Factibilidad de Sistemas de Tiempo - Real

Javier Darío Orozco

Tesis presentada para el
Doctorado en Ingeniería

Departamento de Graduados
Departamento de Ingeniería Eléctrica
Universidad Nacional del Sur

Agradecimientos

Varios hechos concurrentes contribuyeron a que esta tesis sea posible. La mayor parte de ellos no fueron casuales sino resultado de voluntades y deseos de muchos que han colaborado directa e indirectamente.

Quiero agradecer especialmente al Ing. Jorge Santos por su constante estímulo, por el interés puesto en mi trabajo, por su trabajo, por haber sido y seguir siendo ejemplo para mi actividad académica y por su confianza y afecto.

A Rafael Fontao y Antonio Quijano por haber sido mis primeros directores de beca de CONICET. A Omar Alimenti, Carlos Matrángolo y Rodolfo del Castillo que me enseñaron a querer a esta disciplina de la Ingeniería y que, siendo aún alumno de grado, me dieron la oportunidad de iniciarme en la docencia e investigación.

A mis actuales compañeros: Edgardo Ferro, Ricardo Cayssials, Omar Alimenti y Rodrigo Santos por haber compartido discusiones, ideas, alegrías y sinsabores pero, por sobre todo, quiero agradecerles entrañablemente su amistad puesta a prueba en innumerables oportunidades durante tantas horas de convivencia diaria.

Al *Negro* Desages por confiar en mi capacidad y por que, seguramente, se hubiese alegrado por la culminación de esta tesis.

A mi esposa Ana y a mis hijos Francisco y Mariano por sus innumerables formas de ayuda y por las horas de esposo y padre que perdieron; a mi *vieja* que me apoyó en toda mi carrera, les debo y dedico esta tesis.

A todos ellos y a quienes involuntariamente omití, sinceramente gracias.

Prefacio

Esta tesis ha sido desarrollada en el Laboratorio de Sistemas Digitales del Departamento de Ingeniería Eléctrica de la Universidad Nacional del Sur. Según se ha adoptado como norma para los estudios de posgrado del Laboratorio de Sistemas Digitales, esta tesis está basada en publicaciones en actas de congresos y revistas de las cuales soy autor principal o coautor. Estas publicaciones y/o el contenido de esta tesis no han sido ni serán utilizados como contenido fundamental de otra tesis llevada a cabo en dicho ámbito. Las principales publicaciones[1] en orden cronológico inverso son: **Orozco**, J., Cayssials, R., Santos, J. and Santos, R. M., "On the minimum number of priority levels required for the Rate Monotonic scheduling of real-time systems", *Proc. 10th Euromicro Workshop on Real Time Systems – WIP'98* IEEE Computer Society Press, Berlín, 1998; **Orozco**, J., Cayssials, R., Santos, J., y Ferro, E.,"802.4 Rate Monotonic Scheduling in Hard Real-Time Environments: Setting the Medium Access Control Parameters", *Information Processing Letters.* Elsevier Science Publishers, 1997; Santos J., Gastaminza M..L., **Orozco** J., Picardi D. y Alimenti O., "Priorities and protocols in hard real-time LANs" *Computer Communications*,

[1] Actas de congresos: [4], [28], [32], [38], [43] y revistas: [9], [30], [39], [41]

Butterworth-Heinemann, 1997; Ferro, E., **Orozco**, Santos J., J., Cayssials, R. "Sincronización de tareas en Tiempo Real Duro utilizando el método de las ranuras vacías" *Revista de Información Tecnológica,* 1996; Santos J., Gastaminza, M.L., **Orozco**, J. and Matrangolo C., "802.5 priority mechanism in hard real-time RMS applications" *Computer Communications.* Butterworth-Heinemann, 1994; Santos J, y **Orozco** J. "On the priority Mechanism of 802.4 in Hard Real-Time Factory Communications" *IEEE International Symposium on Indutrial Electronics ISIE'94.* Chile, 1994; Santos J. y **Orozco**, J. "Rate Monotonic scheduling in hard real-time systems" *Information Processing Letters.* Elsevier Science Publishers, 1993; Santos J., **Orozco** J. y Alimenti O. "Perfomance Evaluation of Standard Lan Protocols in Time Constrained Enviroments", *Proceedings IEEE INFOCOM'89,* Ottawa, Canadá, 1989.

Javier Orozco
Bahía Blanca, diciembre de 1998

Resumen

Los sistemas de Tiempo-Real son de indudable importancia en: ambientes industriales automatizados, aviónica, robótica, medicina, transacciones bancarias e incluso entretenimiento entre otros. Han dejado de ser exclusivos de complejas aplicaciones militares de defensa para convertirse en una realidad cotidiana. Por lo tanto, rápidamente nos volvemos más y más dependientes del buen funcionamiento de los mismos. Una visión de su importancia es expresada acabadamente por Lui Sha [48] en un artículo del cual se transcribe un párrafo: "*Los sistemas de cómputo en tiempo real son críticos en la infraestructura tecnológica de una nación industrializada. Sistemas modernos de telecomunicaciones, fábricas, sistemas de defensa, aviones y aeropuertos, navegación espacial, experimentos de física de alta energía no pueden operar sin ellos. Efectivamente, los sistemas de cómputo de tiempo real controlan muchos de los sistemas que nos hacen productivos, resguardando nuestra libertad,*

permitiéndonos explorar nuevas fronteras de la ciencia y la ingeniería".

Debido a que el campo de aplicación principal de los desarrollos sobre el tema es el industrial y el soporte de comunicaciones en entornos multimediales, se optó por estudiar el tema considerando las redes normalizadas. Dentro de este marco se analizarán los protocolos y sus mecanismos de prioridades, atacando el problema crítico de identificar el peor caso de carga de un sistema dado y determinar *a priori* si el mismo es *factible*.

Por otro lado, el análisis de los sistemas desde el punto de vista de las comunicaciones y sus características temporales, es extendido a otros sistemas de tiempo real tal como la diagramabilidad de tareas de tiempo real en sistemas multitarea-monoprocesador. Se desarrolla un modelo de diagramador aplicable a distintas instancias del modelo multiusuario-monorecurso contemplando aspectos tales como la implementación de diagramadores imperfectos que provocan inversiones de prioridad, restricciones debidas a un número limitado de niveles de prioridad, las sobrecargas impuestas por el diagramador, el comportamiento de sistemas con requerimientos

mixtos (usuarios de tiempo real y usuarios no de tiempo real) y las particularidades de los diagramadores distribuidos (caso típico en redes) y de los diagramadores concentrados (S.O. multiusuario monoprocesador).

Indice

Capítulo 1

Introducción a los Sistemas de Tiempo-Real

1.1. Introducción.

> *En los Sistemas de Tiempo Real (STR) los resultados no sólo deben ser correctos desde el punto de vista aritmético y lógico sino que deben ser obtenidos antes de un cierto tiempo predefinido* [50].

Sistemas de Control de Procesos, fabricación automática en plantas mediante robots, entornos multimediales y telecomunicaciones son ejemplos concretos de STR donde la computadora cumple un papel esencial.

En ambientes de tiempo real son comunes los sistemas en los que una población de usuarios comparten un único recurso. Debido a restricciones temporales, el uso del mismo debe ser garantizado a cada usuario dentro de un cierto intervalo denominado *plazo de vencimiento* ("deadline") del usuario [44].

El mundo real es esencialmente concurrente. Por lo tanto, un sistema que deba interactuar con él deberá comportarse de la misma forma. Una consecuencia de la concurrencia es la competencia de un conjunto de usuarios por la utilización de un recurso; esto obliga a utilizar mecanismos que arbitren el uso del mismo.

La asignación del recurso a un usuario es realizada por un mecanismo denominado *diagramador* quien, de acuerdo a algún criterio de selección, optará por uno de los usuarios que compiten por el uso del mismo. Ejemplos típicos de sistemas que requieren mecanismos de diagramación son las redes de comunicaciones donde un conjunto de usuarios compite por el uso del medio y procesadores en los cuales deben ejecutarse diferentes tareas. Si el sistema es de tiempo real, es importante que se contemplen las especificaciones temporales de los usuarios de manera tal que el recurso les sea asignado y completen su uso dentro de sus plazos de vencimiento.

A fin de que el diagramador pueda diferenciar los distintos usuarios, en su forma más general[2] se asigna a los últimos una *prioridad* que es función de sus especificaciones temporales. En caso de requerimientos múltiples, el diagramador asignará el recurso de acuerdo a una *disciplina de prioridades* que establece una relación

[2] Los Ejecutivos Cíclicos no utilizan mecanismos basados en prioridades.

binaria de orden sobre el conjunto de usuarios constituyendo un orden total denominado *pila de prioridades*.

> *Dado un conjunto de usuarios de tiempo real, un recurso compartido, y un diagramador, se dirá que dicho sistema multiusuario-monorecurso es factible, si el mismo garantiza el cumplimiento de los vencimientos del conjunto de usuarios.*

La teoría de sistemas de tiempo real no se restringe al estudio de la diagramación ni a los sistemas de cómputo. También se desarrolla en el estudio de los sistemas de fabricación, los sistemas de transporte, los sistemas de control de procesos, comunicaciones, etc. Es, sin embargo, importante notar que los problemas de diagramación de los sistemas de tiempo real difieren de los sistemas de planificación que suelen considerarse en las áreas de investigación operativa. En la mayoría de los problemas de planificación en investigación operativa, hay un sistema determinado con características completamente especificadas y servicio estático. La meta consiste en encontrar los programas estáticos óptimos que minimicen el tiempo de respuesta para un conjunto dado de tareas. En general, en sistemas de tiempo real, no hay generalmente ningún incentivo para minimizar el tiempo de respuesta a no ser el de cumplir con los límites temporales.

Cuando para el funcionamiento de un sistema de tiempo real resulte intolerable la pérdida de algún vencimiento, se dirá que es un sistema de *tiempo real duro*. En ellos, el análisis de factibilidad debe cumplir con las siguientes propiedades:

- Debe poder ser aplicado *a priori*:

En general la validación del sistema debe ser posible previamente a que el mismo se encuentre en funcionamiento, ya que la pérdida de vencimientos es intolerable y puede tener resultados críticos.

- Debe ser *suficiente*:

Si es posible establecer una condición suficiente para analizar la factibilidad de un determinado sistema de TR y esta condición se verifica, es posible asegurar que cada usuario podrá cubrir sus requerimientos.

- Debe ser en lo posible *necesario*:

Si bien la condición de necesidad puede relajarse en pos de la aplicación de métodos analíticos menos complejos, el análisis de factibilidad deberá resultar tan necesario como sea posible. Así, si el sistema es no factible, se pueden identificar con mayor facilidad las causas que determinan esta condición e intentar realizar, si es posible, los ajustes necesarios que permitan modificarlas. Por otro lado un análisis más preciso permite realizar un uso eficiente del recurso permitiendo así la validación de sistemas con mayores requerimientos.

> *Debe destacarse que, para que pueda cumplirse con las condiciones anteriores, resulta imprescindible contar con un modelo exacto del sistema y de sus características dinámicas e identificar con precisión el peor caso de concurrencia sobre el recurso.*

Un análisis muy conservador o pesimista, disminuirá la aplicabilidad de un sistema. Por otro lado, la capacidad ociosa del mismo tiene efectos importantes en los costos de implementación.

1.2. Falsos Conceptos.

La teoría de sistemas de tiempo real es relativamente nueva. La falta de tratamiento adecuado se debe a algunos conceptos erróneos acerca de los mismos.

- *El diseño de sistemas de tiempo real es un proceso empírico*

Es desde luego cierto que el estado del arte en la elaboración de los sistemas en tiempo real es principalmente "ad hoc". Esto no significa que no pueda darse un enfoque científico al problema; más aún, este enfoque es necesario dada la proliferación de sistemas que requieren su utilización y la creciente complejidad de los mismos. El primer vuelo del transbordador espacial fue aplazado, a un costo considerable, por causa de un leve defecto en la regulación del tiempo, defecto que surgió debido a una sobrecarga transitoria de la CPU durante la inicialización del sistema.

- *Los Avances en el hardware cumplirán, con los requisitos de Tiempo Real*

Siempre es posible pensar en una aplicación de tiempo real cuyos requerimientos temporales utilicen totalmente o aún desborden la nueva disponibilidad de potencia de cómputo. Desde luego, nuevamente mayor velocidad de hardware no garantiza el cumplimiento de los vencimientos. Obviamente, podrán ser tratados

problemas hoy no manejables pero si factibles, pero esto no es generalizable a todo sistema futuro. Igualmente obvio resulta el hecho de que los recursos seguirán siendo finitos y no podrá sustituirse el criterio de una utilización racional de los mismos.

- *Todo proceso que interactúe con el medio constituye un sistema de tiempo real.*

Es comúnmente utilizado como argumento de venta de sistemas industriales que, en general, lo único que garantizan es que un conjunto de tareas periódicas (actualización de sensores, computo de rutinas de control y señalización a actuadores) será ejecutado en algún momento dentro de un ciclo que contiene al conjunto. Si bien podría definirse un sistema predecible sobre la base de este modelo, no es posible la especificación de requerimientos temporales individuales, quedando supeditados los tiempos a una distribución más o menos equitativa de los recursos. Este tipo de sistemas efectúa en general un mal uso de los recursos al asignar la misma utilización del recurso tanto a procesos lentos como rápidos. Por este motivo, sistemas que serían factibles con un uso más racional de los recursos, necesitan mayor poder de cómputo. Este falso concepto está íntimamente relacionado con el siguiente.

- *Computación en Tiempo-Real equivale a computación rápida*

El concepto (aunque relativo) de computación rápida está asociado a la minimización del tiempo de respuesta promedio de un conjunto de tareas. El objetivo del cómputo en tiempo real es satisfacer las exigencias críticas de vencimiento de las tareas. Los resultados post-vencimiento pueden resultar inútiles o incorrectos. Antes que ser rápido la propiedad más importante que deberá tener un sistema de

tiempo real es ser predecible esto es que, funcional y temporalmente sea determinístico. La computación rápida contribuye, desde luego, a cumplir con los requisitos temporales, pero por si misma no garantiza la predictibilidad.

- *La programación en Tiempo-Real es codificación en lenguaje máquina.*

Para atender estrictas exigencias de tiempo y suplir la falta de lenguajes que permitan expresar semánticas de tiempo absoluto, la práctica actual en la programación en tiempo real recurre a las técnicas de optimización a nivel máquina. Estas técnicas requieren mucho trabajo especialmente al modificar amplios programas de tiempo real. Un objetivo principal en la investigación sobre sistemas de tiempo real es el de brindar un soporte al programador que le permita independizarse de la administración manual de los recursos del sistema, esto otorgará por añadidura mayor robustez a los mismos.

- *Los sistemas de tiempo real duro son utilizados para soportar aplicaciones lo suficientemente críticas como para que, una falla en su funcionamiento, acarree la pérdida de vidas humanas o catástrofes importantes.*

Si bien es cierto que aviones, aeropuertos, naves espaciales, sistemas de control de centrales nucleares, etc. no pueden funcionar sin ellos y que una falla en sus sistemas puede provocar daños materiales y aún personales importantes, la aplicación de STR duros no se restringe únicamente a este contexto sino que, aún cuando los daños producidos sean pequeños, si el sistema debe cumplir con sus restricciones temporales para asegurar un buen funcionamiento, será

necesario aplicar entonces, para analizar su factibilidad, la misma teoría de TR que en el caso de los primeros.

Una rama de la investigación de STR trata aquellos sistemas en los cuales algunas aplicaciones pueden tolerar la pérdida de vencimientos en un número limitado de oportunidades sin que esto tenga consecuencias relevantes en el comportamiento del sistema. Estas consideraciones no hacen juicio alguno sobre la complejidad o importancia del sistema sino a una característica particular de alguna de las aplicaciones del mismo.

Un sistema simple y sin riesgos humanos y materiales, puede ser absolutamente intolerante a las pérdidas de vencimientos y uno complejo no. La tolerancia o no a la pérdida de vencimientos determinará si el sistema el de tiempo real duro o blando.

1.3. Sistemas Operativos y Lenguajes en Tiempo Real.

1.3.1. Sistemas operativos.

En los sistemas de tiempo real, el sistema operativo y la aplicación están más estrechamente relacionados que en los sistemas usuales de tiempo compartido. Esto se debe a que, además de suministrar abstracciones de alto nivel para programadores de tiempo real, al mismo tiempo debe cumplir con las consideraciones temporales [33]. Estas dependen fundamentalmente de la implementación y del ambiente en que deben operar. Además, casi

todas las técnicas de manejo de los recursos utilizadas en los sistemas operativos existentes no están diseñadas para garantizar que se satisfagan los requisitos de tiempo crítico. Algunos importantes temas de investigación sobre sistemas operativos incluyen:

- **Administración de los recursos basados en tiempo.** Tradicionalmente, la política de asignación de recursos más común consiste en dar acceso según el orden de llegada (FIFO). Esta política ignora totalmente las restricciones temporales de las tareas. Se deben desarrollar políticas de asignación que puedan cumplir con los requisitos de planificación en tiempo real. Tales políticas de asignación de recursos deberían ser aplicables, no solamente al procesador, sino también a los recursos de memoria, de I/O y de las comunicaciones. De hecho, un nuevo paradigma de sistema operativo, es necesario.

- **Facilidades para la especificación del problema.** Las funciones de un sistema operativo en tiempo real deberían poder adaptarse a una gama de requerimientos del usuario y del sistema. Por ejemplo, un sistema operativo en tiempo real debería facilitar una separación entre la *disciplina de prioridades* y el *mecanismo de diagramación*. Así, el usuario podría seleccionar la disciplina de prioridades y el mecanismo de asignación de recursos basado en prioridades más conveniente para una aplicación, plataforma o situación particular.

- **Soporte integral para la diagramación de sistemas.** Los principios de diagramación en tiempo real deben aplicarse a

los recursos del sistema, a las tareas de aplicación y a la concepción global del sistema operativo. Para que una secuencia de acciones se cumpla en un determinado plazo, se deben satisfacer las exigencias en tiempo y forma de los recursos para cada acción de la secuencia. Retrasos que ocurran inoportunamente en cualquier etapa del proceso, pueden acarrear incumplimiento de los vencimientos.

- **Tolerancia a fallas.** Producir resultados correctos es tan importante como producirlos a tiempo. Sin embargo, esta visión del problema puede relajarse en aquellos sistemas en los cuales es posible definir un nivel mínimo de servicio que debe ser garantizado. Es decir, en caso de imposibilidad de cumplir con el vencimiento de una tarea es preferible la obtención de un resultado aproximado a la ausencia completa del mismo. Cualquiera sea la postura frente al problema, la solución requiere el agregado de redundancia al sistema ya sea a nivel de hardware, de software o de ambos.

  El manejo de errores requiere el cumplimiento de una serie de pasos que van desde la detección del error y la ubicación de la falla hasta la rehabilitación del sistema. Todos estos pasos deben realizarse teniendo en cuenta consideraciones temporales y de robustez del sistema. Por otro lado resulta obvio que, si el mecanismo de recuperación es lento, no predecible o inseguro no cumplirá con los requerimientos de tiempo real. La solución resulta de resolver el compromiso entre caro y rápido (hardware) y lento y barato (software).

1.3.2 Lenguajes.

A medida que aumenta la complejidad de los sistemas de tiempo real, irá creciendo la demanda de abstracciones de programación proporcionadas por lenguajes que otorguen la debida consideración a las necesidades de cobertura de los requisitos de tiempo real. Los puntos importantes de investigación incluyen:

- **Soporte para el manejo de parámetros temporales.** Las construcciones del lenguaje deberían soportar la expresión de exigencias temporales. Por ejemplo, la organización de tareas debería soportar la aparición de una excepción cuando no se cumple el vencimiento de una tarea. Por otro lado, el entorno de programación debería ofrecer al programador las funciones necesarias para controlar y seguir el rastro de la utilización de recursos por los módulos software durante el proceso de desarrollo, aún permitiendo una estimación de tiempos antes de obtenido el conjunto de ejecutables final. Esto abarca la capacidad de desarrollar modelos dinámicos de cualquier porción del sistema que permitan realizar un análisis de desempeño en términos temporales antes de construirlo [1].

- **Chequeo de factibilidad.** Dado un conjunto de algoritmos de diagramación bien definidos, el análisis de factibilidad nos permite determinar si los requisitos temporales pueden cumplirse. Con el soporte adecuado para el manejo del tiempo, puede ser posible efectuar chequeo de factibilidad en tiempo de compilación. Esta idea es similar al concepto de chequeo de tipos.

- **Soporte para los programas distribuidos y la tolerancia a fallas.** El problema de predecir el comportamiento de los programas en tiempo real es más complejo en el contexto de los sistemas distribuidos. Las características especiales de la tolerancia a fallas pueden añadirse a la semántica del lenguaje; esto incluye mecanismos de recuperación basados en mediciones de tiempo y facilidades para implementar sistemas con redundancia [2].

1.4. Comunicaciones en Tiempo Real.

Con la creciente utilización de sistemas de tiempo real distribuidos, está aumentando la necesidad de disponer de redes de comunicación de datos capaces de comunicar mensajes de tiempo real.

Los diseñadores de redes de comunicación para los sistemas en tiempo real distribuidos deberán afrontar la responsabilidad y el desafío de construir una plataforma sobre la que se edificarán sistemas predecibles, estables y extensibles. Para lograr su objetivo, el sistema de comunicación en tiempo real debe poder satisfacer los vencimientos a nivel mensaje, los cuales están definidos por los parámetros temporales de las tareas comunicantes de tiempo real.

Los requerimientos de tiempo son manejados no solamente por la comunicación entre los procesos de aplicaciones sino también, por las funciones del sistema operativo con límites de tiempo invocadas por los procesos de aplicación. Los tipos de redes para este contexto se diferencian de las redes que no son de tiempo real en que, en

éstas, es suficiente verificar que las comunicaciones son correctas desde el punto de vista lógico; sin embargo, en redes de tiempo real es necesario verificar también la que son correctas desde el punto de vista temporal. La ingeniería de redes ha colaborado para que resulte posible conservar la integridad de la información transportada pero no han obtenido una solución completa a los problemas temporales. Cumplir con las restricciones temporales incluye asegurar la diagramabilidad de los mensajes síncronos (de tiempo real) y de mensajes asíncronos (no de tiempo real) [28], así como asegurar que las exigencias de tiempo de respuesta de los últimos sean satisfechas en la medida que lo permita el subsistema de tiempo real. Asegurar que se cumplan las restricciones temporales para los sistemas de comunicaciones en tiempo real estáticos mediante la tecnología actual es difícil. Asegurar que se cumplan en entornos dinámicos es un desafío substancial para las próximas generaciones de investigadores.

Si bien un canal de comunicaciones es un recurso tanto como un procesador, hay puntos a diferenciar entre el problema de la diagramación en un canal de comunicaciones y el problema de la diagramación en un procesador [43], [38], [41], [44].

- A diferencia del procesador, que tiene sólo un punto de acceso, un canal de comunicaciones es accedido por un conjunto de nodos distribuidos que compiten entre sí por su utilización. De ahí la necesidad de utilizar protocolos distribuidos que utilicen mecanismos de diagramación que permitan implementar disciplinas de prioridades sin provocar excesiva sobrecarga en la comunicación.

- Mientras que los algoritmos apropiativos son útiles para diagramar tareas en un procesador, la apropiación del canal durante la transmisión de un mensaje significará la pérdida del mismo aún cuando parcialmente haya sido leído por el nodo destino.

- La aplicabilidad del análisis de factibilidad a una red de comunicaciones no sólo debe concentrarse en las características del mecanismo de diagramación implementado a nivel del protocolo de acceso al medio sino que deberá considerar las características topológicas y geográficas de la misma. Una determinada red local y una metropolitana pueden tener idénticos mecanismos de acceso al medio. Aún así, si la factibilidad del sistema es función de la ubicación de los nodos en el canal tanto absoluta como relativa, ésta exigencia puede llegar a ser respetada en una red local pero ya no es razonable en una red de varias decenas de kilómetros. Esta restricción introduce nuevas variables al problema lo que hace que el mismo deba ser diferente para cada una de ellas.

- En sistemas distribuidos de tiempo real, las comunicaciones entre tareas residentes en diferentes procesadores deberán ocurrir en un tiempo acotado a fin de garantizar que la aplicación se ejecute antes de su vencimiento.

Debido a las dificultades que presenta la implementación de protocolos de tiempo real, sólo un limitado número de ellos está disponible para comunicaciones en tiempo real. Los protocolos de pasaje de ficha y en especial los de ficha temporizada aparecen como

los más aptos para tal fin ya que cuentan con la propiedad de que la ficha alcanza, a intervalos de tiempo acotados, cada nodo de la red. Ellos han sido incorporados en varios standards incluyendo Token Ring (802.5), Fibre Distributed Data Interface (FDDI), Token Bus (802.4), High-Speed Data Bus (HSDB), High-Speed Ring Bus (HSRB).

En el Capítulo 5 se desarrollan los temas de redes de comunicaciones en tiempo real con profundidad.

Capítulo 2

Diagramadores y Disciplinas de prioridades.

2.1. Modelo de Usuario.

Se considera un usuario de tiempo real a aquél que realiza requerimientos periódicos sobre un recurso los cuales deberán ser satisfechos antes de un cierto tiempo denominado plazo de vencimiento, *D* ("deadline") contado a partir del instante de generación del requerimiento. Cada usuario requerirá el uso del recurso potencialmente en una secuencia infinita de oportunidades. En cada período, un usuario requerirá la utilización del recurso durante un intervalo de duración *C*.

Si el intervalo entre requerimientos no es armónico, se tomará, para dicho usuario, como su período, al mínimo intervalo entre dos requerimientos sucesivos

Luego, a fin de permitir el análisis de factibilidad, debe conocerse a priori para cada usuario *i*:

- La máxima utilización del recurso por período, C_i.
- El mínimo intervalo entre dos requerimientos sucesivos, T_i.
- El vencimiento, D_i.

Luego, un usuario τ_i estará completamente caracterizado por la terna (T_i, D_i, C_i) y la expresión $U_i = C_i / T_i$ representará el *factor de utilización* del mismo.

En lo que sigue, se utilizará para representar la evolución de un sistema de *m* usuarios que requieren el uso de un recurso, un diagrama temporal como el mostrado en la Figura: 2.1.

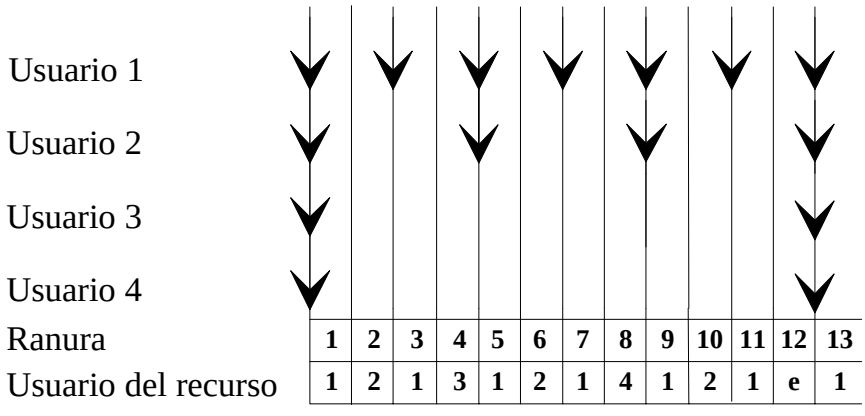


Figura: 2.1,
Diagrama temporal de requerimientos y asignación

Las puntas de flecha indican los instantes de generación de los requerimientos sobre el recurso. Como ejemplo, en un sistema multitarea-monoprocesador corresponderían a una tarea lista para ser ejecutada, en una red a un mensaje listo para ser enviado. Cada división vertical corresponde un instante de activación y la fila inferior indica el usuario que ocupa el recurso.

2.2. Modelo de Diagramador.

Cuando sobre un recurso concurren requerimientos de diferentes usuarios, es necesario adoptar una política de asignación del mismo e implementar un mecanismo que la lleve a cabo. Dicho mecanismo es denominado *diagramador*.

Existen diferentes algoritmos para implementar sistemas multitarea-monoprocesador en entornos no de TR. En ellos la principal motivación del diseño no pasa por respetar las restricciones temporales de los usuarios sino minimizar los tiempos de respuesta o distribuir el recurso en forma equitativa. En sistemas de TR el diagramador deberá garantizar el cumplimiento de los vencimientos.

Se dirá que un diagramador implementa una *disciplina de prioridades* si se establece una relación lineal de orden sobre el conjunto de tareas asignando una prioridad a cada una de ellas. La misma es utilizada por el diagramador para determinar, en caso de requerimientos concurrentes, que tarea tiene el derecho de utilizar el procesador en cada instante de activación. Si esta relación se establece durante la inicialización del sistema y permanece invariante en el tiempo tendremos un diagramador por prioridades

fijas. Si por el contrario, las prioridades cambian durante el funcionamiento del sistema tendremos un mecanismo de prioridades variables.

En su forma más general, un diagramador basado en prioridades diferencia a los *m* distintos usuarios de un sistema **S**(*m*) por su prioridad cumpliendo con las siguientes reglas operativas:

- El diagramador asignará el recurso al usuario de mayor prioridad de entre los que se encuentran "listos" en cada instante correspondiente a una asignación.
- El recurso permanecerá ocioso sólo si no existen usuarios listos para utilizarlo.

Con el fin de soportar el uso de disciplinas de prioridades variables, se adiciona la siguiente regla:

- El diagramador realiza el mantenimiento de la pila de prioridades, la cual debe actualizarse en cada instante correspondiente a una activación.

A partir del instante en que se efectiviza un requerimiento, el diagramador deberá garantizar la asignación del recurso durante las *C* unidades de tiempo solicitadas por el usuario dentro del plazo de vencimiento *D* del mismo. Claramente, al no poder tolerarse la pérdida de vencimientos, $C \leq D \leq T$.

Si luego de asignado el recurso, el usuario completa su utilización hasta satisfacer el requerimiento actual sin que pueda ser interrumpido tendremos un diagramador *no-apropiativo*; por el contrario, se dirá que es apropiativo si un usuario puede ser

interrumpido temporalmente en la utilización del recurso por un usuario de mayor prioridad. En este caso se dirá que el diagramador es *apropiativo*. La figura 2.2 muestra un ejemplo de un sistema dos usuarios $\mathbf{S}(2)=\{\tau_1,\tau_2\}=\{(T_1,D_1,C_1),(T_2,D_2,C_2)\}=\{(3,3,2\ ),\ (9,9,2)\}$.

En la primer fila se muestra el comportamiento de un sistema no-apropiativo y en la segunda el caso apropiativo. En él se observa que τ_1, de mayor prioridad que τ_2, desplaza a éste en la utilización del recurso.

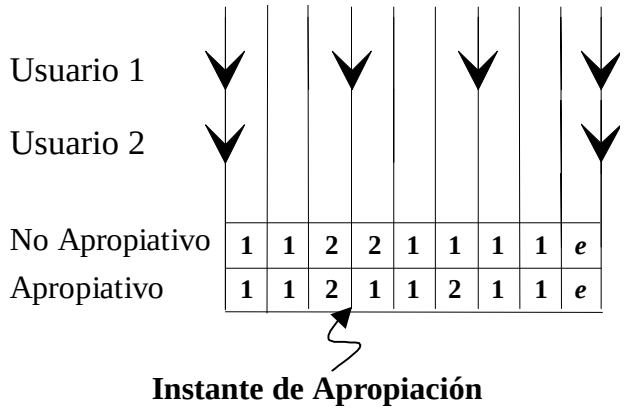


Figura: 2.2
Asignación del recurso mediante un diagramador no-apropiativo y uno apropiativo

Cuando el diagramador debe quitar temporalmente el recurso a un usuario, lo hace preservando las condiciones en que se encontraba el sistema en el momento de la apropiación de manera que el desplazado no perciba diferencias en el mismo al retomar la utilización del recurso. Este proceso de preservación de contexto hace que la utilización de diagramadores apropiativos representa una sobrecarga para el sistema comparado con el caso de los no apropiativos. A pesar de ello, es mucho más eficiente el uso de

diagramadores apropiativos ya que permiten diagramar sistemas que otro tipo no lograría.

La figura 2.3 muestra un ejemplo de un sistema dos usuarios $\mathbf{S}(2)=\{\tau_1,\tau_2\}=\{(T_1,D_1,C_1),(T_2,D_2,C_2)\}=\{(3,3,2),\ (9,9,3)\}$. En la primer fila se muestra el comportamiento de un sistema apropiativo y factible. En la segunda fila, el caso no-apropiativo, se observa que τ_1, aún siendo de mayor prioridad que τ_2 no puede completar su tiempo de utilización antes de su segundo vencimiento.

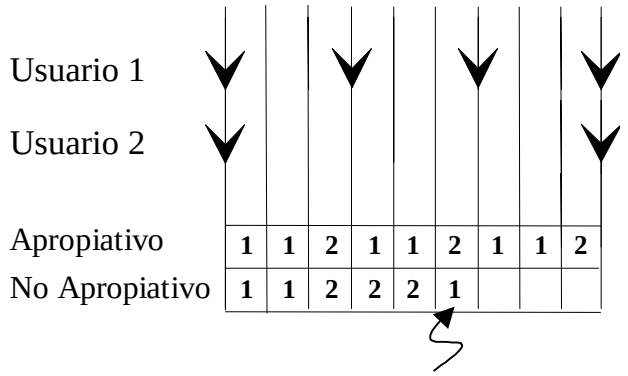


Figura: 2.3
Sistema dos usuarios $S(2)=\{\tau_1,\tau_2\}=\{(T_1,D_1,C_1),(T_2,D_2,C_2)\}=\{(3,3,2),\ (9,9,3)\}$.

Los distintos entornos donde deben aplicarse los modelos descriptos, hace necesario que los mismos se adapten a fin de contemplar: la implementación de diagramadores imperfectos que provocan inversiones de prioridad; las restricciones debidas a un número limitado de niveles de prioridades; el comportamiento de sistemas con requerimientos mixtos (usuarios de tiempo real y usuarios no de tiempo real) y las particularidades de los diagramadores distribuidos (caso típico en redes) y de los

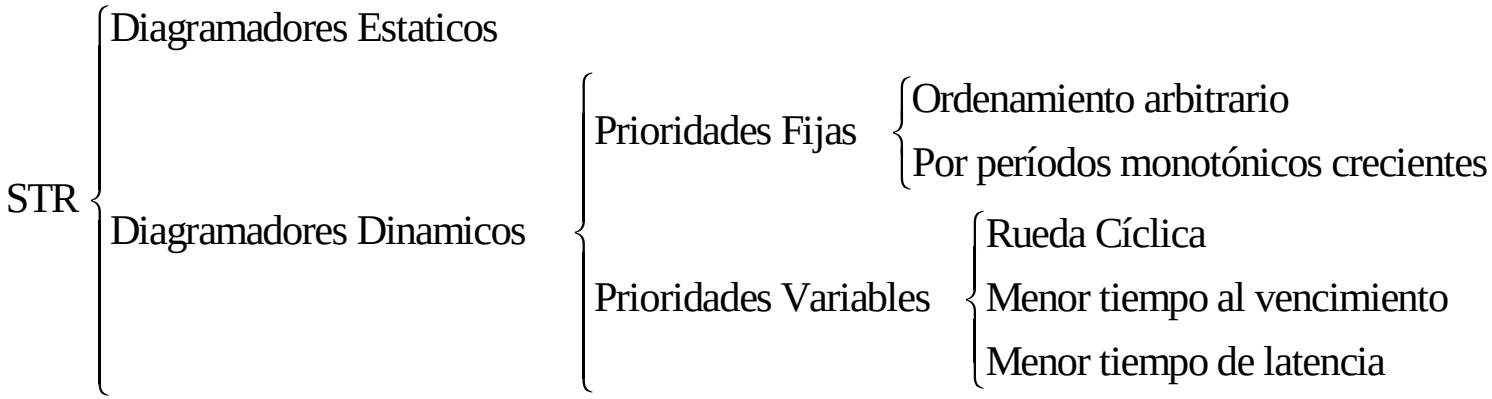
diagramadores concentrados (S.O. multiusuario monoprocesador). Estas consideraciones hacen que deban convenirse nuevas reglas operativas usuario/diagramador para cada caso particular.

Como el modelo responde a distintas implementaciones del modelo multiusuario-monorecurso, en lo que sigue se hablará de sistemas multitarea-monoprocesador, mecanismos de acceso al medio en redes, etc. según resulte más claro dada su principal aplicación o el contexto en que se lo referencia.

Según sus características, los diagramadores pueden clasificarse en *estáticos* o *dinámicos* y estos últimos a su vez pueden utilizar disciplinas de prioridades *fijas* o *variables*. El cuadro: 2.1 muestra una clasificación más completa.

2.2.1. Diagramadores Estáticos.

Se dice que un diagramador es estático cuando tiene preestablecida la secuencia en que deben ejecutarse las tareas a fin de que las mismas constituyan un sistema factible. Usualmente utiliza para su implementación una o más tablas que indican la secuencia de activación de las tareas y que permanecen invariantes en el tiempo. Un punto importante y que también caracteriza a los diagramadores estáticos es que no pueden tratar arribos de nuevas tareas ni variaciones en las características temporales de las mismas. Su forma más inmediata son los *ejecutivos cíclicos.*



Cuadro: 2.1
Clasificación de los STR

2.2.1.1 Ejecutivos Cíclicos.

Las primeras soluciones al problema de obtener sistemas de tiempo real factibles en sistemas operativos multitarea-monoprocesador fueron por medio de *ejecutivos cíclicos* (EC). En éstos, un algoritmo prepara *"off-line"* una tabla que es recorrida cíclicamente por el diagramador en tiempo de ejecución. Esta tabla, a la manera de un *calendario* **[8]**, [19], determina una secuencia fija de tareas a ser ejecutadas. En general puede asumirse que cada requerimiento es periódico y debe ser atendido antes de que se registre el próximo ($T=D$). A fin de poder acomodar a todos los requerimientos dentro de sus vencimientos resulta necesario que la longitud de la tabla sea múltiplo de los períodos, con lo que tendrá una longitud igual al mínimo común múltiplo (MCM) de los mismos.

La longitud de la tabla determinará la duración de un ciclo completo de ejecución denominado *ciclo mayor*. El intervalo entre la ejecución de dos tareas consecutivas es denominado *ciclo menor*. La duración de éste último es calculada de manera de permitir la factibilidad del sistema tomando en cuenta además las limitaciones impuestas por el procesador. En cada activación, las tareas son invocadas para ser ejecutadas durante un ciclo menor o *ranura*. Luego, la periodicidad de cada tarea determinará el número de veces y la posición que ocupa la misma en la tabla del EC.

Se debe notar que, el sistema no es *apropiativo* es decir, una tarea deberá finalizar su ejecución dentro de un ciclo menor. Si su duración es menor o no hay tareas listas para ser ejecutadas, una tarea ociosa ocupará el tiempo disponible. Un diseño simple **[24]**,

consiste en utilizar un contador descendente que genere una interrupción cuando alcanza el valor de cuenta cero. El mismo es precargado con un valor que corresponde temporalmente a un ciclo menor. A fin de que el ciclo mayor resulte de una longitud razonable pueden hacerse los períodos múltiplos entre sí [3].

Ejemplo 2.1: Supongamos un sistema con períodos 20ms, 40ms, 80ms y 240ms, MCM=240. En la Figura 2.4 se muestra la evolución temporal del sistema.

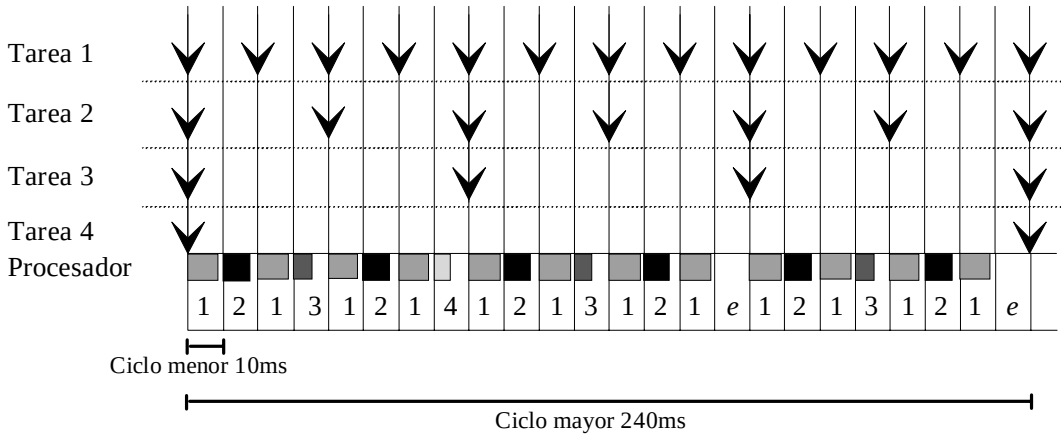


Figura: 2.4
Diagrama de asignación utilizando un ejecutivo cíclico.

En la Figura: 2.4, cada flecha vertical representa el instante en que una tarea está lista para ser ejecutada y cada división representa una ranura. En este caso el ciclo mayor tiene una longitud de 240ms o 24 ranuras de 10ms. En la parte inferior se muestra la secuencia de ejecución contenida en la tabla del EC y una representación de la ocupación del procesador. Las ranuras marcadas "*e*" son *ranuras*

vacías. Como al inicio de una ranura ninguna tarea del sistema estaba lista para ser ejecutada, el procesador ejecuta un lazo ocioso.

2.2.1.1.1 Ventajas de los ejecutivos cíclicos.

- En principio, mediante los ejecutivos cíclicos (EC), se puede implementar cualquier sistema que resulte factible por otro mecanismo de diagramación.
- La utilización del procesador está perfectamente establecida *a priori*. Por lo tanto, es posible predecir completamente el comportamiento futuro del sistema y tratar entonces sistemas con precedencia y cooperativos.
- La sobrecarga del sistema a causa del diagramador es mínima ya que la secuencia de tareas a ser ejecutada está preestablecida.
- Se puede prever durante el diseño del EC, puntos de activación para el arribo de tareas futuras, no existentes en el sistema inicial. Las tareas asincrónicas podrán ocupar las ranuras libres predeterminadas o aún, si la tarea que arriba es sincrónica, se conocerá de antemano, el período mínimo que la misma deberá tener para que el sistema resulte factible.

2.2.1.1.2 Desventajas de los ejecutivos cíclicos.

- La principal desventaja de los ECs es su rigidez para tolerar cambios en el sistema. Estos cambios no sólo pueden producirse una vez desarrollado el sistema sino incluso,

durante la fase de implementación del mismo. La experiencia con los ejecutivos cíclicos ha demostrado que, si el ciclo es muy grande, la planificación tiende a ser "ad hoc" en su naturaleza, muy ardua para generar y muy difícil de modificar. En su caso más general, la construcción de una tabla para obtener un sistema factible, es un problema NP-completo particularmente en sistemas distribuidos.

- Si bien dentro de las ventajas se citó la baja sobrecarga que introduce el diagramador, la utilización incompleta del tiempo disponible en una ranura produce una subutilización del procesador. Esta pérdida de rendimiento se produce al tener tiempos de activación fijos frente a tiempos de ejecución de las tareas variables. En este caso, se debe considerar como tiempo de ejecución el mayor de ellos dejando ocioso al procesador, cuando el tiempo de utilización es menor al de peor caso.

2.2.2. Diagramadores Dinámicos.

En estos diagramadores, el diagramador asigna el recurso basado en una disciplina de prioridades. Dicha disciplina puede asignar las prioridades a los usuarios en forma fija o variable.

En las disciplinas de prioridades fijas se asigna una prioridad a cada usuario y la misma permanece invariante en el tiempo. En las variables, algún criterio es utilizado para producir cambios en las mismas en cada instante de activación.

2.2.2.1 Disciplinas de Prioridades Variables.

Las disciplinas de prioridades variables asignan, en cada instante de activación, una prioridad a cada tarea. Esto se realiza de acuerdo a algún criterio de urgencia que asigna mayor prioridad a aquellas tareas cuyos vencimientos están más próximos; mediante cálculos basados en los vencimientos, tiempos de ejecución y periodicidad de las tareas o algún criterio preestablecido e invariante para producir el cambio. El diagramador realiza el mantenimiento de la pila de prioridades de manera de tener al tope de la misma, en cada instante de activación, al usuario con mayor prioridad y por ende el que posee mayor derecho de utilización del recurso.

2.2.2.1.1 Rueda cíclica.

La disciplina de Rueda Cíclica (RC) es muy utilizada en redes locales ya que algunas topologías, como la anillo, la implementan naturalmente. Por otro lado también es utilizada en sistemas operativos tanto de tiempo real como de propósito general.

Pertenece al grupo de disciplinas que tienen un criterio preestablecido para producir los cambios de prioridades; tal como su nombre lo indica, las mismas son rotativas. Por este motivo otorgan el uso del recurso en forma equitativa entre todos los usuarios.

Existen diferentes implementaciones de la rueda cíclica pero todas se basan en reordenar la pila de prioridades de manera que el que se encuentra al tope de la misma en un dado instante pasa al último lugar y todos los demás ascienden. En particular se pueden diferenciar dos formas de implementar una rueda cíclica de acuerdo

al momento en que se produce la reordenación. Si el recurso es asignado por una ranura por vez y en cada ranura se produce la reordenación tenemos una rueda cíclica apropiativa. Si en cambio el recurso es asignado por tiempo ilimitado y sólo se produce la reordenación en el momento en que es liberado tenemos una rueda cíclica no apropiativa. En todas ellas, el máximo factor de utilización esperable para cada usuario es de 1/*m* donde *m* representa el número de usuarios del sistema. Por ello, en el caso apropiativo, una condición suficiente para analizar la factibilidad del sistema está dada por:

$$U_i = \frac{C_i}{T_i} \leq \frac{1}{m} \tag{2.1}$$

En el caso de sistemas no apropiativos la condición es más restrictiva debiendo cumplirse:

$$T_{\min} \geq \sum_{i=1}^{m} C_i \tag{2.2}$$

Donde T_{min} es el menor período del sistema. El peor caso de carga para la tarea de menor vencimiento ocurre en el instante en que todas las tareas se encuentran listas y la tarea de menor período se encuentra en la base de la pila de prioridades. Luego deberá tolerar que el resto de los usuarios utilicen el recurso para disponer del mismo para sí antes de su vencimiento.

Si bien esta disciplina puede tratar sistemas con un factor de utilización del 100%, el uso del recurso es asignado en partes iguales a todos los usuarios. Ej.: Dado un sistema de tres tareas:

S(m)={(C_1,T_1),(C_2,T_2),(C_3,T_3)}= {(1,3),(2,6),(3,12)} y D_i= T_i,. El sistema verifica la expresión (2.1), luego, con una pila de prioridades inicial arbitraria [2, 3, 1], la evolución del sistema con un diagramador apropiativo, se muestra en la Figura: 2.5.

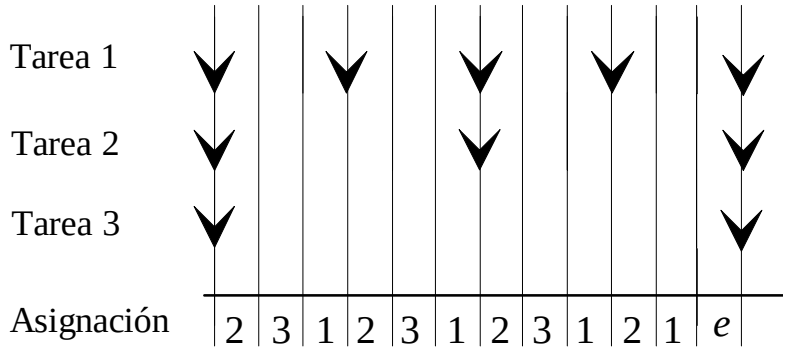


Figura: 2.5
Evolución del sistema S(m)={(C_1,T_1),(C_2,T_2),(C_3,T_3)}= {(1,3),(2,6),(3,12)} con un diagramador apropiativo.

Si el diagramador no es apropiativo, el sistema resulta no diagramable (no verifica la expresión (2.2)). La evolución del sistema para una pila inicial [1, 2, 3] está dada en la Figura: 2.6 donde, en la ranura 6 la tarea 1 no puede cumplir con su segundo requerimiento. Esto se debe a que el mínimo período que admite el sistema es de 6 ranuras.

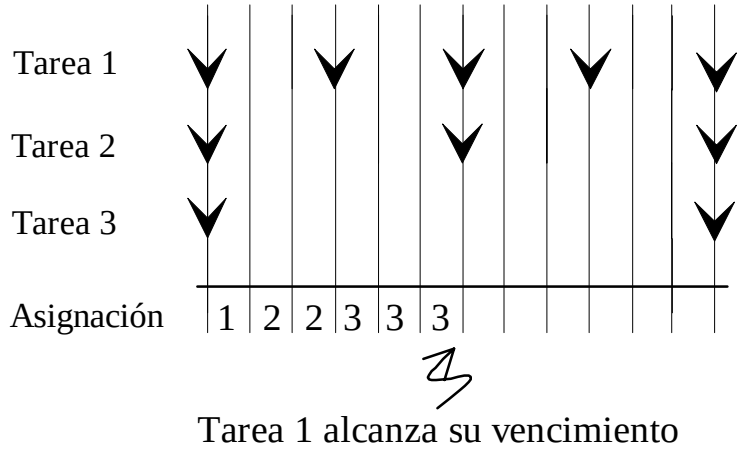


Figura: 2.6
Evolución del sistema S(m)={(C_1,T_1),(C_2,T_2),(C_3,T_3)}= {(1,3),(2,6),(3,12)} con una pila inicial [1, 2, 3]

2.2.2.1.1.1 Ventajas de la Rueda Cíclica.

- Es muy utilizada en redes locales ya que algunas topologías, como la de anillo, la implementan naturalmente.
- Es sencilla de implementar en diagramadores concentrados es decir toda la información necesaria para el mantenimiento de la pila de prioridades está disponible en todo momento para el diagramador. Esto hace que no sólo sea utilizada en sistemas operativos de tiempo real sino de propósito general.
- Admite factores de utilización de hasta el 100%.

2.2.2.1.1.2 Desventajas de la Rueda Cíclica.

- Si bien la Rueda Cíclica es simple en su concepción, su implementación en redes de tiempo real con protocolos de acceso al medio distribuidos, es natural sólo en anillos

reales o virtuales por pasaje de ficha. Si este no fuera el caso, será necesario un mecanismo que permita mantener la coherencia de las pilas de prioridades en cada nodo a fin de determinar unívocamente el derecho de acceso al medio ante requerimientos concurrentes. En sistemas operativos multitarea-monoprocesador o multitarea-multiprocesador con control centralizado su implementación es simple y por lo tanto, su uso es frecuente.

2.2.2.1.2 Menor Tiempo al Vencimiento.

La disciplina de Menor Tiempo al Vencimiento (MTV) asigna el procesador a la tarea que, estando en condiciones de ser ejecutada, le resta menos tiempo para alcanzar su vencimiento.

MTV es un algoritmo óptimo en el sentido de que si un conjunto de tareas no puede ser diagramado por MTV ninguna disciplina de prioridades lo hará [23]. La condición para determinar la factibilidad está dada por la expresión (2.3). Se observa que el único requerimiento que se tiene sobre el sistema es que el factor de utilización del mismo sea a lo sumo 1 es decir, se admite una utilización del 100% del procesador.

$$\sum_{i=1}^{m} \frac{C_i}{T_i} \leq 1 \qquad (2.3)$$

En cada instante de activación, se selecciona entre las tareas listas aquélla cuyo vencimiento esté más cercano. Para ello, el diagramador debe conocer en cada instante de activación o bien los vencimientos y el momento en que cada tarea arribó a la cola de tareas listas; o

bien directamente el tiempo que resta al vencimiento de cada una de ellas.

Si la tarea τ_i produce su j_ésima generación en el instante t_j, su prioridad en un instante t con $t_j \leq t \leq (t_j+D_i)$, será inversamente proporcional a $(t_j+D_i) - t$. El primer término de la expresión anterior determina el instante en que se producirá el vencimiento de la j_ésima generación de la tarea τ_i. Si el tiempo al vencimiento para dos o más tareas es el mismo, una regla de desempate debe ser aplicada. En la misma puede usarse cualquier criterio de selección, por ejemplo la que primero arribó, la de mayor tiempo de ejecución, etc.

Como hemos dicho, la disciplina de prioridades MTV es óptima. Cabe aclarar, que dicha condición es válida en sistemas de tiempo real monoprocesador con tareas independientes que no comparten recursos. En sistemas en donde existen recursos compartidos o relaciones de precedencia entre las tareas, y en sistemas multiprocesador la misma no se mantiene.

Además de resultar compleja la implementación de un diagramador MTV por la cantidad de información que debe considerarse para cada tarea, el comportamiento del sistema ante una falla por sobrecarga no provoca daños parciales, sino que produce una crisis global del sistema. Este fenómeno, denominado "efecto dominó" consiste en que, cuando el sistema está sobrecargado, si una tarea tiene tiempos de ejecución mayores a los previstos o una tasa de arribo superior a la predeterminada podrá adelantar un vencimiento. Por ser la más urgida podrá desplazar a todo el resto del sistema por el tiempo que dure su mal funcionamiento. Más aún,

si perdiese su vencimiento podrá causar la pérdida de los vencimientos de todas las tareas que requieran posteriormente el procesador. Este comportamiento es intolerable en muchos sistemas donde pueden ocurrir sobrecargas esporádicas debidas a errores en la determinación de las especificaciones temporales.

2.2.2.1.2.1 Ventajas del MTV

- MTV es óptima en sistemas de tiempo real monoprocesador con tareas independientes.
- Es versátil en el momento de considerar sistemas dinámicos en los que el sistema debe tomar la desición de incorporar una tarea nueva o no en función de la factibilidad del conjunto.

2.2.2.1.2.2 Desventajas del MTV

- Implementación compleja.
- La pérdida de un vencimiento puede causar la pérdida de los vencimientos de todas las tareas del sistema.

2.2.2.1.3 Menor Tiempo de Latencia

Si la tarea τ_i produce su j_ésima generación en el instante t_j (Figura 2.7), en un instante t con $t_j \leq t \leq (t_j+D_i)$ se habrá ejecutado un tiempo c_i tal que $0 \leq c_i \leq C_i$. Luego en dicho instante t la *latencia* de τ_i será $l_i(t)=[(t_j+D_i)-t] -(C_i - c_i)$. Luego, el diagramador asigna el procesador a la tarea con menor l_i. La disciplina MTL es, como MTV, óptima, con lo que la condición de factibilidad para un sistema

MTL estará dada por la Ec. (2.3). Desgraciadamente, esta disciplina conserva también las desventajas de MTV [51].

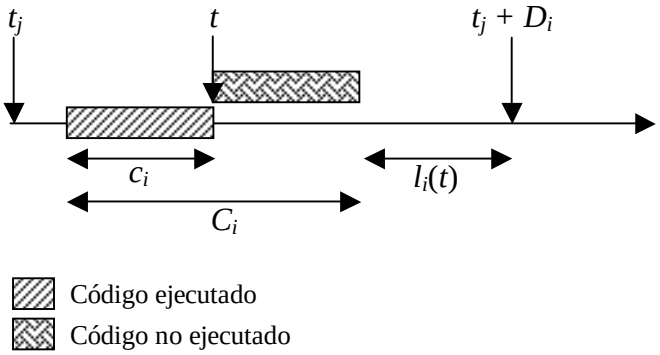


Figura: 2.7
Diagrama temporal del comportamiento de un sistema MTL.

2.2.2.2.3 Disciplinas de Prioridades Fijas.

Un usuario periódico τ_i esta caracterizado en S(m) por su mayor tiempo de ejecución C_i, su período T_i y su vencimiento D_i. Salvo que se indique lo contrario, se asume que el vencimiento de una tarea es igual a su período con lo que deberá concluirse con la ejecución de la misma antes de finalizar un período.

Mediante esta disciplina, el diagramador asigna el recurso compartido al usuario que tiene mayor prioridad dentro de los que se encuentran listos para ser activados, recorriendo la pila de prioridades en sentido decreciente de prioridad hasta encontrar al primero listo para utilizar el recurso.

Liu y Layland demuestran en **[23]** que la disciplina de prioridades fijas óptima resulta de obtener la pila de prioridades asignando

prioridades decrecientes a períodos crecientes. La misma se denomina disciplina por *períodos monotónicos crecientes* (PMC). El concepto de optimalidad implica que si un sistema es factible con una disciplina de prioridades fijas y una pila de prioridades arbitraria, lo será también por PMC. La recíproca no es cierta con lo que PMC es el mejor mecanismo de prioridades fijas.

Si bien el algoritmo de diagramación es sencillo, no lo es el análisis de la factibilidad.

En **[23]** se demuestra que un sistema de m usuarios periódicos $\mathbf{S}(m)$, que utilice un diagramador que implemente prioridades fijas PMC, es factible si:

$$U_{S(m)} = \sum_{i=1}^{m} U_i = \sum_{i=1}^{m} \frac{C_i}{T_i} \leq m\left(2^{\frac{1}{m}} - 1\right) \tag{2.4}$$

Donde U_i *y* $U_{S(m)}$ representan el factor de utilización de τ_i y del sistema $\boldsymbol{S}(m)$ respectivamente.

La expresión (2.4) da una cota superior para el factor de utilización en función del número de tareas. Esta cota, si bien es de muy fácil obtención, permite utilizaciones muy bajas del recurso cuando el número de usuarios es relativamente pequeño. A medida que la cantidad de usuarios aumenta, la cota tiende a ln 2=(0,6993). La utilización de esta cota conduce a una sub-utilización del recurso dado que el análisis conduce a conclusiones suficientes pero no necesarias.

Ejemplo 2.2: En la Figura: 2.8 se muestra un ejemplo sencillo de un sistema multiusuario-monoprocesador donde tres tareas de tiempo

real compiten por el uso del procesador. $\mathbf{S}(3)=\{(C_1,T_1),(C_2,T_2),(C_3,T_3)\}=\{(2,1),(3,1),(6,1)\} = 1/2+1/3+1/6 =1>0,78$ y, a pesar de ello, el sistema es factible.

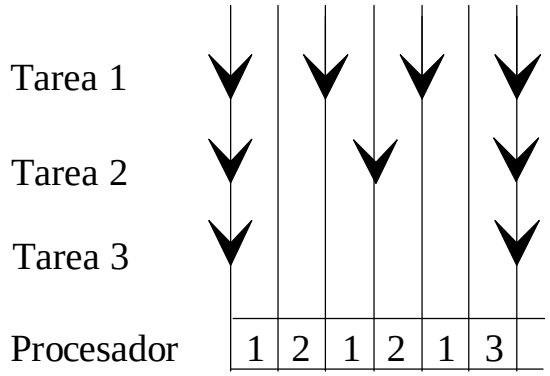


Figura: 2.8
Diagrama de asignación de un sistema de tres tareas utilizando PMC

PMC es un standard de facto adoptado por IBM, Honeywell, Mc Donnell Douglas, Boeing, General Electric, General Dinamics, Magnavox, Mitre, NASA, Naval Air Warfare Center y Paramax. Tradicionalmente, el análisis de sistemas PMC ha sido objeto de numerosas restricciones tanto en sus vencimientos y en sus períodos como así también en la posibilidad de atender usuarios esporádicos o asincrónicos para los cuales se preveía la utilización de servidores específicos para atender sus actividades. Por otro lado el análisis contemplaba sólo tareas absolutamente independientes. En el Capítulo 4 se realizará un análisis del mecanismo de prioridades fijas y se propondrán soluciones para la determinación de la factibilidad de dichos sistemas en condiciones menos pesimistas y liberando a los mismos de alguna de las restricciones antes mencionadas.

2.2.2.3.3.1 Ventajas de los diagramadores PMC

- Son de fácil implementación tanto en software como en hardware.
- Son óptimos en el contexto de prioridades fijas.
- Algunas variantes como las presentadas en esta Tesis permiten adaptarlos a diferentes circunstancias operativas.
- Una sobrecarga producida por una tarea sólo puede provocar la pérdida de vencimientos a las tareas de menor prioridad. Si el sistema es apropiativo las tareas de mayor prioridad pueden continuar su ejecución con normalidad, viéndose afectadas por la mayor utilización del procesador las que le suceden en la pila de prioridades.

2.2.2.3.3.2 Desventajas de los diagramadores PMC

- No se adaptan fácilmente a sistemas con incorporación dinámica de tareas ya que, la determinación de la factibilidad del sistema requiere el cómputo de la factibilidad de todas las tareas de menor prioridad a fin de aceptar o no la incorporación de dicha tarea a la pila.
- Si por seguridad se toman parámetros del sistema muy conservadores, el cálculo se torna muy pesimista. Esto requiere un conocimiento exacto de los parámetros del mismo para no producir una fuerte subutilización del sistema.

Ejemplo 2.3: Supongamos un sistema **S**(5) con $T_i=D_i$:

i	C_i	T_i
1	2	4
2	2	8
3	2	14
4	2	24
5	2	96

el mismo es factible y posee un factor de utilización: 0,9762. Si incrementamos el tiempo de ejecución de la tarea_2 a 2,01, $\mathbf{S}^{1}(5)$ resulta:

i	C_i	T_i
1	2	4
2	2,01	8
3	2	14
4	2	24
5	2	96

con un factor de utilización de 0.9774

Con un incremento de sólo el 0,5 % en el valor computado como tiempo de ejecución de la tarea_2, el cálculo de factibilidad del sistema (ver Capítulo 4) determina la no factibilidad del sistema. Con este nuevo valor el máximo subsistema factible de $\mathbf{S}(5)$ es $\mathbf{S}(2)$:

i	C_i	T_i
1	2	4
2	2,01	8

con un factor de utilización de sólo 0,751.

Con dicha modificación, para conservar una buena utilización del sistema y obtener un sistema factible, se requiere que los vencimientos mínimos cambien como sigue, $\mathbf{S}^{3}(5)$:

i	C_i	T_i
1	2	4
2	2,01	8
3	2	14,02
4	2	38,05
5	2	70,09

con un factor de utilización de 0.9465

Nótese que con una disciplina como MTV tanto $\mathbf{S}(5)$ como $\mathbf{S}^1(5)$, $\mathbf{S}^2(5)$ y $\mathbf{S}^3(5)$ resultan factibles.

Capítulo 3

Disciplinas de Prioridades Fijas, Método de las Ranuras Vacías

3.1. Introducción.

En el siguiente apartado se analiza el método de las ranuras vacías. Si bien el mismo ha sido desarrollado para analizar la factibilidad de sistemas diagramados por una disciplina de prioridades fijas, provee resultados generales para cualquier disciplina de prioridades [28], [39].

Se considerarán usuarios (τ_i) con requerimientos periódicos. Un conjunto de m usuarios constituyen un sistema **S**(m) caracterizado por el período de generación de los requerimientos (T_i) y sus vencimientos (D_i) que, salvo que se indique lo contrario, se asumen iguales a los períodos.

Inicialmente se asumen algunas restricciones funcionales alguna de las cuales, serán relajadas oportunamente.

3.2. Consideraciones Generales.

El tiempo se considera ranurado[3]; cada ranura T_r tiene una duración que dependerá de diferentes parámetros del sistema en consideración. La ranura es una unidad indivisible por lo que, puede considerarse al intervalo T_r como la unidad de tiempo y a los períodos múltiplos de T_r. Luego, las expresiones *inicio de la ranura t* e *instante t* pueden ser usadas indistintamente.

Si bien el desarrollo subsiguiente puede extenderse a un análisis temporal donde la unidad de tiempo no sea T_r, en general, la división del tiempo en ranuras permite describir fácilmente varias aplicaciones reales: en redes locales la comunicación se realiza fraccionando la información en mensajes los cuales tienen, dependiendo del protocolo utilizado, una conformación mínima indivisible; la generación de tareas manejada por temporizadores periódicos o interrupciones externas periódicas es otro ejemplo. Aún cuando se traten sistemas apropiativos, es posible encontrar

[3] Salvo que se indique lo contrario, esta interpretación no limita el análisis de sistemas con intervalos de activación no periódicos.

intervalos indivisibles tales como el ciclo de ejecución de una instrucción que es posible utilizarlo como unidad de tiempo.

Se dice que una ranura es una *ranura vacía* (*e*) cuando no hay requerimientos pendientes sobre el recurso y por lo tanto el mismo permanece ocioso en dicho intervalo. Esta última consideración es importante ya que el diagramador sólo podrá dejar una ranura vacía si y solo si no existen requerimientos pendientes.

3.1. El Método de las Ranuras Vacías.

En **[23]** se ha demostrado que la peor condición de carga de un sistema de tareas independientes ocurre cuando todos los usuarios producen requerimientos simultáneos. Si ello sucede en el instante t=1, la expresión

$$R_m(t) = \sum_{h=1}^{m} \left\lceil \frac{t}{T_h} \right\rceil \qquad (3.1)$$

indica el número de requerimientos generados en el intervalo cerrado [1,t]. $\lceil\ \rceil$ indica el operador monádico techo siendo $\lceil x \rceil$ el menor entero mayor o igual a x.

Luego, de (3.1), pueden inmediatamente obtenerse las siguientes conclusiones:

C1) $R(y)$ - $R(x)$ es el número de ranuras requeridas por el sistema en el intervalo [x+1, y].

C2) $\mathcal{R}(x+1) - \mathcal{R}(x)$ es el número de ranuras requeridas por el sistema en el instante $x+1$. Si el sistema es factible y $\mathcal{R}(x+1) > \mathcal{R}(x)$, el intervalo $[x+1, x+(\mathcal{R}(x+1)- \mathcal{R}(x))]$ es saturado. Si $\mathcal{R}(x+1)= \mathcal{R}(x)$, la ranura $x+1$ puede estar libre u ocupada.

C3) Si $\mathcal{R}(x) > x$, la ranura x está ocupada; si $\mathcal{R}(x) < x$, la ranura x puede estar libre u ocupada.

$\mathcal{R}(x)$ es monotónica creciente y tiene la siguiente propiedad:

Lema 3.1:

$$\mathcal{R}(x+y) = \mathcal{R}(x) + \mathcal{R}(y) - j \qquad x, y \in \mathbf{N} \quad j \in \{0,1,....,m\}$$

Demostración:

$$\left\lceil \frac{(a+b)}{z} \right\rceil = \begin{cases} \lceil a/z \rceil + \lceil b/z \rceil \\ ó \\ \lceil a/z \rceil + \lceil b/z \rceil - 1 \end{cases}$$

Por lo tanto

$$\left\lceil \frac{(a+b)}{z} \right\rceil = \left\lceil \frac{a}{z} \right\rceil + \left\lceil \frac{b}{z} \right\rceil - j \qquad j \in \{0,1\}$$

Luego

$$\mathcal{R}(x+y) = \sum_{h=1}^{m} \left\lceil \frac{(x+y)}{T_h} \right\rceil = \sum_{h=1}^{m} \left(\left\lceil \frac{x}{T_h} \right\rceil + \left\lceil \frac{y}{T_h} \right\rceil - j_h \right) =$$

$$= \sum_{h=1}^{m} \left\lceil \frac{x}{T_h} \right\rceil + \sum_{h=1}^{m} \left\lceil \frac{y}{T_h} \right\rceil - \sum_{h=1}^{m} j_h = \mathcal{R}(x) + \mathcal{R}(y) - j =$$

donde $j = \sum_{h=1}^{m} j_h$

Como $j_h \in \{0, 1\}$, $j \in \{0, 1, ..., m\}$.

Debe notarse que la peor condición de carga del sistema se produce en la ranura M+1 donde M es el mínimo común múltiplo de los períodos de **S**(m).

Para que el sistema sea factible, todos los requerimientos del mismo en el intervalo [1, M] deben ser atendidos dentro del mismo. En consecuencia, $\mathcal{A}(M) \leq M$. Si $\mathcal{A}(M)>M$ el sistema no será factible[4].

Calculando **(3.1)** en t=M

$$\mathcal{R}_m(M) = \sum_{h=1}^{m} \left\lceil \frac{M}{T_h} \right\rceil = \sum_{h=1}^{m} \frac{M}{T_h} \leq M$$

Luego

$$\sum_{h=1}^{m} \frac{1}{T_h} \leq 1 \qquad (3.2)$$

La expresión **(3.2)** indica la condición que debe cumplir todo sistema factible. Si $\mathcal{R}_m(M)=M$, se dirá que **S**(m) es *saturado*. Si $\mathcal{R}_m(M) \leq M$, existirá alguna disciplina de prioridades que haga a **S**(m) factible.

[4] Esta conclusión es extensible a cualquier disciplina de prioridades.

Dado un sistema factible no saturado sea **E**={e_1,e_2,...,e_s) el conjunto de ranuras vacías en el intervalo [1,M], donde s =M-$\mathcal{A}_m$(M). Debido a la periodicidad de los patrones de generación de requerimientos, el conjunto de ranuras vacías en [1, ∞] es {e_1+kM, e_2+kM, ... , e_s+kM} donde $k \in \{0,1,2,...\}$.

Teorema 3.1

La j-ésima ranura libre en un sistema de m usuarios es

$$e_{j(m)} = \text{menor } x \mid x = j + \mathcal{A}_m(x)$$

Demostración:

En el intervalo [1, e_i-1] hay exactamente i-1 ranuras vacías. Dado que e_i es vacía, todas las ranuras requeridas en [1, e_i-1] han sido asignadas. Por lo tanto:

$$\mathcal{A}(e_i - 1) = (e_i\text{-}1)\text{-}(i\text{-}1) = e_i - i$$

Dado que no hay requerimientos en el instante e_i:

$$\mathcal{A}(e_i - 1) = \mathcal{A}(e_i) = e_i - i$$

Resta probar que e_i es el menor $x|\mathcal{A}(x) = x\text{-}i$. Para esto es suficiente notar que el número de ranuras vacías en [1, x] para todo $x<e_i$ tienen una cota superior en i-1. Esto implica que $\mathcal{A}(x) \geq x\text{-}(i\text{-}1) > x - i$.

Corolario:

Dado un sistema factible, el conjunto de ranuras vacías que deja el mismo es independiente de la disciplina de prioridades y el mecanismo de diagramación utilizado.

Demostración:

Dado que $\mathcal{A}_m(x)$ es función sólo de los períodos del sistema, el corolario se desprende directamente a partir de la prueba del Teorema 3.1.

Teorema 3.2:

Dado un sistema factible no saturado operando bajo una disciplina de prioridades

(a) $\forall\ i\ e_{i+1}-e_i \le e_1$

(b) La ranura M es una ranura vacía.

Demostración:

Para probar (a) es suficiente demostrar que $\forall i\ e_{i+1} \in [e_i+1,\ e_i+e_1]$. El intervalo $[e_i+1,\ e_i+e_1]$ tiene e_1 ranuras, y puesto que e_i es vacía, no hay requerimientos pendientes antes del instante e_i+1. El número g de requerimientos generados en el intervalo puede ser calculado usando la conclusión C1 y el Lema 3.1:

$$g = \mathcal{A}(e_i+e_1) - \mathcal{A}(e_i) = \mathcal{A}(e_i)+\mathcal{A}(e_1)-j-\mathcal{A}(e_i)$$

$$g = \mathcal{A}(e_1)-j \qquad j \in \{0, 1, \ldots, m)$$

Como $\mathcal{A}(e_1) = e_1 - 1$

tenemos que $g = e_1 - 1 - j < e_1$

Dado que el número de ranuras requeridas es menor que el número de ranuras disponibles en el intervalo, debemos tener al menos una ranura vacía lo que prueba (a).

Para probar (b) recordamos que:

s = M-$\mathcal{A}$(M) es el número de ranuras vacías en [1, M] y

$e_{s+1} = e_1 + M$

De (a), $e_{s+1} - e_s \leq e_1$

Luego,

$e_1 + M - e_s \leq e_1 \Leftrightarrow M \leq e_s$

Como también $e_s \leq M$, esto conduce a que $e_s = M$.

Capítulo 4

Análisis de Factibilidad de Sistemas PMC

4.1. Introducción

En el Capítulo 2 se ha mencionado que uno de los problemas del el análisis de la factibilidad de sistemas PMC es el pesimismo del mismo. La expresión (2.4) da una cota superior para el factor de utilización en función del número de tareas. Esta cota, si bien es de muy fácil obtención, permite utilizaciones muy bajas del recurso. En lo que sigue se discutirá el análisis de factibilidad de sistemas de tiempo real operando con una disciplina de prioridades fijas con un

factor de utilización que puede superar dicha cota. Se demuestra además una condición necesaria y suficiente para ello.

4.2. Sistemas PMC

Supongamos un sistema factible, no saturado $\mathbf{S}(m\text{-}1)=\{\tau_1, \tau_2,\ldots., \tau_{m-1}\}$ periódico y con tiempos de utilización unitarios, operando bajo una Disciplina por Prioridades Fijas (DPF).

¿Bajo que condiciones es posible expandir $\mathbf{S}(m\text{-}1)$, factible, a un sistema $\mathbf{S}(m)=\{\tau_1, \tau_2,\ldots., \tau_m\}$ también factible agregando un usuario τ_m en la base de la pila de prioridades?

Teorema 4.1:

Sea $\mathbf{S}(m\text{-}1)=\{\tau_1, \tau_2,\ldots., \tau_{m-1}\}$ factible y no saturado operando bajo una DPF y sea $e_{1(m-1)}$ la primer ranura libre de $\mathbf{S}(m\text{-}1)$. El sistema $\mathbf{S}(m)$ resultante de agregar un usuario en la base de la pila será diagramable sss $T_m \geq e_{1(m-1)}$.

Demostración:

Sea $\mathbf{S}(m\text{-}1)$ factible y no saturado y $T_m \geq e_{1(m-1)}$. El usuario τ_m generará sus requerimientos en los instantes 1, T_m+1, $2.T_m+1$,... es decir en $k.T_m+1$ con $k \in \{0 \cup \mathbf{N}\}$. Para demostrar que $\mathbf{S}(m)$ es factible es suficiente probar que la generación de cada requerimiento en los instantes $k.T_m+1$ encuentran una ranura disponible para su asignación en $[k.T_m+1, (k+1)T_m]$. La longitud del mismo es $T_m \geq e_{1(m-1)}$. En el Teorema 3.2 se ha probado que dicho intervalo contiene al menos

una ranura libre en **S**(m-1) con lo que, dicha ranura libre podrá ser ocupada por τ_m; por lo tanto, **S**(m) es factible.

De la misma manera, si **S**(m) es factible, es claro que **S**(m-1) es factible y no saturado y el usuario τ_m de más baja prioridad, $T_m \geq e_{1(m-1)}$.

Teorema 4.2:

Si **S**(m) es tal que min(**T**(m)) ≥ m, **S**(m) es factible independientemente de como se ordene la pila de prioridades de la DPF.

Demostración:

Sea **S**(m)=$\{\tau_1, \tau_2, \ldots, \tau_m\}$, min(**T**($m$)) ≥ m ordenados arbitrariamente. Procediendo por inducción sobre m, si m=1 el sistema es obviamente factible. Si m>1, supongamos válido el teorema para cualquier conjunto de m-1 usuarios que satisfagan la condición relativa al número de usuarios y al min(**T**(m)). Luego, el subsistema **S**(m-1)=$\{\tau_1, \tau_2, \ldots, \tau_{m-1}\}$ es factible dado que m-1<m≤min(**T**(m-1)). Luego, por el teorema 3.1, la primer ranura libre que deja el subsistema **S**(m-1) es $e_{1(m-1)}$=m. Dado que min(**T**(m))≥ m, por el Teorema 4.1 se concluye que **S**(m) es factible..

Teorema 4.3:

Si **S**(m) es tal que min(**T**(m))<m, luego **S**(m) es factible sss para todo $i \in$ [min(**T**(m))+1,…,m] se verifica

$$\sum_{h=1}^{i-1} \frac{1}{T_h} \leq 1 \qquad (4.1)$$

y

$$T_i \geq \min\ t \in \left[\min(\mathbf{T}(m)), T_i\right] | t = 1 + \sum_{h=1}^{i-1} \left\lceil \frac{t}{T_h} \right\rceil = e_{1(i-1)} \qquad (4.2)$$

Demostración:

S(m) será factible si lo son los subsistemas **S**(1), **S**(2),...,**S**(m). Por el Teorema 4.2, los subsistemas **S**(1),...,**S**(min(**T**(m))) son factibles. Para verificar la factibilidad de **S**(min(**T**(m))+1),...,**S**(m), razonaremos recursivamente sobre i. Supongamos que **S**(i-1) es factible y que se cumplen las inecuaciones (4.1) y (4.2). Como se cumple (4.1), **S**(i-1) es no saturado por lo que puede expandirse a un sistema de i usuarios. Aplicando los Teoremas 3.1 y 4.1, si (4.2) se cumple, **S**(i) es factible.

La prueba es constructiva en el sentido que indica un procedimiento para probar la factibilidad de un sistema operado por una DPF. En **[23]** se ha demostrado que si un sistema es factible bajo una DPF lo es bajo la disciplina por períodos monotónicos crecientes PMC. Luego, para analizar la factibilidad de un sistema, resultará conveniente asignar prioridades a los usuarios por PMC. Si el sistema resultase no factible, no lo será bajo ninguna DPF.

Por otro lado, el método permite identificar el mayor subsistema factible. Esto permitiría al diseñador del sistema considerar la alternativa de reducir el número de usuarios del sistema de resultar el mismo no factible en alguna instancia del proceso de diseño.

4.3. Sistemas PMC con Tiempos de Utilización no Unitarios, Apropiativos no Cooperativos.

Se dice que un sistema es apropiativos si un usuario puede ser interrumpido en la utilización del recurso (en unidades enteras de ranuras) por otro usuario de mayor prioridad. Se dirá luego que es perfectamente apropiativo si este fraccionamiento puede realizarse en intervalos de una ranura. El no considerar la cooperación entre usuarios inhibe la ocurrencia de bloqueos.

Teorema 4.4:

Si el número de ranuras solicitadas por período por un usuario τ_i es $C_i \geq 1$, la j-ésima ranura libre en un sistema de m usuarios es

$$e_j(m) = \text{menor } t \mid t = j + W_m(t) \tag{4.3}$$

donde

$$W_m(t) = \sum_{h=1}^{m} C_i \left\lceil \frac{t}{T_i} \right\rceil \tag{4.4}$$

indica el número de ranuras requeridas en el intervalo [1, t].

Demostración:

$\mathbf{S}(m) = \{\tau_1, \tau_2, ..., \tau_m\}$, con $\mathbf{C}(m)=\{C_1, C_2, ..., C_m\}$ y $\mathbf{T}(m)=\{T_1, T_2, ..., T_m\}$, es equivalente a $\mathbf{S}^*(m') = \{\tau_1^*, \tau_2^*, ..., \tau_{m'}^*\}$ con $m' = \sum_{h=1}^{m} C_h$, $\mathbf{C}^*(m') = \{C_i^* = 1,\ i \in [1, m']\}$,

$$\mathbf{T}^*(m') = \left\{ \begin{array}{l} \left(\forall i \in [1, C_1], T_i^* = T_1\right), \left(\forall i \in [C_1+1, C_1+C_2], T_i^* = T_2\right), \\ \ldots\ldots\ldots\ldots\ldots, \left(\forall i \in [1+\sum_{h=1}^{m-1} C_h, m'], T_i^* = T_m\right) \end{array} \right\}$$

Aplicando el Teorema 3.1 a $\mathbf{S}^*$

$e_j(m')$ = menor $t \mid t = j + \mathcal{R}_{m'}(t)$ =

$$= j + \sum_{h=1}^{m'} \left\lceil \frac{t}{T_h^*} \right\rceil = j + \sum_{h=1}^{C_1} \left\lceil \frac{t}{T_h^*} \right\rceil + \sum_{h=C_1+1}^{C_1+C_2} \left\lceil \frac{t}{T_h^*} \right\rceil + \ldots + \sum_{h=m'-C_m+1}^{m'} \left\lceil \frac{t}{T_h^*} \right\rceil =$$

$$= j + C_1 . \left\lceil \frac{t}{T_1} \right\rceil + C_2 . \left\lceil \frac{t}{T_2} \right\rceil + \ldots + C_m . \left\lceil \frac{t}{T_m} \right\rceil = j + \sum_{h=1}^{m} C_h . \left\lceil \frac{t}{T_h} \right\rceil = j + W_m(t)$$

Luego

$e_{j(m)}$ = menor $t \mid t = j + W_m(t)$

Para que el sistema pueda resultar factible, todas las ranuras requeridas en el intervalo [1, M] deben ser asignadas dentro del mismo. En consecuencia, $W_m(\mathrm{M}) \leq \mathrm{M}$. Si $W_m(\mathrm{M}) > \mathrm{M}$ el sistema no podrá ser factible bajo ninguna disciplina de prioridades.

Calculando **(3.1)** en t=M para $\mathbf{S}^*$

$$\mathcal{R}_m(\mathrm{M}) = \sum_{h=1}^{m'} \left\lceil \frac{\mathrm{M}}{T_h^*} \right\rceil = \sum_{h=1}^{m} C_h \left\lceil \frac{\mathrm{M}}{T_h} \right\rceil = \sum_{h=1}^{m} C_h \left\lceil \frac{\mathrm{M}}{T_h} \right\rceil = \sum_{h=1}^{m} C_h . \frac{\mathrm{M}}{T_h} \leq \mathrm{M}$$

Con lo que

$$\sum_{h=1}^{m} \frac{C_h}{T_h} \leq 1 \qquad (4.5)$$

Teorema 4.5:

Sea $\mathbf{S}(m\text{-}1)=\{\tau_1, \tau_2,\ldots., \tau_{m-1}\}$, $\mathbf{C}(m\text{-}1)=\{C_1, C_2,\ldots., C_{m-1}\}$, $C_i \geq 1$, factible y no saturado operando bajo una DPF y sea $e_{j(m-1)}$ la j-esima ranura libre de $\mathbf{S}(m\text{-}1)$. El sistema $\mathbf{S}(m)$ resultante de agregar un usuario en la base de la pila será diagramable sss $T_m \geq e_{j(m-1)}$ con $j=C_m$.

Demostración:

Sea $\mathbf{S}(m\text{-}1)$ factible y no saturado y $T_m \geq e_{C_m(m-1)}$. El usuario τ_m generará sus requerimientos en los instantes 1, T_m+1, 2.T_m+1,... es decir en $k.T_m$+1 con $k \in \{0 \cup \mathbf{N}\}$. Para demostrar que $\mathbf{S}(m)$ es factible es suficiente probar que el número de ranuras solicitadas en la generación de cada requerimiento en los instantes $k.T_m$+1 encuentran C_m ranuras disponibles para su asignación en $[k\ T_m+1, (k+1)T_m]$. La longitud del intervalo es $T_m \geq e_{C_m(m-1)}$. Utilizando los resultados del Teorema 3.2 y el razonamiento del Teorema 4.4, se concluye fácilmente que dicho intervalo contiene al menos C_m ranuras libres de $\mathbf{S}(m\text{-}1)$ con lo que, las mismas podrán ser ocupadas por τ_m; por lo tanto, $\mathbf{S}(m)$ es factible.

De la misma manera, si $\mathbf{S}(m)$ es factible, es claro que $\mathbf{S}(m\text{-}1)$ es factible y no saturado y para el usuario τ_m de mas baja prioridad, $T_m \geq e_{C_m(m-1)}$.

Teorema 4.6:

Si $\mathbf{S}(m)$ es tal que $\min(\mathbf{T}(m)) \geq \sum_{h=1}^{m} C_h$, $\mathbf{S}(m)$ es factible independientemente de como se ordene la pila de prioridades de la DPF.

Demostración:

Sea $\mathbf{S}(m)=\{\tau_1,\tau_2,\ldots,\tau_m\}$, $\mathbf{C}(m)=\{C_1,C_2,\ldots,C_m\}$, $\min(\mathbf{T}(m)) \geq \sum_{h=1}^{m} C_h$, $C_i \geq 1$, ordenados arbitrariamente. Procediendo por inducción sobre m, si $m=1$ el sistema es obviamente factible. Si $m>1$, supongamos válido el teorema para m-1 usuarios cumpliendo con la condición relativa al número de usuarios y al $\min(\mathbf{T}(m))$. Luego el subsistema $\mathbf{S}(m\text{-}1)$ es factible dado que $\sum_{h=1}^{m-1} C_h < \sum_{h=1}^{m} C_h \leq \min(\mathbf{T}(m\text{-}1))$. Luego, por el teorema 4.4, $\mathbf{S}(m\text{-}1)$ deja C_m ranuras libres en $e_{C_m(m-1)} = \sum_{h=1}^{m} C_h$ con lo que, en el intervalo [1, T_m], habrán al menos C_m ranuras libres dejadas por $\mathbf{S}(m\text{-}1)$ y, por el Teorema 4.5, se concluye que $\mathbf{S}(m)$ es factible.

Teorema 4.7:

Si $\mathbf{S}(m)$ es tal que $\min(\mathbf{T}(m)) < \sum_{h=1}^{m} C_h$, luego $\mathbf{S}(m)$ es factible sss para todo i se verifica

$$\sum_{h=1}^{i-1} \frac{C_h}{T_h} < 1 \qquad (4.6)$$

y

$$T_i \geq \min\ t \in \left[\min(\mathbf{T}(m)), T_i\right] | t = C_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil = e_{C_i(i-1)} \qquad (4.7)$$

Demostración:

S(m) será factible si lo son los subsistemas **S**(1), **S**(2),...,**S**(m). Por el Teorema 4.6, los subsistemas **S**(1),...,**S**(min(**T**(m))) son factibles. Para verificar la factibilidad de **S**(min(**T**(m))+1,...,**S**(m)), razonaremos recursivamente sobre i. Asumamos que **S**(i-1) es factible y las expresiones (4.6) y (4.7) se verifican. De (4.6), **S**(i-1) es no saturado por lo que puede expandirse a un sistema de i usuarios. Aplicando los Teoremas 4.4 y 4.5, si (4.7) se cumple, **S**(i) es factible.

Se ha demostrado que un sistema **S**(m) de tareas periódicas e independientes será factible bajo la disciplina PMC sss se verifican las expresiones (4.6) y (4.7), donde la primera desigualdad indica la condición para que un subsistema **S**(i-1) admita la incorporación de un nuevo usuario y la segunda la condición que deben cumplir el período del usuario τ_i para que el mismo pueda utilizar el recurso antes de alcanzar su vencimiento.

4.4. Prioridades Fijas VMC.

En el análisis previo se ha considerado que **T**(m)=**D**(m). Sin embargo, el caso más general es con $T_i \geq D_i$.

Las expresiones (4.6) y (4.7) son función del requerimiento que realizan los usuarios sobre el sistema lo cual depende del período de los mismos y del número de ranuras solicitadas por período. Por otro

lado, la factibilidad del mismo dependerá de que las *C* ranuras solicitadas por cada usuario sean asignadas antes de alcanzado el vencimiento. Por lo tanto, la expresión (4.7) tomará la forma:

$$T_i \geq D_i \geq \min\ t \,|t = C_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil = e_{C_i(i-1)} \qquad (4.8)$$

De lo anterior se desprende que la pila óptima para un sistema cuyos vencimientos son menores que sus períodos resulta de ordenar la misma por vencimientos monotónicos crecientes (VMC) ya que se asigna mayor prioridad a las tareas con vencimientos más críticos. Notemos que el Teorema 4.7 es válido para cualquier disciplina de prioridades fijas por lo tanto la expresión (4.8) es aplicable a un sistema con una DPF ordenada VMC.

4.5. Diagramación de Sistemas PMC con Inversión de Prioridades.

La próxima generación de sistemas de tiempo real deberá afrontar el desafío de manejar sistemas con comportamientos cambiantes. Si bien la disciplina de Prioridades Fijas se ha vuelto un standard de facto, la misma permanece invariante en su versión original [2]. Sin embargo es posible encontrar sistemas en los cuales el cumplimiento estricto de una disciplina PMC se realiza luego de un intervalo en el cual se produce un cierto número de asignaciones aleatorias. Este fenómeno es común en los diagramadores distribuidos de redes locales donde, luego de un transitorio de establecimiento con asignaciones aleatorias se consigue un estado estacionario PMC. Este

comportamiento aleatorio puede producir *inversiones de prioridad* en el sentido de que a un usuario de baja prioridad se le asigna el recurso cuando compite con otro usuario de más alta prioridad [16]. Para poder tratar sistemas de este tipo es necesario reformular algunos aspectos de la teoría de diagramación a fin de poder determinar la factibilidad de los mismos.

4.5.1. Sistemas *k*-diagramables

Si un sistema es diagramable aún cuando en el intervalo [1, *k*] se produzcan inversiones de prioridad y, a partir de la ranura *k* +1 el diagramador cumple estrictamente con la disciplina PMC, se dirá que el sistema es *k*-diagramable [28].

A continuación se demuestran las condiciones necesarias y suficientes para que un sistema sea *k*-diagramable.

Teorema 4.8:

Un sistema **S**(*m*) es *k*-diagramable sss

$$\forall i \in \{1,2,...,(m-I)\} \qquad T_i \geq e_{(C_i+k)(i-1)} \tag{4.9}$$

donde

$$I = \max_{1 \leq s \leq m} s \mid \sum_{h=m}^{m-s+1} \frac{C_h}{k} \leq 1$$

Demostración:

Supongamos el peor caso de carga y el peor caso de inversión de prioridades para cada usuario. Los primeros *m-I* usuarios son los únicos que pueden ser afectados por *k* inversiones de prioridad y, en consecuencia, podrán utilizar el recurso cumpliendo con su restricción de tiempo si su plazo de vencimiento es mayor o igual que la primera ranura que dejan libre los usuarios que los preceden en prioridad, a los cuales deben sumarse las *k* ranuras ocupadas por los usuarios que se vieron atendidos por inversión de prioridad.

Por lo tanto:

$$\forall i \in [1, m-I]$$

$$T_i \geq \text{ menor } t \mid t = C_i + W_t(i-1) + k = e_{(C_i+k)(i-1)} \tag{4.10}$$

para $i=m\text{-}I$

$$T_{m-I} \geq \text{menor t} \mid \text{t} = C_{m-I} + k + \sum_{h=1}^{m-I-1} C_h \cdot \left\lceil \frac{t}{T_h} \right\rceil =$$

$$= C_{m-I} + k + C_1 \left\lceil \frac{t}{T_1} \right\rceil + C_2 \left\lceil \frac{t}{T_2} \right\rceil + C_{m-I-1} \left\lceil \frac{t}{T_{m-I-1}} \right\rceil = t^* \tag{a}$$

Los usuarios con $i > m\text{-}I$ pueden sufrir sólo $k - \sum_{h=m-I}^{m} C_h$ inversiones que es el número máximo de ranuras ocupables por los usuarios de menor prioridad. Luego para $i=m\text{-}I+1$

$$T_{m-I+1} \geq \text{menor } t \mid t = C_{m-I+1} + (k - C_{m-I+1}) + \sum_{h=1}^{m-I} C_h . \left\lceil \frac{t}{T_h} \right\rceil =$$

$$= k + C_1 \left\lceil \frac{t}{T_1} \right\rceil + C_2 \left\lceil \frac{t}{T_2} \right\rceil + C_{m-I-1} \left\lceil \frac{t}{T_{m-I-1}} \right\rceil + C_{m-I} \left\lceil \frac{t}{T_{m-I}} \right\rceil \qquad \text{(b)}$$

Ahora bien, $t=t^*$ es solución de (a) y $t^* \leq T_{m-I}$. Por lo tanto $C_{m-I} . \lceil t^*/T_{m-I} \rceil = C_{m-I}$. En consecuencia t^* es la menor solución de (b) con lo que resulta suficiente que $T_{m-I+1} \geq T_{m-I}$. Siguiendo este razonamiento hasta $i=m$ resulta que para $i > m-I$ basta solamente que **S**(m-I) sea k-diagramable y $T_{i+1} \geq T_i$. Como esta última condición es necesaria en un sistema PMC, luego **S**(m) será k-diagramable sss

$$\forall i \in \{1,2,\ldots,(m-I)\} \qquad T_i \geq e_{(C_i+k)(i-1)} \qquad (4.11)$$

donde

$$I = \max_{1 \leq s \leq m} s \mid \sum_{h=m}^{m-s+1} \frac{C_h}{k} \leq 1$$

Se pueden extender las consideraciones anteriores a sistemas en los cuales las asignaciones aleatorias no sólo se producen en el intervalo [1,k] sino también en las k ranuras siguientes a aquélla en la que se han satisfecho todos los requerimientos pendientes. A estas ranuras las denominaremos *singularidades*.

Lema 4.1:

Si un sistema es *k*-diagramable, lo será aún si el diagramador produce *k* inversiones de prioridad luego de cada singularidad.

Demostración:

Si $\mathbf{E}(t)$ es el conjunto de ranuras libres del sistema en el intervalo [1, *t*] y $|\mathbf{E}(t)|$ es la cardinalidad del mismo, las singularidades indicadas verifican

$$W_m(t) + |\mathbf{E}(t)| = t \tag{4.12}$$

La ecuación (4.12) indica que, independientemente del mecanismo de prioridades utilizado, la ranura *t*, que es igual al número de ranuras ocupadas más las ranuras libres en el intervalo [1, *t*] es una ranura en la que se han satisfecho todas los requerimientos del sistema $\mathbf{S}(m)$. Luego, la ranura siguiente a una singularidad es funcionalmente idéntica a la ranura 1 sólo que con una carga menor o igual a la de ella. Por lo tanto, si un sistema es *k*-diagramable con una asignación aleatoria a partir del instante de máxima carga (ranura 1) lo será si se producen *k* inversiones luego de cada singularidad.

4.5.2. Sistemas *k*-diagramables de Requerimiento Mixto

En esta sección se analizará la aplicación de las discusiones anteriores al diagramador de un kernel de tiempo real en el cual coexistan tareas sincrónicas y asincrónicas. Un diagramador es en

este contexto es una porción del kernel de un sistema operativo encargada de implementar y manejar colas de asignación de tareas a la CPU cuando es invocado por una interrupción de una fuente externa o un timer.

En sistemas de tiempo real, la periodicidad o aperiodicidad de los requerimientos permite clasificarlos en tres clases: *esporádicos de tiempo real* (alarmas, manejadores de errores críticos etc.), *sincrónicos de tiempo real* y *asincrónicos* (requerimientos no periódicos y sin un vencimiento definido).

Considerando una división más gruesa podemos dividir las tareas en sincrónicas y asicrónicas. En las primeras es imprescindible cumplir con su vencimiento sin importar el momento dentro del intervalo $[a*T_i,\ a*T_i + D_i]$ ($a \in \mathbf{N}$) en que el requerimiento es atendido. Para tareas asincrónicas, si bien no es necesario cumplir con un vencimiento, es preferible que las mismas sean atendidas lo más rápido posible dentro de las constricciones del sistema de tiempo real. El *tiempo de respuesta* de una tarea asincrónica será medido como el tiempo transcurrido desde el momento en que la misma se encuentra *lista* para ser ejecutada y el momento en que la misma es activada.

A fin de analizar el sistema se toma un modelo simplificado de diagrama de estados de un sistema operativo[6] (Figura 4.1). En el mismo, una tarea podrá encontrarse en estado *activo* si le fue asignada la CPU, en estado de *listo* si se encuentra en condiciones de ser ejecutada y está esperando por la CPU y *suspendido* cuando el requerimiento anterior ha sido completamente satisfecho y se encuentra esperando la ocurrencia de un evento periódico y con

vencimiento definido para tareas sincrónicas y aperiódico y sin vencimiento para las asincrónicas que la pase al estado listo. En realidad, la diversidad de eventos que pueden provocar una transición de una tarea del estado de suspendida al de listo hace necesaria la implementación de diferentes colas: tareas esperando eventos periódicos (lectura de la medición de un sensor, generación de una salida a un actuador son ejemplos típicos) y tareas esperando eventos asincrónicos (manejo de un teclado).

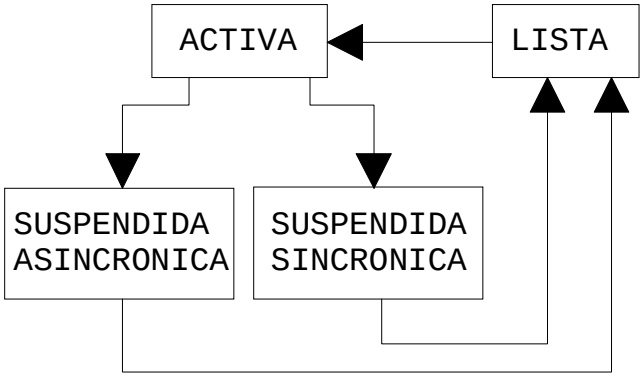


Figura 4.1
Modelo simplificado de diagrama de estados de un sistema operativo.

Cada una de estas colas es implementada como una lista enlazada de los bloques de control de tarea (TCB) los que contienen el identificador de la tarea, el estado en que se encuentra, su prioridad dentro del sistema, el contexto de la misma (descriptor del proceso) y un enlace a la próxima tarea (Figura 4.2).

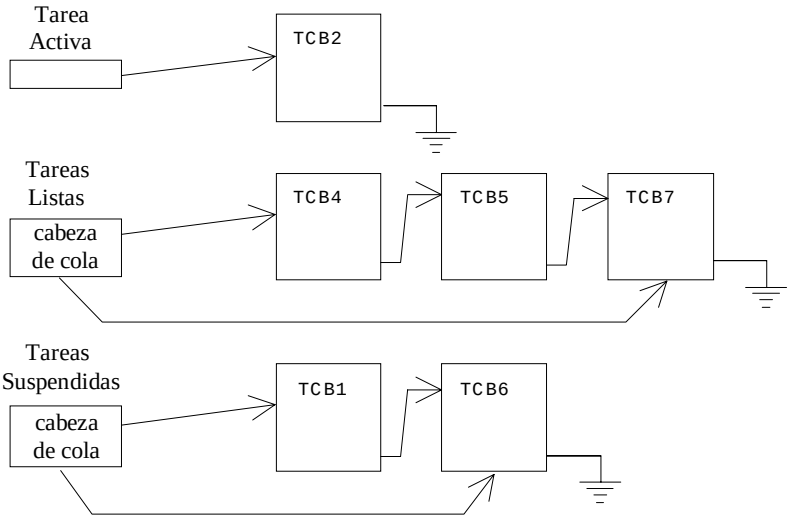


Figura 4.2
Estructura de colas de un sistema operativo

Ejemplo 4.4: En la Figura 4.3 se muestra el comportamiento de un sistema $\mathbf{S}(2)=\{(C_1,T_1=D_2),\ (C_2,T_2=D_2)\} = \{(2,3),\ (2,6)\}$ y la asignación para cada ranura con un mecanismo PMC. Las flechas verticales indican el instante de generación de cada requerimiento.

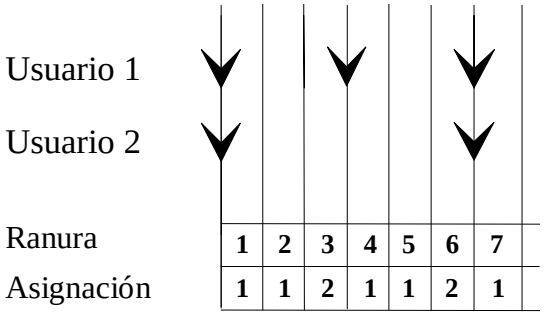


Figura 4.3
Evolución de un sistema PMC: $S(2) = \{(C_1,T_1=D_2),\ (C_2,T_2=D_2)\} = \{(2,3),\ (2,6)\}$

En la Tabla 4.1 se muestra, para cada ranura, el estado en que se encuentran las tareas del ejemplo 4.4 en el intervalo [1, 6].

Ranura	Activa	Listo	Espera
1	1	2	
2	1	2	
3	2		1
4	1	2	
5	1	2	
6	2		1

Tabla 4.1
Tabla de estados del ejemplo 4.4

Un mecanismo para atender a un sistema de requerimiento mixto consiste en un diagramador que utilice mecanismos de inversión de prioridad para asignar el procesador a *k* requerimientos asincrónicos luego de detectada una singularidad. De esta manera el requerimiento asincrónico es atendido rápidamente sin afectar al sistema de tiempo real. Aplicado a un sistema monoprocesador-multitarea, puede implementarse una cola de tareas sincrónicas listas sobre la cual se aplica estrictamente una disciplina PMC y otra de tareas asincrónicas listas con asignación aleatoria (o cualquier otra disciplina de fácil implementación). Cuando la primera es vaciada (singularidad), las siguientes *k* ranuras pueden ser asignadas, si las hubiese, a n tareas con tiempo de utilización C_i cuya suma sea menor o igual a *k*, de la segunda cola independientemente de la evolución de la cola sincrónica.

Teorema 4.9

Dado un sistema **S**(*m*), diagramable no saturado de *m* usuarios sincrónicos de tiempo real será posible adicionar *I* usuarios

asincrónicos utilizando un diagramador con inversión de prioridad si $\mathbf{S}'(m+I)=\mathbf{S}(m)\cup\mathbf{A}(I)$ es k-diagramable, donde $\mathbf{A}(I)$ son los I usuarios asincrónicos adicionados a $\mathbf{S}(m)$.

Demostración:

si $\mathbf{S}'(m+I)= \mathbf{S}(m) \cup \mathbf{A}(I)=\{s_1, s_2,..., s_m, a_1, a_2,..., a_I\}$

$\mathbf{D}'(m+I) = \{D_1, D_2,..., D_m\}$

y

$\mathbf{T}'(m+I)=\{T_1, T_2,...,T_m\}$

$\mathbf{D}'(m+I) = \mathbf{D}(m)$ y $\mathbf{T}'(m+I)= \mathbf{T}'(m)$ ya que los usuarios del conjunto $\mathbf{A}(I)$ no tienen vencimiento ni período definido.

Aplicando el Teorema 4.8 ec. (4.11), sobre $\mathbf{S}'(m+I)$

$$\forall i \in \{1,2,...,(m-I)+I\} = \{1, 2,...,m\}$$
$$T_i \geq e_{(k+C_i)(i-1)} \qquad (4.13)$$

Luego si $\mathbf{S}'(m)$ verifica (4.13), es k-diagramable independientemente de la frecuencia que registren los requerimientos de $\mathbf{A}(I)$ y podrá ser atendido por una disciplina por inversión de prioridad.

Para encontrar el máximo número de inversiones compatible con un sistema sincrónico $\mathbf{S}(m)$, es posible obtener el máximo k que hace el sistema $\mathbf{S}(m)$ k-diagramable. Si $k = 0$ el diagramador deberá atender al sistema estrictamente con una disciplina PMC. Si $k\geq1$ es

posible incorporar $I = \max_{1 \le s \le m} s \mid \sum_{h=m}^{m-s+1} \frac{C_h}{k} \le 1$ usuarios asincrónicos con inversión de prioridad.

La técnica para la obtención del máximo *k* se muestra en el siguiente ejemplo.

Ejemplo 4.5:

S(3)={1,2,3}, T(3)= D(3)={3,5,8}, C(3)={1,1,1}

Para *k*=1, **S**(4)={1,2,3,A_1}, aplicando la ec. (4.13)

$$T_i \ge e_{(k+1)(i-1)}$$
$$3 \ge 1+1 = 2$$
$$5 \ge menor\ t | t = 2 + \left\lceil \frac{t}{3} \right\rceil = 3$$
$$8 \ge menor\ t | t = 2 + \left\lceil \frac{t}{3} \right\rceil + \left\lceil \frac{t}{5} \right\rceil = 5$$

Para *k*=2, **S**(4)={1,2,3,A_1}, aplicando la ec. (4.13)

$$T_i \ge e_{(k+1),(i-1)}$$
$$3 \ge 2+1 = 3$$
$$5 \ge menor\ t | t = 3 + \left\lceil \frac{t}{3} \right\rceil = 5$$
$$8 \ge menor\ t | t = 3 + \left\lceil \frac{t}{3} \right\rceil + \left\lceil \frac{t}{5} \right\rceil = 8$$

Luego *k*=2 es el máximo número de inversiones que admite **S**(*m*).

El número de tareas asincrónicas que podrán ser atendidas es función de su tiempo de ejecución. Es posible incorporar una tarea asincrónica con *C*=2 o dos tareas asincrónicas con *C* = 1 a fin de que los requerimientos asincrónicos puedan ser atendidos con el mínimo

tiempo de respuesta. En general el conjunto de tareas asincrónicas **A**(1, 2, ...,*I*) debe cumplir con $\sum_{h=1}^{I} C_h \leq k$

En la Figura 4.4 se muestra el comportamiento del sistema del ejemplo 4.5 con I=1 y C=2. Las flechas verticales indican el instante de generación de un requerimiento. Las ranuras marcadas ↑ indican los puntos singulares de **S**(m) a partir de los cuales se producen asignaciones aleatorias y α indica la asignación del procesador en cada ranura.

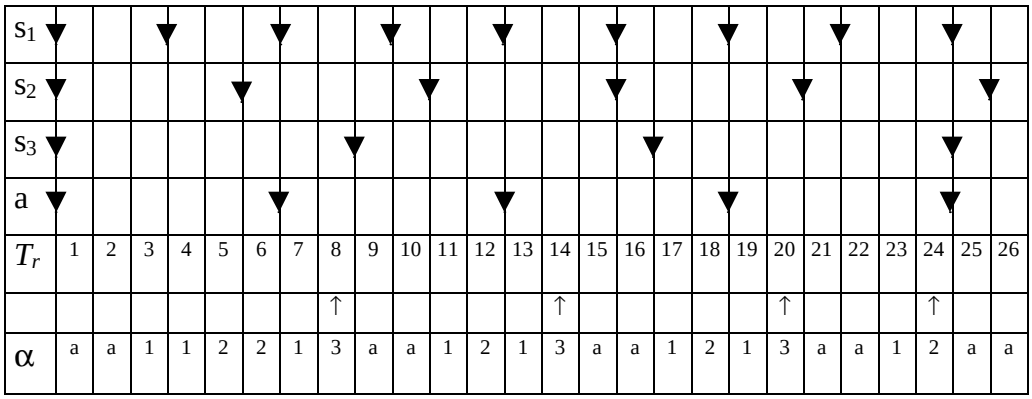


Figura 4.4
Evolución del sistema del ejemplo 4.5.

4.5.3. Analisis de la Diagramabilidad de Sistemas PMC con Bloqueos

La teoría de diagramación de STR por PMC considera que las tareas del sistema comparten un único recurso (v.g. el procesador). En la práctica, sin embargo, lo más frecuente es que compartan otros recursos como los dispositivos de entrada/salida, datos o porciones de código, sobre los cuales deben utilizarse mecanismos de protección a fin de garantizar su consistencia e integridad. Son necesarios por lo tanto mecanismos de exclusión mutua que aseguren que las tareas accedan a una zona compartida en forma no simultánea. A este tipo de región se las denomina zonas críticas. Estas zonas no son apropiables, de modo que una vez que una tarea empezó a usar una de ellas no puede ser desplazada por otra tarea, aunque sea de mayor prioridad.

Todo lo anterior produce inversiones de prioridad cuya duración puede producir la caída del sistema al no respetar los vencimientos. El ejemplo típico está dado por un sistema **S**(3) en el cual, en un instante dado, la tarea τ_1 ha completado un ciclo de ejecución. La tarea τ_3 que tiene ejecución pendiente accede entonces al procesador y a una zona de datos compartidos. Mientras se encuentra en ella, la tarea τ_1 debe ser ejecutada de nuevo y se apropia del procesador con lo cual se interrumpe la ejecución de τ_3. En ese momento despierta τ_2 que sólo usa el procesador pero no puede acceder a él porque se está ejecutando τ_1 que es de mayor prioridad. τ_1 requiere entonces el uso de los datos compartidos, pero su acceso le está vedado porque τ_3

aún no los liberó, con lo cual la ejecución de τ_1 queda suspendida. τ_2 accede entonces al procesador y puede tenerlo por tiempo indefinido por cuanto tanto τ_1 como τ_3 tienen su ejecución suspendida. Se trata de una clara inversión de prioridades que puede ser tan larga como lo sea el tiempo de ejecución de τ_2. Una primera solución al problema consiste en negar la apropiación del procesador cuando una tarea de menor prioridad se encuentra en una zona crítica. Esto degrada la performance del sistema pues tareas en las que no se produce bloqueo pierden tiempo de ejecución.

Mejores soluciones son el protocolo de prioridad heredada (PPH) y el protocolo techo (PT) desarrollados por Sha et al (1990) a los cuales es aplicable el método de diagramación por ranuras vacías anteriormente descripto.

La idea básica del protocolo de prioridad heredada es que cuando una tarea bloquea otra u otras de mayor prioridad, durante el tiempo que dura el bloqueo su ejecución se realiza con la prioridad de la tarea de mayor prioridad bloquedada. Se dice entonces que la tarea de menor prioridad heredó la prioridad mayor.

Como ya expresáramos, un mecanismo de protección de las zonas críticas son los semáforos binarios. Los mismos, notados s, son cerrados o abiertos por operaciones atómicas P(s) y V(s). Cuando una tarea entra en una zona crítica, cierra el semáforo y sólo ella puede abrirlo, cosa que hace al abandonar la zona.

La porción de código de una tarea τ_i comprendida entre operaciones P(s) y V(s) representa la sección crítica de la tarea τ_i controlada por el semáforo s cuya duración es medida en término de

ranuras. Sha [47] demostró que, con el protocolo de prioridades heredadas, una tarea τ_i puede estar bloqueada a lo sumo por sólo una sección critica de cada uno de los semáforos que comparte.

Como consecuencia de todo lo anterior, en el momento de la compilación es posible calcular el peor caso de bloqueo para cada tarea. Sin embargo, a pesar de ser mejor que el protocolo PMC puro, el protocolo de prioridades heredadas no evita las condiciones de bloqueo total o los bloqueos encadenados. El bloqueo total conduce a una paralización del sistema. Si una tarea está en posesión de un recurso que no puede liberar antes de apropiarse de otro y, si éste está en poder de otra tarea que no puede liberarlo hasta no apropiarse del recurso en poder de la primera, se produce un bloqueo total, a veces denominado abrazo mortal. Los bloqueos encadenados se producen cuando una tarea de alta prioridad es bloqueada sucesivamente por las restantes tareas de menor prioridad. Aunque cada una de ellas pueda hacerlo sólo una vez, en el peor caso la tarea de mayor prioridad será bloqueada por la suma de las duraciones más largas de bloqueo de cada una de las tareas de menor prioridad, lo que es equivalente a incrementar su tiempo de ejecución en esa duración.

El protocolo techo extiende el protocolo de prioridades heredadas y corrige ambas deficiencias. La idea básica es asignar a cada semáforo la prioridad de la tarea de más alta prioridad que puede usarlo. Una tarea puede cerrar el semáforo que custodia una zona crítica sss la prioridad de la tarea es mayor que la prioridad del semáforo cerrado de mayor prioridad. En [34] se demuestra formalmente que el protocolo evita el bloqueo total y que cada tarea

puede ser bloqueada a lo sumo durante la duración de una sola sección crítica. El análisis del peor caso se reduce entonces a determinar la sección crítica de mayor duración entre todas las que pueden bloquear esa tarea. Esa duración expresada en mútipos del tiempo de ranura es notada B_i y corresponde al máximo bloqueo posible. Luego

$$B_i = \max\left(\bigcup_{j=i+1}^{m} B_{i,j} \right) \tag{4.14}$$

donde $B_{i,j}$ representa la duración de la sección crítica de mayor duración entre las que τ_i comparte con τ_j de menor prioridad.

4.5.3.1. Cálculo de la Factibilidad

A fin de determinar la factibilidad de un sistema PMC con bloqueos, la expresión (4.11) debe modificarse para contemplar no sólo el tiempo de ejecución de τ_i y los posibles retardos por apropiaciones de tareas de mayor prioridad, sino también el tiempo que la tarea puede permanecer bloqueada por tareas de prioridad menor. En esencia, siguiendo el razonamiento utilizado en el Teorema 4.5, un sistema **S**(m-1) podrá ser expandido a **S**(m) por incorporación de τ_i sss T_i es mayor que la (C_i+B_i)_ésima ranura que deja libre **S**(i-1) más las k posibles inversiones de prioridad que pudiera producir el diagramador. Las expresion (4.11) se transforma entonces en:

$$\forall i \qquad \sum_{h=1}^{i\text{-}1} \frac{C_i}{T_h} < 1 \quad y$$

$$T_i \geq e_{(C_i+k+B_i)(i-1)} = \text{menor } t | t = C_i + k + B_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil \tag{4.15}$$

4.6. Sistemas PMC con Limitados Niveles de Prioridad

En numerosas implementaciones de diagramadores PMC, el principal inconveniente que debe afrontarse es que el número disponible de prioridades es limitado y, en general puede ser menor al numero de usuarios del sistema reduciendo así, los posibles niveles de utilización del mismo. En el contexto de los sistemas operativos de tiempo real esta restricción es posible salvarla ya que, en principio, podría diseñarse un diagramador con tantos niveles de prioridad como sea necesario. En redes de comunicaciones, canales de transmisión de datos etc. el diagramador es implementado por medio de un hardware vinculado al control de acceso al medio donde es razonable pensar en un conjunto limitado de recursos. Por otro lado, por tratarse usualmente de mecanismos de diagramación distribuidos, la implementación de sistemas con un número muy grande de niveles de prioridad tornaría muy complejo tanto al diagramador como a procedimientos de mantenimiento de la pila de prioridades distribuida [32].

En lo que sigue, centraremos la terminología en sistemas multitarea-monoprocesador donde las tareas son usuarios de un

recurso compartido, el procesador. Esta consideración no inhibe la extensión del método a cualquier sistema multiusuario-monorocurso.

Consideramos un conjunto de tareas independientes, periódicas y apropiativas $\mathbf{S}(m)=\{\tau_1, \tau_2, \ldots, \tau_m\}$ con períodos $\mathbf{T}(m)=\{T_1,T_2,\ldots,T_m\}$, vencimientos $\mathbf{D}(m) = \{D_1, D_2, \ldots, D_m\}$ y tiempos de ejecución $\mathbf{C}(m)=\{C_1,C_2,\ldots,C_m\}$ a ser ejecutado en un sistema monoprocesador con P niveles de prioridad.

Si $P < m$, para solucionar el problema del número limitado de niveles de prioridad, debe realizarse una partición sobre el conjunto de m tareas en clases de prioridad, de manera que el número de clases resultante sea menor o igual al número de niveles de prioridad disponibles. En principio las tareas pertenecientes a una clase tienen la misma prioridad para el diagramador y, frente a un requerimiento simultáneo de un par o más de ellas podrá utilizarse cualquier criterio de desempate incluso uno aleatorio.

En [21] se propone una técnica para determinar la diagramabilidad PMC de sistemas de tareas períodicas, apropiativas e independientes con un número limitado de niveles de prioridad mediante la utilización de una grilla logarítmica construida en base a la relación $(T_m / T_1)^{1/P}$ donde P representa el número de niveles de prioridad. Luego de construida la grilla $\{0, L_1,\ldots, L_P\}$, si el período T_j de la tarea τ_j es tal que $T_j \in (L_{i-1}, L_i]$, para $1 \le i \le P$, se asignará la prioridad i a τ_j. Si bien el método es sumamente simple, es fácil encontrar ejemplos de particiones obtenidas por el mismo que resultan no factibles.

4.6.1. Efecto del Número Limitado de Niveles de Prioridad.

Al reducir el número de niveles de prioridad, varias tareas compartirán una misma clase resultando por lo tanto de prioridades indistinguibles para el diagramador. Si más de una tarea perteneciente a una misma clase se encuentra lista para ser ejecutada, el diagramador podrá realizar una asignación aleatoria produciendo un efecto similar al de las inversiones de prioridad descriptas en el apartado previo. El concepto de inversión de prioridad aplicable en el contexto de diagramadores con ilimitado número de niveles de prioridad debe interpretarse ahora como la utilización del procesador por tareas que, aún teniendo mayor período, utilizan el procesador antes que otra de menor período, por pertenecer a una misma clase. Aún así, el concepto de inversión de prioridad podría extenderse a inversiones entre clases.

Un diagramador que utilice la disciplina PMC inter-clase y aleatoria intra-clase será denominado RAN/PMC.

Para lo que sigue, se introduce la siguiente notación τ_i^j indica el *i*-ésimo elemento dentro de la clase *j*; C_i^j representa su tiempo de ejecución y T_i^j y D_i^j sus vencimientos respectivamente. τ_i indica el *i*-ésimo elemento se **S**(*m*)**.** La equivalencia entre la nueva notación y la anterior está dada por: $C_1^2 = C_{|Q_1|+1}$, $T_1^2 = T_{|Q|+1}$.

Teorema 4.10:

Dado **S**(m) PMC diagramable y una partición **Q(S**(m)) ordenada PMC tal que:

Q(S(m))=$\{Q_1, Q_2, \ldots. Q_P\}$, $\sum_{h=1}^{P}|Q_h| = m$ y $\forall \tau \in$ **S**(m) $\exists!$ $Q \mid \tau \in Q$.

Q(S(m)) será RAN/PMC diagramable si y sólo si:

$$\forall \tau \in \{\tau_1^1, \tau_1^2, ..., \tau_1^p\} \qquad T_1^j \geq e_{(C_1^j + \varepsilon_1^j)(\tilde{\mathbf{Q}}(j))} \qquad (4.16)$$

donde

$$\varepsilon_1^j = \sum_{h|\tau_h^j \neq \tau_1^j} C_h^j \quad \text{y} \quad \tilde{\mathbf{Q}}(j) = \bigcup_{v=1}^{j-1} Q_v$$

ε^j representa el número de ranuras necesarias para ejecutar todas las tareas pertenecientes a la clase j excluyendo a τ_1^j y $\tilde{\mathbf{Q}}(j)$ el subconjunto de tareas de **S**(m) pertenecientes a las clases de mayor prioridad que τ_1^j.

Demostración:

De acuerdo a la expresión (4.16), la tarea de menor período de cada clase deberá soportar no sólo la ejecución de las tareas pertenecientes a clases de mayor prioridad sino la ejecución de aquellas que, perteneciendo a la misma clase se ejecutan previamente.

Como el número de tareas de mayor período que pueden ejecutarse previamente a τ_1^j está limitado a ε_1^j, si se verifica la expresión (4.9)

$\forall \tau_1^j$ con $k = \varepsilon_1^j$, se cumple (4.16) $\forall \tau_1^j$ con lo que, la primer tarea de cada clase resulta diagramable.

Si se verifica (4.16) $\forall \tau_1^j$, luego para una clase cualesquiera:

$$T_1^j \geq \min t' \left| t' = C_1^j + \sum_{2 \leq h \leq |Q_j|} C_h^j + \sum_{\forall \tau_l \in \tilde{\mathbf{Q}}(j)} C_l . \left\lceil \frac{t'}{T_l} \right\rceil \right. \tag{4.17}$$

Para resultar diagramable, una tarea τ_n^j ($1 \leq n \leq |Q_j|$) completará su ejecución en el peor caso de carga luego que todas las tareas pertenecientes a las clases de mayor prioridad lo hayan hecho a las que deben sumársele todas las de su propia clase. En este último caso se deberá considerar que se producirá más de un arribo de las tareas de menor período que T_n^j, aún cuando pertenezcan a la misma clase. Con lo que:

$$T_n^j \geq \min t'' | t'' = C_n^j + \sum_{n+1 \leq h \leq |Q_j|} C_h^j + \sum_{1 \leq h \leq n-1} C_h^j . \left\lceil \frac{t''}{T_h^j} \right\rceil + \sum_{\forall \tau_l \in \tilde{\mathbf{Q}}(j)} C_l . \left\lceil \frac{t''}{T_l} \right\rceil \tag{4.18}$$

calculando la expresión anterior para $t'' = t'$

$$C_n^j + \sum_{n+1 \leq h \leq |Q_j|} C_h^j + \sum_{1 \leq h \leq n-1} C_h^j . \left\lceil \frac{t'}{T_h^j} \right\rceil + \sum_{\forall \tau_l \in \tilde{\mathbf{Q}}(j)} C_l . \left\lceil \frac{t'}{T_l} \right\rceil =$$

$$= \sum_{n \leq h \leq |Q_j|} C_h^j + \sum_{1 \leq h \leq n-1} C_h^j . \left\lceil \frac{t'}{T_h^j} \right\rceil + \sum_{\forall \tau_l \in \tilde{\mathbf{Q}}(j)} C_l . \left\lceil \frac{t'}{T_l} \right\rceil$$

como $t' \leq T_1^j$, $\left\lceil \frac{t'}{T_1^j} \right\rceil = 1$

luego, para $\tau_h^j \in Q_j, j > 1, T_h^j \geq T_1^j$

$$\sum_{1 \leq h \leq n-1} C_h^j . \left\lceil \frac{t'}{T_h^j} \right\rceil = \sum_{1 \leq h \leq n-1} C_h^j$$

con lo que

$$\sum_{n+1 \leq h \leq |Q_j|} C_h^j + \sum_{1 \leq h \leq n-1} C_h^j . \left\lceil \frac{t'}{T_h^j} \right\rceil = \sum_{1 \leq h \leq |Q_j|} C_h^j$$

Reemplazando en la primer expresión tenemos que *t"=t'* resulta ser solución de (4.18) y (4.17).

$$\sum_{1 \leq h \leq |Q_j|} C_h^j + \sum_{\forall \tau_l \in \tilde{\mathbf{Q}}(j)} C_l . \left\lceil \frac{t'}{T_l} \right\rceil \leq T_1^j \leq T_n^j$$

Luego basta que se cumpla (4.16) para determinar la factibilidad de todos los elementos de cada clase y luego de **Q**(**S**(*m*)).

Corolario:

Una partición **Q(S**(*m*))={Q_1, Q_2, …. Q_P} es RAN/PMC factible con $P \leq m$ ssi **S**(*m*) es diagramable PMC.

Demostración:

Si **Q(S**(*m*))={Q_1, Q_2, …. Q_P} es diagramable RAN/PMC luego lo será para $P=m$ con lo que resulta **S**(*m*) PMC diagramable. Por otro lado si **S**(*m*) es PMC diagramable, por el Teorema 4.10 será posible encontrar una partición **Q(S**(*m*))={Q_1, Q_2, …. Q_P} con $P \leq m$ RAN/PMC diagramable.

Ejemplo 4.6: $\mathbf{S}(5)=\{\tau_1, \tau_2, \ldots, \tau_5\}$, $\mathbf{T}(5)=\mathbf{D}(5)=\{6, 10, 14, 18, 18\}$, $\mathbf{C}(5)=\{2, 2, 2, 2, 2\}$, $\mathbf{Q}(\mathbf{S}(5))= \{Q_1, Q_2\}=\{\{\tau_1, \tau_2, \tau_3\}, \{\tau_4, \tau_5\}\}= \{\{\tau_1^1, \tau_2^1, \tau_3^1\}, \{\tau_1^2, \tau_2^2\}\}$.

Aplicando (4.16):

$$\forall \tau \in \{\tau_1^1, \tau_1^2\} \qquad T_1^j \geq e_{(C_1^j+\varepsilon_1^j)(\tilde{\mathbf{Q}}(j))}$$

$$\text{para } \tau_1^1 \quad 6 \geq e_{(2+4)(\varnothing)} \Rightarrow 6 \geq \text{menor } t | t = 6 + 0 = 6$$

$$\text{para } \tau_1^2 \quad 18 \geq e_{(2+2)(Q_1)} \Rightarrow 18 \geq \text{menor } t | t = 4 + 2.\left\lceil \frac{t}{6} \right\rceil + 2.\left\lceil \frac{t}{10} \right\rceil + 2.\left\lceil \frac{t}{14} \right\rceil = 18$$

dentro de un período de τ_2^1, τ_1^1 producirá $\lceil 10/6 \rceil=2$ arribos, requiriendo $2.\lceil 10/6 \rceil=4$ ranuras para su ejecución (ec.(3.1) y (4.4)). Aún cuando τ_1^1 se ejecute completamente en todos sus arribos antes que τ_2^1 y que esta última soporte la ejecución de τ_3^1, se ve que la ec. (4.16) es suficiente para analizar la diagramabilidad del sistema. Por otro lado y en general, si τ_1^j puede soportar la ejecución del resto de las tareas de su propia clase, más su propia ejecución, más la de las clases superiores, también podrán hacerlo τ_3^j y aquéllas que tienen un período mayor que τ_1^1 y soportan la misma carga. Luego aplicando (4.10)

para τ_2^1 $\quad 10 \geq e_{(2+2)(\{\tau_1^1\})} \Rightarrow 10 \geq \text{menor } t | t = 4 + 2.\left\lceil \frac{t}{6} \right\rceil = 6$

para τ_3^1 $\quad 14 \geq e_{(2+0)(\{\tau_1^1, \tau_1^2\})} \Rightarrow 14 \geq \text{menor } t | t = 2 + 2.\left\lceil \frac{t}{6} \right\rceil + 2.\left\lceil \frac{t}{10} \right\rceil = 6$

En forma idéntica se pueden obtener los resultados para Q_2.

4.6.2. Un Método Alternativo para el Cálculo de la Factibilidad de Sistemas PMC.

Utilizando los resultados previos, se presenta un método para determinar la factibilidad de sistemas PMC con una complejidad en general menor a la de los métodos tradicionales.

Del Teorema 4.10 y su corolario, si existe una partición $\mathbf{Q}(\mathbf{S}(m))=\{Q_1, Q_2, \ldots. Q_P\}$ RAN/PMC factible con $P \leq m$, $\mathbf{S}(m)$ es diagramable PMC, con lo que es suficiente encontrar una partición factible para demostrar la diagramabilidad PMC de $\mathbf{S}(m)$.

4.6.2.1. Método de cálculo.

Se define

$$\varphi_i = e_{(C_i)(\mathbf{S}(i-1))} \tag{4.19}$$

φ_i representa, para la tarea τ_i, la ranura en la cual se han satisfecho los requerimientos de las tareas que le preceden más las C_i ranuras necesarias para su ejecución.

Por la ec. (4.8) si $T_i \geq \varphi_i$ en el intervalo [1, φ_i] existirán suficientes ranuras como para atender los requerimientos de las tareas de mayor prioridad más los de τ_i, de lo contrario el sistema es no diagramable. Por el Teorema 4.10 si $T_{i-1} \geq \varphi_i$, τ_i podrá pertenecer a la clase $\{\tau_{i-1}, \tau_i\}$ y resultar aún diagramable. Nótese que esta incorporación no afecta a la diagramabilidad de τ_i ya que ha sido considerada la carga impuesta por τ_{i-1}, en el cálculo de φ_i. Continuando el razonamiento, se podrán seguir incorporando tareas a la clase mientras $T_{i-l} \geq \varphi_i$.

Si, para una dada τ_x, $x<i$, $T_x<\varphi_i$, como no puede incorporarse a la clase a la que pertenece τ_i , se calcula φ_x y se procede de igual manera intentando incorporar tareas a la nueva clase. Luego, si se inicia el procedimiento en τ_m y se obtiene una partición **Q**(**S**(m)), **S**(m) es PMC diagramable.

Ejemplo 4.7: sobre la base del ejemplo 4.6

S(5)=$\{\tau_1, \tau_2, \ldots, \tau_5\}$, **T**(5) = **D**(5) = {6, 10, 14, 18, 18}, **C**(5) = {2, 2, 2, 2, 2}

$$\varphi_5 = \text{menor } t | t = 2 + 2*\left\lceil \frac{t}{6} \right\rceil + 2*\left\lceil \frac{t}{10} \right\rceil + 2*\left\lceil \frac{t}{14} \right\rceil + 2*\left\lceil \frac{t}{18} \right\rceil = 18$$

con lo que podrá crearse la clase $\{\tau_4, \tau_5\}$

$$\varphi_3 = \text{menor } t | t = 2 + 2*\left\lceil \frac{t}{6} \right\rceil + 2*\left\lceil \frac{t}{10} \right\rceil = 6$$

luego podrá crearse la clase $\{\tau_1, \tau_2, \tau_3\}$

Como ambas clases cubren al sistema, el mismo es diagramable.

Ejemplo 4.8:

S(9)={τ_1 , τ_2 ,…., τ_9}, **T**(9) = **D**(9) = {5, 6, 7, 7, 9, 10, 20, 34, 54}, **C**(9)={1, 1, 1, 1, 1, 1, 1, 1, 1}.

i	T_i	φ_i	Test	clase
1	5		5≥5	1
2	6		6≥5	1
3	7		7≥5	1
4	7		7≥5	1
5	9	5	9≥5	1
6	12	12	12≥12	2
7	20	18	20≥18	3
8	34		34≥34	4
9	54	34	54≥34	4

Si reducimos el período de T_6 en dos unidades, T_6=10 el sistema es no diagramable ya que 10 < 12.

i	T_i	φ_i	Test	clase
9	54	54	54≥54	n
8	34	34	34≥34	n-1
7	20	18	20≥18	n-2
6	10	12	**10<12**	-

4.6.2.2. Determinación del mínimo número de niveles de prioridad

Según se ha indicado, la determinación del mínimo número de niveles de prioridad necesario para hacer diagramable a un sistema, resulta importante para reducir el número de prioridades a manejar por el diagramador cuando la complejidad del mismo o los recursos de sistema involucrados en la diagramación, sean función del número de prioridades.

Teorema 4.11:

Dado **S**(m), si es diagramable, se obtiene una partición mínima si:

$$\forall Q_i \quad x = \max x | T_1^i \geq \varphi_x \quad \text{y} \quad \tau_1^{i+1} = \tau_{|\tilde{\mathbf{Q}}(i+1)|+1} \tag{4.20}$$

Demostración:

Si $Q_1 = \{\tau_1, \ldots, \tau_j, \ldots, \tau_x\}, \quad T_1 \geq \varphi_x$

y

$Q_2 = \{\tau_j, \ldots, \tau_w\}, \quad T_j \geq \varphi_w, \quad 1 \leq j \leq x$

$w \geq x$ ya que φ es monotónica creciente y $T_j \geq T_1$

Si $Q_1' = \{\tau_1, \ldots, \tau_y\}, \quad y = \max y | T_1 \geq \varphi_y$

y

$Q_2' = \{\tau_{y+1}, \ldots, T_z\},$

resulta $z \geq w$

luego, $\left|Q_1 \cup Q_2\right| \leq \left|Q_1' \cup Q_2'\right|$

Siguiendo el razonamiento, el procedimiento conduce a una partición con el mínimo número de clases.

De manera similar se puede demostrar que se obtiene una partición mínima si:

$$\forall Q_i \quad T_1^i = \min T \in \mathbf{S}(m) \geq \varphi_{|\mathbf{Q}(i)|} \quad \text{y} \quad \tau_{|\mathbf{Q}(i-1)|}^{i-1} = \tau_{|\tilde{\mathbf{Q}}(i)|} \tag{4.21}$$

Nótese que mediante la ec. (4.20) y la ec.(4.21) es posible implementar un algoritmo para obtener una partición mínima.

Ejemplo 4.9*:* Dado el siguiente sistema:

i	T_i	C_i
1	5	1
2	10	2
3	10	1
4	10	1
5	15	1
6	18	1
7	20	1
8	20	1
9	20	1
10	20	1

Mediante la eq (4.20) se obtiene la siguiente partición:

i	T_i	C_i	φ_i	verificación	Clase
1	5	1	1	5≥1	1
2	10	2	3	5≥3	1
3	10	1	4	5≥4	1
4	10	1	5	5≥5	1
5	15	1	7	15≥7	2
6	18	1	8	15≥8	2
7	20	1	9	15≥9	2
8	20	1	10	15≥10	2
9	20	1	18	20≥18	3
10	20	1	20	20≥20	3

$\mathbf{Q}(\mathbf{S}(m))=\{(\tau_1,\ \tau_2,\ \tau_3,\ \tau_4),\ (\ \tau_5,\ \tau_6,\ \tau_7,\ \tau_8),\ (\tau_9,\ \tau_{10})\}$

Mediante la eq (4.21) se obtiene la siguiente partición:

i	T_i	C_i	φ_i	Verificación	Clase
10	20	1	20	20≥20	3
9	20	1		20≥20	3
8	20	1		20≥20	3
7	20	1		20≥20	3
6	18	1	8	18≥8	2
5	15	1		15≥8	2
4	10	1		10≥8	2
3	10	1		10≥8	2
2	10	2		10≥8	2
1	5	1	1	5≥1	1

$\mathbf{Q}(\mathbf{S}(m))=\{(\tau_1),(\ \tau_2,\ \tau_3,\ \tau_4,\ \tau_5,\ \tau_6),\ (\tau_7,\ \tau_8,\ \tau_9,\ \tau_{10})\}$

Según se desprende del ejemplo, la utilización de la eq. (4.21) permite analizar la diagramabilidad PMC del sistema y simultáneamente obtener la mínima partición que lo hace factible.

Obviamente estas soluciones no son, en general, las únicas posibles ya que la relación de compatibilidad entre los elementos de una clase ($\forall \tau_i^j \in Q_j, T_1^j \geq \varphi_i^j$) hace que sea posible definir clases no disjuntas. Mediante el procedimiento que se propone a continuación es posible encontrar todas las soluciones al problema de manera de dejar al diseñador la libertad de seleccionar de entre ellas la que considere más adecuada.:

Tomemos como base el ejemplo 4.9. Aplicando la ec. (4.20) para cada τ_i se obtiene

i	T_i	C_i	φ_i	Clase
1	5	1	1	**A**=$\{\tau_1, \tau_2, \tau_3, \tau_4\}$
2	10	2	3	**B**= $\{\tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$
3	10	1	4	**C**= $\{\tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$
4	10	1	5	**D**= $\{\tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$
5	15	1	7	**E**= $\{\tau_5, \tau_6, \tau_7, \tau_8\}$
6	18	1	8	**F**= $\{\tau_6, \tau_7, \tau_8, \tau_9\}$
7	20	1	9	**G**= $\{\tau_7, \tau_8, \tau_9, \tau_{10}\}$
8	20	1	10	**H**= $\{\tau_8, \tau_9, \tau_{10}\}$
9	20	1	18	**I**= $\{\tau_8, \tau_9, \tau_{10}\}$
10	20	1	20	**J**= $\{\tau_8, \tau_9, \tau_{10}\}$

Mediante un procedimiento similar al utilizado en las tablas de implicación se obtiene:

A = $\{\tau_1, \tau_2, \tau_3, \tau_4\}$

E $\subseteq$ **D** $\subseteq$ **C** $\subseteq$ **B** = $\{\tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$

F = $\{\tau_6, \tau_7, \tau_8, \tau_9\}$

J $\subseteq$ **I** $\subseteq$ **H** $\subseteq$ **G** = $\{\tau_7, \tau_8, \tau_9, \tau_{10}\}$

La cobertura del sistema esta dada por:

i	Clase
1	**A**
2	**A** o **B**
3	**A** o **B**
4	**A** o **B**
5	**B**
6	**B** o **F**
7	**B** o **F** o **G**
8	**B** o **F** o **G**
9	**F** o **G**
10	**G**

Luego las todas las soluciones del sistema están dadas por la siguiente proposición lógica:

(A)(A+B)(B)(B+F)(B+F+G)(F+G)(G) ≡ ABG

Luego las clases **A**, **B** y **G** son suficientes para cubrir el sistema. Como las mismas son no disjuntas es necesario seleccionar un subconjunto de cada una de ellas de manera de obtener una partición.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9	τ_{10}
A	✔	✔	✔	✔						
B		✔	✔	✔	✔	✔	✔	✔		
G							✔	✔	✔	✔

A$\cap$**B**=$\{\tau_2, \tau_3, \tau_4\}$

B$\cap$**G**=$\{\tau_7, \tau_8\}$

Luego, expresando por simplicidad la partición por la cardinalidad de cada clase (conservando el ordenamiento PMC), podemos hallar inmediatamente las siguientes particiones mínimas:

1	{(1),(7),(2)**}**
2	{(2),(6),(2)**}**
3	{(3),(5),(2)**}**
4	{(4),(4),(2)**}**
5	{(1),(6),(3)**}**
6	{(2),(5),(3)**}**
7	{(3),(4),(3)**}**
8	{(4),(3),(3)**}**
9	{(1),(5),(4)**}**
10	{(2),(4),(4)**}**
11	{(3),(3),(4)**}**
12	{(4),(2),(4)}

Las mismas resultan de asignar a distintas clases las tareas τ_2, τ_3, τ_4 y τ_7, τ_8. Debe notarse que, aún si se alterase el orden de las tareas mencionadas en el proceso de asignación a una clase (por ejemplo asignando a una clase τ_4 y a la siguiente τ_2 yτ_3) el sistema seguiría siendo diagramable ya que el orden dentro de una clase es irrelevante para el diagramador, no así el orden entre clases.

Por otro lado, particionando las clases de una partición mínima, se obtienen todas las particiones factibles del sistema con lo que, si el número de clases de prioridad disponible es mayor al mínimo obtenido es suficiente con particionar cualquier clase de una de las particiones en subclases hasta lograr el número de clases requerida. Las posibilidades que da el método son sumamente útiles para el diseñador del sistema ya que puede optar entre diferentes particiones de igual cardinalidad aplicando criterios vinculados directamente con las características funcionales de la aplicación.

Capítulo 5

Redes Locales en Tiempo -Real

5.1. Introducción

La problemática a resolver en las redes de acceso-múltiple proviene de la necesidad de compartir un medio de comunicaciones único entre una comunidad de usuarios. Como los nodos pueden simultáneamente competir por acceder al medio compartido, se necesita un mecanismo para asignar el derecho de utilización del canal de comunicaciones. El algoritmo distribuido usado por los nodos para compartir el medio se denomina protocolo de acceso-múltiple [18]. Hay al menos dos dificultades a tener en cuenta [18]: si los nodos distribuidos utilizan una política de consenso para compartir el medio, tienen que explícita o implícitamente intercambiar información. Esto implica el uso del mismo medio para

tal fin lo que torna al problema recursivo. Por otro lado, como los nodos están físicamente distribuidos, no pueden conocer simultáneamente el estado de todo el sistema.

En este capítulo se analiza una división taxonómica de los protocolos de acceso-múltiple, la relación Disciplina de Prioridades – Protocolos y la factibilidad de sistemas de tiempo real basados en las normas 802.5 (Anillo con ficha) y 802.4 (Barra con ficha) obedeciendo a un sistema de prioridades fijas en una red que trabaja en tiempo real duro.

Se demuestra que, aunque los mecanismos de prioridades de las normas son incapaces de implementar una disciplina de prioridades fijas estricta, es posible en ambos obtener un conjunto de condiciones que permite analizar la factibilidad de los mismos

Además, los resultados alcanzados se extienden para abarcar los problemas de tráfico de mensajes asincrónicos.

5.1 Disciplinas y Protocolos

En lo que sigue los protocolos más representativos de cada clase taxonómica serán analizados en relación con la disciplina de prioridades que pueden implementar.

La segunda capa del segundo nivel del modelo ISO-OSI (Data Link Control) está dividida en dos subcapas. La más baja, denominada de Control de Acceso del Medio (MAC), contiene el conjunto de reglas para acceder el medio físico. Este conjunto de reglas, llamado Protocolo de Acceso al Medio, está ligado a la

implementación de un mecanismo para dirimir el conflicto de requerimientos concurrentes para acceder al medio físico. Algunos protocolos pueden utilizar mecanismos que implementen una o más disciplinas de prioridades. Recíprocamente, una disciplina puede ser implementada por varios protocolos.

En los últimos años se han propuesto diferentes mecanismos para resolver el problema del acceso contencioso a redes de comunicaciones [17], [18], [5], [10], [13], [25]. La taxonomía dada por Kurose en [18] divide a los protocolos de acceso múltiple en dos clases: con acceso controlado y con acceso contencioso. Dicha clasificación es mostrada en la figura 5.1. Sin embargo una taxonomía completa aún no está completamente definida.

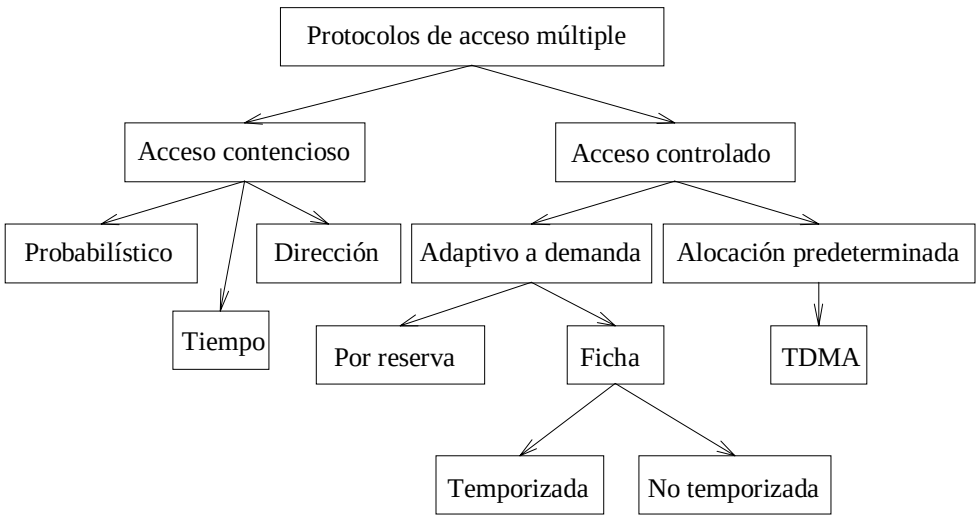


Figura 5.1
Taxonomía de los protocolos de acceso al medio

En la taxonomía dada por Kurose, se dividen los protocolos de acceso múltiple en dos clases: *acceso contencioso* y *acceso controlado*.

5.1.1 Protocolos de Acceso Contencioso

En los protocolos de acceso contencioso todo el ancho de banda del canal es utilizado por el nodo que ganó el derecho de acceso al mismo. Al no existir coordinación explícita entre los nodos para dirimir el acceso al medio es probable la ocurrencia de colisiones. Estas tienen lugar cuando dos o más nodos transmiten mensajes al mismo tiempo sobre el canal. Esta transmisión simultánea requiere que los mensajes colisionantes sean retransmitidos ya que se han visto corrompidos. Para producir la retransmisión se deberá utilizar un mecanismo que controle que, los nodos colisionantes no vuelvan a hacerlo indefinidamente (CSMA/CD). Existen diferentes políticas para controlar las colisiones sobre el canal pero como en un ambiente de TR duro, la pérdida de un vencimiento no es tolerable, no es admisible la utilización de protocolos de acceso no determinísticos tal como las diferentes variedades de CSMA [53].

5.1.2 Protocolos de Acceso Controlado

Los protocolos de acceso controlado se caracterizan por tener un acceso al medio libre de colisiones (dos o más nodos nunca intentan transmitir mensajes simultáneamente). Esto se logra preasignando el medio a intervalos regulares a los distintos nodos del sistema.

Dentro de esta clase, los hay sensibles dinámicamente a la demanda, es decir, se adaptan a variaciones en los requerimientos del sistema y los que no. Entre estos últimos, el más difundido es el TDMA (Acceso múltiple por división de tiempo). En ellos el tiempo es dividido en ranuras las cuales son luego asignadas a los nodos. En general esto produce una subutilización del sistema al asignar el medio a nodos sin mensajes pendientes para transmitir, con lo que la ranura permanecerá vacía.

En protocolos sensibles a la demanda, el medio le es otorgado a los nodos en forma secuencial con un orden predeterminado (pasaje de ficha) o por medio de un mecanismo de reserva (ronda de reserva). Algunos protocolos del paso de la ficha conocidos son el FDDI, el IEEE 802.4 y el IEEE 802.5. Fundamentalmente, estos protocolos no producen subutilización del recurso por tiempo ocioso ya que el derecho de acceso es transferido a otro nodo en caso de falta de mensajes generados.

Siguiendo la taxonomía de la Fig. 5.1 los principales protocolos determinísticos son analizados en conjunción con las disciplinas de prioridades mencionadas en el Capítulo 2, haciendo hincapié en la disciplina PMC.

5.1.3. Protocolos por Pasaje de Ficha

En un protocolo de pasaje de ficha, se conectan nodos en un anillo con topología real o virtual. Dichos nodos son visitados secuencialmente por la ficha. El derecho de acceso al medio es dado por la posesión de la ficha. Si el nodo con el derecho a acceder al

medio no tiene mensaje generado, simplemente pasa la ficha al nodo siguiente. Si por el contrario el nodo tiene un mensaje, entonces lo transmite.

En [12], [55] y [46], ambos la ficha y el anillo son reales (Real Token-Real Ring, RTRR). En [21] la topología de la red es una barra en la que se forma un anillo lógico; éste es entonces, un protocolo de anillo virtual-ficha real (Real Token-Virtual Ring, RTVR).

En [5] y [26] se usa como ficha un silencio para indicar la transferencia del derecho de transmisión de un nodo a otro; este es por lo tanto un anillo-virtual con ficha-virtual (VTVR). La característica común a todo estos protocolos es que naturalmente implementan una disciplina de rueda cíclica. Ninguna pila real se agrega a cada nodo, pero cada de ellos secuencialmente tiene el derecho de acceso al medio. Si el nodo no tiene un mensaje, la ranura no queda vacía ya que el nodo pasa la ficha inmediatamente a su sucesor en el anillo. Este procedimiento es equivalente a una pila única virtual en la que, después de cada transmisión, el nodo transmisor pasa a ocupar la base de la misma.

En los protocolos de ficha temporizada ([11], [46]) se dividen los mensajes en clases de prioridad. Los mensajes que pertenecen a la clase más alta son transmitidos utilizando una rueda cíclica con los que se les garantiza un determinado tiempo de acceso. Los mensajes de clases más bajas son transmitidos sólo cuando los mensajes de más alta prioridad dejan tiempo disponible para la transmisión de mensajes de baja prioridad.

5.1.4. Protocolos de Reserva

En protocolos de reserva, la primera parte de la ranura es utilizada por los nodos para realizar reservas anticipadas del medio. Nuevamente, la reserva puede ser real (transmisión de portadora, transmisión de un patrón de ruido o un mensaje) o virtual (silenciamiento del canal). Para hacer la reserva, los nodos deben tener un acceso libre de conflictos al canal y eventualmente se usará una disciplina de prioridades diferente en el período de reserva y en los de transmisión de mensajes.

En el protocolo de reserva propuesto en [17], el período de reserva se subdivide en (m-1) miniranuras asignados a los nodos en el orden de la pila de prioridades, la primera ranura al nodo que se encuentra en el tope de la pila, etc. Si el nodo que ocupa la posición i_ésima en la pila, $1< i< (m\text{-}1)$, está listo para transmitir y detecta ausencia de portadora en las (i-1) ranuras previas, comienza a transmitir portadora en la i_esima miniranura y transmite su mensaje (m-i) miniranuras después.

El nodo que ocupa la posición del m en la pila, transmite si tiene un mensaje generado y no ha detectado portadora en los (m-1) miniranuras de reserva.

En [35], se describe otra versión de este protocolo. Hay m miniranuras asignadas a los nodos en cualquier orden (por ej.: posición en el canal). Si un nodo tiene un mensaje generado, transmite un ruido durante su miniranura de la reserva. Al final del período de reserva todos los nodos saben cual de ellos está en condiciones de transmitir. Si sólo un mensaje es permitido después

de una reserva, el medio es tomado por el nodo más alto en la pila de entre los que realizaron sus reservas. Ambas versiones sirven los protocolos de rueda cíclica y prioridades fijas. En ambos todos los nodos deben conocer la pila de prioridades al comienzo de cada ranura.

En [10] se propone un protocolo de acceso al medio priorizado libre de colisiones (PAC). En él, después de percibir el último bit de un mensaje, todo los nodos en el sistema comienzan un período de planificación con un tiempo de maduración que es una función de la posición del nodo en la barra, la posición del nodo en la pila de prioridades y del nodo que transmitió en última oportunidad. Si se alcanza el tiempo de maduración prefijado y el medio permanece silencioso, el nodo transmitiría. Si por el contrario, observa actividad en el medio antes del fin del tiempo de maduración, significa que un nodo de más alta prioridad usó su derecho de acceso. Como la pila distribuida es modificada por todo los nodos según ciertas reglas preestablecidas a partir de una pila inicial conocida por todos ellos, este protocolo puede implementar una rueda cíclica o una disciplina de prioridades fijas.

En la disciplina de menor tiempo al vencimiento, los nodos transmiten mensajes de reserva declarando el tiempo que resta para alcanzar sus vencimientos. Estos mensajes se transmiten luego según el orden en la barra, por ejemplo de izquierda a derecha. A medida que el último bit de la reserva del último nodo se propaga hacia el primero, dicha reserva es sucesivamente observada por todos los otros nodos desde la derecha a izquierda. Luego, el nodo en el tope

de la pila sigue inmediatamente con su mensaje con lo que, el protocolo implementa la disciplina de menor tiempo al vencimiento.

De lo anterior se desprende que, puede establecerse una relación entre el conjunto de disciplinas {RC, PF MTV} y el conjunto de protocolos que sirven dicha disciplina. Sin pretender abarcar todos los protocolos existentes, los mencionados: Ficha Real-Anillo Real, Ficha Real-Anillo Virtual, Ficha Virtual-Anillo Virtual, Control de Acceso Priorizado, Reserva por Miniranuras y Reserva por Menor Tiempo al Vencimiento, son modelos utilizados por diferentes protocolos normalizados.

Conocidos los parámetros de la red, es posible determinar qué protocolos son implementables en la misma. Desgraciadamente hay un aspecto recursivo en el problema de decidir el par diciplina-protocolo: a fin de saber si una determinada disciplina es implementable, se debe conocer el tiempo de ranura el que es una función de la disciplina y el protocolo. Por esto, un par disciplina/protocolo se adopta y se analiza su comportamiento

5.2. Protocolos Normalizados

En lo que sigue se analiza el comportamiento en tiempo real de los protocolos normalizados 802.5 (Anillo con ficha) y 802.4 (Barra con Ficha). El primero de ellos puede implementar dos tipos de mecanismos, por ficha y por reserva, y el segundo por ficha temporizada y no temporizada. La elección de los mismos para ser desarrollados en esta tesis se debe a que sus diferentes variantes cubren la división taxonómica de los protocolos adaptivos a la

demanda con lo que, los resultados obtenidos, con menores variantes, son utilizables en otras redes normalizadas cuyos protocolos de acceso al medio son similares.

5.2.1 La Norma 802.5 Anillo con Ficha.

La norma Anillo con Ficha especifica una red local basada en un conjunto de estaciones conectadas en serie por medio de un medio físico. La información es transferida bit a bit de una estación a la próxima. Cada estación regenera y retransmite cada bit recibido a la siguiente estación del anillo. La dirección para la cual está destinada la información circulante, copia la misma y la retransmite en su sentido de rotación, siendo la estación que generó la información la encargada de eliminarla de anillo [12].

El mecanismo de acceso al medio de la norma 802.5 posee un mecanismo de prioridades por reserva explícita con ocho niveles de prioridad que adapta la prioridad de servicio del anillo a la más alta prioridad de los mensajes generados y listos a ser transmitidos.

Para ello tanto la ficha como el encabezamiento de los mensajes poseen un byte de *control de acceso* con tres bits de prioridad (prioridad de servicio del anillo) y tres bits de reserva.

Cuando un nodo captura la ficha puede transmitir mensajes de prioridad igual o mayor que la prioridad de servicio del anillo y, en caso de que no pueda hacerlo, si la reserva de prioridad es menor que la del mensaje a transmitir eleva el valor de la reserva a dicha prioridad. Si la reserva es mayor, retransmite la ficha tal como la

recibió. Un tratamiento similar reciben el byte de reserva del encabezamiento de mensaje.

El la Figura 5.2 a y b se muestra un ejemplo donde la numeración de los nodos indica por simplicidad su identificación y prioridad de los mensajes a transmitir simultáneamente. Corresponderá un número más alto a una prioridad más alta. El primer número en la representación de una trama indica la prioridad actual del anillo; el segundo, el valor de la reserva y el último tramo, que podrá ser lleno o vacío, indicará si el último nodo en recibir la ficha ha podido o no transmitir su mensaje.

En la Figura 5.2a la prioridad del anillo es 4 y el nodo 3 colocó su reserva. Como el nodo 5 tiene un mensaje, puede transmitirlo luego de retransmitir la ficha ya que su prioridad es más alta que la del anillo.

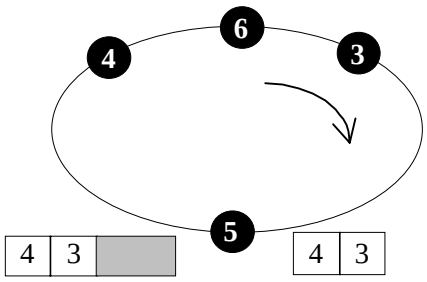


Figura 5.2a

En la Figura 5.2b la prioridad del anillo es 6, el nodo 3 colocó su reserva. A pesar de tener un mensaje, el nodo 5 tiene una prioridad menor que la del anillo por lo que se encuentra imposibilitado de transmitir pero no de levantar el valor de la reserva.

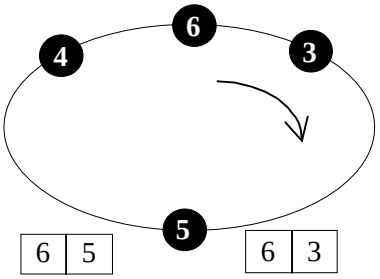


Figura 5.2b

El nodo que transmite un mensaje lo retira del anillo y si el nivel de reserva es mayor que la prioridad de servicio del anillo eleva esta última al valor de la reserva, almacena la prioridad de servicio del anillo anterior y transmite la ficha con la nueva prioridad y la reserva en cero. El mismo nodo disminuirá la prioridad de servicio del anillo cuando reciba una ficha con una reserva menor o igual a la almacenada.

En la Figura 5.3a se repite la situación de la Figura 5.2a donde el nodo 5 transmite su mensaje.

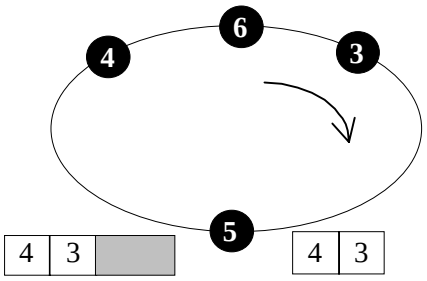


Figura 5.3a

En la Figura 5.3b el nodo 5 retira el mensaje del anillo y como la reserva tiene valor 6 eleva la prioridad del anillo a dicho valor, almacenando el antiguo valor de la prioridad de servicio del anillo.

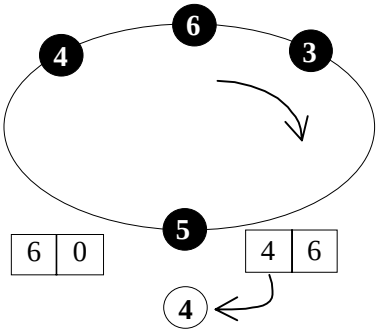


Figura 5.3b

En la Figura 5.3c el nodo 5 repone la prioridad de servicio del anillo al valor que había almacenado previamente ya que la prioridad de la reserva es menor a la que había almacenado cuando elevó dicho valor.

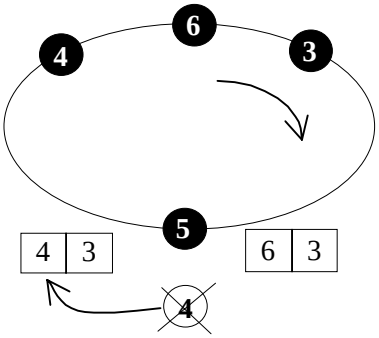


Figura 5.3c

El tiempo que la ficha demora en visitar a todos los nodos del sistema cuando ninguno de ellos tiene mensajes listos para transmitir es mucho menor que el tiempo de ranura. En sucesivas vueltas, tanto

la prioridad de servicio del anillo como la reserva caen a cero. Si en estas condiciones se produce una generación simultánea de dos o más mensajes, el medio no será necesariamente asignado al nodo de mayor prioridad ya que transmitirá el nodo que capture primero la ficha, produciendo una inversión de prioridad. El mismo caso encontramos en la ranura siguiente a la ranura_*t* no vacía en la que se transmitió el último mensaje pendiente.

5.2.1.1. Análisis de factibilidad utilizando una rueda cíclica

Consideremos que cada nodo transmite mensajes de una ranura de longitud. Dado que un anillo real implementa naturalmente una rueda cíclica y la mera posesión de la ficha garantiza la accesibilidad al medio, un STR es factible operando en una red 802.5 que no utilice el mecanismo de prioridades propuesto en la norma si se cumple:

$$T_{\min} \geq m$$

5.2.1.2. Análisis de factibilidad utilizando PMC.

El protocolo anillo con ficha 802.5 es un ejemplo típico de prioridad global donde cada mensaje tiene asignada una prioridad y el mensaje de más alta prioridad del sistema es el transmitido [52]. Como hemos visto, esta aseveración es válida en estado estacionario luego de producirse, en el peor caso, una inversión de prioridad.

En una red 802.5, luego de una ronda sin transmisiones pueden ocurrir dos cosas:

- Todos los nodos generan mensajes
- Sólo algunos nodos generan mensajes

En ambos casos puede producirse una inversión de prioridad. En el primer caso la carga del sistema es idéntica a la ranura t=1 y en el segundo la situación es similar a esta sólo que con una carga menor. Por el Lema 4.1 si **S**(m) es 1-diagramable bajo una disciplina PMC lo será si se producen inversiones de prioridad en la ranura siguiente a aquella en la que se han satisfecho todas las transmisiones pendientes. A pesar de esto, si verifica las condiciones de 1-diagramabilidad el sistema será factible (Teorema 4.8 con k=1).

Por lo tanto, un sistema será factible en una red 802.5 operando en Tiempo-Real si se cumple:

$$\forall i \in \{1,2,\ldots,(m-I)\} \qquad T_i \geq e_{(C_i+1)(i-1)} \tag{5.22}$$

Ejemplo 5.1: En la Figura 5.4 se muestra el comportamiento de un sistema de 3 nodos **S**(3)={(C_1, T_1), (C_2, T_2), (C_3, T_3)} = {(1, 3), (1, 4), (1, 4)}. Observamos que en las ranuras 4, 8 y 12 se satisfacen todos los requerimientos pendientes. Por lo tanto el sistema en las ranuras 5, 9 y 13 tendrá una demanda menor o igual a la que tenía en la ranura 1. Por otro lado observemos que el sistema tolera una inversión de prioridad en la primer ranura y en las ranuras 5 y 9, siendo la 13 funcionalmente idéntica a la inicial.

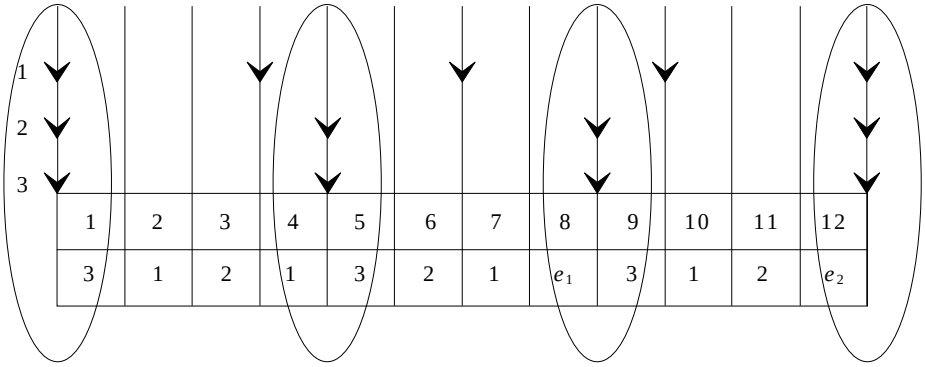


Figura 5.4
Evolución de un sistema de 3 nodos S(3)={(C_1, T_1), (C_2, T_2), (C_3, T_3)} = {(1, 3), (1, 4), (1, 4)}.

Ejemplo 5.2: Supongamos un sistema de 15 nodos operando sobre una red 802.5 con tiempos de utilización unitarios **S**(15)=(T_1, T_2,.., T_{15}}={4, 8, 10, 16, 20, 30, 30, 40, 50, 60, 90, 90, 90, 90, 100}. Los 15 períodos deben ser ubicados en hasta 8 niveles de prioridad. Si aplicamos el Teorema 4.11 obtenemos una partición mínima factible con tres clases de prioridad Q_1={1}, Q_2={2, 3, 4, 5}, Q_3={6, 7, 8, 9, 10, 11, 12, 13, 14, 15} con lo que, cualquier partición obtenida de la misma en 8 clases resultará factible (por ejemplo la partición {(1), (2), (3), (4, 5), (6,7), (8), (9,10), (11,12,13,14,15)}).

5.2.1.1 Tráfico mixto

El tráfico en la red puede caracterizarse por:

- Mensajes de tiempo real asincrónicos (esporádicos) pero con vencimiento duro (alarmas).
- Mensajes de tiempo real sincrónicos con vencimiento duro y período definido.
- Mensajes asincrónicos no de tiempo real.

En la medida que el sistema de tiempo real no sea saturado se dispondrá de ranuras libres para el tráfico de mensajes no de tiempo real. Los mensajes tipo "alarmas" son tratados como mensajes de tiempo real sincrónicos de máxima prioridad; sin embargo como son esporádicas la mayor parte del tiempo producirán ranuras vacías disponibles para el tráfico asincrónico que serán atendidos por un servidor de baja prioridad sin constricciones de tiempo. En [49] se muestra que este tipo de servidor no altera el funcionamiento del sistema de TR.

El *tiempo de latencia* de una alarma es el tiempo transcurrido desde la generación de la misma y el comienzo de su transmisión. En la Figura 5.5 se muestra la latencia de una alarma producida en el nodo_B en el momento en que la ficha abandona al mismo y hay mensajes generados en los nodos A y C.

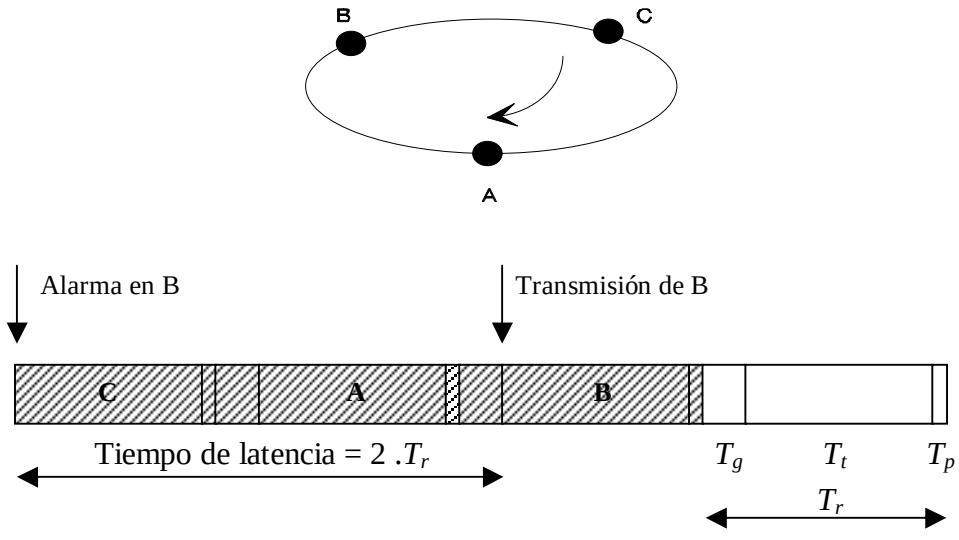


Figura 5.5
Latencia de la Alarma B. Tiempo de ranura = T_g (tiempo para aplicar la disciplina de prioridades) + T_t (tiempo de transmisión) + T_p (tiempo de propagación)

El nodo_B deberá esperar la transmisión de los nodos C y A antes de comenzar su transmisión. Luego el tiempo necesario para que el último bit del mensaje de alarma alcance al receptor será: tiempo de latencia + tiempo de transmisión + tiempo de propagación

5.2.2. La Norma 802.4 en Tiempo Real.

El Protocolo de Manufactura Automatizada (MAP) ha sido diseñado como protocolo de comunicaciones en plantas industriales y, por lo tanto en entornos de tiempo real [15]. Existe una versión completa del mismo que cubre siete niveles y una reducida de sólo tres (MiniMAP) que ocupa el primero, segundo y séptimo. Esta última es ampliamente recomendada para aplicaciones de tiempo real [36]. En ambas versiones, la norma 802.4 es el protocolo estándar.

Si una red local es utilizada en aplicaciones de tiempo real, los conflictos para acceder al medio físico deben ser resueltos mediante una o una combinación de disciplinas determinísticas que garanticen la factibilidad del sistema. Las más usadas son: Rueda Cíclica y PMC.

El estándar 802.4 ofrece un mecanismo de prioridades a fin de que la misma sea de utilidad en la implementación de sistemas de tiempo real. Pertenece a la familia de las redes de ficha temporizada junto a FDDI. Aún cuando la norma pueda implementar una rueda cíclica no puede implementar estrictamente una disciplina PMC debido a que, en ciertas circunstancias pueden producirse inversiones de prioridad [13], [36]. A fin de poder determinar la factibilidad del sistema, es necesario analizar el protocolo con el objeto de determinar el peor

caso de inversión de prioridad para cada nodo. Al final del capítulo se encontrará un compendio con la notación utilizada en el mismo a fin de facilitar su lectura.

5.2.2.1. El mecanismo de prioridades de la norma barra con ficha.

El derecho de acceso al medio físico está dado por la posesión de una ficha, la cual circula por todas las estaciones residentes en el medio. Conforme la misma pasa de un predecesor a un sucesor, se configura un anillo virtual sobre una barra real.

La norma 802.4 pertenece a la clase de protocolos que no permiten implementar estrictamente un mecanismo de prioridades PMC [38], [41]. Resulta evidente que, ante una generación simultánea de mensajes en todos los nodos del sistema, el acceso al medio será otorgado aleatoriamente al nodo que posea la ficha en dicho instante sin importar su prioridad.

La norma provee un mecanismo de prioridades de cuatro niveles denominadas *clases de acceso* y en cada estación, cuatro colas se corresponden a dichos niveles. El ancho de banda es asignado midiendo el tiempo de rotación de la ficha alrededor del anillo virtual. Los mensajes de mayor prioridad (clase_6) en cualquier estación son transmitidos siempre con la única condición de que no se exceda un cierto tiempo denominado *Tiempo de Retención de Ficha de Alta Prioridad (HPTHT, High Priority Token Holding Time).*

Las clases de menor prioridad tienen asignado un *Tiempo de Rotación de Ficha (TTRT, Target Token Rotation Time).* Para cada clase, la estación mide el tiempo que demora la ficha en circular por el anillo. Si la misma retorna a la estación en menos tiempo que el *TTRT*, podrán transmitirse mensajes de una dada prioridad hasta que el tiempo remanente en el temporizador se agote.

Como el tiempo de propagación es despreciable, puede considerare la red conformada por nodos virtuales los cuales poseen una única pila de una clase particular. Luego cada estación estará conformada por cuatro nodos virtuales que se corresponden con las cuatro clases mencionadas. Por lo tanto el número de nodos del sistema (m) será cuatro veces el de estaciones.

Cuando la ficha arriba a un nodo, el valor del *TTRT* es cargado en un contador regresivo y la cuenta comienza. Un nodo estará luego en condiciones de transmitir si, en el próximo arribo de la ficha, el valor del contador no es cero. Por ello y dado que la unidad de tiempo es de una ranura y que los mensajes transmitidos son de longitud unitaria, podemos considerar que entre dos arribos sucesivos de la ficha a un nodo, el valor de dicho contador se decrementará en tantas unidades como mensajes haya *oído* el nodo en cuestión. Luego, en lo que sigue utilizaremos la cantidad de mensajes transmitidos en una ronda para calcular el valor de los contadores en cada nodo.

5.2.2.2. Análisis de factibilidad utilizando una rueda cíclica

Consideramos que cada nodo transmite mensajes de una ranura de longitud. Dado que un anillo virtual o real implementa naturalmente una rueda cíclica y que la mera posesión de la ficha garantiza la accesibilidad al medio, un STR operando sobre una red 802.4 es factible si todos los nodos pertenecen a la clase_6 y se cumple:

$$T_{\min} \geq n_6 \tag{5.1}$$

donde $T_{\min}$ representa el menor período del sistema y n_6 el número de nodos de clase_6 que, en este caso, coincide con el número de nodos del sistema.

El problema de que coexistan mensajes de alta y baja prioridad ha sido tratado en [13], [36]. Si sus conclusiones son aplicadas a un tráfico de tiempo real restringido a mensajes de longitud unitaria más mensajes asincrónicos (no de tiempo real), una correcta inicialización de parámetros permitiría configurar un sistema factible si se cumple:

$$T_{\min} \geq n_6 + P \tag{5.2}$$

donde n_6 representa el número de nodos de tiempo real de clase_6 y P el máximo número de ranuras permitidas para el tráfico de clases de menor prioridad (clases 4, 2 y 0) en una rotación de la ficha. En [36] se demuestra que P es el máximo *TTRT* de los nodos de clase 4. En este contexto, es irrelevante si un nodo asincrónico transmite un

mensaje de longitud *TTRT*, *TTRT* nodos transmiten mensajes de longitud unitaria o cualquier combinación de ambos.

Obviamente, para satisfacer (5.2), el período mínimo del sistema debe ser mayor que el necesario para (5.1). Este es el precio que debe pagarse por admitir tráfico heterogéneo.

El orden de los nodos en el anillo lógico es fácilmente modificable por lo que es posible admitir las siguientes condiciones:

a) Las estaciones de tiempo real no poseen tráfico de baja prioridad.

b) Los nodos de alta prioridad son puestos contiguos en el anillo.

c) Se asume que n_6 es mayor que P.

Para los nodos de tiempo real, el peor caso de carga ocurre cuando, en el momento en que pasa la ficha a sus sucesor, se produce una generación simultánea en todos los nodos. Para todos los nodos del subconjunto $\{1, 2,...,n_6\text{-}P\}$, los mensajes transmitidos por los nodos de tiempo real que se encuentran aguas abajo[5] inhibirán la transmisión de los nodos de las clases de menor prioridad dado que, en el momento de recibir la ficha, estos últimos tendrán agotado los contadores de rotación. Luego el subconjunto de tiempo real no es interferido por mensajes asincrónicos.

[5] Denominaremos nodos *i*_aguas abajo al conjunto de nodos comprendidos entre el nodo_(*i*+1) y el nodo_*m*] y nodos *i*_aguas arriba a los comprendidos entre el nodo_1 y el nodo_(*i*-1).

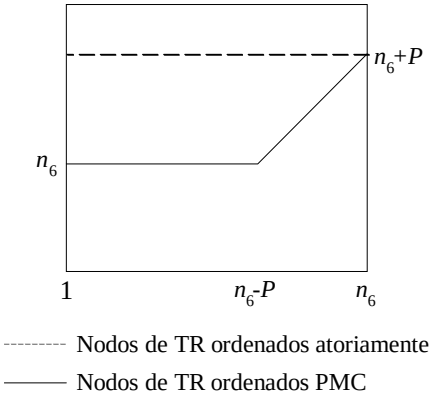


Figura 5.6
Número de inversiones de prioridad sobre los nodos de clase_6 *vs* posición del nodo en el anillo lógico en una red 802.4

Desde el nodo_(n_6-P+1) en adelante (Figura 5.6), menos de P mensajes serán transmitidos por nodos de clase_6 y, entonces, algunos mensajes de clases inferiores podrán transmitirse. Si una generación simultánea ocurre cuando la ficha deja al nodo n_6-P+v, $1 \le v \le P$, se transmitirán P-v mensajes de tiempo real y los nodos de menor prioridad tendrán oportunidad de transmitir v mensajes. Por lo tanto desde el nodo_(n_6-P+1) en adelante, los períodos tienen una cota inferior que se incrementa en pasos de a uno hasta alcanzar el nodo_n_6 cuyo período deberá ser al menos n_6+P.

Luego, deben satisfacerse.

$$\forall i \in \{1,\ 2,\ \ldots,\ n_6 - P\} \qquad T_i \ge n_6$$

y (5.3)

$$\forall i \in \{n_6 - P + v\} \qquad T_i \ge n_6 + v$$

De (5.3) resulta evidente que es más fácil lograr un sistema factible si no sólo se agrupan los nodos de tiempo real en el anillo lógico sino que se los ordena por períodos monotónicos crecientes.

5.2.2.3. Análisis de factibilidad utilizando PMC

En lo que sigue, consideramos a una red 802.4 en estado estacionario, configurada de acuerdo a las consideraciones generales descriptas en el Método de las Ranuras Vacías más las siguientes:

a) Si una estación contiene un nodo de clase_6 no puede contener nodos de otras clases.

b) Los nodos de baja prioridad serán ordenados en el anillo lógico por períodos monotónicos crecientes y tendrán los valores de sus *TTRT* crecientes.

Dado que el mecanismo de prioridades de 802.4 asocia mayores valores de los *TTRT* a las prioridades mayores, una implementación de PMC pareciera posible. El desempate entre nodos con el mismo *TTRT* es resuelto por su posición relativa en el anillo lógico: un nodo que precede a otro en el anillo, lo hará en el derecho a transmitir.

Cuando, a pesar de que un nodo_i posea mensajes para transmitir, los nodos i_aguas abajo transmiten antes que él, se producen inversiones de prioridad. Cuando las inversiones son producidas por nodos con igual valor de *TTRT* que el nodo_i, decimos que las inversiones son por *posición*. Cuando son producidas por nodos pertenecientes a clases de menor prioridad decimos que son por *ficha*. Las inversiones son notadas $p(i)$ y $\phi(i)$ respectivamente.

Los nodos de mayor prioridad no tienen asociado un valor de *TTRT* pero, pueden sufrir inversiones de nodos de la misma clase o de clases de menor prioridad. $R(i)$ denota el *TTRT* del nodo_i. $\tilde{R}(i)$ denota el $\max_{i<k\leq m}\{R(k)|\, R(k) < R(i)\}$, que es *TTRT* de máximo valor menor que $R(i)$.

Ejemplo 5.3*:* En la figura 5.7 se muestra una red de 20 nodos. Los nodos_1 a 7 pertenecen a la clase_6. Los nodos 8 a 20 pertenecen a clases inferiores y sus *TTRT* son mostrados. Supongamos que luego de una ronda sin transmisiones, se produce una generación simultánea en los nodos 8 a 13 cuando la ficha acaba de dejar el nodo_8.

La ficha arriba al nodo_9 y debido a que el nodo no ha oído la transmisión de ningún mensaje desde el último arribo de la ficha a dicho nodo, él se encuentra en condiciones de transmitir y lo hace. La ficha es pasada al nodo_10 el cual también transmite dado que ha escuchado sólo un mensaje en la última ronda (el transmitido por el nodo_9). Los nodos aguas abajo no podrán transmitir dado que su $R(i)$=2. Cuando la ficha arriba nuevamente al nodo_8 este habrá escuchado 2 mensajes y estará aún en condiciones de transmitir. En este proceso el nodo_8 ha sufrido una inversión de posición y una de ficha.

Como hemos visto, con la salvedad de no exceder el valor del *HPTHT*, los nodos de clase_6 pueden transmitir siempre que posean la ficha. Por lo tanto el análisis de factibilidad para los nodos de esta clase es diferente al de las clases de menor prioridad y por ello tratado separadamente.

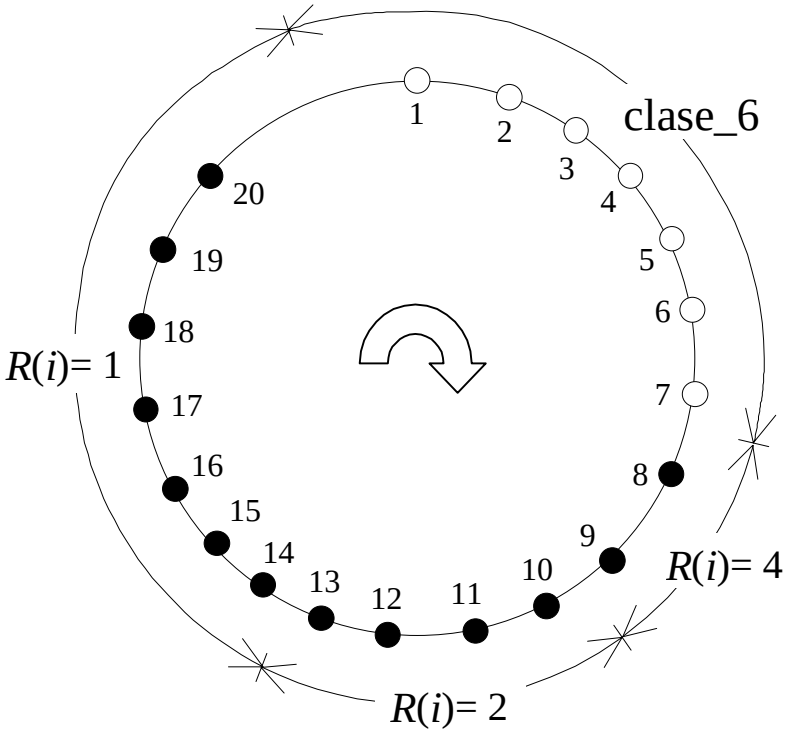


Figura 5.7. Anillo virtual de 20 nodos

5.2.2.3.1. Nodos de clase_6

Si un nodo pertenece a la clase_6, su período deberá tolerar el peor caso de inversiones de prioridades para un nodo de dicha clase más las transmisiones de los nodos que, perteneciendo a la misma clase le preceden en el anillo lógico. Las inversiones provienen de nodos de la misma clase aguas abajo y de nodos de clases de menor prioridad que reciban la ficha antes que expiren sus contadores.

$n(i)$ denota el número de inversiones producidas por nodos de menor prioridad y tiene una cota superior dada por el mayor *TTRT* de

la red. Luego, para que un nodo nodo_*i* de clase_6 resulte factible, deberá cumplirse:

$$\forall i = 1,\ 2,\ \ldots,\ n_6 \qquad T_i \geq n_6 + n(i) \qquad (5.4)$$

$n(i)$=0 para los nodos 1, 2, ..., n_6 – máx(*TTRT*), y se incrementa linealmente hasta el nodo_n_6, el cual puede sufrir máx(*TTRT*) inversiones, suponiendo que existan suficientes nodos como para hacerlo.

Ejemplo 5.4: Supongamos que la ficha abandona al nodo_5 en el momento de producirse una generación simultánea en todos los nodos. $n(5)$=2 dado que los nodos 8 y 9 transmitirán ya que en el momento de recibir la ficha habrán escuchado sólo los mensajes de los nodos 6 y 7. Como n_6 = 7, si $T_5 \geq 9$, el nodo 5 es factible. Esto coincide con el hecho de que los nodos 6, 7, 8, 9, 1, 2, 3 y 4 transmitirán previamente al nodo_5 (figura 5.7).

5.2.2.3.2. Nodos de clases inferiores

En lo que sigue $t(i)$ denota el instante en el cual, luego de una ronda sin transmisiones, la ficha abandona al nodo_*i* y una generación simultánea de mensajes se produce en el nodo_*i* y en todos los nodos *i*_aguas abajo.

Debido a que la prioridad de un nodo depende de su ubicación y de su *TTRT*, resulta irrelevante a que clase pertenece. En cualquier caso su período deberá ser tal que luego de $t(i)$, pueda tolerar las transmisiones de los nodos *i*_aguas arriba con mayor prioridad tanto como las transmisiones de los nodos *i*_aguas abajo produciendo

inversiones de prioridad. $\mathbf{C}_i$ y $|\mathbf{C}_i|$ simbolizan el conjunto de nodos aguas abajo con el mismo *TTRT* del nodo_*i* y la cardinalidad del mismo respectivamente. Obviamente, $|\mathbf{C}_i| = k.R(i) + |\mathbf{C}_i|_{\mathrm{mod}\, R(i)}$, $k \in \{0, 1, 2...\}$. $|\mathbf{C}_i|_{\mathrm{mod}\, R(i)}$ será notado $C_{i,mod.}$ por simplicidad. En lo que sigue se asume que existen suficientes nodos aguas abajo como para producir la cota superior de inversiones de ficha.

Ejemplo 5.5*:* $|\mathbf{C}_{10}| = 3$, $R(10) = 2$, 3 mod 2 = 1. El ejemplo verifica la expresión para k=1 (figura 5.7).

Lema 5.1.

En la rueda que se inicia en $t(i)$, hasta $R(i)$ nodos podrán transmitir inhibiendo a los nodos subsiguientes.

Demostración:

Inmediatamente a $t(i)$, los nodos i+1 al i+$R(i)$ transmitirán un mensaje cada uno. Los contadores de los nodos i_aguas abajo con menor o igual $R(i)$ han expirado y los nodos respectivos no pueden transmitir.

Ejemplo 5.6*:* $\mathbf{C}_{10} = \{11, 12, 13\}$, $R(10) = 2$. En la ronda que se inicia en $t(10)$, transmiten los nodos 11 y 12. Los nodos subsiguientes están inhibidos (figura 5.7).

Lema 5.2:

Luego de $t(i)$, los nodos $\in \mathbf{C}_i$ transmitirán antes que el nodo _i.

Demostración:

Cuando la ficha arriba al nodo_i por primera vez luego de $t(i)$, los nodos i+1 al i+$R(i)$ están en condiciones de transmitir sus mensajes. Por el contrario, los nodos i+$R(i)$+1 al i+2.$R(i)$, no podrán hacerlo. Sin embargo, cuando la ficha arriba por segunda vez a dichos nodos luego de $t(i)$, no han escuchado ningún mensaje desde la ronda previa y, por lo tanto, podrán transmitir. El proceso continúa hasta que todos los nodos $\in \mathbf{C}_i$ han transmitido.

Ejemplo 5.7*:* En la ronda que se inicia en $t(10)$, los nodos 11 y 12 transmitirán y en la siguiente lo hará el nodo_13 (figura 5.7).

Lema 5.3.

$$p(i) = |\mathbf{C}_i|$$

Demostración:

Se concluye inmediatamente del lema 5.2.

Ejemplo 5.8*:* En el ejemplo 5.7, $p(10) = 3 = |\mathbf{C}_{10}|$ (Figura 5.7).

Lema 5.4:

Luego de $t(i)$, el número de inversiones de posición en la última ronda en la cual transmite algún nodo $\in \mathbf{C}_i$ es $R(i)$ o $C_{i,mod}$.

Demostración:

La transmisión de $R(i)$ nodos por ronda continúa hasta que el número de nodos remanente $\in \mathbf{C}_i$ sea mayor o igual a $R(i)$. Si $C_{i,mod}$ =0, será una ronda en la cual los últimos $R(i)$ nodos $\in \mathbf{C}_i$ transmitan. Si $C_{i,mod} \neq 0$ entonces transmitirán los últimos $C_{i,mod}$ nodos.

Ejemplo 5.9*:* En el ejemplo 5.7, $C_{10,mod}$ = 1 y el nodo_13 es el único que transmitirá en la última ronda en la cual algún nodo $\in \mathbf{C}_{10}$ transmita.

En lo que sigue, la primera ronda luego de $t(i)$ en la cual, ocurran menos de $R(i)$ inversiones de ubicación empieza a tomar importancia. Topológicamente comienza entre los nodos i e i+1 y la notaremos V_0. Las rondas sucesivas serán denominadas V_1, V_2,....

Lema 5.5:

Si $C_{i,mod}$ =0, en V_0 el número de inversiones de ficha será $\tilde{R}(i)$.

Demostración:

Cuando los nodos con $TTRT = \tilde{R}(i)$ reciben la ficha en V_0, no han escuchado ningún mensaje. Luego, podrán transmitir hasta un máximo de $\tilde{R}(i)$ nodos.

Ejemplo 5.10*:* $C_{11,\mathrm{mod}}$=0. Luego de $t(11)$, los nodos 12 y 13 transmitirán. La próxima ronda es V_0 debido a que menos de $R(11)$= 2 inversiones de ubicación han ocurrido. En dicha ronda, el nodo_14 transmitirá produciendo $\tilde{R}(11)$ = 1 inversión de ficha (Figura 5.7).

Lema 5.6:

Si $C_{i,\mathrm{mod}} \neq 0$ y $\tilde{R}(i) > C_{i,\mathrm{mod}}$, el número de inversiones de ficha en V_0 será $\tilde{R}(i) - C_{i,\mathrm{mod}}$.

Demostración:

Las $|\mathbf{C}_i|$ inversiones de posición son completadas en la ronda en la cual los últimos $C_{i,\mathrm{mod}}$ nodos transmiten. Luego, el número de

mensajes escuchado por los nodos con $TTRT = \tilde{R}(i)$ es $C_{i,\mathrm{mod}}$ con lo que hasta $\tilde{R}(i) - C_{i,\mathrm{mod}}$ estarán habilitados para transmitir.

Ejemplo 5.11*:* $C_{8,mod}$ =1, $\tilde{R}(8)$ = 2. En V_0, los nodos 9 y 10 transmiten produciendo una inversión de ficha.

Lema 5.7:

Si $C_{i,\mathrm{mod}} \neq 0$ y $\tilde{R}(i) \leq C_{i,\mathrm{mod}}$, en V_0 no se producirán inversiones de ficha.

Demostración:

En V_0, los nodos con $TTRT = \tilde{R}(i)$ han escuchado $C_{i,\mathrm{mod}}$ mensajes con lo que estarán inhabilitados para transmitir.

Ejemplo 5.12*:* $C_{10,\mathrm{mod}}$ = 1, $\tilde{R}(10)$ = 1. En V_0 sólo el nodo_13 transmite y no se producen inversiones de ficha.

Lema 5.8:

En V_0, el número de inversiones de ubicación más el número de inversiones de ficha (simbolizado L_0), es:

$$L_0 = \max(\tilde{R}(i), C_{i,\mathrm{mod}})$$

Demostración:

Se concluye de los lemas 5.5 al 5.7.

Ejemplo 5.13: $C_{8,\mathrm{mod}}$= 1, $\tilde{R}(8)$ = 2 y L_0 = 2. En V_0, los nodos 9 y 10 transmiten con o que se produce un total de dos inversiones: una de posición (nodo_9) y otra de ficha (nodo_10).

Lema 5.9:

El número de inversiones de ficha en V_0 es L_0- $C_{i,\mathrm{mod}}$

Demostración:

Se concluye de los lemas 5.5 al 5.8.

Ejemplo 5.14*:* En el ejemplo 5.13, $\phi(8) = 1 = L_0 - C_{8,\mathrm{mod}}$

Por el Lema 5.9, el número total de inversiones que sufre el nodo nodo_*i* en V_0 será menor que $R(i)$, con lo que estará habilitado para transmitir. Sin embargo, los nodos *i*_aguas arriba, podrían haber generado mensajes tal que, sumados a las inversiones, inhiban al nodo_*i* para transmitir. En las ruedas siguientes, a pesar de haber escuchado mensajes de alta prioridad, algunos nodos *i*_aguas abajo podrán transmitir produciendo nuevas inversiones. Este proceso continúa hasta que no son posibles más inversiones de prioridad. La ronda en la cual esto ocurre es encontrada en los siguientes teoremas.

Teorema 5.1:

Si existieren, las inversiones de ficha tendrán lugar hasta la ronda V_r inclusive, donde $r=\max x \in \mathbf{N}\cup\{0\} \mid x.(\tilde{R}(i)-R(i))+L_0>0$.

Demostración:

Por el lema 5.8, el número de inversiones en V_0 es L_0. En V_0 el nodo_*i* estará inhibido de transmitir si los nodos *i*_aguas arriba han transmitido $H_0=R(i)-L_0$ mensajes. Luego, los nodos con $TTRT=\tilde{R}(i)$ que no transmitieron en V_0 han escuchado H_0 mensajes, con lo que el número de inversiones de ficha en V_1 es $L_1=\tilde{R}(i)-H_0=\tilde{R}(i)-R(i)+L_0$. A fin de mantener inhibido al nodo_*i*, el número de mensages de alta

prioridad que deberá transmitirse en V_1 es $H_1 = R(i) - L_1 = 2.R(i) - \tilde{R}(i) - L_0$. Luego de ello, los nodos con $TTRT = \tilde{R}(i)$ que no transmitieron en V_1 han escuchado H_1 mensajes con lo que, el número de dichos nodos que podrán transmitir en V_2 será $L_2 = \tilde{R}(i) - H_1 = 2.\tilde{R}(i) - 2.R(i) + L_0 = 2.(\tilde{R}(i) - R(i)) + L_0$. En la ronda V_r, el número será $r.(\tilde{R}(i) - R(i)) + L_0$. Las últimas inversiones de prioridad tienen lugar en la ronda $r = \max x \in \mathbf{N} \cup \{0\} | x(\tilde{R}(i) - R(i)) + L_0 > 0$.

Ejemplo 5.15*:* En V_0 luego de $t(11)$, el nodo_14 transmite, produciendo una inversión de ficha. De ahí en adelante, los nodos de clase class_6 podrán inhibir al nodo_11 pero no podrán producirse otras inversiones de ficha. $R(11)=2$, $\tilde{R}(11)=1$, $L_0 = 1$, y r = max x | x (1-2) + 1 > 0 es 0.

Teorema 5.2:

La cota superior en el número de inversiones de ficha sobre el nodo _*i* está dada por:

$$\phi(i) = \frac{\left(r^2 + r\right).\left(\tilde{R}(i) - R(i)\right)}{2} + \left(r + 1\right).\max\left(\tilde{R}(i),\ C_{i,\mathrm{mod}}\right) - C_{i,\mathrm{mod}}$$

Demostración:

En la ronda V_0, el número de inversiones de ficha es $L_0 - C_{i,mod}$, en V_1, es $\tilde{R}(i) - R(i) + L_0$ y en V_2 es $2.\tilde{R}(i) - 2.R(i) + L_0$. El proceso continúa hasta la ronda r en la cual tiene lugar la última inversión de ficha. En esta ronda serán $r.\tilde{R}(i) - r.R(i) + L_0$. Por lo tanto, la cota superior para las inversiones de ficha para el nodo_*i* estará dada por:

$$\phi(i) = \frac{\left(r^2 + r\right).\left(\tilde{R}(i) - R(i)\right)}{2} + (r+1).L_0 - C_{i,\mathrm{mod}} =$$

$$= \frac{\left(r^2 + r\right).\left(\tilde{R}(i) - R(i)\right)}{2} + (r+1).\max\left(\tilde{R}(i), C_{i,\mathrm{mod}}\right) - C_{i,\mathrm{mod}}$$

Ejemplo 5.16*:* En el ejemplo previo, sólo hubo una inversión de ficha. r=0, $R(11)$= 2, $\tilde{R}(11)$=1 y $C_{11,mod}$=0. $\phi(11)=\frac{(0+0)(1-2)}{2}+$ $+(0+1).\max(1,0)=1.$

Corolario:

La cota superior para el total de inversiones de prioridad sobre el nodo_i estará dada por:

$$I(i) = p(i) + \phi(i) =$$

$$= |\mathbf{C}_i| + \frac{(r^2 + r)(\tilde{R}(i) - R(i))}{2} + (r+1).\max\left(\tilde{R}(i),\ C_{i,\mathrm{mod}}\right) - C_{i,\mathrm{mod}} \qquad (5.5)$$

Demostración:

Se concluye directamente del lema 5.3 y del teorema 5.2.

Ejemplo 5.17*:* En el ejemplo previo tuvieron lugar 3 inversiones.

Como $|\mathbf{C}_i|$=2, r=0, $R(11)$=2, $\tilde{R}(11)$=1 y $C_{11,\mathrm{mod}}$=0, $\phi(11)$ = 2 + $+\frac{(0+0)(1-2)}{2}+(0+1)\cdot\max(1,0)=3.$

Para cada nodo de las clases bajas la expresión (5.5) provee la información necesaria para analizar la factibilidad de la red. Cada

uno de ellos deberá tolerar el peor caso de inversión de prioridades más la carga de los nodos de mayor prioridad. Sus períodos deberán ser mayores o iguales que la ranura libre número ($I(i)$+1) que deja el subsistema (i-1) [28]. Esta condición está dada por:

$$\forall i \in \{n_6 + 1, \ldots, m\} \qquad T_i \geq e_{(I(i)+1)(i-1)} \qquad (5.6)$$

La red resultará factible si las expresiones (5.4) y (5.6) se satisfacen. Ellas cubren la factibilidad de la clase_6 y las inferiories respectivamente.

5.3.3.4. Partición en clases de prioridad y asignación de los *TTRT.*

La factibilidad de un sistema que operará sobre una red 802.4 dependerá obviamente de como es particionado en clases de prioridad y como son asignados los *TTRT* de cada nodo. La posición de cada nodo en el anillo lógico define la prioridad dentro de la clase. Las condiciones expresadas anteriormente no sólo permiten analizar la factibilidad de un sistema sino que proveen un método sistemático para realizar la partición de un conjunto de nodos y asignar valor a sus *TTRT*, si se aplican en conjunción con el siguiente teorema y su corolario.

Teorema 5.3:

Si $|\mathbf{C}_i|=0$, el mínimo $\phi(i)$ es obtenido asignando $R(i) \geq 2 . \tilde{R}(i)$.

Demostración:

El mínimo $\phi(i)$ se obtendrá si las inversiones de ficha se concentran en V_0. Por el teorema 5.1, si no se producen inversiones de ficha en V_1, $\tilde{R}(i)$-$R(i)$+ max($\tilde{R}(i)$, $C_{i,\,mod}$) = $2\,\tilde{R}(i)$-$R(i) \leq 0$ y entonces $R(i) \geq 2.\,\tilde{R}(i)$ debe satisfacerse.

Ejemplo 5.18*:* $|\mathbf{C}_9| = 0$, $R(9)=4$ y $\tilde{R}(9) = 2$. En V_0, los nodos 10 y 11 transmiten, produciendo dos inversiones de ficha. Si $\tilde{R}(9)=3$, $R(9) \geq 2.\,\tilde{R}(9)$ no se satisface y ocurrirán tres inversiones de ficha.

Corolario:

Asignar $R(i)>2.\,\tilde{R}(i)$ produce el mismo número de inversiones de ficha que asignar $R(i)=2.\,\tilde{R}(i)$.

Demostración:

Si $R(i)=2.\,\tilde{R}(i)$, luego de $\tilde{R}(i)$ inversiones en V_0, son necesarias otras $\tilde{R}(i)$ transmisiones por parte de los nodos de clase_6 para inhibir al nodo_*i*. Si $R(i)>2.\,\tilde{R}(i)$, a pesar de que son necesarias más transmisiones de nodos de clase_6 para inhibir al nodo_*i*, el número de inversiones de ficha serían las mismas.

Ejemplo 5.19*:* En el ejemplo previo, si $R(9)=5$, el número de inversiones de ficha en V_0 seguirá siendo 2.

Teniendo en cuenta lo anterior, la idea básica para obtener una partición factible se basa en particionar el sistema en un subsistema de clase_6 y otro de las clases bajas. Ellos se denominarán primer y segundo subsistema respectivamente.

En un comienzo suponemos que los m nodos pertenecen al primer subsistema (n_6=m). El sistema resultará factible si $\forall i = 1, 2, ..., m, \ T_i \geq m$ se satisface. Dado que T_1 es el mínimo período de la red, si se verifica $T_1 \geq m$, el sistema es factible. Si no cumple con la expresión el procedimiento separará nodos de la clase_6, uno a uno, y lo incorporará al segundo subsistema comenzando por el nodo_m, asignándole $R(m)$=1.

En cada oportunidad en que un nodo es separado del primer subsistema e incorporado en el segundo, el valor del *TTRT* anterior le es asignado y la expresión (5.6) es usada para verificar la factibilidad del segundo subsistema. Si ésta se verifica se prueba luego la factibilidad del primer subsistema. Si (5.4) no se verifica, el *TTRT* del último nodo separado es incrementado en uno. Si el nuevo *TTRT* es mayor que dos veces el último *TTRT* asignado, el proceso finaliza y el sistema no podrá ser particionado por este método; por el contrario, si el nuevo *TTRT* no es mayor que dos veces el último asignado, se aplica la expresión (5.6) al segundo subsistema con el *TTRT* modificado. Cuando se encuentra un segundo subsistema factible la expresión (5.4) es aplicada al primer subsistema. Si no resultare factible, un nuevo nodo es separado del mismo y el proceso se repite. Si resultase factible se ha encontrado una partición factible y el proceso finaliza. En la figura 5.8 se muestra el algoritmo expresado en un pseudocódigo estandar.

Ejemplo 5.20: En la Tabla 5.1 se especifica una red 802.4 de 20 nodos con sus respectivos períodos. A fin de analizar la factibilidad y dado que T_1< 20, el nodo_20 es separado creando un segundo subsistema. $I(20)$=0 (Teorema 5.2) y $e_{1(19)}$=80. Como T_{20}=210, se

verifica (5.6). Sin embargo, la expresión (5.4) aplicada al subsistema {1, 2, ..., 19} no se verifica. Entonces, el nodo_19 es separado e incorporado al segundo subsistema con un *TTRT*=1. Luego $I(19)$= 1 y $T_{19} \geq e_{2(18)}$=80 (5.6). El proceso continúa hasta asignar *TTRT*=1 al nodo_13, con lo que el segundo subsistema no resulta factible. El *TTRT* es incrementado a 2 lo que hace al subsistema factible. El procedimiento continúa hasta encontrar que un *TTRT*=4 es necesario para incorporar al nodo_9. Luego de incorporar al nodo _8 con *TTRT*=4, ambos subsistemas son factibles y se ha encontrado una solución al problema. El resultado es mostrado en la tabla 5.1.

```
if T(1) ≥ m then
    n6 = m
    return ("El sistema es factible")
end if
n6 = m - 1
i = m
R(m) = 1
while ( Exp.(5.4) ( 1...n6 ) = FALSE  or Exp.(5.6) ( n6+1....m ) = FALSE ) do
    while ( Exp. (5.6) ( n6+1....m ) = FALSE ) do
        R(i) = R(i) + 1
        if R(i) > 2. R~(i)  then return ("No se ha encontrado solución")
    end while
    if ( Ineq. (5.4) ( 1...n6 ) = FALSE )  then
        i = i - 1
        n6 = n6 -1
        R(i) =  R~(i)
    endif
end while
return ("El sistema es factible", n6, R( ))
```

Figura 5.8. Un algoritmo para obtener una partición factible.

i	T_i	*TTRT*	*minT*(i)
1	7	clase_6	7
2	8	clase_6	7
3	10	clase_6	7
4	10	clase_6	8
5	10	clase_6	9
6	15	clase_6	9
7	19	clase_6	9
8	20	4	18
9	20	4	19
10	40	2	30
11	50	2	37
12	50	2	30
13	70	2	37
14	80	1	80
15	80	1	80
16	210	1	80
17	210	1	80
18	210	1	80
19	210	1	80
20	210	1	80

Tabla 5.1. Una red 802.4 de 20 nodos y sus respectivos períodos. La tercer columna indica los TTRT asignados a cada nodo y la última el período mínimo para cada nodo para asegurar la factibilidad del sistema.

Apéndice A

Símbolos y Definiciones

$\lceil x \rceil$	operador monádico techo. $\lceil x \rceil = x$ si x=INT(x); si no $\lceil x \rceil$= INT(x)+1
B_i	máximo bloqueo sobre la tarea_i
$\mathbf{C}_i$	conjunto de nodos i_aguas abajo.
C_i	Tiempo de utilización del recurso por el usuario_i. En el caso de un sistema de tareas, representa el tiempo de ejecución.
D_i	Vencimiento de la tarea/nodo_i.
$e_{j(m)}$	j_ésima ranura que deja libre libre **S**(m).
HPTHT	Temporizador de rotación de ficha de clases de alta prioridad (High Priority Token Holding Time).
$I(i)$	número total de inversiones de prioridad que afectan al nodo_i.
L_0	número total de inversiones de prioridad en V_0.
L_1, L_2 ,...,	número total de inversiones de prioridad en V_1, V_2, …
n_6	número de nodos en clase_6
nodos i_aguas abajo	nodos i+1, i+2, …, m.
nodos i_aguas arriba	nodos 1, 2, …, i-1.

$p(i)$	número de inversiones de posición. Inversiones producidas por nodos *i*_aguas abajo con $TTRT=R(i)$.
$\mathcal{P}_m(t)$	número de arribos del sistema $\mathbf{S}(m)$ en el intervalo [1, *t*]
$R(i)$	*TTRT* del nodo_*i*.
r	ronda en la cual se produce la última inversion de ficha $r=\max x \in \mathrm{N}\cup\{0\} \mid x(\tilde{R}(i)-R(i))+L_0>0$
$\tilde{R}(i) = \max_{i<k\leq m}\{R(k), R(k)<R(i)\}$	máximo $TTRT < R(i)$.
$\phi(i)$	número de inversiones de ficha. Inversiones de prioridad producidas por los nodos *i*_aguas abajo con $TTRT<R(i)$.
τ_i	tarea_$i \in \mathbf{S}(m)$.
$\mathbf{S}(m)$	Conjunto de *m* usuarios de tiempo real. En general cada uno de ellos es completamente especificado como una terna (*C*,*T*,*D*)
$t_0(i)$	instante en el cual la ficha abandona el nodo_*i* y una generación de mensajes se produce simultáneamente en él y en todos los nodos *i*_aguas abajo, luego de una ronda sin transmisiones.
T_i	Período de la tarea/nodo_*i*.
T_{min}	Mínimo periodo de $\mathbf{S}(m)$.
T_r	Tiempo de ranura
$TTRT$	Temporizador de rotación de ficha (Target Token Rotation Time).
$U_{\mathbf{S}(m)}$	factor de utilización del sistema.
V_0	ronda luego de $t_0(i)$ con menos de $R(i)$ inversiones de posición.
V_r	última ronda con inversiones de ficha.
$W_m(t)$	número de ranuras requeridas por el sistema $\mathbf{S}(m)$ en el intervalo [1, *t*]

Apéndice B

Lemas y Teoremas

Lemas y Teoremas del Capítulo 3

Lema 3.1:

$$\mathcal{R}(x+y) = \mathcal{R}(x) + \mathcal{R}(y) - j \qquad x, y \in \mathbf{N} \quad j \in \{0,1,....,m\}$$

Teorema 3.1:

La *j*-ésima ranura libre en un sistema de *m* usuarios es

$e_{j(m)}$ = menor $x \mid x = j + \mathcal{A}_m(x)$

Teorema 3.2:

Dado un sistema factible no saturado operando bajo una disciplina de prioridades

(a) $\forall\ i\ e_{i+1}\text{-}e_i \leq e_1$

(b) La ranura M es una ranura vacía.

Lemas y Teoremas del Capítulo 4

Lema 4.1:

Si un sistema es *k*-diagramable, lo será aún si el diagramador produce *k* inversiones de prioridad luego de cada singularidad.

Teorema 4.1:

Sea $\mathbf{S}(m\text{-}1)=\{\tau_1, \tau_2,\ldots, \tau_{m-1}\}$ factible y no saturado operando bajo una DPF y sea $e_1(m\text{-}1)$ la primer ranura libre de $\mathbf{S}(m\text{-}1)$. El sistema $\mathbf{S}(m)$ resultante de agregar un usuario en la base de la pila será diagramable sss $T_m \geq e_1(m\text{-}1)$.

Teorema 4.2:

Si $\mathbf{S}(m)$ es tal que $\min(\mathbf{T}(m)) \geq m$, $\mathbf{S}(m)$ es factible independientemente de como se ordene la pila de prioridades de la DPF.

Teorema 4.3:

Si $\mathbf{S}(m)$ es tal que $\min(\mathbf{T}(m)) < m$, luego $\mathbf{S}(m)$ es factible sss para todo $i \in [(\mathbf{T}(m))+1,\ldots,m]$ se verifica

$$\sum_{h=1}^{i-1} \frac{1}{T_h} \leq 1 \quad \text{y} \quad T_i \geq \min\ t \in \left[\min(\mathbf{T}(m)), T_i\right] | t = 1 + \sum_{h=1}^{i-1} \left\lceil \frac{t}{T_h} \right\rceil = e_1(i-1)$$

Teorema 4.4:

Si el número de ranuras solicitadas por período por un usuario τ_i es $C_i \geq 1$, la *j*-ésima ranura libre en un sistema de *m* usuarios es $e_j(m)$=menor $t \mid t = j + W_m(t)$,donde $W_m(t) = \sum_{h=1}^{m} C_i \left\lceil \frac{t}{T_i} \right\rceil$ indica el número de ranuras requeridas en el intervalo [1, *t*].

Teorema 4.5:

Sea $\mathbf{S}(m\text{-}1)=\{\tau_1, \tau_2,\ldots, \tau_{m-1}\}$, $\mathbf{C}(m)=\{C_1, C_2,\ldots, C_m\}$, $C_i \geq 1$, factible y no saturado operando bajo una DPF y sea $e_j(m\text{-}1)$ la *j*-esima ranura libre de $\mathbf{S}(m\text{-}1)$. El sistema $\mathbf{S}(m)$ resultante de agregar un usuario en la base de la pila será diagramable sss $T_m \geq e_j(m\text{-}1)$ con $j=C_m$.

Teorema 4.6:

Si $\mathbf{S}(m)$ es tal que $\min(\mathbf{T}(m)) \geq \sum_{h=1}^{m} C_h$, $\mathbf{S}(m)$ es factible

independientemente de como se ordene la pila de prioridades de la DPF.

Teorema 4.7:

Si **S**(m) es tal que $\min(\mathbf{T}(m)) < \sum_{h=1}^{m} C_h$, luego **S**($m$) es factible sss para todo i se verifica:

$$\sum_{h=1}^{i-1} \frac{C_h}{T_h} < 1 \quad \text{y} \quad T_i \geq \min\ t \,|\, t = C_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil = e_{C_i}(i-1)$$

Teorema 4.8:

Un sistema **S**(m) es k -diagramable sss $\forall i \in \{1,2,...,(m-I)\}$,

$T_i \geq e_{(C_i+k)(i-1)}$ donde, $I = \max_{1 \leq s \leq m} s \mid \sum_{h=m}^{m-s+1} \frac{C_h}{k} \leq 1$

Teorema 4.9:

Dado un sistema **S**(m), diagramable no saturado de m usuarios sincrónicos de tiempo real será posible adicionar I usuarios asincrónicos utilizando un diagramador con inversión de prioridad si **S'**($m+I$)=**S**(m)$\cup$**A**(I) es k-diagramable, donde **A**(I) son los I usuarios asincrónicos adicionados a **S**(m).

Teorema 4.10:

Dada una partición sobre **Q(S**(m)) ordenada PMC tal que: **Q(S**(m))={Q_1, Q_2, …. Q_P}, $\sum_{h=1}^{P} |Q_h| = m$ y $\forall \tau \in$ **S**(m) $\exists!$ $Q \mid \tau \in Q$. **Q(S**(m)) será RAN/PMC diagramable si y sólo si

$$\forall \tau \in \{\tau_1^1, \tau_1^2, ..., \tau_1^p\} \qquad T_1^j \geq e_{(C_1^j + \varepsilon_1^j)(\tilde{\mathbf{Q}}(j))}$$

donde

$$\varepsilon_1^j = \sum_{h \mid \tau_h^j \neq \tau_1^j} C_h^j \quad \text{y} \quad \tilde{\mathbf{Q}}(j) = \bigcup_{v=1}^{j-1} Q_v$$

ε^j representa el número de ranuras necesarias para ejecutar todas las tareas pertenecientes a la clase j excluyendo a τ_1^j y $\tilde{\mathbf{Q}}(j)$ el subconjunto de tareas de **S**(m) pertenecientes a las clases de mayor prioridad que τ_1^j.

Teorema 4.11:

Dado **S**(*m*), si es diagramable, se obtiene una partición mínima si $\forall Q_i \quad x = \max x | T_1^i \geq \varphi_x$ y $\tau_1^{i+1} = \tau_{|\tilde{\mathbf{Q}}(i+1)|+1}$

Lemas y Teoremas del Capítulo 5

Lema 5.1:

En la rueda que se inicia en *t*(*i*), hasta *R*(*i*) nodos podrán transmitir siguiendo inhibiendo a los nodos subsiguientes.

Lema 5.2:

Luego de *t*(*i*), los nodos ∈ $\mathbf{C}_i$ transmitirán antes que el nodo _*i*.

Lema 5.3:

$$p(i) = |\mathbf{C}_i|$$

Lema 5.4:

Luego de *t*(*i*), el número de inversiones de posición en la última ronda en la cual transmite algún nodo ∈ $\mathbf{C}_i$ es *R*(*i*) o $C_{i,mod}$.

Lema 5.5:

Si $C_{i,mod}$ =0, en V_0 el número de inversiones de ficha será $\tilde{R}(i)$.

Lema 5.6:

Si $C_{i,\mathrm{mod}} \neq 0$ y $\tilde{R}(i) > C_{i,\mathrm{mod}}$, el número de inversiones de ficha en V_0 será $\tilde{R}(i) - C_{i,\mathrm{mod}}$.

Lema 5.7:

Si $C_{i,\mathrm{mod}} \neq 0$ y $\tilde{R}(i) \leq C_{i,\mathrm{mod}}$, en V_0 no se producirán inversiones de ficha.

Lema 5.8:

En V_0, el número de inversiones de ubicación más el número de inversiones de ficha (simbolizado L_0), es:

$L_0 = \max(\tilde{R}(i), C_{i,\mathrm{mod}})$

Lema 5.9:

El número de inversiones de ficha en V_0 es L_0- $C_{i,\mathrm{mod}}$

Teorema 5.1:

Si existieren, las inversiones de ficha tendrán lugar hasta la ronda V_r inclusive, donde r=max $x \in \mathbf{N} \cup \{0\} \mid x.(\tilde{R}(i) - R(i)) + L_0 > 0$.

Teorema 5.2:

La cota superior en el número de inversiones de ficha sobre el nodo _i está dada por:

$$\phi(i) = \frac{\left(r^2 + r\right).\left(\tilde{R}(i) - R(i)\right)}{2} + \left(r+1\right).\max\left(\tilde{R}(i),\ C_{i,\mathrm{mod}}\right) - C_{i,\mathrm{mod}}$$

Corolario:

La cota superior para el total de inversiones de prioridad sobre el nodo_i estará dada por:

$$I(i) = p(i) + \phi(i) =$$

$$= \left|\mathbf{C}_i\right| + \frac{(r^2 + r)(\tilde{R}(i) - R(i))}{2} + (r+1).\max\left(\tilde{R}(i),\ C_{i,\mathrm{mod}}\right) - C_{i,\mathrm{mod}}$$

Teorema 5.3:

Si $|\mathbf{C}_i|$=0, el mínimo $\phi(i)$ es obtenido asignando $R(i) \geq 2.\tilde{R}(i)$.

Corolario:

Asignar $R(i) > 2.\tilde{R}(i)$ produce el mismo número de inversiones de ficha que asignar $R(i) = 2.\tilde{R}(i)$.

Referencias

[1] Alonso A, Dueñas J.C., León G., and Puente J.A.,"An Environment for ditributed prototyping of Real-Time Systems", *Control Engineering Practice,* Vol 3, No 6, pp871-876, 1995.

[2] Audsley, N.C., Burns, A., Davies, R.I. and Wellings A.J., "Integrating Best Effort and Fixed Priority Scheduling", *Dept. of CC, University of York,* ftp.york.ac.uk/pub.

[3] Audsley, N.C., Bate I.J. and Burns, "Putting fixed priority scheduling into engineering practice for Safety Critical Applications", *Proc. of the Real Time Technology and Applications Symposium, pp 2-10,* IEEE Technical Committee on Real Time Systems, 1996.

[4] Cayssials, R., Orozco, J., Santos, J. y Ferro, E., "Priority Inversions in Rate Monotonic 8802/4: Analysis of the Worst Case". Actas de las *XXII Conferencia Latinoamericana de Informática*. Colombia. 1996.

[5] Chlamtac I, W R Franta and K D Levin “BRAM: The Broadcast Recognizing Access Method”, *IEEE Transactions on Communications* Vol 27 No 8 (August. 1989) pp 1183-1190.

[6] Cooling, J.E., “Software Design for Real-time Systems”, Chapman and Hall, 1991.

[7] Douglas Locke C., "Software Architecture for Hard Real-Time Applications: Ciclic Executives vs. Fixed Priorities Executives", *Real-Time Systems*, Vol 4 No 1 (Marzo 1992) pp 37-53.

[8] Dwamm A, Reisinger W, Schawabl W, Kopetz H, “The Real Time Oerating System of MARS”, ACM Operating Systems Review, 23(3), pp. 141-151, 1989.

[9] Ferro, E., Orozco, J. Santos, J., Cayssials, R., “Sincronización de tareas en Tiempo Real Duro utilizando el método de las ranuras vacías”, *Revista de Información Tecnológica*, Vol. 7, N^{o} 4, pag. 101-108. 1996.

[10] Gold Y I and W R Franta, “An Efficient Collision-Free Protocol for Prioritized Access Control of Cable or Radio Channels” Computer Networks Vol 7 (1983) pp 83-98.

[11] IEEE Std. 802.4. “Token-Passing Bus Access Method and Physical Layer Specifications”. *Institute of Electrical and Electronics Engineers, Inc.* 1985.

[12] IEEE Std. 802.5. “Token-Ring Access Method and Physical Layer Specifications”. *Institute of Electrical and Electronics Engineers, Inc.* 1985.

[13] Janetzky, D. and Watson, K.S.. “Perfomance evaluation of the MAP Token-Bus in real-time applications”. *Advances in Local Area Networks*, IEEE Press, New York , pp 411-427. 1987.

[14] Joseph, M. and Pandya, P. “Finding response times in a real time system”. *The Computer Journal*, 29,5, pp. 390-395. 1986.

[15] Kaminsky, M. Protocols for communicating in the factory. *IEEE Spectrum* Vol 23 Nro. 4 (April 1986) pp. 56-62.

[16] Katcher I.D., Arakawa H. “Engineering and Analisis of Fixed Priority Schedulers”. *IEEE Trans. on Software Engineering*, Vol.: 19 N^{o} 9, 1993.

[17] Kleinrock L and M Scholl “Packet Switching in Radio Channels: Conflict-Free Multiple Access Schemes” *IEEE Transactions on Communications* Vol 28 No 7 (July 1980) pp 1015-1029.

[18] Kurose J F, M Schwartz and Y Yemini “Multiple-Access Protocols and Time-Constrained Communications”, Computing Surveys Vol 16 No1 (March 1984) pp 43-70.

[19] Laplante P., “Real-Time systems design and analisys”, IEEE Press 1992.

[20] Lehoczky J P, Sha L and Ding Y, “The Rate Monotonic Scheduling Algorithm. Exact characterization and average case behaviour”, Proc. IEEE Real-Time Systems Symp., 1987.

[21] Lehoczky J.P. and Sha L. Perfomance of Real-time Bus Scheduling Algorithms. *ACM Performance*, Vol.14 , pag. 44-53. 1986

[22] Leung J and Whitehead J, “On the Complexity of Fixed Priority scheduling of periodic real-time tasks”, *Performance Evaluation*, Vol. 2, No. 4, 1982, pp. 237-250.

[23] Liu, C L and Layland J W “Scheduling algorithms for multiprogramming in hard real time environments”, *JACM* Vol 20 No 1 (Jan.1973) pp 46-61.

[24] Locke D.C., "Software Architecture for Hard Real-Time Applications: Ciclic Executives vs. Fixed Priorities Executives", Real-Time Systems, Vol 4 No 1 (Marzo 1992) pp 37-53.

[25] Malcolm, N. and Zhao, W. The timed-token protocol for real-time communications. *IEEE Computer*, 27 (1): 35-41, 1994.

[26] Nadkarni A P and K S Vastola “An Analysis of the Silent Channel Access Method” *Proc. 23d Ann Allerton Conf Monticelo IL* (October 1985).

[27] Obenza, R. 1993. Rate monotonic analysis for real-time systems. *IEEE Computer*, 26(3):73-74.

[28] Orozco J, Santos J, "Diagramación de tareas en sistemas operativos de tiempo real PMC con inversiones de prioridad", *Anales 23 JAIIO*, Buenos Aires 6 al 8 de Septiembre de 1994 pp 1.25-1.35.

[29] Orozco, J., Cayssials, R., Ferro, E. y Santos, J., "Design of a Learning Fuzzy Production System to Solve an NP-Hard Real-Time Assignment Problem". *VIII IEEE Euromicro Workshop on Real Time Systems 96*, L'Aquila, Italia, Junio 1996.

[30] Orozco, J., Cayssials, R., Santos, J., y Ferro, E., "802.4 Rate Monotonic Scheduling in Hard Real-Time Environments: Setting the Medium Access Control Parameters". *Information Processing Letters*, Elsevier Science Publishers, Volume 62, 1997. pp: 47-55.

[31] Orozco, J., Cayssials, R., Santos, J., y Ferro, E., "Precedence Constraints in Hard Real-Time Distributed Systems". *Proceedings Third IEEE International Conference on Engineering of Complex Computer Systems*. Como, Italy. September 8-12, 1997. pp 33-38.

[32] Orozco, J., Cayssials, R., Santos, J. and Santos, R. M. "On the minimum number of priority levels required for the Rate Monotonic scheduling of real-time systems" *10th Euromicro Workshop on Real Time Systems – WIP'98*, Berlín, 1998

[33] Ramamritham K., Stankovic, J., "Schedulling Algorithms and Operating Systems Support for Real time Systems", Technical report, University of Massachusetts, 1993.

[34] Rajkumar R., "Synchronization in Real-Time Systems", *Kluwer Academic Publishers*, USA (1991).

[35] Rothauser E.H. and Wild D. "MLMA: A Collision-Free Multiaccess Method" *Proceedings IFIP Congress'77* Amsterdam The Netherlands North-Holland (1977) pp 431-436.

[36] Rzehak, H., Elnakhal, A. and Jaeger R. "Analysis of Real-Time Properties and Rules for Setting Protocol Parameters of MAP

Networks". *The Journal of Real-Time Systems*, 1,3, pp. 221-241. 1989.

[37] Santos J., Gastaminza M.L., Orozco J., Picardi D. y Alimenti O.; "Priorities and protocols in hard real-time LANs"; *Computer Communications*, Butterworth-Heinemann, pp.: 507 a 514, Vol.14, No.9; 1991.

[38] Santos, J. and Orozco, J. 1994. On the priority mechanism of 802.4, in hard real-time factory communications. *Proc. IEEE International Symposium on Industrial Electronics*, pp. 281-285.

[39] Santos, J. y Orozco, J., "Rate Monotonic scheduling in hard real-time systems", *Information Processing Letters*, Elsevier Science Publishers, No: 48 1993, pp 39-45.

[40] Santos, J., Ferro. E., Orozco. J., y Cayssials, R., "A Heuristic Approach to the Multitask-Multiprocessor Assignment Problem Using the Empty-Slots Method and Rate Monotonic Scheduling". *Real-Time Systems Journal*, Volume 13, Number 2, September 1997. pp: 167-200.

[41] Santos, J., Gastaminza, M.L., Orozco, J. and Matrangolo, C. "802.5 priority machanism in hard real-time RMS applications". *Computer Communications*, Butterworth-Heinemann, 17,6, pp. 439-442. 1994.

[42] Santos, J., Orozco. J. y Alimenti, O., "Redes locales en ambientes con restricciones de tiempo", *VI Simpósio brasileiro de redes de computadores*, Belo Horizonte (Brasil) desde el 28/3 al 30/3 de 1988.

[43] Santos, J., Orozco. J. y Alimenti, O., "Perfomance Evaluation of Standard Lan Protocols in Time Constrained Enviroments". Anales *IEEE INFOCOM'89*, Ottawa, Canadá, 1989.

[44] Santos, J., Orozco. J., "Sistemas Isomorfos en tiempo real duro", J.Santos, J.Orozco, Actas *XIX Conferencia Latinoamericana de Informática (CLEI)* y *XXII Jornadas Argentinas de Informática e investigación operativa*, Buenos Aires, 1993.

[45] Sathaye, S. “Scheduling real-time traffic in packet-switched networks”. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA. 1994.

[46] Scott Mac Lean R “Performance Characteristics of the FDDI Token Ring Priority Mechanism” Technical Report CSRI-230 University of Toronto (August 1989).

[47] Sha Lui, Rajkumar R y Lehoczky J, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization", IEEE Trans on Computers, Vol 39 No 9 (Setiembre 1990) pp 1175-1185.

[48] Sha Lui, "A Systematic Approach to Designing Distributed Real-Time Systems", IEEE Computer, Sep. 1993, pág 68-78.

[49] Sprunt B, Sha L and J. Lehoczky, “Aperiodic task scheduling for hard real-time systems”, Real-Time Systems, Vol 1 No 1 (June 1989) pp 27-60.

[50] Stankovic, J.A. “A serious problem for next-generation systems”, IEEE Computer, Vol 21 No 10 (Octubre 1988) pp 10-19.

[51] Stankovic, J.A., Spuri, M., Di Natale, M., Buttazzo, G., “Implications of Classical Scheduling Results for Real Time Systems”, IEEE Computer, Vol 28 No 6 (June 1995) pp 16-25.

[52] Tindell, K. “Analysis of Hard Real Time Communications”, Real Time Systems Research Group, *Dept. of CC, University of York*, ftp.york.ac.uk/pub.

[53] Tobagi, F.A. “Multiaccess Protocols in Packet Communication Systems”, *IEEE Transactions on Communications* Vol. COM-28, No.4, pp. 468-487 (April 1980).

[54] Warren, C., “Rate Monotonic Scheduling”, IEEE Micro, Vol 11 No 3 (June 1991) pp 34-38, 102.

[55] Wilkes M.V. and Needham N.R., “The Cambridge Digital Communication Ring” Proc. of Local Area Communication Symposium, Boston (May 1979) pp 49-60.