UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ESTUDIO DE SISTEMAS RECONFIGURABLES EN TIEMPO REAL CON BASE EN UN ESQUEMA MULTIAGENTE REACTiVO

TESIS

QUE PARA OPTAR POR EL GRADO DE:
DOCTOR EN INGENIERÍA

PRESENTA:
OSCAR ALEJANDRO ESQUIVEL FLORES

TUTOR**:**
DR. HÉCTOR BENÍTEZ PÉREZ
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

COMITÉ TUTOR:
DR. SERGIO MARCELLIN JACQUES
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

DR. LUIS AGUSTÍN ÁLVAREZ ICAZA LONGORIA
INSTITUTO DE INGENIERÍA

MÉXICO, D.F. ENERO 2013.

JURADO ASIGNADO:

Presidente:	Dr. Sergio Marcellin Jacques
Vocal:	Dr. Héctor Benítez Pérez
Secretario:	Dr. Luis Álvarez Icaza Longoria
Suplente:	Dr.Demetrio Fabián García Nocetti
Suplente:	Dr. Jorge Ortega Arjona

Lugar donde se realizó la tesis:

Departamento de Ingeniería en Sistemas Computacionales y Automatización (DISCA) del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) de la Universidad Nacional Autónoma de México (UNAM).

DIRECTOR DE TESIS:

Dr. Héctor Benítez Pérez

Este trabajo se desarrolló en el Departamento de Ingeniería en Sistemas Computacionales y Automatización (DISCA) del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) de la Universidad Nacional Autónoma de México (UNAM) bajo la dirección del Dr. Héctor Benítez Pérez. Se contó con el valioso apoyo de una beca para estudios doctorales del Consejo Nacional para la Ciencia y Tecnología (CONACYT). Apoyos adicionales fueron proporcionados por el Posgrado en Ciencia e Ingeniería de la Computación, el Programa de Apoyo a los Proyectos de Investigación Tecnológica PAPIIT IN 103310 y el Proyecto ICYTDF PICCO 1053.

A la memoria de Adela Mejía, Antonio Flores y Nicolasa Razo.

A Manuel Esquivel.

Mis abuelos, ejemplos para mí de modestia, voluntad y bondad.

Ma ontlami immixtzi, immoyollotzin.
No desaparezca tu rostro, tu corazón.
No pierdas tu propia estima.

In ticuauhtin, in tocelome.
Somos águilas, somos ocelotes.
Somos los que tenemos fuerza, decisión.

Zan ticqualtiliz immocuic, immotlahtol.
Sólo tú harás bueno tu canto, tu palabra.
De ti depende hablar con honestidad, con sinceridad.

FIGURAS POÉTICAS DEL NÁHUATL CLÁSICO.

Agradecimientos

Durante el desarrollo de este trabajo hubo muchas personas que de diferente forma me impulsaron, apoyaron y orientaron para materializar este logro. Aprecio a cada amistad, profesor, compañero, familiar quienes, aún con una sugerencia breve o un ímpetu sencillo, siempre me alentaron para continuar. Sinceramente, ¡gracias!

Agradezco especialmente:

A Ma. del Carmen Flores y Juan G. Esquivel, mis padres, por su capacidad para superar constantemente dificultades y limitaciones. Gracias por darme amor, protección y educación.

A todos los compañeros del Posgrado en Ciencia e Ingeniería de la Computación y de otros posgrados que con su ayuda y asesoría me dieron la pista para aclarar ideas y asimilar conceptos. A todos ustedes les externo mi reconocimiento, son muy brillantes.

A los amigos, cariños y amores que durante el camino hacia la obtención del grado de PhD llegaron, se fueron o que permanecen aún en mi periferia por sufrirme y comprender mi condición de posgraduante neurótico, impaciente e inconstante.

Al pueblo de México por continuar sosteniendo la educación pública y gratuita a pesar que cada vez sea más difícil el esfuerzo.

Al CONACYT por brindarme el apoyo económico para mis estudios doctorales, a la Coordinación del Posgrado en Ciencia e Ingeniería de la Computación por el apoyo económico proporcionado para presentar artículos en congresos nacionales e internacionales. Además, al Programa de Apoyo a los Proyectos de Investigación Tecnológica PAPIIT IN 103310 y al Proyecto ICYTDF PICCO 1053 que complementaron algunos gastos de verdad importantes.

A todos y cada uno de los revisores que invirtieron tiempo y paciencia en el arbitraje de resultados preliminares de este trabajo. En particular, agradezco a los revisores de la *Revista Iberoamericana de Automática e Informática Industrial* por iluminar el camino en un momento verdaderamente oscuro.

A los profesores que participaron en mi examen de candidatura, Dr. Sergio Rajsbaum y Dr. Gonzalo Barajas, por compartir su conocimiento y estimular mi trabajo con sus apreciaciones.

A los profesores que formaron parte de mi comité tutoral y que semestralmente enfrentaron el momento difícil de encausar el rumbo de mis conjeturas muchas veces erróneas: Dr. Sergio

Marcellin por su profesionalismo y claridad al exigirme un trabajo pulcro. Al Dr. Luis Álvarez Icaza por dedicarme tan generosamente su tiempo y transmitirme de manera sencilla, amable y precisa su conocimiento tan amplio.

Al Dr. Jorge Ortega por armonizar y darle sensatez a mis escritos, ideas y presentaciones, siempre acompañado de agudos comentarios y reflexiones simpáticas.

Finalmente, a mi director de tesis, Dr. Héctor Benítez Pérez, para quien el trabajo no es lo más importante, sólo que no se posterga innecesariamente. Valoro sobremanera tu conducción, tu amistad y los incesantes apremios diarios para obtener el grado de Doctor. Mi más sincero, profundo y entrañable agradecimiento por compartirme tu visión sobre construir, avanzar, lograr...

Kyoto
Noviembre, 2012.

Índice general

Índice de figuras

Índice de tablas

Estudio de Sistemas Reconfigurables en Tiempo Real con Base en un Esquema Multiagente Reactivo
Tesis Doctoral

Oscar Alejandro Esquivel Flores
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México

Resumen

El avance de la tecnología en las dos últimas décadas ha impulsado la fusión del control y comunicación en los sistemas industriales. La integración de dispositivos distribuidos por medio de redes de comunicación potencia el procesamiento, la ejecución y la toma de decisiones del sistema. Sin embargo, la utilización de estos sistemas distribuidos para el control de procesos que requieren altos grado de precisión y confianza conduce a la necesidad de generar estrategias para diseñar sistemas confiables, capaces de responder ante incertidumbres o fallas durante la ejecución. Este reto aumenta cuando se añade el cumplimiento de restricciones de tiempo y los recursos con los que se cuenta son limitados. Una estrategia para atacar esta problemática es la reconfiguración de la estructura del sistema.

Las propuestas de reconfiguración se orientan a modificar el controlador, sustitución de dispositivos o cambio de los parametros de planificación, lo que favorece la calidad del control, el cumplimiento de los plazos de tiempo de las tareas que se llevan a cabo y el manejo de los recursos. La consecuencia es un sistema confiable y seguro aún cuando se den degradaciones inherentes al proceso.

En este trabajo se estudian los sistemas de control en red en entornos de tiempo real con la finalidad de elaborar estrategias de reconfiguración que disminuyan el impacto que provoca la presencia de incertidumbres o fallas en el sistema y que promuevan un manejo óptimo de los recursos disminuyendo los efectos que provoca el uso del medio de comunicación en el desempeño del sistema. Las estrategias de reconfiguración que se presentan en este trabajo están diseñadas con base en el enfoque multiagente que aprovecha el uso dispositivos inteligentes en los sistemas bajo estudio, lo cual favorece la toma decisiones de forma conjunta lográndose la obtención de objetivos comunes. También se propone el manejo de las frecuencias de transmisión para planificar la tasa de datos a través del medio de comunicación, con esto se evita el modelado de los retardos de tiempo y las conjeturas que sobre éstos se hacen para facilitar su manejo.

La aportación de este trabajo consiste en proponer, analizar y evaluar la efectividad de dos aproximaciones de reconfiguración dinámica cuyo objetivo es mantener la certidumbre del sistema, aún ante la presencia de fallas, y garantizar un índice de desempeño en la respuesta del sistema.

Por medio de simulaciones numéricas se implementa y evalúa la viabilidad y precisión de las propuestas aquí planteadas y se toma como caso de estudio un prototipo de helicóptero con dos grados de libertad.

Study of Real-Time Reconfigurable Systems Based on a Reactive Multi-Agent Scheme
PhD Thesis

Oscar Alejandro Esquivel Flores
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México

Summary

The advancement of technology in the last two decades has boosted the fusion of communication and control in industrial systems. The integration of distributed devices using a communication network enhance processing, implementation and decision-making of the distributed systems. Nevertheless, distributed control systems working on processes that require high degree of accuracy and reliability lead to the need to develop strategies to design systems with the ability to respond to uncertainty during execution. This challenge increases when adding time restrictions and limited resources. A strategy to tackle this problem is to reconfigure the system structure.

Reconfiguration proposals aim to modify the controller, replacement of devices or scheduling parameters changes, in order to guarantee a quality control level, meeting the deadlines of the tasks that take place in the system and optimum resource management. In consequence, the system continues still to be reliable and dependable even when inherent degradation of the process occurs.

In this thesis networked control systems in real-time environments are studied in order to develop reconfiguration strategies to reduce the effect of uncertainties and faults during the life-time of the system. Reconfiguration strategies presented in this work are designed based on multi-agent approach which uses the properties of smart devices to improve joint decisions in order to achieve a common goal. Also, this proposal manage transmission frequencies to schedule the volume of data sent through the communication network, this avoids to model time delays and several assumptions on them.

The main contribution of this work consists on proposing, analyzing and evaluating the effectiveness of two dynamic reconfiguration approaches in order to maintain the reliability of the system, and proper management communication media, even in the presence of faults, and ensure a level of performance.

Numerical simulations are implemented to evaluate the feasibility and accuracy of the proposals. A prototype of helicopter with two grade of freedom is used as case of study.

Capítulo 1

Introducción

En años recientes, la reconfiguración forma parte de las estrategias para mantener el funcionamiento y cierto nivel de desempeño de un sistema dinámico, aún bajo la presencia de fallas. Como punto de partida, entenderemos a la reconfiguración como una acción que modifica la estructura original de un sistema dinámico de manera que la representación de sus estados se modifica. Esta técnica ha sido utilizada comúnmente como herramienta para el manejo y aislamiento de fallas; no obstante, su utilización se ha manifestado en áreas clásicas de la ingeniería, como es el control.

Por otra parte, las arquitecturas distribuidas han aumentado su presencia en la industria debido, principalmente, a su bajo costo y mantenimiento sencillo. Así, se han implementado sistemas de control distribuido que aprovechan las ventajas del cómputo y la comunicación, lo que ha resultado en los sistemas de control en red. Tales sistemas constituyen una subclase de los sistemas de tiempo real en los cuales el valor de cada tarea no depende sólo de su exactitud en el cálculo sino también en el tiempo en que el resultado esté disponible. Varios son los retos a los que se enfrenta el diseño y análisis de los sistemas de control en red: cumplir con las restricciones temporales, planificar el conjunto de tareas, utilizar de manera óptima los recursos disponibles y mantener un nivel de desempeño, entre otros. Lo anterior ha motivado varias líneas de investigación enfocadas en resolver los problemas que se presentan en el análisis, diseño e implementación de los sistemas distribuidos en tiempo real.

Un enfoque para analizar los sistemas distribuidos en tiempo real, basado en el uso de sensores inteligentes y el avance en la tecnología de actuadores, es el enfoque multiagente reactivo. Esta forma de conceptualizar un sistema distribuido aprovecha la descentralización de componentes, la autonomía de cada uno de ellos y la capacidad de colaboración para lograr objetivos comunes, en el diseño de nuevos esquemas de reconfiguración dinámica.

Reconfiguración, tiempo real, planificación, sistemas multiagentes y sistemas de control en red son las áreas que confluyen en este trabajo. Se hace una revisión de las arquitecturas distribuidas, se estudian los principios del tiempo real y revisan, con cierta profundidad, los algoritmos de planificación por prioridades. Se examinan las características de los sistemas de control en red, las distintas problemáticas que surgen en su funcionamiento, y las líneas de investigación que han surgido para analizar y diseñar sistemas de esta clase que sean más eficientes. Se propone utilizar el enfoque multiagente reactivo para modelar la estructura de mecanismos de reconfiguración descentralizados y acotados, que disminuyan el efecto de las

fallas e incertidumbres en el sistema y aseguren ciertos niveles de control y servicio de los recursos.

En este trabajo se plantean dos estrategias de reconfiguración dinámica:

1. La primera se enfoca en atenuar el efecto de distintos tipos de fallas que se presentan en el sistema distribuido en tiempo real. Por medio de la comunicación de los componentes distribuidos, entendidos como los agentes del sistema, se realiza un trabajo cooperativo que resulta en un acuerdo distribuido que define el tipo de reconfiguración que va a realizar.
2. La segunda estrategia utiliza las frecuencias de transmisión de datos que se envían por la red, con la finalidad de planificar la utilización este elemento compartido. El comportamiento dinámico de las frecuencias de transmisión de datos es caracterizado por medio de un modelo lineal invariante en el tiempo, que al ser controlado equilibra el uso de la red. El objetivo es enviar datos por la red de comunicación con una frecuencia delimitada dentro de una región en la cual la comunicación no impacte en el rendimiento del sistema.

Ambas estrategias son implementadas en el modelo de un helicóptero de dos hélices y por medio de simulaciones numéricas en *Matlab/Simulink* se estudia la respuesta temporal del sistema usando el error cuadrático medio como índice de desempeño. Se utiliza el software *Truetime* como herramienta de simulación de tiempo real. Con base en este análisis se obtienen los criterios para evaluar la eficiencia de tales propuestas.

A continuación se proporciona la motivación para realizar este trabajo, los objetivos que conducen la investigación, la metodología utilizada, las metas que se pretenden alcanzar, la contribución que aporta esta investigación al conocimiento en el campo de la reconfiguración de sistemas distribuidos y una breve síntesis de los contenidos del presente trabajo.

Motivación

Con el gran avance de la tecnología en la primera década de este siglo, se ha facilitado reunir sistemas de cómputo compuestos por un gran número de CPUs, conectados mediante una red de alta velocidad (Tanenbaum y Van-Renesse, 1985; Tanenbaum, 1995).. Estos sistemas distribuidos se han usado en la industria, el comercio y la investigación, debido a su flexibilidad en términos de poco esfuerzo y corto tiempo para su implantación, además de ser confiables en el sentido de garantizar ciertos niveles de funcionalidad ante la presencia de fallas e incertidumbres. La respuesta de los sistemas distribuidos para continuar su operación en una situación de falla es modificar estructuralmente su entorno (Almeida, 2001).

Actualmente, los sistemas distribuidos se ejecutan en ambientes de tiempo real y se utilizan en aplicaciones críticas, como el control de automóviles y aeronaves, plantas nucleares, lanzamientos de naves espaciales y robótica. La característica principal de un sistema distribuido en tiempo real (RTDS por *Real Time Distributed Systems*) consiste en que la validez de la respuesta que proporciona no es suficiente, también debe darse dentro de un plazo determinado, es decir, no depende únicamente de los resultados de las operaciones que genere, sino también del instante en que los resultados se producen (Stankovic, 1988).. Por tanto, diseñar y analizar un RTDS es un problema difícil de enfrentar para los diseñadores de

sistemas en tiempo real debido a que los RTDS están envueltos normalmente en ambientes complejos y altamente no-determinísticos, donde se tiene que garantizar el cumplimiento de los requisitos temporales, tolerar fallas y tener un grado considerable de predicibilidad (Stankovic, 1995).

Cuando un sistema de control está cerrado por medio de un canal de comunicación compartido por otras entidades, entonces el sistema de control es clasificado como un sistema de control en red (NCS por *Networked Control System*) (Gupta y Mo-Yuen, 2010). Las tendencias actuales en la investigación en control y sus aplicaciones se orientan a la integración de componentes distribuidos por medio de redes de comunicación para lograr cómputo, sensado y ejecución distribuida. El estudio de los NCS se ha incrementado considerablemente en los últimos 10 años debido a que estos sistemas han resultado de gran efectividad en la industria (Li y Wang, 2008). Sin embargo, existen claras desventajas de utilizar un medio de comunicación común para realizar el control. Entre ellas destacan los retardos de tiempo inducidos por la red y el retardo inherente en los sistemas de control digital. Además, mientras aumenta la demanda de uso de la red, el tráfico provoca una latencia de datos considerable, la cual impacta en la estabilidad y desempeño del sistema de control. Los efectos negativos de la latencia de datos en el rendimiento del sistema dinámico se agravan por la pérdida de datos debida a la saturación de los *buffers* (Ray, 1989*).* Por lo tanto, varios factores deben ser considerados para obtener un nivel de desempeño óptimo en un NCS que no se centran solamente en el diseño del controlador sino también en la transmisión de los datos por la red (Balbastre, 2002).

Recientes investigaciones han utilizado distintas metodologías para enfrentar los problemas anteriores: el modelado y control de los retardos (Halevi y Ray, 1988a; Nilsson, 1998;Lian y otros, 2001c; Méndez-Monroy y Benítez-Pérez, 2009), la elección de periodos de muestreo que eviten sobrecarga del sistema (Lian y otros, 2002, 2003; Méndez-Monroy y Benítez-Pérez, 2011a; Peng y otros, 2008), el control reconfigurable (Quiñones-Reyes y otros, 2007; Benítez-Pérez y García-Nocetti, 2005; Méndez-Monroy y otros, 2009), la integración de la comunicación y control (Arzen y otros, 2000; Törngren y otros, 2006; Xia y Sun, 2008b) y planificación retroalimentada (Cervin y Eker, 2000; Sename y otros, 2003; Xia y Sun, 2008a; Zhou y Xie, 2005).

En este contexto, la reconfiguración desempeña un papel importante pues contribuye a mantener el sistema en condiciones de funcionamiento tales que el rendimiento permanezca en un nivel aceptable y se evite la suspensión o reinicio como respuesta a la presencia de fallas. El objetivo de la reconfiguración dinámica es permitir al sistema físico cambiar de una configuración a otra en tiempo de ejecución sin que esta modificación impacte demasiado en la ejecución del sistema (Kramer y Magee, 1985). La configuración inicial de un sistema está definida por medio de una estructura de entidades de hardware y software, con lo que la reconfiguración dinámica acarrea operaciones de remplazo, modificación, migración, adición y remoción de tales entidades; acciones que deben realizarse sin que afecten el cumplimiento de las restricciones temporales. Un modelo de reconfiguración dinámica es propuesto por Almeida (2001); Almeida y otros (2001),, que se centra en el diseño estructural previo del sistema y en las modificaciones de reconfiguración.

Algunas estrategias que se han desarrollado para obtener cierto nivel de servicio (QoS) y calidad de control (QoC) utilizan control reconfigurable (Benítez-Pérez y García-Nocetti, 2005; Benítez-Pérez y otros, 2005, 2007a, 2010; Quiñones-Reyes y otros, 2007), propuestas en las que se responde dinámicamente a variaciones en los retardos de tiempo. El control

reconfigurable se basa en los dispositivos inteligentes (Benítez-Pérez y García-Nocetti, 2003) para la toma de decisiones. Una generación nueva de aplicaciones distribuidas se caracteriza por la cooperación entre entidades. En últimos años el uso de dispositivos inteligentes permite integrar distintas funcionalidades, como el sensado de datos y la identificación de fallas, además del intercambio de información entre dispositivos. Estos elementos inteligentes o *agentes* pueden entenderse como entidades de hardware o software, que recopilan información de su entorno, poseen la capacidad de decidir sobre sus acciones y colaboran entre sí para lograr un objetivo común (Wooldridge y Jennings, 1995; Corkill, 2003). Un agente es por tanto un nodo del sistema distribuido y a su vez es un dispositivo inteligente capaz de tomar decisiones consensuadas que influyen en la ejecución del sistema físico.

De esta forma, la reconfiguración dinámica enfrenta distintos retos al ser utilizada como estrategia de tolerancia a fallas en sistemas que integran el control y la comunicación en entornos de tiempo real. En este trabajo se mostra que utilizar el enfoque multiagente beneficia los mecanismos de reconfiguración al aprovechar de la distribuición de componentes, los dispositivos inteligentes y la comunicación.

Objetivo

El objetivo general de este trabajo consiste en:

Estudiar el impacto de reconfigurar sistemas dinámicos que se ejecutan en entornos con restricciones temporales bajo un ambiente concurrente y distribuido.

Los objetivos particulares consisten en:

- *Estudiar las características y ventajas de los sistemas multiagente para construir esquemas de reconfiguración dinámicos.*
- *Analizar el efecto de los sistemas multiagente sobre el comportamiento dinámico de un sistema distribuido en tiempo real con base en una representación en ecuaciones en diferencias de su conducta.*
- *Diseñar esquemas de reconfiguración dinámica de sistemas distribuidos bajo escenarios con falla de manera que se cumplan los requerimientos de tiempo real.*

Metodología

La metodología utilizada en este trabajo consiste en plantear los antecedentes relativos al tiempo real, planificación, esquemas de reconfiguración y las consideraciones que se toman en cuenta para el análisis y diseño de los NCSs, y las arquitecturas multiagente. Por medio de la revisión de los trabajos relacionados que se han publicado en años recientes respecto a los NCSs, y los trabajos clásicos en materia de planificación, se esbozan las problemáticas que surgen al diseñar sistemas de control de manera distribuida y el efecto de reconfigurarlos de manera estática. Basado en lo anterior, se proponen esquemas de reconfiguración utilizando la arquitectura multiagente, con la finalidad de atenuar el efecto de alguna posible falla en los agentes del sistema y que permitan mantener ciertos niveles de estabilidad/rendimiento. Se analiza un caso de estudio de un sistema de control en red y por medio de simulaciones numéricas se implementan los esquemas de reconfiguración propuestos evaluando su funcionamiento con base en el cálculo del valor absoluto del error de salida.

De manera general, los objetivos de este trabajo se evalúan al mostrar que los esquemas de reconfiguración planteados son viables en el caso de estudio elegido y son ejecutados dentro de intervalos de tiempo acotados.

Metas

Se establecieron como metas:

1. Plantear una arquitectura híbrida para realizar la reconfiguración, que cumpla con los requerimientos de los sistemas de tiempo real y aproveche las características de los modelos multiagente.
2. Verificar la viabilidad de la reconfiguración de la estructura de un RTDS como estrategia eficaz para el manejo de fallas y para el uso óptimo los recursos computacionales compartidos.

Contribución

Las contribuciones de este trabajo son:

1. El diseño de esquemas de reconfiguración dinámica, restringida al análisis de planificación del RTDS, asegurando con esto una transición de estados sin alterar los plazos de ejecución.
2. Proponer un esquema de reconfiguración con base en el manejo de frecuencias de transmisión de datos que, al ser controlable, equilibra la cantidad de datos que son transmitidos por la red. Esto con el propósito de evitar la presencia de retardos, pérdida de datos y sobrecarga.

Logros

Se presenta un estudio de la reconfiguración en entornos distribuidos con restricciones temporales y se valida esta estrategia como una herramienta efectiva para el manejo de escenarios con falla que provocan déficit en el desempeño del sistema. La reconfiguración que aquí se plantea disminuye la complejidad del modelado y control de retardos, sin hacer reducciones excesivas sobre el modelado de la red y los retardos de tiempo. Los esquemas de reconfiguración propuestos mostraron ser efectivos, respecto al valor absoluto del error de salida, en las simulaciones realizadas.

Organización de la Tesis

Este trabajo se desarrolla en siete capítulos, incluyendo esta introducción, y una sección adicional de apéndices que complementan la investigación.

En el Capítulo 2, se presentan los **Antecedentes**. Se hace una revisión del modelo de reconfiguración planteado por Almeida (2001) y se plantean los conceptos fundamentales relativos a los sistemas distribuidos, sistemas de tiempo real, planificación y arquitecturas

multiagente; exponiendo las condiciones de planificabilidad para los algoritmos de planificación por prioridades.

En el Capítulo 3, **Sistemas de Control en Red**, se presentan los sistemas de control en red y las líneas de investigación desarrolladas para su análisis y diseño. Se exponen los efectos del cambio en los parámetros de tiempo en los NCSs o de variaciones en la cantidad de datos transmitidos. Se expone una propuesta de arquitectura multiagente para el manejo de fallas.

En **Impacto de la Tasa de Transmisión de Datos en los NCSs**, capítulo 4, por medio de un ejemplo introductorio se plantea la problemática de elegir un periodo de sensado en un NCS de forma que equilibre el uso de la red y la efectividad del control. Se detalla el caso de estudio consistente en el control de un helicóptero con dos grados de libertad, se resalta el modelado matemático del sistema físico, el modelo del controlador de vuelo y sus características como un sistema distribuido.

En el Capítulo 5, **Reconfiguración por Medio de Parámetros de Planificación**, se desarrolla una estrategia de reconfiguración con base en el acuerdo entre los agentes encargados de hacer la lectura de los estados del sistema distribuido. Se plantean las acciones de estos agentes, el acuerdo consensuado y se hace el análisis de planificación para asegurar el mínimo impacto de esta reconfiguración.

En **Reconfiguración con Base en el Cambio de Frecuencias de Transmisión**, capítulo 6, se desarrolla la propuesta de modificación de frecuencias de transmisión de datos como estrategia de reconfiguración dinámica. Se desarrolla el concepto de las regiones de planificabilidad local y global dentro de las cuales el NCS elegido tiene un comportamiento estable y se plantea el modelo lineal invariante en el tiempo de las frecuencias de transmisión. El capítulo cierra con el análisis de planificación de este tipo de reconfiguración.

En el Capítulo 7, **Simulaciones Numéricas**, se verifica la validez y precisión de los esquemas reconfiguración propuestos simulando el entorno de tiempo real con la herramienta de simulación *TrueTime*, cuyos módulos se basan en *Matlab/Simulink*. Se calcula el valor absoluto del error bajo tres distintos escenarios: libre de falla, con falla, y falla con reconfiguración; y se contrastan para evaluar la efectividad de la reconfiguración.

Se concluye el trabajo con una sección de **Conclusiones y Trabajo Futuro**, capítulo 8, en donde se exponen los comentarios finales y se sugiere el trabajo futuro.

Se anexan en la sección **Apéndices** los apartados: cota de utilización del algoritmo de planificación *Rate Monotonic*, esquema de control de regulación cuadrática lineal (LQR), descripción de los módulos y funciones primitivas con los que funciona la herramienta de simulación de Tiempo Real *TrueTime*, modelos en *Matlab/Simulink* utilizados para simular el control del RTDS elegido como caso de estudio, los códigos en *Matlab* desarrollados para las simulaciones y la producción científica generada por este trabajo.

Capítulo 2

Antecedentes

En este capítulo se exponen los conceptos básicos en los que está enmarcada esta investigación. El propósito consiste en introducir las características de los sistemas distribuidos, principalmente en su arquitectura e interacción entre sus elementos. Se expone el modelo básico de reconfiguración y se plantean los principios elementales de los sistemas en tiempo real resaltando los algoritmos de planificación de tareas periódicas. Se presenta el enfoque de la inteligencia artificial distribuida que conlleva el enfoque de los sistemas multiagente y se finaliza con una revisión de las características de un sistema tolerante a fallas, meta a alcanzar con la reconfiguración. Con este conjunto de elementos se construye el andamiaje del análisis y diseño de la reconfiguración de sistemas distribuidos en tiempo real con base en un esquema multiagente.

2.1 Sistemas distribuidos

En años recientes se han fabricado dispositivos de alta complejidad capaces de calcular a alta velocidad y manejar memoria masivamente, lo que ha resultado en mejores sistemas distribuidos, en el sentido de dejar de ser tan restringidos y rígidos tanto en su construcción como en su comportamiento. Actualmente, se han experimentando los efectos de conjunción de avances significativos en la tecnología de la información y comunicación. Por el momento, el efecto más significativo es la llegada de la revolución producida por los circuitos integrados, especialmente en forma de VLSI (*very large scale integration*) cuyo resultado ha sido la mejora enorme en la capacidad de procesamiento, almacenamiento, comunicación y el crecimiento de las redes de computadoras. Asimismo, surge la necesidad de resolver problemas como la distribución de recursos y la comunicación entre las diferentes redes o subredes que pueden ser o no del mismo tipo y que de principio no fueron diseñadas para comunicarse entre ellas. Las ciencias de la computación han puesto su interés en estos aspectos y han creado un área de conocimiento conocido como sistemas distribuidos.

Según Tanenbaum y Van-Steen (2007) un sistema distribuido (SD) es:

> *Una colección de computadoras independientes que el usuario percibe como un único sistema*.

Esta definición considera la unión de componentes individuales que colaboran para obtener un objetivo común.

Para Coulouris y otros (2012) un sistema distribuidos es:

Un sistema en el cual componentes de hardware o software, localizados en computadoras conectadas en red, comunican y coordinan sus acciones únicamente por el paso de mensajes.

Esta definición precisa el modelo físico mínimo de un SD como un conjunto expansible de nodos interconectados por una red de computadoras para el paso de mensajes.

Una de las motivaciones principales para el uso de un SD es compartir recursos pero manteniendo servicios confiables, por lo que deben utilizarse técnicas de replicación al fallar alguno de los componentes. Al diseñar un SD deben tomarse en cuenta ciertas características para que las aplicaciones tengan un desempeño confiable, que tienen que ver con aspectos como la diversidad de los equipos de cómputo, la tecnología utilizada en la comunicación, la diferencia en los sistemas operativos, la disponibilidad de las aplicaciones, las limitantes en el flujo de la información y los mecanismos de transmisión segura de información. El campo de estudio de los SDs se encarga de estudiar y proponer soluciones a problemas que se presentan en la construcción de aplicaciones distribuidas. Tales aplicaciones deben contar con características propias de la arquitectura del sistema y que por sí mismas puede ser un verdadero reto en su implementación; estas características se refieren a (Coulouris y otros,2012):

- *Escalabilidad:* Si el sistema permanece efectivo y eficiente cuando hay un incremento o reducción en el número de recursos y de usuarios se dice que el sistema es escalable.

- *Seguridad:* Es necesario contar con un mecanismo de seguridad que garantice la confidencialidad, integridad y disponibilidad de datos.

- *Extensibilidad:* Esta característica se refiere a un sistema que puede ser ampliado y reimplementado de diversas formas. La extensibilidad de un SD está determinada por el número de recursos nuevos que pueden ser agregados y puestos a disposición para el uso de distintos programas clientes.

- *Manejo de fallas:* Es vital contar con mecanismos que prevengan o corrijan los errores que un sistema puede genera. En los SDs se busca aislar la falla debido a que los componentes se encuentran separados; técnicas como la replicación y enmascaramiento de fallas son idóneos para este tipo de sistemas.

- *Concurrencia:* Tanto los servicios como las aplicaciones proveen recursos que pueden ser compartidos por distintos componentes en un SD, por lo que existe la posibilidad que varios componentes accedan a un mismo recurso compartido al mismo tiempo. Entonces, es necesario que el recurso pueda atenderlos concurrentemente, garantizando el funcionamiento correcto en cada solicitud. Para garantizar consistencia de datos se aplican técnicas de sincronización como semáforos, banderas y paso de mensajes.

- *Transparencia:* Es el encubrimiento de la separación de componentes del SD desde la visión del usuario, el sistema es un todo, no sólo una colección de componentes independientes.

- *Heterogeneidad:* Las redes de cómputo están formadas por equipos de cómputo autónomos que en general son diferentes. La heterogeneidad se aplica en el tipo de red, sistema operativo, lenguajes de programación e implementación de las aplicaciones.

- *Calidad de Servicio* (QoS): Los usuarios se benefician de los servicios que provee el SD y la calidad con el que se ofrece. Las principales propiedades de los sistemas que afectan la calidad del servicio son la confiabilidad, seguridad y desempeño. La adaptabilidad para satisfacer las cambiantes configuraciones del sistema y la disponibilidad de los recursos es un aspecto importante en la calidad del servicio. El aspecto del rendimiento de la calidad del servicio se definió originalmente en términos de la respuesta y el rendimiento computacional, pero se ha redefinido en términos de capacidad para cumplir con las garantías de temporales.

En general se pueden distinguir tres tipos de SDs: sistemas de cómputo distribuido, sistemas de información distribuidos, sistemas distribuidos embebidos (Tanenbaum y Van-Steen,2007).

- Los *sistemas de cómputo distribuido* están subdivididos en clusters de computadoras (*cluster computing*) y en mallas de cómputo (*grid computing*). Los clusters de computadores son una colección de estaciones de trabajo o PCs con características similares, conectadas por medio de una red local de alta velocidad, y cada nodo corre el mismo sistema operativo. En las mallas de cómputo los sistemas son construidos como federaciones de sistemas de computadoras donde cada sistema es regido por una administración diferente y pueden ser totalmente diferentes en el hardware, software y la tecnología de red utilizada.

- Los *sistemas de información distribuidos* son aquellos en los que se tiene una gran cantidad de de aplicaciones en red y se desea lograr interoperabilidad entre los distintos componentes. Un ejemplo de estos sistemas son las aplicaciones de bases de datos.

- Los *sistemas embebidos* también son citados como sistemas penetrantes (pervasive systems). En estos sistemas están caracterizados por ser pequeños, móviles, utilizan baterías y su comunicación es inalámbrica (aunque no todas estas características se deben conjuntar en un solo dispositivo). Ejemplos de estos sistemas son los dispositivos domésticos como PDAs, teléfonos inteligentes; los dispositivos para el cuidado de la salud conectados a una red de área corporal que están monitoreando distintos órganos (Tanenbaum y Van-Steen, 2007) y las redes de sensores utilizadas para el procesamiento de información.

Modelos descriptivos de SDs. Coulouris y otros (2012) mencionan tres modelos descriptivos bajo las cuales el diseño de un SD puede ser caracterizado y analizado:

1. *Modelo físico:* Considera los tipos de computadoras, dispositivos y su interconectividad que constituyen al SD.
2. *Modelo de arquitectura:* Describe un sistema en términos de las tareas computacionales y de comunicación que efectúan los componentes individuales u organizados en grupos.
3. *Modelo fundamental:* Consiste de una abstracción dirigida a describir soluciones a problemas particulares que enfrentan los SD.

Es de interés para este trabajo estudiar ciertas características de la arquitectura y de la interacción entre procesos en los SDs. A continuación se revisan de manera general varios aspectos de los modelos de arquitectura con el propósito de definir ciertos elementos en los que se basa el análisis y diseño de la reconfiguración de los SDs.

La arquitectura de un SD la consideraremos como la estructura que posee el sistema en términos de sus componentes individuales y sus interrelaciones. Coulouris y otros (2012) mencionan cuatro elementos que constituyen las arquitecturas distribuidas modernas: el tipo de entidades que se están comunicando en el sistema distribuido, cómo éstas se están comunicando (paradigmas de comunicación), la tarea específica que cada componente realiza y la colocación de cada componente dentro del sistema:

- *Entidades de comunicación.* Desde una perspectiva de sistemas, las entidades que se comunican en un SD son los *procesos* que tienen paradigmas de comunicación entre ellos. Desde la perspectiva de programación se consideran los objetos, componentes y, actualmente, los servicios de red.

- *Paradigmas de comunicación.* Tres tipos de paradigmas de comunicación son considerados: comunicación interprocesos, invocación remota, comunicación indirecta. La comunicación interproceso se refiere a la comunicación a bajo nivel entre los procesos del sistema distribuido que incluye primitivas de paso de mensajes, acceso directo a APIs y comunicación multicast[1]. La invocación remota es el paradigma de comunicación más común de los sistemas distribuidos, cubre una amplia gama de técnicas basadas en un intercambio bidireccional de información entre las entidades, y da lugar a la operación, procedimiento o método remoto. Estas formas de comunicación constituyen el intercambio en dos direcciones: emisor-receptor. La comunicación indirecta incluye comunicación grupal, cola de mensajes y memoria compartida distribuida, entre otros.

- *Roles y responsabilidades.* En el sistema distribuido los procesos, componentes o servicios, interactúan con los demás para ejecutar varias actividades; al hacerlo cada elemento adopta un rol que es fundamental para establecer la arquitectura que se va a adoptar. Dos estilos arquitectónicos que se derivan del rol de los procesos individuales son: cliente-servidor y par a par (*peer-to-peer*).

- *Colocación.* Considerar el lugar donde los objetos o servicios se van a situar en la infraestructura física repercutirá en la complejidad de la red que conectará al grupo de computadoras. La colocación de componentes és crucial en el rendimiento, confiabilidad y seguridad del SD. Las estrategias de colocación de componentes se enfocan en el mapeo de servicios a múltiples servidores, almacenamiento de objetos en la cache, uso de código móvil[2], y agentes móviles[3].

Para Coulouris y otros (2012), dada la complejidad de los SDs, es de utilidad organizar los servicios que provee en capas (*layering*). En este tipo de organización la complejidad del sistema está particionada en varias capas o niveles, donde cada capa hace uso de los servicios ofrecidos por la capa superior. Una capa dada, por tanto, ofrece una abstracción de software y hardware sin que capas superiores o inferiores sean conscientes de como es implementada.

En la figura 2.1 se muestra la organización de capas (Coulouris y otros, 2012):

[1] Forma de transferencia de datos en donde es posible enviar información de un solo emisor a muchos puntos diferentes (receptores) simultáneamente.

[2] Los Applets son bien conocidos y utilizados ampliamente como código móvil. El usuario que ejecuta un navegador selecciona un vínculo a un Applet, cuyo código se almacena en un servidor web, el código se descarga en el navegador y se ejecuta allí (Coulouris y otros, 2012).

[3] Consiste de un código en ejecución que viaja de una computadora a otra en la red llevando acabo una tarea como recolección de información.

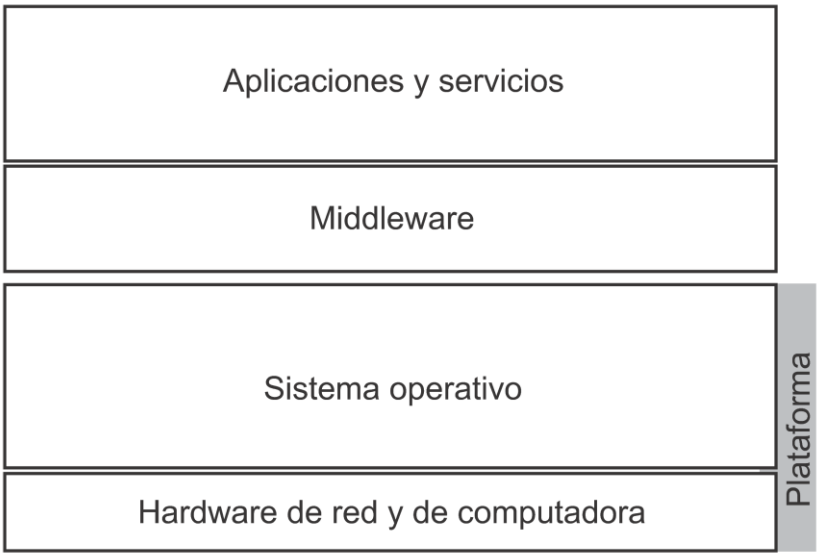

Figura 2.1: Organización por capas en un sistema distribuido.

Esta distribución por capas introduce dos términos importantes: la *plataforma* y el *middleware*. La plataforma consiste de las capas más bajas de hardware y software. Estas capas proveen servicios a las capas superiores que están implementadas independientemente en cada computadora, facilitando la comunicación y coordinación entre procesos.

Un sistema distribuido debería ser relativamente fácil de escalar. Esta característica es una consecuencia directa de tener computadoras independientes. En el sistema distribuido los usuarios y aplicaciones pasan desapercibidas de las partes que son reemplazadas o las que son añadidas (Tanenbaum y Van-Steen, 2007). Con la finalidad de integrar la funcionalidad de distintos tipos de computadoras y redes, los sistemas distribuidos frecuentemente están organizados por medio de una capa de software situada entre la capa de más alto nivel, la de usuarios y aplicaciones, y la capa inferior, consistente de sistemas operativos y de los mecanismos de comunicación. En la figura 2.2 se muestra el lugar donde se ubica la capa middleware en el sistema distribuido (Tanenbaum y Van-Steen, 2007):

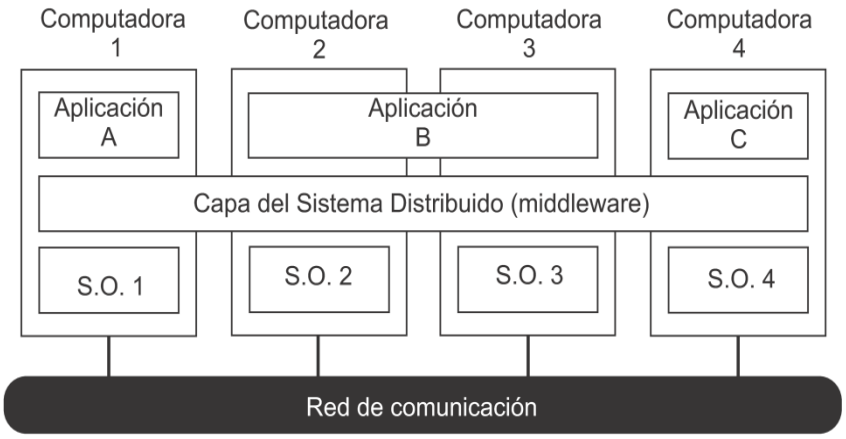

Figura 2.2: Localización de la capa middleware en el sistema distribuido.

Otro aspecto importante a considerar parte de una perspectiva abstracta que describa soluciones a problemas particulares. Coulouris y otros (2012) presentan una clasificación que reúne los requerimientos para lograr características de desempeño y confiabilidad en los procesos y la red de comunicación reunidos en lo que llama el *modelo descriptivo fundamental*:

- *Interacción.* La ejecución de las actividades de los SDs se realiza por medio de procesos que interactúan entre sí por medio del paso de mensajes, lo que resulta en una comunicación coordinada entre procesos. La interacción refleja el impacto en la comunicación debido a la presencia de retardos y la precisión con la que los procesos pueden coordinarse cuando éstos aparecen. Además, existe la dificultad de crear la noción de un tiempo global.

- *Fallas.* La operación correcta o deseada de un SD es amenazada por fallas que ocurren en cualquier computadora y/o en la red. El modelo de falla define y clasifica las fallas que pueden ocurrir en el SD, y provee las bases para el análisis de los efectos que provocan.

- *Seguridad.* La naturaleza modular de elementos de los SDs y la interacción que tienen con sistemas externos son factores que exponen al sistema a ataques de agentes internos o externos. Un modelo de seguridad define y clasifica la forma como estos ataques se producen.

Interacción. Los SDs están compuestos por múltiples procesos y sus interrelaciones son complejas. El comportamiento y estado de los procesos puede ser descrito mediante un *algoritmo distribuido*. Este algoritmo define los pasos a seguir por cada uno de los procesos y la transmisión de mensajes a otros procesos para coordinar su actividad. La tasa de transmisión de cada proceso y el tiempo de la transmisión no es determinístico, por lo que es difícil describir los estados de un algoritmo distribuido y es necesario manejar las fallas de uno o más procesos que intervienen o por fallas en la transmisión. La interacción entre los procesos realiza toda la actividad en el SD. Dos factores afectan la interacción entre los procesos de un SD: la comunicación y la noción global de tiempo.

La comunicación en una red de computadoras tiene las siguientes características de desempeño en relación a la latencia, ancho de banda y *jitter*.

1. El retardo desde el inicio de la transmisión de un mensaje en un proceso y el inicio de su recepción por otro proceso es conocido como *latencia*. La latencia incluye:
   - El tiempo que toma en transmitirse el primer bit de una cadena a través de la red para alcanzar su destino.
   - El retardo de acceso a la red, el cual se incrementa considerablemente cuando hay alta carga de datos.
   - El tiempo que toman los servicios de comunicación del SD para los procesos de envío y recepción.
2. La cantidad total de información que puede transmitir la red en un tiempo dado, se conoce como *ancho de banda*.
3. *Jitter* es la variación en el tiempo en la entrega de una serie de mensajes.

Cada computadora en un SD tiene su propio reloj interno, el cual puede ser usado por los procesos locales para obtener el valor del tiempo actual. Por lo tanto 2 procesos en diferentes computadoras pueden asociar marcas de tiempo (*timestamps*) a sus eventos. Sin embargo, si dos procesos leen sus relojes al mismo tiempo pueden tener lecturas diferentes. Esto se debe al desvío del reloj (*clock drift*), término que se refiere a la tasa con la cual el reloj de una computadora se desvía de un reloj "exacto" de referencia. En un SD es complicado establecer límites en el tiempo que puede tomar la ejecución de los procesos, la entrega de mensaje o el desvío de relojes. Por lo tanto, se establecen dos submodelos para el manejo del tiempo en los SDs (Coulouris y otros, 2012; Hadzilacos y Toueg, 1994):

- *Síncronos*: Es aquel sistema en el que el tiempo de ejecución de cada acción de los procesos tiene cotas superior e inferior y son conocidas, cada mensaje transmitido por la red es recibido en un tiempo acotado que es conocido y cada proceso tiene un reloj local cuya tasa de desvío respecto al tiempo de referencia tiene una cota conocida. El problema inmediato de estos sistemas consiste en garantizar el cumplimiento de las cotas.

- *Asíncronos* En este modelo no existen cotas para la velocidad de ejecución de los procesos, cada acción puede tener una duración arbitraria, existe retardo en la transmisión de mensajes que pueden ser arbitrariamente largos y el desvío de reloj es variable. Este modelo asíncrono no hace suposiciones acerca de los instantes de tiempo involucrados en las ejecuciones.

Actualmente los SDs son asíncronos debido a la necesidad de los procesos y los canales de comunicación de compartir la misma red. Sin embargo, varios problemas de diseño no pueden ser resueltos por un sistema asíncrono, en particular cuando se involucran ciertos aspectos de tiempo. No obstante, es importante resaltar que en realidad los SDs son híbridos, es decir, síncronos en sus procesos internos y asíncronos en la comunicación.

Debido a que los relojes no pueden ser sincronizados perfectamente en un SD, Lamport (1978) propuso un modelo de reloj lógico que puede funcionar para proveer un orden entre los eventos de un proceso en distintas computadoras de un SD.

Modelo de falla. En un SD los canales de comunicación están propensos a fallas, esto es que se alejan de un comportamiento que se considera correcto o deseado. El modelo de fallas descrito por Coulouris y otros (2012) define la forma en la cual una falla puede ocurrir, con la finalidad de comprender los efectos que la falla trae consigo. Hadzilacos y Toueg (1994) proveen una clasificación de fallas en los procesos y en la comunicación, retomada por Coulouris y otros (2012) presentándola como: fallas por omisión, fallas arbitrarias y fallas de tiempo.

- Las *fallas por omisión* se dan cuando las actividades de los procesos o canales de comunicación no se llevan a cabo. La falla más frecuente en los procesos es la *ruptura* (*crash*). Esto significa que el proceso está detenido y no realiza ninguna acción. El método de detección de rupturas se basa en el uso de tiempos fuera (*timeouts*). Este es un método en el cual un proceso permite un periodo de tiempo fijo para que algo ocurra. En sistemas asíncronos un tiempo fuera puede indicar sólo que un proceso no responde. Esto puede ser una ruptura, una ejecución lenta o

un mensaje que no llega. La ruptura de un proceso se llama *falla por paro* (*fail-stop*) si otro proceso puede detectar que el proceso ha colapsado. El comportamiento de falla por paro puede ser producido en un sistema asíncrono si los procesos usan tiempo fuera para detectar cuando otros procesos fallan al responder y con esto garantizar que sean entregados los mensajes. El canal de comunicación produce una falla por omisión si no transporta un mensaje del *buffer* de salida del proceso *p* al proceso *q*. Esto es conocido como fuga de mensajes (*dropping messages*) y es provocado generalmente por la ausencia de espacio en el buffer del receptor o por un error en la transmisión de la red y detectado por la *suma de comprobación* (*checksum*) del mensaje. Hadzilacos y Toueg (1994) se refiere a la pérdida de mensajes de tres formas: falla por omisión en el envío, falla por omisión en la recepción y falla por omisión en la comunicación.

- Las *fallas arbitrarias o bizantinas* describen un tipo de falla que es difícil de tipificar, es decir, que cualquier tipo de error puede ocurrir. Por ejemplo, un proceso puede producir errores en los datos de un programa o puede regresar un valor erróneo al responder alguna invocación. Una falla arbitraria de un proceso es aquella en la cual un proceso puede omitir una acción o ejecuta una acción que no correspondía. Las fallas arbitrarias en los procesos no se pueden detectar si se trata de observar si el proceso responde a las invocaciones porque puede omitir arbitrariamente la réplica. Las fallas arbitrarias en la comunicación surgen cuando el contenido del mensaje está corrupto, mensajes "no existentes" son entregados o mensajes reales son entregados duplicadamente.

- Las *fallas temporales* ocurren en los SDs síncronos donde los límites temporales son establecidos en los tiempos de ejecución de los procesos, tiempo de entrega de los mensajes, y el desvío del reloj. Los sistemas de tiempo real están diseñados para garantizar las limitaciones de tiempo, pero requieren redundancia de hardware.

En general, la exposición de los modelos que describen los elementos que integran los SDs pretende dar un panorama general de la estructura de estos sistemas, con la finalidad de identificar aspectos importantes en su análisis y diseño. En capítulo 2 se retomarán los principios de los modelos de interacción y el modelo de fallas para contextualizar los problemas que se presentan en los sistemas distribuidos, el impacto que tienen en ciertas aplicaciones y los trabajos relacionados que han propuesto estrategias para enfrentarlos.

2.2 Modelo de reconfiguración

Los SDs han sido utilizados en los sectores industriales, de investigación y comercio. Debido a sus características, han sido empleados en aplicaciones críticas y de alta demanda (Almeida, 2001). Por esta razón, se requiere que el SD no tenga interrupciones ni reinicializaciones en ningún momento. En la práctica, el hecho de reconfigurar implica la pausa del sistema para reinstalar componentes y reanudar posteriormente. La *reconfiguración dinámica* evita el paro y permite al sistema cambiar, en tiempo de ejecución, sin que esta modificación tenga un efecto en desarrollo del sistema (Almeida y otros, 2001). El propósito de la reconfiguración dinámica consiste en hacer que un sistema evolucione incrementalmente a partir de una configuración actual a otra configuración (Kramer y Magee, 1985), de manera que este cambio impacte lo menos posible a la ejecución del sistema. Una configuración del sistema es entendida como una estructura de entidades de hardware, software y sus interacciones. Estudios recientes consideran la reconfiguración como un cambio de la

estructura del sistema que conduce a una nueva representación de sus estados (Benítez-Pérez y otros, 2010). En entornos con restricciones de tiempo se requiere la reconfiguración de forma dinámica y que afecte lo menor posible a la ejecución del sistema, si se considera que el retardo de esta transición es manejable.

En las líneas de investigación sobre reconfiguración (Kramer y Magee, 1985; Almeida, 2001) se propone un modelo de reconfiguración que se concentra en definir los cambios y las restricciones a preservar durante el proceso de reconfiguración, en el diseño previo del sistema y en los cambios específicos que se harán durante la reconfiguración. Un modelo de reconfiguración dinámica es propuesto por Almeida (2001), donde se integran la información de reconfiguración, el diseño de actividades de reconfiguración, el manejo de los cambios y la ejecución de la reconfiguración resultante. En la figura 2.3 se muestra la estructura del modelo de reconfiguración dinámica (Almeida, 2001):

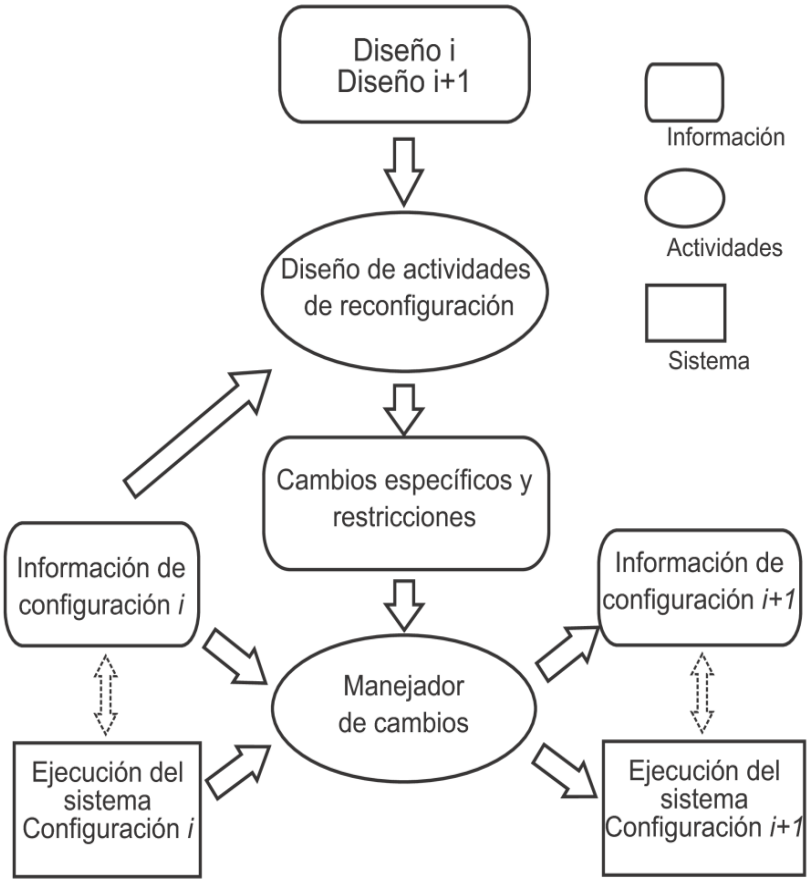

Figura 2.3: Modelo de reconfiguración dinámica.

En este modelo, el *diseño de las actividades de reconfiguración* define los cambios específicos y las restricciones que se deben cumplir durante la reconfiguración. Los cambios son propuestos en términos de *entidades* y *operaciones* sobre estas entidades y son aplicados bajo el control de un *manejador de cambios*. El resultado de estos cambios es la evolución del sistema de una *configuración actual* a otra *configuración resultante*. En este modelo de reconfiguración dinámica, el manejador de cambios usa la *información de reconfiguración* que hace referencia a las relaciones entre entidades. Las restricciones de reconfiguración son predicados impuestos al procesos de reconfiguración que restringen su ejecución; en el

contexto de este trabajo, la reconfiguración estará restringida a la planificabilidad del sistema, es decir, “el
proceso de reconfiguración deberá ser ejecutado antes de cierto plazo”.

El diseño de actividades de reconfiguración es la definición de las acciones que se llevan acabo durante la secuencia de ejecución del sistema. Estas actividades proveen mecanismos para que el sistema sea efectivo ante cambios imprevistos. Este diseño de actividades utiliza la configuración inicial del sistema y las nuevas configuraciones para identificar las modificaciones que se implementarán y producir un conjunto bien definido de cambios a realizar.

El manejador de cambios controla el proceso de reconfiguración del SD. La funcionalidad del manejador de cambios debe garantizar que: los cambios propuestos sean eventualmente aplicados al sistema, las modificaciones hechas deben resultar en un sistema estable y útil, donde las restricciones de reconfiguración sean satisfechas. El proceso de reconfiguración en un sistema en ejecución es intrusivo (Almeida, 2001). La reconfiguración implica la creación, destrucción, migración o reemplazo de entidades, y puede interferir con las interacciones entre entidades. El manejador de cambios debe asegurar que las partes del sistema que interactúan con las entidades bajo reconfiguración no fallen debido al proceso. La *preservación de la consistencia* (Almeida, 2001). es un requerimiento importante de la reconfiguración. Esta preservación se refiere a asegurar que la interacción entre entidades y su coordinación continúen siendo íntegras y funcionales, de manera que el sistema se desempeñe de manera deseada o correcta.

Para (Almeida, 2001). un sistema reconfigurado S_{i+1} es una *evolución incremental correcta* de un sistema S_i, si S_{i+1} preserva el nivel de desempeño (que sea correcto) y si el comportamiento de las entidades afectadas difieren mínimamente del comportamiento previo a la reconfiguración. Sin embargo, esta definición es un tanto restrictiva si se considera la funcionalidad del SD como un todo. Las fallas en los nodos posiblemente no permitan conservar el nivel de desempeño inicial, no obstante la funcionalidad del sistema permanece.

La implementación de este modelo de reconfiguración ha sido realizada en sistemas basados en *object-middleware* Almeida y otros (2001). Estas implementaciones basan la ejecución de los mecanismos de reconfiguración en el manejador de cambios, lo que produce una reconfiguración centralizada. Por otra parte, esta clase de soluciones se ha utilizado poco en entornos en los que se tienen que cumplir plazos temporales, lo cual demanda una reconfiguración acotada en el tiempo. Otra desventaja consiste en que la posible solución requiere información a priori de las configuraciones iniciales (configuración *i*) y posteriores (configuración *i*+1) que dependen totalmente del caso de estudio particular.

2.3 Sistemas en tiempo real

El cómputo en tiempo real es un campo que se ha desarrollado extensivamente en los últimos años. Por esto, cualquier definición sobre tiempo real que se ofrezca podría desestimar ciertos aspectos sobre este concepto. No obstante, se coincide con la definición dada por Gambier (2004):

Un sistema en tiempo real (STR) es aquél en el cual la precisión de un resultado no depende únicamente de la precisión lógica de su cálculo, si no también del tiempo en el cual tal resultado esté disponible.

Esta definición enfatiza la importancia del tiempo como una de las entidades principales del sistema, lo que conduce a la existencia de restricciones temporales asociadas con las actividades que ejecuta el sistema. A nivel de procesos, la diferencia entre el tiempo real y lo que no lo es consiste en que las *tareas* de tiempo real están caracterizadas por un *plazo*, el cual es el máximo tiempo en el cual una tarea debe terminar su ejecución. Dependiendo de las consecuencia que traiga consigo el incumplimiento de un plazo, las tareas de tiempo real se clasifican en (Liu, 2000):

- *Críticas o duras*: Una tarea se dice que es dura si al cumplirse fuera de un plazo provoca consecuencias catastróficas en el sistema.
- *Flexibles o suaves*: Una tarea se dice suave si el incumplimiento de un plazo no compromete al sistema y su consecuencia es mínima.

Un sistema operativo capaz de manejar tareas de tiempo real duro es llamado un *sistema de tiempo real duro*. En general, una aplicación real combina actividades críticas y flexibles, por lo que el sistema operativo en tiempo real está diseñado para manejar ambos tipos de tareas, garantizando el cumplimiento de los plazos de las tareas críticas y minimizando el tiempo promedio de respuesta de las tareas flexibles. Algunas tareas críticas pueden ser: el sensado de datos, detección de condiciones críticas, control a bajo nivel de componentes críticos del sistema, mientras que entre las tareas flexibles están: el intérprete de comandos en una interfaz de usuario, desplegado de mensajes en pantalla, actividades gráficas, etc (Butazzo, 2005; Liu, 2000).

La mayoría de los STR usados para aplicaciones de control están basados en *kernels*. Estos *kernels* son versiones modificadas de sistemas operativos de tiempo compartido, lo que provoca que compartan ciertas características de los sistemas de tiempo compartido que no son adecuados para el manejo de sistemas de tiempo real. En [**¡Error! No se encuentra el origen de la referencia.**] se listan las características de los sistemas de tiempo real: multitarea, planificación basada en prioridades, respuesta a interrupciones externas, mecanismos de comunicación y sincronización, cambio de contexto, manejo del reloj de tiempo real; con el propósito de mostrar que éstas constituyen ciertas limitantes de los sistemas de tiempo real.

En general, se requiere que un STR tenga las siguientes propiedades básicas para ejecutar aplicaciones críticas (Butazzo, 2005):

- *Temporalidad:* Los resultados deben ser correctos no solamente por el valor proporcionado sino en el tiempo requerido. Por ejemplo, los sistemas operativos de tiempo real deben proveer mecanismos para el manejo de tareas y tiempos.
- *Diseñado para carga máxima:* Los sistemas de tiempo real no deben colapsar cuando se presentan condiciones de carga máxima, por lo que deben ser diseñados con sistemas de verificación de retardos para manejar todos los escenarios posibles y liberar carga en el medio de comunicación.
- *Previsibilidad:* Para garantizar un mínimo nivel de funcionamiento, el sistema debe prever las consecuencias de cualquier decisión de planificación. Si no puede

garantizarse la ejecución de una tarea dentro de sus restricciones de tiempo, el sistema debe notificar esta situación y plantear acciones alternativas para hacerle frente a la situación.

- *Tolerante de fallas:* Fallas individuales en hardware y software podrían no causar daño mayor, sin embargo componentes críticos del sistema en tiempo real deben ser diseñados para ser tolerante a las fallas.

- *Mantenibilidad:* La arquitectura de un sistema de tiempo real debe ser tal que asegure que modificaciones en ciertos componentes sean sencillas de realizar.

En particular una de las propiedades más importantes de un sistema de tiempo real duro es la predicibilidad (Stankovic, 1988). Esto es que, basado en las características del *kernel* y en la información asociada a cada tarea, el sistema sea capaz de predecir la evolución de las tareas y garantizar en lo sucesivo que todas las restricciones de tiempo críticas se cumplan. Sin embargo, esto pocas veces es posible dado el alto grado de no determinismo de los SDs. Varios son los factores que provocan incertidumbre en la planificación como lo son las características del procesador y del *kernel*, los algoritmos de planificación utilizados, los mecanismos de sincronización, el manejo de la memoria, la comunicación, los mecanismos para el manejo de interrupciones, entre otros (Butazzo, 2005).

Los sistemas distribuidos cuya funcionalidad está restringida al cumplimiento de plazos temporales se denominan sistemas distribuidos en tiempo real (SDTR) y constituyen el objeto de estudio de este trabajo. Debido a la creciente complejidad en la interacción de los elementos de un SDTR, es necesario realizar un análisis más detallado de todos los aspectos de tiempo real que influyen en las acciones del sistema.

Una aplicación de tiempo real se caracteriza por un conjunto Γ de actividades llamadas ***tareas***, asociadas con ciertas restricciones temporales que deben ser cumplidas para lograr un comportamiento deseado. Es frecuente que las *n* tareas de tiempo real $\tau_0, \tau_1, \tau_2, \ldots, \tau_n$ se ejecuten transcurrido cierto tiempo. De aquí que, una tarea debe entenderse como una secuencia de activaciones[4] idénticas a_i, *i*=1,...,*n* , separadas cada una de ellas por una distancia temporal (Butazzo, 2005).

Los parámetros que caracterizan una tarea de tiempo real son Balbastre (2002); Butazzo (2005):

- *Tiempo de llegada* (r_i): Es el instante en el cual la activación está lista para ejecutarse. También se le refiere como tiempo de liberación.
- *Tiempo de cómputo* (c_i): Es el tiempo que tarda cada activación en ejecutar las acciones que tiene asignadas sin ser interrumpido. Si se trata de un sistema de tiempo real duro, se considerará el peor tiempo de respuesta (WCET: *Worst Case Execution Time*)
- *Plazo de entrega* (d_i): Es el tiempo límite en el cual la activación debe haber terminado. Si la activación acaba en ese instante la activación es planificable. Si todas las activaciones acaban antes el sistema es planificable.
- *Tiempo de inicio* (s_i): Es el tiempo en que la tarea inicia su ejecución.

[4] En distintas referencias, (Balbastre, 2002; Butazzo, 2005; Cheng, 2002; Liu, 2000) se maneja indistintamente este término como instancia, invocación o job.

- *Tiempo de finalización* (f_i): Es el tiempo en que finaliza la tarea, también conocido como tiempo de respuesta. Para que una activación sea planificable se debe cumplir $f_i \leq d_i$.
  - *Desplazamiento*(φ_i): Ocasionalmente, una activación puede tener un desfase o desplazamiento pues no tiene por que comenzar cuando llega el tiempo de inicio. El tiempo de activación de la primera instancia periódica también se le llama *fase*.

Los parámetros de tiempo de una tarea se muestran en la figura 2.4.

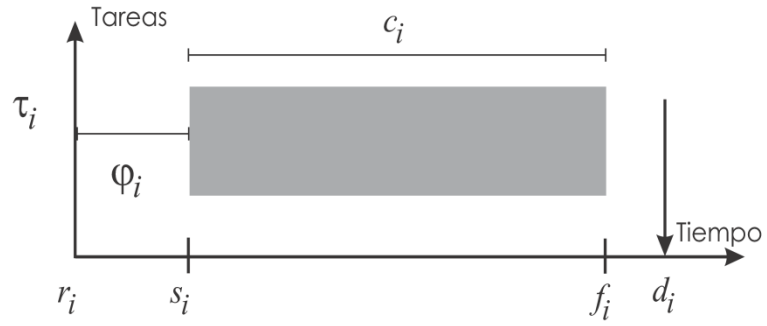

Figura 2.4: Parámetros de tareas.

Butazzo (2005) considera otros parámetros de tiempo adicionales de tareas:

- *Criticalidad*: Parámetro relativo a las consecuencias de no cumplir un plazo.
- *Valor* (v_i): Representa la importancia relativa de la tarea respecto a otras.
- *Tardanza* (l_i): Representa el retardo del cumplimiento de una tarea respecto a su plazo $l_i = f_i - d_i$.
- *Laxitud* (x_i): Es el tiempo máximo que una tarea puede retrasarse en su activación para completar su plazo.

Cabe mencionar que si todas las tareas tiene fase igual a cero $\varphi = 0$, $\forall i$=1,…,*n*; es decir, comienzan en el instante inicial, se dice que el sistema es *síncrono*, en caso contrario se denomina sistema *asíncrono*.

Existen otras clasificaciones para las tareas de tiempo real que se designan en función de la característica que se tomen en cuenta (Balbastre, 2002; Butazzo, 2005):

- En función de la *forma de ejecución*: pueden ser expulsable (*preemptive* o no expulsable. Una tarea expulsable es aquella cuya ejecución puede ser interrumpida por otras tareas y reanudada más tarde. Una tarea no expulsable debe ejecutarse hasta que se complete su tiempo de cómputo, sin interrupción.
- En función de las *consecuencias si se produce una falla*: las tareas pueden ser *críticas* si una falla causa consecuencias catastróficas, o *acríticas* si el fallo de estas tareas no causa un grave perjuicio. Las tareas acríticas suelen dedicarse a funciones auxiliares, como mantenimiento del sistema, monitorización de variables no críticas, etc.

- En función de las *características temporales*:
  - ***Periódicas***: si su ejecución debe repetirse a un intervalo constante de tiempo, llamado periodo (p_i). Una tarea periódica es una secuencia de invocaciones periódicas idénticas activadas a una tasa constante. Se denota una tarea periódica como τ_i. Liu (2000) describe una tarea periódica como una secuencia de activaciones que tienen los mismos parámetros. Esta clase de tareas se caracteriza con los parámetros temporales $(r_i, d_i, p_i, \varphi_i)$. La k-ésima activación de la tarea periódica τ_i, denominada como a_{ik}, se produce en el instante $\varphi_i + (k-1)p_i$, y debe terminar antes del instante $\varphi_i + (k-1)p_i + d_i$.
  - ***Aperiódicas***: si pueden activarse en instantes de tiempo no predecibles. Por ejemplo las tareas aperiódicas son los eventos generados por alarmas. Las tareas aperiódicas, a su vez, tienen tres clasificaciones:
    - *Sin plazo*: no son críticas, pues no tiene ningún plazo límite que cumplir para su finalización, pueden llegar en cualquier instante y sólo se definen por su tiempo de cómputo c_i.
    - *Con plazo firme*: tiene un plazo de ejecución el cual no es crítico y se caracterizan por su tiempo de cómputo y su plazo (c_i, d_i).
    - *Esporádicas*: se produce un falla considerable si no finaliza antes de un plazo y se definen con los mismos parámetros que una tarea periódica (c_i, d_i, p_i), teniendo en cuenta que el periodo indica el tiempo mínimo entre dos activaciones consecutivas.

2.4 Planificación

Referencias básicas sobre planificación, algoritmos de planificación propuestos en años recientes y su correspondiente análisis de planificabilidad son Butazzo (2005); Liu (2000); Cheng (2002), de donde se toman varios aspectos para identificar las consideraciones necesarias en el análisis de planificabilidad de las estrategias de reconfiguración que se plantean más adelante.

Una parte distintiva de los sistemas operativos de tiempo real es el manejo de tareas llamado *planificación*. De acuerdo con [**¡Error! No se encuentra el origen de la referencia.**], para definir el problema de planificabilidad se requiere especificar tres conjuntos: un conjunto de *n* tareas $\Gamma = \{\tau_1, \tau_2, \tau_3, \dots, \tau_4\}$, un conjunto de *m* procesadores $\Pi = \{\pi_1, \pi_2, \pi_3, \dots, \pi_4\}$ y un conjunto de *s* tipos de recursos $\Phi = \{\phi_1, \phi_2, \phi_3, \dots, \phi_4\}$. Menciona que la precedencia entre relaciones de las tareas puede ser especificada a través de un grafo acíclico dirigido, y las restricciones de tiempo pueden ser asociadas con cada tarea. En este contexto, expone, la planificación significa asignar procesadores de Π y recursos de Φ a las tareas de Γ con la finalidad de completar todas las tareas bajo las limitaciones impuestas. Este problema en su forma general es NP-completo y por lo tanto intratable desde la perspectiva del cómputo. En efecto, la complejidad de los algoritmos de planificación es de alta relevancia en los sistemas dinámicos de tiempo real, donde las decisiones de planificación deben ser tomadas en línea durante la ejecución de las tareas.

En particular, cuando un único procesador tiene que ejecutar un conjunto de tareas concurrentes, el procesador tiene que ser asignado a las distintas tareas de acuerdo a un criterio predefinido llamado *política de planificación*. El conjunto de reglas que en cualquier momento determina el orden en que las tareas se ejecutan es conocido como *algoritmo de planificación*. La operación específica de la asignación del procesador a una tarea seleccionada por el algoritmo de planificación se conoce como *despacho*; por lo tanto, una tarea que podría

ejecutarse en el procesador puede estar en ejecución si ha sido seleccionada por el algoritmo de planificación o en espera del procesador si hay otra tarea en ejecución (Liu, 2000).. En la figura 2.5 se muestra la cola de tareas listas en espera de ejecución (Butazzo, 2005).

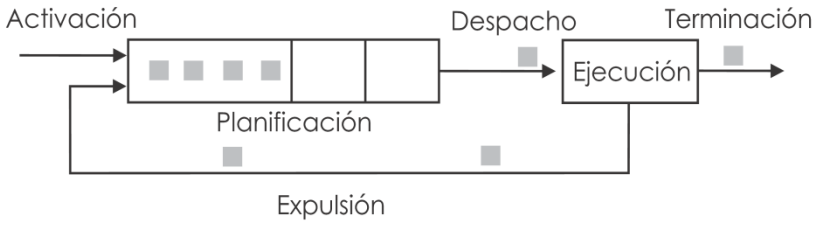

Figura 2.5: Cola de tareas listas en espera de ejecución.

Tanenbaum (1993) y Butazzo (2005) se refieren a una tarea que potencialmente puede ejecutarse en el procesador, independientemente de su disponibilidad, como *tarea activa*. Una tarea en espera del procesador se nombra *tarea disponible*, mientras que la tarea en ejecución se le refiere una *tarea en ejecución****¡Error! Marcador no definido.***. Todas las tareas disponibles en espera del procesador se mantienen en una cola llamada *cola disponible* Los sistemas operativos que manejan diferentes tipos de tareas, puede tener más de una cola disponible. En muchos sistemas operativos que permiten la activación dinámica de tareas, la tarea en ejecución puede ser interrumpida en cualquier momento, de modo que una tarea más importante que llega al sistema inmediatamente puede utilizar el procesador y sin esperar en la cola de disponibilidad. En este caso, la tarea en ejecución se interrumpe y se inserta en la cola de disponibilidad, mientras que la CPU se asigna a la tarea más importante justo a su llegada. La acción de suspender la tarea en ejecución para ser insertada en la cola de disponibilidad se llama expulsión (*preemption*). En la figura 2.6 se muestran los estados en los cuales puede encontrarse un proceso o tarea (Tanenbaum, 1993; Butazzo, 2005).

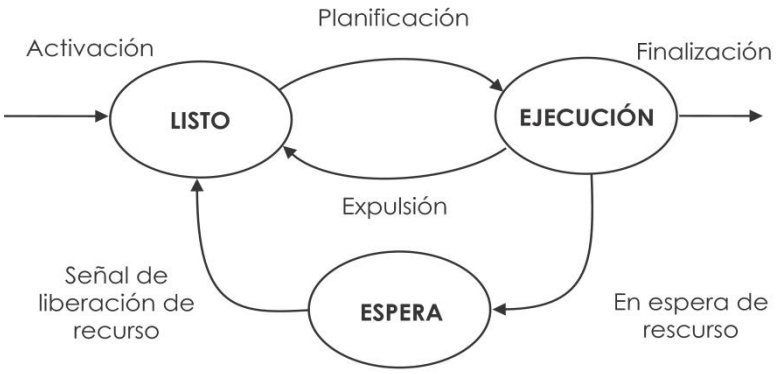

Figura 2.6: Estados de un proceso.

Planificación. Dado un conjunto de tareas $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_4\}$ que son ejecutadas en un único procesador, una *planificación* es una asignación de tareas en el procesador, de modo que cada tarea se ejecuta hasta su finalización. Más formalmente, Butazzo (2005) define una planificación como una función $\sigma = \mathbb{R}^+ \rightarrow \mathbb{N}$ tal que $\forall t \in \mathbb{R}^+, \exists t_1, t_2$ de manera que $t \in [t_1, t_2)$ y. $\forall t' \in [t_1, t_2) \sigma(t) = \sigma(t')$ En otras palabras $\sigma(t)$ es una función escalón entera y $\sigma(t) = k$, con $k > 0$, significa que la tarea τ_k está en ejecución al tiempo t, mientras que $\sigma(t) = 0$ significa que el procesador se mantiene en espera.

En la figura 2.7 se muestra un ejemplo de planificación al ejecutarse un conjunto de tareas $\Gamma = \{\tau_1, \tau_2, \tau_3\}$ en un procesador (Butazzo, 2005):

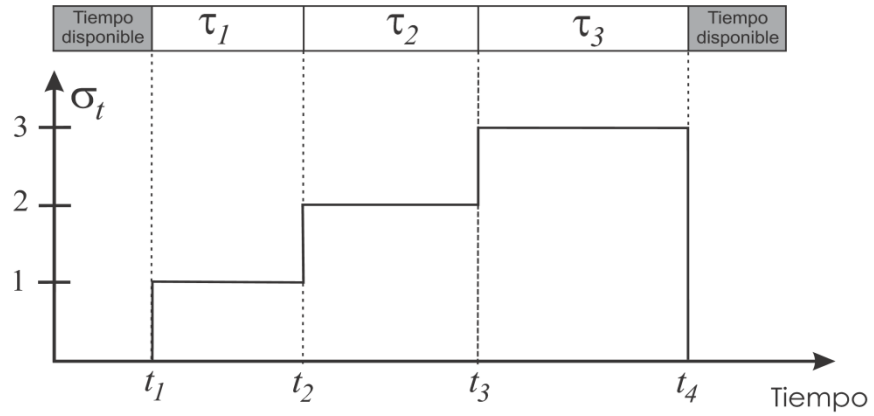

Figura 2.: Planificación de tres tareas τ_1, τ_2, τ_3.

- Para los tiempos t_1, t_2, t_3 y t_4 el procesador realiza un *cambio de contexto*.
- Cada intervalo $[t_i, t_{i+1})$ en el cual σ_t es constante se refiere a una porción de tiempo en la que la tarea τ_i se ejecuta. El intervalo $[x, y)$ identifica todos los valores de t tales que $x \leq t < y$.
- Una planificación expulsable es aquella en la que las tareas en ejecución pueden ser suspendidas arbitrariamente en cualquier momento para asignar la CPU a otra tarea conforme a la política de planificación predefinida. En esta clase de planificación, las tareas pueden ser ejecutadas en intervalos de tiempo disjuntos.
- Una planificación se dice *factible* si todas las tareas pueden ser completadas de acuerdo a un conjunto de restricciones específicas.
- Un conjunto de tareas se dice *planificable* si existe al menos un algoritmo que puede producir una planificación factible.

La planificación expulsable del mismo conjunto de tareas Γ se muestra en la figura 2.8 (Butazzo, 2005).

2.5 Clasificación de los algoritmos de planificación

Liu (2000), Balbastre (2002), Butazzo (2005) y Cheng (2002) reducen la complejidad de la construcción de un planificador simplificando la arquitectura del sistema. Eligen entre un modelo expulsable o no, utilizar prioridades de ejecución preestablecidas, eliminar la prece-

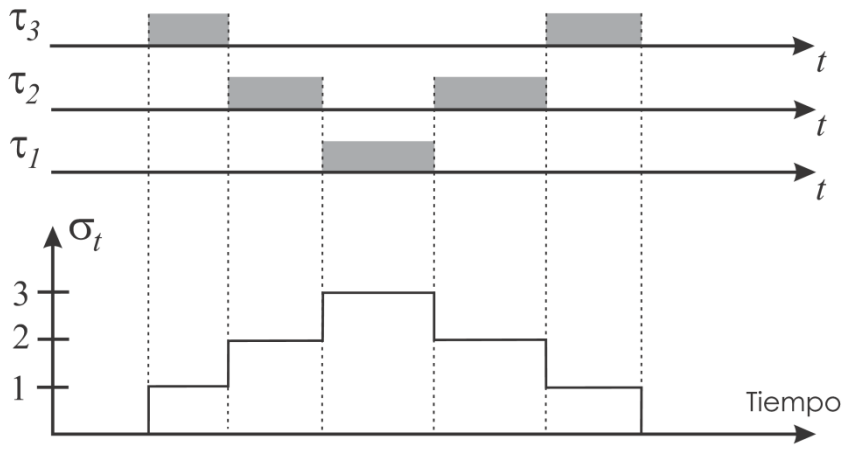

Figura 2.8: Planificación expulsable de tres tareas τ_1, τ_2, τ_3.

dencia y/o las limitantes en los recursos, asumir activación de tareas simultáneas, conjuntos de tareas homogéneas (actividades exclusivamente periódicas o aperiódicas solamente). De aquí que estos elementos son utilizados para clasificar varios tipos de algoritmos de planificación y han sido propuestos en las fuentes citadas.

Entre una gran variedad de algoritmos de planificación de tareas en tiempo real, se encuentran principalmente los siguientes:

- *Expulsables*: Con los algoritmos expulsivos, las tareas en ejecución pueden ser interrumpdas en cualquier momento para asignar el procesador a otra tarea activa, según una política de planificación predefinida.

- *No expulsables*: Con algoritmos no expulsivos una tarea iniciada es ejecutada por el procesador hasta su finalización. En este caso, todos las decisiones de planificación se toman una vez que una tarea finaliza su ejecución.

- *Estáticos*: Algoritmos estáticos son los que basan las decisiones de planificación en parámetros fijos asignados a las tareas antes de su activación.

- *Dinámicos:* Algoritmos dinámicos basan las decisiones de planificación en parámetros dinámicos que pueden cambiar durante la evolución del sistema.

- *Fuera de línea*: Un algoritmo de planificación se utiliza fuera de línea si es ejecutado sobre el conjunto total de tareas antes de la activación de tareas. La planificación generada de esta forma se almacena en una tabla y posteriormente se ejecuta por un despachador.

- *En-línea*: Se dice que un algoritmo de planificación se utiliza en línea si las decisiones de planificación se toman en tiempo de ejecución cada vez que una nueva tarea entra en el sistema o cuando termina una tarea en ejecución.

- *Óptimo*: Un algoritmo se dice que es óptimo si minimiza alguna función de costo definida sobre el conjunto de tareas. Cuando no existe tal función y la única preocupación es lograr una planificación factible, entonces un algoritmo se dice factible (subóptimo) si al utilizar ese algoritmo siempre se encuentra tal planificación.

- *Heurístico*: Un algoritmo se dice que es heurístico si tiende a encontrar una óptima planificación, pero no la garantiza en todos los casos.

Las clasificaciones recientes de los algoritmos de planificación los agrupan principalmente en (Gambier, 2004):

- Un planificador *estático* (*off-line*) requiere que la información completa acerca del problema de planificación (número de tareas, plazos, prioridades, periodos, etc.) sea conocida *a priori* pero con implementación en línea, es decir, la configuración es cambiada en tiempo de ejecución; con ello el problema de planificación puede ser resuelto antes que el planificador sea ejecutado.

- Un planificador es *dinámico* (*on-line*) si en tiempo de ejecución es viable conocer los parámetros de planificación y realizar los cambio de configuración.

Gambier (2004) resalta la ventaja del determinismo de los planificadores estáticos y como desventaja la falta de flexibilidad. Por el contrario, la planificación en línea es muy flexible, pero carece de certidumbre; la planificación en línea puede tener un pobre desempeño si el sistema llega a experimentar sobrecarga.

Puede consultarse Gambier (2004), Butazzo (2005) y Balbastre (2002) la clasificación los planificadores dinámicos. En esta clase de planificadores, existe una familia de planificadores llamados por *prioridades*, en los cuales a cada tarea se le asocia un valor de importancia, de forma que siempre se ejecuta la tarea activa de mayor prioridad. Los planificadores por prioridades pueden subdividirse en: planificadores por *prioridades fijas* que asignan un valor de prioridad inicial a cada tarea, de forma que cada tarea mantiene su prioridad durante toda la ejecución. Si, por el contrario, la prioridad de cada tarea cambia durante la ejecución, entonces se dice que el planificador es por *prioridades dinámicas*.

Dentro de los planificadores por prioridades fijas los más conocidos son el *Rate Monotonic Scheduler* (RMS) (Liu y Layland, 1973) que asigna la mayor prioridad a la tarea con el periodo más corto, y el *Deadline Monotonic Scheduler* (DMS) donde la tarea más prioritaria es la que tiene el menor plazo. Respecto a prioridades dinámicas, los algoritmos más ampliamente estudiados son el *Earlier Deadline First* (EDF) (Liu y Layland, 1973), que prioriza según el plazo absoluto y el *Least Laxity First* (LLF) que asigna la mayor prioridad a la tarea con menor holgura[5].

En la figura 2.9 se presenta un resumen de la clasificación de los algoritmos de planificación desarrollados (Gambier, 2004):

2.6 Planificación de tareas periódicas

En particular, dado que en varias aplicaciones de tiempo real la mayor demanda de actividades la constituyen actividades periódicas, nos enfocaremos en hacer el análisis de planificabilidad de los algoritmos de planificación de tareas periódicas por prioridades más conocidos. El análisis de planificabilidad se realiza para cada algoritmo con la finalidad de derivar una *prueba de garantía* para un conjunto de tareas. De acuerdo con los planteamientos de Butazzo (2005), Liu (2000), Cheng (2002)y Balbastre (2002) es importante, en principio

[5] Una tarea con un tiempo de ejecución pequeño en relación a su plazo, tendrá una mayor holgura.

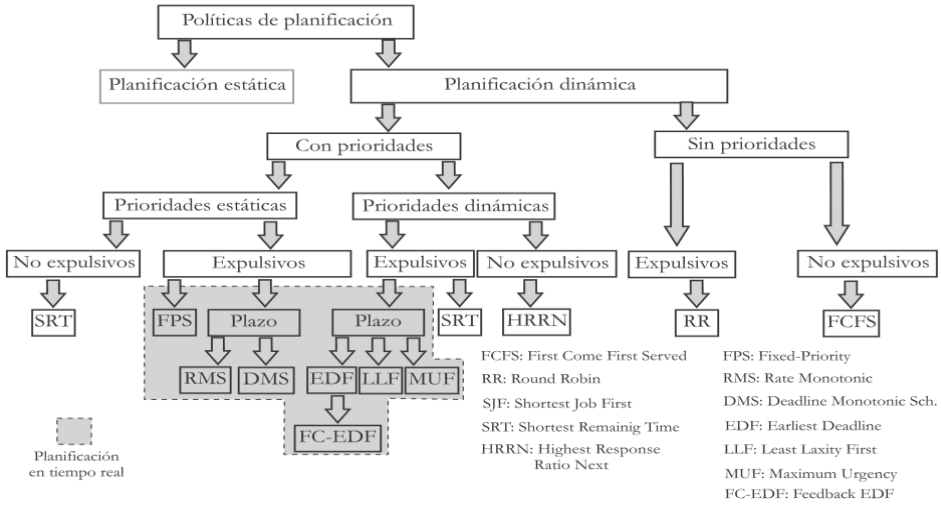

Figura 2.9: Algoritmos de planificación.

definir la notación que se utilizará, y desarrollar el concepto de *factor de utilización*.

La notación para representar los parámetros de tarea utilizada en el análisis de planificabilidad (Butazzo, 2005) se presenta en la tabla 2.1:

Para simplificar el análisis de planificabilidad, se hacen los siguientes supuestos sobre las tareas (Butazzo, 2005; Liu, 2000):

1. Las instancias de una tarea periódica τ_i son activadas con una frecuencia constante. El intervalo $p_i$entre dos activaciones consecutivas es el *periodo* de la tarea.
2. Todas las instancias de una tarea periódica τ_i tienen el mismo peor tiempo de ejecución c_i.
3. Todas las instancias de una tarea periódica τ_i tiene el mismo plazo relativo d_i igual o menor al periodo p_i.
4. Todas las tareas en Γ son independientes, esto es que no hay relaciones de precedencia ni restricciones de recursos.

Así, en los casos en que se den los supuestos anteriores, una tarea periódica τ_i puede ser completamente caracterizada por la fase φ_i, el periodo p_i y el peor tiempo de ejecución c_i y denotada como (Butazzo, 2005):

$$\Gamma = \{\tau_1, (\varphi_i, p_i, c_i), i = 1, \dots, n\}$$

El tiempo de liberación $r_{i,k}$ y el plazo absoluto $D_{i,k}$ de la k−ésima instancia pueden obtenerse haciendo:

$$r_{i,k} = \varphi_i + (k-1)p_i$$
$$D_{i,k} = r_{i,k} + p_i = \varphi_i + kp_i$$

Símbolo	*Denota a*
Γ	Conjunto de tareas periódicas $\tau_0, \tau_1, \dots, \tau_n$.
τ_i	Tarea periódica genérica.
$\tau_{i,j}$	*j*-ésima instancia de la tarea τ_i.
$r_{i,j}$	Tiempo de liberación de la *j*-ésima tarea τ_i.
φ_i	Fase de τ_i correspondiente al tiempo de liberación de la primera instancia $\varphi_i = r_i, 1$.
d_i	Plazo relativo de la tarea τ_i.
D_i	Plazo absoluto de la *j*-ésima instancia de la tarea $\tau_i (D_{i,j} = \varphi_i + (i-1)p_i + d_i)$
$s_{i,j}$	Tiempo de inicio de la *j*-ésima instancia de la tarea τ_i. Es el tiempo en que inicia su ejecución.
$f_{i,j}$	Tiempo de finalización de la *j*-ésima instancia de la tarea τ_i. Es el tiempo en que se completa la ejecución.

Tabla 2.1: Notación de los parámetros del modelo de tareas en tiempo real

En este contexto, se dice que una tarea τ_i es factible si todas sus activaciones terminan antes de sus plazos. Un conjunto de tareas Γ se dice que es planificable (factible) si todas las tareas que comprende el conjunto Γ son factibles.

Hiperperiodo Se define el *hiperperiodo* Λ de un conjunto de tareas periódicas Γ como el intervalo de longitud Λ a partir del cual se repite el mismo orden de activación de las tareas con el que se ejecutó la primera activación de cada una de ellas, siendo este el horizonte de representación (Balbastre, 2002).. El máximo número *n* de activaciones en cada hiperperiodo es igual a

$$\sum_{i=1}^{n} \frac{\Lambda}{p_i}$$

El hiperperiodo es el mínimo común múltiplo de los periodos de todas las tareas de Γ (Liu, 2000).

Factor de utilización. Liu y Layland (1973) propusieron un *factor de utilización* de procesador. Dado un conjunto Γ de *n* tareas periódicas, el factor de utilización del procesador es la fracción de tiempo de procesador gastado en la ejecución del conjunto de tareas Γ. Dado que $\frac{c_i}{p_i}$ es la fracción de tiempo de procesador gastado en la ejecución de la tarea τ_i, el factor de utilización para las *n* está dada por la suma de todas las relaciones de este tipo para cada tarea y acotada como sigue:

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1 \tag{2.1}$$

El factor de utilización del procesador provee una medida de la carga computacional del

CPU debido al conjunto de tareas periódicas. Aunque la utilización del CPU puede ser mejorada al incrementar los tiempos de ejecución o decrementando sus periodos, existe un valor máximo de *U* bajo el cual Γ es planificable y arriba del cual Γ no es planificable. Tal límite depende del conjunto de tareas (de las relaciones particulares entre periodos de tareas) y del algoritmo usado para planificar las tareas.

Sea $U_x(\Gamma, A)$ la cota superior del factor de utilización del procesador para un conjunto de tareas Γ bajo un algoritmo dado *A*. Cuando $U = U_x(\Gamma, A)$, se dice que el conjunto de tareas Γ hace una utilización completa del procesador. En esta situación, Γ es planificable por *A* pero un incremento en el tiempo de cómputo en cualesquiera de las tareas hará el conjunto Γ infactible. Para un algoritmo dado *A*, la mínima cota superior $U_{mx}(A)$ del factor de utilización del procesador es el mínimo de los factores de utilización sobre todos los conjuntos de tareas que utilizan totalmente el procesador:

$$U_{mx}(A) = \min_{\Gamma} U_x(\Gamma, A) \tag{2.2}$$

En la figura 2.10 se ilustra el significado de $U_{mx}(A)$ para algún algoritmo de planificación *A*. Los conjuntos de tareas Γ_i mostrados en la figura difieren del número de tareas y por el valor de sus periodos. Al planificarse por el algoritmo *A*, cada conjunto de tareas Γ_i utiliza totalmente el procesador cuando su factor de utilización U_i (cambiado al modificar los tiempos de cómputo de tarea) alcanza una cota superior particular U_{x_i}. Si $U_i \leq U_x$, entonces Γ_i es planificable, de lo contrario Γ_i no es planificable.

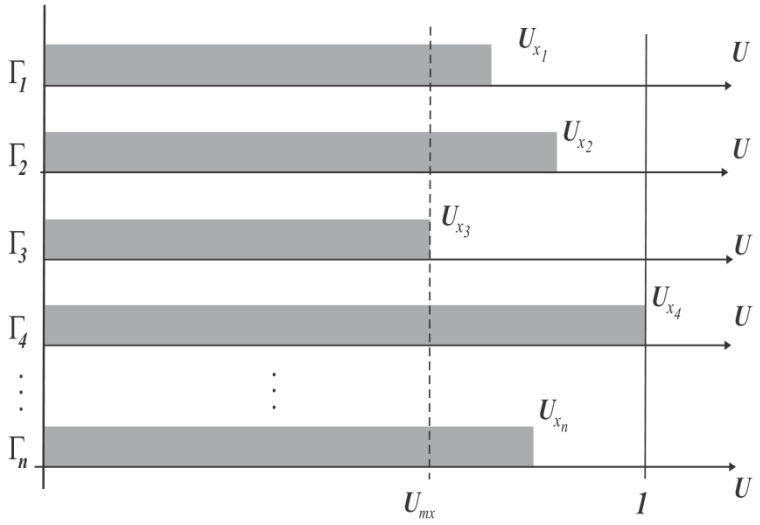

Figura 2.10: Representación de la mínima cota máxima de los factores de utilización.

Butazzo (2005) resalta que cada conjunto de tareas puede tener una diferente cota máxima. Dado que $U_{mx}(A)$ es el mínimo de todas las cotas máximas, cualquier conjunto de tareas que tenga un factor de utilización del procesador abajo de $U_{mx}(A)$ es planificable por *A*.

$U_x(A)$ define una característica importante de los algoritmos de planificación porque permite verificar fácilmente la planificabilidad de un conjunto de tareas. En efecto, cualquier conjunto de tareas cuyo factor de utilización esté por debajo de esta cota es planificable por el

algoritmo *A*. En cambio, la utilización por arriba de esta cota puede conseguirse sólo si los periodos de las tareas están debidamente relacionados (Liu, 2000; Butazzo, 2005).

Si el factor de utilización del procesador de un conjunto de tareas es mayor que uno, el conjunto no puede ser planificable por ningún algoritmo. Es sencillo mostrar lo anterior considerando periodos y tiempos de cómputo estandarizados a unidades de tiempo enteras. Sea P el producto de todos los periodos, $P = p_1 p_2 \ldots p_n$; si $U > 1$ se tiene que $UP > P$ lo cual puede escribe como[6]:

$$\sum_{i=1}^{n} \frac{P}{p_i} c_i > P \tag{2.3}$$

El factor $\frac{P}{p_i}$ representa el número de veces que τ_i es ejecutada en el intervalo *P*, mientras que $\frac{P}{p_i} c_i$es el tiempo de cómputo total requerido por τ_i en el intervalo *P*. Por lo tanto la suma del lado izquierdo de la desigualdad 2.3 representa la demanda total de tiempo de cómputo solicitado por todas las tarea en *P*. Claramente, si la demanda total excede el tiempo disponible del procesador, no hay una planificación factible para el conjunto de tareas (Butazzo, 2005; Cheng, 2002).

2.7 Análisis de planificabilidad

Las tareas periódicas representan la mayor demanda en los sistemas de tiempo real. La necesidad de tareas periódicas surge de la adquisición de datos por sensores, lazos de control y sistemas de monitoreo entre otros. Tales actividades requieren ser ejecutadas cíclicamente a tasas específicas derivadas de los requerimientos de la aplicación. Cuando la aplicación consiste de varias tareas concurrentes con restricciones temporales individuales, el sistema operativo tiene que garantizar que cada instancia periódica sea activada regularmente a su propia tasa y sea completada dentro de su plazo. Para que una activación sea planificable se debe cumplir que el tiempo de finalización de la tarea sea menor o igual que su plazo $f_i \leq d_i$. En general, si todas las activaciones acaban antes o en el plazo, se dice que el sistema de tiempo real es planificable (Butazzo, 2005).

Factibilidad y optimalidad. Como se mencionó en la sección 2.5, una planificación válida es una *planificación factible* si cada tareas completa su plazo (en general si cumple con sus restricciones temporales). Se dice que un conjunto de tareas es planificable de acuerdo a un algoritmo de planificación si al usar el dicho algoritmo siempre se produce una planificación factible (Liu, 2000). El criterio más utilizado para medir el desempeño de un algoritmo de planificación es su habilidad para encontrar una planificación factible de un conjunto de tareas, siempre y cuando tal planificación exista. Un algoritmo es *óptimo* si al usar el algoritmo siempre produce una planificación factible si el conjunto de tareas es factible de planificar. Al contrario, si algoritmo óptimo falla al encontrar una planificación factible se concluye que el conjunto de tareas dado no puede ser planificado por ningún otro algoritmo.
Adicionalmente a los criterios basados en la factibilidad, otras medidas de desempeño incluyen el valor promedio y valor máximo de la finalización de una tarea respecto a su plazo (*lateness*), la tardanza (*tardiness*), el tiempo de respuesta, y las taza de incumplimiento de plazos

[6] tómese el conjunto de periodos $p_1 p_2 \ldots p_n > 0$

(*miss rate*) (Liu, 2000)..

La mayoría de algoritmos de planificación expulsivos son manejados por prioridades. En este tipo de planificación cada tarea tiene asociado un valor de prioridad, o de importancia, de forma que siempre se ejecuta la tarea activa de mayor prioridad. Estos planificadores se subdividen en *planificación de prioridad fija* y *planificación de prioridad dinámica*. Los planificadores de prioridad fija asignan un valor de prioridad inicial a cada tarea, de forma que cada tarea mantiene su prioridad durante toda la ejecución; por el contrario, si la prioridad de cada tarea cambia durante la ejecución, entonces se dice que el planificador es dinámico (Balbastre, 2002).. Para realizar esta planificación se asume que el tiempo que tarda el sistema en hacer el cambio de contexto es despreciable (Xia y Sun, 2008a).

En particular cuatro algoritmos básicos para el manejo de tareas periódicas han sido desarrollados y utilizados extensivamente: *Rate Monotonic* (RM) (Liu y Layland, 1973), *Deadline Monotonic* (DM) (Leung y Whitehead, 1982), *Earliest Deadline First* (EDF) (Liu y Layland, 1973), *Least Laxity First* (LLF) (Mok y Dertouzos, 1978); los dos primeros manejan prioridades fijas, y los dos siguientes son de prioridad dinámica.

En la teoría de planificabilidad se desarrollan dos fases: una primera fase de análisis en la cual se obtiene como respuesta si el conjunto de tareas es planificable o no por el algoritmo elegido, y una segunda fase en la que se planifica el proceso; en caso de ser un algoritmo dinámico la planificación se realiza en tiempo de ejecución Balbastre (2002). Algunas *pruebas de planificabilidad* conocidas para algoritmos de planificación por prioridades, tanto estáticos como dinámicos, son presentadas a continuación.

Planificación de prioridad fija. Un algoritmo de planificación asigna la misma prioridad a todas las instancias de cada tarea. Esto significa que la prioridad de cada tarea se fija con relación a otras tareas. Si varias tareas están listas para ejecutarse al mismo tiempo, la tarea con la mayor prioridad tiene acceso al procesador. En caso que se encuentre lista una tarea con mayor prioridad que la que se encuentra en ejecución, la tarea en ejecución es expulsada por la otra tarea (Xia y Sun, 2008a). El algoritmo de prioridad fija más utilizado es el *Rate Monotonic* (RM), resultado de los trabajos de gran relevancia realizados por Liu y Layland (1973). Este algoritmo asigna prioridades a las tareas basado en sus periodos: el periodo más corto tiene la prioridad más alta. Se ha mostrado que el algoritmo RM es óptimo cuando $d_i = p_i$ para todas las tareas (Butazzo, 2005; Liu, 2000; Cheng, 2002).

El algoritmo de prioridad fija *Deadline Monotonic* (DM) asigna prioridades a las tareas de acuerdo a sus plazos relativos: el plazo relativo más corto tiene la prioridad más alta. Claramente, cuando el plazo relativo de cada tarea es proporcional a su periodo, los algoritmos RM y DM son idénticos. Cuando los plazos relativos son arbitrarios, el algoritmo DM funciona mejor en el sentido que algunas veces puede producir una planificación factible cuando el algoritmo RM falla, mientras que el algoritmo RM siempre falla cuando el algoritmo DM falla (Liu, 2000). Una condición de planificabilidad suficiente para la planificación RM es la siguiente:

Teorema 2.1 *Un sistema que contiene n tareas independientes, expulsivas y periódicas con plazos relativos iguales a sus periodos* $(d_i = p_i)$ *es planificable de a cuerdo al algoritmo RM si la utilización total U satisface qu*e:

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq n\left(2^{\frac{1}{n}} - 1\right) \tag{2.4}$$

Dado que el teorema 2.1 sólo proporciona una condición suficiente, el sistema puede seguir siendo planificable si la utilización del CPU es mayor que la cota superior dada por la ecuación (2.4). En el apéndice A puede consultarse la obtención de la cota de utilización del teorema 2.1.

Lehoczky y otros (1989) muestran que el promedio de utilización real factible para conjuntos grandes de tareas elegidos aleatoriamente es alrededor de 88%. Cuando el número de tareas tiende a infinito, la cota de utilización dada por la ecuación 2.4 se aproxima a $ln(2) \approx 0.693$. Esto conduce al siguiente corolario (Xia y Sun, 2008a):

Corolario 2.2 *Si la tasa de utilización no es mayor que 69.3%, entonces el sistema descrito en el teorema 2.1 es planificable con RM.*:

Pruebas de factibilidad necesarias y suficientes han sido desarrolladas para el algoritmo RM basados principalmente en el cálculo del peor tiempo de respuesta de cada tarea. La siguiente condición necesaria y suficiente se cumple para la planificación de RM (Audsley yotros, 1991; Joseph y Pandya, 1986):

Teorema 2.1 *Un sistema que contiene n tareas independientes, expulsivas y periódicas conplazos relativos iguales a sus periodos* $(d_i = p_i)$ *es planicable con RM si y solo si* $w_i \leq d_i, \forall i \in \{1, \ldots, n\}$, *donde* w_i *es el peor tiempo de respuesta de la tarea i dado por:*

$$w_i = c_i + \sum_{j \in hp(i)} \left\lceil \frac{c_i}{p_i} \right\rceil c_j \tag{2.5}$$

donde $hp(i)$ *es el conjunto de tareas con mayor prioridad que la tarea i y* $\left\lceil \frac{w_i}{p_i} \right\rceil$ *denota la función techo.*

Gracias a la facilidad de su implementación y óptimo desempeño, el algoritmo RM tiene presencia amplia en la mayoría de los sistemas operativos en tiempo real, además que el desarrollo de lenguajes de programación en tiempo real (*Real Time Java*) y estándares (POSIX y UML) han sido influenciados por la teoría alrededor de este algoritmo (véase Xia y Sun (2008a) y las referencias incluidas ahí).

Planificación de prioridad dinámica. En contraste con los algoritmos de planificación de prioridad fija, los algoritmos de planificación de prioridad dinámica asignan diferentes prioridades a las instancias individuales de cada tarea. La prioridad de cada tarea en relación con otras tareas cambia tanto como las instancias son liberadas y completadas. Los algoritmos EDF y LLF son óptimos para tareas expulsivas e independientes Cheng (2002).

El algoritmo de prioridades dinámicas *Earliest-Deadline-First* (EDF) fue presentado también por Liu y Layland (1973). Este algoritmo asigna prioridades a las instancias individuales de las tareas de acuerdo a su plazo absoluto. Una condición de planificabilidad necesaria y suficiente para EDF es:

Teorema 2.4 *Un sistema que contiene n tareas independientes, expulsivas y periódicas conplazos relativos iguales a sus periodos* $(d_i = p_i)$ *es planicable con EDF si y solo s:*

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1 \tag{2.6}$$

Para un conjunto de tareas en el cual algunas tareas tiene plazo relativo d_i menor que su respectivo periodo, no existe una condición necesaria y suficiente de planificabilidad. Sin embargo, una condición suficiente existe para la planificación por el algoritmo EDF de un conjunto de tareas con plazos iguales o menores que sus respectivos periodos. El siguiente teorema generaliza el anterior (Cheng, 2002):

Teorema 2.5 *Un sistema que contiene n tareas independientes, expulsivas y periódicas conplazos relativos iguales a sus periodos* $(d_i \leq p_i)$ *es planicable con EDF si y sólo si:*

$$U = \sum_{i=1}^{n} \frac{c_i}{\min(d_i, p_i)} \leq 1 \tag{2.7}$$

El término $\frac{c_i}{\min\ (d_i, p_i)}$ es la densidad de la tareas i. Cabe decir que si el plazo relativo y el periodo de cada tarea son iguales, entonces el teorema 2.4 es el mismo que el teorema2.5.

Se ha demostrado que EDF es óptimo en relación a otros algoritmos de planificación expulsivos. Esto significa que si existe una planificación factible para un conjunto de tareas, entonces debe ser planificable con EDF (Dertouzos, 1974). Existen varias ventajas con la planificación por EDF como lo es la total utilización del procesador cumpliéndose todos los plazos de tareas. Una desventaja proviene del hecho que es más intuitivo asignar plazos a las tareas que asignar prioridades, ya que asignar prioridades requiere del conocimiento global de todas las demás prioridades, mientras que el asignar plazos requiere sólo del conocimiento local. No obstante que EDF es utilizado en la investigación, carece de uso en la industria, su implementación es más compleja que RM y cuando existe sobrecarga todas las tareas tienden a perder sus plazos (Xia y Sun, 2008a).

Debido que el teorema 2.4 es solo un condición suficiente, un conjunto de tareas que no satisface esta condición puede o no puede ser planificable con EDF. En general, el siguiente teorema es utilizado para determinar si un conjunto de tareas no es planificable por EDF Cheng (2002):

Teorema 2.6 *Dado un sistema que contiene n tareas independientes, expulsivas, periódicas y utilización U. Sea* d_{max} *el máximo plazo relativo entre todos los plazos de tareas,* Λ *el mínimo común múltiplo de todas esas tareas y s(t) la suma de los tiempos de ejecución de las tareas con plazos absolutos menores que t. Este conjunto de tareas no es planicable por EDF si y solo si cualesquiera de las siguientes condiciones se cumplen:*

1. $U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$
2. $\exists t < \min\ \left(\Lambda + d_{max}, \left(\frac{U}{1-U}\right) \max_{1 \leq i \leq n}(p_i - d_i)\right)$

En Krishna (1997) puede encontrarse la prueba del teorema anterior y en Balbastre (2002) se muestra el uso de ciertas funciones de tiempo necesarias para derivar diferentes condiciones necesarias y suficientes para la planificación por EDF, reinterpretaciones del teorema 2.6.

El siguiente algoritmo de planificación dinámica, el *Least-Laxity-First* (LLF) (también es conocido como *Minimum-Laxity-First* o *Least-Slack-Time-First*), considera a $c(i)$ el tiempo de cómputo restante de una tarea al tiempo i. En el tiempo de liberación de una tarea, $c(i)$ es el tiempo de cómputo de esta tarea. Sea $d(i)$ el plazo de una tarea en relación al tiempo actual i, entonces la laxitud (relajamiento) de una tarea al tiempo i es $(d_i - c_i)$. Asi, la laxitud de una tarea es el tiempo máximo que la tareas puede retardar su ejecución sin perder su plazo en el futuro. El algoritmo LLF ejecuta para cada instante la tarea lista con la menor laxitud, en caso que haya más de una tarea con la misma laxitud, LLF elige indistintamente una (Cheng, 2002). La laxitud más pequeña tiene mayor prioridad (Liu, 2000).

2.8 Sistemas multiagente

Desde su creación a mediados de la década de 1970, la inteligencia artificial distribuida (IAD) ha evolucionado y se ha diversificado rápidamente. Hoy en día es un campo de investigación y de aplicación que reúne resultados, conceptos e ideas de muchas otras disciplinas, incluyendo la inteligencia artificial (IA), informática, sociología, economía, organización y gestión de la ciencia y la filosofía, etc. Su carácter multidisciplinario hace difícil de caracterizar con precisión la IAD de manera breve. La siguiente definición ha sido utilizada como punto de partida para explorar este campo y la tomaremos como referencia inicial (Weiss, 1999):

> *La IAD es el estudio, construcción y aplicación de sistemas multiagente, es decir, sistemas en los que interaccionan varias entidades inteligentes que persiguen un conjunto de objetivos o el llevar a cabo un conjunto de tareas.*

De acuerdo con (Weiss, 1999), un *agente* es una entidad física y/o computacional que percibe y actúa sobre su entorno. Es autónomo en el sentido que su comportamiento depende de su propia experiencia (al menos parcialmente). Es una entidad inteligente que opera de forma flexible y racional en una variedad de circunstancias ambientales dada sus capacidades de percepción. La flexibilidad de comportamiento y la racionalidad son alcanzadas por un agente con base en procesos primordiales, tales como la resolución de problemas, planificación, toma de decisiones y aprendizaje. Como entidad que interactúa, un agente puede ser afectado en sus actividades por otros agentes, y quizá también por seres humanos. Un patrón clave de la interacción en sistemas multiagente es la coordinación orientada a objetivos y orientada a tareas. Ambas coordinaciones funcionan en situaciones cooperativas y en situaciones competitivas. En el caso de la cooperación, varios agentes tratan de combinar sus esfuerzos para lograr en grupo lo que cada individuo aislado no puede, y en el caso de la competencia, varios agentes tratan de conseguir lo que sólo alguno de ellos puede tener. El objetivo a largo plazo de la IAD es desarrollar mecanismos y métodos que permiten a los agentes interactuar como los seres humanos y comprender la interacción entre entidades inteligentes, ya sean informáticas, humanas o ambas. Este objetivo plantea una serie de retos, todos en torno a la cuestión elemental de cuándo y cómo interactuar con quién.

Para Bond y Gasser (1988); Huhns y Singh (1998),, dos razones básicas justifican el uso de la IAD y han servido como los impulsos principales detrás del crecimiento de este campo desde hace dos décadas:

1. Los sistemas multiagente tienen la capacidad de jugar un rol primordial en las ciencias de la computación y sus aplicaciones, no solo actualmente sino también a futuro. Las plataformas de computación y ambientes de información actuales están distribuidos, son abiertos y heterogéneos. Los sistemas computacionales ya no son sistemas aislados, están conectados entre sí y con sus usuarios. La creciente complejidad de los sistemas informáticos y de la misma información también se observa en sus aplicaciones. Esto a menudo supera el nivel convencional de la computación centralizada porque se requiere, por ejemplo, el procesamiento de enormes cantidades de datos. Los equipos tienen que actuar más como *individuos* en lugar de sólo *partes*; este enfoque holístico está siendo aplicado actualmente en el modelado de sistemas dinámicos y por el tipo de planificación en tiempo real (Benítez-Pérez y otros, 2009). La IAD proporciona una tecnología que maneja la interacción de alto nivel y las complejas aplicaciones para el cálculo de los modernos sistemas de información.

2. Los sistemas multiagente tienen la capacidad de desempeñar un papel importante en el desarrollo y análisis de modelos y teorías de la interactividad en las sociedades humanas (Jennings, 1994; Wooldridge y Jennings, 1995). Los seres humanos interactúan de diversas maneras y en muchos niveles: por ejemplo la observación y el modelado de unos a otros, al solicitar y ofrecer información, al negociar y discutir, al desarrollar una visión compartida de su entorno, al detectar y resolver conflictos, al formar y disolver la organización de estructuras tales como equipos, comités y economías. Muchos procesos de interacción entre los seres humanos son todavía poco comprensibles a pesar de que son una parte integral de nuestra vida cotidiana. Tecnologías de la IAD nos permiten explorar sus fundamentos sociológicos y psicológicos.

Cenjor y García (2005) mencionan que actualmente se requiere que los sistemas productivos sean más flexibles para realizar distintas acciones que mejoran su capacidad y con ello lograr sus objetivos. De aquí que su complejidad también ha aumentado. Tradicionalmente, esta clase de sistemas utilizaban un control centralizado, lo que se ha modificado hacia el control distribuido. Los sistemas distribuidos, como ya se mencionó, consisten en la división del sistema completo en partes o módulos autogestionados que interaccionan entre sí. Estas interacciones se rigen en función de los elementos en los que se base el control del sistema. Los sistemas de control tradicionales son jerárquicos, se aplican en casos de poca flexibilidad y en secuencias simples de producción; sin embargo, hace unos años se ha dado un giro hacia los sistemas de control distribuido basado en métodos negociados. Estos nuevos sistemas se dividen en distintos componentes que constituyen unidades autónomas de ejecución de estrategias, dotadas de sus propias capacidades de control. Las propiedades principales de estos componentes son la autonomía y cooperación.

Agentes. Con el desarrollo del software se crea un nuevo paradigma, el de los agentes, que rebasa a las metodologías orientadas a objetos. Estas entidades constituyen objetos mejorados en el sentido que los objetos son pasivos y no activan un comportamiento, los agentes son capaces de establecer relaciones con otros agentes, relaciones no previstas a la hora del diseño de dicho agente y son capaces de adaptar al entorno todas las correspondencias de carácter racional (Cenjor y García, 2005). Las relaciones entre agentes no siguen una estructura jerárquica: su relación es heterárquica, es decir, incluyen en sus

interacciones con otros agentes procesos de negociación ya que no existen restricciones de niveles, negocian entre ellos y en función de las necesidades del sistema (García y Cenjor, 2007). Los agentes son más adaptables y flexibles que los sistemas jerárquicos. Las estructuras heterárquicas representan el reto de mayor relevancia en el diseño de sistemas multiagente. En dichas estructuras las organizaciones jerárquicas son eliminadas o al menos están restringidas. La cooperación y competencia entre agentes deben reemplazarse por la supervisión y las órdenes de rango (García y Cenjor, 2007).

Existen varias clasificaciones de los agentes que se postulan en función del comportamiento de cada uno de los agentes (García y Cenjor, 2007).

- *Agentes reactivos*: Reciben información del entorno mediante sensores, y una vez que analizan la información recibida, actúan en consecuencia para garantizar el logro de los objetivos. Al ejecutar las acciones, el agente sólo necesita tomar repetidas lecturas del sensor y compararlas con las condiciones de ese instante para ejecutar las acciones pertinentes. Si más de una condición del estado del sistema coincide con las lecturas se da paso a la reacción. Cada condición está asociada con una lista de condiciones de menor peso a las que inhibe; por tanto, una acción solo se ejecuta si no existen otras condiciones que inhiban la condición que la produce.

- *Agentes deliberativos*: Se consideran agentes de carácter proactivo. Son agentes que trabajan por medio de la deducción lógica y que asumen el estado del entorno mediante una base de datos de fórmulas y predicados lógicos. La toma de decisiones se modela por medio de un conjunto de reglas de deducción. El problema de estos agentes es que la toma de decisiones está basada en la suposición de entornos estáticos. Para resolver esto, Wooldridge y Jennings (1995) propusieron el concepto de agente deliberativo BDI (por beliefs-desires-intentions), tienen una carga de creencias, deseos e intenciones propias.

- *Agentes híbridos*: Son agentes que combinan la reactividad y la proactividad. Un agente híbrido es autónomo, reactivo, proactivo, además es capaz de establecer comportamientos sociales. La desventaja consiste en que resulta difícil diseñar estos agentes de tal forma que se coordinen las distintas categorías para un funcionamiento coherente, esto es que resulta complicado determinar hasta qué punto tienen un carácter reactivo y a partir de cuál es proactivo.

Sistemas multiagente. Actualmente, se intenta construir conjuntos de entidades autónomas e inteligentes que cooperan para desarrollar un trabajo, cuya comunicación es posible por medio de mecanismos basados en el envío y recepción de mensajes. La tendencia es dar solución a problemas como cooperación entre agentes para cumplir un objetivo: la generación de subtareas y la asignación de éstas a un grupo de agentes que sea capaz de ejecutarlas, así como el desarrollo de lenguajes y algoritmos que funcionen en forma paralela (García y otros, 2010).. Hoy en día, los desarrollos basados en varios agentes han tenido dos vertientes: los agentes robóticos y los agentes computacionales. Esto conduce a fuertes distinciones en las características dinámicas y cinemáticas inherentes a los agentes robóticos y los agentes computacionales que deben ser tratadas por separado para definir las técnicas de coordinación y cooperación en cada caso (García y otros, 2010).

Los requerimientos que se exigen de un sistema multiagente (SMA) trabajando en entornos reales son los siguientes (García y otros, 2010):

- Adquirir conductas de acuerdo con las circunstancias.
- Tomar decisiones adecuadas ante la ocurrencia de sucesos no previstos en el entorno.
- Desempeñar tareas con eficiencia y en tiempo real.
- Tener conciencia de la existencia de otros agentes en el entorno.

No obstante que las arquitecturas multiagentes han dado buenos resultados en la consecución de los objetivos de control, cabe mencionar que estas arquitecturas basadas en la negociación entre sistemas independientes tienen el inconveniente de depender excesivamente de la calidad de la información y del medio por el cual se comunican.

Modelo de interacción multiagente. Para manejar la complejidad de los sistemas, se propone utilizar estructuras y técnicas que descompongan el problema en problemas más pequeños, que de igual manera puedan ser divididos en otros más pequeños todavía. De tal forma, la solución del problema es el conjunto de pequeñas soluciones producto de varias descomposiciones. Los sistemas orientados a agentes constituyen una alternativa para diseñar sistemas distribuidos de control. Para esto se requiere descomponer el problema en varias entidades autónomas que puedan actuar e interactuar de forma flexible para lograr su objetivos, que tengan control de ellas mismas y sobre sus propias acciones (Ramírez-González y otros, 2007). Los sistemas basados en agentes proporcionan una solución descentralizada basada en vistas parciales y locales que provienen del entorno dinámico y proveen al sistema de un alto grado de flexibilidad además de dotarlo de robustez. Los agentes son entidades identificables con objetivos definidos, situados en un entorno particular.

De acuerdo con Corkill (2003) un *sistema multiagente colaborativo* se basa en la división del sistema en entidades pequeñas que encapsulan los detalles individuales y la complejidad asociada a cada una, de manera que los cálculos requeridos por las funciones involucradas en cada entidad se logran por medio de la colaboración entre ellas. Las entidades que conforman el sistema cuentan con una vista parcial que cubre el ámbito en el que se desempeña con base en su función primordial. No obstante el sistema reacciona de la forma deseada si todas las acciones son coordinadas. En los sistemas distribuidos, la complejidad de los problemas a resolver exige puntos de vista locales, dado que son muy amplios para ser analizados como un todo. Soluciones basadas en vistas parciales con frecuencia ofrecen soluciones flexibles y prácticas. Esta forma de pensar en forma local constituye una alternativa prometedora para resolver problemas complejos a gran escala.

Corkill (2003) propone un SMA que utiliza una memoria distribuida (MD) accesible a todos los agentes para trabajar de forma distribuida. Esta propuesta la utiliza Ramírez (2006) con fines de planificación de un SDTR. Distintas representaciones de este sistema multiagente se derivan de acuerdo al tipo de manejo de esta memoria.

Ramírez (2006) menciona que una forma de combinar un conjunto de diversos módulos de software es conectarlos de acuerdo a los requerimientos de flujo de datos. Esta forma de comunicación es una elección correcta cuando el conjunto de entidades, así como las comunicaciones entre ellos son estáticas. En caso que la interacción entre módulos sea dinámica y no se determine el momento en que cada módulo aparece o desaparece, entonces es necesario utilizar protocolos privados de comunicación. En este último caso se utiliza la *arquitectura de blackboard* (BB).

La arquitectura de BB consta de tres componentes (Ramíez-González y otros, 2007):

- La *fuente de conocimiento* (FC) que funciona como una dupla condición-acción.
- El *blackboard* que es un repositorio de datos global y contiene datos de entrada, posibles soluciones, soluciones parciales y soluciones finales. Cada modificación del blackboard puede provocar nuevas fuentes de conocimiento.
- El *controlador* que realiza la selección de la fuente de conocimiento que va a interactuar y que conduce a la solución.

En el BB se depositan todas las posibles alternativas para llegar a una solución. Estas alternativas son seleccionadas por medio de las competencias que cada agente y de una *fuente de conocimiento* (Corkill, 2003) donde se almacena información con el fin de encontrar las mejores soluciones.

Con base en un SMA que utiliza una arquitectura de BB se puede disponer de distintos tipos de comunicación (Corkill, 2003), de acuerdo a la disposición del BB. Un caso particular consiste en que la comunicación entre agentes se hace por medio del envío de mensajes de un agente a todos los demás o a un subconjunto de ellos, y la recepción de mensajes de todos o sólo de alguno de ellos. Cada agente dispone de una MD y una FC.

Se pueden obtener distintos tipos de interacción entre agentes de acuerdo a la disposición de la FC, el BB y el controlador; lo cual flexibiliza la implementación del sistema multiagente en cada caso de estudio particular.

2.9 Tolerancia a fallas

Un sistema se dice que falla si interrumpe la ejecución de la función que tiene como propósito. Una falla puede ser el paro total de una función o la ejecución de una función de forma anormal. Dubrova (2008) define la *tolerancia a fallas* como la habilidad de un sistema para continuar desempeñando sus funciones no obstante la presencia de alguna *falla*. El objetivo del diseño tolerante a fallas es minimizar la probabilidad de aparición de una falla. A grandes rasgos, la tolerancia a fallas se asocia a la confiabilidad, la operación exitosa y la ausencia de interrupciones. Un sistema tolerante a fallas debería de ser capaz de manejar fallas en componentes individuales de hardware o de software, fallas de energía u otro tipo de fallas inesperadas que afecten su funcionamiento. La tolerancia a fallas es necesaria ya que en la práctica es imposible construir un sistema perfecto. El problema fundamental es que al aumentar la complejidad de los sistemas, la confiabilidad se deteriora drásticamente, a menos que se tomen medidas que compensen los efectos de las fallas. Otro problema consiste en que es inevitable que factores ambientales inesperados no sean tomados en cuenta, o errores provocados por los usuarios no sea previstos, esto conduce a que diversas fallas sean provocadas por situaciones no contempladas en el diseño del sistema.

Existen varias propuestas para lograr la tolerancia a fallas. Un enfoque muy común es el uso de *redundancia* (Dubrova, 2008; Oosterom, 2005; Benítez-Pérez y otros, 2007b).

La redundancia consiste en proveer funciones adicionales al sistema que en condiciones de operación sin falla son innecesarias de utilizar. Estas adiciones pueden ser la replicación de componentes de hardware, la verificación de algún bit adjunto a una cadena de datos, algunas líneas de código que verifican la exactitud de los resultados del programa. El propósito de incorporar redundancia es elevar la confiabilidad del sistema. Existen dos tipos de redundancia posible (Dubrova, 2008):: redundancia espacial y redundancia temporal. La primera provee componentes adicionales, funciones o datos; conforme al componente que se agrega la redundancia espacial se clasifica en redundancia de hardware, software o de información. La redundancia temporal consiste en la repetición de cálculos o de transmisión de datos y el resultado es comparado con una copia almacenada de resultados previos.

Originalmente, la tolerancia a fallas fue usada para hacer frente a defectos físicos en los componentes de hardware. Los diseñadores empleaban estructuras redundantes con escrutinio (voto) (Latif-Shabgahi, 2004) para eliminar el efecto de los componentes defectuosos, detección de errores, o corrección de códigos para detectar o corregir errores de información, técnicas de diagnóstico para localizar componentes averiados. La necesidad de tolerancia a fallas persiste, ya que los sistemas llevan a cabo tareas críticas en cuanto a seguridad. Actualmente, el desarrollo de aplicaciones de cómputo en tiempo real, particularmente la construcción de software embebido en dispositivos inteligentes, requiere de software tolerante a fallas. En lugar de implementar nuevas funcionalidades de hardware, se diseñan nuevos conjuntos de instrucciones para ejecutar las acciones deseadas. Si se requieren cambios en la funcionalidad, las instrucciones son modificadas en lugar de construir diferentes dispositivos físicos.

El fin principal de la tolerancia a fallas es lograr un sistema sea fiable (*dependable system*). La necesidad de un sistema que garantice buenos resultados es primordial, por ejemplo, en aeroplanos, misiones espaciales, plantas químicas y nucleares, o en implantes biomédicos como los marcapasos.

Atributos de sistemas fiables. Dubrova (2008) describe los atributos de que deben tener los sistemas que garantizan resultados fiables: confiabilidad, disponibilidad y seguridad.

- *Confiabilidad* La confiabilidad $R(t)$ de un sistema en el tiempo t es la probabilidad que el sistema opere sin falla en un intervalo $[0,t]$, dado un correcto desempeño el sistema en el tiempo 0. La confiabilidad es una medida de la continuidad en la entrega de un servicio correcto.

- *Disponibilidad* La disponibilidad de un sistema se refiere a la fracción de tiempo en la cual el sistema se mantiene en operación sin interrupción o mantenimiento. La disponibilidad $A(t)$ de un sistema al tiempo t es la probabilidad que el sistema funcione correctamente en el instante de tiempo t. Frecuentemente es necesario determinar el intervalo de disponibilidad, definido como:

$$A(T) = \frac{1}{T}\int_0^T A(t)dt$$

$A(T)$ es el promedio de disponibilidad en un punto dentro de un intervalo de tiempo T. Este intervalo podría ser el tiempo de vida del sistema o el tiempo en el que se completa alguna tarea particular.

- *Seguridad* La seguridad es una extensión de la confiabilidad en el sentido de fallas que pueden provocar riesgos graves. En condiciones de seguridad, las fallas tienen niveles de riesgo: fallas seguras, fallas inseguras. La seguridad de un sistema $S(t)$ es la probabilidad que el sistema continúe su operación de forma correcta o suspenda su operación cuando se presenta una falla segura (con poco riesgo).

Impedimentos para la fiabilidad. Dubrova (2008) desarrolla los impedimentos para que un sistema continúe realizando sus funciones correctamente y hace una distinción entre los conceptos: desperfecto (*fault*), error y falla (*fail*).

- Desperfecto: Se refiere a un defecto físico o avería que ocurre en algún componente de hardware o software, por ejemplo cortos circuitos o ejecución incoherente de los programas.

- Error: Es la desviación de un cálculo certero o correcto, que se presenta debido a un desperfecto. Los errores son asociados usualmente a valores incorrectos en los estados del sistema, por ejemplo la obtención de valores imprecisos o datos alterados durante la transmisión.

- Falla: Es la ausencia de ejecución de alguna acción que se espera o se requiere en el sistema. Se dice que un sistema tiene una falla si el nivel de servicio que se entrega se desvía del nivel acordado en la especificación del sistema durante un periodo de tiempo particular.

Es clara la diferenciación que se hace de los tres conceptos anteriores y la manera de ubicarlos durante la ejecución del sistema. En este trabajo se maneja en general el término *falla* para caracterizar tanto los defectos físicos como la ejecución inapropiada o deteriorada del sistema.

Origen de las fallas. La procedencia de las fallas tiene distintas fuentes: especificación e implementación incorrectas, defectos de fabricación, factores externos (Dubrova, 2008):

- La *especificación incorrecta* se produce por algoritmos, arquitecturas, o requerimientos incorrectos. Un ejemplo típico es el caso cuando la especificación de requerimientos ignora aspectos del medio ambiente en el cual el sistema opera.

- La *implementación incorrecta* está relacionada con los errores en el diseño que ocurren cuando la implementación del sistema no cumple con las especificaciones precisas. En el hardware, esto incluye mala selección de componentes, errores lógicos, o mala sincronización. En software, se relaciona con los *bugs* en el código del programa o una errónea reutilización de software.

- Los *defectos en componentes* incluyen imperfecciones en la manufactura del hardware, defectos aleatorios en los dispositivos y el desgaste de componentes.

- Los *factores externos* provienen fuera de los límites del sistema, generados por el ambiente y por el usuario u operador del sistema. Los factores externos incluyen fenómenos que afectan directamente la operación del sistema, tales como

temperatura, vibración, descargas eléctricas, radiación electromagnética. Fallas provocadas por los usuarios pueden considerarse como involuntarias o maliciosas.

Tipificación de fallas. Para (Dubrova, 2008) las fallas en los sistemas pueden ser tipificadas agrupándolas en las siguientes categorías: fallas en común, fallas de hardware (Alag y otros, 2001) y fallas de software.

1. *Fallas en común.* Son aquellas fallas que ocurren simultáneamente en dos o más componentes redundantes. Estas fallas son provocadas por fenómenos que crean dependencia entre las unidades de redundancia las cuales generan una falla simultáneamente entre los dispositivos, por ejemplo los buses de comunicación compartido. Otra causa puede adjudicarse al diseño erróneo de copias redundantes de hardware o software que fallan en circunstancias idénticas.

2. *Fallas de hardware.* Estas fallas se pueden agrupar en fallas permanentes, transitorias e intermitentes.
   - *Fallas permanentes:* Son aquellas fallas que permanecen hasta que una acción correctiva se lleva acabo. Estas fallas son usualmente provocadas por algún defecto físico en el hardware, tales como cortos circuitos, ruptura de interconexiones, etc.
   - *Fallas transitorias:* Son fallas que permanecen activas durante un periodo de tiempo corto. Dada su corta duración, estas fallas son detectadas frecuentemente a través de errores que resultan de su propagación. Las fallas transitorias son predominantes en memorias de computadora. La causa de las fallas transitorias son, en la mayoría de los casos, debida a factores ambientales como descargas electrostáticas o campos electromagnéticos. Por ejemplo, un pico de carga de voltaje puede provocar que un sensor dé una lectura errónea durante unos cuantos milisegundos antes de proveer la lectura correcta.
   - *Fallas intermitentes:* Son fallas que transitorias que tienen una aparición repetitiva.

3. *Fallas de software.* Las fallas de software y hardware difieren en varios aspectos. Primeramente el software no se desgasta o deforma, ni se rompe o es afectado por los factores ambientales como los dispositivos físicos. Asumiendo que el software es determinista, es decir, se supone que funciona de la misma forma en las mismas circunstancias a menos que haya problemas en el hardware que modifique ciertos aspectos como almacenamiento de datos o direcciones lógicas. Debido a que el software no cambia desde que se carga y comienza su ejecución, la tolerancia a fallas no es efectiva por replicación de módulos porque cada copia tendría fallas idénticas. En segundo lugar, el software es mejorado constantemente durante la vida del sistema. Esto aumenta la confiabilidad y seguridad. Por otra parte, la corrección de errores en el código no necesariamente garantiza mayor confiabilidad, dado que pueden generarse problemas inesperados adicionales producto de estas correcciones. Finalmente, el software es difícil de verificar dada su complejidad. Pruebas y métodos de depuración tradicionales son poco efectivos para sistemas de software grandes. Debido a esta verificación incompleta en la mayoría del software, la mayoría de las fallas de software son fallas de diseño provocadas por malas interpretaciones de las especificaciones por parte de los programadores.

Medios para lograr sistemas confiables. La tolerancia a fallas se combina con otros métodos para lograr un sistema que responda efectivamente ante escenarios con falla. Estos

métodos adicionales son la prevención de fallas, supresión de fallas y predicción de fallas (Dubrova, 2008).

Con el propósito de asegurar un sistema confiable, distintas propuestas se enfocan en la *detección e identificación de fallas* (FDI por *Fault Detection and Identification*) (Heredia y Ollero, 2009) y en la *adaptación y aislamiento de fallas* (FAI por *Fault Accommodation and Isolation*) (Cork y otros, 2005; Cork y Walker, 2007).

Como se ha mencionado, el propósito de la tolerancia a fallas se centra en asegurar el funcionamiento correcto de un sistema ante la presencia de fallas. Uno de los medios para lograr lo anterior es la redundancia de componentes, esta estrategia permite *enmascarar* la falla (Karimi y otros, 2010) o *detectar* la falla.

El *enmascaramiento de falla* es el proceso de asegurar que únicamente los valores correctos sean manejados por el sistema cuando existe una falla. Esto se lleva a cabo evitando que el sistema sea afectado por el error, por medio de la corrección inmediata del error o compensando el error de alguna manera. Un ejemplo de enmascaramiento de falla es la triple redundancia modular con voto mayoritario (Latif-Shabgahi y otros, 2003).

La *detección de falla* es el proceso de determinar que una falla ha ocurrido en el sistema. Las técnicas de detección de falla son las *pruebas de aceptación* y la *comparación*. Las pruebas de aceptación consisten en evaluar el resultado de un proceso y permitir su continuidad si el resultado fue satisfactorio. La comparación es utilizada por sistemas con componentes redundantes, una diferencia en los resultados indica la presencia de una falla.

Por otra parte, *la prevención de falla* se logra por medio de técnicas de control de calidad durante la especificación, implementación y fabricación de los dispositivos o elaboración del software. En cuanto a la *supresión de falla*, esta se realiza durante la ejecución del sistema y en la fase de vida de operación del sistema. Durante la ejecución, se realiza la verificación, diagnóstico y corrección, miestras que en la vida de operación del sistema se realizan mantenimientos correctivos y preventivos.

Finalmente, la *predicción de falla* se realiza por medio de la evaluación del comportamiento del sistema respecto a las ocurrencias de falla. En general, esta evaluación consiste en conocer el grado de falla del sistema en términos de probabilidades.

2.10 Resumen

En este capítulo se ha dado una descripción de los conceptos fundamentales en los cuales está enmarcado este trabajo. En primer lugar, se ofreció un panorama general sobre los modelos que caracterizan un sistema distribuido enfatizando la arquitectura y los problemas que surgen al utilizar esta clase de sistemas. En segundo término se estableció el proceso de reconfiguración como el cambio o transición en un sistema que tiene como propósito reaccionar ante un escenario que perturbe el correcto funcionamiento del mismo. El esquema de reconfiguración mostrado se tomó como base para elaborar una estrategia de reconfiguración considerando las restricciones temporales. Se hizo una revisión de los conceptos de tiempo real, planificación y de los algoritmos de planificación de tareas

periódicas, fundamentales para enmarcar la reconfiguración tal que se garantice la planificabilidad de un sistema. Debido a la distribución de dispositivos, se explicó la necesidad de una reconfiguración no centralizada, y la utilización de dispositivos inteligentes. Se mencionó que el enfoque multiagente es prometedor para elaborar estrategias de reconfiguración distribuida y que uno de los propósitos de la reconfiguración es lograr un sistema distribuido en tiempo real confiable y tolerante a fallas.

Capítulo 3

Sistemas de Control en Red

La mayoría de los sistemas utilizados en la industria o investigación contienen más de un procesador y pueden ser multiprocesador o distribuidos. Los problemas que se abordan en los sistemas distribuidos incluyen, entre otros, la planificación del medio de comunicación y la tolerancia a fallas. En este capítulo se presenta una tipo particular de sistema distribuido en tiempo real denominado sistema de control en red. Esta clase de sistemas han sido estudiados ampliamente desde la década pasada, dado que ha aumentado su uso por las ventajas de tener un bajo costo, mantenimiento sencillo y por ser muy flexibles respecto a actualizaciones y cambios en los dispositivos. Resultado del análisis y diseño de los sistemas de control en red, se han identificado las principales problemáticas que presentan y se han propuesto distintas estrategias para manejar la presencia de retardos en el ciclo de control, incertidumbres en el proceso y la optimización del medio de comunicación que, por ser compartido, es demandado por todos los nodos que desean transmitir. Estos factores impactan directamente en la calidad del servicio y la calidad del control del sistema. Los sistemas de control en red, los problemas que enfrenta y las propuestas de solución recientes son presentados a continuación con el propósito de delimitar los objetivos de las estrategias de reconfiguración.

3.1 Descripción general de un sistema de control en red

Cuando un sistema de control retroalimentado tradicional el lazo está cerrado por medio de un canal de comunicación, el cual puede ser compartido por otros nodos fuera del sistema de control, entonces se dice que el sistema de control es un *Sistema de control en Red* (NCS por *Networked Control System*) (Gupta y Mo-Yuen, 2010). Una característica fundamental de estos sistemas es que la información para realizar el control es intercambiada entre los componentes del sistema de control usando una red compartida. De forma general, la investigación de los NCS se puede clasificar en (Gupta y Mo-Yuen, 2010):

- *Control de la red:* Estudia e investiga la comunicación y el tipo de redes adecuadas para los NCS en tiempo real, es decir, por ejemplo los problemas de control de ruteo, reducción de congestión, comunicación eficiente de los datos, protocolos de red, etc.
- *Control sobre la red:* Estudia y diseña estrategias de control y sistemas de control que funcionen mediante la red, con la finalidad de minimizar el efecto de parámetros adversos (retardos, pérdida de datos, *jitter*) sobre el rendimiento del NCS.

Para Baillieul y Antsaklis (2007), la investigación de los NCS radica principalmente en la intersección de tres áreas de investigación: sistemas de control, redes de comunicación e información y ciencias de la computación. Los NCS pueden beneficiarse de los desarrollos teóricos en la teoría de la información y la ciencia de la computación. Las principales dificultades en la fusión de los resultados de estos campos de estudio han sido la diferencias en el énfasis que se le da a cada investigación. En el campo de la información, los retardos no son motivo de interés, ya que es más importante transmitir con exactitud. En contraste, en los sistemas de control los retrasos sí son de importancia primordial. Los retrasos son mucho más importantes que la exactitud de la información transmitida, por el hecho que los sistemas de control retroalimentados son menos sensibles a tales inexactitudes. De manera similar, en la investigación tradicional en la ciencia de la computación, el tiempo no ha sido un problema central debido a que típicamente las computadoras interactuaban con otras computadoras o un operador humano,no con el mundo físico. Recientemente, áreas como el Tiempo Real han comenzado a abordar las cuestiones de las limitantes de tiempo duro donde los sistemas de cómputo deben reaccionar dentro cotas de tiempo específicas, lo cual es esencial para los sistemas de procesamiento integrados que tratan directamente con el mundo físico. Por esta razón, este trabajo estudia los NCS subrayando las limitantes temporales y de comunicación, con el propósito de identificar cómo estas impactan al resultado de control. En principio, se describe la estructura básica de un NCS.

Arquitectura de un NCS. Los componentes principales de un NCS son: el proceso físico por controlar, un sensor que contiene un convertidor analógico-digital (A/D), un controlador, un actuador que contiene un convertidor digital-analógico (A/D) y una red de comunicación. Un diagrama esquemático de un lazo cerrado de control en red típico se muestra en la figura 3.1 (Xia y Sun, 2008a):

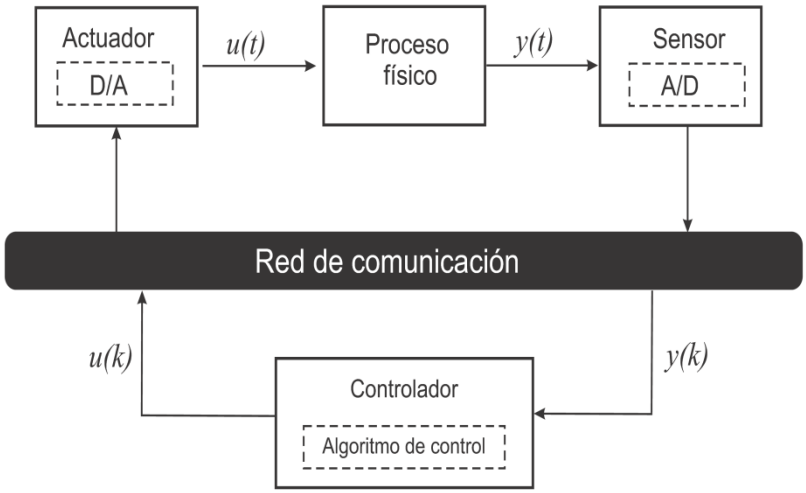

Figura 3.1: Diagrama de un lazo de control en red.

El sistema a controlar es generalmente un proceso físico en tiempo continuo, por ejemplo, un motor de corriente continua (motores DC), un péndulo invertido, etc. Las entradas $u(t)$ y salidas $y(t)$ del proceso físico son señales de tiempo continuo. El convertidor A/D transforma los resultados del proceso en señales digitales cada instante de muestreo. El tiempo entre dos instantes de muestreo consecutivos es el periodo de muestreo. El convertidor A/D, puede ser una unidad separada o integrada en el sensor, transforma las entradas continuas a discretas.

El controlador se encarga de la ejecución de programas de software que procesan la secuencia de datos de la muestra de acuerdo con algunos algoritmos de control específicos para producir la secuencia de comandos de control (Xia y Sun, 2008a,b). Para que estas señales digitales sean aplicables a los procesos físicos, el convertidor D/A las transforma en señales de tiempo continuo con la ayuda de un circuito de retención que sostiene la entrada al proceso hasta que un nuevo comando de control esté disponible desde el controlador. El método más común de retención es de orden cero(Halevi y Ray, 1988a), que mantiene la entrada constante durante el periodo de muestreo[1].

En un entorno distribuido, las secuencias de datos de la muestra y los comandos de control deben ser transmitidos desde el sensor hasta el controlador y del controlador al actuador, respectivamente, a través de la red de comunicaciones (Halevi y Ray, 1988b). El tipo de red puede ser cableada (*fieldbus* e internet) o bien inalámbricas (WLAN, ZigBee, Bluetooth). En general, todos los bloques del sistema pueden ser activados por el mismo método: tiempo o evento; no obstante, es frecuente que en varios sistemas de tiempo real el sensor sea activado por el tiempo, mientras que el controlador, el actuador y la comunicación de red se activen por evento (Cervin y Eker, 2000).

Métodos de diseño. Los sistemas de control basados en el muestreo de datos son una clase de sistemas híbridos que contienen tanto las señales de tiempo continuo, como las de tiempo discreto. Existen dos enfoques diferentes para diseño de NCS basados en el muestreo los datos: en tiempo discreto y discretización de tiempo continuo Xia y Sun (2008a); Arzen yotros (1999, 2000); Astrom y Witternmark (1997); Wittenmark y otros (2002).

El diseño en *tiempo discreto* considera el proceso desde el punto de vista de la computadora, donde sólo son considerados los valores de las entradas y salidas del sistema a cada instante de muestreo. Al tratar el proceso como un sistema de tiempo discreto, el controlador digital puede ser diseñado directamente en el dominio de tiempo discreto. Con este método, el proceso físico sigue siendo un sistema de tiempo continuo, y cualquiera otra información sobre lo que sucede entre los instantes de muestreo se ignora. Para el diseño del controlador, en primer lugar, debe derivarse una versión del modelo del sistema con base en el muestreo de datos, donde se asume que se utiliza muestreo periódico y retenedor de orden cero.

Para un sistema de tiempo continuo descrito por el modelo en espacio de estados invariante en el tiempo (Xia y Sun, 2008a):

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{3.1}$$

Al resolver la ecuación del sistema 3.1 se obtiene el estado del sistema en el intervalo $kh \leq t < kh + h$ de la siguiente manera (Xia y Sun, 2008a):

[1] Otro tipo de dispositivo de retencion es el retenedor de primer orden donde la entrada se calcula con base a la salida actual y previa del controlador usando un polinomio de primer orden.

$$
\begin{aligned}
x(t) &= e^{A(t-kh)}x(kh) + \int_{kh}^{t} e^{A(t-s')}Bu(s')ds' \\
&= e^{A(t-kh)}x(kh) + \int_{kh}^{t} e^{A(t-s')}ds'Bu(kh) \\
&= e^{A(t-kh)}x(kh) + \int_{kh}^{t-kh} e^{As}dsBu(kh) \\
&= \Phi(t,kh)x(kh) + \Gamma(t,kh)u(kh)
\end{aligned}
$$

(3.2)

La ecuación 3.2 describe el cambio de estado durante el periodo de muestreo. Al considerar el proceso en el siguiente momento de muestreo, puede derivarse la siguiente descripción del sistema en tiempo discreto:

$$x(kh+h) = \Phi x(kh) + \Gamma u(kh)$$
$$y(kh) = Cx(kh) + Du(kh)$$

donde

$$\Phi = e^{Ah}$$
$$\Gamma = \int_{0}^{h} e^{As}dsB$$

Utilizando técnicas similares, es posible derivar la versión del modelo del sistema en tiempo continuo del sistema válido para el muestreo aperiódico (Xia y Sun, 2008a).

3.2 Retardos de tiempo

Xia y Sun (2008a) mencionan que en el lazo de control de la figura 3.1 las operaciones de conversión A/D, incluyendo la ejecución de los programas de control, la conversión D/A y transmisión de datos, requieren necesariamente cierta cantidad de tiempo para llevarse a cabo. Cuando los recursos computacionales y de comunicación son limitados (tiempo de procesamiento y ancho de banda) debe haber un retardo que considerar desde la lectura del sensor hasta la actuación de la ley de control (Peng y otros, 2009; Xia y Sun, 2008b). Es importante resaltar que las propiedades de los parámetros de tiempo asociados con lazos de control en el mundo real pueden variar debido a múltiples factores tales como la distribución de los recursos y las variaciones de la carga de trabajo, que afectan a su vez la calidad del control (QoC) de los sistemas de control.

En la figura 3.2 se muestra una instancia de ejecución del ciclo de control en diagrama de tiempo (Xia y Sun, 2008a).

En un NCS pueden darse los siguientes fenómenos (Xia y Sun, 2008a):

- El retardo provocado por el tiempo que requiere cada elemento del ciclo de control.
- El retardo puede ser variable en el tiempo.
- El retardo puede ser superior al periodo de muestreo .

- Los datos se pueden perder.

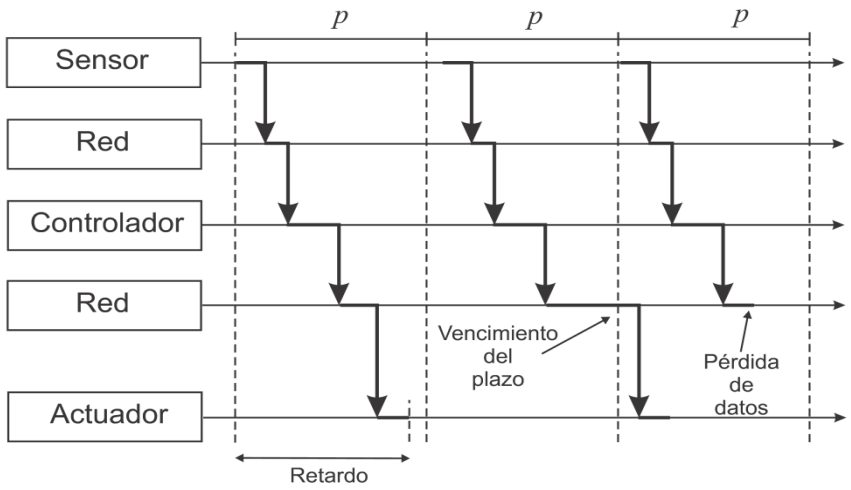

Figura 3.2: Diagrama de tiempo del lazo de control.

Nilsson (1998) menciona que en el lazo de control distribuido los actuadores y sensores están conectados a una red de comunicación. Estas entidades reciben y envían información del controlador. El controlador está conectado a la red, y se comunica con los sensores y actuadores por el envío de mensajes a través de la red. El envío de mensajes por la red comúnmente lleva un tiempo. Dependiendo de la política de planificación en el sistema, este tiempo de transferencia puede tener diferentes características. El tiempo de transferencia puede, en algunas configuraciones, ser casi constante, pero en muchos casos éste varía de manera aleatoria. Por lo tanto, la magnitud del retardo de transferencia depende de la carga de la red, de otras comunicaciones de entrada y perturbaciones eléctricas, etc. Dependiendo de cómo los nodos sensor, actuador y controlador estén sincronizados, se pueden obtener varias configuraciones. Algunas configuraciones requieren que los nodos sean *activados por evento* o *manejados por reloj*. La activación por evento indica que un nodo comienza su actividad dada la ocurrencia de un suceso, por ejemplo, cuando un nodo recibe información de otro nodo a través de la red. Activación por reloj significa que los nodos comienzan su actividad en un tiempo específico, por lo que la actividad de los nodos puede ejecutarse periódicamente.

Existen esencialmente tres tipos de retardos en el sistema:

1. Retardo de comunicación entre el sensor y el controlador δ_k^{sc}
2. Retardo computacional en el controlador δ_k^{c}
3. Retardo de comunicación entre el controlador y el actuador δ_k^{ca}

el subíndice *k* indica una posible dependencia del tiempo para el retardo. En la figura 3.3 se muestran la localización de los retardos de comunicación y control en el ciclo de control retroalimentado (Nilsson, 1998):

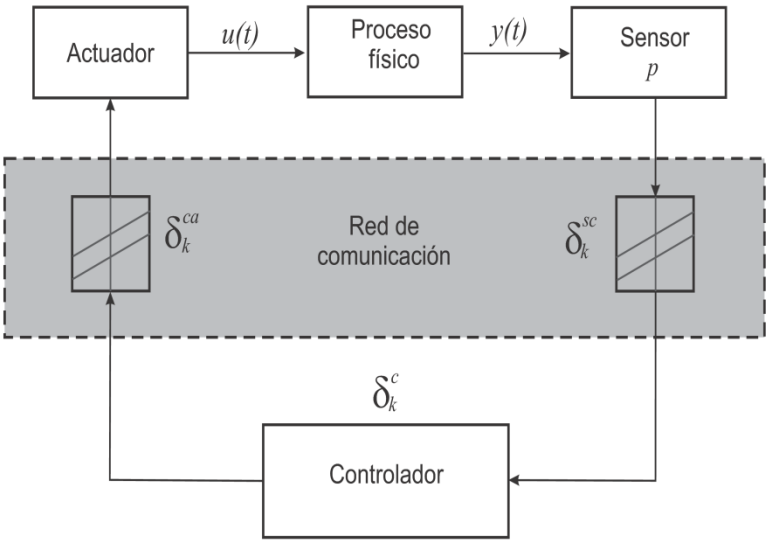

Figura 3.3: Localización de los retardos en el ciclo de control de un NCS.

El retardo general en el ciclo de control δ_k para el NCS es el tiempo desde la lectura de la señal hasta su aplicación por el actuador y está dado por la expresión:

$$\delta_k = \delta_k^{sc} + \delta_k^{c} + \delta_k^{ca}$$

Un problema importante a tratar es la variabilidad de los retardos, pues esta hace que el sistema sea variante en el tiempo, y que los resultados teóricos de análisis y diseño de sistemas invariantes en el tiempo no puedan ser utilizados directamente.

Para Nilsson (1998), desde el punto de vista del control con muestreo de datos, es natural la lectura de la salida del proceso físico cada lapso de tiempo equidistante *p*. También es natural mantener el retardo de control tan corto como sea posible. La razón es que el retraso de tiempo conduce a un retraso en la fase, lo cual generalmente degrada la estabilidad y desempeño del sistema (Nilsson, 1998). Lo anterior sugiere una configuración con el controlador y actuadores activados por evento, lo cual significa que el cálculo de la nueva señal de control, respectivamente la conversión D/A de la nueva señal de control se realiza tan pronto como llegue la nueva información del nodo sensor y del nodo controlador respectivamente. En la figura 3.4 se muestra el diagrama de tiempos del proceso previamente descrito (Nilsson, 1998), Un inconveniente de esta configuración es que el sistema se hace variable en el tiempo, esto se ve en la figura 3.4 en donde la salida del proceso cambia a tiempos irregulares.

Efecto del retardo. La red introduce niveles no confiables (no determinísticos) de servicio (QoS) en términos de los retardos, pérdida de datos y jitter. En los NCS que funcionan en entornos de tiempo real, si el retraso de tiempo excede el límite especificado de tiempo tolerable, la planta o el dispositivo puede ser dañado o degradarse el rendimiento. Como se muestra en la figura 3.5 (Gupta y Mo-Yuen, 2010),, en los sistemas de Tiempo Real duro, la función de utilidad η de inmediato tiende a cero tan pronto como el plazo t_d para que la tarea o conjunto de tareas sea cumplido, por lo tanto, la tarea debe ser completada antes del plazo. En sistemas de Tiempo Real suave la función de utilidad se degrada gradualmente a η_{min}.

Modelado de retardo. Nilsson (1998) describe la forma en que los retardos pueden ser modelados. Menciona que para analizar los sistemas de control con retardos en la red se

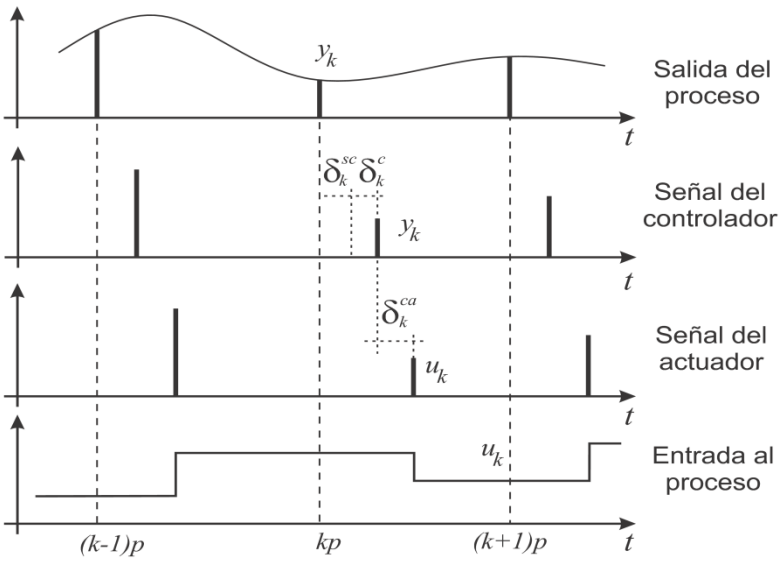

Figura 3.4: Diagrama de tiempo del proceso de control en un NCS.

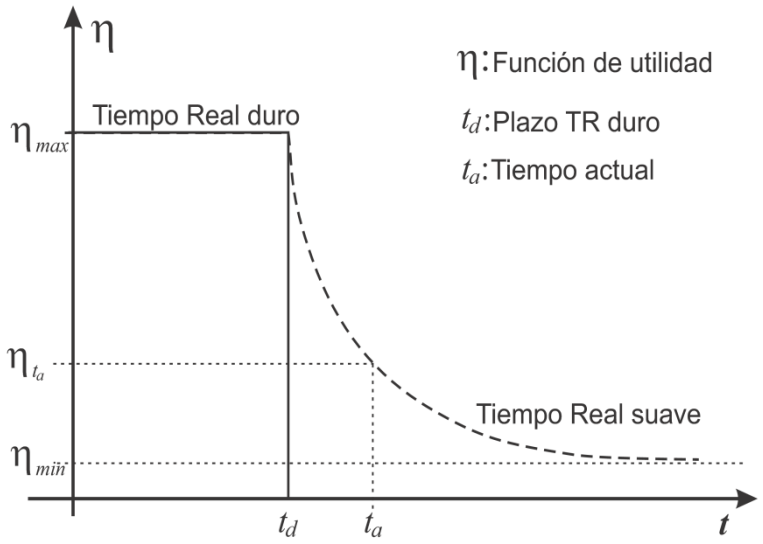

Figura 3.5: Función de utilidad para NCS en Tiempo Real.

requiere modelar el ciclo de control considerando el que el retardo varía debido a los cambios de carga de la red, las políticas de planificación en la red y en los nodos y debido a fallas de la red. Se enfoca principalmente en el modelado de los retardos desde la perspectiva de retardos probabilísticos regidos por procesos markovianos. La descripción de los tres tipos de modelado de retardos es la siguiente (Nilsson, 1998):

1. *Retardos constantes.* El modelo más simple es el de retardo constante para todas las transferencias de datos en la red de comunicación. Este modelo puede ser utilizado incluso si la red tiene retardos variables, por ejemplo, si el tiempo de los procesos es mucho más largo que el retardo introducido por la comunicación. En este caso el valor

medio o el peor caso de retardo puede ser usado en el análisis. Una manera de lograr retardos contantes es introduciendo buffers de tiempo después de cada transferencia. Haciendo estos buffers más largos que el peor caso de retardo, el tiempo de transferencia puede ser visto como constante. Este método fue propuesto en (Luck y Ray, 1990). Una desventaja de este método es que el retardo de control llega a crecer más de lo necesario.

2. *Retardo aleatorio independiente de una transferencia a otra* Los retardos de la red comúnmente son aleatorios. El retardo en la red puede ocasionarse debido a varias causas (Nilsson, 1998):

    - Espera a que la red esté desocupada.
    - Si muchos mensajes están pendientes, la espera puede incluir la transmisión de mensajes de espera.
    - Pueden ocurrir colisiones en algunas redes si dos nodos tratan de enviar al mismo tiempo y la solución puede incluir un tiempo de espera aleatorio para evitar otra colisión la próxima vez.

    Como las actividades en el sistema usualmente no están sincronizadas unas con otras, el número de las causas de retardo anteriores es aleatorio. Para tomar en cuenta la aleatoriedad de los retardos de la red en el modelo, los retardos en el tiempo pueden ser modelados tomándolos de una distribución probabilística. Para mantener un sistema sencillo para el análisis, se asume el retardo de transferencia independiente de los retardos anteriores. En un sistema de comunicación real, el tiempo de transferencia estará, sin embargo, correlacionado con el último retardo de transferencia. Por ejemplo, la carga de la red, el cual es uno de los factores que afecta el retardo, está típicamente variando con una constante de tiempo más lenta que el periodo de tiempo de muestra en un sistema de control, por ejemplo el tiempo entre dos transferencias (Nilsson, 1998; Méndez-Monroy y Benítez-Pérez, 2011b)

3. *Retardo aleatorio con distribución de probabilidad regidas por una cadena de Markov.* Para modelar el fenómeno de colas en la red, y las variaciones de carga en la red, se utiliza un modelo de red con memoria o estado. Una manera de modelar las dependencias entre las muestras es dejar la distribución de los retardos de la red sean regidos por el estado de una cadena subyacente de Markov. Efectos tales como la variación de la carga de red pueden ser modelados haciendo una transición en la cadena de Markov cada vez que una transferencia es hecha en la red de comunicación.

La caracterización de los retardos (Halevi y Ray, 1988a,b; Lian y otros, 2001c; Zhang y Yu, 2008) forma parte importante de los esfuerzos en el análisis (Lian y otros, 2001a) y diseño (Lian y otros, 2002) de los NCSs, lo que ha resultado en el diseño de estrategias de control reconfigurable (Benítez-Pérez y García-Nocetti, 2005; Benítez-Pérez y otros, 2009, 2010), planificación (Sename y otros, 2003), y análisis de estabilidad (Yu y otros, 2005) de los NCSs. No obstante, el modelado y control de los retardos sobrepasa los objetivos de este trabajo; más aún, se desea que las estrategias de reconfiguración que se plantean en este trabajo eviten o disminuyan la presencia de retardos inducidos por la red.

3.3 Comunicación en tiempo real

En los NCS, la red se convierte en el recurso compartido por varias entidades de comunicación. Por lo tanto, el manejo de los recursos de comunicación es de especial interés.

Los mensajes son intercambiados entre los nodos. Comúnmente, estos mensajes están fragmentados en segmentos para su transmisión a través de la red. Cada segmento es manejado por la red como una unidad básica de transmisión llamada paquete y la transmisión de paquetes no es expulsiva. La transmisión de paquetes por la red es análoga a la ejecución de tareas en los procesadores; así como las tareas, los mensajes pueden ser periódicos, aperiódicos o esporádicos. Un mensaje también tiene un tiempo de transmisión correspondiente al peor caso y un plazo relativo. La transmisión de un mensaje puede simplemente verse como una tarea, con el tiempo de transmisión como el tiempo de ejecución.

Típicamente las redes de comunicación son diseñadas con base en el protocolo OSI (*Open System Interconnection*) de siete capas (Tanenbaum, 1995). Es responsabilidad del protocolo garantizar la comunicación en tiempo real. Problemas como el manejo de recursos son abordados en diferentes niveles en el protocolo de capas y combinan varias estrategias. Desde el punto de vista de la planificación, la red es un recurso compartido por el que compiten distintos mensajes. El rol crucial de coordinar el acceso de los nodos activos recae en los protocolos de control de acceso al medio (MAC por *Medium Access Control*) Xia y Sun (2008a).

Dependiendo de como les es permitido a los nodos acceder al medio, dos categorías de protocolos MAC son identificados (Kumar y otros, 2006):

1. *Acceso controlado*: El tiempo y el orden de cada nodo que accede al medio son controlados por algoritmos específicos, por tanto, no hay colisiones. Son llamados protocolos MAC libres de contenciones. Un ejemplo de estos protocolos es el TDMA (*Time Division Multiplex Access*) el cual es comparable con la planificación fuera de línea de los procesadores. Varios fieldbus como Profibus, P-NET, IEC 1158, WorldFIP, Interbus, FF, etc. pertenecen a esta categoría.

2. *Acesso aleatorio*: Cada nodo puede solicitar acceso al medio en un tiempo arbitrario, de manera aleatoria. Esto conduce a colisiones entre diferentes nodos. Estos protocolos MAC se basan en contenciones. El protocolo de Acceso Múltiple con escucha de portador (CSMA por *Carrier Sense Multiple Access*) y sus variantes son ejemplos típicos.

A continuación se describen de manera general los protocolos MAC con base en la exposición de Xia y Sun (2008a) y Lian y otros (2001b).

TDMA. En TDMA hay una estación base que coordina los nodos conectados a la red. El tiempo sobre la red es dividido en segmentos de tiempo de tamaño fijo. Cada nodo tiene asignado un cierto número de segmentos en los cuales puede enviar y recibir mensajes. De esta manera, a múltiples nodos se les permite acceder a una red compartida sin colisiones. TDMA es determinista, lo cual es idóneo para aplicaciones de tiempo real. Sin embargo, debido a esta inflexibilidad, TDMA no se usa en aplicaciones como Internet con tráfico indefinido.

CSMA/CD. Significa CSMA con detección de colisiones. Consiste en un conjunto de reglas para determinar cómo los nodos de la red responden cuando existen colisiones. El

término *Carrier sense* se utiliza para referirse al hecho que varios nodos escuchan a la red antes de intentar transmitir. Si la red está ocupada, esperan hasta que la red esté disponible. *Múltiple acceso* significa que más de un nodo puede sensar (escuchando y en espera de transmitir) a cada instante. *Detección de colisiones* significa que cuando múltiples nodos transmiten accidentalmente al mismo tiempo pueden detectar la colisión.

CSMA/CD es un estándar en IEEE 802.3 e ISO 8802.3. Las redes con el estándar Ethernet usan CSMA/CD para resolver los conflictos de contención en el medio de comunicación. En Ethernet, cualquier nodo puede intentar enviar un paquete en cualquier momento. Si la red está disponible, entonces el nodo que envía empieza a transmitir. Mientras transmite, escucha para detectar si hay una colisión. Si la colisión ocurre, cada nodo participante detiene la transmisión y espera un intervalo de tiempo aleatorio para retransmitir. Este tiempo aleatorio es determinado por el algoritmo estándar *binary exponential backoff* (BEB): el tiempo de retransmisión es elegido aleatoriamente entre 0 y $(2^i - 1)$ *lapsos de tiempo*, donde *i* denota la *i*-ésima colisión detectada por el nodo y un lapso de tiempo se refiere al mínimo tiempo necesario para la transmisión de ida y vuelta. Después de 10 colisiones detectadas, el intervalo es fijado al máximo de 1023 lapsos de tiempo. Un paquete será descartado después de 16 colisiones.

La ventaja principal del CSMA/CD consiste en que es sencillo de implementar, lo que ha ayudado a hacer de Ethernet el estándar para las redes de área local. Sin embargo, CSMA/CD es un protocolo no determinista y no soporta ninguna priorización de mensajes. Las colisiones son el mayor problema en redes con alta carga debido a que el retardo puede no ser acotado.

CSMA/CA. Significa CSMA con evasión de colisiones. Este protocolo es otra modificación del CSMA, donde se evita la colisión más que la detección y es usado para mejorar el desempeño de CSMA. Se diseñó para reducir la probabilidad de colisiones entre los múltiples nodos que acceden al medio en el punto donde la colisiones suelen ocurrir. Este es el significado propiamente de evasión colisiones. A diferencia de CSMA/CD que trata una colisión después de haber sido detectada, CSMA/CA actúa para evitar colisiones antes de que sucedan. CSMA/CA es usado principalmente donde CSMA/CD no puede ser implementado debido a la naturaleza del medio de comunicación. Un ejemplo típico es el protocolo IEEE 802.11, el cual especifica dos funciones que coordinan el acceso al medio: la DCF (*Distributed Coordination Function*), basada en CSMA/CA, y la PCF (*Point Coordination Function*). A diferencia de los nodos cableados, los nodos inalámbricos no pueden detectar colisiones debido a que son *half-dupex* es decir que no pueden enviar y recibir señales al mismo tiempo. Esta es la razón por la que no es aplicable a comunicaciones inalámbricas.

En IEEE 802.11, cada nodo sensa el medio antes de iniciar la transmisión. La detección de portador se realiza por alguno de dos mecanismos: uno físico y otro virtual. Los lapsos de tiempo se dividen en múltiples cuadros o segmentos y el intervalo de tiempo entre segmentos se llama espacio intersegmento (IFS, *inter-frame space*). Diferentes IFS son definidos para proveer niveles de prioridad para el acceso a medios inalámbricos[2].

Si se encuentra ocupado el medio al momento de sensar, el nodo espera el final de la transmisión actual y comienza la contención, denominado proceso de retroceso (*backoff process*). El nodo selecciona un tiempo de retroceso *rs* aleatorio dado por *rs=random*()·*lapsodetiempo* donde *random*() es un entero pseudo aleatorio proveniente de una

[2] Short IFS (SIFS), PCF IFS (PIFS), DCF IFS (DIFS) y extended IFS (EIFS) (Xia y Sun, 2008a).

distribución uniforme sobre el intervalo [0,*VC*] donde la ventana de contención (*VC*) es un entero dentro del rango de valores de las características de la capa física VC_{min} yVC_{max}, esto es, $VC_{min} \leq VC \leq VC_{min}$, y lapso de tiempo es el valor correspondiente de la capa física. La *VC* se incrementa exponencialmente al incrementarse el número de intentos de retransmisión del paquete. Durante el proceso de retroceso, el temporizador de retroceso se decrementa en términos del segmento de tiempo tanto como el medio esté desocupado. Cuando el medio está ocupado, el temporizador se detiene. Cuando el tiempo de retroceso concluye, si la red continúa desocupada, el paquete es enviado. El nodo que tiene el retardo de contención más corto gana y transmite su paquete. Los otros nodos solo esperan la siguiente contención (al finalizar el envío de este paquete). Si otra colisión ocurre, un nuevo tiempo de retroceso es elegido y el procedimiento de contención inicia de nuevo hasta que en algún momento el límite sea excedido. Debido a que la contención se basa en una función aleatoria y se usa para cada paquete, a cada nodo se le da una misma oportunidad de acceder al medio de comunicación. Una vez recibido un paquete íntegro, el nodo receptor espera un intervalo IFS y transmite un segmento de acuse de recibido positivo (ACK) de regreso al nodo emisor, lo que indica una transmisión exitosa. Si el nodo emisor no recibe el ACK, se asume que el paquete colisionó y se intenta retransmitir y entrar otra vez al procedimiento de retroceso. Para reducir la posibilidad de colisiones, la *VC* se duplica hasta el límite VC_{max} después de cada intento de retransmisión. *VC* se restaura al valor de VC_{min} después de cada transmisión exitosa. Ante la presencia de un error, un paquete tiene que ser retransmitido por el nodo emisor.

CSMA/CA entrega su mejor servicio, ofreciendo garantías en ancho de banda y retardos. Como CSMA/CD, este es un protocolo no determinista, pero tiene varias ventajas. Por ejemplo, CSMA/CA es idóneo para protocolos de red como TCP/IP, adaptándose muy bien a condiciones de tráfico variable y es muy robusto contra interferencias.

CSMA/BA. Significa CSMA con arbitraje de bit de paridad. Es un protocolo usado comúnmente para CAN [3](*Controller Area Network*) y DeviceNet para resolver colisiones. Este protocolo es referido algunas veces como un caso especial de CSMA/CD o CSMA/CA, también es llamado CSMA/AMP (CSMA con arbitraje sobre prioridad de mensajes). Como CSMA/CD y CSMA/CA, cada nodo en la red usando CSMA/BA debe esperar a que prescriba el periodo de inactividad antes de comenzar a enviar un mensaje. Una vez que su periodo transcurre, cada nodo en la red puede transmitir su mensaje. Arbitraje de bit de paridad significa que las colisiones son resueltas por medio de un bit de identificación, basado en una prioridad preprogramada de cada mensaje en el campo del identificador. Para CAN, el identificador es una única secuencia de 11 o 29 bits, que asigna una prioridad al mensaje y habilita al receptor para filtrar mensajes. Debido a que CAN es una red de *broadcast*, los mensajes no contienen direcciones explícitas del destinatario ni del emisor.

El mecanismo de arbitraje de CAN funciona de la siguiente manera: Un número de identificador de mensaje de tipo binario con valor pequeño indica la mayor prioridad. Un mensaje CAN asociado con la más alta prioridad ganará el arbitraje. Esto es logrado al transmitir datos por medio de un modelo binario de bits *dominantes* y *recesivos* donde el dominante es un "0" lógico y recesivo es un "1" lógico. Un bit dominante siempre sobre escribe un bit recesivo en el bus CAN. Si múltiples nodos están transmitiendo simultáneamente y un nodo transmite un 0, entonces todos los nodos monitoreando la red leerán un 0. Sólo si todos los nodos transmiten un 1 todos los nodos leerán un 1.

[3] CAN es un sistema estándar de comunicación de bus serial y escalable, originalmente desarrollado por BOSCH a mediados de los ochenta para aplicaciones automotrices.

CAN se comporta como un compuerta lógica AND. Durante la transmisión del campo del identificador, si un nodo transmite un 1 y lee un 0, esto significa que hubo una colisión con al menos un mensaje de mayor prioridad y, consecuentemente, este nodo aborta la transmisión del mensaje. El mensaje de mayor prioridad que se transmite procederá sin percibir ninguna colisión y transmitirá de manera exitosa.

El proceso de arbitraje de CAN es no destructivo, en el sentido que el nodo que gana el arbitraje sólo continúa con la transmisión del mensaje sin que el mensaje sea destruido o corrompido por otros nodos. El determinismo de CAN lo hace particularmente atractivo para usarse en Sistemas de Control en Tiempo Real.

Debido a su concepción, el protocolo CAN ha ganado popularidad en la industria de automatización, control médico, aplicaciones automotrices, etc. La mayor desventaja de CAN comparada con otras redes es su tasa de transmisión lenta. Esto es provocado principalmente por el mecanismo de arbitraje de bit de paridad, el cual impone estrictas limitantes sobre las características físicas de la red incluyendo longitud y tasa de transmisión de datos. Por ello, tasas de transmisión de datos mayores a 1 Mbps sólo son posibles si la longitud de la red es menor que 40 mts. Para redes de mayor longitud, la tasa de transmisión de datos decrece a 125 kbps para 500 mts.

3.4 Manejo de fallas en los NCS

En general, de acuerdo con el tipo de falla que se presenta en un sistema, se sigue una estrategia para mantenerlo con cierto grado de confiabilidad (Dubrova, 2008). En la tabla 3.1 se muestran los tipos de falla (sección 2.9) que se presentan en cualquier sistema genérico y las estrategias para hacerles frente.

Tipo	*Subtipo*	*Estrategia*
Común		Prevención, predicción
	Permanente	Prevención
Hardware	Transitoria	Enmascaramiento
	Intermitente	Predicción
Software		Detección, prevención

Tabla 3.1: Tipo de fallas y estrategias de solución

En particular, en la sección 2.1, se planteó el modelo de fallas de un SD donde se mencionan las fallas por omisión, fallas arbitrarias y fallas de tiempo que se presentan en los procesos y en la comunicación. Este tipo de fallas pueden reunirse en fallas de hardware y software. En la tabla 3.2 se muestran este tipo de fallas (Coulouris y otros, 2012).

Un aspecto importante a considerar en la actividad de los NCS es el funcionamiento del hardware. Posibles fallas en el medio de red y en los nodos impactan directamente en el desempeño del sistema. Diversas propuestas han abordado el problema de asegurar lecturas correctas en los sensores (Sharma y otros, 2007) pues en ciertas aplicaciones críticas como el control de aeronaves (Cork y otros, 2005; Oosterom y otros, 2002) los datos erróneos provenientes de sensores con falla en la lectura resultan en consecuencias graves. Es por esto que la redundancia de sensores (Benítez-Pérez y García-Nocetti, 2005) ha sido propuesta para enmascarar la falla en la lectura de los datos (Karimi y otros, 2010) y asegurar por medio del uso de técnicas de voto (Latif-Shabgahi y otros, 2003) un sensado confiable.

Tipo de falla	*Afecta a*	*Descripción*
Paro	Procesos	El proceso se detiene y permanece detenido. Otro proceso puede detectar su estado. El proceso se detiene y permanece detenido.
Ruptura	Procesos	Otro proceso probablemente no puede detectar su estado.
Omisión en la comunicación	Medio de comunicación	Un mensaje insertado en el buffer de salida nunca llega al buffer de llegada.
Omisión al enviar	Procesos	Un proceso completa el envío pero el mensaje nunca es colocado en el buffer de salida
Omisión al recibir	Procesos	Un mensaje es puesto en el buffer de llegada pero el proceso nunca lo recibe.
Arbitraria	Procesos o medio de comunicación	El proceso o el medio de comunicación experimenta un comportamiento arbitrario: Pueden recibir o transmitir mensajes arbitrarios en tiempo arbitrarios o tener omisiones, un proceso puede parar o ejecutar una acción incorrecta.

Tabla 3.2: Efecto de fallas por omisión y arbitrarias en procesos y medio de comunicación

Sharma y otros (2007) proponen la caracterización de fallas en la lectura de sensores de manera sencilla. Comienza por centrarse en un conjunto pequeño de fallas, con base en observaciones de distintas implementación e identifica breves picos en las lecturas de los sensores, ruido de larga duración y desplazamientos anómalos constantes. La observación de las estadísticas de las lecturas es lo que ofrece el criterio para determinar falla de los sensores. En la tabla 3.3 se describen las fallas en la lectura de los sensores.

Tipo de falla	*Descripción*
Breve	Existe un cambio esporádico en el valor medido entre dos datos sucesivos. La falla afecta a un punto de muestreo individual.
Ruido	La varianza de las lecturas del sensor aumenta. Estas fallas afectan lecturas sucesivas cada momento.
Constantes	Los sensores reportan un valor constante para varias lecturas sucesivas. Este valor constante es muy alto o bajo respecto a la lectura correcta del sensor y no tiene relación con el fenómeno físico que se mide.

Tabla 3.3: Tipos de fallas en lectura de sensores.

Redundancia de sensores. La redundancia es uno de los métodos más importantes para lograr la tolerancia a fallas y puede implementarse en tres formas, incluyendo métodos *estáticos* (enmascaramiento de falla), *dinámicos* (detección de fallas, diagnóstico de fallas,

aislamiento de fallas y ubicación de la falla), e *híbridos* (enmascaramiento de falla y detección y localización de fallas) (Karimi y otros, 2010).. El objetivo en redundancia estática es enmascarar el efecto de la falla en la salida del sistema.

La redundancia *N*−modular (RMN) es uno de los principales métodos de redundancia estática en hardware. La triple redundancia modular (TRM) es la forma más simple de RMN, que se forma a partir de *N*=3 módulos redundantes y una unidad de voto que arbitra las salidas de los módulos. En la figura 3.6 se muestra el sistema de TRM (Zarafshan y otros, 2010).

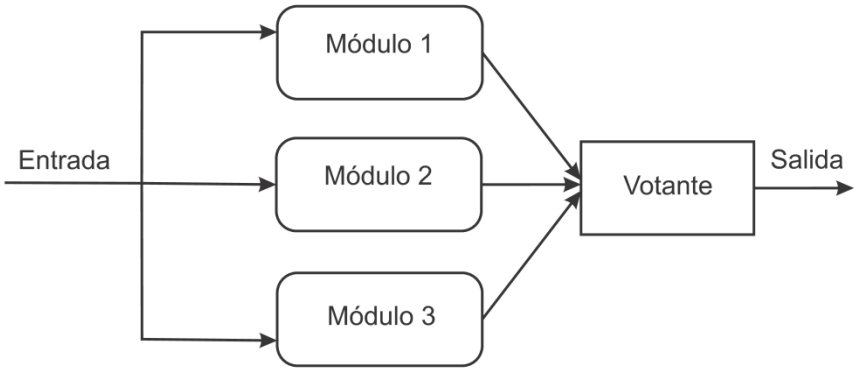

Figura 3.6: Triple redundancia modular.

La unidad de voto (votante) realiza un algoritmo de votación con el fin de arbitrar entre diferentes salidas de módulos redundantes y enmascarar el efecto de la falla(s) a partir de la salida del sistema. Con base en la aplicación, es posible utilizar diferentes tipos de algoritmos de votación.

Algoritmos de voto. Los algoritmos de voto (Latif-Shabgahi y otros, 2001, 2003; Benítez-Pérez y García-Nocetti, 2005; Benítez-Pérez y otros, 2007b) se utilizan ampliamente en los sistemas de control de tiempo real y de alta seguridad crítica. Se han aplicado en el reconocimiento de patrones, procesamiento de imágenes y en la organización de sistemas humanos, con la finalidad de arbitrar entre los resultados redundantes de procesamiento en módulos de hardware o software redundante. Los algoritmos de votación se pueden clasificar en:

- *Voto basados en acuerdos:* Voto donde se toma un acuerdo, que puede darse por pluralidad o por mayoría.

- *Voto sin acuerdo:* Producen una salida sin tomar en cuenta un acuerdo entre los resultado de las unidades redundantes.

En algunas aplicaciones es necesario utilizar los votantes del segundo tipo, donde se incluyen los algoritmos de voto de cálculo de media y de voto ponderado promedio. Aunque estos últimos tipos de algoritmos de voto son los más utilizados para aplicación de alta disponibilidad, el voto ponderado promedio es a menudo más confiable que el voto de cálculo de mediana (Zarafshan y otros, 2010).. Mientras que el votante mediano simplemente selecciona el valor medio de los resultados, el voto ponderado promedio asigna un peso a cada entrada con base en su prioridad predeterminada o sus diferencias, por lo que la participación de las entradas correctas será mayor que las que tienen errores. Algunas propuestas ofrecen tolerancia a fallas mediante la combinación de redundancia, algoritmos de voto y cómputo suave (redes neuronales o lógica difusa) (Oosterom y Babuska, 2000; Oosterom y otros, 2002; Zarafshan y otros, 2010) capaces de mejorar la tasa de fiabilidad del sistema.

Algoritmo de voto ponderado promedio . Un algoritmo de voto utilizado frecuentemente es el de *voto ponderado promedio* (Latif-Shabgahi y otros, 2003; Latif-Shabgahi, 2004), que usa triple redundancia modular para enmascarar fallas de hardware y software en tiempo de ejecución. Las salidas de tres módulos independientes que operan en paralelo con las mismas entradas alimentan una unidad de voto, que sirve de árbitro para operar entre ellos (figura 3.6). La unidad de voto recibe tres datos x_1, x_2, x_3, cada uno proporcionado por un módulo, y genera una salida *y*. De principio, el mecanismo consiste en determinar el *grado de cercanía* basado en la ecuación:

$$S_{ij} = \frac{1}{1 + pd_{ij}^{q}}$$

donde la función S_{ij}, llamada *indicador de acuerdo*, que proporciona un valor que indica la consistencia de las lecturas de entrada *i*, *j* basadas en su distancia numérica $d_{ij} = \|x_i - x_j\|$. Los parámetros *p*, *q* corresponden a la tasa de desplazamiento y el punto medio de la función, respectivamente.

Latif-Shabgahi y otros (2003) utiliza una tipificación de la función indicador de acuerdo, con valores de *q*=2 y $p = (0.5)^{-q}$, es decir:

$$S_{ij} = \frac{1}{1 + \left(\frac{d_{ij}}{0.5}\right)^{2}}$$

Una vez calculado el indicador de acuerdo de todos los pares de entradas, se asigna un peso de cada entrada *i* por medio de la expresión:

$$W_i = \frac{\sum_{j=1, j\neq i}^{n} S_{ij}}{n-1}$$

que indica la influencia del acuerdo entre cualesquiera entrada particular y las otras conforme a su peso. El valor de salida de la unidad de voto está dada por:

$$y = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Finalmente, se utiliza un comparador

$$|y - x_1| \leq \epsilon$$

que maneja un umbral de precisión ϵ. El comparador interpreta la salida del votante como correcta o incorrecta, es decir, si las entradas tienen integridad numérica o no.

3.5 Tolerancia a fallas mediante SMA

En el la sección 2.8 se plantea el modelo de interacción multiagente en el que la combinación de diversos módulos de software y hardware se conectan de acuerdo a los requerimientos del flujo de datos.

Ramírez (2006) resalta que cuando la interacción entre módulos es dinámica y no se precisa el momento en que cada módulo aparece o desaparece (falla), entonces es necesario utilizar una arquitectura de *blackboard* (BB). En (Ramírez-González y otros, 2007) se propone un algoritmo de planificación que maneja el cambio de frecuencias y prioridades de ejecución de tareas. El algoritmo está basado en un sistema multiagente cooperativo construido bajo una arquitectura de BB en el esquema distribución homogénea (*full-fledged*) (Corkill, 2003) para el manejo de información. En el BB cada agente puede escribir con el propósito de proveer una ruta de solución al problema. Se eligió tal configuración debido a que se requiere que los agentes sean independientes en su funcionamiento, esto es que continúen compartiendo información a pesar de que uno de ellos deje de funcionar (Ramírez, 2006).

Los agentes son los actores principales que continuamente están visualizando el BB para conocer los eventos que se presentan en el ambiente. Son capaces de modificar el BB si toman la decisión de realizar un cambio. Cada agente cuenta con un BB en el que escriben con el fin de actualizar los datos utilizados por cada tarea; el BB es implementado en el sistema como una tabla que tiene información para asignar ancho de banda, esta información refleja uno de los estados del ambiente en que funciona el SMA (Ramírez-González y otros, 2007). Cada agente cuenta también con una fuente de conocimiento (FC), basadas en competencias y vistas locales, que le permite interpretar información y ejecutar acciones conforme a sus objetivos. En la figura 3.7 (Ramírez, 2006) se presenta un esquema de comunicación entre agentes con distribución homogénea en el que cada agente, entidades hardware o software, poseen una FC y un BB; cada arista entre agentes corresponde a una comunicación por paso de mensaje.

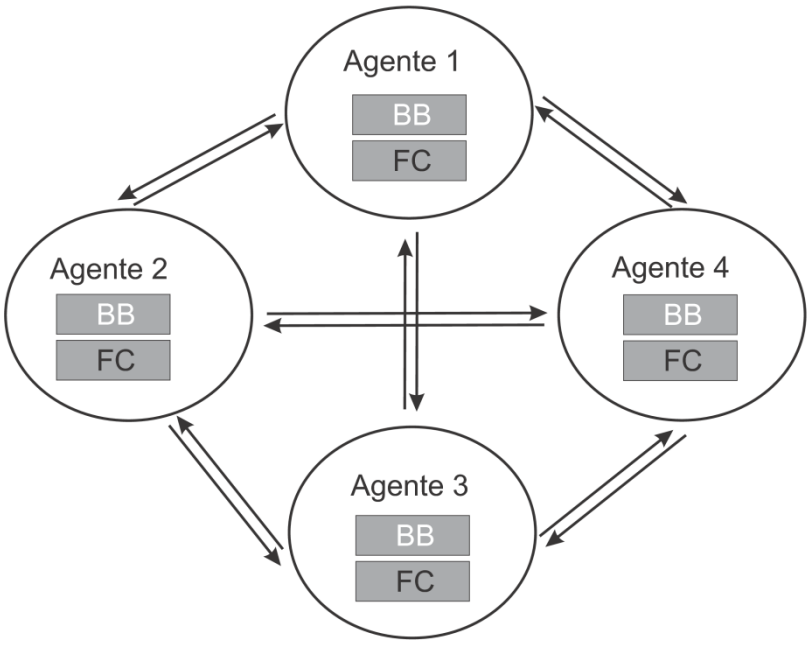

Figura 3.7: Sistema de interacción multiagente con blackboard (BB) y fuente de conocimiento (FC).

Este esquema de interacción multiagente muestra su efectividad para implementar estrategias de control reconfigurable basado en el trabajo cooperativo y el diagnóstico de falla usando la técnica Takagi-Sugeno (Ramírez-González y otros, 2007; Benítez-Pérez y otros, 2010, 2011).

Una propuesta interesante para lograr tolerancia a fallas de los NCS (TFNCS) con base en un SMA es desarrollada por Mendes y otros (2007, 2009). Tal propuesta surge al considerar los sistemas tradicionales de control tolerante a fallas (FTC por *fault tolerant control*) limitados en su capacidad de extensibilidad (sección 2.1) y reconfiguración, e inseguros por la centralización de dispositivos que provocaría una suspensión total motivada por una falla local. Mendes y otros (2009). consideran a los agentes como una tecnología que provee de manera natural una forma de superar tales problemas y de diseñar e implementar sistemas FTC distribuidos e inteligentes dentro de ambientes complejos y abiertos.

En (Mendes y otros, 2007) se exponen tres modelos de arquitectura multiagente para la FTNCS:

1. En la primera propuesta, el SMA tiene relaciones independientes. Los agentes FTC no tienen relaciones de dependencia con otros, son totalmente autónomos. Con esto se pretende lograr autonomía y modularidad, no tener agentes manejados por otros, que cada agente posea conocimiento de los demás agentes y de su entorno, tener metas y motivaciones propias. Esta arquitectura consiste de un *agente de control tolerante a fallas* a_{tf} responsable de monitorear parte del proceso, y sugiere acciones para la reconfiguración del controlador o de sustitución de entidades. Estos agentes son visibles a los demás agentes. Si alguna falla afecta a alguna parte del proceso de un a_{tf} que no le es posible reconfigurar, es capaz de solicitar una acción a otro agente a_{tf}. Todos los agentes son relacionados al proceso de monitoreo y reconfiguración deben reportar su estado al *agente de decisión de FTC*, que es el responsable de dar el resultado de la supervisión del estado actual del sistema. Algunas de las ventajas de esta propuesta son: la completa autonomía de los requerimientos FTC, estructura de relaciones independientes, menor comunicación en el sistema; sin embargo, la desventaja consiste principalmente en el incremento del número de agentes necesarios para tener un sistema FTC.

2. La segunda propuesta consiste en considerar un SMA autónomo con relaciones unilaterales entre las tareas de diferentes agentes, donde un agente depende de otro pero no viceversa. Agentes con las mismas tareas tienen relaciones recíprocas. Los agentes son más simples, en el sentido de tener poca competencia y tareas sencillas. Los agentes son totalmente autónomos respecto a los problemas de comunicación. En esta arquitectura se definen tres tipos de agentes: el *agente de detección de falla* (a_{fd}), el *agente de aislamiento e identificación de falla* (a_{fi}), el *agente de reconfiguración* (a_{fr}). Los agentes a_{fd} y a_{fi} no tienen percepción de los procesos ni actúan sobre ellos. El *agente de decisión de FTC* funciona de la misma manera que en la arquitectura previa. Las ventajas consisten en que se tienen agentes más simples, total competencia entre las tareas de los agentes para lograr sus metas, separación de tareas, menor negociación y comunicación. La desventaja consiste en que el agente de decisión de FTC debe tratar con mayor número de agentes. En la figura 3.8 se muestra un diagrama de esta propuesta de FTNCS basado en arquitectura SMA.

3. La tercera propuesta considera la construcción de una arquitectura dividida en federaciones. Las arquitecturas en federaciones pueden ser: facilitadores,

corredores y mediadores. En particular, la federación de facilitadores se encarga de intermediar la comunicación entre agentes y provee un servicio de ruteo de mensajes.

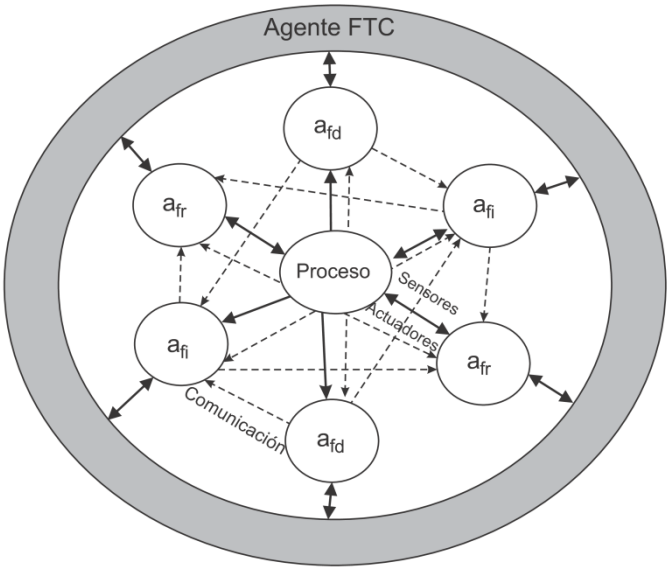

Figura 3.8: Sistema multiagente para el control tolerante a fallas.

Mendes y otros (2007, 2009)presentan una implementación de la arquitectura en federaciones modificada usando la herramienta FTNCS-MAS desarrollada específicamente para construir arquitecturas de este tipo. La herramienta está basada en *MATLAB/Simulink, xPC Target* y la herramienta de Cómputo Distribuido de *MATLAB*. En dicha implementación se tiene varios agentes facilitadores que reúnen federaciones de agentes a_{fd}, a_{fr} y a_{fi}, particionan el problema de FTC y cada federación se comunica por medio de UDP/IP.

Es pertinente señalar en este punto que los estrategias para lograr un sistema confiable como identificación de fallas, aislamiento y alojamiento de falla, detección, etc., sobrepasan el propósito de este trabajo. Es por esto que a partir de este momento se asume que el SDTR o NCS que se tome como caso de estudio tiene un mecanismo de identificación de fallas y sólo nos enfocaremos en las estrategias de reconfiguración. En el contexto de los SMA se tomarán en cuenta sólo agentes sensores y de reconfiguración.

3.6 Resumen

En este capítulo se establecieron las propiedades de un NCS y se expuso la arquitectura típica de un lazo de control cerrado por medio de una red de comunicación. Se hizo la revisión de los retardos de tiempo que se presentan en los NCS, se mencionó el efecto que provoca en los SDTR y se expuso simplificadamente la forma en que se han modelado. Se mencionaron los tipos de redes utilizadas actualmente para la transmisión de datos en tiempo real y se ofreció un panorama general sobre las fallas que pueden presentarse en los NCS. Se explicó que las estrategias de solución a fallas en la lectura de los sensores son tratadas por medio de redundancia y los algoritmos de voto, en particular, el de voto ponderado promedio. Se finalizó, revisando dos propuestas con base en SMA para el manejo de fallas en las que se resalta la interacción entre agentes, los roles y funciones que cada uno desempeña.

Capítulo 4

Impacto de la Tasa de Transmisión de datos en los NCSs

Las aplicaciones en las que se utilizan los NCS han crecido considerablemente en los últimos años. Actualmente, los NCS funcionan bajo restricciones de tiempo real en aplicaciones críticas, lo que ha resultado en el aumento de su complejidad. Las investigaciones en el área de los NCS han abordado diversos aspectos durante las etapas de análisis y diseño (Lian y otros, 2001c, 2002, 2003).

Recientemente, la tendencia de los SD es integrar cómputo, comunicación y control; lo cual involucra el uso de arquitecturas de control en red con bus común. Con esto se mejora la eficiencia, flexibilidad, y confiabilidad. No obstante, esta arquitectura acarrea diferentes tipos de retardo entre sensores-controladores-actuadores, como consecuencia de compartir el medio de comunicación y por el proceso propio de comunicación (Halevi y Ray, 1988a; Branicky y otros, 2003). Cervin y otros (2003) discuten la posibilidad de analizar el desempeño de controladores bajo los efectos de modificaciones en los intervalos de muestreo y retardos de comunicación y para ello sugieren la herramienta de análisis de tiempo real *TrueTime* (Cervin y otros, 2007).

En este capítulo se muestra el impacto de la tasa de transmisión de datos en los NCSs, que se resulta en la necesidad de planificar el medio de comunicación común. Se propone realizar dicha planificación por medio de reconfiguración dinámica. Se presenta una métrica para evaluar el desempeño de los NCSs y tener un valor cuantitativo de la calidad del control (QoC) del sistema. Esta métrica está dada en términos del error de salida del sistema a considerar. Como motivación se presenta un ejemplo en el que se muestra el efecto que provoca en el QoC la elección de diferentes tasas de transmisión de datos. Finalmente, se expone el sistema físico en el que se basa el análisis e implementación de las estrategias de reconfiguración que se proponen. Este caso de estudio consiste del control de un prototipo de helicóptero de dos hélices con dos grados de libertad. El comportamiento no lineal del helicóptero es de interés pues permite hacer consideraciones más cercanas al comportamiento de sistemas reales con restricciones temporales.

4.1 Consideraciones de diseño de un NCS

Para Menéndez y Benítez-Pérez (2010), la política de planificación de un SDTR debe maximizar la relación entre el número de tareas cuya ejecución se realiza contra el número de

tareas que llegan un nodo para ser ejecutadas, minimizar el costo de las comunicaciones por la red y balancear la carga de los nodos del sistema. Asumen para el análisis de la planificación dinámica del medio de comunicación lo siguiente:

- Las tareas pueden llegar dinámicamente a cualquier nodo del sistema.
- El estado de los nodos cambia constantemente.
- El algoritmo de planificación requiere información acerca del estado de varios nodos.
- La información disponible de los nodos remotos no es exacta debido a los retardos en la comunicación.
- Las decisiones de planificación se toman con base en información desactualizada.

Un SDTR puede contener diferentes tipos de recursos, entre los que se incluyen los recursos computacionales y de comunicación. En un NCS el desempeño de los lazos de control no depende únicamente del diseño del algoritmo de control, sino también de la planificación de los recursos. Por tal motivo, es necesario encontrar una planificación de los *recursos compartidos* de manera que se cumplan las restricciones temporales impuestas a las tareas de cómputo y a la comunicación de mensajes. El mecanismo de envío y recepción de mensajes es proporcionado por los protocolos de comunicación (sección 3.3). Los mensajes suelen tener un plazo desde que se solicita su envío, hasta que se reciben. Para garantizar estos plazos y acotar el tiempo de envío, se utilizan protocolos deterministas (Yu y otros, 2005). Asimismo, en muchos casos, la arquitectura de la red introduce una sobrecarga grande. En los NCS la tasa de envío de datos depende exclusivamente del periodo de las tareas del ciclo de control, en particular del periodo de muestreo. Aunque se asume que una tasa de muestreo alta mejora el desempeño del control en los sistemas digitales, esto induce un carga alta en el tráfico de la red. Este aumento induce a su vez retardos que pueden degradar el desempeño global del sistema (Lian y otros, 2001b, 2002; Xia y Sun, 2008a). Por lo anterior, encontrar una tasa óptima de transmisión de datos que no provoque saturación o bajo muestreo es un elemento fundamental en el análisis y diseño de los NCS(Menéndez y Benítez-Pérez, 2010).

Cuando se utiliza un SDTR para implementar un lazo de control se requiere estudiar el protocolo de red, evaluar el desempeño de la red de comunicación e identificar el impacto que provocan los retardos en el desempeño de control. Lian y otros (2001c) discuten ciertas consideraciones para el diseño de la red de comunicación y los parámetros de control con base en lo que llaman *gráfica de diseño de NCSs* (networked control design chart). Esta gráfica surge de comparar el rendimiento de control contra la tasa de transmisión de datos. Cuando se selecciona una red de comunicación para aplicaciones de control, surgen dos problemas importantes a tratar: el tiempo que tarda un mensaje desde que se emite hasta que se recibe y la confiabilidad de la transmisión. Principalmente, el periodo de muestreo y el ancho de banda del sistema de control son dos parámetros de control que se consideran para caracterizar el desempeño de control y tienen una fuerte influencia en los parámetros de la red. Debido a que un NCS es esencialmente un sistema discreto, elegir una tasa de muestreo apropiada en el sensado y en la actuación es tan importante como el diseño del controlador. La tasa de muestreo depende del ancho de banda del sistema de control el cual se define como la máxima frecuencia a la cual la salida del sistema puede seguir una referencia de entrada de manera satisfactoria (Lian y otros, 2003).

Debido a la relación estrecha entre la red y los parámetros de control, Lian y otros (2002, 2003) analizan la arquitectura de los NCS con la finalidad de elegir la mejor tasa de muestreo con el que un NCS tiene el desempeño deseado. Para garantizar el desempeño de control deseado, todos los nodos necesitan información actualizada de los demás nodos, por esta razón Lian y otros (2001b) consideran la tasa de muestreo y la tasa de transmisión de datos iguales para todos los nodos, esto es que se transmite cada dato en cuanto se toma su lectura. Sin embargo, elegir una tasa de muestreo rápida para garantizar un desempeño aceptable de control podría potencialmente saturar la red y eventualmente incrementar el tiempo total de transmisión, por consiguiente, degradar el desempeño de control. Como alternativa se propone desacoplar el periodo de muestreo y tasa de transmisión de datos Lian y otros (2002). Para lograr el desempeño deseado de control (QoC) y el desempeño deseado de la red (QoS) bajo un ancho de banda limitado Lian y otros (2003) utilizan estimadores de estados para ajustar las tasas de comunicación y garantizar con ello el desempeño de control estipulado.

La gráfica de diseño de un NCS (figura 4.1) (Lian y otros, 2002) provee una forma de elegir una tasa de transmisión de datos. En esta gráfica se compara el desempeño de control contra el periodo de muestreo en tres tipos de sistemas de control: continuo, digital y en red. Las regiones “peor”, “inaceptable”, “aceptable”, “mejor” son definidas basadas en las especificaciones requeridas del sistema de control (sobrepico, tiempo de asentamiento, error en estado estacionario, etc.). El eje de desempeño en la figura 4.1 puede se elegido de tal forma que refleje un subconjunto de estas métricas. Lian y otros (2002) establecen la existencia de puntos de modificación de desempeño, correspondientes a distintos periodos de muestreo.

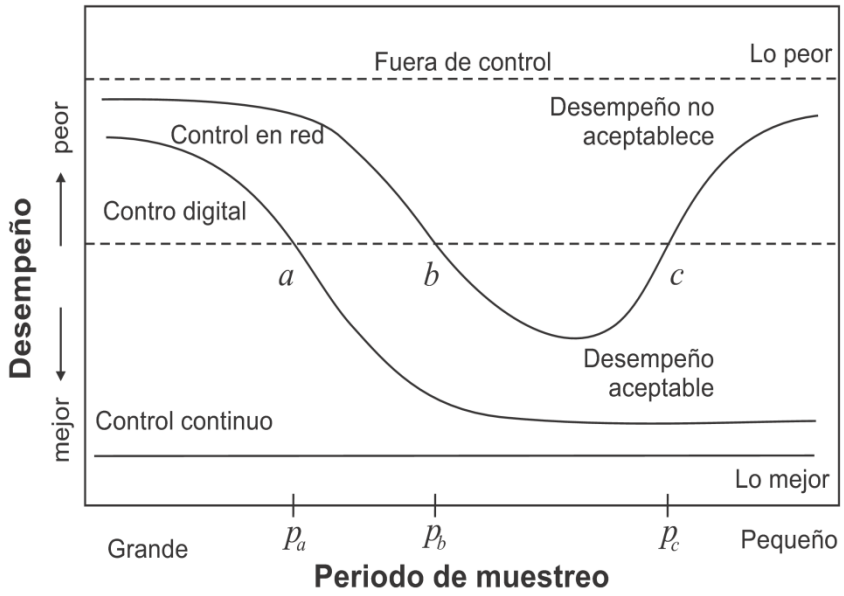

Figura 4.1: Desempeño de control respecto al periodo de muestreo

Debido a que el desempeño de los sistemas de control continuo no está en función del periodo de muestreo, el nivel de desempeño es constante para una ley de control fija. Para el control digital, el desempeño depende sólo del periodo de muestreo asumiendo que no existen otras incertidumbres.

En este caso, en la figura 4.1 el punto de degradación *a* corresponde al periodo de muestreo p_a y puede ser estimado con base en la relación entre el ancho de banda y la tasa de muestreo. En el caso del control en red se presentan los puntos *b* y *c* correspondientes a los periodos de muestreo p_b y p_c (dentro del intervalo $[p_b, p_c]$ es posible lograr el desempeño óptimo del sistema). El punto *b* puede determinarse haciendo una estadística que determine las características de los retardos inducidos por la red y el retardo de los dispositivos para distintos valores del periodo de muestreo. Mientras el valor del periodo de muestreo continua disminuyendo, el tráfico de red se hace pesado y la posibilidad de pérdida de datos aumenta cuando se tiene una red con ancho de banda limitada, esto resulta en retardos de tiempo más largos Lian y otros (2001c).Esta situación provoca la existencia del punto *c* en el que vuelve a perderse desempeño de control debido a la sobrecarga de la red.

En síntesis, en los NCSs la operación usando periodos de muestreo grandes resulta en la degradación del desempeño de control debido a la baja tasa de datos para calcular la ley de control, mientras que la operación con periodos de muestreo pequeños sobrecargan la red y el desempeño de control también es afectado.

Lian y otros (2006) muestran el efecto de elegir un periodo de muestreo particular y el impacto de los retardos de tiempo en el desempeño de NCS, en los que confluyen varios agentes y estiman de forma determinista los tiempos de muestreo que delimitan la región de control aceptable del NCS; por lo tanto, el periodo de muestreo es un parámetro de importancia fundamental para mantener cierto nivel de desempeño (estabilidad) del sistema.

Si la cantidad de datos generados por los sensores es mayor que los datos que pueden ser transmitidos por la red, entonces la red se satura y los datos quedan en espera en el buffer. Por lo tanto, se debe considerar un ancho de banda y un periodo de muestreo efectivos para el diseño de los NCS. Si los parámetros de planificación del sistema producen una planificación factible, es posible modificar el periodo de muestreo del lazo de control dentro del conjunto de restricciones impuestas por la calidad de desempeño de control (QoC) y la calidad del servicio (QoS) para obtener un desempeño óptimo global. Sin embargo, es posible que se presente pérdida de desempeño en el límite de los parámetros de planificación, aún cuando no se tenga otro tipo de incertidumbres, aunque esto no sucede en cualquier caso.

Por esta razón, es necesario considerar dentro de los objetivos de reconfiguración la modificación de los periodos de muestreo para lograr niveles de desempeño deseados. Con esto se logra manejar el efecto de los retardos de tiempo inducidos por la red sin desarrollar un control específico para ello. La reconfiguración de los periodos de muestreo de un sistema de control en red constituye una importante base metodológica de este estudio. En las secciones siguientes se muestra como se aprovechan estos resultados para definir esquemas de reconfiguración efectivos.

4.2 Calidad de control

El requisito esencial de un sistema de control es la estabilidad[1]. Existen varias versiones de la definición de estabilidad, por ejemplo, se dice que un sistema es *estable* acotado en entrada/acotado en salida (BIBO por Bounded-input/Bounded-Output) si para cualquier entrada acotada, la salida es acotada. Si la salida del sistema no converge y produce

[1] Este análisis se restringe a sistemas lineales invariantes en el tiempo.

oscilaciones sinusoidales se dice que es un sistema *marginalmente estable*. Un sistema es *inestable* si la salida del sistema diverge. La estabilidad de un sistema a lazo cerrado puede determinarse por la ubicación de sus polos. Un sistema de control lineal e invariante en el tiempo es estable cuando en el dominio de tiempo continuo todos sus polos se encuentran en la mitad izquierda del plano complejo *s* o en el dominio de tiempo discreto todos sus polos se encuentran dentro del círculo unitario en el plano *z* (Xia y Sun, 2008a).

Además de la estabilidad, el comportamiento transitorio es otro foco de atención para el diseño de los sistemas de control. Hay varias propiedades que se pueden utilizar para evaluar el comportamiento transitorio de un sistema de control a lazo cerrado. Por ejemplo: el tiempo de asentamiento y el sobrepico. El diseño de controladores que considere las propiedades anteriormente descritas, así como el análisis formal de estabilidad, trascienden el objetivo de este trabajo, sin embargo puede consultarse en Chen (1999) y Hespanha (2009).

Un criterio generalmente utilizado para evaluar el QoC[2] considera el error de salida, que se define como la diferencia entre la referencia o valor nominal *r*(*t*) y la salida del sistema *y*(*t*). Algunos de estos índices de desempeño se presentan tanto en la forma de tiempo continuo como en la forma de tiempo discreto, donde $t_0(k_0)$ y $t_f(k_f)$ son los tiempos (continuo/discreto) iniciales y finales del intervalo de evaluación:

- Integral del valor absoluto del error:

$$IAE = \int_{t_0}^{t_f} |e(t)|dt \approx \sum_{k=k_0}^{k_f} |r(kh) - y(kh)|$$

- Integral del valor absoluto del error ponderado en el tiempo:

$$ITAE = \int_{t_0}^{t_f} t|e(t)|dt \approx \sum_{k=k_0}^{k_f} |r(kh) - y(kh)|$$

- Integral del cuadrado del error:

$$ISE = \int_{t_0}^{t_f} e^2(t)dt \approx \sum_{k=k_0}^{k_f} \big(r(kh) - y(kh)\big)^2$$

- Integral del cuadrado del error ponderado en el tiempo:

$$ITSE = \int_{t_0}^{t_f} te^2(t)dt \approx \sum_{k=k_0}^{k_f} k\big(r(kh) - y(kh)\big)^2$$

Mientras que la IAE y ISE pondera todos los errores por igual, ITAE e ITSE pondera los errores con penalización en el tiempo, esto es, los errores posteriores tienen mayor peso que los iniciales. Criterios diferentes pueden ser elegidos para cumplir diferentes necesidades de evaluación del desempeño. El valor más grande corresponde al peor QoC.

[2] El valor del QoC también es referido como el costo de control.

4.3 Ejemplo del efecto del periodo de muestreo en un NCS

La planificación de los NCS tienen varias características similares (Xia y Sun, 2008a). La planificación de tareas en un procesador puede ser análoga a la planificación de paquetes que se transmiten por medio de la red. Por otra parte, trabajos que atacan el problema de planificación de NCS asumen necesariamente que el periodo de muestreo es menor que la *máxima cota de retardo permisible* (MADB por *maximum allowable delay bounds*) sin considerar la relación entre el periodo de muestreo y el rendimiento (Xia y Sun, 2008a).. Si la cantidad de datos generados por los sensores es mayor que la cantidad de datos que la red puede transmitir, entonces la red se satura y los datos estarán encolados en el buffer. Por lo tanto, un periodo de muestreo efectivo y un ancho de banda apropiados deben ser considerados simultáneamente en el diseño y análisis de un NCS. Si los parámetros originales del sistema son planificables, es posible reconfigurar el periodo de muestreo del o de los lazos de control dentro del conjunto de restricciones impuestas por el QoC y QoS. Por lo tanto es importante explorar la interacción entre el rendimiento de control, el periodo de muestreo y los algoritmos de planificación empleados.

Para ilustrar el efecto del periodo de muestreo en el desempeño del sistema, se considera como ejemplo la simulación del control distribuido de un motor DC bajo restricciones de tiempo real. Para este NCS se tiene:

- El sistema físico a controlar (planta) es un motor DC con función de transferencia $\frac{1000}{s^2+s}$.
- El controlador utiliza un esquema de control proporcional integral derivativo (PID).
- El ciclo de control se realiza utilizando tres procesadores, cada uno realiza la acción de sensado, control y actuación; además de un medio de comunicación común por el que se intercambia información.
- Los nodos controlador y actuador son activados por evento, mientras que el nodo sensor es activado por tiempo.
- El algoritmo de planificación en el nodo sensor es de prioridad fija.
- El tipo de red elegida es CAN con una tasa de transmisión de datos de 80,000 (bits/s) y un tamaño mínimo de paquetes de 40 bytes.
- El tiempo de simulación es de 10 segundos.

Se elige la integral del valor absoluto del error (IAE) como índice de desempeño del sistema

$$IAE = \int_{t_0}^{t_f} |e(t)| dt \approx \sum_{k=k_0}^{k_f} |r(kh) - y(kh)|$$

donde $r(t)$ es la señal de referencia, $y(t)$ la señal de salida del sistema, t_0y t_f son los tiempos inicial y final de la simulación.

La simulación de los bloques de kernel y de red se hacen con la herramienta de tiempo real *TrueTime* (Cervin y otros, 2003, 2007) basada en *MATLAB y Simulink*.

En la figura 4.2 se muestra el diagrama de bloques para la simulación del sistema completo. El sensor (nodo 4) toma la lectura $y(t)$ del sistema físico (DC Servo) y envía, por medio del paso de mensajes a través de la red, la lectura al controlador (nodo 3). El controlador recibe la señal de referencia por la entrada A/D y calcula la ley de control que envía por la red al actuador (nodo 2), la entrada de control $u(t)$ es aplicada al proceso físico. Bloques auxiliares como los visualizadores y un nodo de interferencia (generador de tráfico) complementan el modelo para efectos de análisis.

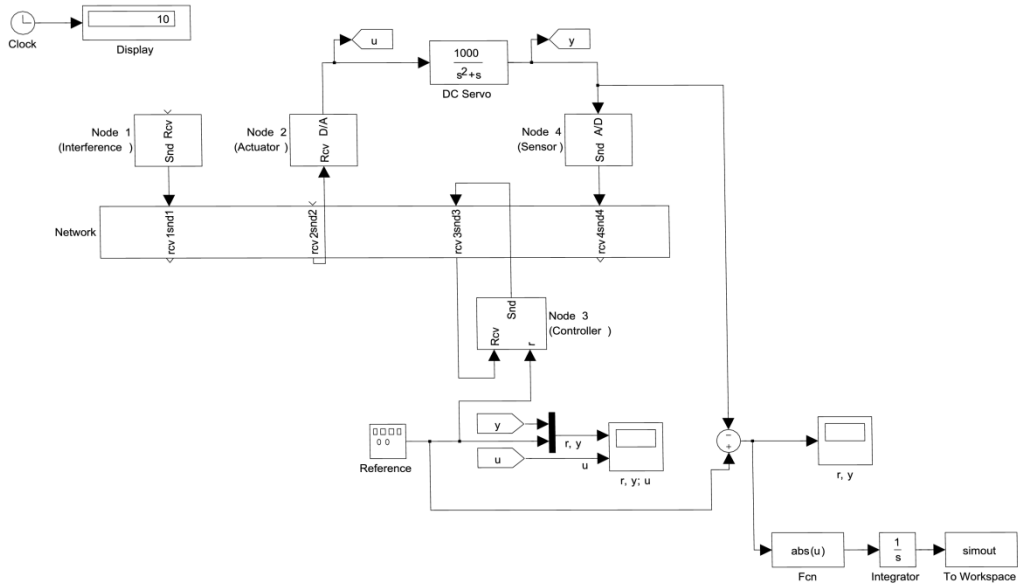

Figura 4.2: Modelo en Matlab/Simulink del control de un motor DC.

Los tiempos de muestreo (periodos de tareas del sensor) se tomaron desde 2 milisegundos hasta 14 milisegundos (ms.). Para cada modificación del periodo de muestreo se calcula el valor del IAE y se resumen en la tabla 4.1.

Tabla 4.1: Valores de la IAE para diferentes valores del periodo de muestreo en el sistema de control de un motor DC

	Periodo de muestreo (ms.)												
	2	3	4	5	6	7	8	9	10	11	12	13	14
IAE	18.92	9.36	6.71	5.14	4.36	3.71	3.35	3.06	2.74	3.19	3.56	4.01	4.54

Como puede observarse en la tabla 4.1 y en la figura 4.3 la relación entre el IAE y el periodo de muestreo genera una curva cóncava [3]. En ésta se observa que para periodos de muestreo de 2 milisegundos (ms.) el valor del IAE es de 18.92, cuando el periodo es de 10 ms. el valor del

[3] Se ajustaron los datos a una curva de segundo grado por medio de mínimos cuadrados

IAE es el mínimo con 2.74, finalmente, para valores del periodo de muestreo mayores a 14 ms. el valor del IAE crece considerablemente. De lo anterior es posible subrayar que existen, efectivamente, ciertos valores en los periodos de muestreo que sirven como cota superior e inferior para obtener un nivel de desempeño aceptable, es decir, valores cercanos al mínimo valor del IAE.

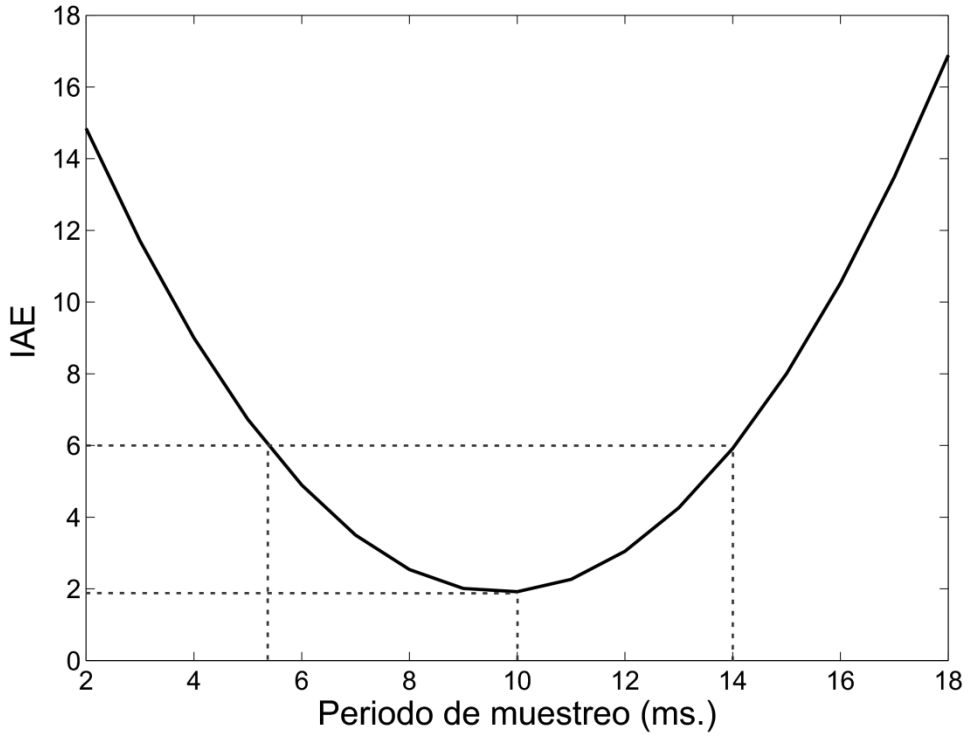

Figura 4.3: Relación del IAE y el periodo de muestreo.

Para periodos de muestreo inferiores a 4 ms., el número de datos que se envían por la red se incrementa y el tráfico de la red provoca pérdida de datos además de la aparición de retardos. En cambio, periodos de muestreo superiores a 14 ms. resulta en el envío de pocos datos al controlador lo que provoca un deterioro en el cálculo de la ley de control. En las figuras 4.4, 4.6, 4.5se muestra la respuesta del sistema para diferentes periodos de muestreo. Para el periodo de muestreo 10 ms., donde el IAE tiene el mínimo valor (figura4.4), el sistema logra un seguimiento aceptable de la señal de referencia asignada. Para periodos de muestreo de 2 ms. (figura4.5) y 14 ms. (figura4.6) el sistema pierde el seguimiento con lo que el error de salida se incrementa.

4.4 Planteamiento del problema

Como se puede observar de la sección anterior, la elección del periodo de muestreo en un NCS impacta directamente en el desempeño del sistema. La consideración principal es la utilización del medio de comunicación compartido. Las redes de los NCS comúnmente son de ancho de banda limitado, lo que restringe la cantidad de datos que se transmiten, de tal manera que el efecto de saturar la red o subutilizarla es uno de los objetivos de análisis en las investigaciones hechas en los últimos años (Lian y otros, 2001c, 2002, 2003). Se han propuestos diversas metodologías para atender este problema, entre ellas están las que proponen el uso de la planificación retroalimentada (Cervin y Eker, 2000) y el uso de técnicas de codiseño (Xia y Sun, 2008a,b).

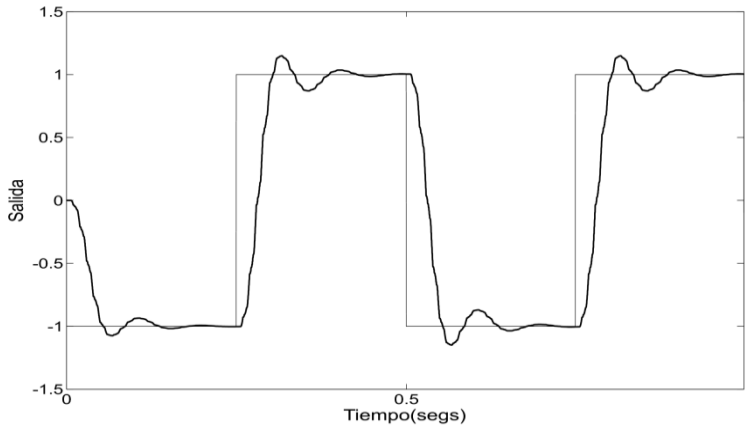

Figura 4.4: Seguimiento de la señal de referencia para un periodo de muestreo de 10 ms.

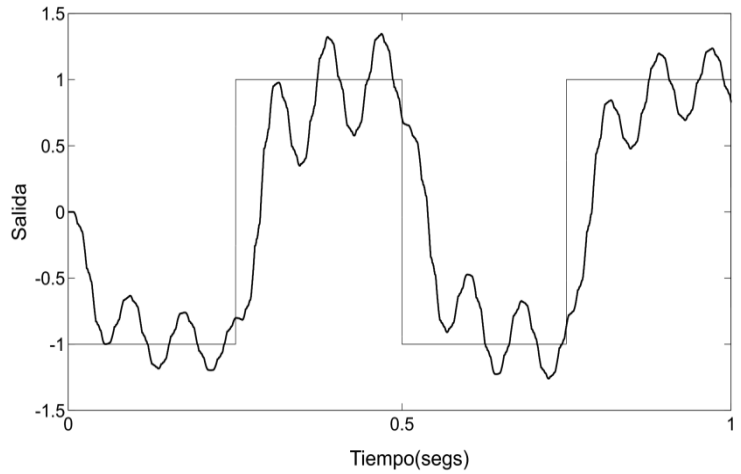

Figura 4.5: Seguimiento de la señal de referencia para un periodo de muestreo de 2 ms.

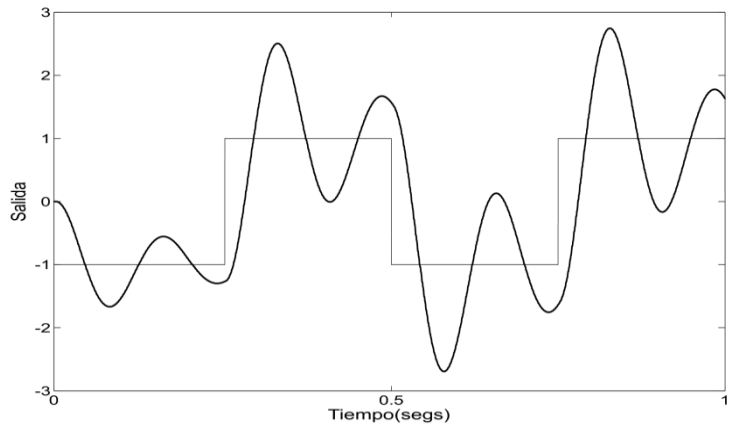

Figura 4.6: Seguimiento de la señal de referencia para un periodo de muestreo de 14 ms.

En este trabajo, se busca reconfigurar los parámetros de planificación de tareas con el objetivo de encontrar un periodo de muestreo que equilibre el uso de la red, asegurando el cumplimiento de las restricciones de tiempo.

Uno de los problemas a resolver consiste en:

> *Mantener o modificar, por medio de la reconfiguración dinámica, los periodos de muestreo de un NCS de manera que permanezcan dentro de un intervalo donde el sistema no pierda desempeño, evitando la pérdida de plazos.*

Derivado de lo anterior, para NCSs con distintas señales a sensar o con varios lazos de control, el uso de un único canal de comunicación afecta el desempeño del sistema debido a la competencia entre nodos por el recurso. Por lo tanto, otro problema a resolver es:

> *Encontrar una planificación del recurso de red por medio de la reconfiguración de las tasas de transmisión de tal forma que se reduzcan los retardos en la comunicación.*

Un NCSs confiable es necesario en tareas de alta precisión y críticas. La ocurrencia de una falla, no sólo requiere un mecanismo de reconfiguración, entre otros, para recuperarse de dicho escena, sino que ciertas estrategias de tolerancia a fallas, que requieren mayor intercambio de datos, como lo puede ser la redundancia de sensores (Benítez-Pérez y García-Nocetti, 2003,2005), aumenta la carga sobre la red. De aquí, que nuestro siguiente problema a resolver consiste en:

> *Diseñar estrategias de reconfiguración dinámica que permitan tolerar fallas, al tiempo que manejen la planificación del medio de comunicación por medio de la reconfiguración de los parámetros de tiempo real.*

El SDTR elegido para propósitos de implementación en este trabajo consiste del prototipo de un helicóptero de 2 hélices con dos grados de libertad (DOF por *degree of freedom*). En las siguientes secciones se introduce y describe el prototipo del helicóptero y el diseño del controlador.

4.5 Caso de estudio

Los sistemas de control en red se han aplicado de manera importante en los sistemas de vuelo de aeronaves comerciales. En la industria aérea distintas técnicas de cómputo suave (Oosterom, 2005), lógica difusa difusa(Oosterom y Babuska, 2000; Méndez-Monroy y Benítez-Pérez, 2011a; Quiñones-Reyes y otros, 2012), redes neuronales (Liu y otros, 2008; Napolitano y otros, 1999) se han probado e implementado para el control de vuelo de helicópteros (Castillo y otros, 2007; García-Sanz y otros, 2006; López-Martínez y otros, 2007; Romero y otros, 2010) donde se resalta el modelo dinámico de helicópteros de hasta 4 hélices y las técnicas de control no lineal utilizadas. Otros estudios se enfocan en la detección, identificación y manejo de fallas para mejorar la estabilidad del vuelo (Oosterom y otros, 2002; Cork y Walker, 2007; Heredia y Ollero, 2009; Satnam y otros, 2001; Silvestri y otros, 1994; Belcastro, 2001).

Como caso de estudio se utiliza el prototipo de helicóptero de la marca *Quanser*, para evaluar el impacto de la reconfiguración distribuida con las características planteadas en las secciones precedentes. El modelo dinámico del vuelo y el modelo de control de posición del helicóptero están implementados en *Matlab/Simulink* y utilizan la herramienta de simulación de tiempo real *Truetime (Cervin y otros, 2003, 2007; Henriksson y otros, 2006a,b).* Este simulador facilita el análisis temporal de diversos sistemas distribuidos y su diseño, proporciona diversos módulos que simulan un sistema operativo de tiempo real y distintos tipos de redes de comunicación (Ohlin y otros, 2007).. En el anexo C se exponen las características y bloques de simulación de esta herramienta.

El prototipo de helicóptero utiliza dos motores de corriente directa que hacen girar los propulsores que controlan la elevación de la nariz del helicóptero sobre el eje horizontal o cabeeo y la desviación respecto al eje vertical también conocido como guiñada. Estos ángulos son medidos por medio de *encoders* de alta resolución (Quanser, 2006).

En la tabla 4.2 se muestran las especificaciones y parámetros del modelo como longitudes, masas y momentos de inercia asociados al helicóptero:

Tabla 4.2: Parámetros del modelo en Simulink/Matlab del Helicóptero 2DOF.

Símbolo	MATLAB	Descripción	Valor	Unidad
$B_{eq,p}$	B_eq_p	Amortiguamiento eje cabeceo	0.800	*N/V*
$B_{eq,y}$	B_eq_y	Amortiguamiento eje guiñada	0.318	*N/V*
m_{heli}	m_heli	Masa total helicóptero	1.387	*kg*
$m_{m,p}$	m_motor_p	Masa motor cabeceo	0.292	*kg*
$m_{m,y}$	m_motor_y	Masa motor guiñada	0.128	*kg*
m_{shield}	m_shield	Masa del propulsor escudo	0.167	*kg*
m_{props}	m_props	Masa propulsores cabeceo, guiñada, escudos y motores	0.754	*kg*
$m_{body,p}$	m_body_p	Masa sobre eje cabeceo	0.633	*kg*
$m_{body,y}$	m_body_y	Masa sobre el eje guiñada	0.667	*kg*
m_{shaft}	m_shaft	Masa del mango del eje cabeceo	0.151	*kg*
L_{body}	L_body	Longitud total helicóptero	0.483	*m*
l_{cm}	l_cm	Centro de masa sobre eje cabeceo	0.186	*cm*
L_{shaft}	L_shaft	Longitud mango del eje guiñada	0.280	*m*
$J_{body,p}$	J_m_p	Momento de inercia sobre el eje cabeceo	0.012	$kg{\cdot}m^2$
$J_{body,y}$	J_m_y	Momento de inercia sobre el eje guiñada	0.013	$kg{\cdot}m^2$
J_{shaft}	J_shaft	Momento de inercia del mango sobre eje guiñada y punto final	0.004	$kg{\cdot}m^2$
J_p	J_p	Momento de inercia motor/escudo sobre pivote cabeceo	0.018	$kg{\cdot}m^2$
J_y	J_y	Momento de inercia motor/escudo sobre pivote guiñada	0.008	$kg{\cdot}m^2$
$J_{eq,p}$	J_eq_p	Momento total de inercia sobre pivote de cabeceo	0.038	$kg{\cdot}m^2$
$J_{eq,y}$	J_eq_y	Momento total de inercia sobre pivote de guiñada	0.043	$kg{\cdot}m^2$

Modelo Dinámico. El modelo matemático de la dinámica del prototipo del helicóptero (figura 4) (Quanser, 2006) es no lineal con dos grados de libertad correspondientes al ángulo de cabeceo θ (*pitch degree*) y al ángulo de guiñada φ (*yaw degree*). El ángulo de cabeceo es

definido como positivo cuando la nariz del helicóptero está sobre el eje horizontal y el ángulo de guiñado es positivo cuando gira a la derecha del eje vertical. Existen fuerzas F_p, F_y que actúan perpendicularmente a los ejes con lo que el torque del eje cabeceo es aplicado a la distancia r_p y el torque de guiñada a la distancia r_y. La fuerza de gravedad F_g genera un torque en el centro que jala hacia abajo la nariz del helicóptero. El centro de masa es la distancia l_{cm} medida desde el ángulo de cabeceo y a lo largo del cuerpo del helicóptero.

Las ecuaciones no lineales de movimiento se calculan utilizando las ecuaciones Euler-Lagrange:

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}_1} - \frac{\partial}{\partial q_1}L = Q_1$$

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}_2} - \frac{\partial}{\partial q_2}L = Q_2$$

donde *L* es el Lagraniano y es la diferencia entre la energía cinética y la energía potencial del sistema *L=T−V*.

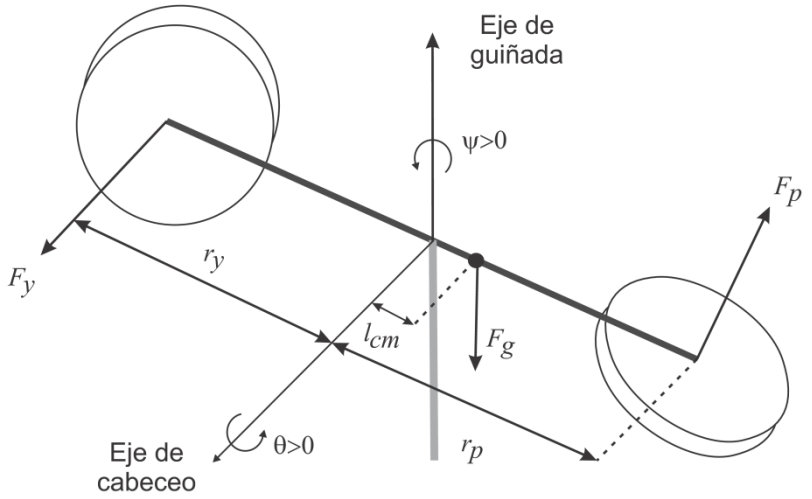

Figura 4.7: Modelo de la dinámica del Helicóptero con 2 grados de libertad.

Además de las ecuaciones de Euler-Lagrange, para calcular las ecuaciones de movimiento se utilizan las coordenadas y fuerzas generalizadas, la fricción rotacional de los ejes, los torques aplicados a los ejes, así como los voltajes de los motores de los ejes. Las ecuaciones de movimiento son:

$$\left(J_{eq,p} + m_{heli}l_{cm}^2\right)\ddot{\theta} = K_{pp}V_{m,p} + K_{py}V_{m,y} - m_{heli}g l_{cm}\cos\theta - B_p\dot{\theta} - m_{heli}g l_{cm}^2 \sin\theta\cos\theta\psi^2 \quad (4.1)$$

$$\left(J_{eq,y} + m_{heli}l_{cm}^2\cos\theta^2\right)\ddot{\psi} = K_{yy}V_{m,y} + K_{yp}V_{m,p} - B_p\dot{\psi} + 2m_{heli}l_{cm}^2\sin\theta\cos\theta\dot{\psi}\,\dot{\theta} \quad (4.2)$$

El análisis de movimiento, de la energía potencial y cinética, así como la derivación de las ecuaciones no lineales de movimiento del helicóptero pueden consultarse en [**¡Error! No se encuentra el origen de la referencia.**]. Con las ecuaciones de movimiento se deriva el modelo lineal en el espacio de estados y el modelo de control del helicóptero.

Modelo lineal en espacio de estados. Como el helicóptero representa un sistema no lineal,

se pueden linealizar las ecuaciones 4.1 y 4.2 alrededor del punto de equilibrio $(\theta_0 = 0, \psi_0 = 0, \dot{\theta}_0 = 0, \dot{\psi}_0 = 0)$, lo que resulta en:

$$\left(J_{eq,p} + m_{heli} l_{cm}^2\right)\ddot{\theta} = K_{pp} V_{m,p} + K_{py} V_{m,y} - B_p \dot{\theta} - m_{heli} g l_{cm} \tag{4.3}$$

$$\left(J_{eq,y} + m_{heli} l_{cm}^2\right)\ddot{\psi} = K_{yy} V_{m,y} + K_{yp} V_{m,p} - B_y \dot{\psi} - 2 m_{heli} l_{cm}^2 \theta \dot{\psi} \dot{\theta} \tag{4.4}$$

Sustituyendo el estado $x = [\theta \quad \psi \quad \dot{\theta} \quad \dot{\psi}]^\intercal$ en las ecuaciones 4.3 y 4.4 y resolviendo para $\dot{x}$ se obtiene el modelo lineal de espacio de estados:

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\dfrac{B_p}{J_{eq,p} + m_{heli} l_{cm}^2} & 0 \\ 0 & 0 & 0 & -\dfrac{B_y}{J_{eq,y} + m_{heli} l_{cm}^2} \end{bmatrix} x$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \dfrac{K_{pp}}{J_{eq,p} + m_{heli} l_{cm}^2} & \dfrac{K_{py}}{J_{eq,p} + m_{heli} l_{cm}^2} \\ \dfrac{K_{yp}}{J_{eq,y} + m_{heli} l_{cm}^2} & \dfrac{k_{yy}}{J_{eq,y} + m_{heli} l_{cm}^2} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x \tag{4.5}$$

donde K_{pp}, k_{yy}, K_{py}, K_{yp} son la constantes de torque usadas para obtener los torques acoplados que actuan sobre los ejes de cabeceo y guiñado.

Para el modelo de espacio de estados los vectores de entrada u y salida y son $u = \left[V_{m,p}, V_{m,y}\right]^\intercal$ e y = $[x_1, x_2, x_3, x_4]^\intercal$; $V_{m,p}$ es el voltaje de entrada del motor del eje de cabeceo y $V_{m,y}$ es el voltaje del motor del eje de guiñado. Nótese que por la identidad en la matriz de salida todos los estados son medibles. Sin embargo, en realidad sólo las posiciones de los ángulos de cabeceo y guiñado son medidos por los encoders y la velocidad de los ángulos es calculada digitalmente usando la transformada de Laplace:

$$V_f(s) = \frac{\omega_c^2 s}{s^2 + 2\zeta\omega_c + \omega_c^2} X(s) \tag{4.6}$$

donde $X(s)$el la transformada de Laplace de la posición medida, $V_f(s)$es la transformada de Laplace de la derivada de la salida filtrada (es decir, la velocidad filtrada), ω_c es el corte de frecuencia del filtro, y ζ es el amortiguamiento del filtro. Con la finalidad de diseñar un controlador usando la técnica de regulación cuadrática lineal, se asume que todos los estados son medibles.

Diseño del controlador. Dos esquemas de control son utilizados para controlar el vuelo del helicóptero: un FF+LQR (*Feed-Forward Linear Quadratic Regulator*) y un FF+LQR+I (*Feed-Forward Linear Quadratic Regulator Integrator*). El control FF+LQR regula el eje de cabeceo del helicóptero usando prealimentación (FF) y compensadores proporcionales de la velocidad (VP), mientras que la regulación del eje guiñado utiliza solo la VP. El control FF+LQR+I usa un integrador en el lazo retroalimentado para reducir el error en estado estacionario. Usando la FF y la velocidad integral proporcional (PIV) se regula el ángulo de cabeceo y sólo con la PIV se controla el ángulo de guiñado.

En ambos controladores FF+LQR y FF+LQR+I, la posición del ángulo de cabeceo se regula usando el ciclo de prealimentación que compensa el torque gravitacional $\tau_g = m_{heli} g l_{cm} \cos\theta_d$ expresado en las ecuaciones no lineales de movimiento 4.1 y 4.2. El control por prealimentación es:

$$u_{ff} = K_{ff} \frac{m_{heli} g l_{cm} \cos\theta_d}{K_{pp}}$$

donde θ_d es el ángulo de cabeceo deseado y K_{ff} es la ganancia de control de prealimentación. Esto aplica la cantidad de voltaje necesario para hacer flotar al helicóptero en la posición deseada.

Control FF+LQR. El control de la velocidad proporcional $[u_{lqr}, u_{lqr,y}]' = -K \cdot x$ se diseña por medio de la técnica de regulación cuadrática lineal (LQR por *Linear Quadratic Regulator*). En el apéndice B se introduce la forma general del problema LQR.

Usando el modelo lineal en espacio de estados 4.5 y las matrices

$$Q = \begin{bmatrix} 200 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 \\ 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 200 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

el algoritmo de control LQR genera la ganancia de control:

$$k = \begin{bmatrix} 14.1 & 1.33 & 7.33 & 0.924 \\ -1.33 & 14.1 & -0.261 & 7.99 \end{bmatrix}$$

El control FF+LQR que converge $(\theta, \psi, \dot{\theta}, \dot{\psi}) \longrightarrow (\theta_d, \psi_d, 0, 0)$, donde θ_d es el ángulo deseado de cabeceo y ψ_d es el ángulo deseado de guiñado, es:

$$\begin{bmatrix} u_p \\ u_y \end{bmatrix} = \begin{bmatrix} k_{ff} \frac{m_{heli} g l_{cm} cos\theta_d}{K_{pp}} \\ 0 \end{bmatrix} - \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix} \begin{bmatrix} \theta - \theta_d \\ \psi - \psi_d \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Control FF+LQR+I Con la adición de un integrador al control FF+LQR se elimina el error en estado estacionario. Se requiere introducir las ecuaciones de estado $\dot{x}_5 = \theta$ y $\dot{x}_6 = \psi$, de forma que el modelo lineal en espacio de estados 4.5 aumentado queda como sigue:

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-B_p}{J_{eq,p} + m_{heli} l_{cm}^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{-B_y}{J_{eq,y} + m_{heli} l_{cm}^2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} x$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \dfrac{K_{pp}}{J_{eq,p} + m_{heli} l_{cm}^2} & \dfrac{K_{py}}{J_{eq,p} + m_{heli} l_{cm}^2} \\ \dfrac{K_{yp}}{J_{eq,y} + m_{heli} l_{cm}^2} & \dfrac{K_{yy}}{J_{eq,y} + m_{heli} l_{cm}^2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x \tag{4.7}$$

Repitiendo el mismo proceso LQR anterior y usando las matrices:

$$Q = \begin{bmatrix} 200 & 0 & 0 & 0 & 0 & 0 \\ 0 & 150 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 200 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

se obtiene la siguiente ganancia de control:

$$k = \begin{bmatrix} 18.9 & 1.98 & 7.48 & 1.53 & 7.03 & 0.770 \\ -2.22 & 19.4 & -0.450 & 11.9 & -0.770 & 7.03 \end{bmatrix}$$

Así, el controlador FF+LQR+I queda como sigue:

$$\begin{bmatrix} u_p \\ u_y \end{bmatrix} = \begin{bmatrix} k_{ff} \dfrac{m_{heli} g l_{cm} cos\theta_d}{K_{pp}} \\ 0 \end{bmatrix} - \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix} \begin{bmatrix} \theta - \theta_d \\ \psi - \psi_d \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$-\begin{bmatrix} \int k_{15}(\theta - \theta_d) + \int k_{16}(\psi - \psi_d) \\ \int k_{25}(\theta - \theta_d) + \int k_{26}(\psi - \psi_d) \end{bmatrix}$$

La implementación del modelo del helicóptero en *Matlab/Simulink*, los módulos y submódulos que lo componen pueden consultarse en el apéndice D.

La propuesta experimental para expresar este prototipo como un NCS se hace en los capítulos siguientes en los que se plantean los esquemas de reconfiguración dinámica.

go de métodos de diseño de controladores discretos en el tiempo que pueden ser utilizados tales como la asignación de polos y diseños lineales cuadráticos.

4.6 Resumen

En los sistemas de control digital y en los NCS, el periodo de muestreo impacta en la calidad del control. Este efecto sobre el desempeño del sistema es cuantificado por la integral del valor absoluto del error, IAE. Por medio de un ejemplo se muestra que existe un intervalo de los periodos de muestreo en el cual un NCS puede desempeñarse de forma aceptable, fuera de este rango el sistema se degrada. El SDTR elegido como caso de estudio consiste de un prototipo de helicóptero con dos grado de libertad. El modelo dinámico, el modelo lineal y el controlador LQR para helicóptero son expuestos para dar una visión general sobre este caso de estudio.

Capítulo 5

Reconfiguración por Medio de Parámetros de Planificación

Un característica deseable de los NCS es un sistema altamente disponible (sección2.9). Esta propiedad ha sido estudiada consistentemente para poder garantizar un sistema confiable. Algunas estrategias de tolerancia a fallas integran enmascaramiento de fallas y estrategias de detección y aislamiento de fallas (FDI pot *Fault Detection and Isolation*) con la finalidad de construir sistemas altamente seguros y disponibles (Benítez-Pérez y otros, 2007a).La integración de elementos periféricos inteligentes como elementos autónomos capaces de auto-diagnóstico (self-diagnosis), permite la reconfiguración de las estrategias de control, usando la información de cada uno de éstos para llevar a cabo un proceso de decisión. Para Benítez-Pérez y García-Nocetti (2003), la tolerancia a fallas por medio de sensores inteligentes en un sistema distribuido necesita de un planificador con la finalidad de tomar en cuenta el comportamiento del sistema bajo escenarios libres de falla y escenarios con falla. Este planificador estático tiene la capacidad de llevar a cabo la reconfiguración en línea para ambos escenarios.

En un NCS, en el que sensores, actuadores y controladores están conectados por un medio de comunicación compartido donde se tiene un constante intercambio de datos, los sistemas tradicionales de control tolerante a fallas (CTF) limitan las capacidades de expansión y reconfiguración del control. Los sistemas de CTF centralizados pueden conducir a un punto de suspensión el momento que se dé lugar una falla. Una propuesta para el manejo de fallas en un SDTR es utilizar la reconfiguración de los parámetros de planificación con base en una arquitectura multiagente distribuida que utiliza sensores inteligentes. Esto con la finalidad de definir el tipo de reconfiguración de acuerdo al escenario particular de falla. En esta sección se expone el trabajo relacionado con esta propuesta de reconfiguración, las capacidades de los agentes, el acuerdo y la forma de construirlo entre agentes y la arquitectura propuesta.

5.1 Trabajo relacionado

Menéndez y Benítez-Pérez (2010) plantean una estrategia de planificación de un SDTR en la que determinan el *periodo base de muestreo* del sistema ϱ, con el cual se obtienen los *periodos operacionales* de todos los nodos que participan en el NCS. Aunque todos los nodos pueden usar el periodo base como su periodo de operación, la estrategia plantea la opción de tener diferentes periodos para cada nodo por medio del periodo base y un *factor de dispersión* λ que funciona como el porcentaje de desfasamiento entre las tareas que se ejecutan en un nodo respecto al periodo base. El periodo base es el periodo operacional para las tareas

periódicas del controlador en el agente de control correspondiente. El periodo base y el factor de dispersión son usados para obtener el periodo de muestreo actual de cuatro sensores de acuerdo a las siguientes ecuaciones:

$$p_{s1} = \rho(1 + \lambda)$$
$$p_{s2} = \rho(1 - \lambda)$$
$$p_{s3} = \rho(1 + 1.1\lambda)$$
$$p_{s4} = \rho(1 - 1.1\lambda)$$

Menéndez y Benítez-Pérez (2010) utilizan como caso de estudio el prototipo de helicóptero Quanser y calculan el valor de los rangos del valor de dispersión desde 0% hasta 20%. Para el valor de 0% los cuatro sensores tienen el mismo periodo que el controlador, cuando la dispersión es 20%, los sensores tienen periodos de operación con 20% y 22% arriba y abajo del valor del periodo base. Para valores de la dispersión mayores a 25% se experimenta pérdida de desempeño en el sistema. Los resultados que se presentan utilizan el valor del IAE como índice de desempeño del sistema.

Menéndez y Benítez-Pérez (2010) elaboran la planificación de la siguiente forma: el nodo planificador ejecuta una tarea periódica con periodo p_{sch}, el nodo que elige para asignarle una ventana de tiempo es elegido siguiendo una distribución de probabilidad. El identificador del nodo (*id*) elegido por el planificador se escribe en una memoria compartida, los nodos listos para transmitir verifican este valor, si en la memoria compartida se encuentra escrito su *id* el nodo transmite, de lo contrario espera uno o más instantes de tiempo p_{sch}. En la figura 5.2 se muestra un ejemplo de la lectura de cuatro sensores de las señales $(\theta_0 = 0, \psi_0 = 0, \dot{\theta}_0 = 0, \dot{\psi}_0 = 0)$ correspondientes al ángulo de cabeceo, ángulo de guiñada, variación del ángulo de cabeceo y variación del ángulo de guiñada, y la transmisión de estos datos hacia el controlador supervisada por el nodo planificador. Los bloques marcados como *S*1, *S*2, *S*3 y *S*4 representan los agentes sensores que envían los datos de su lectura hacia el agente controlador *ctrl* bajo la supervisión del agente planificador *sch* que asigna la tasa de transmisión de los datos hacia el controlador.

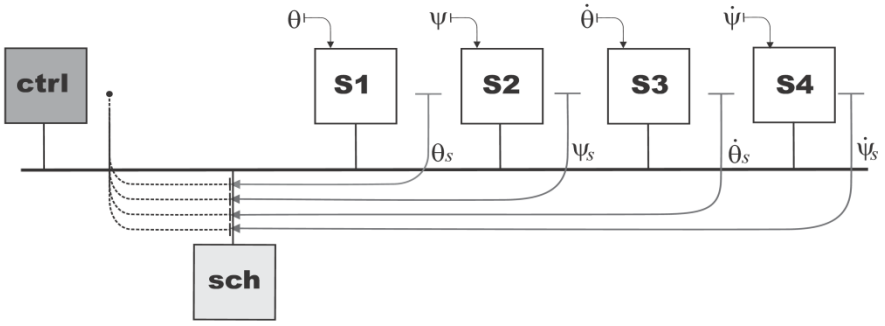

Figura 5.1: Comunicación sensores-controlador supervisada por el planicador.

Bajo esta estrategia de planificación, el agente controlador obtiene 1/3 de las asignaciones de ancho de banda, mientras que el resto de los nodos comparten el restante 2/3. Para Menéndez y Benítez-Pérez (2010) este procedimiento es factible debido que asumen al SDTR como un proceso dinámico en el cual los retardos son acotados e invariantes en el tiempo. Fuera de línea, los autores realizan el análisis de elección de periodos de transmisión adecuados (sección 4.1) para determinar parámetros de planificación específicos que no generen retardos muy grandes. En la figura 5.2 se muestra la actividad que tiene la red para transmisiones de sensores y controlador.

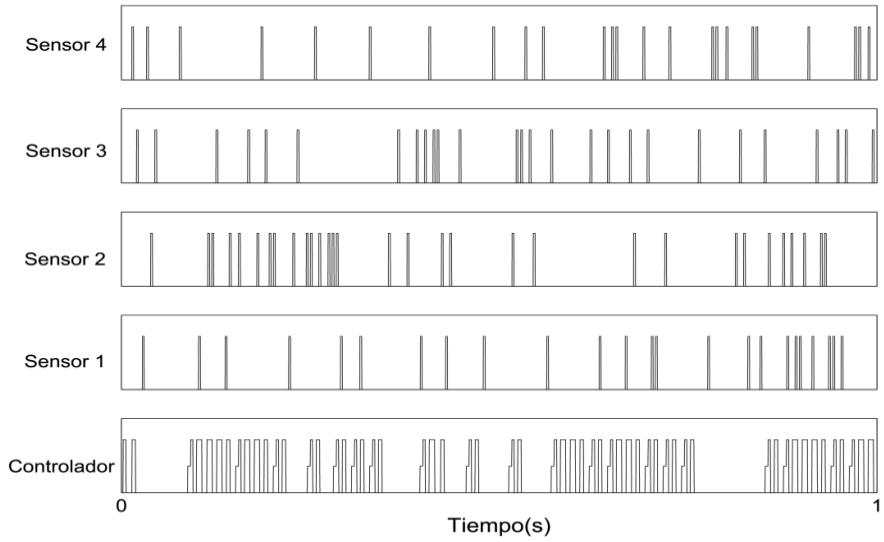

Figura 5.2: Envío de mensajes de sensores y controlador a través de la red en un RTDS.

5.2 Propuesta de reconfiguración

El manejo de redundancia de sensores basados en consolidación de señales y enmascaramiento de fallas (ver sección 3.4) es una tecnología utilizada y probada en el control de vuelo de aeronaves (Oosterom, 2005). Recientemente, la implementación de la redundancia se realiza utilizando dispositivos inteligentes definidos como elementos periféricos que poseen capacidades de autodiagnóstico de fallas, autocompensación de perturbaciones y de comunicación digital (Benítez-Pérez y García-Nocetti, 2003, 2005). Estos dispositivos inteligentes son básicamente sensores y actuadores.

En la estrategia de planificación que se plantea en este trabajo, se propone utilizar dispositivos inteligentes con el propósito de implementar una arquitectura multiagente donde la comunicación entre entidades genera intercambio de información para la toma de decisiones descentralizadas. Cada agente podrá percibir su entorno y cooperar con los demás agentes. Los dispositivos inteligentes son entidades de hardware que potencian el software y que se utilizaremos para realizar procesos de decisión. Actualmente el uso de elementos inteligentes permite integrar funciones de control (Benítez-Pérez y García-Nocetti, 2003, 2005). Cada dispositivo inteligente será un *agente del sistema*. Se desea que los agentes obtengan y manejan información de control, perciban eventos en su entorno y reaccionen ante estos durante la ejecución del sistema. El propósito consiste en que los agentes puedan comunicarse

y colaborar entre sí para decidir un plan de reconfiguración basado en los parámetros de planificación.

En el contexto del modelo de reconfiguración planteado en la sección 2.2, se propone que cada sensor funcione como un generador de información de reconfiguración, en lugar de ser agentes que detecten e identifiquen fallas en el sistema. Asumiremos que los sensores están dotados de un mecanismo para detectar e identificar la falla y sólo nos enfocaremos en el manejo de la falla por medio de la reconfiguración. Con la información de falla que cada agente sensor posee, por medio de un acuerdo se decide el tipo de reconfiguración. Cada esquema de reconfiguración que se utiliza ante ciertas fallas en particular se basa en el análisis fuera de línea. El conjunto de toda la información local conforma la visión global de los estados del sistema.

El sistema de reconfiguración está basado en un agente planificador el cual modifica los parámetros de planificación basados en el manejo la asignación del ancho de banda de la red compartida. Con base en el esquema de reconfiguración propuesto por Lunze y Steen (2002); Lunze y otros (2003) los pasos de la reconfiguración basada en agentes son:

- Detección e identificación de la falla.
- Compartir información para acordar el modelo de reconfiguración.
- Encontrar la secuencia de acciones de reconfiguración que conduzca al sistema a nuevos estados de equlibiro.
- Ejecutar la reconfiguración (acotada en tiempo) para reestabilizar la planta a un nuevo equilibrio.

En la figura 5.3 se muestra el proceso de reconfiguración donde cada agente individual interactúa con los demás agentes, tanto en su acción de control como en su tarea de reconfiguración. La visión local de cada agente sobre lo que sucede al sistema se comparte con otros agentes de tal manera que se integra un conocimiento global de su funcionamiento.

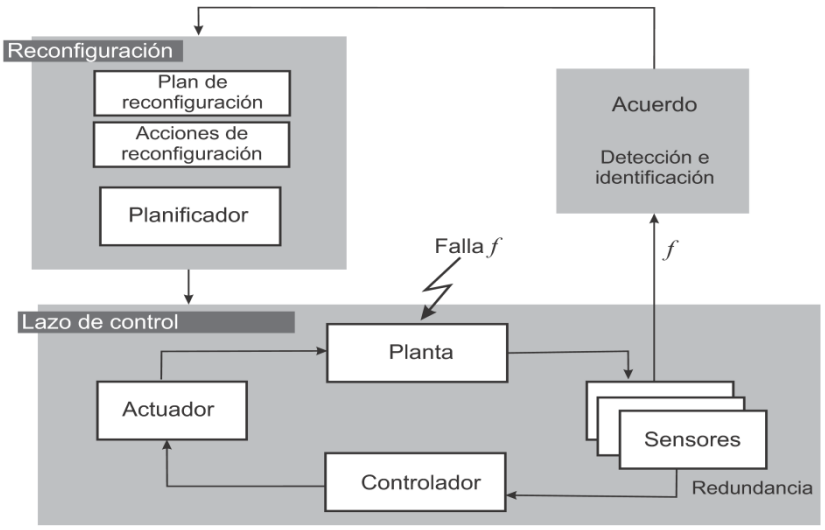

Figura 5.3: Reconfiguración distribuida entre agentes

Consideremos un NCS con redundancia de sensores. Cada nodo es un agente que realiza alguna de la tareas del lazo retroalimentado: control, sensado o actuación. Para m señales de los estados del sistema se tienen $B_1, B_2, B_3, .., B_m$ bloques de sensado con r_i agentes sensores cada uno. Denotaremos como $a(i,j)$al agente j del bloque i que lleva acabo la lectura de la señal x_{ij} para $i \in (1, \dots, m), j \in (1, \dots, r_i)$.

El mecanismo de tolerancia a fallas consiste en utilizar triple redundancia de sensores (Latif-Shabgahi y otros, 2003; Latif-Shabgahi, 2004), con la finalidad de enmascarar la falla por medio de un algoritmo de voto que arbitre entre fallas producidas en los sensores. Con base en la arquitectura multiagente descrita anteriormente (sección 3.5), la conjunción de sensores redundantes permite confederar grupos de sensores que toman la lectura de los estados del sistema y funcionan como agentes de identificación de falla (a_{fi}). El agente planificador funciona como el agente de reconfiguración (a_{fr}) que ejecuta la modificación de los parámetros de planificación basado en el acuerdo previo. Cada agente a_{fi} puede desempeñarse con las siguientes acciones (figura 5.4):

- Leer las señales de los estados del sistema.
- Identificar las fallas en el sistema.
- Comunicarse con otros agentes.
- Acordar el tipo de reconfiguración conforme a la falla presentada.

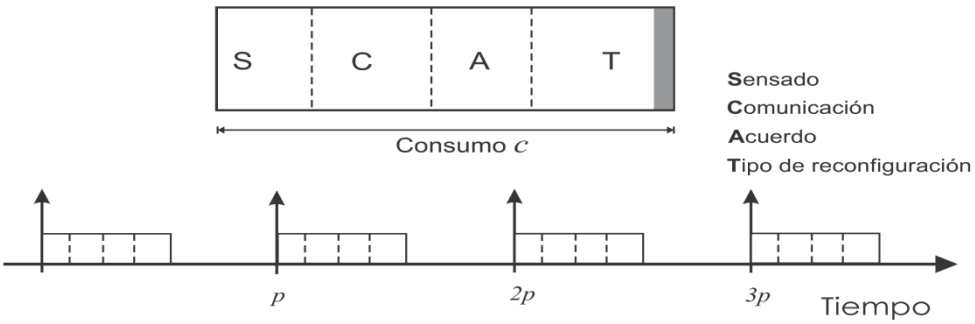

Figura 5.4: Acciones definidas para los agentes sensores.

Cada agente sensor $a_{i,j}$ ejecuta una tarea periódica con periodo p_i y tiempo de ejecución o consumo c_i, que cuenta con las acciones arriba descritas. Para cada np $(n = 0,1,2,3, \dots)$periodos de muestreo, un valor de la señal medida $x_{ij}(n)$ y un valor $f_{ij}(n)$ que tipifica la posible falla. Cabe señalar que la detección e identificación de fallas está fuera del alcance de este trabajo; sin embargo, se asume que el sistema está provisto de un mecanismo que permite detectar, identificar y tipificar las fallas que se presentan en la lectura de datos (sección 3.4). Con el conocimiento previo de los efectos de fallas en sensores, es posible diseñar un plan de reconfiguración para cada una de las fallas que pueda presentarse.

Con la información (x_{ij}, f_{ij}) que cada agente a_{ij} posee, se lleva a cabo el proceso de reconfiguración con base en un acuerdo entre agentes. Para hacer este acuerdo, es necesario que el intercambio de información se realice antes de la finalización del plazo de la tarea; no obstante, no siempre es posible esto. Dicho acuerdo demanda recursos de la red, lo que conduce al aumento del tráfico, saturación y pérdida de datos. Esto se traduce en información incompleta para realizar el acuerdo (Lynch, 1996) y la reconfiguración. Para lograr la reconfiguración de los periodos de tareas y la realización del acuerdo entre agentes, se propone resolver la pérdida de datos de la siguiente manera:

- *Modificación de periodos para cada bloque de sensores*. La modificación de periodos en cada bloque de sensores intenta evitar colisiones y saturación de la red cada periodo de muestreo. El cambio de periodos de todos los elementos del sistema distribuido se realiza por medio de la modificación de los parámetros ϱ y λ. Este planteamiento es propuesto inicialmente en Menéndez y Benítez-Pérez (2010)y se retoma en este trabajo como estrategia de reconfiguración.

- *Reconstrucción de datos*. Es inevitable que en algún momento exista una latencia grande en la entrega de datos, lo que genera pérdida de desempeño. Se propone que la información no disponible sea reconstruida por medio de un valor por defecto definido previamente (un valor estimado producto de la experiencia), o bien, por un valor histórico, que podría ser el promedio de todos los valores registrados previamente.

5.3 Acuerdo

Como se ha mencionado, la redundancia de sensores aumenta la confiabilidad en las mediciones. Se desea tener un alto grado de seguridad en la lectura, de tal forma que ante cualquier tipo de falla en los sensores se activa el enmascaramiento de falla por medio de un algoritmo de voto. La información de los sensores representa el estado dinámico del sistema en cada tiempo *t*. Bajo un escenario con falla, la información local que cada sensor percibe se comparte con los otros agentes para llegar un acuerdo sobre la lectura consolidada y que la reconfiguración que se lleve a cabo. La información de la señal medida y la tipificación de falla (x_{ij}, f_{ij}) de cada agente sensor se intercambia con los *pares* del mismo bloque de sensado. Cada agente a_{ij} cuenta entonces con la tupla de datos $y_{ij} = \{(x_{i1}, f_{i1}), (x_{i2}, f_{i2}), (x_{i3}, f_{i3}), \dots, (x_{ij}, f_{ij})\}$ donde $i = 1, \dots, m$, y $j = 1,2,3$.

Con esta tupla de valores, cada agente realiza el acuerdo $A(y_{ij})$en colaboración con los otros agentes de su mismo bloque. Cada agente *j* del bloque de lectura *i* posee la lectura de un estado del sistema en el tiempo t_1: la señal medida x_{ij} y la tipificación de falla f_{ij}. En el tiempo t_2 el agente ha intercambiado información local con los otros dos agentes de su mismo bloque. Al paso de un tiempo t_3 cada agente ha calculado el valor $A(y_{ij})$ del acuerdo (figura 5.5).

Tómese el acuerdo $A(y_{ij})$ como una función que depende de la tupla de valores y_{ij}. Esta función devuelve el valor *acordado* de la señal medida consolidada $\bar{x}$ basándose en la tipificación de la falla que experimenta algún sensor en el tiempo t_k. Este valor acordado será el que prevalezca para ser enviado al controlador. Dados *m* boloques de sensado con triple redundancia de sensores, se propone la función de acuerdo $A(y_{ij})$ para $i = 1, \dots, m$, y $j = 1,2,3$.de la siguiente forma:

$$A(y_{ij}) = \begin{cases} \bar{x} = x_{i1,} & f_{i1} = f_{i2} = f_{i3} \\ \bar{x} = \dfrac{(x_{i1} + x_{i2})}{2}, & f_{i1} = f_{i2} \neq f_{i3} \\ \bar{x} = \dfrac{(x_{i1} + x_{i3})}{2} & f_{i1} = f_{i3} \neq f_{i2} \\ \bar{x} = \dfrac{(x_{i2} + x_{i3})}{2}, & f_{i1} \neq f_{i2} = f_{i3} \\ \bar{x} = x_{i1}, & f_{i1} \neq f_{i2} \neq f_{i3} \end{cases} \quad (5.1)$$

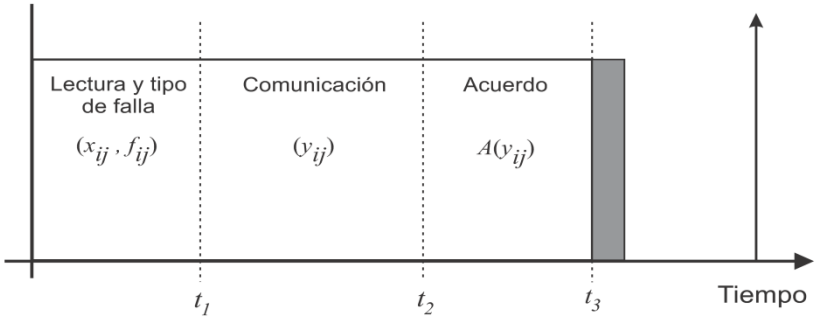

Figura 5.5: Información del agente para realizar el acuerdo

Esta función de acuerdo se interpreta de la siguiente manera:

- Si un agente del bloque *i* ha recibido de los otros dos agentes de su mismo bloque una tipificación de falla igual a la propia, el valor de la lectura $\bar{x}$ será valor local sensado por el agente a_{ij} que hace la llamada al acuerdo.

- Si el agente recibe de los otros dos agentes una tipificación de falla igual a la suya y otra distinta, se descarta el valor que difiere y la lectura acordada $\bar{x}$ es el promedio de los datos sensados con igual tipificación de falla.

- Si el agente recibe dos tipificaciones de falla diferentes a la suya pero iguales entre sí, se toma el promedio de los datos sensados con igual tipificación.

- Si el agente recibe dos valores de tipificación de falla distintos al suyo y aún diferentes entre sí, entonces la lectura acordada es el dato local del agente a_{ij}, al que se asigna arbitrariamente el índice *i*.

Este acuerdo lo hace cada uno de los agentes que sensan los estados del sistema para dar paso a la reconfiguración del sistema. En este trabajo nos enfocamos en las fallas que se presentan en el sensado de datos, debido a que en un sistema con alto grado de seguridad y confiabilidad es importante responder a fallas en los sensores o en los datos que estos registran. Las fallas que se consideran aquí son fallas de lectura como ruido, corrimiento y retraso. Las pruebas de reconfiguración fueron realizadas induciendo una falla a un sensor de un bloque de sensado en cualquier momento de la ejecución del sistema.

5.4 Análisis de planificación

Cada uno de los agentes del sistema de control distribuido realiza un conjunto de tareas $\tau_i = \{\tau_1, \tau_2, \dots, \tau_m\}$ en el que cada tarea τ_k realiza alguna acción del lazo de control retroalimentado. Las demás tareas son tareas adicionales del sistema y son relevantes en cuanto al tiempo de procesamiento que requieren.

Las tareas periódicas que ejecuta cada agente son planificadas por medio de un algoritmo por prioridades. En particular, si cada agente realiza *n* tareas periódicas con periodo p_i y tiempo de ejecución c_i con $i = 1,2, \dots, n$; la planificación de las tareas que se llevan a cabo en cada agente se garantiza si se cumple (Liu y Layland, 1973):

$$U < n\left(2^{\frac{1}{n}} - 1\right) < 1 \tag{5.2}$$

donde la utilización del procesador (agente) está definida como:

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i}$$

El cálculo de la cota de utilización para RM (ecuación 5.2) puede consultarse en el apéndice A. La planificación local de cada agente está en función de los periodos de las tareas y sus tiempos de ejecución. En este este trabajo, la política de planificación utilizada es el algoritmo RM que es un algoritmo de prioridades fijas (sección 2.7). Para cualquier conjunto arbitrario de tareas periódicas, el mínimo valor máximo de utilización de procesador para el cual el conjunto de tareas es planificable es 1. Esta cota decrece mientras *n* aumenta, de tal forma que para valores grandes, la mínima cota máxima converge a ln(2)≈0.69. Así, la utilización de procesador debe ser menor al 69% para asegurar la planificación de cualquier conjunto arbitrario de tareas.

El lapso de tiempo que transcurre para completar el ciclo de control retroalimentado para este caso en particular es:

$$T = nt_s + t_{cm}^{sc} + t_c + t_{cm}^{ca} + t_a \tag{5.3}$$

donde t_{cm}^{sc} el tiempo de comunicación entre sensor y controlador, t_c es el tiempo del controlador, el t_{cm}^{ca} tiempo de comunicación entre controlador y actuador, t_a es el tiempo del actuador (figura 5.6)

Cuando se incluye el enmascaramiento de fallas el tiempo para completar el ciclo de control retroalimentado es:

$$T = nt_s + t_{cm}^{sc} + t_{cm}^{fst} + t_{cm}^{fsc} + t_c + t_{cm}^{ca} + t_a \tag{5.4}$$

donde t_{cm}^{fst} es el tiempo de comunicación entre los sensores y la unidad de enmascaramiento y t_{cm}^{fsc} es el tiempo entre la falla y el consenso. En la figura 5.7 se muestra el diagrama de tiempo para las tareas de control, la comunicación y el proceso de voto.

En ambos casos, ecuaciones 5.3, 5.4, el valor de *T* debe ser acotado y conocido para que la reconfiguración sea planificable. El proceso de reconfiguración consiste en modificar los

periodos de tareas y consumos cuando suceda un evento que altere la condición de planificación para lo cual se calculan los nuevos periodos de tareas.

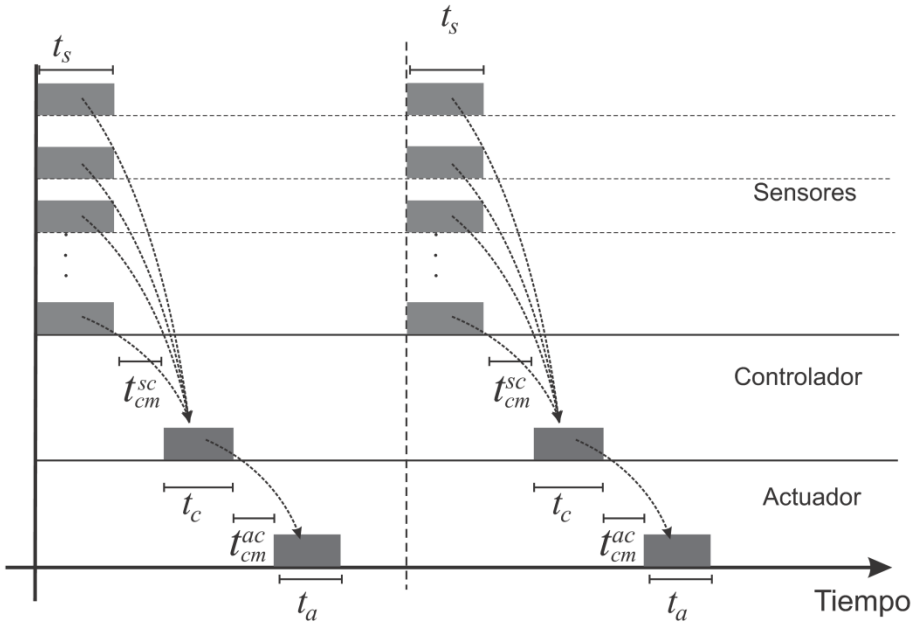

Figura 5.6: Diagrama de tiempo de las tareas de control y comunicación.

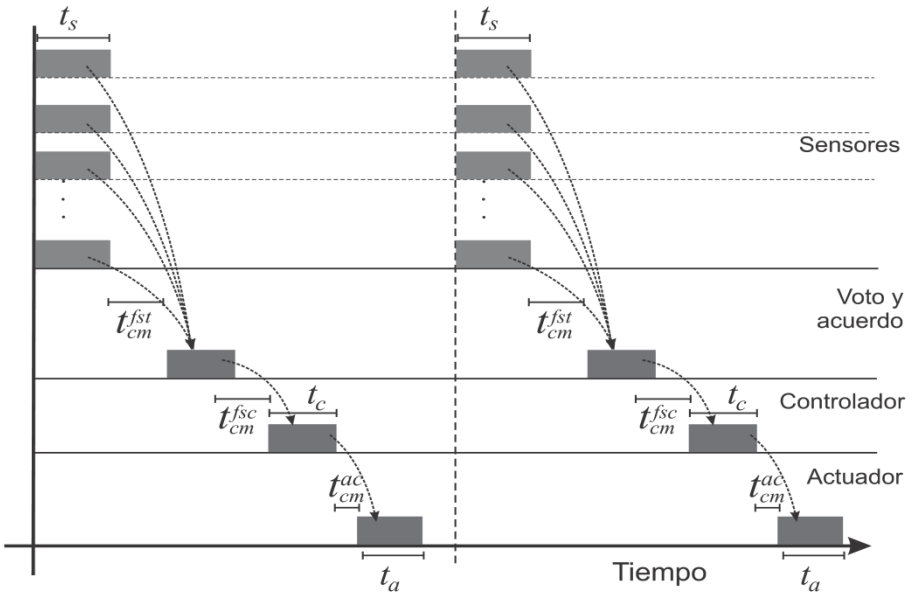

Figura 5.7: Diagrama de tiempo de las tareas de control, comunicación, voto y acuerdo.

Se busca la planificación de un conjunto de tareas por medio de la obtención de un conjunto factible de periodos que aseguran la restricción (5.2) bajo un algoritmo de planificación de prioridades fijas como el RM. La obtención de este conjunto de periodos depende de un conjunto adicional de periodos que sirve como cota máxima, que puede tomarse de diferente manera de acuerdo al caso de estudio. No obstante que la búsqueda de este nuevo conjunto de periodos puede ser computacionalmente demandante, al tratarse como

un problema de programación entera disminuye el tiempo de cálculo. El problema a resolver es asegurar que el efecto de la reconfiguración no impacte al sistema; es decir, que cumpla con los plazos de tareas o bien se tenga una mínima tasa de pérdida de plazos. Dicha reconfiguración ayuda a disminuir los efectos no lineales del sistema cuando aparecen un número considerable de retardos de tiempo, producto de estos escenarios.

La propuesta de reconfiguración en el sentido de la planificación consiste en que dado un evento que modifique la planificabilidad local de algún agente del sistema:

- Calcular un nuevo conjunto de periodos para retomar la planificabilidad
- Realizar el cambio en los periodos correspondientes
- Mantener la planificabilidad global

Es importante mencionar que el ajuste en los periodos de muestreo provocará un retardo menor en el desempeño del sistema que es relevante mostrar, así mismo se induce un retardo adicional producto de la reconfiguración:

$$T_r = t_{cp}^r + t_{rp}^r \tag{5.5}$$

donde t_{cp}^r es el tiempo del cálculo de nuevos periodos y t_{rp}^r el tiempo necesario para cambiar los periodos.

La planificación local no se realiza por medio de un acuerdo entre agentes, sin embargo, es necesario que ésta se asegure para lograr una efecto positivo de la reconfiguración y el tipo se toma por medio del acuerdo.

5.5 Resumen

En este capítulo se describió el trabajo relacionado con la planificación de la red, con base en la modificación de periodos de muestreo en un SDTR. Se propuso una estrategia de reconfiguración dinámica con base en el ajuste de parámetros planificación. La implementación de esta estrategia utiliza redundancia de sensores y dispositivos inteligentes que soportan las propiedades de los agentes: percepción de su entorno, comunicación y negociación para lograr acuerdos. Bajo escenarios con falla en la lectura de sensores, se toman acuerdos entre agentes para consolidar la señal medida y determinar el tipo de reconfiguración. La reconfiguración que se planteó incluye el uso de enmascaramiento de falla y modificación de los parámetros de planificación. Se hizo énfasis en mostrar que las transiciones provocadas por la reconfiguración propuesta no afectan los plazos del ciclo de control.

Capítulo 6

Reconfiguración con Base en el Cambio de Frecuencias de Transmisión

En los sistemas de control distribuido, los lazos de control retroalimentado se cierran a través de un medio de comunicación común. El desempeño de los lazos de control dependen no sólo del diseño del algoritmo de control, también es influido por la planificación del medio de red compartido (Seung, 1995; Cervin, 1999; Cervin y Eker, 2000; Mendez-Monroy y otros, 2009). No obstante que altas tasas de sensado mejoran el desempeño en los sistemas tradicionales de control por computadora, esto puede inducir altas cargas de tráfico en el medio de comunicación en los NCS (Lian y otros, 2001a).. En redes de ancho de banda limitada, cargas de tráfico altas incrementan los retardos inducidos por la red y esto puede degradar el desempeño de control. Por lo tanto, encontrar un periodo de muestreo certero que acote o disminuya el retardo inducido por la red es un elemento importarte a considerar en las etapas de análisis y diseño de los NCS.

Se han propuesto distintas estrategias para asegurar el desempeño de control ante la presencia de retardos distintas estrategias para asegurar el desempe~no de control ante la presencia de retardos (Halevi y Ray, 1988a,b; Méndez-Monroy y Benítez-Pérez, 2009; Sename y otros, 2003; Yu y otros, 2005); sin embargo, éstas se centran en la naturaleza de los retardos (Nilsson, 1998; Tipsuwan y Chow, 2003) y el control propio del retardo (Lincoln, 2003; Zhang y Yu, 2008). La planificación de un NCS (Albertos y otros, 2000; Seto y otros, 1996; Walsh y Ye, 2001) se ha estudiado extensamente y varios algoritmos se han propuesto para aprovechar al máximo los recursos limitados (Xia y otros, 2004b,a; Menéndez y Benítez-Pérez, 2010) y minimizar el efecto de los retardos. Si nos enfocamos únicamente en el manejo de los periodos de muestreo de un NCS, recientemente se han propuesto métodos para balancear los periodos de muestreo y la cantidad de datos que pueden ser transmitidos por la red. Peng y otros (2008, 2009) modifican la frecuencia de transmisión de datos f_i, de un nodo cualquiera i, para o equilibrar la tasa de envío de datos por la red, con la finalidad de obtener un desempeño deseado. Esta estrategia se enfoca en la optimización de la frecuencia en los términos expuestos, descarta el control de los retardos y no utiliza alguna técnica de control reconfigurable para manejar su efecto.

En este capítulo se propone una estrategia de reconfiguración que considere el desempeño de control y la planificación de la red por igual, con base en la modificación de las frecuencias de transmisión de los agentes de un NCS. El capítulo inicia con un ejemplo que motiva al análisis del impacto de la frecuencia de transmisión en el desempeño de control, expone las consideraciones para modelar el sistema de frecuencias de transmisión como un sistema lineal

invariante en el tiempo y presenta el análisis de planificación para el tipo de reconfiguración propuesta. Finalmente, se presenta el modelo completo de cambio de frecuencias de transmisión. Este sistema de frecuencias de transmisión datos se asume como un subsistema que puede ser controlable dentro de una región acotada por las frecuencias de transmisión máximas y mínimas en las que el sistema es planificable y por tanto se mantiene un nivel de desempeño deseado.

6.1 Región de planificabilidad

Con base en la idea de manejar la cantidad de datos que se envían a través de la red, se utilizará la frecuencia de transmisión como parámetro, esto facilita el análisis para la elección de periodos de tareas. Consideremos la *frecuencia de transmisión* como la relación inversa del periodo p_i de las tareas periódicas que transmiten datos por la red, es decir, $f_i = {}^{i}/_{p_i}$ Con base en el análisis del efecto de los periodos de muestreo de un NCS en el desempeño de control (sección 4.3), se desea encontrar frecuencias de transmisión de datos acotadas que utilicen la red de manera que no saturen el medio de comunicación ni provoquen submuestreo. Se propone controlar un conjunto de frecuencias de trasmisión de datos , por medio del vector de entrada *u*(*t*) de tal forma que las salidas *y*(*t*) estén en una región donde el conjunto de frecuencias cumpla con:

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} < \sum_{i=1}^{n} c_i f_i \leq 1 \tag{6.1}$$

Esto es que durante la evolución en el tiempo del NCS, las frecuencias de transmisión de datos son estabilizadas, por medio de un controlador, dentro una región en la que la relación de periodos y consumos (expresado en términos de la frecuencia de transmisión), de tal forma no sobrepase la carga máxima de datos que puede transmitir la red.

De acuerdo a la configuración del caso de estudio, es posible encontrar conjuntos de sensores que tomen lecturas a diferentes frecuencias. Así, cada sensor tiene una cota superior de frecuencia de transmisión de datos delimitada por el ancho de banda y una cota inferior de transmisión de datos impuesta por los requerimientos de control [1]. Por lo anterior, un NCS con redundancia de sensores, permite determinar diferentes regiones de frecuencia de transmisión de datos dentro de las cuales cada nodo puede transmitir. Las regiones a las que están restringidas las frecuencias de transmisión de datos aseguran el óptimo desempeño de control respecto al error de salida (evaluado por el IAE). A estas regiones les llamaremos *regiones de planificabilidad del NCS* y las denotaremos como $\mathcal{L}_i$.

De esta forma, para cada agente sensor *i* que transmite datos por la red con una frecuencia de transmisión f_r^i, existe una región $\mathcal{L}_i$ acotada superior e inferiormente por f_x^i y f_m^i respectivamente, donde *i*=1,2,⋯,*k*. Estas regiones de planificabilidad pueden ser expresadas de la siguiente manera:

[1] Ver sección **¡Error! No se encuentra el origen de la referencia.**

$$\mathcal{L}_1 = \{(x_1, y_1 | x_1 \in \mathbb{R}^+, f_m^1 \leq y_1 \leq f_x^1)\}$$
$$\mathcal{L}_2 = \{(x_2, y_2 | x_2 \in \mathbb{R}^+, f_m^2 \leq y_2 \leq f_x^2)\}$$
$$\vdots$$
$$\mathcal{L}_k = \{(x_k, y_k | x_k \in \mathbb{R}^+, f_m^k \leq y_k \leq f_x^k)\}$$

En esta representación, no es necesario que $f_x^k \leq f_x^{k-1} \leq \cdots \leq f_x^2 \leq f_x^1$ o. $f_m^k \leq f_m^{k-1} \leq \cdots \leq f_m^2 \leq f_m^1$ En la figura 6.1 se muestran tres regiones de transmisión acotadas superior e inferiormente por f_x^i y f_m^i:

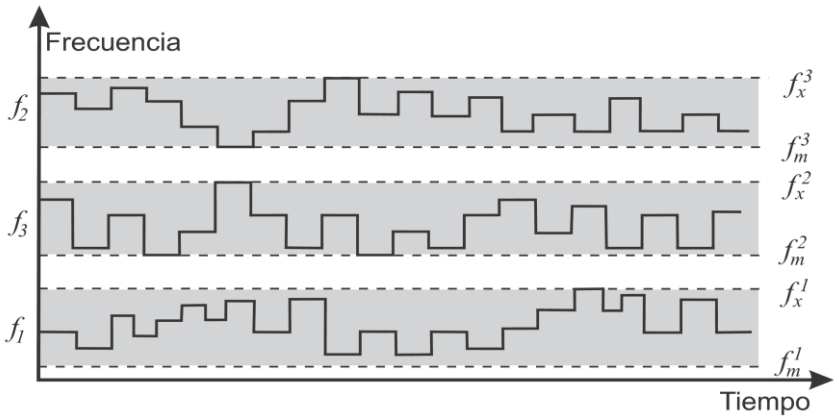

Figura 6.1: Regiones de frecuencias de transmisión acotadas por frecuencias máximas f_x^i y frecuencias mínimas f_m^i.

La planificabilidad puede ser por rangos sobre sensores en regiones separadas que garantizan de manera local la región de acción en su frecuencia de transmisión de datos. Como ya se mencionó, un caso particular es que la región de transmisión de datos sea común para todos los nodos; otro caso consistiría en que las regiones sean continuas. Cabe mencionar que en los límites de las regiones puede experimentar bajo muestreo o tráfico en la red.

En la figura 6.2 se puede ver la dinámica de un conjunto de frecuencias de transmisión f_1, f_2, f_3 y el efecto deseado al controlarlas dentro de sus regiones correspondientes $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$.

Sin pérdida de generalidad, se puede asumir el caso en el que todos los sensores compartan una región común. En esta región la tasa de transmisión de datos de todos los sensores es la óptima para lograr el desempeño deseado. En la figura 6.3 se muestra este caso.

La reconfiguración del NCS puede obtenerse al modificar el periodo p_i de cada una de las tareas periódicas en los sensores utilizadas para transmitir datos a través de la red de comunicación común. El periodo modificado estará dentro de una región de planificabilidad que optimiza el desempeño de control. En la figura 6.4 se muestra la modificación de los periodos de tareas de tal forma que ninguna tarea pierda su plazo y equilibre la tasa de transmisión de datos por la red.

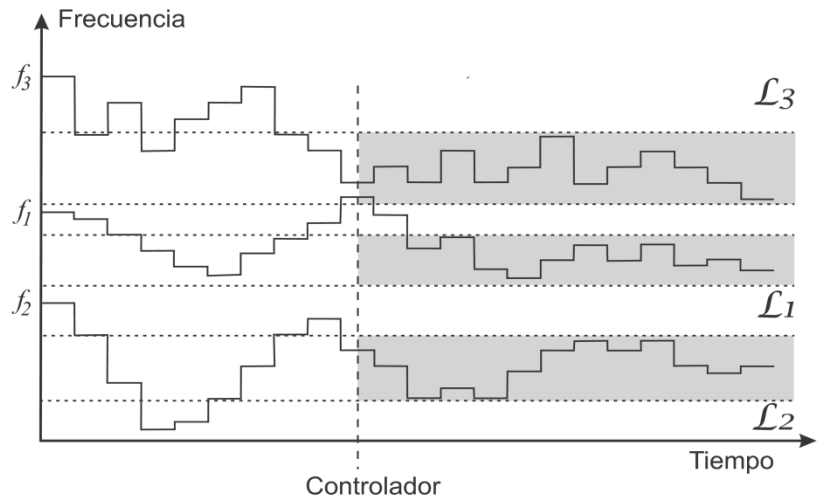

Figura 6.2: Regiones de planificabilidad donde convergen las diferentes frecuencias de transmisión

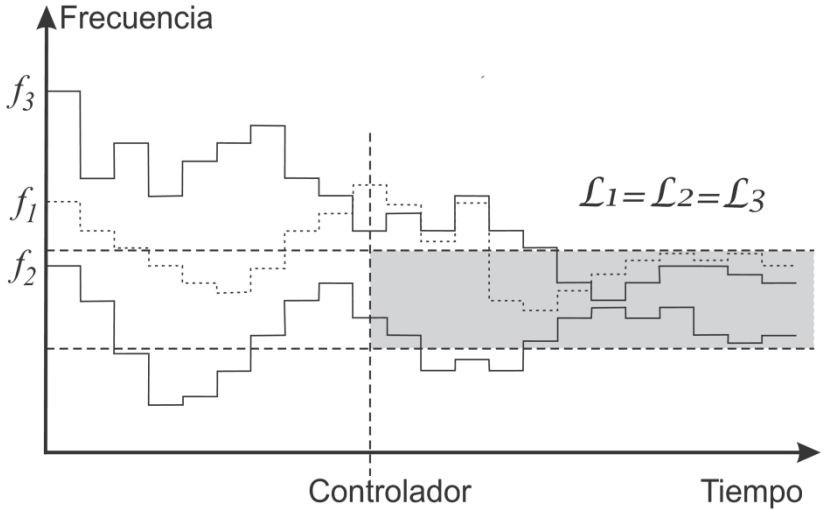

Figura 6.3: Región de planificabilidad común a la que convergen las tres diferentes frecuencias de transmisión.

6.2 Modelo del sistema de frecuencias de transmisión

La idea de modelar un sistema dinámico consiste en tener una representación de la realidad, específicamente un sistema en el contexto matemático es un dispositivo que acepta una señal de entrada y produce una señal de salida. Los sistemas continuos en el tiempo pueden ser descritos de manera lo suficientemente precisa por medio de ecuaciones diferenciales ordinarias. Es posible describir estos sistemas usando espacio de estados o como una ecuación diferencial ordinaria que relaciona la entrada y la salida. La forma general del modelo en espacio de estados de un NCS es:

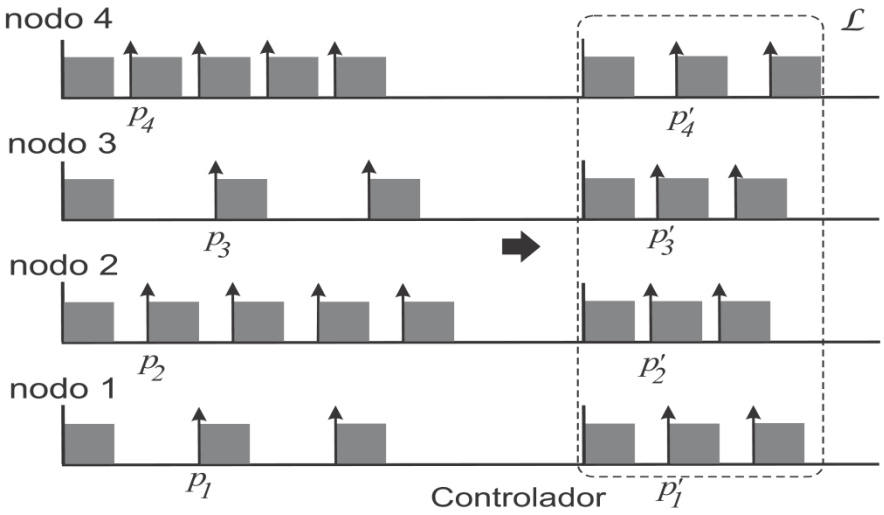

Figura 6.4: Cambio de periodos de sensado modificados por un controlador dentro de una región de planificabilidad *L*.

$$\begin{aligned}\dot{x}_1(t) &= h_1(x_1, x_2, x_3, \ldots . x_k, u_1, u_2, u_3, \ldots, u_l, t)\\ \dot{x}_2(t) &= h_2(x_1, x_2, x_3, \ldots . x_k, u_1, u_2, u_3, \ldots, u_l, t)\\ &\ldots\\ \dot{x}_k(t) &= h_k(x_1, x_2, x_3, \ldots . x_k, u_1, u_2, u_3, \ldots, u_l, t)\end{aligned} \tag{6.2}$$

donde $\dot{x}(t)$ denota la la evolución en el tiempo del vector de estados del sistema con dimensión *k*, *u* es el vector de la señal de entradas de dimensión *l*.

Para el caso invariante en el tiempo $h(\cdot,\cdot)$ y linealizando (6.2) en un rango de operación, se tiene:

$$\begin{aligned}\dot{x}_1(t) &= a_{11}x_1 + \ldots + a_{1k}x_k, b_{11}u_1 + \ldots + b_{1l}\,u_l\\ \dot{x}_2(t) &= a_{21}x_1 + \ldots + a_{2k}x_k, b_{21}u_1 + \ldots + b_{2l}\,u_l\\ &\ldots\\ \dot{x}_k(t) &= a_{k1}x_1 + \ldots + a_{kk}x_k, b_{k1}u_1 + \ldots + b_{kl}\,u_l\end{aligned} \tag{6.3}$$

Para el conjunto de funciones h_i invariantes en el tiempo, asumimos que se asegura la existencia y unicidad de sus soluciones. La forma en espacio de estados del sistema (6.3) de múltiples entradas - múltiples salidas (MIMO por *multiple input multiple output*) e invariante en el tiempo (LTI por *Linear Time Invariant*) es:

$$\begin{aligned}\dot{x}(t) &= A\boldsymbol{x} + B\boldsymbol{u} + D\boldsymbol{w}\\ y(t) &= C\boldsymbol{x} + E\boldsymbol{v}\end{aligned} \tag{6.4}$$

donde $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{w}$ son vectores y *A,B,C,D,E* son matrices con las dimensiones apropiadas. La solución de esta ecuación existe y es única para cualquier condición inicial $x(0) = x_0$ y cualquier señal de entrada *u*(*t*) para cualquier $0 \leq t < t_f$.

Existe analogía de representación de los sistemas continuos para sistemas discretos en tiempo. La notación común ignora el tiempo actual completamente y refiere a un sistema

discreto en el tiempo como un mapeo desde la secuencia de entrada $u[k]$, $k_0 \leq k \leq k_f$ a una salida $y[k]$, $k_0 \leq k \leq k_f$donde k es entero. La representación lineal en espacio de estados de sistemas discretos es:

$$x[k+1] = h(x[k], u[k])$$
$$y[k+1] = g(x[k], u[k])$$

Representación de la dinámica de frecuencias. Sea un NCS con *r* agentes, donde se ejecutan *n* tareas que transmiten datos por medio de la red de comunicación. La tarea τ_i con periodo p_i, tiempo de ejecución c_i tiene una frecuencia de transmisión de datos dada por la relación $1/p_i$ donde $i = 1, \ldots, r$. Una parte de la dinámica del sistema distribuido será la evolución de las frecuencias de transmisión de datos de los agentes que transmiten por la red, expresado de la forma:

$$\dot{x} = \zeta(f, u)$$

.

Se asume que existe una relación entre las frecuencias de transmisión $f_1, f_2, \ldots, f_3$ de tal forma que estas relaciones son los coeficientes de un sistema lineal invariante en el tiempo 6.4, donde $A \in \mathbb{R}^{k \times k}$ es la matriz de relaciones entre frecuencias de los agentes, $B \in \mathbb{R}^{k \times l}$ es la matriz con las escalas de las frecuencias, $C \in \mathbb{R}^{m \times k}$ son las frecuencias ordenadas $x \in \mathbb{R}^n$ es el vector de frecuencias reales de los agentes, la entrada $u \in \mathbb{R}^n$ está en función de la frecuencia de referencia y la frecuencia real, e $y \in \mathbb{R}^n$ es el vector de las frecuencias de salida.

Este trabajo propone tomar los coeficientes $a_{ij} \in A$ como los valores que están en función de las *frecuencias de transmisión mínima* (f_m) de los *n* agentes que transmiten datos tal que:

$$a_{ij} = \varphi(f_m^1, f_m^2, \ldots, f_m^n)$$

De manera equivalente, los valores de los coeficientes $b_{ij} \in B$ estarán en función de las *frecuencias de transmisión máxima*, (f_m) es decir

$$a_{ij} = \varphi(f_m^1, f_m^2, \ldots, f_m^n)$$

La entrada *u* estará dada por:

$$u = k^s e$$
$$u = k^s(\bar{y} - y)$$
$$u = k^s(f_m - f_r)$$

donde $f_r = (f_r^1, f_r^2, \ldots, f_r^n)$ es el vector de las frecuencias reales de transmisión con las que el agente *n* transmite y k^s es la ganancia de la planificación. De manera que:

$$x(k+1) = Ax(k) + Bu(k)$$
$$x(k+1) = Af_r + B\left(k^s(f_m - f_r)\right) \quad (6.6)$$

$$x(k+1) = Af_r + Bk^s f_m - Bk^s f_r$$
$$x(k+1) = (A - Bk^s)f_r - Bk^s f_m$$

Sea un NCS donde *n* agentes transmiten datos por medio de la red, con una *frecuencia de transmisión real* (f_r) acotada por las frecuencias mínimas (f_m) y máximas (f_x) entre las cuales cada agente pueda transmitir asegurando la restricción dada por 6.1, se proponen los coeficientes de las matrices del sistema 6.6 de la siguiente forma:

$$a_{ij} = \begin{cases} \frac{\Lambda}{f_m^i} & i = j \\ \\ \frac{f_m^j}{f_m^i} & i \neq j \end{cases} \qquad b_{ij} = \begin{cases} f_x^i & i = j \\ \\ 0 & i \neq j \end{cases} \quad \ldots . b_{ij} = \begin{cases} 1 & i = j \\ \\ 0 & i \neq j \end{cases}$$

donde $\Lambda = mcd(f_m^1, f_m^2, \ldots, f_m^n)$ es el máximo común divisor de las frecuencias mínimas [2].

Dado que:

$$f_m = [f_m^1, f_m^2, \ldots, f_m^n]'$$
$$f_x = [f_x^1, f_x^2, \ldots, f_x^n]'$$
$$f_r = [f_r^1, f_r^2, \ldots, f_r^n]'$$

la ecuación 6.6 se reescribe como:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ \\ \vdots \\ \\ x_n(k+1) \\ x_c(k+1) \end{bmatrix} = \begin{bmatrix} \frac{\Lambda}{f_m^1} & \frac{f_m^2}{f_m^1} & \frac{f_m^3}{f_m^1} & \cdots & \frac{f_m^n}{f_m^1} & 0 \\ \frac{f_m^1}{f_m^2} & \frac{\Lambda}{f_m^2} & \frac{f_m^3}{f_m^2} & \cdots & \frac{f_m^n}{f_m^2} & 0 \\ \frac{f_m^1}{f_m^3} & \frac{f_m^2}{f_m^3} & \frac{\Lambda}{f_m^3} & \cdots & \frac{f_m^n}{f_m^3} & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ \frac{f_m^1}{f_m^4} & \frac{f_m^2}{f_m^4} & \frac{f_m^3}{f_m^4} & \cdots & \frac{\Lambda}{f_m^n} & 0 \\ c_1 & c_2 & c_3 & \cdots & c_n & 1 \end{bmatrix} \begin{bmatrix} f_r^1 \\ f_r^2 \\ f_r^3 \\ \\ \vdots \\ \\ f_r^n \\ x_c \end{bmatrix} +$$

$$\begin{bmatrix} f_x^1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & f_x^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & f_x^3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & f_x^n & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \begin{bmatrix} k_1^s & 0 & 0 & \cdots & 0 & 0 \\ 0 & k_2^s & 0 & \cdots & 0 & 0 \\ 0 & 0 & k_3^s & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & k_n^s & 0 \\ 0 & 0 & 0 & \cdots & 0 & k_c^s \end{bmatrix}$$

[2] Ver **Hiperperiodo** sección **¡Error! No se encuentra el origen de la referencia.**.

$$\left(\begin{bmatrix} f_m^1 \\ f_m^2 \\ f_m^3 \\ f_m^4 \\ x_c \end{bmatrix} - \begin{bmatrix} f_r^1 \\ f_r^2 \\ f_r^3 \\ f_r^4 \\ x_c \end{bmatrix}\right)$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ \vdots \\ y_n(k) \\ y_c(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} f_r^1 \\ f_r^2 \\ f_r^3 \\ \vdots \\ f_r^n \\ x_c \end{bmatrix}$$

donde se tiene un estado aumentado x_c que representa la restricción de planificabilidad (6.1).

Se sigue entonces que:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ \vdots \\ x_n(k+1) \\ x_c(k+1) \end{bmatrix} = \begin{bmatrix} \Lambda & \frac{f_m^2}{f_m^1} & \frac{f_m^3}{f_m^1} & \cdots & \frac{f_m^n}{f_m^1} & 0 \\ \frac{f_m^1}{f_m^2} & \Lambda & \frac{f_m^3}{f_m^2} & \cdots & \frac{f_m^n}{f_m^2} & 0 \\ \frac{f_m^1}{f_m^3} & \frac{f_m^2}{f_m^3} & \Lambda & \cdots & \frac{f_m^n}{f_m^3} & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ \frac{f_m^1}{f_m^4} & \frac{f_m^2}{f_m^4} & \frac{f_m^3}{f_m^4} & \cdots & \Lambda & 0 \\ c_1 & c_2 & c_3 & \cdots & c_n & 1 \end{bmatrix} \begin{bmatrix} f_r^1 \\ f_r^2 \\ f_r^3 \\ \vdots \\ f_r^n \\ x_c \end{bmatrix} +$$

$$+ \begin{bmatrix} k_1^s f_x^1 (f_m^1 - f_r^1) \\ k_2^s f_x^2 (f_m^2 - f_r^2) \\ k_3^s f_x^3 (f_m^3 - f_r^3) \\ \\ \\ k_n^s f_x^n (f_m^n - f_r^n) \\ k_1^s (f_m^1 - f_r^1) + k_2^s (f_m^2 - f_r^2) + \cdots + k_c^s (x_c^r - x_c) \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ \vdots \\ y_n(k) \\ y_c(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} f_r^1 \\ f_r^2 \\ f_r^3 \\ \vdots \\ f_r^n \\ x_c \end{bmatrix}$$

A continuación se deriva el polinomio característico de la matriz de parámetros *A*. Para facilitar el cálculo, se consideran sólo tres nodos del sistema y sus respectivos tiempos de ejecución, con lo que la matriz *A* queda:

$$A = \begin{bmatrix} \lambda - \frac{\Lambda}{f_m^1} & -\frac{f_m^2}{f_m^1} & -\frac{f_m^3}{f_m^1} & 0 \\ -\frac{f_m^1}{f_m^2} & \lambda - \frac{\Lambda}{f_m^2} & -\frac{f_m^3}{f_m^2} & 0 \\ -\frac{f_m^1}{f_m^3} & -\frac{f_m^2}{f_m^3} & -\lambda - \frac{\Lambda}{f_m^3} & 0 \\ -c_1 & -c_2 & -c_3 & \lambda - 1 \end{bmatrix}$$

El polinomio característico se obtiene resolviendo la ecuación $|\lambda I - A| = 0$. Entonces, sea $\lambda I - A$ de la forma:

$$\lambda I - A = \begin{bmatrix} \lambda - \frac{\Lambda}{f_m^1} & \frac{f_m^2}{f_m^1} & \frac{f_m^3}{f_m^1} & 0 \\ \frac{f_m^1}{f_m^2} & \frac{\Lambda}{f_m^2} & \frac{f_m^3}{f_m^2} & 0 \\ \frac{f_m^1}{f_m^3} & \frac{f_m^2}{f_m^3} & \frac{\Lambda}{f_m^3} & 0 \\ c_1 & c_2 & c_3 & 1 \end{bmatrix}$$

El determinante de $|\lambda I - A|$ es de la forma:

$$\begin{aligned} &\left(\lambda - \frac{\Lambda}{f_m^1}\right)\left[\left(\lambda - \frac{\Lambda}{f_m^2}\right)\left(\lambda - \frac{\Lambda}{f_m^3}\right)(\lambda - 1) - (\lambda - 1)\left(\frac{f_m^3}{f_m^2}\right)\left(\frac{f_m^2}{f_m^3}\right)\right] \\ &+ \left[\left(-\frac{f_m^1}{f_m^2}\right)\left(\lambda - \frac{\Lambda}{f_m^3}\right)(\lambda - 1) - (\lambda - 1)\left(\frac{f_m^3}{f_m^2}\right)\left(\frac{f_m^1}{f_m^3}\right)\right] \\ &- \frac{f_m^3}{f_m^1}\left[\left(\frac{f_m^1}{f_m^2}\right)\left(\frac{f_m^2}{f_m^3}\right)(\lambda - 1) + (\lambda - 1)\left(\lambda - \frac{\Lambda}{f_m^2}\right)\left(\frac{f_m^1}{f_m^3}\right)\right] \end{aligned}$$

Se puede probar que el sistema es observable y controlable en todo momento considerando los límites tanto inferior y superior que son limitantes importantes en la respuesta en frecuencia del sistema, entonces el sistema se considera con una representación conservadora respecto del conjunto de respuestas posible.

6.3 Análisis de planificabilidad

Para que el sistema sea planificable bajo un algoritmo de prioridades fijas es necesario considerar que el conjunto de *n* tareas $\Gamma_j = \{\tau_1, \tau_2, \cdots, \tau_n\}$ que se realizan en cada procesador *j* cumplan (Liu y Layland, 1973; Butazzo, 2005):

$$U_j = \sum_{i=1}^{n} \frac{c_i}{p_i} < 1 \tag{6.7}$$

donde c_i es el tiempo de cómputo y p_i el periodo de la tarea τ_i perteneciente al procesador *j* para $i = 1, \cdots, n$ y $j = 1, \cdots, k$.

Se asume que para cada procesador *j* con *n* tareas existen *s* posibles valores de configuraciones para c_i y p_i que satisfacen (6.7), esto es un conjunto $\mathcal{H}_j^*$ que contiene todas las posibles configuraciones correspondientes a los parámetros de tareas bajo los cuales el conjunto Γ_{ij} de tareas es planificable:

$$\mathcal{H}_j^* = \{\mathcal{H}_j^1, \mathcal{H}_j^2, \ldots, \mathcal{H}_j^s\}$$

donde $\mathcal{H}_j^1, \mathcal{H}_j^2, \ldots, \mathcal{H}_j^s$ son conjuntos maximales de configuraciones locales que son planificables, tal que tengan la mayor utilización posible, sin provocar tráfico en la red. Así, un conjunto global de configuraciones factibles de ser planificable estará dado por:

$$\overline{\mathcal{H}} = \{\mathcal{H}_1^1, \mathcal{H}_1^2, \ldots, \mathcal{H}_1^s, \mathcal{H}_2^1, \mathcal{H}_2^2, \ldots, \mathcal{H}_2^s, \mathcal{H}_3^1, \mathcal{H}_3^2, \ldots, \mathcal{H}_3^s, \mathcal{H}_k^1, \mathcal{H}_k^2, \ldots, \mathcal{H}_k^s\}$$

El problema a resolver es reconfigurar los periodos de tarea de cada nodo sensor basado en la frecuencia de transmisión de datos, de manera tal que asegurando la planificabilidad local de cada conjunto $\mathcal{H}_j^*$ se logre una planificación global $\overline{\mathcal{H}}$. Para lograr lo anterior, en cada procesador *j* es posible modificar los parámetros de tareas a través del tiempo:

$$\mathcal{H}_j' \underset{\varphi_1}{\rightarrow} \mathcal{H}_j' \underset{\varphi_2}{\rightarrow} \mathcal{H}_j' \underset{\varphi_3}{\rightarrow} \mathcal{H}_j' \underset{\varphi_4}{\rightarrow} \cdots \underset{\varphi_r}{\rightarrow} \mathcal{H}_j^* \tag{6.8}$$

donde $\underset{\varphi_1}{\rightarrow}, \underset{\varphi_2}{\rightarrow}, \cdots, \underset{\varphi_r}{\rightarrow}$ son *r* modificaciones locales de frecuencias de transmisión y $\mathcal{H}_j'$ es cualquier subconjunto de configuraciones de parámetros de tareas no planificable para algún procesador *j*. Si durante la ejecución del sistema existe una secuencia como (6.8) es posible converger a un estado del sistema dentro de donde el sistema es planificable globalmente.

El modelo de transmisión de frecuencias propuesto tiene la finalidad de modificar la frecuencia de transmisión de los nodos sensores por medio de un control LQR donde la ganancia *k* modifica el subconjunto de configuraciones de los parámetros de tareas, posiblemente no planificables, hacia una nueva configuración factible de ser planificable.

En la figura 6.5 se muestra un diagrama de tiempo donde en el instante t_0 cuatro conjuntos de tres tareas cada uno, pertenecientes a cuatro procesadores P_1, P_2, P_3, P_4, tienen diferentes configuraciones de periodos y consumos tales que tres conjuntos son planificables $\mathcal{H}^*$ y un conjunto de tareas no planificable $\mathcal{H}'$, esto es $\mathcal{H}_1^1, \mathcal{H}_2^2, \mathcal{H}_3', \mathcal{H}_4^4$. Tres conjuntos de

tareas cumplen los plazos de tiempo (todas las tareas dentro de la región) y un conjunto donde las tares no cumplen con los plazos (tareas fuera de la región). Después de la primera transición $\underset{\varphi_1}{\rightarrow}$ en el tiempo t_1 un subconjunto de configuraciones de tareas continúa siendo no planificable, después de la segunda transición $\underset{\varphi_2}{\rightarrow}$ en el tiempo t_2 todos los subconjuntos de configuración de parámetros de tareas son planificables.

Si todos los subconjuntos de configuraciones de parámetros de tareas logran estar en el conjunto de configuraciones $\bar{\mathcal{H}}$ después de *r* transiciones, el sistema se habrá reconfigurado de tal forma que la utilización de la red es óptima sin bajo muestreo ni generación de tráfico.

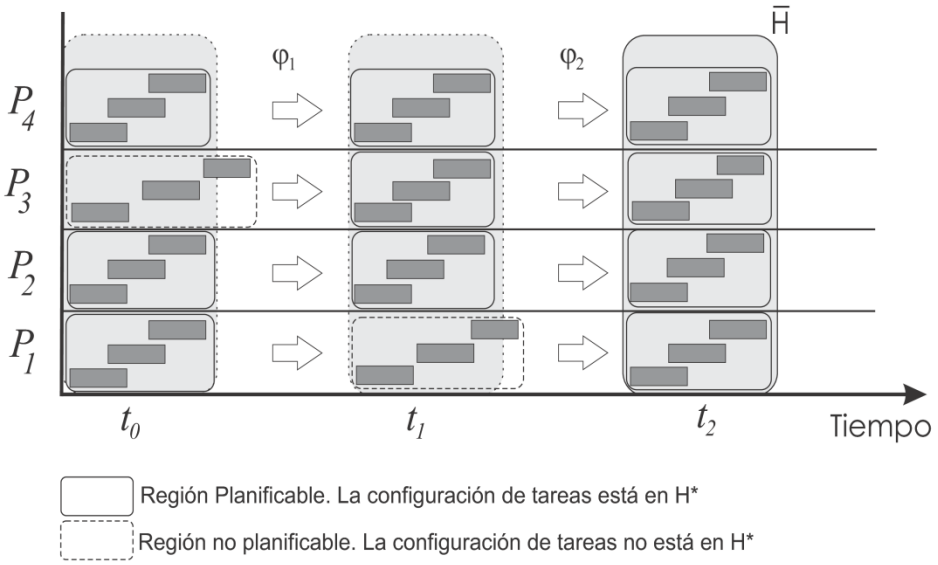

Figura 6.5: Transiciones de grupos de tareas dentro de regiones no planificables y planificables

6.4 Resumen

En este capítulo se mostró la propuesta de planificación del medio de comunicación común por medio de la reconfiguración de las frecuencias de transmisión de datos de los sensores de un NCS. Con base en el impacto que provoca la elección de un periodo de muestreo en el nivel de desempeño de los NCSs se mostró como identificar un intervalo en el cual se pueda elegir un periodo de muestreo adecuado que no implique una tasa de transmisión de datos grande que provoque tráfico en la red y, de manera similar, no disminuya el envío de datos provocando pérdida de la calidad de control.

El manejo de la holgura para elegir un periodo de muestreo certero, es la idea inicial de donde se partió para hacer la representación dinámica de las frecuencias de transmisión de datos. Se propuso controlar las frecuencias de transmisión de datos acotándolas dentro de una región de planificabilidad que puede común para todos los nodos del SD que utilizan la red de comunicación, una región individual para cada nodo involucrado en la transmisión de datos. Cada región está acotada por la frecuencia máxima y la frecuencia mínima, dentro del cual el uso de la red es planificable respecto a su utilización.

Se presentó el análisis de planificabilidad tomando en cuenta los parámetros de tareas que definen configuraciones planificables, para cada transición la nueva frecuencia elegida conduce a situarse dentro de una región planificable. La participación conjunta de los agentes

sensores descentraliza la reconfiguración y proporciona la información suficiente para equilibrar la carga del medio de comunicación compartido.

La reconfiguración de las frecuencias de transmisión de datos hace uso del rango de movilidad de las frecuencias, tal que se haga un uso óptimo de la red tomando en cuenta las relaciones de frecuencias entre todos los nodos que transmiten. Este tipo de reconfiguración es factible, dinámica y descentralizada.

Capítulo 7

Simulaciones Numéricas

En esta sección se muestra la simulación de las implementaciones de las propuestas de reconfiguración planteadas anteriormente. Las propuestas fueron simuladas usando *TrueTime* como herramienta de simulación de tiempo real, basado en *Matlab* y *Simulink*. Se agregan módulos de *Simulink* al SDTR del prototipo de helicóptero para caracterizarlo como un NCS que utiliza una red de comunicación común. La propuesta de reconfiguración con base en los parámetros de planificación se estudia contrastando tres escenarios: escenario sin falla, escenario con falla y escenario con falla y reconfiguración. En la propuesta de reconfiguración con base en las frecuencias de transmisión se plantea el escenario con falla y se modifican las frecuencias de transmisión de datos de forma dinámica. La efectividad de ambas propuestas es evaluada por medio del cálculo de la integral del valor absoluto del error (IAE).

7.1 Reconfiguración de los elementos de planificación

Menéndez y Benítez-Pérez (2010) presentan un estudio de planificación con base en la modificación de los periodos de transmisión, en el que desarrollan una estrategia global de planificación basada en el análisis del impacto de los periodos de tareas en un NCS. En su trabajo muestran que el desempeño del sistema no depende únicamente de los periodos de muestreo, también atiende a la dispersión entre estos periodos. La estrategia consiste de un periodo de operación global para el sistema en general, llamado *periodo base* ϱ**¡Error! Marcador no definido.** y factor de desplazamiento de inicio de las tareas llamado *factor de dispersión* λ. El periodo base y el factor de dispersión son usados para obtener el *periodo de operación* de cada nodo o agente del sistema:

$$n_i = \rho(1 \pm \lambda)$$

Cuando todos los agentes del sistema están compitiendo por la red para obtener ancho de banda de acuerdo al algoritmo de control de acceso a la red, el sistema experimenta variaciones en la transmisión de datos que se reflejan en los retardos en la comunicación y trafico, esto provoca pérdida de desempeño. La estrategia de planificación consiste en la utilización de planificador global que distribuya el ancho de banda entre cada nodo por medio de la asignación de una ventana de tiempo cuando estos transmiten independientemente del protocolo utilizado. Este nodo planificador sólo se considera como un controlador de acceso a la red.

El control de vuelo de prototipos de helicópteros se ha tomado como caso de estudio para implementar esquemas de control basados en lógica difusa para el manejo de retardos acotados (Méndez-Monroy y Benítez-Pérez, 2011a; Quiñones-Reyes y otros, 2012). Estudios

preliminares de planificación usando el sistema de control de vuelo del helicóptero (Menéndez y Benétez-Pérez, 2010) han mostrado que el NCS no presenta saturación en la red ni bajo muestreo para valores de $\rho \in [10,20]$ milisegundos y $\lambda \in [0,0.20]$. En la figura (7.1) se muestra la respuesta de los ángulos de cabeceo y guiñado obtenida al simular el vuelo del helicóptero durante 60 segundos utilizando un periodo base de 17 milisegundos y una dispersión de 10%. Para valores del periodo base menores a 5 milisegundos se apreció un error de seguimiento considerable, mientras que para valores mayores a 20 milisegundos (figura 7.2) el sistema experimenta sobrecarga en la red que provoca latencias grandes que deterioran el rendimiento de control.

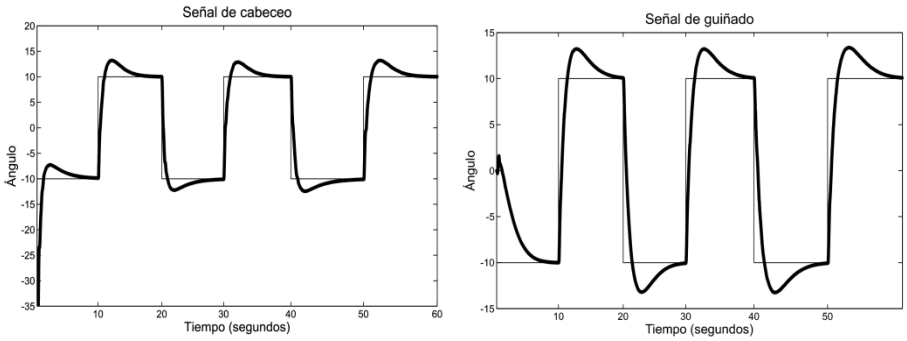

Figura 7.1: Respuesta del sistema de control en red para un periodo base de muestreo de 17 ms.

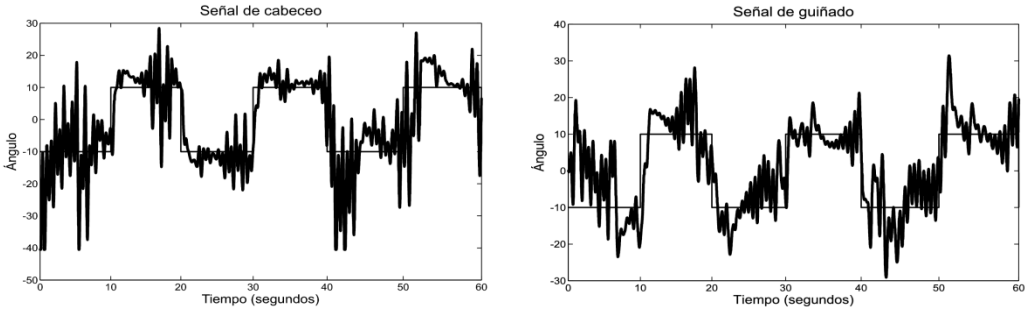

Figura 7.2: Respuesta del sistema de control en red para un periodo base de muestreo de 28 ms.

Para el primer enfoque de reconfiguración de esta tesis, se experimenta con una arquitectura que utiliza triple redundancia de sensores (Dubrova, 2008; Benítez-Pérez y García-Nocetti, 2005) con el propósito de manejar las posibles fallas que se experimenten durante la ejecución. Esto aumenta la cantidad de nodos que intentan tener acceso a la red, por ello es necesario manejar el ancho de banda de manera equitativa para que cada nodo transmita. Adicionalmente se muestra el análisis de planificación del esquema de reconfiguración que se utiliza con el propósito de asegurar que esta transición impacte al sistema.

Para esta propuesta, se agregó al sistema distribuido del helicóptero un submódulo que funciona como un NCS (figura 7.3). Este sistema contiene redundancia de sensores y un planificador de tareas global.

El sistema consta de 16 procesadores con kernel de tiempo real, conectados por medio de una red tipo CSM/AMP(CAN) (sección 3.3) con una tasa de envío de datos de 80,000 bits/s sin probabilidad de pérdida de datos. El primer agente en el modelo (figura 7.3), en la extrema izquierda, es el agente controlador que utiliza los valores provenientes de los sensores y para calcular las salidas de control u_py.u_y Los 12 agentes sensores siguientes (figura 7.3) están organizados en 4 bloques que sensan las señales $\theta, \psi, \dot{\theta}, \dot{\psi}$ correspondientes al ángulo de cabeceo, derivada del ángulo de cabeceo, ángulo de guiñada y la derivada del ángulo de guiñada respectivamente. Dos agentes actuadores, localizados a la extrema derecha abajo (figura 7.3), reciben las señales u_p y u_y. Se incluye un agente planificador, extrema derecha arriba (figura 7.3), que organiza la actividad de los otros agentes del sistema y es responsable de la asignación periódica del ancho de banda utilizada por todos los agentes. Los nodos sensores y el planificador ejecutan tareas periódicas mientras que el controlador y actuador realizan tareas aperiódicas activadas por evento.

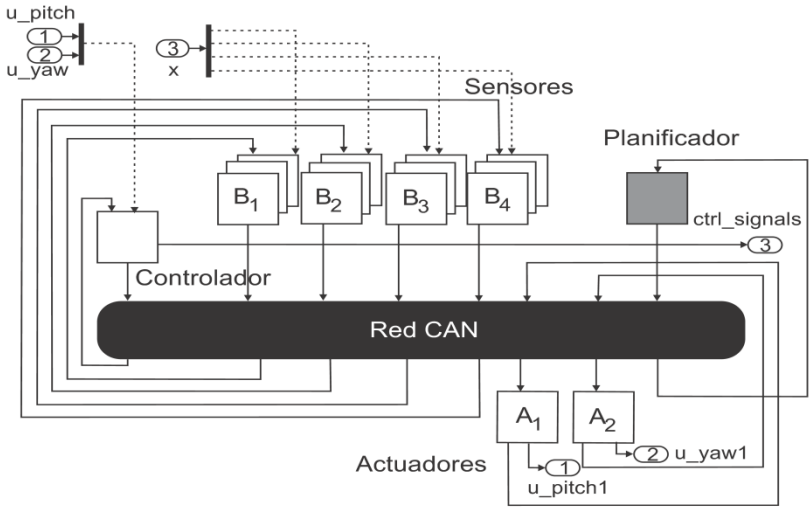

Figura 7.3: Sistema distribuido con módulos de sensores

Los experimentos consisten en inducir una falla en uno o más bloques de sensores. Distintos tipos de falla pueden acontecer en los sensores como pueden ser el retraso en la lectura, pérdida de un sensor, ruido exesivo o un tipo de falla conocida como bizantina (Sharma y otros, 2007) que tiene efectos arbitrarios. Se plantea un plan de reconfiguración para cada tipo de falla particular y se evalúa la acción de la reconfiguración por medio del cálculo del IAE.

Para el caso de estudio se revisaron tres escenarios**¡Error! Marcador no definido.**:

- No existe falla en el sistema.
- Se presentan fallas en los sensores y no hay reconfiguración.
- Se presentan fallas y se reconfigura el sistema.

Cada bloque de sensores tiene un periodo de operación p_i y tiempo de cómputo c_i como se muestra en la tabla 7.1.

Tabla 7.1: Periodos de muestreo y tiempos de cómputo de los bloques de sensores

Bloque	c_i	p_i
1	0.001	$\rho(1+\lambda)$
2	0.001	$\rho(1-\lambda)$
3	0.001	$\rho(1-1.1\lambda)$
4	0.001	$\rho(1+1.1\lambda)$

Escenario sin falla. De de las cuatro lecturas correspondientes a los estados del sistema $\theta, \psi, \dot{\theta}, \dot{\psi}$ se hace el análisis de la respuesta de la señal θ correspondiente al ángulo de cabeceo. Se simula el vuelo del prototipo durante 30 segundos y se calcula el valor de la IAE durante el tiempo total de la simulación.

El propósito de la reconfiguración consiste en restablecer al sistema ante escenarios con falla, para esto se evalúa el desempeño del NCS en un escenario sin falla. Se exploran las variaciones de la IAE respecto a la modificación de los valores de los parámetros ρ y λ, que servirá como referencia para determinar una cota mínima y máxima dentro de las cuales el sistema es planificable. En la Tabla 7.2 se muestra el valor de la IAE respecto a las variaciones de ρ dentro del intervalo $[0.010\ , 0.025]$ milisegundos y λ en el intervalo $[0.0\ , 0.020]$.

Tabla 7.2: Valores de la IAE al variar ρ y λ en un escenario sin falla

	Periodo base (milisegundos)															
Dispersión	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0.0	16.85	15.38	17.58	18.09	10.38	9.96	9.51	9.01	8.98	55.86	13.55	60.55	55.68	10.29	21.86	10.52
0.05	15.45	9.41	9.99	14.67	10.40	9.23	9.41	8.74	8.79	45.41	8.94	65.33	17.24	10.61	24.37	9.89
0.10	13.59	8.59	9.20	11.84	9.73	9.68	8.67	8.85	8.92	28.99	9.18	67.43	14.19	10.7.3	24.03	11.33
0.15	13.40	9.18	9.90	13.65	9.86	9.60	9.08	8.82	8.69	29.06	9.59	41.15	20.55	10.81	22.95	10.42
0.20	12.43	7.39	9.19	17.97	10.09	9.29	7.89	7.97.	7.99	45.05	8.58	26.95	15.19	10.06	23.15	7.80

Para valores del periodo base dentro del intervalo [0.10:,0.18] milisegundos el desempeño del sistema no presenta un IAE muy grande, menor a 10 unidades; no obstante al aumentar el periodo base, resulta en un aumento del valor de la IAE lo que representa una disminución de la calidad del control. En la figura 7.4 se observa la relación entre IAE y distintos valores de los parámetros de muestreo ρ y λ cuando no existe falla en el sistema.

En la figura 7.5 se muestra la lectura de los tres sensores del bloque B_1 que leen la señal del ángulo de cabeceo, cuando no existe falla alguna en el sistema y se fijan los valores para $\rho = 17$ milisegundos y $\lambda = 0.10$. El valor de la lectura de los tres sensores es idéntica, lo que deriva en un valor del IAE de 8.85 unidades.

La consistencia de las lecturas de los tres sensores del ángulo de cabeceo, se hacen en condiciones normales de operación, no existe ruido o retraso en las lecturas provocadas por alteraciones en el hardware (Dubrova, 2008) o en los convertidores A/D. La actividad de la red muestra una distribuición equitativa en el envío de datos de los sensores a través de la red(figura 7.6).

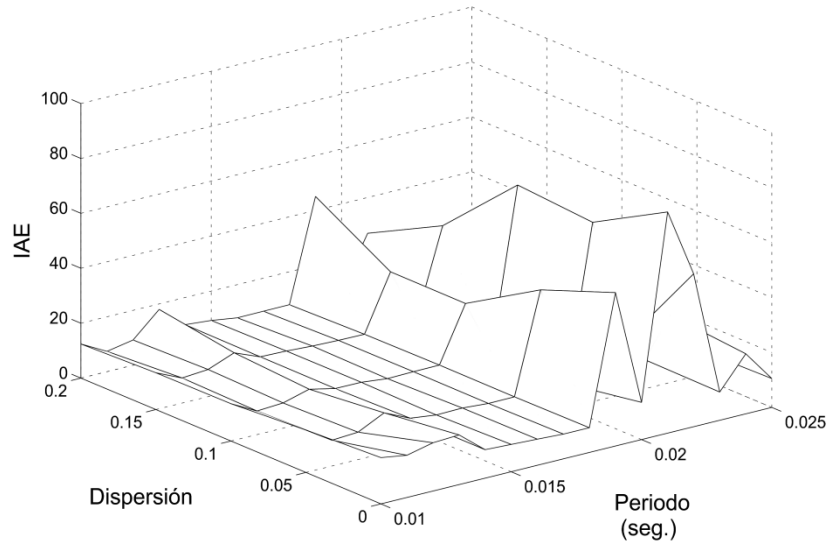

Figura 7.4: Relación de la IAE respecto a las variaciones del periodo base ϱ y dispersión λ en un escenario sin falla

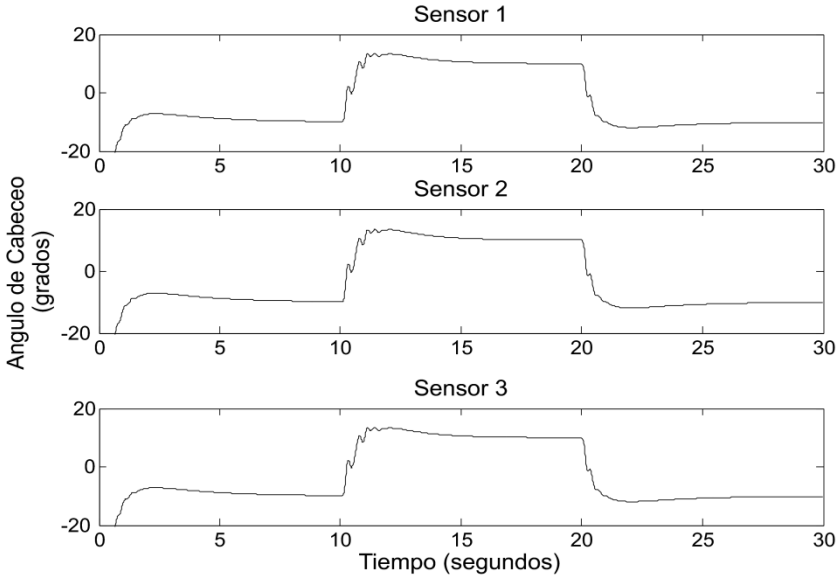

Figura 7.5: Lectura de la señal del ángulo de cabeceo en escenario sin falla

Escenario con falla. Los escenarios con falla pueden presentarse cuando hay retraso en la lectura de señales, pérdida de total de sensores, ruido o provocados por eventos arbitrarios conocidos también como fallas bizantinas (Driscoll y otros, 2003);lo que produce lecturas erróneas, falta de datos o cambios de frecuencias de envío. Estos tipo de fallas provoca pérdida de desempeño en el sistema.

Para explorar los escenarios con falla se induce en el sistema un tipo de falla en los agentes de los bloques de sensado. En particular se experimenta con el bloque B1 que hace la

lectura de la señal del ángulo de cabeceo θ. A uno de los sensores se le induce un retraso de tres segundos a partir del segundo $t = 10$ restableciéndose en el segundo $t = 23$. El periodo base del sistema se fijó a $\rho = 15$ milisegundos. y $\lambda = 0.10$, el tiempo total de simulación fue de 30 segs.

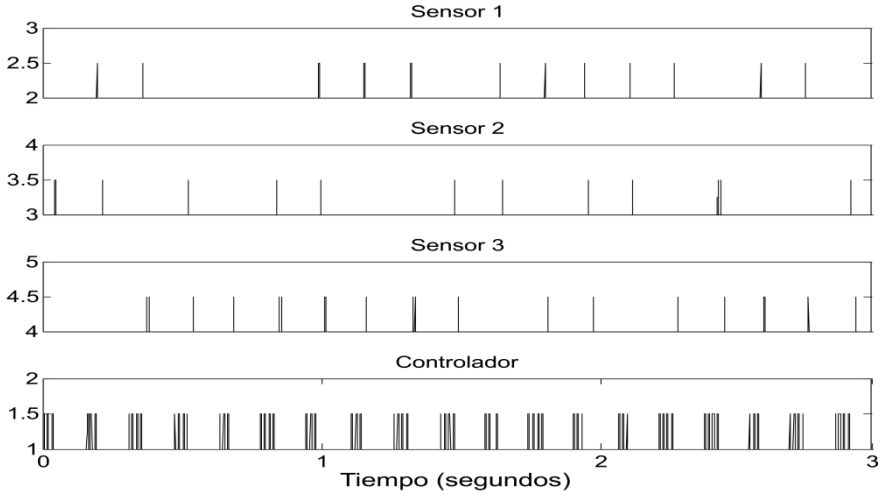

Figura 7.6: Transmisión de datos de los agentes del bloque B1 y agente controlador en escenario sin falla

En la figura 7.7 se observa el retraso en la lectura del tercer sensor del bloque B1 y el efecto que esto provoca, lo que se refleja en oscilaciones en el vuelo del helicóptero.

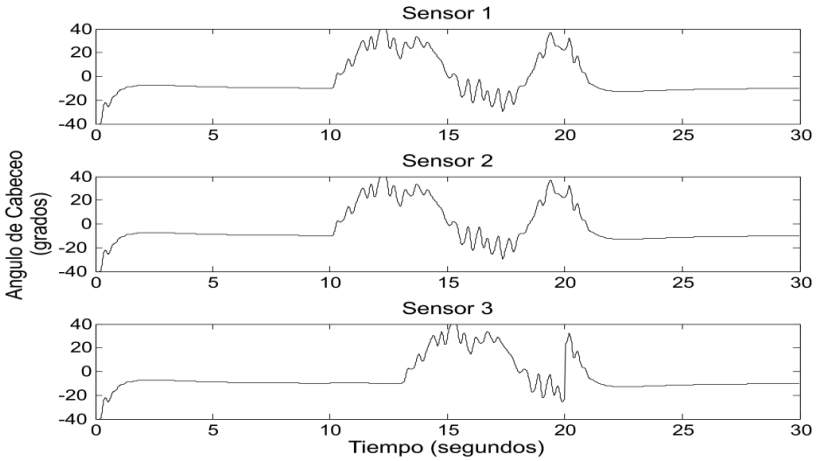

Figura 7.7: Lectura del ángulo de cabeceo por tres sensores con falla de retraso inducida en el tercer agente del bloque B1.

El valor de la IAE se incrementó respecto al escenario sin falla y resultó ser 12.24 para los valores de los parámetros de planificación mencionados arriba. Para este caso el periodo base elegido asegura una utilización que no excede el límite del ancho de banda ni bajo muestreo; no obstante, el efecto de la falla es notorio. Para valores del periodo base mayores a 18 milisegundos o menores 10 milisegundos (tabla 7.3) se agrega un efecto negativo adicional en el valor de la IAE provocado por la pérdida de desempeño del control. En la figura 7.8) se muestra la relación entre el valor de la IAE y distintos valores de ρ y λ cuando el sistema experimenta una falla de retraso en alguno de los sensores que leen la señal del ángulo de cabeceo.

Tabla 7.3: Valores de la IAE al variar ρ y λ en un escenario con falla.

	Periodo base (milisegundos)															
Dispersión	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0.0	18.09	14.74	17.54	14.10	11.49	11.22	15.50	16.69	12.12	56.77	73.46	98.26	88.20	12.12	70.28	17.44
0.05	16.34	9.92	10.57	10.89	11.57	11.15	16.26	12.49	12.49	58.31	40.75	127.2	68.26	19.21	34.09	28.67
0.10	13.52	10.01	10.20	9.42	10.64	12.22	16.99	17.74	13.99	69.76	36.46	101.2	33.86	17.48	29.55	13.80
0.15	15.39	10.88	12.01	11.08	11.17	12.20	19.33	20.16	14.63	93.68	88.41	111.0	55.80	16.87	23.12	16.58
0.20	10.55	8.20	10.18	15.98	10.40	10.40	12.56	16.98	13.32	82.85	14.71	112.4	43.38	13.55	24.43	13.06

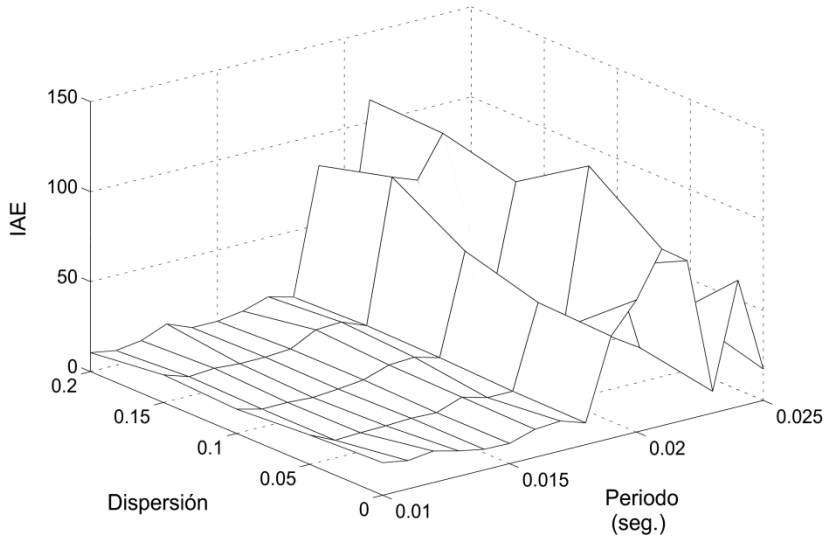

Figura 7.8: Relación de la IAE respecto a variaciones del periodo base ρ y dispersión λ en un escenario con falla.

Escenario con falla y reconfiguración por enmascaramiento. Para evaluar el efecto de la reconfiguración bajo escenarios con falla, se exploran distintas técnicas de manejo de fallas: enmascaramiento de falla, disminución o suspensión de la transmisión de datos del sensor que falla, modificación de los parámetros de planificación. La primera forma de reconfiguración que se analizó fue el enmascaramiento de fallas (Latif-Shabgahi y otros, 2001, 2003). Una vez identificada la falla de retraso, se pone en marcha el intercambio de datos entre los agentes del mismo bloque de sensado, la lectura local de los tres agentes del bloque alimentan la entrada de la unidad de voto y el valor devuelto es transmitido a la misma frecuencia que los otros agentes. Se induce en el tercer sensor del bloque de sensado B1 un retraso de tres segundos que inicia en el segundo 10 y termina en el segundo 23, el periodo base se fijó en $\rho = 15$ milisegundos y la dispersión $\lambda = 0.10$, el tiempo total de simulación fue de 30 segundos. En la figura 7.9 se muestra el registro de la lectura de los tres sensores del bloque. La actividad de la red para este caso se muestra en la figura 7.10.

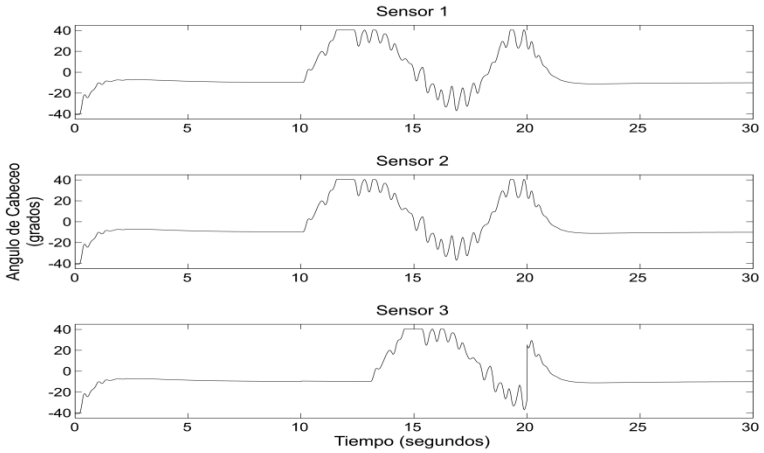

Figura 7.9: Lectura del ángulo de cabeceo por tres sensores con falla de retraso y reconfiguración por enmascaramiento.

En la figura 7.10 se muestra la lectura de los tres sensores cuando en un escenario con falla y se reconfigura por la enmascaramiento.

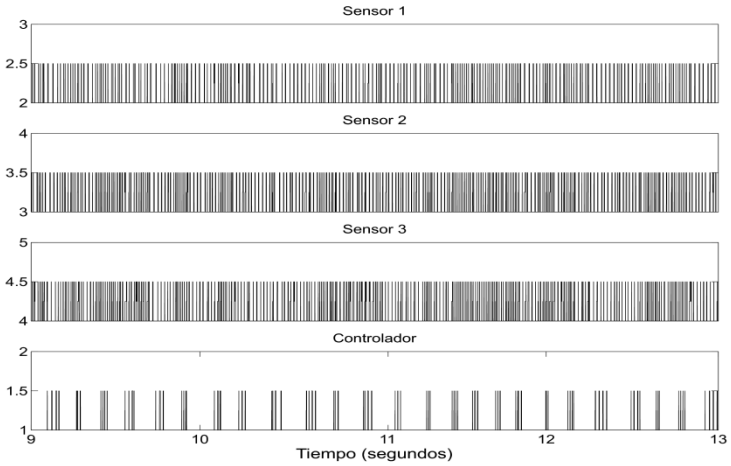

Figura 7.10: Transmisión de datos de los agentes del bloque de sensado y del agente controlador con falla de retraso y reconfiguración por enmascaramiento.

Bajo este escenario se observó que el sistema no logra una respuesta que disminuya el efecto de la falla, el valor del IAE fue de 12.55 unidades, para los valores de ρ y λ mencionados antes. La actividad de la red aumenta por la transmisión y recepción de datos entre agentes. Existe un costo adicional al reconfigurar con base en un algoritmo de enmascaramiento que requiere el intercambio de datos, hay una competencia adicional por el recurso de red que impacta en el desempeño de control. En la tabla 7.4 se muestra el valor de la IAE al modificar ρ dentro del intervalo entre $[10, 25]$ milisegundos. y λ en el intervalo entre $[0.0, 0.20]$.

Tabla 7.4: Valores de la IAE al variar ρ y λ en un escenario con falla y reconfiguración por enmascaramiento.

	Periodo base (milisegundos)															
Dispersión	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0.0	31.71	18.47	31.84	23.89	12.11	12.30	34.28	35.59	18.68	58.60	87.86	107.6	96.13	13.15	80.68	16.98
0.05	19.60	13.75	11.73	11.13	12.16	11.81	37.63	20.84	27.78	67.08	64.80	128.5	53.48	29.58	33.44	16.43
0.10	20.68	13.17	10.85	10.15	12.45	12.55	37.34	25.58	23.17	57.77	61.62	94.6	68.50	34.76	29.95	23.71
0.15	22.34	11.94	17.07	49.16	18.87	13.30	36.89	30.73	22.14	101.77	93.70	105.9	57.01	26.94	23.46	14.42
0.20	23.73	10.44	10.82	66.85	10.87	11.03	14.90	24.58	22.16	72.70	24.24	127.5	48.47	13.68	25.87	21.52

La reconfiguración con base en la utilización de un algoritmo de enmascaramiento no impacta positivamente para compensar el efecto de la falla de retraso. Los valores obtenidos de la IAE (figura 7.11) fueron muy variables aún utilizando el periodo base dentro del intervalo de planificabilidad.

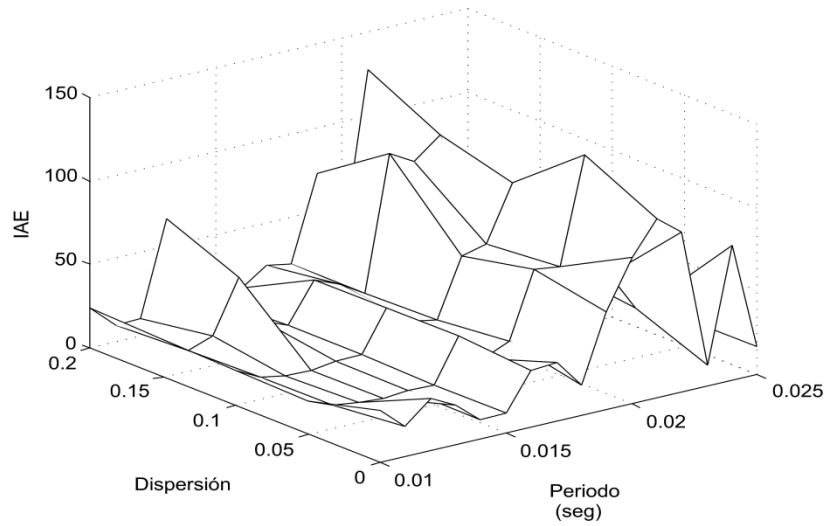

Figura 7.11: Relación de la IAE respecto a variaciones del periodo base ρ y dispersión λ en un escenario con falla y reconfiguración por enmascaramiento

Escenario con falla y reconfiguración por disminución de transmisión. El siguiente tipo de reconfiguración implementado fue la disminución de la tasa de transmisión de datos por el sensor que falla. Se fijó en $\rho = 15$ milisegundos y la dispersión $\lambda = 0.10$. El sensor que identifica la falla en la lectura acuerda disminuir su actividad, la tasa de transmisión disminuye a un tercio de su valor original sin modificar su periodo de muestreo, el valor del IAE fue de 11.2. De manera similar también se probó reconfigurar con base en la suspensión de la transmisión de datos por parte del sensor que experimenta la falla, los sensores sin falla asumen la pérdida de un sensor del bloque, bajo este escenario el valor del IAE fue de 9.3 para los mismos valores ρ y λ del caso anterior.

En la figura 7.12 se muestra la lectura de los tres sensores del bloque cuando se reconfigura por disminución de la transmisión de datos. En la figura 7.13 se muestra la lectura de los tres sensores cuando se reconfigura por la suspensión total de transmisión del agente que falla.

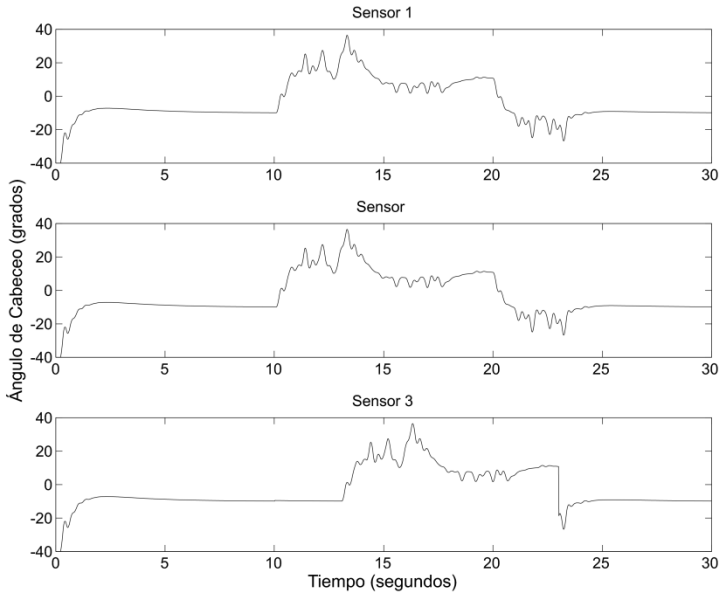

Figura 7.12: Lectura de los sensores del ángulo de cabeceo con falla de retraso y reconfiguración por disminución de envío de datos.

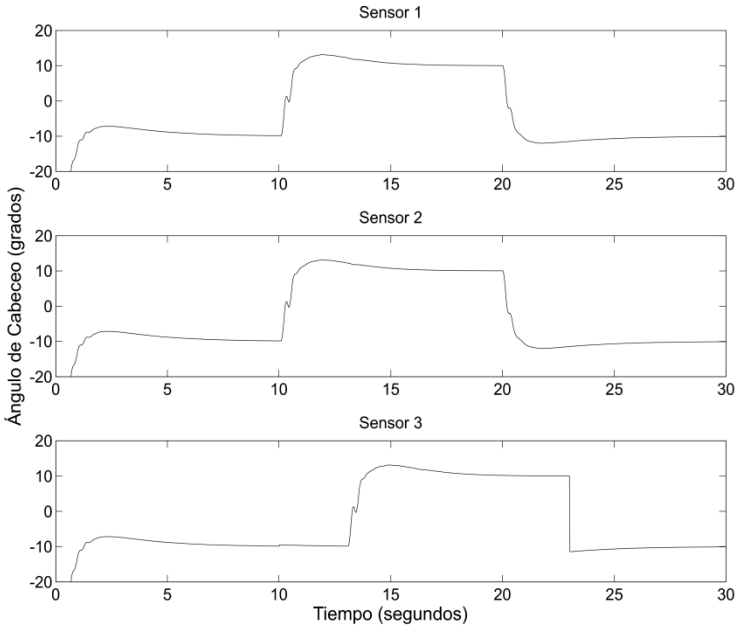

Figura 7.13: Lectura de los sensores del ángulo de cabeceo con falla de retraso y reconfiguración por suspensión total de transmisión del agente con falla

En ambos casos, la transición debido al acuerdo entre agentes es irrelevante permitiendo un buen desempeño. Recuérdese que la transición se lleva acabo considerando el diseño global del controlador para ambos escenarios de retardos de tiempo exclusivamente sin considerar el tráfico de la red de cómputo.

Escenario con falla y reconfiguración por modificación de los parámetros de planificación. Finalmente, se implementa una reconfiguración que modifica los periodos de muestreo de los sensores del sistema distribuido por medio del cambio de valores de ρ y λ. Al inicio de la ejecución los valores de ρ y λ se establecen en 22 milisegundos y 0.10 respectivamente, a partir del momento de inicio de la falla se modifica el periodo base a $\rho = 15$ milisegundos. El resultado de esta reconfiguración muestra que la inestabilidad del sistema provocada por la falla es corregida parcialmente dado que el IAE resultó en 18.48 unidades. En figura 7.14 se muestra el valor de la lectura de los datos y la actividad de los sensores del bloque.

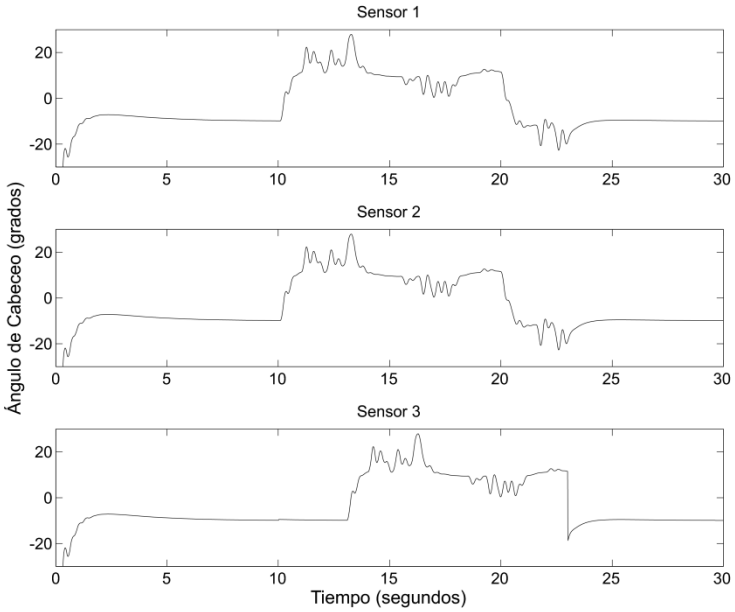

Figura 7.14: Lectura de los sensores del ángulo de cabeceo. Falla de lectura y reconfiguración por cambio de los periodos de muestreo.

Para cualquier falla que ocurra en el sistema y que posiblemente también impacte en la modificación de los parámetros de planificación, se requiere un acuerdo entre los agentes del NCS para evitar una situación catastrófica en el manejo de la red. Por ejemplo las regiones presentadas en las figuras 7.8 y 7.11 donde la IAE tiende a aumentar debido al alto uso de comunicación en la red de cómputo.

Las simulaciones anteriores muestran el impacto que provoca una falla en el sensado de las señales de control. Con base en la redundancia de sensores se construye una arquitectura multiagente donde cada nodo sensor del sistema distribuido funge como una entidad que puede tomar la lectura de uno de los estados del sistema, comunicarse con otros sensores y consensuar decisiones. Se desea que estos agentes manejen los efectos de posibles fallas que se presenten durante la ejecución del sistema por medio de distintos mecanismos de reconfiguración. Estos mecanismos de reconfiguración pueden ser el uso de algoritmos de enmascaramiento, disminución o supresión total de la transmisión de datos del agente que falla, o bien, el cambio de periodos de muestreo.

Con base en los resultados expuestos anteriormente, se observa que el mecanismo de reconfiguración más efectivo cuando existe una falla de retraso en la lectura de los sensores es la suspensión total de la transmisión de datos del sensor que falla, un poco menos efectiva pero válida resultó ser la disminución de la transmisión de datos del agente con falla; sin embargo, esto podría ser improcedente para fallas producidas en dos o más sensores dentro de un mismo módulo de sensado.

La reconfiguración que utiliza el mecanismo de enmascaramiento de falla no resultó ser efectiva dado que aumenta el tráfico de la red. Bajo este escenario de falla se mostró que El impacto de la reconfiguración provoca el aumento del valor de la IAE; no obstante, cabe señalar que el efecto de fallas en dos agentes del mismo módulo de sensado podría disminuirse mediante voto. La reconfiguración que modifica los periodos base de los agentes sensores provoca efectos favorables, sin embargo no se logra definir un rango en los periodos de muestreo en el que se tenga un efecto claramente positivo.

Se desea, por tanto, lograr un grado de determinismo para realizar una reconfiguración que cumpla con las restricciones de tiempo real, las modificaciones de los periodos de muestreo que inciden directamente en la planificación del medio de comunicación deben ser acotadas para mantener un nivel de desempeño del sistema. A continuación se muestran las simulaciones numéricas en las que se implementó la propuesta de reconfiguración con base en las frecuencias de transmisión.

7.2 Reconfiguración de las frecuencias de transmisión

Con la finalidad de estudiar el impacto del uso de la red de comunicación en un SDTR y el efecto de la reconfiguración sobre las frecuencias de transmisión de datos, se incluyó en el modelo del helicóptero 2DOF un subsistema distribuido de tiempo real donde diversos nodos están conectados a través de una red de comunicación compartida. El experimento se enfoca en la planificación de la red por medio de la reconfiguración de las frecuencias de transmisión de datos con el propósito de balancear la cantidad de datos enviados por la red al acotar las frecuencias dentro de una región de planificabilidad.

El subsistema de distribuido consta de 8 procesadores (figura 7.15) con kernel de tiempo real conectados por medio de una red CSMA/AMP (CAN) con tasa de transmsión de datos de 80000 bits/seg. sin pérdida de datos. Los bloques con kernel de tiempo real y el bloque de red fueron simulados con la herramienta de simulación de tiempo real *TrueTime* e implementados con *Simulink/Matlab*. Cuatro nodos sensores (figura 7.15) ejecutan periódicamente las tareas de sensado de los estados del sistema junto con tareas adicionales, cada tarea tiene periodo p_i y un tiempo de ejecución c_i. Los estados del sistema que son leídos corresponden a los ángulos de cabeceo, guiñado y sus respectivas derivadas $\theta, \psi, \dot{\theta}, \dot{\psi}$. En el subsistema distribuido, a la extrema derecha (figura7.15) se localiza el nodo controlador el cual hace el cálculo de la ley de control por medio de una tarea aperiódica activada por evento, el tiempo de ejecución de la tarea del controlador es equivalente al promedio del máximo tiempo que toma en calcularse la ley de control. El nodo controlador envía la señales de control u_p y u_y correspondientes a los voltajes que se aplicarán a los propulsores del las hélices. Dos nodos actuadores situados a la extrema derecha abajo (figura7.15) reciben la señales de control y las aplican a los encoders de cada hélice. Finalmente, el nodo planificador situado a la extrema

derecha arriba (figura 7.15) distribuye el acceso a la red por medio de de la asignación periódica de ancho de banda a los demás nodos del subsistema.

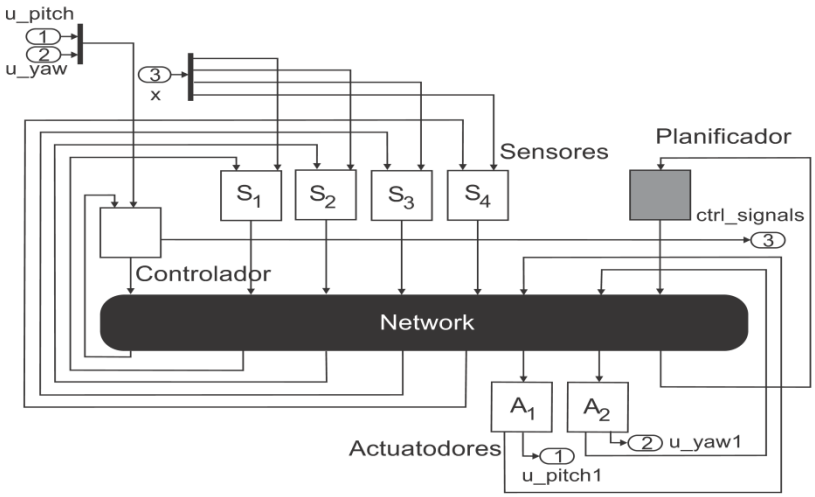

Figura 7.15: Sistema distribuido con planificador de frecuencias de transmisión

Este subsistema es similar al manejado en la propuesta de reconfiguración de los elementos de planificación (sección 7.1) con la diferencia que no usa redundancia de sensores y el enfoque consiste en reconfigurar las frecuencias de transmisión de datos para lograr el uso óptimo del medio de comunicación. Se espera que el cambio de las frecuencias de transmisión de datos y la política de planificación utilizada modifiquen positivamente el desempeño del sistema. El índice de desempeño que se utiliza es la integral del valor absoluto del error. En la figura 7.16 se muestra la transmisión de datos supervisada por el nodo planificador que calcula el control de las frecuencias de transmisión de datos.

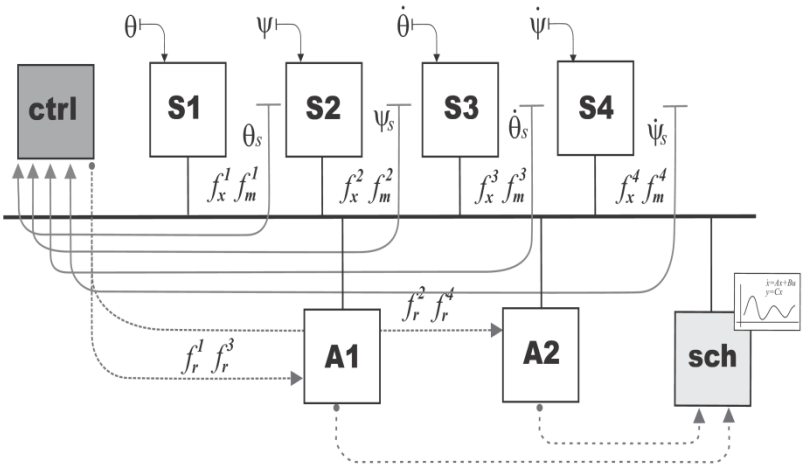

Figura 7.16: Comunicación sensor-controlador-actuador planificando las frecuencias de transmisión.

Debido a que la frecuencia de transmisión de datos en un NCS puede ser alta debido a incertidumbres durante la ejecución provocando sobrecarga en la red y como consecuencia pérdida de rendimiento de control, es importante controlar las frecuencias de transmisión de manera dinámica. Se propone utilizar un esquema de control del tipo Regulador Cuadrático

Lineal[1] para regular las frecuencias de transmisión dentro del rango óptimo. La reconfiguración de las frecuencias de transmisión de datos considera la relación $f = \frac{1}{p}$ para facilitar el manejo dinámico (Peng y otros, 2008, 2009) y el acotar las tasas de transmisión dentro del rango definido por la frecuencia máxima f_x y la frecuencia mínima f_m de transmisión de los nodos.

El cálculo de las frecuencias máximas y mínimas (tabla 7.5 se hizo fuera de línea con base en los valores mínimos y máximos de los periodos de muestreo en los que el sistema mantiene un valor de IAE bajo (Menéndez y Benítez-Pérez, 2010).

Tabla 7.5: Frecuencias de transmisión máxima, mínima, real y tiempo de ejecución agentes sensores

Agente	c_i	f_x^i	f_m^i	f_r^i
s_1	0.001	310	60	40
s_2	0.001	270	50	250
s_3	0.001	270	50	100
s_4	0.001	300	45	50

Cálculo de la ley de control LQR. La matriz *A* que representa las relaciones entre las frecuencias de los agentes sensores del sistema $\dot{x}=Ax+Bu$ queda:

$$A = \begin{bmatrix} 0.083 & 0.833 & 0.833 & 0.750 & 0 \\ 1.200 & 0.100 & 1.000 & 0.900 & 0 \\ 1.200 & 1.000 & 0.100 & 0.900 & 0 \\ 1.333 & 1.111 & 1.111 & 0.111 & 0 \\ 0.001 & 0.001 & 0.001 & 0.001 & 1 \end{bmatrix}$$

con eigenvalores:

$$\lambda_1 = 1.0000$$
$$\lambda_2 = 3.0986$$
$$\lambda_3 = -0.9119$$
$$\lambda_4 = -0.9000$$
$$\lambda_5 = -0.8922$$

Debido al valor de los eigenvalores de *A* el sistema es inestable, esto tiene sentido dado que las frecuencias de transmisión nominales (Tabla 7.5) con las que comienza a funcionar el sistema no son planificables, existe una carga de transmisión de datos que la red no puede transmitir o bien baja transmisión de datos.

El método LQR ha sido elegido porque es necesario usar una respuesta acotada para garantizar la estabilidad, considerando las condiciones de nolinealidad. Esta estrategia no se enfoca en la convergencia de estados, se enfoca en la planificación de elementos desde la perspectivas de la dinámica de sistemas en tiempo real (Butazzo, 2005).. El sistema es invariante en el tiempo dada una cota definida por mínimo periodo de tarea posible para el

[1] LQR *Linear Quadratic Regulator*

cambio de escenarios. En este sentido, la modificación de las frecuencias de transmisión es únicamente un cambio paramétrico el cual es adaptado por un regulador de ganancia. El peso de las matrices *Q* y *R* fueron elegidos de la siguiente forma:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix} \qquad R = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

Las matriz de ganancias *K* y la nueva matriz de estados $A_C = (A - BK)$ son de la forma:

$$K = \begin{bmatrix} 0.317 & 0.004 & 0.004 & 0.004 & -0.059 \\ 0.004 & 0.317 & 0.004 & 0.004 & -0.035 \\ 0.004 & 0.004 & 0.317 & 0.004 & -0.035 \\ 0.004 & 0.004 & 0.004 & 0.317 & -0.032 \\ -0.007 & -0.008 & -0.008 & -0.007 & 2.046 \end{bmatrix}$$

$$A_C = \begin{bmatrix} -98.12 & -0.44 & -0.44 & -0.41 & 18.39 \\ 0.260 & -85.43 & -0.05 & -0.04 & 9.57 \\ 0.260 & -0.05 & -85.43 & -0.04 & 9.57 \\ 0.270 & -0.05 & -0.05 & -94.91 & 9.45 \\ -0.320 & -0.32 & -0.32 & -0.32 & -0.88 \end{bmatrix}$$

Los eigenvalores de A_C que se obtuvieron fueron

$$\lambda_1 = -1.049$$
$$\lambda_2 = -98.007$$
$$\lambda_3 = -94.915$$
$$\lambda_4 = -85.423$$
$$\lambda_5 = -85.382$$

Escenario con falla. La respuesta del NCS utilizando esta estrategia de reconfiguración se exploró simulando durante 60 segundos el caso de estudio e induciendo en ciertos periodos de 15 segundos cambios de frecuencia de transmisión de manera que desestabilicen el sistema, momento en el que inicia el proceso de reconfiguración propuesto.

Escenarios de falla típicos presentan frecuencias de transmisión fuera de la región de planificabilidad lo que resulta en degradación de desempeño. En la figura 7.17 se muestra un escenario típico con falla, donde el valor de las frecuencias de transmisión de los nodos sensores están fuera de la región de planificabilidad donde el sistema es planificable.

El valor de la IAE al utilizar las frecuencias de la tabla 7.6 es de 42 para el seguimiento de la señal del ángulo de cabeceo y 31 para el seguimiento de la señal del ángulo de guiñado.

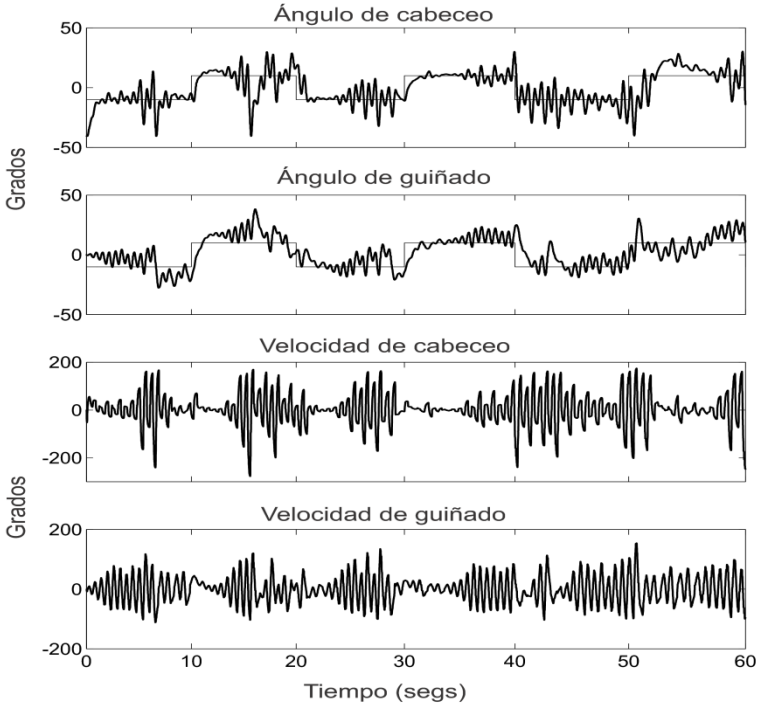

Figura 7.17: Escenario típico con falla en el que las frecuencias de transmisión están fuera de la región de planificabilidad.

Tabla 7.6: Frecuencias de transmisión de datos nominales.

Agente	f_i	p_i
s_1	40	0.025
s_2	250	0.004
s_3	100	0.01
s_4	50	0.02

Escenarios con falla y reconfiguración. Al controlar las frecuencia de transmisión de datos por medio de un esquema LQR, se espera llevar tales frecuencias dentro de la región de planificabilidad en la cual el rendimiento del sistema es el aceptable. La figura 7.18 muestra la dinámica del modelo de frecuencias de transmisión de datos controladas por un controlador LQR.

Para realizar la reconfiguración de las frecuencias de transmisión, se indujo un conjunto de frecuencias arbitrarias durante dos periodos de tiempo de 15 segundos cada uno, el controlador inicia entonces y modifica las frecuencias usando el esquema de control LQR planteado anteriormente. La figura 7.19 muestra la respuesta del sistema al reconfigurar las frecuencias de transmisión de datos.

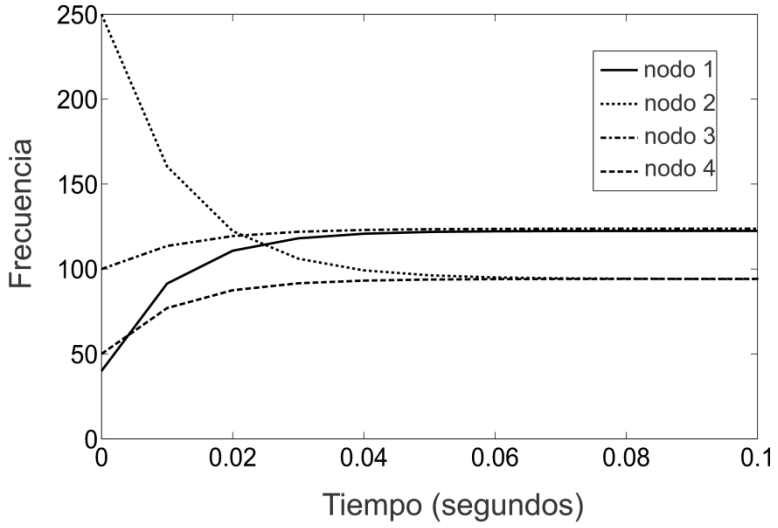

Figura 7.18: Respuesta de las frecuencias de transmisión de datos controladas por un controlador LQR.

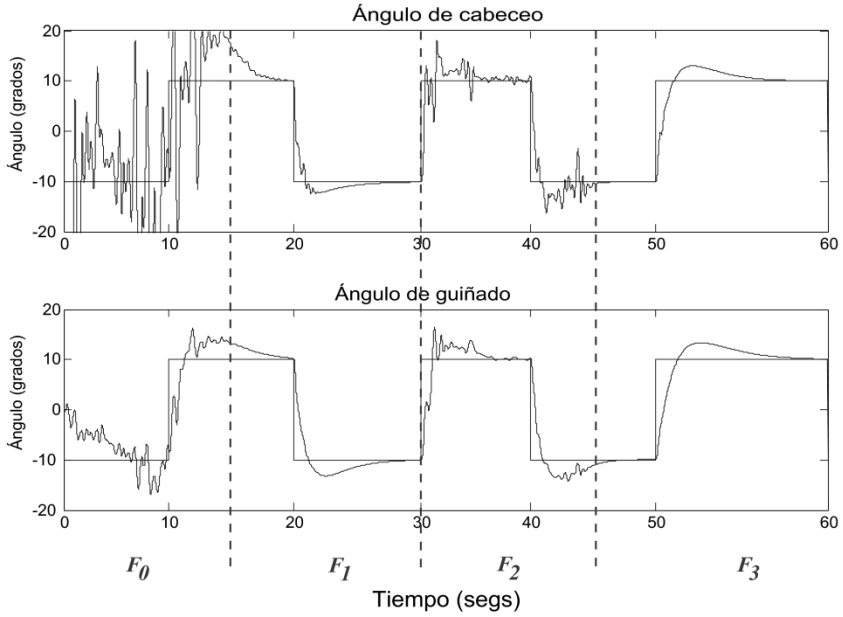

Figura 7.19: Escenario típico con falla y reconfiguración. En ciertos intervalos de tiempo las frecuencias de transmisión están fuera de la región de planificabilidad.

El conjunto de frecuencias nominales iniciales (F_0) están fuera del rango de planificabilidad. Durante 15 segundos los sensores transmiten con estas frecuencias. En el segundo 15 el nodo planificador cambia las frecuencia de transmisión iniciales después de ejecutar el control LQR del modelo de frecuencias para dar lugar a un nuevo conjunto de frecuencias (F_1) con las que se ejecuta el sistema durante 15 segundos. En el segundo 30 se induce un cambio arbitrario de frecuencia igual para todos los sensores (F_2)fuera de la región de planificabilidad. El sistema se ejecuta con esta frecuencia del segundo 30 al 45.

Finalmente en el segundo 45 el modelo de control se activa para reconfigurar las frecuencias y obtener un nuevo conjunto de frecuencias (F_3) para cada agente sensor con las que finaliza la ejecución. El valor de la IAE en este escenario tiene un valor de 30 para el seguimiento de la señal del ángulo de cabeceo y 22 para el seguimiento de la señal del ángulo de guiñado. La tabla 7.7 muestra las frecuencias de transmisión de cada agente sensor por cada intervalo de 15 segundos.

Tabla 7.7: Valores de las frecuencias de transmisión reconfiguradas con base al modelo de control de frecuencias.

	F_0	F_1	F_2	F_3
agente	[0−15)	[15−30)	[30−45)	[45−60)
a_1	45	66	30	58
a_2	250	45	30	66
a_3	100	37	30	58
a_4	50	58	30	66

La figura 7.20 muestra el comportamiento del sistema al inducir un ruido aleatorio en el cambio de frecuencias controladas.

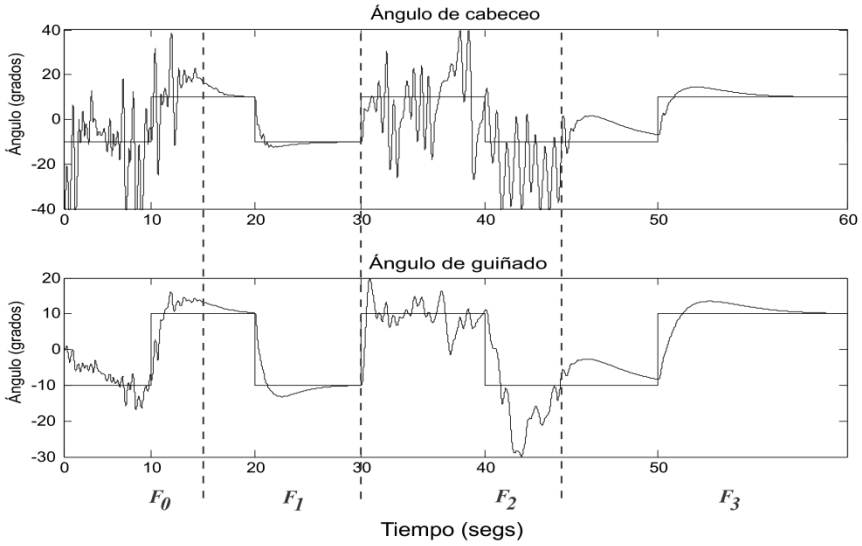

Figura 7.20: Escenario típico con falla y reconfiguración. En ciertos intervalos de tiempo las frecuencias de transmisión están fuera de la región de planificabilidad.

La efectividad del modelo de control de frecuencias es ligeramente menor que el escenario previo. No obstante, la respuesta del sistema se mantiene aún aceptable. El conjunto de frecuencias nominales iniciales (F_0) están fuera del rango de planificabilidad, durante 15 segundos el sistema transmite con esta tasa. En el segundo 15 el nodo planificador cambia las

frecuencia de transmisión anteriores al ejecutar el cálculo de nuevas frecuencias por medio del control LQR para dar lugar a un nuevo conjunto de frecuencias (F_1) con las que se ejecutará el sistema del segundo 15 hasta el segundo 30. En el segundo 30 se induce un cambio aleatorio de frecuencias (F_2) para cada agente sensor, el sistema se ejecuta con esta frecuencia del segundo 30 al 45. Finalmente en el segundo 45 el modelo de control se activa para reconfigurar las frecuencias y obtener un nuevo conjunto de frecuencias (F_3) para cada agente sensor con las que finalizará la ejecución. El valor de la IAE en este escenario tuvo un valor de 32.6 para el seguimiento de la señal del ángulo de cabeceo y 23 para el seguimiento de la señal del ángulo de guiñado. La tabla 7.8 muestra las frecuencias de transmisión de cada agente sensor por cada intervalo de 15 segundos.

Tabla 7.8: Valores de las frecuencias de transmisión reconfiguradas con base al modelo de control de frecuencias.

	F_0	F_1	F_2	F_3
agente	[0−15)	[15−30)	[30−45)	[45−60)
s_1	90	66	90	58
s_2	59	45	59	66
s_3	23	37	23	58
s_4	48	58	48	66

7.3 Resumen

Un efectivo intercambio de datos, entendido como la elección de una tasa de transmisión que utilice al máximo la red sin que provoque saturación, impacta de manera positiva en el desempeño de control. En esta sección se mostró la implementación de los esquemas de reconfiguración propuestos. En el caso de estudio elegido se probó la reconfiguración dinámica de los elementos de planificación en distintos escenarios. Se mostró la implementación de la propuesta de reconfiguración con base en el modelo de frecuencias de transmisión de datos, la cual modifica el tráfico de la red de comunicación. El control LQR ajusta de manera efectiva el conjunto de frecuencias sin que el cálculo de la ley de control y la reconfiguración misma impacte en la transición. Es de resaltar que esta última reconfiguración disminuye el índice de desempeño provocada por una utilización de la red fuera de su ancho de banda o debido a bajo muestreo.

Capítulo 7

Conclusiones y Trabajo Futuro

En este trabajo se ha desarrollado el estudio de la reconfiguración dinámica de un sistema de control en red bajo restricciones de tiempo real. El enfoque utilizado para plantear la reconfiguración se basa los sistemas multiagente debido a la convergencia de elementos individuales en un sistema distribuido.

La reconfiguración, entendida como una transición que modifica la estructura de un sistema y que tiene como consecuencia una nueva representación de sus estados, es una alternativa eficaz para el manejo de fallas, tanto a nivel de software como a nivel de hardware. El esquema de reconfiguración parte de plantear operaciones de modificación, migración, adición, remoción de entidades. Esta propuesta fueron manejadas dichas operaciones a nivel de software.

Con base en las restricciones temporales que plantean los sistemas de tiempo real es necesario hacer el análisis de planificabilidad de escenarios de reconfiguración para asegurar que cada mecanismo utilizado para tal efecto tenga el mínimo impacto durante su acción.

Una parte fundamental del estudio de la reconfiguración corresponde a conocer el impacto que provoca el periodo de muestreo elegido en un sistema de control en red. En esta propuesta se estudia por medio de simulaciones la relación que existe el periodo de muestreo de un sistema de control en red y un índice de desempeño como lo es la integral del valor absoluto del error de seguimiento. Se concluyó que esta relación asocia periodos de muestreo pequeños con una carga grande de datos que se transmiten por medio del canal de comunicación, lo cual provoca en redes con ancho de banda limitada latencias prolongadas y/o pérdida de datos que disminuyen la calidad del control; por otra parte, periodos de muestreo grandes generan pocos datos lo que empobrecen el cálculo de una ley de control eficaz. En este trabajo se muestra que las propuestas de reconfiguración con base en las frecuencias de transmisión de datos en sistemas distribuidos en tiempo real constituyen una herramienta eficaz para planificar el medio de comunicación común. Con esto se cumple el objetivo de conocer el impacto reconfigurar sistemas dinámicos en tiempo real.

El primer esquema de reconfiguración propuesto consiste en construir una arquitectura multiagente que, con base al trabajo cooperativo, sea capaz de tomar decisiones acordadas para llevar acabo un tipo particular de reconfiguración. Ante distintos escenarios con falla se prueban esquemas de reconfiguración como disminución de transmisión de datos, uso de algoritmos de enmascaramiento o suspensión total de transmisión. Es de subrayar que el uso

de algoritmos de enmascaramiento resulta ser una alternativa poco eficaz, queda por demostrar que el intercambio de datos surgido de la necesidad de utilizar una unidad de voto disminuye la efectividad de esta estrategia. La disminución de la actividad de los sensores que presentan fallas es una mejor alternativa. Una condición necesaria para cada estrategia de reconfiguración consiste en asegurar que el retardo provocado por dicha estrategia quede acotado dentro de las restricciones de planificación.

Contribuciones. La mayor aportación de este trabajo es considerar a las frecuencias de transmisión de datos de los sensores de un sistema de control en red como un subsistema lineal invariante en el tiempo que puede ser controlable. Se observó que la planificación del subsistema de frecuencias de transmisión de datos conduce a la estabilidad del sistema, el índice de desempeño que se elige es la integral del valor absoluto del error de seguimiento. El modelado de este subsistema de frecuencias se basa en las relaciones entre las frecuencias mínimas y máximas entre las cuales el sistema es planificable y por tanto tiene un mejor índice de desempeño. Por medio de simulaciones numéricas se obtienen estas cotas de transmisión de datos tomando en cuenta la relación del índice de desempeño contra el periodo de muestreo que es el inverso de la frecuencia de transmisión.

Debido a su sencillez en el diseño e implementación se elige un esquema de control LQR para controlar el subsistema de frecuencias de transmisión sabiendo que el sistema es acotado en su respuesta por los límites de frecuencia, así como por su consumo energético en términos de la velocidad de transición. Claro es que dicha restricción de diseño requiere ser estudiada con elementos de comunicación más flexibles.

Las simulaciones numéricas realizadas en el caso de estudio muestran la viabilidad de la reconfiguración de las frecuencias de transmisión debido a que cada cambio en las frecuencias está acotada por una región de planificabilidad donde el desempeño del sistema es aceptable. No obstante, las cotas máximas y mínimas de las frecuencias entre las cuales es posible transmitir sin saturar la red o deteriorar el nivel del control son particulares al caso de estudio.

Por tanto, este trabajo ofrece una alternativa a la necesidad de responder dinámicamente frente a variaciones en los retardos de tiempo y el efecto que esto provoca en los sistemas distribuidos en tiempo real sin hacer un control propio de los retardos, ofrece la perspectiva de conjugar la arquitectura multiagente y la naturaleza propia de los sistemas distribuidos para obtener un beneficio en favor de los mecanismos de reconfiguración, innovador en el sentido que se propone un método dinámico para planificar el medio de comunicación considerando limitantes en ancho de banda y las restricciones temporales.

Trabajo futuro. Queda como trabajo futuro el desarrollar una metodología para conocer los límites de las frecuencias de transmisión que acoten la cantidad de datos que pueden ser transmitidos por el canal de comunicación sin que impacte el desempeño. En este mismo sentido la reconfiguración planteada se realiza de manera dinámica. Sin embargo, en esta propuesta un planificador centralizado es el que realiza el cálculo de las nuevas frecuencias y ejecuta el cambio, por lo tanto es deseable que dicha modificación se calcule de manera descentralizada y ejecutada localmente en cada nodo o agente.

Se sugiere que la continuación de este trabajo se oriente a reconfigurar sistemas distribuidos que contengan varios lazos de control que comparten un mismo medio de comunicación. Cada lazo de control demandará ancho de banda lo cual representa un reto mayor en la planificación de la red. Esto implica, en el caso de reconfiguración de frecuencias

de transmisión, una disminución de las regiones de planificabilidad para cada uno de los lazos de control involucrados. Por otra parte, la reconfiguración bajo escenarios con falla sería motivo de mayor análisis si se utiliza redundancia de sensores, puesto que aumentaría la cantidad de datos a transmitirse con mayor número de ciclos de control.

Otra línea de investigación que queda abierta para próximos trabajos es la reconfiguración ante la presencia de incertidumbres en redes no dedicadas. En el caso de Ethernet, el tráfico externo al NCS es variable e impacta al rendimiento; añadir los factores de tráfico externo e incertidumbre a la reconfiguración representa un reto nuevo a partir de este trabajo.

Apéndices

Apéndice A

Cota de Utilización de RM

En esta apartado se deduce la máxima cota de utilización para el algoritmo de *Rate Monotonic* (RM) con base en la exposición de los trabajos de Liu y Layland (1973) y Butazzo (2005).

El algoritmo *Rate Monotonic* es una regla muy simple la cual asiga prioridades a las tareas de acuerdo a su tasa de solicitud [1]. Específicamente, tareas con el periodo más corto (mayor demanda) tendrán la prioridad más alta. Dado que los periodos son constantes, RM es un algoritmo de prioridades fijas, esto es, que las prioridades son asignadas a las tareas antes de la ejecución y no cambian durante el tiempo. Además, RM es expulsable, es decir que la ejecución actual de una tarea puede se interrumpida por una nueva tarea que llega con un periodo más corto. Liu y Layland (1973) mostró que RM es subóptimo entre todos los algoritmos de prioridades fijas en el sentido que no hay otro algoritmo de prioridad fija capaz de planificar un conjunto de tareas que no sea planificable por RM, en caso que dicha planificación exista. Liu y Layland (1973) también derivaron la mínima cota máxima U_{mx} de utilización del procesador para un conjunto Γ de *n* tareas periódicas. Butazzo (2005) muestra diversos aspectos a considerar para obtener esta cota de utilización.

Optimalidad. Para mostrar que el algoritmo RM es subóptimo en el sentido planteado anteriormente, Butazzo (2005) muestra que un instante crítico para cualquier tarea se produce cuando la tareas se ha liberado simultáneamente con todas la demás tareas de mayor prioridad. Sea $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ el conjunto de tareas periodicas ordenadas de manera ascendente respecto a sus periodos, con τ_n la tarea con periodo más grande. De conformidad con RM, τ_n será la tarea con la prioridad más baja.

En la figura A.1 se muestra como tiempo de respuesta de la tarea τ_n es retrasado por la interferencia de τ_i tareas con mayor prioridad. En la figura A.2 se muestra que al avanzar el tiempo de liberación de τ_i puede incrementarse el tiempo para que se termine de ejecutar τ_n. Como consecuencia, el tiempo de respuesta de τ_i es más grande cuando su liberación es simultánea con τ_i. Repitiendo este procedimiento para todas las τ_i, $i = 2, \cdots, n-1$ se prueba que el peor tiempo de respuesta de la tarea ocurre cuando se liber simultáneamente con las tareas de mayor prioridad.

[1] La tasa de solicitud de tiempo de procesamiento está determinado por el valor del periodo, por tanto pueden considerarse como conceptos equivalentes

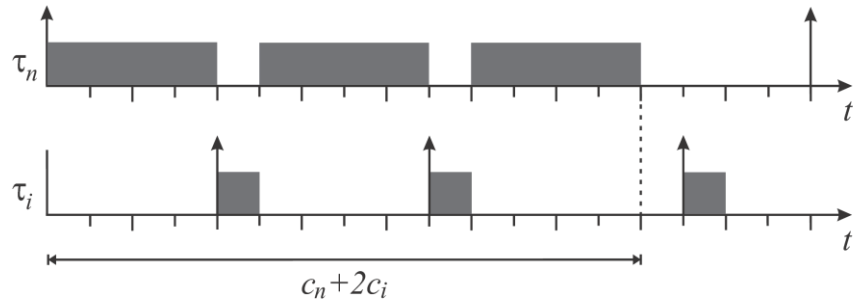

Figura A.1: El tiempo de respuesta de la tarea τ_n es retrasado por la interferencia de τ_i con mayor prioridad.

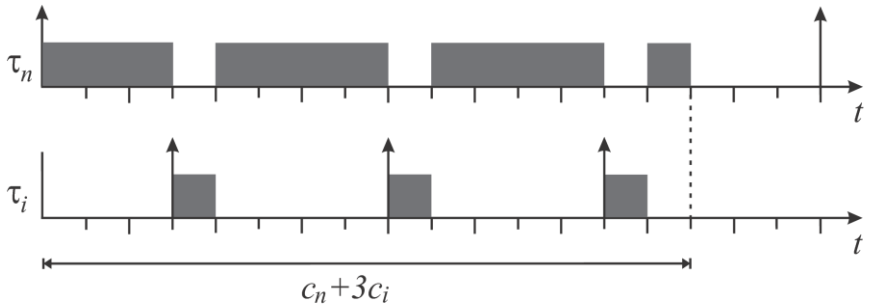

Figura A.2: El tiempo de respuesta de la tarea τ_n. La interferencia puede incrementarse a medida que disminuye el tiempo de liberación dτ_ie.

La primera consecuencia de este resultado es que planificabilidad de tareas puede comprobarse precisamente en sus instantes críticos. En concreto, si todas las tareas son factibles de ser planificadas, entonces el conjunto de tareas es planificable en cualquier otra condición. Basado en este resultado, la optimalidad de RM se justifica si se muestra que:

> *Si un conjunto de tareas es planificable bajo una asignación de prioridades arbitraria, entonces también es planificable por RM*.

Para probar este resultado, [**¡Error! No se encuentra el origen de la referencia.**] considera un conjunto de dos tareas periódicas y τ_1 y τ_2, con $p_1 < p_2$. Si las prioridades no se asignan de acuerdo con RM, entonces la tarea τ_2 recibirá la prioridad más alta. Esta situación se ilustra en la figura A.3 de donde es fácil observar que, en los instantes críticos, la planificación es factible si la desigualdad siguiente se satisface:

$$c_1 + c_2 < p_1 \tag{A.1}$$

Por otra parte, si las prioridades se asignan de acuerdo con RM, la tarea τ_1 tendrá mayor prioridad. En esta situación, que se muestra en las figuras A.4 y A.5, con el fin de garantizar una planificación factible, dos casos deben ser considerados. Sea $F = \lfloor {p_2}/{p_1} \rfloor$ el número[2] de periodos de τ_1 completamente contenidos en p_2.

[2] $\lfloor x \rfloor$ denota el entero más grande menor o igual que *x* y $\lceil x \rceil$ denota el entero más pequeño mayor o igual que *x*

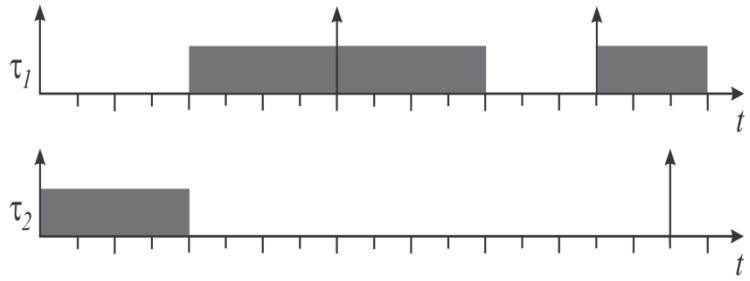

Figura A.3: Planificación de tareas por algún algoritmo diferente a RM.

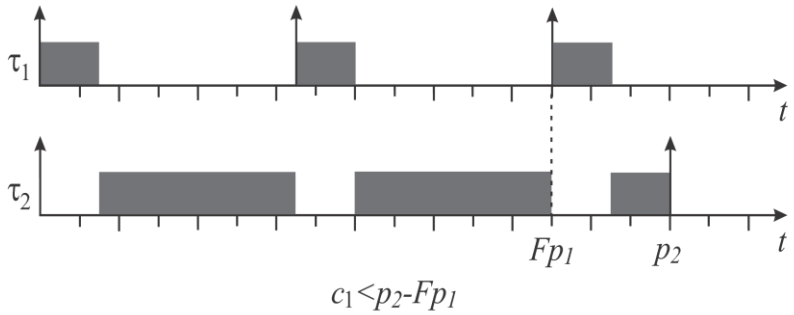

Figura A.4: Planificación de tareas por RM. Caso

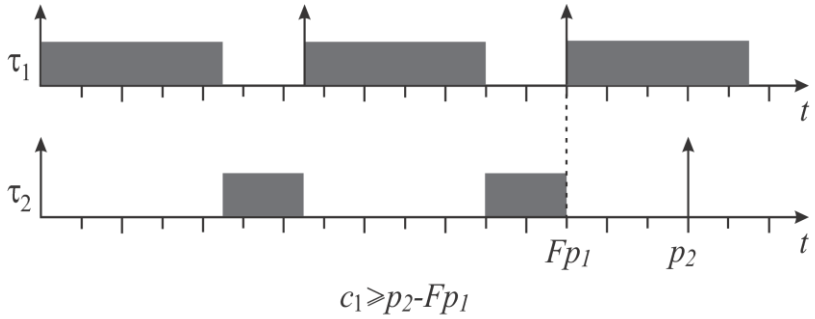

Figura A.5: Planificación de tareas por RM. Caso II.

Butazzo (2005) expone los dos casos de análisis de la siguiente forma:

- **Caso I.** El tiempo de ejecución c_1 es lo suficientemente pequeño tal que todas la solicitudes de τ_1 dentro de la zona de tiempo crítico de τ_2 se completan antes de la segunda solicitud de τ_2. Esto es, $c_1 \leq p_2 - Fp_1$
  En este caso, de la figura A.4 se puede ver que el conjunto de tareas es planificable si:

$$(F+1)c_1 + c_2 \leq p_2 \tag{A.2}$$

  Muestra que la desigualdad (A.1) implica (A.2). Al multiplicar por ambos de la igualdad (A.1) por *F* obtiene

$$Fc_1 + Fc_2 \leq Fp_1$$

y dado que $F \geq 1$, se tiene que

$$Fc_1 + c_2 \leq Fc_1 + Fc_2 \leq Fp_1$$

Butazzo (2005) además añade c_1 a cada lado y tiene que

$$(F+1)c_1 + c_2 \leq Fp_1 + c_1$$

Dado que asume que $c_1 \leq p_2 - Fp_1$ obtiene

$$(F+1)c_1 + c_2 \leq Fp_1 + c_1 \leq p_2$$

con lo que se satisface A.2.

- **Caso II.** La ejecución de la última solicitud de τ_1 en la zona de tiempo crítico de τ_2 superpone la segunda solicitud de τ_2. Esto es, $c_1 \geq p_2 - Fp_1$En este caso, de la figura A.5 se puede observar que el conjunto de tareas es planificable si

$$Fc_1 + c_2 \leq Fp_1 \tag{A.3}$$

Nuevamente, la desigualdad (A.1) implica a (A.3). Al mutiplicar ambos lados de (A.1) por *F* obtiene

$$Fc_1 + Fc_2 \leq Fp_1$$

y, dado que $F \geq 1$, se puede escribir

$$Fc_1 + c_2 \leq Fc_1 + Fc_2 \leq Fp_1$$

lo que satisface la (A.3)

Con lo anterior, Butazzo (2005) demuestra básicamente que dadas dos tareas periodicas τ_1 y τ_2, con periodos $p_1 < p_2$, si la planificación es factible mediante una asignación de prioridad arbitraria, entonces también es factible por RM. Esto muestra que RM es subuóptimo. Butazzo (2005) menciona que el resultado anterior puede extenderse a un conjunto de *n* tareas periódicas. El cálculo de la mínima cota superior U_{mx} del factor de utilización del procesador para el algoritmo RM que realiza Butazzo (2005) se muestra a continuación.

Cálculo de U_{mx} de dos tareas. De principio, se consideran dos tareas τ_1 yτ_2, con periodos $p_1 < p_2$ con la finalidad de calcular U_{mx} para RM. Entonces, se tiene que:

- La asignación de prioridades se hace conforme a RM, esto es que τ_1 es la tarea con prioridad más alta.
- Es necesario calcular la cota máxima de utilización U_x para el conjunto colocando tiempos de cómputo de tareas para lograr la mayor utilización del procesador.

- Minimizar la cota máxima U_x con respecto a todos los demás parámetros de tarea.

Como hizo anteriormente, Butazzo (2005) define $F = \lfloor {p_2}/{p_1} \rfloor$ como el número de periodos de τ_1 contenidos completamente en p_2. Sin pérdida de generalidad, el tiempo de cómputo c_2 se ajusta para utilizar el procesador en su totalidad. De nuevo, Butazzo (2005)considera dos casos:

Caso I. El tiempo de ejecución c_1 es lo suficientemente pequeño tal que todas la solicitudes de τ_1 dentro de la zona de tiempo crítico de τ_2 se completan antes de la segunda solicitud deτ_2. Esto es, $c_1 \leq p_2 - Fp_1$

En esta situación, descrita en la figura A.6, el valor más grande posible de c_2 es

$$c_2 = p_2 - c_1(F+1)$$

y la correspondiente cota superior U_x es

$$U_x = \frac{c_1}{p_1} + \frac{c_2}{p_2} = \frac{c_1}{p_1} + \frac{p_2 - c_1(F+1)}{p_2}$$

$$= 1 + \frac{c_1}{p_1} - \frac{c_1}{p_2}(F+1) =$$

$$= 1 + \frac{c_1}{p_2}\left[\frac{p_2}{p_1} - (F+1)\right]$$

Debido a que la cantidad entre corchetes es negativa, U_x es decreciente en c_1 y dado que $c_1 \leq p_2 - Fp_1$, el mínimo valor de U_x ocurre cuando

$$c_1 = p_2 - Fp_1$$

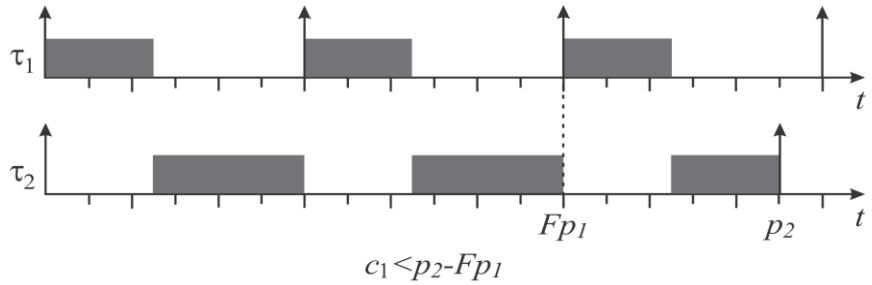

Figura A.6: La segunda solicitud de τ_2 es liberada cuando τ_1 está en espera.

- **Caso II.** La ejecución de la última solicitud de τ_1 en la zona de tiempo crítico de τ_2 se superpone la segunda solicitud de τ_2. Esto es, $c_1 \geq p_2 - Fp_1$
  En esta circunstancia, descrita en la figura A.7, el valor más grande posible para c_2 es

$$c_2 = (p_1 - c_1)F$$

y la correspondiente cota superior U_x es

$$U_x = \frac{c_1}{p_1} + \frac{c_2}{p_2} = \frac{c_1}{p_1} + \frac{(p_1 - c_1)F}{p_2}$$

$$= \frac{p_1}{p_2}F - \frac{c_1}{p_1} - \frac{c_1}{p_2}F =$$

$$= \frac{p_1}{p_2}F + \frac{c_1}{p_2}\left[\frac{p_2}{p_1} - F\right] \tag{A.4}$$

Debido a que la cantidad entre los corchetes es positiva, U_x es creciente en c_1 y dado que $c_1 \geq p_2 - Fp_1$, el mínimo valor de U_x ocurre cuando

$$c_1 = p_2 - Fp_1$$

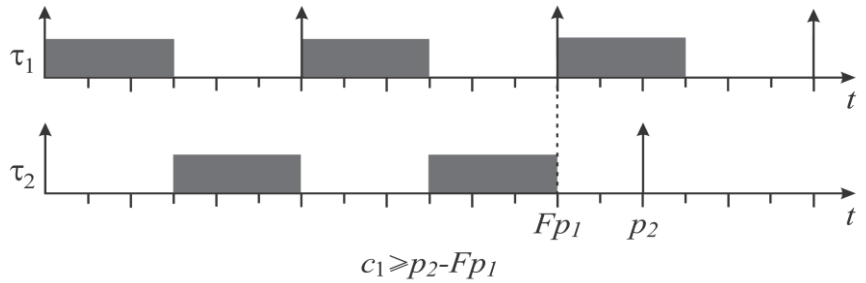

Figura A.7: La segunda solicitud de τ_2 es liberada cuando τ_1 está activa.

Butazzo (2005) menciona que en ambos casos, el valor mínimo de U_x ocurre para

$$c_1 = p_2 - Fp_1$$

Por tanto, usando el mínimo valor de c_1, de la ecuación A.4 obtiene:

$$U_x = \frac{p_1}{p_2}F + \frac{c_1}{p_2}\left(\frac{p_2}{p_1} - F\right) =$$

$$= \frac{p_1}{p_2}F + \frac{(p_2 - p_1F)}{p_2}\left(\frac{p_2}{p_1} - F\right) =$$

$$= \frac{p_1}{p_2}\left[F + \left(\frac{p_2}{p_1} - F\right)\left(\frac{p_2}{p_1} - F\right)\right] \tag{A.5}$$

Butazzo (2005) simplifica la notación haciendo $G = {p_2}/{p_1} - F$. Entonces tiene:

$$U_x = \frac{p_1}{p_2}(F + G^2) = \frac{(F + G^2)}{{p_2}/{p_1}}$$

$$= \frac{(F + G^2)}{\left({p_2}/{p_1} - F\right) + F} = \frac{F + G^2}{F + G}$$

$$= \frac{(F + G) - (G - G^2)}{F + G} = 1 - \frac{G(1 - G)}{F + G} \tag{A.6}$$

Dado que $0 \leq G \leq 1$, el término $G(1-G)$es positivo. Por tanto, U_x es creciente en *F*. Como consecuencia, el valor mínimo de U_x ocurre para el mínimo valor de *F*, es decir, *F*=1. Así,

$$U_x = \frac{1+G^2}{1+G} \tag{A.7}$$

Minimizando U_x sobre *G*, Butazzo (2005) obtiene:

$$\frac{dU_x}{dG} = \frac{2G(1+G)-(1+G^2)}{(1+G)^2} =$$

$$\frac{G^2+2G-1}{(1+G)^2}$$

y $\frac{dU_x}{dG} = 0$ para $G^2+2G-1=0$, la cual obtiene dos soluciones: $G_1 = -1-\sqrt{2}$ y $G_1 = -1+\sqrt{2}$.

Dado que $0 \leq G \leq 1$, la solución negativa $G = G_1$ se descarta. Así, de la ecuación (A.7), la mínima cota máxima de U_x está dada por $G = G_2$:

$$U_{mx} = \frac{1+\left(\sqrt{2}+1\right)^2}{1+\left(\sqrt{2}+1\right)} = \frac{4-2\sqrt{2}}{\sqrt{2}} = 2\left(\sqrt{2}-1\right)$$

Esto es,

$$U_{mx} = 2\left(2^{1/2}-1\right) \approx 0.83 \tag{A.8}$$

Butazzo (2005) hace notar que si p_2 es múltiplo de p_1, $G = 0$ y la utilización de procesador se hace 1. En general, el factor de utilización de dos tareas puede ser calculada como una función de la relación $k = {}^{p_2}/_{p_1}$. Para una *F* dada, la ecuación (A.5) puede escribirse como

$$U_x = \frac{F+(k-F)^2}{k} = k-2F+\frac{F(F+1)}{k}$$

Minimizando *U* sobre *k* Butazzo (2005)obtiene

$$\frac{dU_x}{dG} = 1-\frac{F(F+1)}{k^2}$$

y $\frac{dU_x}{dk} = 0$ para $.k^* = \sqrt{F(F+1)}$. Por tanto, para una *F* dada, el valor mínimo de U_x es

$$U_x^* = 2\sqrt{F(F+1)}-F$$

En la tabla A.1, Butazzo (2005) reúne algunos valores de k^* y U_x^* como función de *F*.

F	k_i^*	U_i^*
1	$\sqrt{2}$	0.828
2	$\sqrt{6}$	0.899
3	$\sqrt{12}$	0.928
4	$\sqrt{20}$	0.944
5	30	0.954

Tabla A.1: Valores de k_i^* y U_i^* como funciones de *F*.

Cálculo U_{mx} para n tareas. Partiendo del cálculo anterior, Butazzo (2005) determina las condiciones que permiten calcular la mínima cota máxima del factor de utilización del procesador como sigue:

$$\begin{cases} F = 1 \\ c_1 = p_2 - Fp_1 \\ c_2 = (p_1 - c_1)F \end{cases}$$

lo cual puede ser reescrito como:

$$\begin{cases} p_1 < p_2 < 2p_1 \\ c_1 = p_2 - p_1 \\ c_2 = 2p_2 - p_1 \end{cases}$$

Generalizando para cualquier conjunto arbitrario de *n* tareas, las peores situaciones para la planificación del conjunto de tareas que utiliza totalmente el procesador son:

$$\begin{cases} p_1 < p_n < 2p_1 \\ c_1 = p_2 - p_1 \\ c_2 = p_2 - p_1 \\ \cdots \\ c_{n-1} = p_n - p_{n-1} \\ c_n = p_1 - (c_1 + c_2 + \cdots + c_{n-1}) = 2p_1 - p_n \end{cases}$$

Así, el factor de utilización del procesador es:

$$U = \frac{p_2 - p_1}{p_1} + \frac{p_3 - p_2}{p_2} + \cdots + \frac{p_n - p_{n-1}}{p_{n-1}} + \frac{2p_1 - p_n}{p_n}$$

Butazzo (2005) define

$$R_i = \frac{p_{i+1}}{p_i}$$

y hace notar que con

$$R_1 R_2 \cdots R_{n-1} = \frac{p_n}{p_1}$$

el factor de utilización puede ser escrito como

$$U = \sum_{i=1}^{n-1} R_i + \frac{2}{R_1 R_2 \cdots R_{n-1}} - n$$

Para minimizar U sobre R_i, $i = 1, \cdots, n-1$. Butazzo (2005) hace

$$\frac{\partial U}{\partial R_k} = 1 - \frac{2}{R_i^2 \left(\prod_{i \neq k}^{n-1} R_i\right)}$$

Así, define $P = R_1 R_2 \ldots R_{n-1}$, U es mínimo cuando

$$\begin{cases} R_1 P = 2 \\ R_2 P = 2 \\ R_{n-1} P = 2 \end{cases}$$

Esto es que, cuando todos las R_i tienen el mismo valor:

$$R_1 = R_2 = \cdots = R_{n-1} = 2^{1/n}$$

Butazzo (2005) sustituye el valor anterior en *U* para obtener:

$$U_{mx} = (n-1)2^{1/2} + \frac{2}{2^{(1-1/n)}} - n =$$
$$= n2^{1/n} - 2^{1/n} + 2^{1/n} - n =$$
$$= n\left(2^{1/n} - 1\right)$$

Por tanto, para un conjunto arbitrario de tareas periódicas, la mínima cota máxima del factor de utilización del procesador bajo la planificación por RM es:

$$U_{mx} = n\left(2^{1/n} - 1\right) \tag{A.9}$$

Esta cota decrece cuando *n* aumenta, esto se muestra en la tabla A.2.

n	U_{mx}	n	U_{mx}
1	1.000	6	0.735
2	0.823	7	0.729
3	0.780	8	0.724
4	0.757	9	0.721
5	0.743	10	0.718

Tabla A.2: Valores de U_{mx} como función de *n*

Para valores grandes de *n*, la mínima cota máxima converge a:

$$U_{mx} = \ln(2) \approx 0.69$$

Butazzo (2005) usa la sustitución $y = \left(2^{1/n} - 1\right)$, para obtener $n = \frac{\ln(2)}{\ln(y+1)}$ y por tanto

$$\lim_{n\to\infty} n\left(2^{1/n} - 1\right) = (\ln(2)) \lim_{n\to\infty} \frac{y}{ln(y+1)}$$

Utilizando la regla de L'Hopital:

$$\lim_{y\to\infty} \frac{y}{ln(y+1)} + \lim_{y\to\infty} \frac{1}{1/(y+1)} = \lim_{n\to\infty} (y+1) = 1$$

para obtener finalmente que

$$\lim_{n\to\infty} U_{mx} = \ln(2)$$

Apéndice B

Regulación Cuadrática Lineal

En esta sección se introduce a la forma general del problema de la regulación cuadrática lineal (LQR) con base en el trabajo de Hespanha (2009).

Dado un sistema continuo, lineal e invariante en el tiempo:

$$\begin{array}{ll} \dot{x} = Ax + Bu & x \in \mathbb{R}^n, u \in \mathbb{R}^k \\ y = Cx & y \in \mathbb{R}^m \end{array}$$

el problema de *regulación cuadrática lineal* (LQR[1]) consiste en encontrar control $u(t), t \in [0, \infty]$ que conduzca al valor más pequeño posible el siguiente criterio:

$$J_{LQR} = \int_0^\infty y(t)'Qy(t) + u(t)'Ru(t)dt \tag{B.1}$$

donde *Q* y *R* son matrices positivas definidas. El término:

$$\int_0^\infty y(t)'Qy(t)dt$$

provee una medida de la *energía de la salida* y el término:

$$\int_0^\infty u(t)'Ru(t)dt$$

provee una medida de la *energía de la señal de control*. En la regulación cuadrática lineal se busca un controlador que minimice ambas energía. No obstante, el decremento de la energía de la salida controlada requerirá una señal de control grande y una señal de control pequeña conducirá a grandes salidas de control.

El papel de la constante de las matrices *Q* y *R* es estipular un promedio entre estos dos objetivos:

1. Cuando *R* es mucho más grande que *Q*, la forma más efectiva para decrementar J_{LQR} es emplear una entrada de control pequeña con el costo de obtener salidas grandes.

[1] Linear Quadratic Regulator

2. Cuando *R* es mucho más pequeña que *Q*, la forma más efectiva de decrementar J_{LQR} es obtner una salida muy pequeña, no obstante de conseguirse con el costo de emplear una entrada de control grande.

Para el sistema B.1, existe una función $H\big((x(\cdot);u(\cdot)\big)$ que involucra a la entrada y al estado del sistema. Para cualquer *P* simétrica la función *H* de la forma:

$$H\big((x(\cdot);u(\cdot)\big) = \int_0^\infty (Ax(t) + Bu(t))'Px(t) + x(t)'P(Ax(t) + Bu(t))\,dt$$

su valor no dependerá de una señal de entrada específica $u(\cdot)$si no únicamente de la condición inicial $x(0)$ al ser tomada como solución del sistema.

Supóngase que es posible expresar el criterio *J* para ser minimizado al elegir una *u*(·) en la siguiente forma:

$$J = H\big((x(\cdot);u(\cdot)\big) + \int_0^\infty \Lambda(x(t),u(t))dt \tag{B.2}$$

La expresión:

$$J = H\big((x(\cdot);u(\cdot)\big) + \int_0^\infty x'(A'P + PA + C'QC' - PBR^{-1}B'P)x + (u' + x'K')R(u + Kx)dt$$

seleccionando la matriz *P* tal que:

$$A'P + PA + C'QC' - PBR^{-1}B'P = 0$$

se obtiene una expresión como B.2 con:

$$\Lambda = (x,u) = u'x'K'R(u + Kx)$$

la cual tiene un mínimo igual a cero para:

$$u = -Kx \qquad K = R^{-1}B'P \tag{B.3}$$

resultando en el siguiente lazo cerrado:

$$\dot{x} = Ax + BKx = (A - BR^{-1}B'P)x$$

Por lo tanto la ley de control B.3 estabiliza el sistema a lazo cerrado mientras minimiza el criterio LQR B.1.

Apéndice C

TrueTime

Para Cervin y otros (2003), hoy en día los sistemas de control embebidos a menudo contienen kernel de tiempo real multitarea, la utilización de redes. Además, los algoritmos de control en tiempo real y el software de control son considerados a la par. Por tanto, son necesarias nuevas herramientas informáticas para el diseño en tiempo real y de control se necesitan. Los lazos de control en red se componen de sensores, actuadores y cálculos para el control que reciden en diferentes nodos; dentro de los nodos individuales de controladores se implementan como una o varias tareas en un microprocesador con un sistema operativo de tiempo real. Este sistema operativo utiliza multiprogramación para ejecutar diversas tareas. Ancho de banda de la red de comunicación y tiempo del procesador son considerados como recursos compartidos por los cuales las tareas compiten. Diferentes fuentes de determinismo temporal como tiempos de ejecución de las tareas o los retardos de comunicación afectan el rendimiento de control, sin embargo este no determinismo se puede reducir mediante la elección de una plataforma de implementación adecuada. Las limitantes de la plataforma de implementación deben ser consideradas en sistemas con recursos informáticos limitados Cervin y otros (2003), por lo tanto, algunas herramientas están disponibles para analizar y simular los efectos de las restricciones temporales en el rendimiento del control.

TrueTime Cervin y otros (2003, 2007); Henriksson y otros (2006a); Ohlin y otros (2007) versión 1.5 es un simulador de red y sistema de control embebido basado en Matlab/Simulink, desarrollado en la Universidad de Lund en 1999. TrueTime se utiliza como una plataforma experimental para la investigación de sistemas dinámicos de control en tiempo real. Es posible estudiar los esquemas de compensación que ajustan el algoritmo de control basados en las mediciones de las variaciones temporales. TrueTime permite estudiar de manera general y detallada modelos de tiempo para NCS. TrueTime se puede utilizar para investigar cómo el no determinismo afecta al comportamiento del sistema, para desarrollar nuevos esquemas para ajustar dinámicamente los parámetros de control, para experimentar nuevos enfoques como codiseño de control y planificación de la red, y para simular sistemas de control basados tareas activadas por eventos.

El software consiste en una biblioteca de bloques basados en Simulink (figura C.1), el *bloque de kernel* simula un kernel de tiempo real que ejecuta tareas definidas por el usuario y el manejo de interrupciones. Para comunicar bloques de kernel (nodos), se utilizan varios *bloques de red*, esto hace muy sencillo desarrollar simulaciones de NCS.

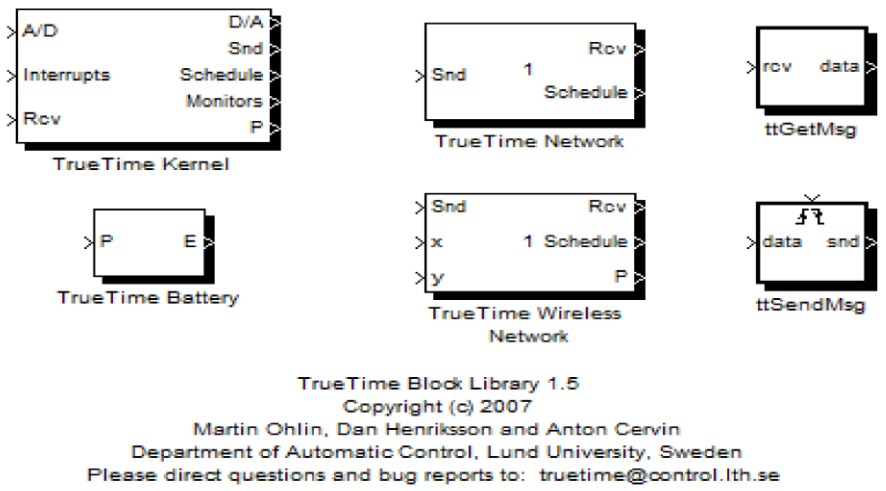

Figura C.1: Librería de Bloques de TrueTime 1.5

C.1 Bloque de Kernel

Un nodo de cómputo es simulado usando el bloque de kernel de TrueTime, este nodo tiene un kernel genérico de tiempo real, convertidores A/D y D/A, e interfaces de red. Un *script* de inicialización se utiliza para configurar el bloque, en este script es posible crear varios objetos como tareas, temporizadores, menejadores de interrupciones, semáforos, etc., estos objetos establecer el software que se ejecuta en el nodo. El kernel llama continuamente a las funciones de las tareas y de los manejadores de interrupción. Código en Matlab o en C++ puede utilizarse para escribir scripts de inicialización y los códigos de objetos. La principal ventaja de utilizar C es la velocidad; no obstante, los archivos de Matlab (archivos .m) son muy fáciles de utilizar. Varias políticas de planificación pueden utiilzarse en los bloques kernel, estas pueden ser de prioridad fija o de prioridad dinámica.

La *tarea* el la construcción principal en el entorno TrueTime, este objeto se utiliza para simular actividades periódicas y aperiódicas, por ejemplo, el controlador, las tareas de entrada-salida pueden ser periódicas, mientras que la comunicación y el controlador manejado por evento pueden ser tareas aperiodicas. Un conjunto de atributos y una función de código definen una tarea; atributos como nombre, tiempo de liberación, peor caso de tiempo de ejecución, el presupuesto, los plazos relativos y absolutos, la prioridad, el periodo. El tiempo de liberación y el plazo absoluto son atributos que constantemente son actualizados por el kernel durante la simulación, mientras que el periodo y la prioridad se mantienen constantes, aunque pueden cambiar por llamadas a funciones primitivas del kernel. Un ejemplo de la definición de una tarea es la siguiente:

```
function sensor_init(arg)
% Initialize TrueTime kernel
ttInitKernel(1, 0, 'prioFP'); %Inputs,Outputs,FixedPriority
% Create sensor task
offset = 0;
prio = 1;
period = 0.010;
```

```
ttCreatePeriodicTask('sens_task', offset, period, ...
prio, 'senscode', data);
```

La primitiva del kernel `ttInitkernel ()` inicializa un nodo. El kernel es inicializado especificando el número de canales A/D y D/A, y la política de planificación. La función prediseñada de prioridad `prioFP` especifica la planificación de prioridad fija. Los planificadores *Rate Monotonic* `prioRM`, *Earliest Deadline First* `prioEDF` y *Deadline Monotonic* `prioRM` son políticas de planificación prediseñadas adicionales.

Las *interrupciones* pueden ser generadas de dos formas. Una interrupción externa es asociada con uno de los canales de interrupción externo del bloque de computadora, cuando la señal del correspondiente canal cambia de valor la interrupción se dispara. La utilidad de este tipo de interrupción consiste en simular controladores distribuidos que se ejecutan cuando las mediciones llegan de la red. La interrupción interna funciona contruyendo temporizadores, cuando un temporizador tinaliza la interrupción se activa. Un manejador de interrupciones definido por el usuario es planificado cuando un interrupción interna o externa ocurre. Una interrupción, así como una tarea, realizan acciones y son planificadas con prioridad al máximo nivel. Un manejador de interrupciones es definido por su nombre, prioridad y código de función Cervin y otros (2007).]. Un ejemplo de la definición de un manejador de interrupciones es el siguiente:

```
%Initialize the network
ttCreateInterruptHandler('nw_handler1',prio,'msgRcvSensor');
ttInitNetwork(4, 'nw_handler1');
```

Cervin y otros (2007). mencionan que la ejecución de la simulación ocurre a tres distintos niveles de prioridad: el de interrupción (máxima prioridad), de kernel y de tarea(mínima prioridad). La ejecución puede ser expulsiva o no expulsiva. A nivel de interrupción, los manejadores de interrupción son planificados de acuerdo a prioridades fijas. A nivel de tareas, se usa la planificación por prioridad dinámica. Para cada punto de planificación, la prioridad de las tareas está dada por una función de prioridad definida por el usuario que es una función de atributos de tareas. Esto hace sencillo simular diferentes políticas de planificación.

El *código* asociado con las tareas y el menjo de interrupciones es planificado y ejecutado por el kernel durante el progreso de la simulación. El código puede dividirse en segmentos que interactúan con otras tareas y con el ambiente al inicio de cada segmento de código. El tiempo de ejecución de cada segmento es regresado al código de función y puede ser modelado como constante, aleatorio o dependiente de algún dato. Durante la simulación, el kernel guarda el segmento acutal y llama el código de la función con los argumentos apropiados. La ejecución reanuda en el siguiente segmento cuando la tarea se ha ejecutado el tiempo asociado en el segmento anterior. Un ejemplo del código de función es el siguiente:

```
function [exectime, data] = senscode(seg, data)
switch seg,
case 1,
% Receive data from analog input
data.y = ttAnalogIn(1);
exectime = 0.0005;
case 2,
```

```
% Shows the current time
ttCurrentTime
% Send message (80 bits) to node 3 (controller)
ttSendMsg(3, data, 80)
exectime = 0.0004;
case 3,
exectime = -1; % finished
end
```

Apéndice D

Modelos de Matlab/Simulink

Las simulaciones del helicóptero de dos hélices se realizaron en Matlab/Simulink. A continuación se presentan los modelos en simulink del lazo cerrado de control, el modelo lineal, referencias de voltaje y ángulos y controlador FF+LQR+I.

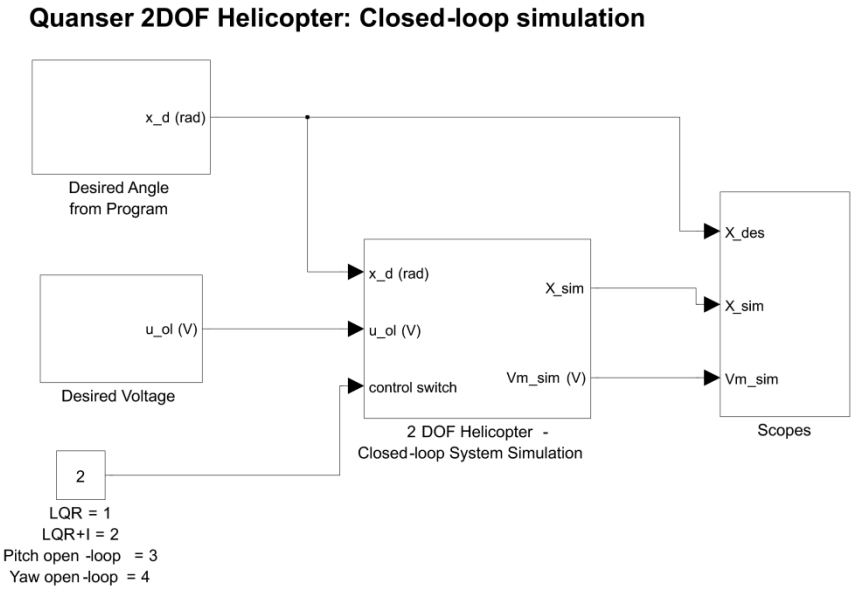

Figura D.1: Helicóptero Quanser 2DOF: Simulación lazo cerrado.

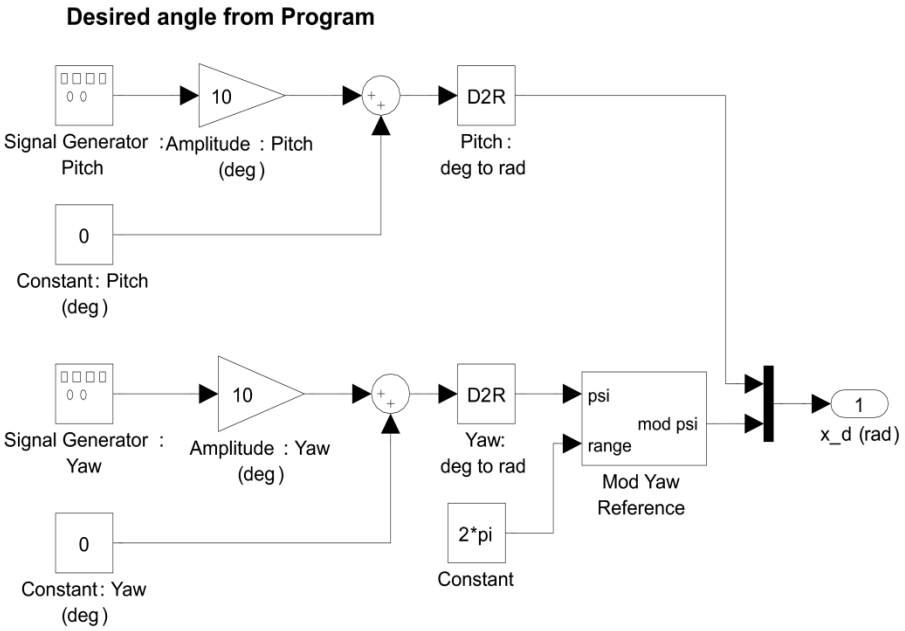

Figura D.2: Ángulo deseado de los ejes de cabeceo y guiñado

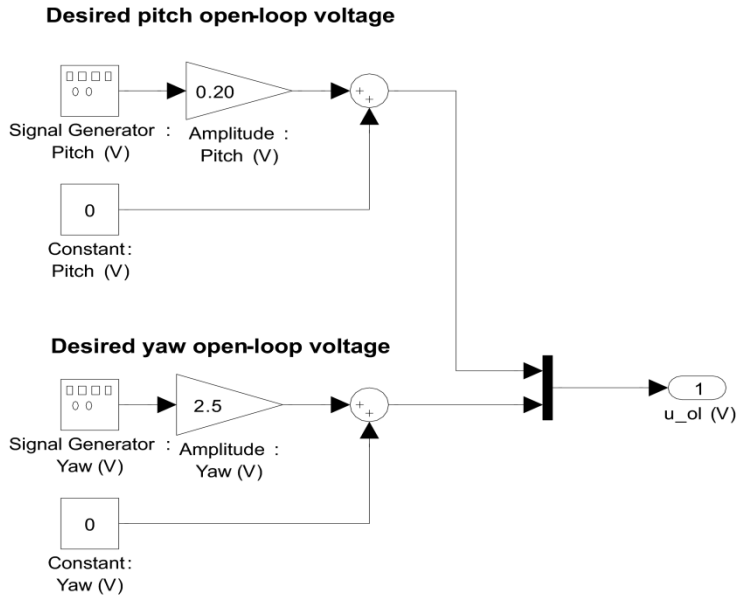

Figura D.3: Voltaje deseado de los ejes de cabeceo y guiñado.

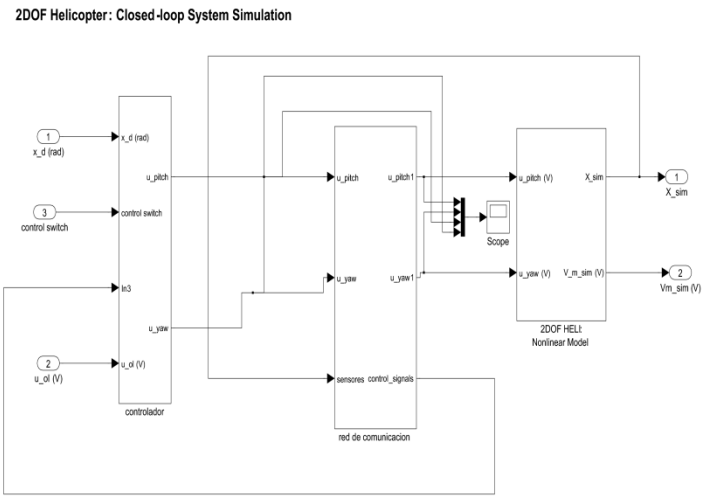

Figura D.4: Subsistema de lazo cerrado de control.

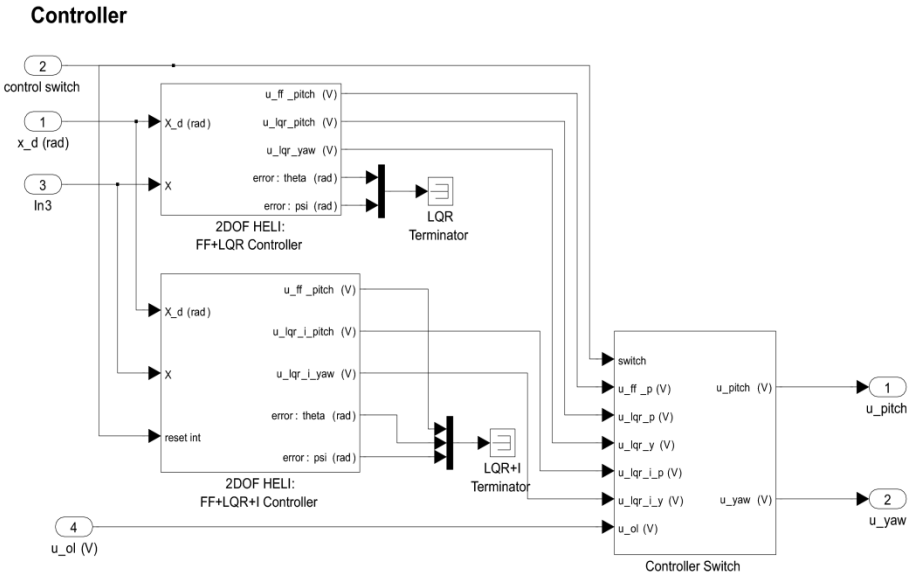

Figura D.5: Controladores del helicóptero.

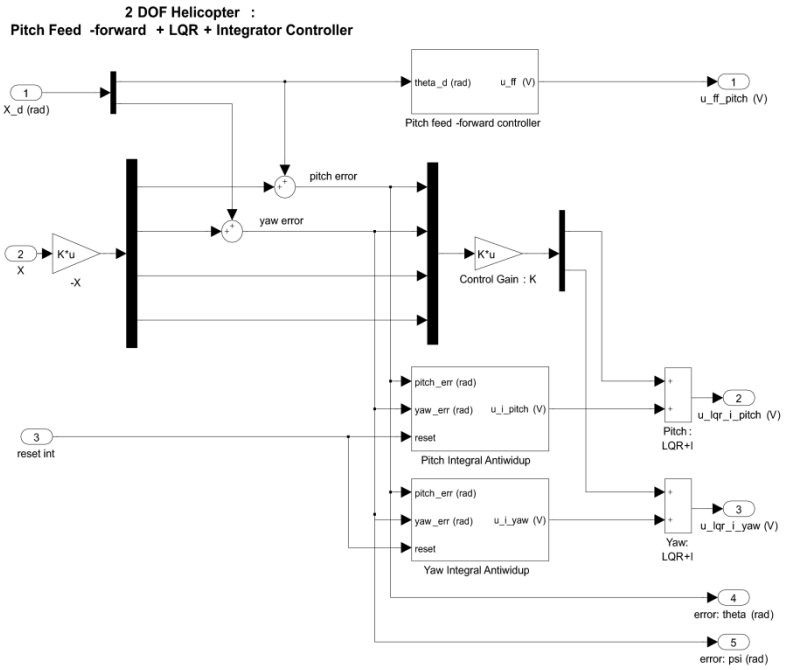

Figura D.6: Controlador FF+LQR+I.

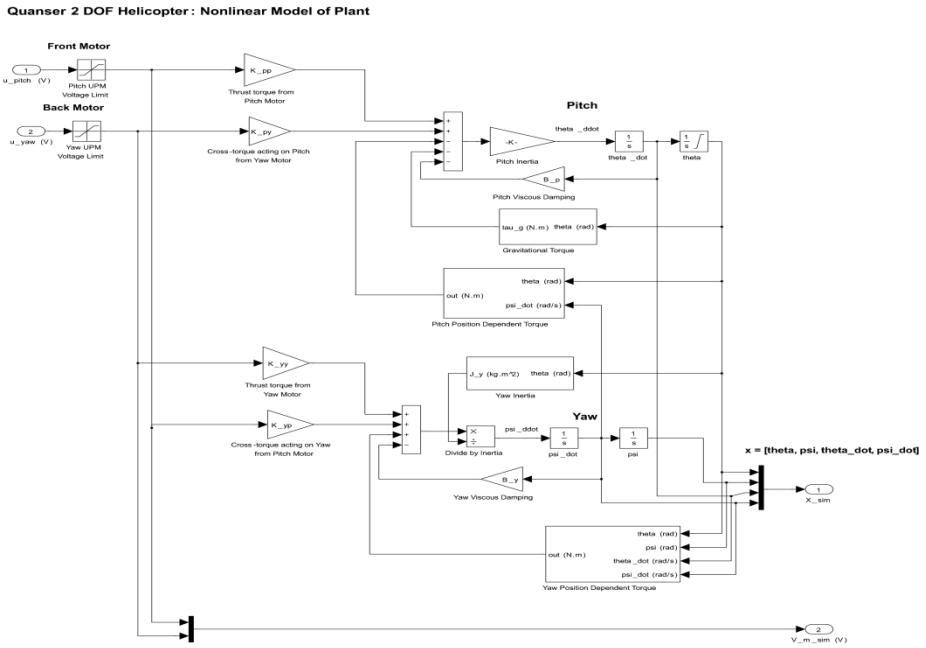

Figura D.7: Modelo no lineal.

Apéndice E

Software

Las simulaciones numéricas se realizaron por medio de la construcción de un sistema de control en red en tiempo real. El modelo fue implementado en Simulink/Matlab y se utilizó el simulador Truetime como herramienta de simulación de tiempo real. Cada bloque de nodo tenía una función de inicialización del kernel y una funcíón de ejecución de tareas. A continuación se muestran los códigos de estas funciones para cada nodo del sistema:

E.1 Funciones de inicialización

```
global sensor_init_period sensor_code actuator_init_period ...
       actuator_code actuator_receive_time ...
       controler_period controler_time scheduler_period
       extra_task red dispersion xs xs1 xs2 xs3 ni

%sobre-muestreo 0.0010 o menor, bajo-muetreo 0.0270 o mayor;
%c=1;

for dispersion=0.1:0.1
c=1;
tabla=0;
for i=1:16
    cont(i)=0;
end
cont_tot=0;
for red=0.025:0.025
d=sprintf('dat_contadores_%d_%d.mat',red*1000,10*(dispersion));
d=sprintf('dat_cont_fault2_%d_%d.mat',red*1000,10*(dispersion))
 save(d,'cont','cont_tot');
 dispersion=0.1;


% periodos y consumos sensores
sensor_init_period(2)=red*(1+dispersion);
sensor_code(2)=0.001;
sensor_init_period(3)=red*(1-dispersion);
```

```
sensor_code(3)=0.001;
sensor_init_period(4)=red*(1-dispersion*1.1);
sensor_code(4)=0.001;
sensor_init_period(5)=red*(1+dispersion*1.1);
sensor_code(5)=0.001;

% periodos y consumos actuadores
actuator_init_period(14)=10;
actuator_code(14)=0.0001;
actuator_receive_time(14)=0.001;
actuator_init_period(15)=10;
actuator_code(15)=0.0001;
actuator_receive_time(15)=0.001;

% periodos y consumos controlador
controler_period=red;
controler_time=0.00001;

% periodo planificador
scheduler_period=red;%0.01;

ni = 1;

%tareas adicionales taskpcn (node period consumtion)

extra_task(1,1,1)=5;     extra_task(2,1,1)=5;
extra_task(1,1,2)=.01;   extra_task(2,1,2)=.01;
extra_task(1,2,1)=8;   extra_task(2,2,1)=8;
extra_task(1,2,2)=.01;  extra_task(2,2,2)=.01;
extra_task(1,3,1)=6;   extra_task(2,3,1)=6;
extra_task(1,3,2)=.01;  extra_task(2,3,2)=.01;
extra_task(1,4,1)=7;   extra_task(2,4,1)=7;
extra_task(1,4,2)=.01;  extra_task(2,4,2)=.01;

extra_task(3,1,1)=5;   extra_task(4,1,1)=5;
extra_task(3,1,2)=.01;  extra_task(4,1,1)=5;
extra_task(3,2,1)=8;   extra_task(4,2,1)=.01;
extra_task(3,2,2)=.01;  extra_task(4,2,2)=8;
extra_task(3,3,1)=6;   extra_task(4,3,1)=.01;
extra_task(3,3,2)=.01;  extra_task(4,3,2)=6;
extra_task(3,4,1)=7;   extra_task(4,4,1)=.01;
extra_task(3,4,2)=.01;  extra_task(4,4,2)=7;


extra_task(5,1,1)=5;    extra_task(6,1,1)=5;
extra_task(5,1,2)=.01;   extra_task(6,1,2)=1;
extra_task(5,2,1)=8;   extra_task(6,2,1)=8;
extra_task(5,2,2)=.01;  extra_task(6,2,2)=1;
```

```
extra_task(5,3,1)=6;   extra_task(6,3,1)=6;
extra_task(5,3,2)=.01;  extra_task(6,3,2)=1;
extra_task(5,4,1)=7;   extra_task(6,4,1)=7;
extra_task(5,4,2)=.01;  extra_task(6,4,2)=1;


sim('rtds4');

  tabla(c)=errcuad(qq,1);
  c=c+1;
end
  d_t=sprintf('dat_tabla_%d.mat',10*(dispersion));
  d_t=sprintf('dat_tabla_fault2_%d.mat',10*(dispersion));
  save(d_t,'tabla');
end


function controler_init(arg)
global controler_period
ttInitKernel(2, 4, 'prioFP'); % nbrOfInputs, nbrOfOutputs, fixed
priority
prio = 1;
offset=0;
period=controler_period;
ttCreatePeriodicTask('pid_task1', offset, period, prio,
'controlercode',arg);

% Initialize network
prio=2;
ttCreateInterruptHandler('nw_handler1', prio, 'controler_rcv',arg);
ttInitNetwork(arg, 'nw_handler1'); % node #(arg) in the network

 Initialize network activity if node 1 replaced with block
 if arg==1
     data=arg;
     deadline = 2;
     prio = 1;
     ttCreateTask('pid_task9', deadline, prio, 'initnet', data);
     ttCreateJob('pid_task9');
 end
missed deadline
prio=4;
ttAttachDLHandler(taskname, handlername)
ttCreateInterruptHandler('dl_handler1', prio, 'deadline1');
ttAttachDLHandler('pid_task2', 'dl_handler1');

function sensor_init(arg)
global sensor_init_period extra_task;
```

```
ttInitKernel(2, 0, 'prioEDF'); % nbrOfInputs, nbrOfOutputs, fixed
priority
prio = 1;
offset=0;
sarg = sens_arg(arg);
data.rsm = 0;     % contador de ejecuciones(round)
data.ssm = 0;     % sumatoria de datos sensados
data.rsn = 0;
data.ssn = 0;
data.i1 = 1;      %contadores de señal
data.i2 = 1;
data.i3 = 1;
data.ns = arg;   % argumento, id de procesador
data.bs = sarg;  % bloque de sensado de procesador

period = sensor_init_period(sarg);
ttCreatePeriodicTask(['pid_task',arg], offset, period, prio,
'sensorcode',data);

% Initialize network
prio=9;
ttCreateInterruptHandler('nw_handler1', prio, 'sens_rcv',arg);
ttInitNetwork(arg, 'nw_handler1'); % node(# arg) in the network

% Tareas adicionales extra tasks
prio=2;
for i=1:4
    n=arg-1;
    period=extra_task(n,i,1);
    c=extra_task(n,i,2);
    tn=arg*10+i;
    ttCreatePeriodicTask(['pid_task',tn], 0, period, prio,
'extrataskcode',c);
end
function scheduler_init(arg)
global scheduler_period con
ttInitKernel(1, 0, 'prioFP'); % nbrOfInputs, nbrOfOutputs, fixed
priority
con=0;
prio = 1;
offset=0;
period=scheduler_period;
ttCreatePeriodicTask('pid_task1', offset, period, prio,
'schedulercode',arg);

% Initialize network
prio=2;
ttCreateInterruptHandler('nw_handler1', prio, 'scheduler_rcv',arg);
```

```
ttInitNetwork(arg, 'nw_handler1'); % node #(arg) in the network


function actuator_init(arg)
global actuator_init_period extra_task;
ttInitKernel(0, 1, 'prioFP'); % nbrOfInputs, nbrOfOutputs, fixed
priority
prio = 2;
offset=0;
period=actuator_init_period(arg);
    ttCreatePeriodicTask(['pid_task',arg], offset, period, prio,
'actcode',arg);

% Initialize network
prio=1;
ttCreateInterruptHandler('nw_handler1', prio, 'act_rcv',arg);
ttInitNetwork(arg, 'nw_handler1'); % node(# arg) in the network
% extra tasks

prio=2;
for i=1:4
    n=arg-1;
    period=extra_task(n,i,1);
    c=extra_task(n,i,2);
    tn=arg*10+i;
    ttCreatePeriodicTask(['pid_task',tn], 0, period, prio,
'extrataskcode',c);
end

function [exectime, data] = actcode(seg, data)
global actuator_code
switch seg,
    case 1,
        exectime=actuator_code(data);
    case 2,
        exectime = -1;
end
```

E.2 Control LQR de las frecuencias de transmisión

```
function [y]= frecnewj(r)

%entra un vector r con 4 frec iniciales y la referencia

%frecuencias mínimas
fm1 = 100 ;
```

```
fm2 = 100 ;
fm3 = 100 ;
fm4 = 100 ;

%frecuencias máximas
fx1 =  300;
fx2 =  300;
fx3 =  300;
fx4 =  300;

%frecuencias reales
fr1 = r(1);      %fr1 =  5500;
fr2 = r(2);      %fr2 =  5000;
fr3 = r(3);      %fr3 =  2500;
fr4 = r(4);      %fr4 =  3000;

%frecuencia base: máximo comun divisor de las frecuencias mínimas
fmb= gcd(fm4,gcd(fm3,gcd(fm2,fm1)));


A1 = [ fmb/fm1  fm2/fm1  fm3/fm1  fm4/fm1  0;
       fm1/fm2  fmb/fm2  fm3/fm2  fm4/fm2  0;
       fm1/fm3  fm2/fm3  fmb/fm3  fm4/fm3  0;
       fm1/fm4  fm2/fm4  fm3/fm4  fmb/fm3  0;
       0.001     0.001   0.001    0.001    1 ];

A=[ 2 5 10 2 1;
    3 9 6 1  3;
    8 6 1 1  4;
    2 3 4 1  0;
    1 3 5 2  2;]



B = [fx1  0     0      0     0;
       0   fx2   0      0     0;
       0     0    fx3   0     0;
       0     0     0    fx4   0;
       1     1     1      1     1 ]

C = [1 0 0 0 0;
     0 1 0 0 0;
     0 0 1 0 0;
     0 0 0 1 0;
     0 0 0 0 1]

 D= [ 0 0 0 0 0;
      0 0 0 0 0;
```

```
      0 0 0 0 0;
      0 0 0 0 0;
      0 0 0 0 0 ]

  %Polos del sistema
p=eig(A)

%sys=ss(A,B,C,D)
%step(sys)

%Linear Quadratic Regulator
Q1=C'*C;

Q = [10000 0 0  0 0;
     0 10000 0  0 0;
     0 0 10000  0 0;
     0 0  0 10000 0;
     0 0  0  0 10000];

R = [1 0 0 0 0;
     0 1 0 0 0;
     0 0 1 0 0;
     0 0 0 1 0;
     0 0 0 0 1];
K = lqr(A,B,Q,R)
Ac = [(A-B*K)]
e= eig(A-B*K)
Bc = [B*K];
Cc = [C];
Dc = [D];
%y=Cc*u';


% T=0:0.01:50;
T=0:0.01:1;
[m,n]=size(T);

%{
U=[u(1)*ones(1,n);
   u(2)*ones(1,n);
   u(3)*ones(1,n);
   u(4)*ones(1,n);
   0.001*ones(1,n)];
%}

U=[r(5)*ones(1,n);
   r(5)*ones(1,n);
   r(5)*ones(1,n);
```

```
   r(5)*ones(1,n);
   0.001*ones(1,n)];

Xc= [r(1) r(2) r(3) r(4) 0];

[Y,X]=lsim(Ac,Bc,Cc,Dc,U,T,Xc);

%sys=ss(A,B,C,D)
%step(sys)
figure
plot(T,abs(Y(:,1)),'k-' );
hold on
plot(T,abs(Y(:,2)),'k:');
plot(T,abs(Y(:,3)),'k-.');
plot(T,abs(Y(:,4)),'k--');
title ('Frequency Transition Control');
legend('agent1','agent2','agent3','agent4'); %,'consume')
```

Apéndice E

Software

Como resultado del presente trabajo se obtuvo la siguiente producción científica:

Capítulos de Libro

- Esquivel-Flores O., and Benitez-Perez H., "Frequency Transition based upon dynamic Consensus for a Distributed System", *Mexican International Conference on Artificial Intelligence*, Lecture Notes on Computer Science LNAI 6437, pp: 399-409; México, 2010.
- Oscar Esquivel-Flores and Héctor Benítez-Pérez, "Issues on Communication Network Scheduling using Numerical Simulations", *Numerical Simulations, InTech*, 2012.

Artículos en Revistas Arbitradas

- O. Esquivel-Flores, H. Benítez-Pérez, P. E. Mendez-Monroy, Antonio Menéndez, "Frequency Transition for scheduling management using dynamic system approximation for a kind of NCS", *ICIC Express Letters B*, vol 1, no. 1, pp. 93-98, 2010.
- Esquivel-Flores, O., Benitez-Perez H., "Study of Reconfigurable Distributed Systems in Real-Time Environment Through Multi Agents using a Strategy of Schedulling Approximation"; *ICIC Express Letters ICIC-EL*, vol. 5, no. 10,pp. 3591- 3596, 2011.
- Esquivel-Flores O., and Benitez-Perez H.; "Reconfiguración Dinámica de Sistemas Distribuidos en Tiempo Real basada en Agentes", *Revista Iberoamericana de Automática e Informática industrial*, vol. 9, no. 3, pp. 300-313, 2012.
- Esquivel-Flores, O., Benitez-Perez H.; "Frequency Transmission Control of local Networked Control Systems Approach", *Journal of Applied Research and Technology*, vol. 10, no. 3, 2012.
- O. Esquivel-Flores, H. Benítez-Pérez, P. Méndez-Monroy, J. Ortega-Arjona, "Bounded Communication Between Nodes of a Networked Control System as a Strategy of Scheduling", *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 6, pp. 481-502, 2012.

Artículos en Memorias con Arbitraje Internacional

- Esquivel-Flores O., and Benitez-Perez H., "Scheduling based upon frequency transition. A dynamical model of reltaionships between agents in a NCS", *International Conference on Agents and Artificial Intelligence*, vol. 2, pp: 389-393, Italia, 2011.

- Esquivel-Flores O., Benitez-Perez H., “A Study of Dynamic System Approximation for Frequency Transition based upon Scheduling Management Approach", *Proceedings of Eurocast*, pp: 56-58, España, 2011.
- O. Esquivel-Flores, H. Benítez-Pérez, “Distributed Scheduling Based Upon Frequency Transmission Using Two Dynamic Controlled Plants and a PID Controller”, *6th International Conference on Soft Computing and Intelligent Systems & 13th International Symposium on Advanced Intelligent Systems*, Kobe, Japón, 2012.
- O. Esquivel-Flores, H. Benítez-Pérez, “Fault Tolerant Scheduling Strategy in a NCS Based on Frequency Transition”, *6th International Conference on Soft Computing and Intelligent Systems & 13th International Symposium on Advanced Intelligent Systems*, Kobe, Japón, 2012.

Artículos en Memorias con Arbitraje Nacional

- Esquivel-Flores, O., Benítez-Pérez H. “Study Of Reconfigurable Distributed Systems In Real-Time Environments Through Multiagents", *IEEE ENC*, PhD WorkShop, 2009.

Artículos Aceptados

- O. Esquivel-Flores, H. Benítez-Pérez, P. Méndez-Monroy, “Scheduling Strategy Using Frequency Transition for a Helicopter Simulation as a Network Control System Approximation", *Journal on Studies in Informatics And Control. With Emphasis on Useful Applications of Advanced Technology*, 2012.
- H. Benítez-Pérez, Alma Benítez-Pérez, Jorge Ortega-Arjona and O. Esquivel-Flores, "Fuzzy Networked Control Systems design considering scheduling restrictions," *Journal on Advances in Fuzzy Systems*, 2012.

Bibliografía

Alag, S.; Agogino, A. y Morjaria, M.: A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics".Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2001, (15), pp. 307-320.

Albertos, P.; Crespo, A.; Ripoll, I.; Valles, M. y Balbastre, P.: "RT control scheduling to reduce control performance degrading". En: Decision and Control, 2000. Proceedings of the 39th IEEE Conference on, volumen 5, pp. 4889-4894, 2000.

Almeida, J.: Dynamic Recon_guration of Object-Middleware-based Distributed Systems. Tesina o Proyecto, University of Twente, 2001.

Almeida, J.; Wegdam, M.; Van-Sinderen, M. y Nieuwenhuis, L.: "Transparent dynamic reconfiguration for CORBA". En: Distributed Objects and Applications, 2001. DOA '01. Proceedings. 3rd International Symposium on, pp. 197-207, 2001.

Arzen, K.; Bernhardsson, B.; Eker, J.; Cervin, A.; Persson, P.; Nilsson, K. y Sha, L.: "Integrated Control and Scheduling". Informe técnico ISSN 0820-5316, Department of Automatic Control, Lund Institute of Technology, 1999.

Arzen, K.; Cervin, A.; Eker, J. y Sha, L.: "An introduction to control and scheduling co-design". En: Decision and Control, 2000. Proceedings of the 39th IEEE Conference on, volumen 5, pp. 4865-4870, 2000.

Astrom, K. y Witternmark, B.: Computer Controlled Systmes: Theory and Design. Prentice Hall, 1997.

Audsley, N.; Bums, A.; Richardson, M. y Welling, A.: "Hard real-time scheduling: the deadline monotonic approach". En: Proc. of IEEE Workshop on Real-Time Operating Systems and Software, pp. 133-137, 1991.

Baillieul, J. y Antsaklis, J.: "Control and communication Challenges in Networked Real-Time Systems". En: Prooceedings of the IEEE, volumen 95, 2007.

Balbastre, P.: Modelos de tareas para la integraci_on del control y la planificación en sistemas de tiempo real. Tesis doctoral, Universidad Polit_ecnica de Valencia, 2002.

Belcastro, C.: "Application of failure detection, identification, and accommodation methods for improved aircraft safety". En: American Control Conference, 2001. Proceedings of the 2001, volumen 4, pp. 2623-2624, 2001.

Benítez-Pérez, H.; Cárdenas-Flores, F. y F., García-Nocetti: "An Implementation of Recon_gurable Network Control based upon Automata Proposal". International Journal of Computers, Communications & Control , 2007a, II(4), pp. 314-327.

Benítez-Pérez, H.; Cárdenas-Flores, F. y García-Nocetti, F.: "Red Reconfigurable

Mediante el Modelo de Control Predictivo para Tres Bandas Transportadoras como Caso de Estudio". Información Tecnológica, 2009, 20(1), pp. 111-127.

Benítez-Pérez, H. y García-Nocetti, F.: "Reconfigurable Distributed Control using Smart Peripheral Elements". Control Engineering Practice, 2003, 11(9), pp. 975-988.

Benítez-Pérez, H. y García-Nocetti, F.: Reconfigurable Distributed Control. Springer Verlag, 2005.

Benítez-Pérez, H.; Garcíaa-Zavala, A. y García-Nocetti, F.: "A Proposal for On-Line Reconfiguration Based upon a Modification of Planning Scheduler and Fuzzy Logic Control Law Response". En: ISSADS, Lecture Notes in Computer Science, pp. 141-152. Springer, 2005.

Benítez-Pérez, H.; Ortega-Arjona, J.; Cárdenas-Flores, F y Quiñones-Reyes, P.: "Reconfiguration control strategy using Takagi-Sugeno model predictive control for network systems - a magnetic levitation case study". Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 2010, 224(8), pp. 1022-1032.

Benítez-Pérez, H.; Ortega-Arjona, J.; Cárdenas-Flores, F y Quiñones-Reyes, P.: "Reconfigurable Takagi Sugeno Fuzzy Logic Control predictive for a Class of Nonlinear System Considering Communication Time Delays". International Journal of Computers, Communications & Control,, 2011, 6(3), pp. 384-399.

Benítez-Pérez, H.; Ortega-Arjona, J. y Latif-Shabgahi, G.: "Definition and Empirical Study of Voters for Redundant Smart Sensor Systems". Computaci_on y Sistemas, 2007b, 11(1), pp. 39-60.

Bond, A. y Gasser, L.: Readings in Distributed Artificial Intelligence. Morgan Kaufmann,1988.

Branicky, M.; Liberatore, V. y Phillips, S.: "Networked control system co-simulation for co-design". En: American Control Conference, 2003. Proceedings of the 2003, volumen 4, pp. 3341-3346, 2003.

Butazzo, G.: Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Springer, 2005.

Castillo, P.; García, P.; R., Lozano y P., Albertos: "Modelado y estabilización de un helicóptero con cuatro rotores". Revista Iberoamericana de Automática e Informática Industrial, 2007, 4(1), pp. 41-57.

Cenjor, A. y García, A.: "Control Basado en Agentes Mejorados con Tecnología Auto-ID".Revista Iberoamericana de Automática e Informática Industrial, 2005, 2(3), pp. 48-60.

Cervin, A.: "Improved scheduling of control tasks". En: Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on, pp. 4-10, 1999.

Cervin, A. y Eker, J.: "Feedback scheduling of control tasks". En: Decision and Control, 2000. Proceedings of the 39th IEEE Conference on, volumen 5, pp. 4871-4876, 2000.

Cervin, A.; Henriksson, D.; Lincoln, B.; Eker, J. y _Arzen, K.-E.: "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime ". Control Systems, IEEE, 2003, 23(3), pp. 16-30.

Cervin, A.; Ohlin, M. y Henriksson, D.: "Simulation of Networked Control Systems Using TrueTime". En: Proc. 3rd International Workshop on Networked Control Systems: Tolerant to Faults, Nancy, France, 2007.

Chen, C.: Linear System Theory and Design. Oxford University Press, 1999.

Cheng, A.: Real-Time Systems. Scheduling, Analysis, and Veri_cation. Wiley Interscience, 2002.

Cork, L. y Walker, R.: "Sensor Fault Detection for UAVs using a Nonlinear Dynamic

Model and the IMM-UKF Algorithm". En: Information, Decision and Control, 2007. IDC '07, pp. 230-235, 2007.

Cork, L.; Walker, R. y Dunn, S.: "Fault Detection, Identi_cation and Accommodation Techniques for Unmanned Airborne Vehicles". En: Australian International Aerospace Congress, Australian International Aerospace Congress (AIAC), Melbourne, 2005.

Corkill, D.: "Collaborating Software: Blackboard and Multiagent Systems & the future". En: Proceedings of the International Lisp Conference, volumen 3, pp. 123-118, 2003.

Coulouris, G.; Dollimore, J.; Kindberg, T. y Blair, G.: Distributed Systems. Addison Wesley, 2012.

Dertouzos, M.: "Control Robotics. The Procedural Control of Physical Processes". Information Processing, 1974, 74.

Driscoll, K.; Hall, B.; Sivencrona, H. y Zumsteg, P.: "Byzantine Fault Tolerance, from Theory to Reality". En: Vasile Palade; Robert Howlett y Lakhmi Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, volumen 2773 de Lecture Notes in Computer Science, pp. 235-248, 2003.

Dubrova, E.: Fault Tolerant Design: An introduction. Kluwer Academic Publishers, 2008. Gambier, A.: "Real-time control systems: a tutorial". En: Control Conference, 2004. 5th Asian, volumen 2, pp. 1024-1031, 2004.

García, A. y Cenjor, A.: "Sistema Heterárquico de Control Basado en Agentes para Sistemas de Fabricación: La Nueva Tecnología PROHA". Revista Iberoamericana de Automática e Informática Industrial, 2007, 4(1), pp. 83-94.

García, C.; Saltarén, R.; López, J. y Aracil, R.: "Desarrollo de una interfaz de Usuario para el Sistema Robótico Multiagente SMART". Revista Iberoamericana de Automática e Informática Industrial, 2010, 7(4), pp. 17-27.

García-Sanz, M.; Elso, J. y Egaña, I.: "Control del ángulo de Cabeceo de un Helicóptero como Benchmark de Diseño de Controladores". Revista Iberoamericana de Automática e Informática Industrial, 2006, 3(2), pp. 111-116.

Gupta, R. y Mo-Yuen, C.: "Networked Control System: Overview and Research Trends". Industrial Electronics, IEEE Transactions on, 2010, 57(7), pp. 2527-2535.

Hadzilacos, V. y Toueg, S.: "A modular approach to the specification and implementation of fault-tolerant broadcasts". Informe técnico, Department of Computer Science, Cornell University, Ithaca N.Y., 1994.

Halevi, Y. y Ray, A.: "Integrated Communication and Control Systems: Part I-Analysis". Journal of Dynamic Systems,Measurement, and Control , 1988a, 4(110), pp. 367-373.

Halevi, Y. y Ray, A.: "Integrated Communication and Control Systems: Part II-Design Considerations". Journal of Dynamic Systems Measurement and Control , 1988b, 4(110).

Henriksson, D.; Cervin, A.; Andersson, M. y _Arz_en, K.E.: "TrueTime: Simulation of Networked Computer Control Systems". En: Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems, Alghero, Italy, 2006a.

Henriksson, D.; Redell, O.; El-Khoury, J.; Cervin, A.; T□orngren, M. y Arz_en, K.E.: "Tools for Real-Time Control Systems Co-Design". En: Hans Hansson (Ed.), ARTES -A network for Real-Time research and graduate Education in Sweden 1997-2006, Department of Information Technology, Uppsala University, Sweden, 2006b.

Heredia, G. y Ollero, A.: "Sensor fault detection in small autonomous helicopters using observer/Kalman filter identifiation". En: IEEE International Conference on Mechatronics, 2009., pp. 1-6, 2009.

Hespanha, J.P.: Linear System Theory. Princeton University Press, 2009.

Huhns, M. y Singh, M.: Readings in Agents. Morgan Kaufmann, 1998.

Jennings, N.: Cooperation in Industrial Multi-Agent Systems. World Scientific, 1994.

Joseph, M. y Pandya, P.: "Finding Response Times in a Real-Time System". BCS Computer Journal, 1986, 29(5), pp. 390-395.

Karimi, A.; Zarafshan, F.; Jantan, A.B.; Ramli, A. y Saripan, M.: "An optimal parallel average voting for fault-tolerant control systems". En: International Conference on Networking and Information Technology (ICNIT),, pp. 360-363, 2010.

Kramer, J. y Magee, J.: "Dynamic con_guration for distributed systems". IEEE Transactions on Software Engineering, 1985, 11(4), pp. 424-436.

Krishna, Shin K., C.: Real time systems. Mc Graw Hill International, 1997.

Kumar, S; Raghavan, V. y Deng, J.: "Medium Access Control Portocols for Ad Hoc Wireless Networks: A Survey". Ad Hoc Networks, 2006, 4, pp. 326-358.

Lamport, L.: "Time, Clcks, and the Ordering of Events in a Distributed System". Communications of the ACM, 1978, 21(7).

Latif-Shabgahi, G.: "A novel algorithm for weighted average voting used in fault tolerant computing systems". Microprocessors and Microsystems, 2004, 28(7), pp. 357-361.

Latif-Shabgahi, G.; Bass, J.M. y Bennett, S.: "History-based weighted average voter: a novel software voting algorithm for fault-tolerant computer systems". En: Parallel and Distributed Processing, 2001. Proceedings. Ninth Euromicro Workshop on, pp. 402-409, 2001.

Latif-Shabgahi, G.; Bennet, S. y Bass, J.M.: "Smoothing voter: a novel voting algorithm for handling multiple errors in fault-tolerant control systems". Microprocessors and Microsystems, 2003, 27(7), pp. 303313.

Lehoczky, J.; Sha, L. y Ding, D.: "The Rate-monotonic Scheduling Algorithm: Exact Characterization and Average Case Behaviour". En: Proc. of IEEE Real-Time Systems Symposium, pp. 166-171, 1989.

Leung, J. y Whitehead, J.: "On the complexity of _xed-priority schedulings of periodic real time tasks". Performance Evaluation, 1982, 4(2), pp. 237-250.

Li, L. y Wang, F.: "Control and Communication Synthesis in Networked Control Systems". International Journal of Intelligent Control and Systems, 2008, 13(2), pp. 81-88.

Lian, F.; Moyne, J. y Tilbury, D.: "Analysis and modeling of networked control systems: MIMO case with multiple time delays". En: American Control Conference, 2001. Proceedings of the 2001, volumen 6, pp. 4306-4312, 2001a.

Lian, F.; Moyne, J. y Tilbury, D.: "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet". Control Systems, IEEE, 2001b, 21(1), pp. 66-83.

Lian, F.; Moyne, J. y Tilbury, D.: "Time Delay Modeling and Sample Time Selection for Networked Control Systems". En: International Mechanical Engineering Congress and Exposition, 2001. Proceedings of ASME-DSC, volumen XX, pp. 1-8, 2001c.

Lian, F.; Moyne, J. y Tilbury, D.:: "Network design consideration for distributed control systems". Control Systems Technology, IEEE Transactions on, 2002, 10(2), pp. 297-307.

Lian, F.; Yook, J.; Otanez, P.; Tilbury, D. y Moyne, J.: "Design of sampling and transmission rates for achieving control and communication performance in networked agent systems". En: American Control Conference, 2003. Proceedings of the 2003, volumen 4, pp. 3329-3334, 2003.

Lian, F.; Yook, J.; Tilbury, D. y Moyne, J.: "Network architecture and communication modules for guaranteeing acceptable control and communication performance for networked multi-agent systems". Industrial Informatics, IEEE Transactions on, 2006, 2(1), pp. 12-24.

Lincoln, B.: Dynamic Programming and Time-Varying Delay Systems. Tesis doctoral, Automatic Control. Lund Institute of Technoloy, 2003.

Liu, C. y Layland, J.: "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment". Journal of the Association for Computing Machinery, 1973, 20(1), pp. 46-61.

Liu, J.: Real-Time Systems. Prentice Hall, 2000.

Liu, X.; Wu, Y.; J., Shi y Zhang, W.: "Adaptive fault-tolerant ight control system design using neural networks". En: Industrial Technology, 2008. ICIT 2008. IEEE International Conference on, pp. 1-5, 2008.

López-Martínez, M.; Ortega, M.; Vivas, C. y Rubio, F.: "Control No Lineal Robusto de una Maqueta de Helicóptero con Rotores de Velocidad Variable". Revista Iberoamericana de Automática e Informática Industrial, 2007, 4(3), pp. 46-60.

Luck, R y Ray, A.: "An observer-based compensator for distributed delays". Automatica, 1990, 26(5), pp. 903-908.

Lunze, J.; Rowe-Serrano, D. y Steffen, T.: "Control reconfiguration demonstrated at a two-degrees of freedom helicopter model". En: In proceeding of EC'03, 2003.

Lunze, J. y Steffen, T.: "Hybrid Reconfigurable Control". En: Sebastian Engell; Goran Frehse y Eckehard Schnieder (Eds.), Modelling, Analysis, and Design of Hybrid Systems, volumen 279 de Lecture Notes in Control and Information Sciences, pp. 267-284. Springer Berlin / Heidelberg, 2002.

Lynch, N.: Distributed Algorithms. Morgan Kaufmann Publishers, 1996.

Mendes, M.J.; Santos, B. y S_a-Da-Costa, J.: "Multi-agent Platform and Toolbox for Fault Tolerant Networked Control Systems". Journal of computers, 2009, 4(4), pp. 303-310.

Mendes, M.J.; Santos, B. y Sa-Da-Costa, J.: "Multi-agent Architectures for Fault Tolerant Networked Control Systems". En: Intelligent Systems and Agents, 2007 International Conference, pp. 195-200, 2007.

Menéndez, A. y Benítez-Pérez, H.: "Scheduling strategy for Real-Time Distributed Systems ". Journal of Applied Research and Technology, 2010, 8(2), pp. 177-185.

Méndez-Monroy, E. y Benítez-Pérez, H.: "Fuzzy control with estimated variable sampling period for non-linear networked control systems: 2-DOF helicopter as case study". Transactions of the Institute of Measurement and Control , 2011a.

Méndez-Monroy, E.; Benítez-Pérez, H.; Quiñones-Reyes, P.; Cárdenas-Flores, F. y García-Nocetti, F.: "Reconfigurable fuzzy networked control following LMI based on a structural reconfiguration algorithm considering time delays". En: Industrial Technology, 2009. ICIT 2009. IEEE International Conference on, pp. 1-6, 2009.

Méndez-Monroy, P. y Benítez-Pérez, H.: "Neuro-Fuzzy Control with Time Delay Estimation for Linear Networked Control Systems". International Journal of Innovative Computing, Information and Control, IJICIC , 2011b, 7(7), pp. 1-18.

Méndez-Monroy, P. E. y Benítez-Pérez, H.: "Fuzzy control with time delay estimation for networked control systems". En: Electrical Engineering, Computing Science and Automatic Control,CCE,2009 6th International Conference on, pp. 1-6, 2009.

Mok, A. y Dertouzos, M.: "Multiprocessor scheduling in a hard realtimeenvironment". En: In 7th Texas Conference on Computing Systems, , 1978.

Napolitano, M.; Molinaro, G.; Innocenti, M.; Seanor, B. y Martinelli, D.: "A complete hardware package for a fault tolerant ight control system using online learning neural networks". En: American Control Conference, 1999. Proceedings of the 1999, volumen 4, pp. 2615-2619, 1999.

Nilsson, J.: Real-Time Control Systems with Delays. Tesis doctoral, Lund Institute of Technology, 1998.

Ohlin, M; Henriksson, D. y Cervin, A.: TRUETIME 1.5 Reference Manual, 2007. Oosterom, M.: Soft Computing Methods in Flight Control System Design. Tesis doctoral, Delft University of Technology,, 2005.

Oosterom, M. y Babuska, R.: "Virtual sensor for fault detection and isolation in flight control systems - fuzzy modeling approach". En: Decision and Control, 2000. Proceedings of the 39th IEEE Conference on, volumen 3, pp. 2645-2650, 2000.

Oosterom, M.; Babuska, R. y Verbruggen, H.: "Soft computing applications in aircraft sensor management and flight control law reconfiguration". Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2002, 32(2), pp. 125-139.

Peng, C.; Yue, D. y Gu, Z.: "Multiple Sampling Periods Scheduling of Networked Control Systems". En: Control Conference, 2008. CCC 2008. 27th Chinese, pp. 447-451, 2008.

Peng, C.; Yue, D.; Gu, Z. y Xia, F.: "Sampling period scheduling of networked control systems with multiple-control loops". Math. Comput. Simul., 2009, 79(5), pp. 1502-1511.

Quanser: Quanser 2 DOF Helicopter. User and Control Manual, 2006.

Quiñones-Reyes, P.; Benítez-Pérez, H.; Cárdenas-Flores, F. y García-Nocetti, F.: "Control Reconfigurable Difuso Takagi-Sugeno en Red usando Planificación EDF en XPC Target". Revista IEEE Am_erica Latina, 2007, 5(2), pp. 110-115.

Quiñones-Reyes, P.; Méndez-Monroy, E.; Ortega-Arjona, J.; Benítez-Pérez, H. y Durán-Chavesti, A.: "Fuzzy Control Design for a Class of Nonlinear Network Control System, considering a Helicopter Case Study". International Journal Computing, Communication and Control , 2012, 7(2), pp. 367-378.

Ramírez, T.: Algoritmo de planificación en un sistema de control distribuido basado en una arquitectura multi-agente en tiempo real. Tesina o Proyecto, Universidad Autónoma Metropolitana, 2006.

Ramírez-González, T.; Quiñones-Reyes, P.; Benítez-Pérez, H.; Laureano-Cruces, A. y García-Nocetti, F.: "Reconfigurable Fuzzy Takagi Sugeno Networked Control using Cooperative Agents and Local Fault Diagnosis". En: Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on, pp. 1-5, 2007.

Ray, A.: "Introduction to networking for integrated control systems". Control Systems Magazine, IEEE, 1989, 9(1), pp. 76-79.

Romero, H.; Salazar, S.; Escareño, J. y Lozano, R.: "Estabilización de un mini Helicóptero de Cuatro Rotores basada en ujo _optico y sensores inerciales". Revista Iberoamericana de Automática e Informática Industrial , 2010, 7(2), pp. 49-56.

Satnam, A.; Agogino, M. y Morjaria, M.: "A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics". Artificial Intelligence for Ingineering Design, Analysis and Manufacturing, 2001, 15(4), pp. 307-320.

Sename, O.; Simon, D. y Robert, D.: "Feedback scheduling for real-time control of systems with communication delays". En: Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference, volumen 2, pp. 454-461, 2003.

Seto, D.; Lehoczky, J.; Sha, L. y Shin, K.: "On task schedulability in real-time control systems". En: Real-Time Systems Symposium, 1996., 17th IEEE, pp. 13-21, 1996.

Seung, H.: "Scheduling algorithm of data sampling times in the integrated communication and control systems". Control Systems Technology, IEEE Transactions on, 1995, 3(2), pp. 225-230.

Sharma, A.; Golubchik, L. y Govindan, R.: "On the Prevalence of Sensor Faults in Real-World Deployments". En: Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on, pp. 213-222, 2007.

Silvestri, G.; Verona, F.B.; Innocenti, M. y Napolitano, M.: "Fault detection using neural networks". En: Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, volumen 6, pp. 3796-3799, 1994.

Stankovic, J.: "Misconceptions about real-time computing: a serious problem for nextgeneration
systems". Computer, 1988, 21(10), pp. 10-19.

Stankovic, J..: "Distributed real-time computing". En: Parallel and Distributed Real-Time Systems, 1995. Proceedings of the Third Workshop on, pp. 6-7, 1995.

Tanenbaum, A.: Sistemas operativos modernos. Prentice Hall, 1993.

Tanenbaum, A.: Distributed operating systems. Prentice Hall, 1995.

Tanenbaum, A. y Van-Renesse, R.: "Distributed operating systems". ACM Comput. Surv., 1985, 17(4), pp. 419-470.

Tanenbaum, A. y Van-Steen, M.: Distributed Systems. Principles and Paradigms. Prentice Hall, 2007.

Tipsuwan, Y. y Chow, Mo-Yuen: "Control methodologies in networked control systems". Control Engineering Practice, 2003, 11(10), pp. 1099-1111.

Törngren, M.; Henriksson, D.; Redell, O.; Kirsch, C.; El-Khoury, J.; Simon, D.; Sorel, Y.; Zdenek, H. y Arzen, K.: "Co-design of Control Systems and Their Real-TimeImplementation. A Tool Survey". Informe t_ecnico, Department of Machine Design, Royal Institute of Technology, 2006.

Walsh, G.C. y Ye, Hong: "Scheduling of networked control systems". Control Systems, IEEE, 2001, 21(1), pp. 57-65.

Weiss, G.: Multiagent Systems. A Modern Approach to Distributed Arti_cial Intelligence. The MIT Press, 1999.

Wittenmark, B.; Astrom, K. y Arzen, K.: "Computer Control: An Overview". En: IFAC Professional Brief, , 2002.

Wooldridge, M. y Jennings, N.: "Intelligent Agents". Lecture Notes in Arti_cial Intelligence, 1995.

Xia, F. y Sun, Y.: Control and Scheduling Codesign. Flexible Resource Management in Real-Time Control Systems. Springer, 2008a.

Xia, F. y Sun, Y.: "Control-Scheduling Codesign: A Perspective on IntegratingControl and Computing". CoRR, 2008b.

Xia, F.;Wang, Z. y Sun, Y.: "Improving Quality of Control of Networked Control Systems". En: Proccedings of 7th. Int. Conf. on Information Technology, Hyderabad, India, , 2004a.

Xia, F.;Wang, Z. y Sun, Y.: "Simulation Based Performance Analysis of Networked Control Systems with Resource Constraints". En: procedings of 30th. IEEE IECON, Busan, Korea, , 2004b.

Yu, M.; Wang, L.; Chu, T. y Hao, F.: "Stabilization of Networked Control Systems with Data Packet Dropout and Transmission Delays: Continuous-Time Case". European Journal of Control , 2005, (11), pp. 40-49.

Zarafshan, F.; Latif-Shabgahi, G. y Karimi, A.: "A novel weighted voting algorithm based on neural networks for fault-tolerant systems". En: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, volumen 9, pp. 135-139, 2010.

Zhang, W y Yu, L.: "Technical communique: Modelling and control of networked control systems with both network-induced delay and packet-dropout". Automatica, 2008, 44(12), pp. 3206-3210. ISSN 0005-1098.

Zhou, P. y Xie, J.: "Feedback scheduling for resource-constrained real-time control systems". En: Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on, pp. 800-804, 2005.

Índice alfabético