el algoritmo de frecuencia monótona, es posible revisar al *algoritmo de ordenamiento por planes* como una mejora en el manejo de la planificación en los sistemas de cómputo de tiempo real reconfigurables.

El *algoritmo de ordenamiento basado en planes* [Almeida *et al.*(1999)] hace un plan con base en los valores de los parámetros de las tareas dadas en un instante de tiempo para que sea despachado y aplicado en el sistema de cómputo de tiempo real en un determinado tiempo llamado *ventana de tiempo* V. El tamaño de V es fijo e independiente de los periodos de las variables. Este planificador es casi-dinámico, ya que presenta algunas características de los planificadores dinámicos [Fonseca y Almeida(1999)].

El proceso que se realiza en la operación de un sistema de cómputo de tiempo real utilizando el algoritmo de ordenación por planes se muestra en la Fig. 2.5. En este proceso, el planificador revisa la tabla de los parámetros de las tareas cada ventana de tiempo y determina si el conjunto de trabajos es planificable o no. En caso de que el conjunto sea planificable, se genera un plan que va a ser distribuido por un programa llamado despachador hacia los nodos correspondientes del sistema de cómputo de tiempo real. En los nodos se ejecuta en plan calculado para cada tarea.

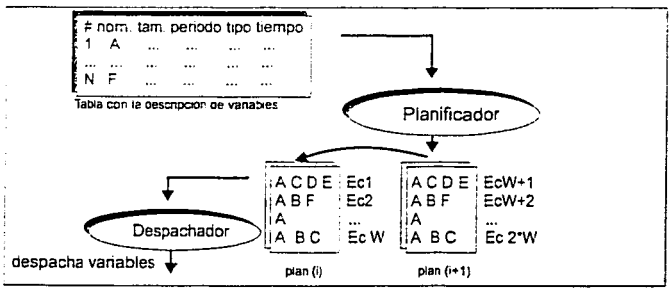


Fig. 2.5: Planificador por planes

[6]Para que el algoritmo de planificación monotónico sea óptimo, en las condiciones iniciales se debe considerar que todas las tareas deben estar listas para su ejecución antes de comenzar la planificación.

Dentro de una ventana V, y siendo P_i el periodo de la *i-ésima tarea* de un conjunto de N tareas, se sabe que hay a lo más S operaciones, dadas por la siguiente ecuación [Almeida *et al.*(1999)]:

$$S = \sum_{i=1}^{N} \left(\left\lceil \frac{V}{P_i} \right\rceil + 1 \right) \tag{2.2}$$

Nótese que la complejidad de S es $O(N)$. En comparación, en un planificador dinámico, éste debe ser invocado cada vez que haya terminado un evento para poder determinar el siguiente plan. Lo anterior lo logra revisando el conjunto de N variables. Por lo tanto el número de operaciones debe ser del orden de $N * S$, lo que resulta en una complejidad de $O(N^2)$. De hecho, las propiedades de un planificador por planes dependen del tamaño de V. Si $V = 0$, el planificador por planes se comportará como un planificador dinámico. Si por el contrario, $V = mcm(P_i)$ (el mínimo común múltiplo de los periodos de las tareas, conocido como *macrociclo* o MC), entonces el planificador por planes se comportará como un planificador estático.

Las características principales del planificador por planes son:

- *Gran flexibilidad comparado con el planificador estático.* Como los planes se realizan cada ventana de tiempo, si ocurre algún cambio es posible tomarlo en cuenta en la próxima generación del plan.
- *Baja utilización en tiempo de ejecución con respecto del planificador dinámico.* Como ya se vio, la complejidad del planificador por planes es de $O(N)$ contra la del planificador dinámico que es de $O(N^2)$.
- *Requerimientos acotados de memoria.* Los planes de las ventanas son acotados, ya que la ventana de tiempo es fija, siendo el tamaño de la ventana ajustado dependiendo del número y longitud del periodo de las tareas.
- *Limitada garantía de planificabilidad.* El algoritmo de ordenamiento por planes, al igual que el planificador dinámico, no puede garantizar la planificabilidad en el plan siguiente, por lo que se requiere un analizador en línea de los planes cada vez que haya un cambio en el conjunto de las tareas.

2.3.3 Análisis de planificabilidad del algoritmo de ordenamiento por planes

La falta de garantía del cumplimiento de las restricciones temporales futuras del algoritmo de ordenamiento por planes hace necesario introducir en tiempo real un analizador de planificabilidad. La función del analizador es determinar cuando una variable puede cumplir o no con sus requerimientos temporales, es decir, debe decidir si un conjunto de tareas es planificable o no. Para el problema de planificación se

supone que las tareas son periódicas, independientes y no-interrumpibles sobre un canal (*bus*) simple [Almeida *et al.*(1999)].

En el caso particular del planificador monotónico, una de las comprobaciones de planificabilidad más comunes se deriva de la hecha por C. L. Liu (sección 2.3.1). Esta comprobación es baja en consumo de proceso, ya que está dada por una sola expresión y es aplicada a tareas interrumpibles. Para poder aplicar esta comprobación de planificabilidad al planificador por planes es necesaria una modificación, debido a que se supone que en el planificador por planes las tareas son no-interrumpibles. Para esto, se tiene que suponer: (*a*) los periodos de todas las tareas son un múltiplo de un *ciclo elemental* CE, es decir, durante un CE varias tareas pueden ser realizadas; y (*b*) se usa la técnica de *inserción de tiempos ociosos* (*idle time insertion*).

La técnica de *inserción de tiempos ociosos* que acomoda las distintas tareas en cada CE se observa en la Fig. 2.6. La política en el orden de ejecución es semejante a la del planificador monotónico, es decir, las tareas que se ejecutan primero son las que tienen menor periodo.

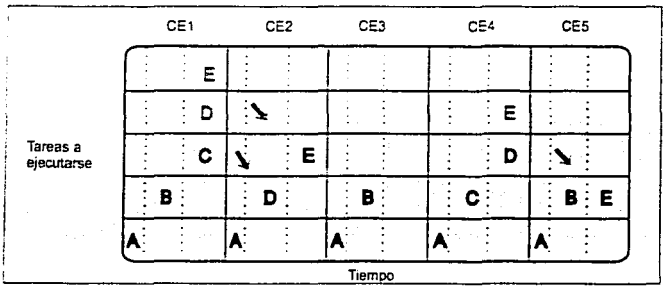


Fig. 2.6: Planificación de tareas de acuerdo planificador por planes

En la Fig. 2.6 se aprecia cómo se acomodan distintas tareas en una ventana de tiempo de cinco CE's. Cada una de las tareas está identificada por una letra: la tarea A tiene un periodo de un CE y un tiempo de ejecución de un quinto de CE; las tareas B, C, D y E tiene un periodo de dos CE's, con un tiempo de ejecución de dos quintos de CE.

De acuerdo a la política del planificador por planes, las tareas de mayor frecuencia son las que tienen mayor prioridad en la ejecución, por lo tanto en la Fig. 2.6 las tareas A y B son las primeras que se ejecutan. En el proceso de acomodar las diferentes tareas, es posible que el total del tiempo utilizado en una tarea al ejecutarse sobrepase el tamaño de un CE (como se muestra en la Fig. 2.6 en el CE_1). Se dice entonces que el CE está sobrecargado. Esto se resuelve mediante posponer la ejecución de las tareas que no se pudieron ejecutar en un CE al siguiente CE. Este proceso se repite

si los siguientes CE's están sobrecargados.

Por ejemplo, como puede verse en la Fig. 2.6, en el CE_1 hay cinco tareas listas a ejecutarse, A, B, C, D y E. Sin embargo, el CE sólo tiene espacio para ejecutar A, B, y C, por lo que D y E se tienen que ejecutar en el siguiente ciclo, CE_2. La misma situación sucede en CE_4. Nótese que cuando una tarea no puede realizarse en cierto ciclo elemental, se añaden flechas en la Fig. 2.6 que indican el lugar en el tiempo en donde se deben de ejecutar.

Es común que no siempre se cubra completamente el tiempo de cada CE. Al tiempo que sobra en cada CE se le llama *máximo tiempo ocioso* X_n. Estos tiempos denotados por X_n se observan en la Fig. 2.7.

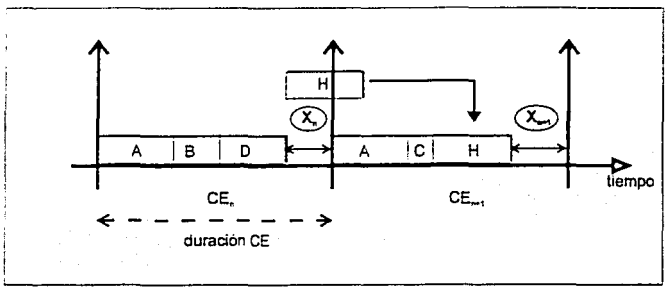


Fig. 2.7: Tiempos ociosos

La técnica de *inserción de tiempos ociosos* da la siguiente condición de planificabilidad. Si N es el número de tareas, P_i el periodo de la variable i, y C_i la duración del tiempo de ejecución, se tiene entonces la siguiente *comprobación de planificabilidad* [Almeida *et al.*(1999)]:

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} < N(2^{\frac{1}{N}} - 1) \times \frac{CE - X'}{CE} \tag{2.3}$$

donde CE es la duración del ciclo elemental, X' es el máximo tiempo ocioso de todas las CE en V. $X' = max(X_n)$ y N es el número de tareas. Esto garantiza que el conjunto de N tareas es planificable con cualquier fase. Por otro lado, cuando $max(X_n)$ se calcula con un gasto computacional alto (muchas operaciones en el CPU), es posible usar la siguiente desigualdad:

$$X' = \max(X_n) \leq \max_{i=1..N}(C_i) \tag{2.4}$$

2.4 Motivación del método

Como se menciona anteriormente, la característica más significativa en los sistemas de cómputo de tiempo real es su funcionamiento restringido en el tiempo. Por esto, en el diseño de tales sistemas, debería considerarse explícitamente los requerimientos temporales para el sistema de cómputo de tiempo real de tal forma que su comportamiento pudiera ser predecible. Esto es, hacer una planificación de las tareas a realizar por el sistema de tiempo real. Sin embargo, a pesar de la importancia que tiene la planificación, no es un requerimiento temporal considerado para muchas aplicaciones.

En general, de acuerdo al modelo propuesto por M. Saksena [Saksena(1998)] (véase el apéndice B), los requerimientos de rendimiento de un sistema admiten muchas posibilidades en las restricciones temporales a nivel sistema, las cuales permiten de nuevo muchas posibilidades en las restricciones temporales de las tareas. Desde el punto de vista del diseño de un sistema de cómputo de tiempo real, el objetivo es sistemáticamente derivar un conjunto de parámetros de tareas que no solamente cumplan con los requerimientos de rendimiento del sistema, si no que además se desea que sean planificables.

Particularmente, en los sistemas de cómputo distribuido de tiempo real disparados por tiempo, las actividades son manejadas dentro de ciclos regulares de tiempo. Las tareas se pueden ejecutar en los nodos y/o en un procesador en forma concurrente, de tal manera que cada tarea puede ser modelada como una tarea periódica que realiza algún cómputo en cierto tiempo durante cada ejecución. Para estudiar la planificación en un sistema de cómputo distribuido de tiempo real reconfigurable, en esta tesis se propone un método que permite analizar los parámetros de las tareas en la dinámica del sistema, y encontrar combinaciones de parámetros adecuados de funcionamiento. Tal análisis puede ser utilizado para ayudar en el diseño y ajuste de los parámetros de las tareas en aplicaciones de tiempo real.

2.5 Resumen

El método desarrollado en el capítulo 3 da una solución al problema de planificación en un sistema de cómputo distribuido de tiempo real, disparado por tiempos y reconfigurable. Para los propósitos de esta tesis, la reconfiguración se entiende como la posible variación en el tiempo de los valores de P y C de las tareas que intervienen en el funcionamiento del sistema de tiempo real. Por este motivo, en las secciones 2.1.1 y 2.1.2 se definen los sistemas de cómputo de tiempo real, así como los sistemas de cómputo de tiempo real distribuidos. Enseguida, en la secciones 2.2 y 2.3 se explica cuál es la función de un planificador, y una clasificación. Esto permite ubicar qué clase de algoritmo de ordenamiento se utiliza en el método propuesto. En la sección 2.3.1, se revisa al algoritmo de ordenamiento por planes de frecuencia monótona, ya que tanto su prueba de planificabilidad como su política de asignación de prioridades

en las tareas son utilizados para el análisis de planificabilidad del planificador por planes. Este último algoritmo (sección 2.3.2) es el finalmente utilizado en el método desarrollado en el capítulo 3.

Capítulo 3

Un método para la planificación

Este capítulo presenta y plantea el método propuesto para planificar el conjunto de tareas de un sistema de cómputo distribuido, de tiempo real, disparado por tiempo, y reconfigurable. El método se ejecuta en dos partes:

- *Fuera de línea.* La primera parte del método consiste en encontrar parámetros de tareas adecuados para una aplicación de cómputo de tiempo real obtenidos fuera de línea. Con este procedimiento, se forma una tabla de conjuntos de parámetros de las tareas que se denomina *tabla de parámetros adecuados* y sirve de base para la segunda parte del método que se ejecuta en línea.
- *En línea.* Esta etapa del método se realiza en línea, es decir, en el tiempo en el que esté funcionando la aplicación. Esta parte sirve para encontrar en línea y mediante la *tabla de parámetros adecuados*, los parámetros de tareas que garantizan un adecuado funcionamiento en implantaciones de sistemas de cómputo de tiempo real duros, distribuidos, no interrumpibles, disparados por tiempo y reconfigurables.

3.1 El método

El método para planificar el conjunto de tareas de un sistema de cómputo de tiempo real distribuido, disparado por tiempo, y reconfigurable que se presenta en esta tesis se propone a fin de analizar los parámetros de las tareas en la dinámica del sistema, y encontrar combinaciones de parámetros adecuados de funcionamiento que cumplan con los requerimientos del sistema y que sean planificables. El método tiene los siguientes pasos:

- Fuera de línea
    1. *Obtención de la tabla de posibles valores.* A partir de los requerimientos de la aplicación y del sistema de cómputo distribuido, se proponen el

número i de tareas, el valor máximo para el periodo de ejecución de cada tarea (P_i), y el valor del tiempo de ejecución de cada tarea (C_i). Con esta información se forma la *tabla de posibles valores* para P's y C's de cada tarea.

2. *Filtrado de listas de configuración planificables con base al algoritmo de ordenamiento por planes.* La tabla de posibles valores puede ser considerada como un conjunto de listas de configuración de los parámetros P's y C's de cada tarea. Utilizando el algoritmo de ordenamiento por planes, las listas son discriminadas de acuerdo a su planificabilidad, es decir, se eliminan *(a)* aquellas listas que no son planificables, y *(b)* aquellas listas que se consideran repetidas dentro de la tabla. De esta forma, se obtiene una nueva tabla que contiene las listas planificables y no repetidas.
3. *Simulación en el sistema de cómputo de tiempo real.* Tomando la nueva tabla de listas planificables y no repetidas, se realiza una simulación de la aplicación en el sistema de cómputo de tiempo real, a fin de seleccionar las listas de parámetros que sean adecuadas de acuerdo con los requerimientos de rendimiento de la aplicación y del sistema de cómputo distribuido. De esta selección, se genera una tabla de parámetros que cumplen con los requerimientos de rendimiento de la aplicación y que son planificables. La tabla obtenida en este paso se conoce como *tabla de parámetros adecuados.*

- En línea

1. *Lectura de una lista de configuración.* De la tabla de parámetros adecuados se lee una lista de configuración para las tareas del sistema de tiempo real. Esta lista establece una configuración para el sistema, y puede escogerse ya sea por parte del operador, o automáticamente, por parte del sistema. En el caso de que el operador seleccionara una lista de parámetros que no se encuentra en la tabla de parámetros adecuados, el sistema es capaz de buscar y seleccionar la lista de configuración más parecida dentro de la tabla de parámetros adecuados.
2. *Cálculo del plan de acuerdo a la lista de configuración.* Partiendo de la lista de configuración, se calcula un plan de ejecución de tareas, por medio de la técnica de inserción de tiempos ociosos. Este plan establece los tiempos de ejecución de las tareas.
3. *Distribución del plan de ejecución de tareas.* El plan de ejecución de tareas se despacha a los diferentes nodos dentro del sistema de cómputo distribuido de tiempo real, para ejecutarse. En el caso en que el sistema distribuido sea reconfigurado durante la ejecución por el operador o automáticamente (agregando o quitando nodos, cambiando prioridades, cambiando los tiempos de ejecución, cambiando los periodos de ejecución, u otros), se propone la prueba de otra lista de configuración, partiendo del paso 1 del procedimiento en línea. Ahora bien, si el sistema de tiempo real funciona adecuadamente, entonces la lista de parámetros se considera la

adecuada, de acuerdo con los requerimientos de la aplicación y del propio sistema.

De esta forma, el método parte de los requerimientos temporales de la aplicación y del sistema de cómputo distribuido, y produce una lista de parámetros adecuados para la planificación de las tareas del sistema de cómputo distribuido de tiempo real. A continuación, se presenta una descripción detallada de cada una de las partes y los pasos del método.

3.2 Descripción del método

La propuesta de diseño se propone aplicable a sistemas de cómputo de tiempo real duros, distribuidos, disparados por tiempo, y reconfigurables. Para manejar la reconfiguración existen varios algoritmos de planificación que pueden ser utilizados. Por ejemplo, el algoritmo de planificación de G. Fohler [Fohler(1995)] propone distribuir la carga de tareas manejando una parte dinámica y una estática. Sin embargo, se utiliza el algoritmo de planificación por planes propuesto por L. Almeida [Almeida *et al.*(1999)] (sección 2.3.2), debido a las siguientes razones:

- Alta flexibilidad comparada con los planificadores estáticos, ya que los planes son recalculados cada periodo de tiempo fijo y da la oportunidad de tomar en cuenta los cambios que hayan ocurrido en ese periodo.
- Bajo tiempo de proceso comparado con los planificadores dinámicos. Esto se debe a que el planificador por planes crea una tabla cada determinado tiempo, teniendo así las características de los planificadores estáticos.
- Requerimientos de memoria acotados. El planificador por planes tiene un conocimiento completo de las actividades del bus en cada plan. La cantidad de información permanece acotada, debido a que la creación de cada plan se hace para una ventana de tiempo.

En la sección 2.2, se establece que el problema de planificación queda completamente definido por la información de los tiempos de proceso y frecuencia de cada una de las tareas. Por ejemplo, se tienen seis tareas cuyos parámetros se escriben en la tabla 3.1.

Tarea	Periodo	Tiempo de proceso
1	P_1	C_1
2	P_2	C_2
3	P_3	C_3
4	P_4	C_4
5	P_5	C_5
6	P_6	C_6

Tabla 3.1: Parámetros de las tareas a planificar

Para simplificar la notación de la tabla 3.1, ésta se presenta en forma de un arreglo como en la expresión 3.1, donde las $P's$ denotan los periodos y las $C's$ denotan los tiempos de proceso. Este arreglo representa la llamada *lista de configuración.*

$$[P_1, P_2, P_3, P_4, P_5, P_6, C_1, C_2, C_3, C_4, C_5, C_6] \tag{3.1}$$

Para analizar la respuesta de un sistema de tiempo real, utilizando la lista 3.1, es necesario hacer un análisis sobre todos los valores posibles que pueda tener los periodos y tiempos de proceso. Aunque el problema se presenta como $np - completo$, se puede hacer un análisis para disminuir el número de casos que generan los diferentes escenarios.

Recordando que, de acuerdo al algoritmo de ordenamiento por planes, los periodos de las tareas son múltiplos del CE, y que los tiempos de proceso no pueden ser mayores a 1 CE, (sección 2.3.2). Así, por ejemplo, se pueden tener los siguientes parámetros para las tareas del ejemplo:

$$[3, 4, 4, 5, 2, 2, 0.1, 0.15, 0.6, 0.8, 0.7, 0.55] \tag{3.2}$$

Es decir, en este ejemplo, la tarea 0 tiene un periodo de 3 CE's y un tiempo de proceso de 0.1 CE. La tarea 1 tiene un periodo de 4 CE's y un tiempo de proceso de 0.15 CE. Y así con las demás tareas.

Ahora bien, en un sistema reconfigurable es posible que en una ventana de tiempo posterior los parámetros de las tareas pueden cambiar debido a algún reajuste (o reconfiguración) del sistema. Por ejemplo, si el periodo de la tarea 0 cambiara de 3 a 2 CE's. Para considerar cambios como éste, se propone el método, de modo que sea posible encontrar los parámetros adecuados de funcionamiento de sistemas de cómputo de tiempo real duros, distribuidos, disparados por tiempo.

El método se describe en las siguientes secciones en forma detallada, a partir de sus partes o procedimientos fuera de línea y en línea, y siguiendo el ejemplo propuesto para su ilustración.

3.2.1 Procedimiento fuera de línea

El procedimiento de transformar las especificaciones de rendimiento al nivel de restricciones temporales de tareas puede ser visto como un problema complejo de restricciones no lineal. Así, con el fin de resolver tal problema, se propone el método como una solución heurística que analiza un tipo de sistema de tiempo real reconfigurable, y que propone un conjunto de tareas que toman en cuenta las especificaciones de rendimiento y que cumplen con sus restricciones temporales.

Para ejemplificar el método en esta etapa fuera de línea, se supone para el ejemplo propuesto y sin perdida de generalidad que el máximo periodo es 5 CE's, la resolución es de 0.2 CE y el número de tareas es 6. A continuación se presentan los tres pasos:

1. *Obtención de la tabla de posibles valores*

   A partir de la información de la tabla 3.1 de tareas a planificar, se genera la tabla de posibles valores. Para obtener esta tabla, en primer lugar se toman las combinaciones sin repetición del número de tareas a planificar desde el valor de 1CE[1] hasta el límite determinado por la aplicación, (véase la tabla 3.2, donde el valor máximo es igual a cinco).

#lista	P_1	P_2	P_3	P_4	P_5	P_6
1	1	1	1	1	1	1
2	1	1	1	1	1	2
3	1	1	1	1	1	3
4	1	1	1	1	1	4
5	1	1	1	1	1	5
6	1	1	1	1	2	2
7	1	1	1	1	2	3
					⋮	⋮
204	4	4	4	4	4	5
205	4	4	4	4	5	5
206	4	4	4	5	5	5
207	4	4	4	5	5	5
208	4	4	5	5	5	5
209	4	5	5	5	5	5
210	5	5	5	5	5	5

Tabla 3.2: Tabla de posibles valores de los periodos para seis tareas

La razón por la cual se toman las combinaciones sin repetición es que cualquier lista de configuración se puede reordenar en pares de periodos y tiempos de ejecución de acuerdo al periodo ascendente sin afectar su planificabilidad. Nótese

[1]Mínimo valor posible de un periodo en el planificador por planes.

que de acuerdo a la ec. 2.3 (sección 2.3.3), la planificabilidad está determinada por el factor de utilización. Por lo tanto, si se altera el orden de los factores, la suma no resulta afectada. Por ejemplo, si se tienen tres tareas, el factor de utilización es el mismo en la ec. 3.3.

$$\begin{aligned} &\frac{C_1P_2P_3 + C_2P_1P_3 + C_3P_1P_2}{P_1P_2P_3} = \frac{C_1P_3P_2 + C_2P_3P_1 + C_3P_2P_1}{P_3P_2P_1} \\ &= \frac{C_2P_1P_3 + C_1P_2P_3 + C_3P_1P_2}{P_1P_2P_3} = \frac{C_2P_3P_1 + C_1P_3P_2 + C_3P_2P_1}{P_3P_2P_1} \end{aligned} \tag{3.3}$$

Para obtener la tabla de los tiempos de ejecución (tabla 3.3), primero se debe considerar la resolución mínima requerida en los tiempos de ejecución de las tareas. Teniendo el valor de esta resolución, el valor de los tiempos de ejecución de cualquier tarea se considera como un número entero de veces el valor de la resolución, hasta cubrir un máximo valor de 1 CE (sección 2.3.2). Una vez determinada la resolución, se forma una tabla con todas las combinaciones de los parámetros posibles en los tiempos de ejecución de las tareas en forma ascendente como se muestra para el ejemplo en la tabla 3.3.

#lista	C_1	C_2	C_3	C_4	C_5	C_6
1	0.2	0.2	0.2	0.2	0.2	0.2
2	0.2	0.2	0.2	0.2	0.2	0.4
3	0.2	0.2	0.2	0.2	0.2	0.6
15.618	0.8	0.8	0.8	0.8	0.8	0.8
15.619	0.8	0.8	0.8	0.8	0.8	1
15.620	0.8	0.8	0.8	0.8	0.8	1
15.621	0.8	0.8	0.8	0.8	1	1
15.622	0.8	0.8	0.8	1	1	1
15.623	0.8	0.8	1	1	1	1
15.624	0.8	1	1	1	1	1
15.625	1	1	1	1	1	1

Tabla 3.3: Tabla de posibles valores de los tiempos de ejecución, $C's$ para seis tareas

A partir de las tablas anteriores, se crea la *tabla de posibles valores* con todas las combinaciones de valores que incluyan tanto a los periodos como a los tiempos de ejecución. Este proceso da como resultado la tabla 3.4 para el ejemplo propuesto.

#lista	P_1	P_2	P_3	P_4	P_5	P_6	C_1	C_2	C_3	C_4	C_5	C_6
1	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.2	0.2
2	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.2	0.4
3	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.2	0.6
4	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.2	0.8
5	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.2	1
6	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.4	0.4
7	1	1	1	1	1	1	0.2	0.2	0.2	0.2	0.4	0.6
												⋮
3.281.248	10	10	10	10	10	10	0.8	0.8	1	1	1	1
3.281.249	10	10	10	10	10	10	0.8	1	1	1	1	1
3.281.250	10	10	10	10	10	10	1	1	1	1	1	1

Tabla 3.4: Tabla de posibles valores para seis tareas

2. *Filtrado de listas de configuración planificables con base al algoritmo de ordenamiento por planes*

   De la tabla 3.4 se retiran las listas de configuración que sean equivalentes desde el punto de vista de la planificación. Utilizando los mismos argumentos utilizados anteriormente, de acuerdo a la ec. 2.3 existen ciertas combinaciones que son equivalentes. Por ejemplo de las posibles combinaciones de la tabla 3.4, a continuación se presentan un conjunto de listas de configuración equivalentes desde el punto de vista de la planificación, tabla 3.5.

P_1	P_2	P_3	P_4	P_5	P_6	C_1	C_2	C_3	C_4	C_5	C_6
1	1	1	1	1	1	0.1	0.1	0.1	0.1	0.1	0.2
1	1	1	1	1	1	0.1	0.1	0.1	0.1	0.2	0.1
1	1	1	1	1	1	0.1	0.1	0.1	0.2	0.1	0.1
1	1	1	1	1	1	0.1	0.1	0.2	0.1	0.1	0.1
1	1	1	1	1	1	0.1	0.2	0.1	0.1	0.1	0.1
1	1	1	1	1	1	0.2	0.1	0.1	0.1	0.1	0.1

Tabla 3.5: Tabla con lista de configuración equivalentes desde el punto de vista de la planificabilidad

La tabla 3.4 representa una amplia gama de combinaciones que pueden ser utilizadas en el análisis para encontrar parámetros adecuados en los valores de las tareas. El conjunto de listas de configuración posibles sin utilizar la filtración es del orden de 3^6. Si se utilizan los argumentos anteriores, el orden es de alrededor de 5^5.

Por otro lado, se hace un filtro de las combinaciones que son planificables según la prueba del planificador de Almeida, mediante un análisis de planificabilidad

(sección 2.3.3) sobre cada lista de configuración, y así saber si es planificable o no. Para obtener las combinaciones que cumplen la comprobación de Almeida del ejemplo, se construye un programa para la verificación. Una representación del resultado se presenta en la tabla 3.6.

#lista	P_1	P_2	P_3	P_4	P_5	P_6	C_1	C_2	C_3	C_4	C_5	C_6
1	1	1	1	1	2	2	0.2	0.2	0.2	0.2	0.2	0.2
2	1	1	1	1	2	4	0.2	0.2	0.2	0.2	0.2	0.2
3	1	1	1	1	3	3	0.2	0.2	0.2	0.2	0.2	0.2
4	1	1	1	1	4	4	0.2	0.2	0.2	0.2	0.2	0.2
5	1	1	1	1	5	5	0.2	0.2	0.2	0.2	0.2	0.2
6	1	1	1	2	2	2	0.2	0.2	0.2	0.2	0.2	0.2
												⋮
3.006	5	5	5	5	5	5	0.4	0.6	0.6	0.8	1	1
3.007	5	5	5	5	5	5	0.4	0.6	0.6	1	1	1
3.008	5	5	5	5	5	5	0.4	0.6	0.8	0.8	0.8	0.8
3.009	5	5	5	5	5	5	0.4	0.6	0.8	0.8	0.8	1
3.010	5	5	5	5	5	5	0.4	0.6	0.8	0.8	1	1
3.011	5	5	5	5	5	5	0.4	0.6	0.8	1	1	1
3.012	5	5	5	5	5	5	0.4	0.6	1	1	1	1

Tabla 3.6: Tabla con lista de configuración planificables según el algoritmo de ordenamiento de Almeida

3. *Simulación en el sistema de cómputo distribuido de tiempo real*

   Después de obtener las combinaciones planificables, es necesario saber si la lista de configuración funciona adecuadamente en el sistema distribuido de tiempo real. Para esto se hace una simulación del sistema, haciendo que funcione con los parámetros propuestos (que ya son planificables, de acuerdo con el paso 2) y observar si su funcionamiento es adecuado de acuerdo con los requerimientos de rendimiento de la aplicación y del sistema de cómputo distribuido. En el caso de que la lista de configuración resulte en un funcionamiento adecuado del sistema, ésta se marca como aceptable y se añade a la *tabla de parámetros adecuados* que cumplen los requerimientos y son planificables. En el ejemplo de seis tareas a planificar, el resultado de la tabla de listas de configuración planificables, considerando los requerimientos de una aplicación y un sistema de cómputo distribuido, podría ser como la mostrada en la tabla 3.7:

#lista	P_1	P_2	P_3	P_4	P_5	P_6	C_1	C_2	C_3	C_4	C_5	C_6	adecuado
1	1	1	1	1	2	2	0.2	0.2	0.2	0.2	0.2	0.2	√
2	1	1	1	1	2	4	0.2	0.2	0.2	0.2	0.2	0.2	×
3	1	1	1	1	3	3	0.2	0.2	0.2	0.2	0.2	0.2	√
4	1	1	1	1	4	4	0.2	0.2	0.2	0.2	0.2	0.2	√
5	1	1	1	1	5	5	0.2	0.2	0.2	0.2	0.2	0.2	×
6	1	1	1	2	2	2	0.2	0.2	0.2	0.2	0.2	0.2	×
												⋮	
3.006	5	5	5	5	5	5	0.4	0.6	0.6	0.8	1	1	×
3.007	5	5	5	5	5	5	0.4	0.6	0.6	1	1	1	√
3.008	5	5	5	5	5	5	0.4	0.6	0.8	0.8	0.8	0.8	√
3.009	5	5	5	5	5	5	0.4	0.6	0.8	0.8	0.8	1	√
3.010	5	5	5	5	5	5	0.4	0.6	0.8	0.8	1	1	×
3.011	5	5	5	5	5	5	0.4	0.6	0.8	1	1	1	√
3.012	5	5	5	5	5	5	0.4	0.6	1	1	1	1	×

Tabla 3.7: Tabla con lista de configuración planificables y adecuados según el algoritmo de ordenamiento de Almeida

Como parte del ejemplo, a continuación se muestra cómo son planificadas las tareas de la primera lista de configuración de la tabla 3.7.

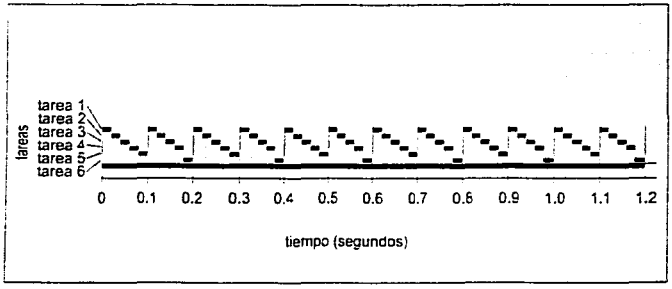


Fig. 3.1: Esquema de ordenación de tareas en el tiempo de la lista de configuración [1,1,1,1,2,2,0.2,0.2,0.2,0.2,0.2,0.2,0.2]

El procedimiento anterior permite filtrar las combinaciones posibles en el sistema que sean planificables y que induzcan una respuesta satisfactoria en el funcionamiento. En consecuencia, con el procedimiento fuera de línea se obtiene la *tabla de parámetros*

adecuados, cuyos datos son adecuados de una manera heurística[2] tanto respecto a la funcionalidad del sistema como también en la teoría de los sistemas de tiempo real.

3.2.2 Procedimiento en línea

Con la información de la *tabla de parámetros adecuados*, es posible ajustar parámetros de desempeño generales para las tareas durante la etapa de diseño y la fase de implantación de un sistema de cómputo de tiempo real duro, distribuido, disparado por tiempos bajo un esquema de reconfiguración, para seleccionar una lista de configuración adecuada respecto a los requerimientos de la aplicación y del sistema distribuido. El procedimiento sugerido utiliza el algoritmo de ordenamiento por planes, tiene el esquema de un ciclo que se ejecuta cada ventana de tiempo V (véase sección 2.3.2).

1. *Lectura de una lista de configuración.*

   Se lee una lista de configuración, que establece una configuración para el sistema. Se da la opción de que esta lista sea seleccionada por parte del operador o por parte del sistema de manera automática. Para el ejemplo propuesto, supóngase que el operador selecciona la siguiente lista de configuración 3.4:

   $$[P_1, P_2, P_3, P_4, P_5, P_6, C_1, C_2, C_3, C_4, C_5, C_6] \tag{3.4}$$

   En el caso de que el operador seleccionara una lista de parámetros que no se encuentra en la tabla de parámetros adecuados, el sistema es capaz de buscar y seleccionar la lista de configuración más parecida dentro de la tabla de parámetros adecuados. Ahora bien, si se trata de la primera vez que se ejecuta el ciclo, el planificador puede tomar información del operador, o utilizar valores de iniciales pre-determinados. Por otro lado, en caso de estar en operación el ciclo, el planificador verifica si ha cambiado la configuración del sistema distribuido, en cuyo caso toma en cuenta nuevos valores. De no ser el caso, los valores se mantienen.

   En el ejemplo, se considera que la lista de configuración 3.4 contiene la información completa de los parámetros de las tareas. El planificador selecciona una lista de configuración de la tabla de parámetros adecuados, y la utiliza para garantizar planificabilidad y el cumplimiento de requerimientos de la aplicación y el sistema.

2. *Cálculo del plan de acuerdo a la lista de configuración.*

   Partiendo de la lista de configuración seleccionada en el paso anterior, se calcula entonces un plan de ejecución con base en los valores de los parámetros de las

[2]Se considera que es heurística porque no hay una solución analítica para decidir si un conjunto de parámetros proporciona un adecuado funcionamiento o no. La decisión depende del conocimiento que se tenga de la aplicación.

tareas, por medio de la técnica de inserción de tiempos ociosos. Este plan establece los tiempos de ejecución de las tareas. Este paso puede ser opcional, ya que con anterioridad se pudo haber guardado planes correspondientes en la tabla de parámetros adecuados. Si esto no es así, entonces se procede a calcular el plan, que será distribuido a los diferentes nodos del sistema de cómputo distribuido.

3. *Distribución del plan de ejecución de tareas.*

   El plan de ejecución se envía a los diferentes nodos para ejecutarse durante una ventana de tiempo y antes de inicializar la siguiente ventana, es decir, se realiza un ciclo por cada ventana de tiempo. Ahora bien, si el sistema distribuido sufre alguna modificación durante la ejecución en una ventana de tiempo por parte del operador o del sistema automáticamente se propone la prueba de otra lista de configuración. Las modificaciones al sistema distribuido pueden deberse a que durante la ejecución se añadan o quiten nodos, se cambien prioridades, los tiempos de ejecución, los periodos de ejecución, etc. De este modo, la planificación se re-comienza partiendo del paso 1 del procedimiento en línea. Si el sistema de tiempo real funciona adecuadamente, entonces la lista de parámetros se considera la adecuada, de acuerdo con los requerimientos de la aplicación y del propio sistema.

3.3 Representación gráfica del método

El proceso del método propuesto puede expresarse gráficamente, como se muestra en la Fig. 3.2.

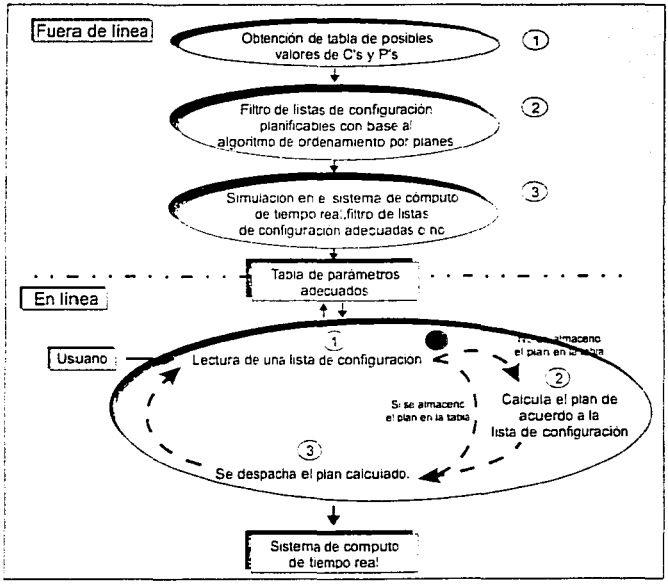


Fig. 3.2: Diagrama esquemático del proceso del método propuesto para encontrar parámetros adecuados de tareas en un sistema de cómputo de tiempo real

De esta forma, se ha propuesto un primer análisis de un método que propone una solución en línea, para encontrar conjuntos de parámetros de las tareas, de un sistema de cómputo de tiempo real, duro, distribuido, disparado por tiempo, que combina tanto el rendimiento de una aplicación como el área temporal del tiempo real.

3.4 Resumen

El presente capítulo muestra el método propuesto en esta tesis. El método tiene como base un análisis heurístico del sistema, no tratando de contraponer un posible análisis formal, sino como una propuesta inicial para comprender la problemática de la reconfiguración en línea, con base a restricciones de tiempo real. El método se basa en la construcción y revisión de la *tabla de posibles valores*, la cual cubre la totalidad de las combinaciones posibles para el ordenamiento de tareas. La información contenida en tal tabla es filtrada (fuera de línea) con base a criterios de planificación básicos, establecidos por el algoritmo de ordenamiento por planes. Una vez establecida, la información de la tabla es usada (en línea) para la verificación de posibles planes, previo a ser distribuida al sistema de cómputo distribuido.

Capítulo 4

Evaluación del método

El capítulo 3 presenta la propuesta de un método que permite encontrar tiempos de ejecución y periodos de aquellas tareas involucradas en el funcionamiento de sistemas de cómputo de tiempo real. Para evaluar el método, este capítulo muestra su aplicación a un caso de estudio particular.

Este capítulo consta de tres secciones. La primer sección presenta como caso de estudio la obtención de un valor estimado de lecturas de sensores, a través de una técnica de votación. Ello permite evaluar la viabilidad del uso del método propuesto. El objetivo es mostrar cómo el método se usa para encontrar valores de parámetros de tareas que se ejecutarán a tiempo de acuerdo a sus plazos límites, manteniendo un adecuado funcionamiento con respecto a los requerimientos del caso de estudio. Dicho caso de estudio se presenta como una serie de pruebas en donde se comparan los valores obtenidos mediante el método y una serie de valores conocidos como referencia. Cabe hacer mención que aún cuando el algoritmo de votación es usado como parte del caso de estudio, no se lleva a cabo ningún análisis o prueba con respecto a la tolerancia a fallas dado que esto último sale de los objetivos del presente trabajo.

La segunda sección describe el proceso de implantación de un programa que simula el caso de estudio. Para su realización se utilizó el sistema operativo de tiempo real RT-Linux.

Finalmente, se muestran los resultados que se obtuvieron del caso de estudio. Estos permiten observar las diferencias entre escenarios con base al método propuesto.

4.1 El caso de estudio

Durante el proceso de interacción entre el sistema de cómputo y el objeto controlado es necesario que el primero conozca cuál es la evolución del estado del segundo, para realizar una adecuada respuesta. Dado que se requiere conocer la evolución del estado del objeto controlado, se utilizan sensores.

El esquema de tolerancia a fallas implica que se deben tener réplicas de sensores en la adquisición de información. Tal replicación de elementos tiene el problema de obtener un sólo valor a partir de las medidas de los diferentes sensores. Una técnica utilizada generalmente para resolver éste problema es la de la *votación* de sensores. Dichos sensores deben de obtener sus valores al mismo tiempo.

Si se piensa que el sistema de cómputo de tiempo real debe actuar en un esquema de tolerancia a fallas, entonces, las diferencias que se presentan respecto al valor promedio de las lecturas de los sensores no ocurriendo fallas en el transcurso del tiempo, ocurren cuando hay diferencias en las lecturas de las réplicas. De esta forma, las lecturas de los sensores dependen principalmente del periodo de muestreo asociado al nodo de votación. Para este caso, la planificación debe asegurar que las tareas se ejecutan bajo ciertos periodos de muestreo dependientes del periodo del nodo de votación, por lo que la planificación de cada tarea se ciñe a periodos de muestreo adecuados en forma general.

Sin embargo, si los periodos correspondientes a cada uno de los sensores y del algoritmo de votación son asignados de manera individual, se puede estudiar el papel de la planificación en dichos sistemas. Es importante hacer notar que el estudio correspondiente a la fiabilidad de esta aproximación con respecto a la tolerancia a fallas no es llevado a cabo en este trabajo debido a que sale del alcance de esta tesis. Este trabajo se enfoca en el análisis de la planificación de la reconfiguración en línea con base al método descrito en el capítulo anterior. Se propone, como caso de estudio que cada sensor lea y transmita la información al nodo de votación de acuerdo al plan generado por el algoritmo de planificación y no como lo sugiere la tolerancia a fallas obtener las lecturas de los nodos al mismo tiempo, véase la Fig. 4.2. Nótese que cada lectura de los nodos sensores es dependiente directa del tiempo asignado por la planificación. De esta manera se obtienen valores diferentes de los diferentes sensores en el tiempo y se resalta la planificación de tareas a nivel individual.

Es importante hacer notar que para propósitos de la presente tesis, este caso de estudio sirve únicamente para ilustrar y realizar una evaluación heurística del método. El caso de estudio se analiza con respecto a su planificabilidad y se complementa con los requerimientos de rendimiento de la aplicación. El objetivo es asegurar el cumplimiento de todos los *plazos límites*, en todas las situaciones, con un rendimiento predecible aun en los escenarios de saturación de peticiones de recursos.

A continuación se presenta el algoritmo pesos promedio (*Weighted Averaging Algorithm*) descrito por Paul R. Lorczak [Lorczak *et al.*(1989)] como elemento encargado de la votación y que se utiliza en el caso de estudio. En el algoritmo de pesos promedio, se tienen n versiones de programas de software que producen n salidas $x_1, x_2, x_3, \ldots, x_n$ relacionados con n valores w_i que cumplen con que:

$$\sum_{i=1}^{n} w_i = 1 \tag{4.1}$$

y se define cada elemento x como:

$$x = \sum_{i=1}^{n} w_i x_i \tag{4.2}$$

El valor x de la ecuación 4.2 es entonces el valor obtenido de las n salidas $x_1, x_2, x_3, \ldots, x_n$ utilizando el algoritmo de pesos promedio.

La elección de los pesos puede ser hecha antes de la implantación de la aplicación. Si por ejemplo se tiene una elección de las $w's$ como $w_1 = w_2 = w_3 = \ldots = w_n$, el resultado es la media usual de las salidas. Sin embargo, el usuario puede requerir pesos diferentes que reflejen el conocimiento de las salidas de acuerdo. por ejemplo, a su fiabilidad. periodo. tecnología. etc.

El peso también puede ser función de las salidas. En el artículo de Lorczak se hace una extensión del algoritmo de pesos promedio utilizando las salidas de la siguiente manera. Dadas las salidas $x_1, x_2, x_3, \ldots, x_n$ se definen los pesos w_i como:

$$w_i = \left(1 + \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{d^2(x_j, x_i)}{a^2} \right)^{-1} \tag{4.3}$$

donde a es una constante fija y la d una métrica de distancia entre los valores de salida x_i, que para este caso es $d^2(x_j, x_i) = (x_i - x_j)^2$.

Se define el valor S como

$$S = \sum_{i=1}^{n} w_i \tag{4.4}$$

obteniendo la nueva salida x como:

$$x = \sum_{i=1}^{n} \left(\frac{w_i}{S} \right) x_i \tag{4.5}$$

Así, el peso asignado a la *i-ésima* salida x_i es inversamente proporcional a la distancia entre la x_i y cada una de las otras salidas. La cantidad S se usa simplemente para normalizar las cantidades w_i, de tal forma que el valor de su suma sea 1.

El caso de estudio se implanta como un programa de cómputo que utiliza cuatro nodos: tres pertenecientes a los sensores, y uno al algoritmo de la votación. Cada nodo tiene asignada una tarea que se ejecuta cada periodo de tiempo. El esquema de comunicación entre los nodos se muestra en la Fig. 4.1.

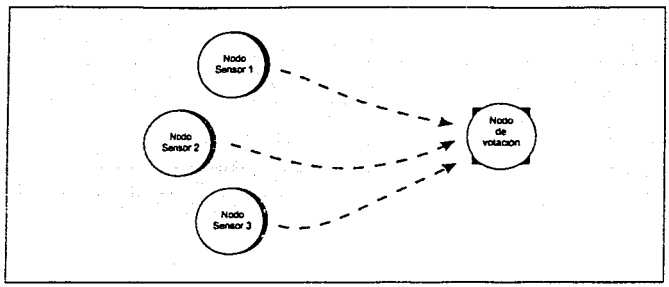


Fig. 4.1: Votación con tres sensores

Cada nodo tiene asociada una tarea, dependiendo si representa un sensor o al algoritmo de votación. Cada tarea tiene asociado un periodo y un tiempo de ejecución. Si una tarea realiza la labor de censado, ésta se encarga de leer y mandar la lectura simulada en software al algoritmo de votación cada periodo de tiempo P_s consumiendo para ello en proceso y comunicación un tiempo C_s. Por otro lado, la tarea que realiza la labor del algoritmo de votación, se encarga de recibir y almacenar la información de los sensores, y cada periodo de tiempo P_{av} procesa la información e informa del resultado de las salidas, utilizando para ello el algoritmo de pesos promedio con un consumo de tiempo C_{av}. Una ejecución típica del caso de estudio con tres sensores se muestra en la Fig. 4.2.

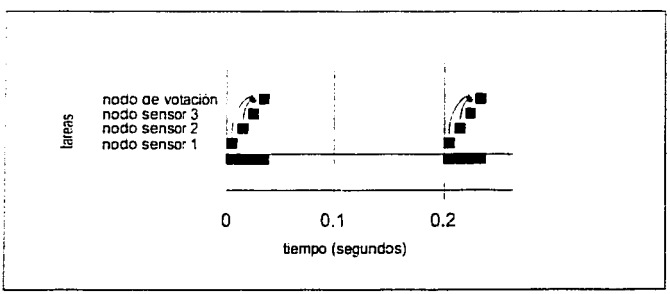


Fig. 4.2: Ejecución de tareas en el tiempo del caso de estudio

Para evaluar el rendimiento de la aplicación, los nodos sensores utilizan como entrada una función seno trasladada verticalmente una unidad, de tal forma que sólo se

obtengan valores positivos, durante el intervalo de tiempo de 0 a 2π segundos. En cuanto al orden de asignación de prioridades, los nodos sensores tienen las prioridades más altas, por lo que se ejecutan primero. De esa forma, el algoritmo de votación se ejecuta con los parámetros de la tarea con mayor o igual periodo que las demás.

Los i sensores simulados en software mandan al final de su tiempo de ejecución t_{iC} el valor $seno\,(t_{iC})$ al nodo de votación. De acuerdo a la notación del algoritmo de pesos promedio, el valor del seno representa al valor x_i. Después de recibir la información de los sensores, el nodo de votación calcula el valor del seno al final de su tiempo de ejecución t_{avC}. Este último nodo calcula el valor de votación de la información de los sensores.

Fuera de línea, éste nodo hace una comparación entre el valor producido por el algoritmo de votación y el valor real del seno en los tiempos previstos para producir una medición de error (Fig. 4.3).

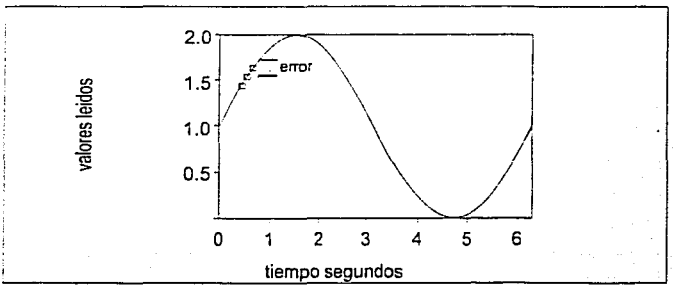


Fig. 4.3: Error del valor x obtenido de la votación marcado por una cruz

4.2 Implantación del programa simulador

Antes de establecer la descripción del programa simulador, es necesario hacer notar que dicho programa utiliza el método propuesto en el capítulo 3, tanto en línea como fuera de línea (Fig. 3.2).

Como se describe en la sección 2.1, un nodo de un sistema de cómputo distribuido en tiempo real tiene dos partes: el núcleo de la computadora y la interfaz de comunicación. Para que una computadora funcione adecuadamente como nodo, es necesario contar con un *sistema operativo de tiempo real*, para manejar las operaciones y las comunicaciones.

Para la implantación del programa simulador se usa al sistema operativo RT-Linux sobre una plataforma Pentium III. RT-Linux es un sistema operativo duro, coexis-

tente con el sistema operativo Linux, que permite crear hilos (*threads*) de tiempo real POSIX.1b, los cuales se ejecutan en instantes precisos y determinados en el tiempo. RT-Linux a sido objeto de análisis y desarrollo por parte de la industria y la comunidad académica [Mendoza *et al.*(2001)], [Alonso *et al.*(2001)], en donde ha sido utilizado en sistemas embebidos y de control de tiempo real computarizado. (véase el apéndice A).

Para entender la implantación del simulador, se debe hacer notar que las tareas ejecutadas en el núcleo de RT-Linux no se desarrollan como aplicaciones no predictivas. En su lugar, las tareas son implantadas como módulos, los cuales se almacenan en el espacio de memoria del núcleo de Linux. Los módulos del núcleo son cargados o descargados dinámicamente en memoria. Para el manejo de los recursos y prioridades, RT-Linux implementa una capa virtual entre las aplicaciones y el hardware. De esta forma el planificador de RT-Linux puede interceptar las llamadas de hardware de las aplicaciones asignando la máxima prioridad a aquellas de tiempo real, (Fig. 4.4).

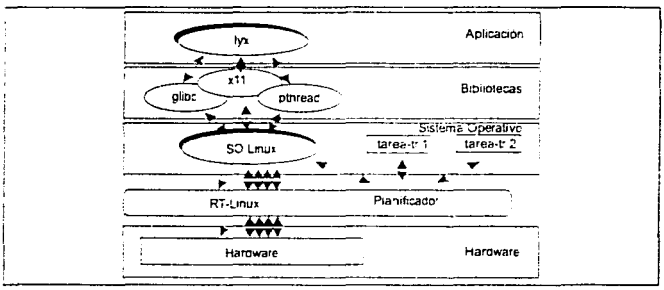


Fig. 4.4: Funcionamiento de RT-Linux

El sistema de cómputo de tiempo real tiene tres secciones: el operador, el objeto controlado y el sistema de cómputo de tiempo real. Así la aplicación tiene un programa consola que sirve de interfaz entre el operador y el sistema de cómputo. Este último programa permite al operador inicializar, poner en marcha, y desactivar los hilos que indican el tiempo de ejecución de los programas que representan a los componentes del sistema. Las funciones correspondientes al sistema de cómputo de tiempo real se realizan en el núcleo del sistema operativo RT-Linux, como tareas de tiempo real. Estas corresponden a la recepción de la lista de configuración, cálculo del plan de acuerdo a esa lista e inicialización, y puesta en marcha de los hilos de cada tarea. Finalmente, el objeto controlado (caso de estudio) es simulado en software, mediante nodos que representan a los sensores y al algoritmo de votación. Los nodos se ejecutan como tareas normales de Linux con restricciones en el tiempo.

La implantación del simulador para el caso de estudio (Fig. 4.1) se lleva a cabo como se muestra en la Fig. 4.7.

El programa simulador funciona de la siguiente forma:

1. Se almacena en memoria el planificador y los *disparadores* de los nodos (sensores y de votación), como parte del *núcleo* del sistema operativo. Se activan, tanto el planificador como los hilos de los nodos, los cuales permanecen en espera de algún comando de ejecución.

   (a) El planificador se ejecuta como hilo. Este programa espera las peticiones del usuario para iniciar la simulación. Inicializa los hilos de los sensores y de votación, se encarga de hacer el plan, y lo distribuye durante cada ventana de tiempo. En la Fig. 4.5 este nodo se denota con la etiqueta planificador.

   (b) El planificador crea el canal de comunicación con la consola, mediante un hilo que recibe los datos y comandos de esta. En la Fig. 4.5 este canal se denota por FIFO-consola.

   (c) Se crean los hilos que sirven como *disparadores* en los diferentes nodos. Estos hilos son controlados por el planificador, y su función es esencialmente enviar una señal, de acuerdo al plan elaborado, a los nodos en el sistema. Estos hilos se encuentran en la parte superior de la Fig. 4.5.

   (d) Cada uno de los hilos de los nodos crea su propio canal de comunicación con su respectiva aplicación. En la Fig. 4.5, se denotan como FIFO-sensor1, FIFO-sensor2, FIFO-sensor3 y FIFO-votación.

   (e) El planificador espera los comandos de la consola.

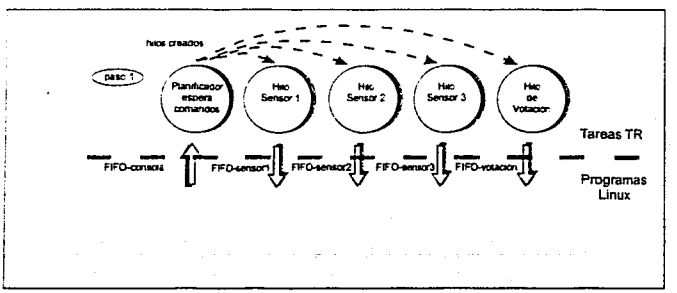


Fig. 4.5: Activación del planificador y los hilos de la aplicación

2. Se ejecutan las aplicaciones que simulan los diferentes nodos (sensores y de votación) Fig. 4.6. En la Fig. 4.7, las aplicaciones están dibujadas en la parte inferior.

   Cada aplicación hace lo siguiente:

   (a) Inicializa su canal de comunicación con su hilo respectivo. Cada hilo, a través de este canal, envía las señales de cuándo procesar y cuándo distribuir la información a los nodos correspondientes.

   (b) Crea e inicializa los canales de comunicación con los demás nodos, tanto de transmisión como de recepción, utilizando *sockets* sobre el protocolo Ethernet. Estos canales de comunicación transportan los valores del $seno(t)$ provenientes de los nodos sensores que servirán de entrada al nodo de votación.

   En la Fig. 4.6, se muestran en forma esquemática los canales de comunicación entre los hilos y sus respectivas aplicaciones (a través de FIFO's RT-Linux).

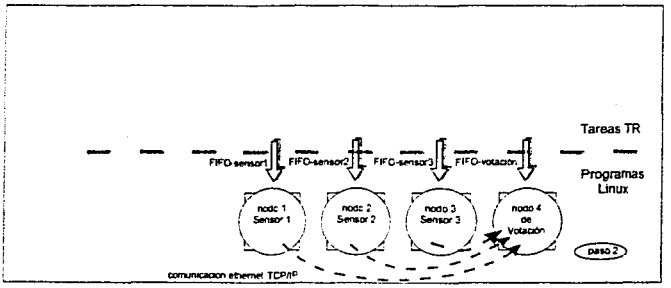


Fig. 4.6: Inicialización de los canales de comunicación entre hilos-aplicaciones y aplicaciones-aplicaciones

3. Se ejecuta la aplicación de la consola, esperando los comandos por parte del operador. La consola es capaz de responder asíncronamente a cualquier evento con respecto a los comandos y datos correspondientes de cada tarea, es decir, los valores $C's$ y $P's$, así como los comandos de inicio, suspensión y detención. El comando inicio hace que los hilos disparadores envíen la señal a sus aplicaciones correspondientes. El comando suspensión hace que los hilos disparadores suspendan la transmisión de la señal a sus aplicaciones correspondientes. El comando detención hace que se suspenda la ejecución de los hilos disparadores.

   La consola del usuario, se ubica en la parte inferior izquierda de la Fig. 4.7, que se utiliza para inicializar la ejecución del simulador del caso de estudio, llevando a cabo el siguiente procedimiento:

(a) Inicializa el canal de comunicación con el planificador;

(b) esperar los comandos por parte del operador;

(c) recibe los parámetros iniciales de cada tarea que el operador provee; y

(d) envía la señal de la inicialización del funcionamiento del sistema.

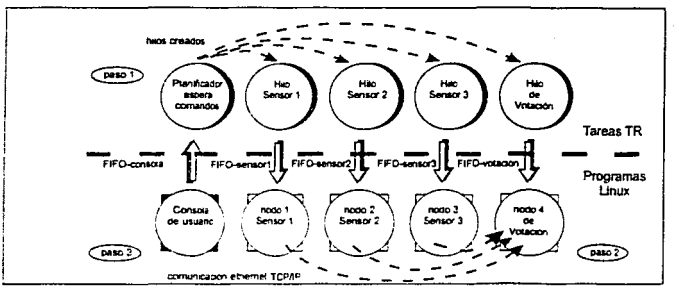


Fig. 4.7: Nodos en RT-Linux

A partir de esto, el planificador hace la prueba de planificación, crea el plan de acuerdo con el algoritmo de ordenamiento por planes (esto durante cada ventana de tiempo), y finalmente programa los hilos de los sensores y de votación con los tiempos adecuados.

Normalmente, la implantación del caso de estudio se realiza en una red de cómputo, distribuyendo uniformemente los nodos en cada computadora. Sin embargo, la simulación se realiza en un solo equipo de cómputo, con el objeto de que los retrasos atribuidos a la red sean uniformes. Si los nodos se ejecutaran en computadoras diferentes, implicaría el uso de una medición más detallada de los tiempos de comunicación, así como de una sincronización en el manejo de relojes entre las computadoras, para obtener resultados adecuados.

Una salida típica de la simulación del caso de estudio utilizando los datos de la tabla 4.1 se muestra en la Fig. 4.8. El valor del ciclo elemental (CE) es de 0.063 segundos por lo que se puede obtener hasta un 100 medidas en el intervalo de tiempo de 0 a 2π segundos.

Tarea	Periodo en CE's	Tiempo de proceso en CE's
Sensor 1	4	0.2
Sensor 2	5	0.2
Sensor 3	7	0.2
Votación	8	0.1

Tabla 4.1: Parámetros de las tareas a planificar

Recuérdese que para simplificar la notación de la tabla 4.1, la información se escribe en forma de lista como:

$$[P_1, P_2, P_3, P_4, C_1, C_2, C_3, C_4] \tag{4.6}$$

Así, la información de la tabla 4.1 se puede escribir como la siguiente lista de configuración:

$$[4, 5, 7, 8, 0.2, 0.2, 0.2, 0.1] \tag{4.7}$$

La lista de configuración 4.7 es planificable de acuerdo al algoritmo de ordenación de Almeida. Esta tiene una ventana de tiempo igual a 24 CE's y en este caso el valor del CE es de 0.063 segundos.

Para que funcione adecuadamente el algoritmo de votación de pesos promedio es necesario obtener con anterioridad las mediciones de los nodos sensores. De acuerdo a este requerimiento, se hizo que el nodo de votación obtuviera la mínima prioridad. Nótese entonces que la política de asignación de tareas es importante y que también es dependiente de la aplicación.

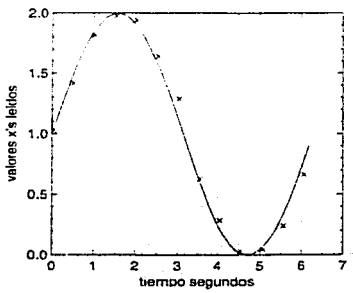


Fig. 4.8: Gráfica de los *valores* x obtenidos por votación marcados con una cruz y la gráfica seno

4.3 Resultados

En esta sección se presentan los resultados que se obtuvieron de las simulaciones del caso de estudio al variar algunos parámetros de la aplicación.

Para hacer las simulaciones se propone utilizar tres sensores. De esa forma, el caso de estudio tendrá cuatro nodos: tres que representan a los sensores, y el último al algoritmo de votación. Si se asume que cada nodo tiene asociada una tarea, se infiere entonces que hay cuatro tareas a planificar. Cada tarea se ejecuta periódicamente (cada tiempo P) y tiene asociado un tiempo fijo de proceso (cada tiempo C).

Para realizar las mediciones de los sensores, se propone que estos lean los valores de una función seno en el intervalo de tiempo de 0 a 2π segundos. Los sensores son simulados por un programa de software.

4.3.1 Resultados fuera de línea

De acuerdo al método de obtención de parámetros adecuados (capítulo 3), en su etapa fuera de línea, se realizan los siguientes pasos para obtener la tabla de parámetros adecuados:

1. *Obtención de la tabla de posibles valores.* Se establece que el valor máximo de los periodos de las tareas que se propone para la ejecución del caso de estudio es 10 CE's, y la resolución del tiempo de proceso es 0.1 CE. De esta forma, los valores posibles de los periodos de las tareas son: $1, 2, 3, \ldots, 10$ CE's, y los valores de los tiempos de ejecución de las tareas comprenden la secuencia $0.1, 0.2, 0.3, \ldots, 1$ CE.

   Si se tienen cuatro tareas, donde los valores de los periodos de las tareas tienen los valores de 1 a 10, se obtiene que el número de combinaciones es de $10 \times 10 \times 10 \times 10 = 10^4$ combinaciones. Debido a que los valores de los tiempos de ejecución también tienen 10 valores diferentes, se obtiene entonces 10^4 combinaciones en los C's. Por lo tanto hay 10^8 combinaciones a planificar.

2. *Filtrado de listas de configuración planificables con base al algoritmo de ordenamiento por planes.* En este paso, primero se deben eliminar las listas de configuración que se consideran repetidas y, una vez obtenidas, discriminarlas de acuerdo a su planificabilidad.

   En este caso, el número de combinaciones sin repetición de los valores de los periodos de las tareas tomando en cuenta cuatro posiciones es de 715. El número combinaciones de diez valores en cuatro posiciones que se obtiene de las combinaciones de los tiempos de ejecución es 10^4. Por lo tanto, el número de listas de configuración resulta ser $715 \times 10^4 = 7.150.000$ combinaciones. Al conjuntar las combinaciones de los periodos y de los tiempos de ejecución, el número de listas de configuración se reduce ya que existen repeticiones en los valores del periodo o de los tiempos de ejecución. Tomando en cuenta lo

anterior, el número final de listas de configuración sin repetición que se obtienen es 4,421,276 combinaciones.

Si ahora se discriminan las listas de configuración que son planificables de acuerdo al planificador de Almeida, se obtienen 619,416 listas de configuración. Para hacer las correspondientes pruebas de planificabilidad, se propone un valor para la ventana de tiempo de 24 CE's.

De esa forma, con el método propuesto y al hacer las simulaciones de la aplicación, sólo es necesario tomar en cuenta el 0.62% del total de las listas de configuración.

3. *Simulación en el sistema de cómputo de tiempo real.* A fin de obtener los parámetros adecuados de las tareas se hace una simulación del caso de estudio. Una vez obtenida la *tabla de parámetros adecuada* se propone hacer dos simulaciones que muestren el efecto en la variación del CE.

   Para realizar la simulación es necesario conocer cuáles son los requerimientos del sistema. En el caso de estudio propuesto, se define la siguiente medida de rendimiento:

   Se considera una *lectura adecuada* de votación x en los valores de los sensores, si la diferencia entre el valor de la lectura x_s de la función seno y la x_{al} obtenida del algoritmo de votación es menor en un cinco por ciento. Este error es obtenido cada vez que se ejecuta la tarea del nodo de votación.

   Puesto que el algoritmo de votación se ejecuta un número de veces durante el periodo de ejecución definido (dependiendo de la lista de configuración que se este probando), se consideran adecuadas aquellas listas de configuración que tengan un 85% de lecturas adecuadas con respecto al número de veces que se haya ejecutado el algoritmo durante el intervalo de tiempo de 0 a 2π segundos.

   Con todos los parámetros definidos de la aplicación, se hacen dos simulaciones para obtener los valores de las listas de configuración que se consideran adecuados.

   (a) Primera simulación. En esta se toma el valor del CE igual a 0.063 segundos. Con dicha resolución del ciclo elemental en un intervalo de tiempo de 0 a 2π unidades de tiempo, las tareas que tienen por periodo al CE se ejecutan 100 veces. Los resultados que se obtuvieron al redondear los porcentajes de lecturas adecuadas de las listas de configuración sin repetición y planificables son los mostrados en la tabla 4.2.

Porcentaje de lecturas adecuadas	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Número de listas de configuración	31	10,774	147,657	197,241	134,415	88,479	29,856	6,965	3,885	112	0

Tabla 4.2: Población de listas de configuración de acuerdo al porcentaje de lecturas adecuadas de la simulación 1

Los datos del número de listas de configuración de acuerdo a su porcentaje de lecturas adecuadas se pueden graficar para ver el comportamiento de la población. En la Fig. 4.9 se muestran los datos de la tabla 4.2. Se observa que la mayor población tiene entre un 20% y 50% de lecturas adecuadas, y al aumentar este porcentaje, la población disminuye.

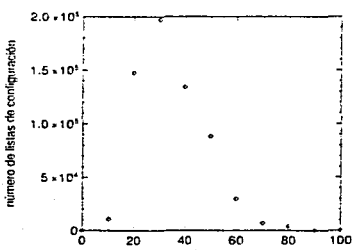


Fig. 4.9: Gráfica del número de listas de configuración de acuerdo al porcentaje de lecturas adecuadas en la simulación 1

Con los valores de la tabla 4.2 se obtiene que solo 112 listas de configuración son adecuadas. Estas corresponden a las listas de configuración que tienen un periodo mínimo como la lista de configuración [1,1,1,3, 0.1,0.1,0.1,0.1]. Conforme los valores de los periodos de las tareas se hacen cada vez más grandes, las diferencias entre el valor real de la función y el valor obtenido del algoritmo de votación se hacen mayores, por lo que solo pocas combinaciones son adecuadas.

Una lectura utilizando los parámetros [2, 4, 7, 10, 0.4, 0.1, 0.4, 0.1] se muestra en la Fig. 4.10.

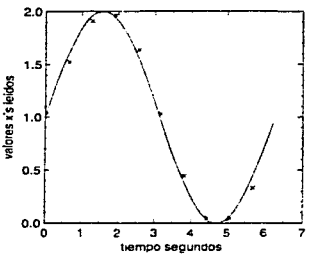


Fig. 4.10: Gráfica de los valores x obtenidos de la lista de configuración $[2, 4, 7, 10, 0.4, 0.1, 0.4, 0.1]$ utilizando los parámetros de la simulación 1

(b) Segunda simulación. En esta, el valor de CE es de 0.013 segundos. En un intervalo de tiempo de 0 a 2π unidades de tiempo, las tareas que tienen por periodo un CE se ejecutan 500 veces.

Los resultados que se obtuvieron al redondear los porcentajes de lecturas adecuadas de las listas de configuración sin repetición y planificables, se presentan en la tabla 4.3.

Porcentaje de lecturas adecuadas	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Número de listas de configuración	0	0	0	0	3	4,257	87,085	243,043	211,101	72,030	1,898

Tabla 4.3: Población de listas de configuración de acuerdo al porcentaje de lecturas adecuadas de la simulación 2

En la Fig. 4.11 se grafican los datos de la tabla 4.3. Se observa que al disminuir el valor del CE (y por lo tanto los periodos y los tiempos de ejecución), el porcentaje de lecturas adecuadas de las listas de configuración aumenta. De hecho no hay combinaciones que tengan un porcentaje significativo menor al 50%.

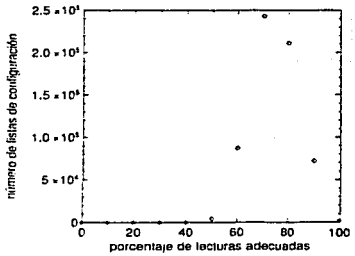


Fig. 4.11: Gráfica del número de listas de configuración de acuerdo al porcentaje de lecturas adecuadas en la simulación 2

Con los valores de la tabla 4.3 se obtiene que 73,928 listas de configuración son adecuadas que representa el 11.93% del total. Para mostrar el comportamiento de una simulación, en la Fig. 4.12 se grafican las lecturas de una prueba utilizando los parámetros [4, 4, 5, 7, 0.3, 0.3, 0.4, 0.4] .

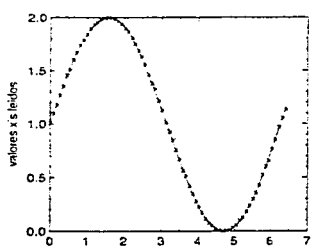


Fig. 4.12: Gráfica de los valores x obtenidos de la lista de configuración [4, 4, 5, 7, 0.3, 0.3, 0.4, 0.4] utilizando los parámetros de la simulación 2

Nótese que en ambas simulaciones los dos grupos de valores de todos los parámetros son los mismos (tanto los periodos como los tiempos de ejecución), a excepción del valor del ciclo elemental. En ambas simulaciones los planes hechos por el planificador son idénticos. Esto se debe a que los tiempos de la planificación (periodos y tiempos de proceso) se consideran en términos del CE, independientemente de la duración de éste. Esto muestra una ventaja adicional del método: un mismo cálculo en la planificación puede ser utilizado variando únicamente el valor del CE.

4.3.2 Resultados en línea

Finalmente se debe probar el método en línea bajo el esquema de reconfiguración. Para presentar los resultados se toman en cuenta dos escenarios:

- En el primer escenario se supone que no hay ningún cambio en la lista de configuración en el tiempo. Ello puede ocurrir cuando el sistema toma una lista de configuración y no ocurren cambios en el sistema, o cuando se quieren planificar listas de configuración no adecuadas y por lo tanto se mantiene la lista de configuración anterior.
- El segundo escenario ocurre cuando constantemente cambia la configuración del sistema por lo que en cada ventana de tiempo cambian las listas de configuración adecuadas.

1. Resultados en línea de la primera simulación.

   (a) Los resultados que se obtienen de la primera simulación cuando no ocurre ningún cambio en el sistema de cómputo de tiempo real se muestran en la Fig. 4.13. En esta figura se muestra los porcentajes de lecturas adecuadas en el periodo de 0 a 2π de las 112 listas de configuración adecuadas que se obtuvieron en la primera simulación. Como lo muestra la Fig. 4.13 todas las listas de configuración tienen un porcentaje mínimo de un 85% de lecturas adecuadas, por lo que todas las listas de configuración funcionan bien en el sistema de cómputo de tiempo real.
   De ésta prueba, se infiere que repitiendo el ciclo una y otra vez los resultados obtenidos son los mismos ya que las condiciones no varían en el tiempo y por que al comienzo de la ejecución de la aplicación se ha comprobado la viabilidad de funcionamiento de la aplicación.

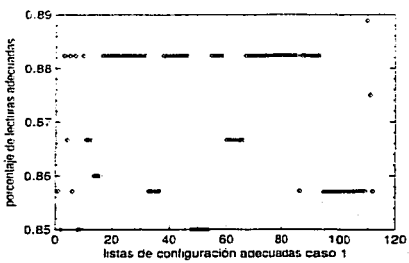


Fig. 4.13: Porcentajes de lecturas adecuadas de las 112 listas de configuración de la primera simulación sin cambios de configuración

(b) Para comprobar la prueba de la primera simulación cuando se cambian al azar las diferentes listas de configuración adecuadas, a continuación se muestra el resultado de 100 simulaciones con una duración de 100 ventanas de tiempo de 24 CE's (Fig. 4.14). La gráfica muestra que en todas las simulaciones los valores del porcentaje de lecturas adecuadas siempre es mayor o igual al 85% propuesto como parámetro de desempeño de la aplicación. También en este caso las listas de configuración funcionan en el sistema de cómputo de tiempo real adecuadamente.

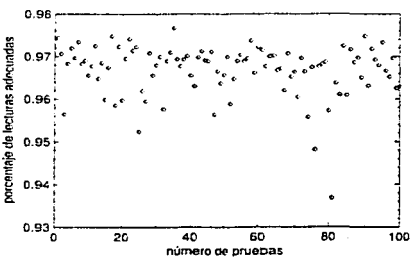


Fig. 4.14: Porcentajes de lecturas adecuadas de 100 pruebas primera simulación utilizando los parámetros de las 112 listas de configuración cambiadas al azar cada ventana de tiempo

2. Resultados en línea de la segunda simulación.

(a) En la Fig. 4.15 se muestra los porcentajes de lecturas adecuadas de las 73,928 listas de configuración adecuadas que se obtuvieron en la segunda simulación cuando no ocurre ningún cambio en el sistema de cómputo de tiempo real. Como en el caso de la primera simulación todas las listas de configuración tienen un porcentaje de lecturas adecuadas de al menos un 85% cumpliendo así que todas las listas de configuración funcionan en el sistema de cómputo de tiempo real adecuadamente.

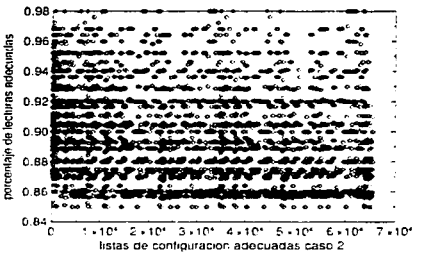


Fig. 4.15: Porcentajes de lecturas adecuadas de las 73,928 listas de configuración de la segunda simulación sin cambios de configuración

(b) El resultado de 100 simulaciones con una duración de 100 ventanas de tiempo de 24 CE's que se realiza para comprobar la prueba de la segunda simulación cuando se cambian al azar las diferentes listas de configuración adecuadas se muestra en la Fig. 4.16. La gráfica muestra que en todas las simulaciones los valores del porcentaje de lecturas adecuadas cumplen con el 85% propuesto como parámetro de desempeño de la aplicación. También en este caso las listas de configuración funcionan en el sistema de cómputo de tiempo real adecuadamente.

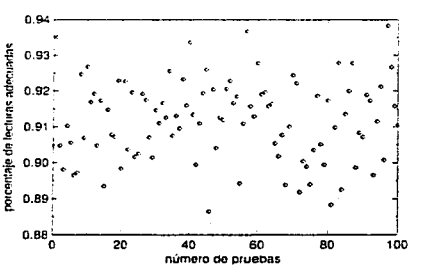


Fig. 4.16: Porcentajes de lecturas adecuadas de 100 pruebas primera simulación utilizando los parámetros de las 73,928 listas de configuración cambiadas al azar cada ventana de tiempo

De esta forma, se prueba satisfactoriamente el método propuesto para obtener parámetros adecuados de listas de configuración.

4.4 Resumen

En este capítulo se implanta un caso de estudio para probar la técnica de obtención de parámetros adecuados. El caso de estudio propuesto utiliza al algoritmo de votación de pesos promedio para evaluar el método propuesto. En la implantación de la aplicación se definen algunos requerimientos que se plantean en el método de manera general.

El caso de estudio es una primera aproximación de la aplicación del método propuesto, y se espera que en trabajos futuros se utilicen otras aplicaciones con requerimientos de rendimiento más elaboradas, donde se pueda medir la eficacia del método propuesto haciendo una contrastación con métodos semejantes, o con aproximaciones que tengan objetivos similares.

Capítulo 5

Conclusiones

El diseño de los sistemas de cómputo de tiempo real no solo debe incorporar los medios para resolver su complejidad funcional, también debe analizar y predecir las propiedades temporales de tiempo real. Sin embargo, en muchos métodos de diseño de las aplicaciones de tiempo real no se provee el adecuado soporte para obtener los rendimientos de las especificaciones del tiempo real. Durante el diseño de una aplicación de tiempo real intervienen muchos parámetros que pueden ser variados para obtener un funcionamiento adecuado, con lo que el problema de diseñar una aplicación resulta complejo. El método propuesto se presenta para encontrar los parámetros adecuados de la tareas como su periodo y tiempo de ejecución, de tal forma que la aplicación de tiempo real satisfaga a las especificaciones de rendimiento de la aplicación

En esta tesis, se hace un análisis del algoritmo planificador por planes para la reconfiguración en línea de un sistema de cómputo de tiempo real, y se establece un método para encontrar una ordenación de tareas como solución adecuada en el funcionamiento de aplicaciones de tiempo real. El método, establecido de manera heurística evalúa al algoritmo de ordenamiento por planes con base a restricciones temporales y a los requerimientos de rendimiento de la aplicación. El método se usa para el análisis de la planificabilidad, utilizando el algoritmo de ordenamiento por planes con el fin de establecer el sub-grupo de soluciones posibles.

Con el metodo propuesto se estudia el impacto que tiene el algoritmo de ordenación por planes sobre un sistema distribuido reconfigurable, con ello se resuelve el problema de determinar *a priori* la planificación adecuada de la reconfiguración en línea de un sistema de computo de tiempo real, con base a su planificación y a la aplicación en particular. De hecho, el análisis de la planificación permite definir la viabilidad de la reconfiguración de las tareas en un sistema distribuido. Sin embargo, es posible que la respuesta de dicho sistema no sea adecuada. Por esto se requiere tomar en cuenta tanto los requerimientos de la planificabilidad como aquellos requerimientos particulares del caso de estudio, a fin de determinar si es o no factible dicha reconfiguración.

Como se ha expuesto, la planificación es un problema *np-completo*, lo que implica el manejo de una cantidad enorme de listas de configuración. Sin embargo, y de acuerdo al método propuesto, éste reduce en un gran porcentaje el número de listas de configuración a analizar. En el caso de cuatro tareas, como se muestra en el caso de estudio, el número de casos posibles a verificar si es adecuado funcionamiento se reduce de 10^8 a 4,421,276 listas de configuración. Este resultado representa el 4.42% del total de combinaciones. Con ello se demuestra la eficacia del método en cuanto a la eficacia en la reducción del espacio de combinaciones posibles.

De esa forma el presente trabajo presenta y desarrolla un método heurístico para garantizar el desempeño adecuado de un sistema de cómputo distribuido, reconfigurable, de tiempo real, utilizando el algoritmo de ordenamiento por planes.

A pesar e que el método resulta adecuado en la obtención de parámetros de las tareas, éste presenta algunas limitantes, sin embargo estas pueden minimizarse como se describe en los siguientes párrafos.

Dependencia de los requerimientos de la aplicación. La solución propuesta está restringida bajo ciertos esquemas de reconfiguración en los requerimientos de la aplicación. Si cambian los requerimientos en el valor máximo de los valores del periodo, la granularidad de los tiempos de proceso, el valor de la ventana de tiempo, o al cambiar el número de tareas a planificar, implica recalcular la tabla de parámetros adecuados. Sin embargo, en ciertas circunstancias los resultados que se obtengan pueden ser utiles con los nuevos cambios de los requerimientos. Por ejemplo, si se calcula una tabla de parámetros adecuados con un valor máximo del periodo igual a 5, y después se requiere un valor máximo de 10, sólo es necesario calcular las combinaciones adecuadas que tengan un periodo mayor a 5.

Dependencia del programa simulador. Una limitante del método propuesto es la necesidad de realizar un programa simulador de la aplicación, lo que no siempre es posible. Además, aún después de construir el programa simulador, su ejecución puede requerir de una gran capacidad de cómputo, por lo que la ejecución del número de casos posibles puede requerir una gran cantidad de tiempo.

Con respecto al problema de obtener un número de listas de configuración muy grande, también se puede hacer pruebas de obtención de parámetros adecuados sobre intervalos de valores de los periodos y los tiempos de ejecución. Por ejemplo, se pueden calcular listas de parámetros adecuados en intervalos de valores de periodos de 1 a 25, 100 a 125 y de 225 a 250 en vez de hacerlo de 1 a 250.

De acuerdo los argumentos descritos anteriormente y de los resultados del caso de estudio se concluye que el método para obtener parámetros adecuados en las tareas de un sistema de cómputo distribuido, reconfigurable, de tiempo real, utilizando el algoritmo de ordenamiento por planes resulta un método viable para utilizarse en otras aplicaciones.

5.1 Trabajos futuros

Parte importante en el desarrollo de este trabajo es la búsqueda de la métrica para determinar cuando un plan es adecuado o no, la presente tesis propone dicha evaluación con base al caso de estudio. Sin embargo, se requiere determinar métricas más generales que permitan englobar la concepción de *"adecuado"* de acuerdo a los requerimientos de la aplicación y el tiempo real.

Como se ha expuesto, la planificación es un problema *np-completo*, debido a que si el valor del periodo es muy grande, la resolución en los tiempos de ejecución aumenta o el número de tareas se incrementa, lo que implica el manejo de una cantidad enorme de listas de configuración. Sin embargo, y de acuerdo al método propuesto, nótese que el cálculo de si una lista de configuración es o no adecuada no depende del cálculo de las otras listas, por lo que el método puede ser paralelizable. Además, la paralelización no sólo se puede implementar en el cálculo de la tabla de parámetros adecuados, sino también se puede implementar al calcular las listas de configuración no repetidas. De esa forma en un trabajo futuro se puede incluir las versiones de los programas de obtención de parámetros que trabajen en forma paralela.

Además de proponer una reducción en el espacio de soluciones dadas por un planificador con base al análisis de la planificación, este trabajo también pretende ser una base para trabajos posteriores en la definición de métodos eficientes para algoritmos de ordenamiento para determinar el impacto de la planificación en sistemas de cómputo de tiempo real.

Glosario

Algoritmo de planificación. (scheduler algorithm) Un algoritmo de planificación o algoritmo de ordenamiento es un conjunto de reglas que determinan que *tarea* debe ser ejecutada en un momento particular.

Ciclo elemental. (elementary cycle) El ciclo elemental CE. es un lapso de tiempo que sirve para dividir al macrociclo.

Comprobación. (test) Una comprobación es un medio por el cual la existencia y calidad de ciertos atributos dentro de un sistema son determinados. Dentro del tiempo real es el análisis *a priori* para saber si el sistema funcionará adecuadamente. Este tipo de análisis se llama *comprobación* o *prueba de planificabilidad.*

Conjunto de tareas planificables. (schedulable task set) Un conjunto de *tareas* es planificable bajo un algoritmo de ordenación, si al utilizar el algoritmo sobre el conjunto de *tareas* ninguna de ellas pierde sus *plazos límites*.

Disparo. (trigger) Es un evento que causa el comienzo de una acción.

Evento. (event) Cualquier suceso que ocurra en el tiempo.

Factor de utilización. (utilization factor) Se define como *factor de utilización* (en el procesador) la fracción del procesador que se invierte en la ejecución de un conjunto de *tareas*.

Macrociclo. (macrocycle) El macrociclo MC (en un sistema de cómputo de tiempo real) es el mínimo común múltiplo de los periodos de todas las *tareas*. Se utiliza en los planificadores estáticos para obtener los límites del plan a obtener.

Operación (operation) Es una acción caracterizada por un tiempo de proceso y el lugar en donde se va a realizar. El lugar denomina a una máquina.

Planificabilidad de un conjunto de tareas. (:) Véase *conjunto de tareas planificables* en este glosario.

Planificación (scheduling) La planificación concierne la asignación de recursos y tiempo a las *tareas* de tal forma que los requerimientos de rendimiento sean obtenidos.

Planificador dinámico. (dynamic scheduler) Un planificador es llamado dinámico (o en línea) si éste hace decisiones de planificación en tiempo de ejecución, seleccionando una de las *tareas* listas a ejecutarse.

Planificador estático. (static scheduler) Un planificador se llama estático si éste hace las decisiones de planificación en tiempo de compilación.

Plazo límite. (deadline) Es el tiempo en el cual un resultado debe de ser producido.

Plazo límite suave. (deadline soft) Si un resultado tiene utilidad aún después de que un *plazo límite* de una tarea se haya cumplido, el plazo es denominado suave.

Plazo límite firme. (deadline firm) Si un resultado ya no tiene utilidad después de que un *plazo límite* de una tarea se haya cumplido, el plazo es denominado firme.

Plazo límite duro. (deadline hard) Un plazo límite se le llama duro (hard), si ocurre una catástrofe cuando la tarea asociada a él no se realiza a tiempo.

Protección. (safety) Un sistema de cómputo se dice que da protección si un mal funcionamiento de éste no produce catástrofes humanas o daños en el medio ambiente.

Seguridad. (security) Previene el acceso no autorizado al sistema y a la información.

Sistema distribuido. (distributed system) Un sistema distribuido es un sistema de cómputo cuyos componentes ya sea de hardware o software están localizados en computadoras en red que se comunican coordinando sus acciones sólo por paso de mensajes.

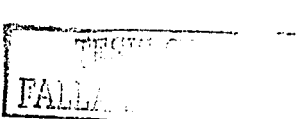
Sistema de cómputo de tiempo real. (real-time system) Es un sistema de cómputo en el cual el correcto funcionamiento del sistema no solo depende de los resultados lógicos de las operaciones, sino también depende del instante físico en el cual estos resultados son producidos.

Taller (workshop) Es el conjunto de máquinas, junto con el espacio físico necesario para realizar cierto proceso.

Tarea. (task) Un trabajo (job) o tarea consiste de un conjunto de *operaciones*.

Trabajo. (job) Un trabajo o tarea (task) consiste de un conjunto de *operaciones*.

Ventana de tiempo. (time window) La ventana de tiempo es un lapso de tiempo que divide al macrociclo. Es un múltiplo del ciclo elemental.



Apéndice A

RT-Linux.

Un sistema de cómputo de tiempo real, puede ser definido como un sistema que realiza sus funciones internas y responde asíncronamente a eventos externos dentro de un tiempo específico. Muchas de las aplicaciones de control y adquisición de datos, caen dentro de esta categoría. Un sistema operativo de tiempo real debe ser capaz de garantizar los requerimientos de los procesos bajo control.

Mientras que los sistemas operativos de tiempo compartido como Unix, enfocan sus esfuerzos en mantener un rendimiento adecuado promedio. Es decir, que el sistema operativo tratará de servir de igual forma a todas las tareas que se están ejecutando en el sistema.

Dentro de los sistemas operativos se puede hacer la clasificación de los que son duros y los que son suaves. Los sistemas operativos suaves son aquellos en los cuales los requerimientos temporales son estadísticamente definidos. Un ejemplo puede ser un sistema de vídeo conferencia. En cambio, en los sistemas de cómputo de tiempo real duro, los plazos límites deben ser garantizados. Un ejemplo, es el controlador de vuelo que regule la altitud de un avión.

Actualmente, existen algunos sistemas operativos de tiempo real. Sin embargo, la mayoría de ellos no son abiertos, estándares, eficientes y baratos. Una solución a tal problema es el sistema operativo de tiempo real RT-Linux. Este sistema operativo usa como base al sistema operativo Linux.

Linux es un sistema operativo tipo Unix, creado por Linus Torvals, que tiene una alta estabilidad, eficiencia, disponibilidad de código y licencia no restrictiva, entre otros atributos. Una característica esencial en cualquier aplicación de cómputo, es la disponibilidad de código, ya que ella permite hacer la verificación del correcto funcionamiento en el sistema y contar además con la capacidad de adaptación según las necesidades.

Linux posee todas las características de un sistema operativo moderno tipo Unix: interfaz gráfica, interconexión bajo muchos protocolos de comunicación, bases de datos, lenguajes de programación, etc. Linux, sin embargo, no tiene todas las características necesarias para ser un sistema operativo de tiempo real. Sobre todo porque

las interrupciones son deshabilitadas frecuentemente durante el funcionamiento del núcleo. otros problemas incluyen la falta de planificación del tiempo compartido, retardos de tiempo impredecibles en el manejo de la memoria virtual y baja capacidad del manejo de la granularidad en el tiempo de ejecución de las tareas.

RT-Linux es una versión de Linux que provee capacidad para manejar tiempo real duro. La idea principal del funcionamiento de RT-Linux se basó en la técnica de máquina virtual. RT-Linux implementa una capa entre el núcleo de Linux y el hardware. de tal forma que puede captar todas las peticiones de proceso y acceso al hardware de cualquier tarea. Ello incluye al núcleo de Linux (y por lo tanto también a su planificador), que se ejecutan como cualquier otra tarea.

La Fig. A.1 muestra el núcleo modificado que soporta tiempo real duro.

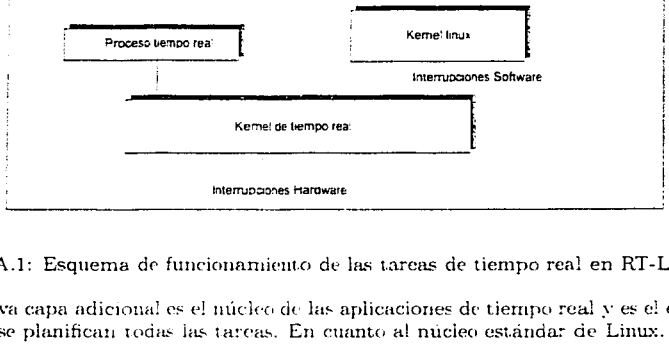


Fig. A.1: Esquema de funcionamiento de las tareas de tiempo real en RT-Linux

La nueva capa adicional es el núcleo de las aplicaciones de tiempo real y es el espacio donde se planifican todas las tareas. En cuanto al núcleo estándar de Linux. éste la ve como verdadero hardware. El planificador de RT-Linux asigna la menor prioridad al núcleo de Linux. el cual corre como una tarea más. De esta forma. se permite al usuario introducir tareas de tiempo real. que al ejecutarse poseen la más alta prioridad.

Los retardos temporales impredecibles son eliminados. ya que tales aplicaciones son pequeñas y operacionalmente limitadas. Las tareas de tiempo real son privilegiadas. ya que tienen acceso directo al hardware y no usan memoria virtual. Las tareas de tiempo real son implementadas como módulos en el espacio del núcleo de Linux y se espera que ellas no hagan llamadas al sistema Linux. Los módulos pueden ser cargados o descargados dinámicamente en memoria. lo que les da una gran ventaja en su ejecución.

El código de inicialización de una tarea de tiempo real inicializa la estructura de tiempo real e informa a RT-Linux de su plazo límite y periodo de ejecución. Los

módulos en Linux son archivos objeto que son cargados en el espacio del núcleo de Linux.

Los módulos deben tener dos funciones en su código. Ellas son:

```
int init_module();
void cleanup_module();
```

La función `init_module()` es llamada cuando el módulo es cargado en memoria, similarmente la función `cleanup_module()`; deberá ser ejecutada cuando el módulo se descargue de la memoria.

Ya que los módulos son archivos objeto tienen que ser compilados de la siguiente manera:

```
gcc -c {banderas} mi_modulo.c
```

El comando anterior producirá un archivo `mi_modulo.o` que puede ser cargado en memoria con el comando `insmod` y después descargado con el comando `rmmod`.

El cuerpo principal de las aplicaciones RT-Linux lo constituyen diferentes hilos (threads) de ejecución. Los hilos son procesos ligeros que comparten un espacio común de memoria, permitiéndoles entre otras cosas una comunicación eficiente entre ellos.

Los hilos en RT-Linux son creados con la función `pthread_create()`, esta función solo puede ser creada dentro de la función init-module().

```
#include <pthread.h>
int pthread_create(pthread_t * hilo,
                   pthread_attr_t * atributo,
                   void *(*rutina) (void *),
                   void *argumentos);
```

El número de identificación del nuevo hilo es almacenado en la dirección apuntada por `hilo`, la función apuntada por `rutina` es el código de `hilo`, a esta función se le pasan los argumentos por medio del parámetro `argumentos`. Finalmente, el elemento `atributo` sirve para asignar los atributos del hilo como su periodo o el tiempo cuando la aplicación debe de comenzar.

Para planificar las tareas de tiempo real, RT-Linux cuenta con un número de rutinas para tal efecto. Las características que las siguientes funciones modifican pueden ser especificadas en el parámetro `attr_t` de la función `pthread_create()`. Las funciones son:

```
int pthread_setschedparam(pthread_t hilo, int polití-
ca, const struct sched_param *parametros);
```

Con esta función se puede especificar la prioridad de `hilo`.

```
int pthread_make_periodic-np(pthread_t hilo,
const struct itimerspec *tiempo);
```

Esta función asigna el periodo de ejecución al hilo `hilo`, por medio de la estructura `itiemrspec`.

```
int pthread_wait_np(void);
```

Esta función detiene la ejecución del `hilo`.

```
int sche_getpriority_max(int política);
int sche_getpriority_min(int política);
```

Esta funciones asignan respectivamente la máxima y mínima prioridad de ejecución.

Con este conjunto de instrucciones es posible crear un pequeño ejemplo de una tarea de tiempo real, que a continuación se muestra:

```
#include <rtl.h>
#include <time.h>
#include <pthread.h>
pthread_t hilo;
void * comienza_rutina (void *arg) {
   struct sched_param p;
   p.sched_priority = 1;
pthread_setschedparam (pthread_self() SCHED_FIFO, &p);
   pthread_make_periodic_np (pthread_self(),
                    gethrtime(), 500000000);
   while (1) {
      pthread_wait_np();
      rtl_printf ("Hola mundo \n");
   }
   return 0;
}
int init_module (void) {
   return pthread_create (&hilo, NULL,
                          comienza_rutina, 0);
}
void cleanup_module (void) {
   pthread_cancel (hilo);
   pthread_join (hilo, NULL);
}
```

El código anterior escribe la cadena "Hola mundo" cada medio segundo en la pantalla.

Además de las rutinas principales que fueron mencionadas, el API de RT-Linux ofrece funciones de medición de tiempo, manejo de RT-Fifos que son canales de comunicación entre aplicaciones "normales" y de tiempo real, memoria compartida, acceso a memoria física, acceso a puertos de entrada y salida, interrupciones duras y suaves y un manejador de acceso al puerto serial.

La última versión del sistema operativo es la versión 3.3 y puede ser bajada de la siguiente dirección `http://www.rtlinux.com`.

Apéndice B

Diseño temporal

Un sistema de cómputo de tiempo real está restringido por los requerimientos temporales que se le hayan impuesto. Cada sistema de cómputo de tiempo real (distribuido o no) tiene un conjunto de tareas que debe realizar. Por lo tanto, se establece que, para que tales sistemas logren terminar a tiempo su cometido, necesariamente las tareas que tiene que realizar deben ser planificadas.

Para cumplir con los requerimientos en tiempo de un sistema de cómputo de tiempo real, en este trabajo se ha tomado el modelo temporal propuesto por M. Saksena [Saksena(1995)]. Este diseño consta de tres diferentes niveles (Fig. B.1):

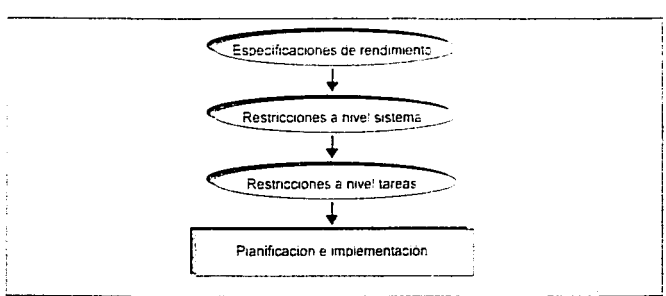


Fig. B.1: Diseño temporal

Los niveles son:

- *Especificaciones de rendimiento*: En este nivel las restricciones temporales son muchas veces implícitas y son especificadas en términos de los requerimientos del rendimiento de la aplicación.

- *Restricciones temporales a nivel sistema*: Estas restricciones son derivadas de las *especificaciones de rendimiento* del sistema y son expresadas como restricciones en las entradas y salidas del sistema, por ejemplo, máxima latencia entre nodo y nodo, mínimo y máximo jitter o máximo periodo de inicialización de una computación, etc.

- *Atributos temporales a nivel tareas*: Al más bajo nivel, las restricciones temporales son expresadas en términos de periodos, plazos límites y fases de las tareas. En este nivel se pueden aplicar las técnicas de planificación y es, a este nivel donde se sitúa la aportación de esta tesis.

En este modelo las relaciones entre los niveles son necesarios pero no suficientes. Nótese que aunque se cumplan con las restricciones a nivel tareas, no necesariamente las restricciones temporales a nivel sistema y/o las especificaciones de rendimiento se deben cumplir. Por lo tanto, se debe hacer una búsqueda de los valores de los parámetros de las tareas que cumplan con los requerimientos en todos los niveles.

Así en esta tesis, se estudia el papel de la planificación de tareas que junto con otros requerimientos sobre la aplicación sirvan para satisfacer las especificaciones de rendimiento que de una aplicación se hayan hecho.

Bibliografía

[Almeida *et al.*(1999)] L. Almeida. R. Pasadas y J. A. Fonseca. *Using a planning scheduler to improve the flexibility of real-time field bus networks. Control engineering Practice*. vol. 7. páginas 101–108. 1999.

[Alonso *et al.*(2001)] A. Alonso. R. López. J. A. de la Puente. B. Álvarez y A. Iborra. *Using linux and ada in the development of distributed computer control systems*. En *IFAC Conference on New Technologies for Computer Control*. páginas 295–300. 2001.

[Colouris *et al.*(2001)] G. Colouris. J. Dollimore y T. Kindberg. *Distributed Systems: Concepts and Design*. Addison Wesley. tercera edición. 2001.

[Conway *et al.*(1967)] R. W. Conway. W. L. Maxwell y L. W. Miller. *Theory of Scheduling*. Addison-Wesley. 1967.

[Devillers y Goossens(2000)] R. Devillers y J. Goossens. *Liu and layland's schedulability test revisited. Information Processing Letters*. vol. 73. páginas 157–161. 2000.

[Fohler(1995)] G. Fohler. *Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems*. En *16th IEEE Proceedings of the Real-Time Systems Symposium*. páginas 152- 161. 1995.

[Fonseca y Almeida(1999)] J. A. Fonseca y L. M. Almeida. *Using planning scheduler in the can network*. En *Proceedings ETFA. International Conference on Emerging Technologies and Factory Automation*. vol. 2. páginas 815–821. 1999.

[Kerr y Slany(1994)] R. M. Kerr y W. Slany. *Research issues and challenges in fuzzy scheduling*. Informe técnico. Christian Doppler Laboratory for Expert Systems. Viena. Austria. 1994.

[Kopetz(1997)] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers. 1997.

[Krishna y Shin(1997)] C. M. Krishna y K. G. Shin. *Real-Time Systems*. McGraw Hill. 1997.

[Liu y Layland(1973)] C. L. Liu y J. W. Layland. *Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the Association for Computing Machinery*. vol. 20. no. 1. páginas 46–61. 1973.

[Liu(2000)] J. W. S. Liu. *Real-Time Systems*. Prentice Hall. 2000.

[Livani *et al.*(1999)] M. A. Livani. J. Kaiser y W. Jia. *Schedulind hard and soft real-time communication in a controller area network. Control Engineering Practice*, vol. 7. páginas 1515–1523. 1999.

[Lorczak *et al.*(1989)] P. R. Lorczak. A. K. Caglayan y D. E. Eckhardt. *A theorical investigation of generalized voters*. En *Nineteenth IEEE International Symposium on Fault-Tolerant Computing*. páginas 444–451. 1989.

[Mendoza *et al.*(2001)] P. Mendoza. J. Vila. S. Terrasa. P. Balbastre y A. Crespo. *Using rt-linux for developing real-time embedded systems*. En *IFAC Conference on New Technologies for Computer Control*. páginas 301–306. 2001.

[Mok(1983)] A. K.-L. Mok. *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*. Tesis Doctoral. Massachusetts Institute of Technology. Estados Unidos. 1983.

[Saksena(1998)] M. Saksena. *Real-time system design: A temporal perpective*. En *IEEE Canadian Conference on Electrical and Computer Engineering*. vol. 1. paginas 405–408. IEEE. 1998.

[Stankovic(1988)] J. A. Stankovic. *Misconceptions about real-time computing: A serious problem for the next generation. IEEE Computer*. vol. 21. no. 10. páginas 10–19. 1988.

[Stankovic *et al.*(1995)] J. A. Stankovic. M. Spuri. M. Di Natale y G. C. Buttazzo. *Implications of classical scheduling results for real-time systems. IEEE Computer*. vol. 28. no. 6. páginas 16–25. 1995.