Evaluación del protocolo de comunicaciones AR-TP

Vanesa Sánchez Moya
Daniel Berjón Díez
Departamento de Ingeniería de Sistemas Telemáticos (DIT)
ETSI Telecomunicación
Univ. Politécnica de Madrid (UPM)
E-28040 Madrid
{vanesa,berjon}@dit.upm.es

Juan Zamorano Flores
Santiago Urueña Pascual
Dept. de Arquitectura y Tecnología de Sistemas Informáticos (DATSI)
Facultad de Informática
Univ. Politécnica de Madrid (UPM)
E-28660 Boadilla del Monte
{jzamora,suruena}@datsi.fi.upm.es

15 de junio de 2007

Resumen

AR-TP es un protocolo de comunicación para sistemas de tiempo real distribuidos que evita el no determinismo de algunas tecnologías de red mediante un mecanismo de paso de testigo. Este protocolo está basado en RT-EP, un protocolo de investigación desarrollado originalmente por la Universidad de Cantabria. AR-TP permite realizar análisis de planificación de la red, haciendo posible usar Ethernet en el ámbito de los sistemas de tiempo real estrictos, y ha sido escrito en su totalidad en Ada y conforme al perfil de Ravenscar.

1. Introducción

Este artículo resume el funcionamiento, implementación y evaluación del protocolo de comunicaciones AR-TP (*Arbitrated Real-Time Ethernet*) para sistemas de tiempo real con requisitos de tiempo estrictos. AR-TP está basado en RT-EP (*Real Time-Ethernet Protocol*), protocolo desarrollado por la Universidad de Cantabria [1]. Fue diseñado como un mecanismo de comunicaciones de tiempo real de investigación de bajo coste y relativamente rápido, y al mismo tiempo que fuera analizable (tiempo real estricto o *hard real-time*).

La tecnología de red Ethernet está ampliamente extendida y proporciona velocidades altas a bajo coste, pero su comportamiento no es predecible ya que puede haber colisiones. RT-EP proporciona una capa de control de transmisión mediante un mecanismo de paso de testigo para evitar dicho indeterminismo temporal. AR-TP mejora la eficiencia y el ancho de banda ofrecido por RT-EP, y también hace más robusta la tolerancia a fallos.

Tanto el análisis temporal de AR-TP como un prototipo del mismo ya ha sido realizado en el pasado [2][3]. Pero para el segundo prototipo recogido en este artículo se ha efectuado ciertas modificaciones en la especificación de AR-TP, y por tanto es necesario volver a evaluar el protocolo y realizar pequeños cambios a las ecuaciones temporales.

Este artículo está organizado de la siguiente forma. La sección 2 describe el funcionamiento del protocolo. La sección 3 analiza su comportamiento temporal, mientras que la sección 4 evalúa la implementación realizada del mismo. La sección 5 propone una posible estrategia para configurar el protocolo para adaptarlo a cada sistema distribuido. Finalmente, la sección 6 enumera las líneas futuras y la sección 7 resume las conclusiones de este artículo.

2. Descripción del protocolo

AR-TP (al igual que RT-EP) utiliza un mecanismo de control de transmisión basado en

paso de testigo. El testigo (*token* en su designación inglesa) es un paquete especial cuya función es la de arbitrar el acceso al medio. Ninguna estación puede enviar hasta que no reciba un testigo que le otorgue ese derecho. La red es un recurso compartido y no desalojable: una vez que se empieza a transmitir un mensaje no se puede mandar otro hasta que se termine la transmisión (aunque sea más urgente). La planificación de los mensajes se realiza mediante prioridades fijas, y para evitar la inversión de prioridad el testigo contiene en cada momento la prioridad más alta de los mensajes listos para ser enviados (y el identificador de la máquina correspondiente).

AR-TP permite el envío de hasta n mensajes en cada fase de transmisión, frente a RT-EP en el que sólo se envía un mensaje como máximo. Consta de tres fases: la *fase de inicialización*, la *fase de arbitraje* y la *fase de transmisión*. La fase de inicialización sólo se realiza al arrancar el sistema, y a diferencia de las otras dos no requiere un comportamiento de tiempo real. El objetivo de esta fase es mandar paquetes de inicialización para descubrir las estaciones de la red, y asignar a cada una un identificador. De esta forma, todas las estaciones se organizan en un anillo lógico (independiente de la topología) que establece en qué orden se transmitirá el testigo durante la fase de arbitraje. Además, como se usa el modo promiscuo, todas las estaciones reciben todos los paquetes.

2.1. Fase de arbitraje

Tras la fase de inicialización, la primera estación en el anillo lógico adopta el papel de coordinadora de la red, poniendo en circulación el primer testigo. El testigo incluye un campo con n puestos (*slots*) para la gestión de los n mensajes más prioritarios en cada ciclo del protocolo. La estación coordinadora en primer lugar escribirá la prioridad de cada uno sus mensajes pendientes en el testigo (junto a su ID de estación). Si no tiene ninguno, las prioridades almacenadas se mantienen con valor nulo, pero si tiene más de n mensajes pendientes elegirá los más prioritarios. A continuación, pasa el testigo a su sucesora en el anillo lógico.

Cuando la estación siguiente recibe el testigo, primero incrementa el número de secuencia. Si la estación no tiene mensajes pendientes, pasa el testigo a la sucesora en el anillo sin modificar los *slots* del testigo. En caso de tener mensajes esperando a ser enviados, se comprueba en primer lugar si aún quedan puestos libres. Si es así, se ocupan de inmediato y se envía el testigo actualizado. Si no quedan puestos libres, o los hay pero no tantos como mensajes pendientes, habrá que realizar un reconocimiento de las prioridades registradas en el testigo. Si resulta que alguna prioridad de las almacenadas es menor que la del mensaje más prioritario en la estación local, esta última se adueña del puesto correspondiente.

Una vez que el testigo ha sido procesado por todas las estaciones del anillo, termina la fase de arbitraje y el testigo contiene las prioridades (y remitentes) de los mensajes más prioritarios. Si el testigo está vacío, no hay ningún mensaje pendiente y la coordinadora comienza otra vuelta tras un retardo configurable W. Esta es una mejora con respecto a *RT-EP* ya que no se introducía ninguna espera aunque no hubiese mensajes pendientes, con el consiguiente gasto de recursos del sistema.

2.2. Fase de transmisión

Después de la fase de arbitraje, las estaciones con los mensajes más prioritarios tienen permiso para transmitir. La coordinadora enviará un paquete de permiso avisando así del comienzo de la fase de transmisión (incluso si se trata de ella misma), a la primera estación que aparezca registrada como ganadora (aunque recordamos que todas las estaciones reciben el paquete ya que están en modo promiscuo). Este paquete especifica en qué orden se transmitirán los mensajes de información. El nodo coordinador establece el orden, pero el protocolo no especifica ningún algoritmo así que cada implementación es libre de elegir uno. En la implementación realizada no se hace ningún tipo de reordenación y, por tanto, los turnos se asignan según vayan acaparando las estaciones cada *slot*.

Todas las estaciones, incluida la coordinadora, analizarán el paquete de permiso para saber si son ganadoras. En el caso de que así fuera, miran el puesto que les ha tocado (pueden ser varios pero en cada momento se considera el más cercano). La estación dueña del primer puesto es la que comienza a enviar información. El paquete de información contiene un campo denominado *Current Slot* ("puesto actual") que indica el número de *slot* utilizado en el envío correspondiente. Cualquier estación con derecho a enviar en un momento dado tendrá que actualizar este campo antes de mandar la información.

La fase de transmisión continúa hasta que todas las estaciones ganadoras envían sus mensajes correspondientes. Las estaciones ganadoras tienen libertad para enviar mensajes con más prioridad que la que especificada en la fase de arbitraje (e.g. se ha generado un nuevo mensaje muy prioritario justo después de retransmitir el testigo). La estación que posea el último turno, una vez que transmita su mensaje o mensajes pasará a ser la nueva coordinadora de la siguiente fase de arbitraje. Nótese que el retardo W sólo se introduce cuando concluye una vuelta de arbitraje sin que haya nuevos mensajes pendientes.

2.3. Paquetes del protocolo

Esta implementación de AR-TP utiliza Ethernet como medio de transmisión, pero puede ser sustituida por otras redes de medio compartido. Se define un parámetro ($ARTP_{MTU}$) para indicar el tamaño máximo del mensaje de información que puede transmitir AR-TP. Este tamaño depende de la capa inferior utilizada, y por lo tanto se ajusta a la estructura de la trama de Ethernet.

Los paquetes de AR-TP no conservan la estructura que tenían en RT-EP (se ha introducido nuevos campos). A parte de los paquetes de inicialización, hay de dos tipos:

- **paquete de testigo**: utilizado para la transmisión de los testigos así como para el paquete de permiso (figura 1).

  El campo *Type* está presente en todos los paquetes y tiene dos subcampos: la clase del paquete (e.g. testigo, info) y la versión del protocolo (si RT-EP o AR-TP). La combinación de ambos subcampos forman las iniciales de los distintos paquetes utilizados por los protocolos de acorde con el código ASCII.

  En el campo *slots* quedan registrados los n mensajes más prioritarios junto a los ID de las estaciones remitentes de dichos mensajes. El campo *Slot Index* apunta al primer *slot* del campo contiguo que se encuentre libre. De esta forma las estaciones no tienen que recorrer todo el campo *Slots* cuando vayan a registrar algún mensaje pendiente.

- **paquete de información**: se utiliza en la transmisión de datos (figura 2).

  Se han añadido dos nuevos campos con respecto a RT-EP: *Current Slot* y *Congestion Priority*. El primero de ellos señala el *slot* utilizado por la estación que está enviando el mensaje de información. El resto de estaciones ganadoras procederán a enviar sus mensajes pendientes cuando comprueben que les pertenece el turno inmediatamente posterior al que está en curso.

  El campo *Congestion Priority* no se utiliza en la implementación actual pero se tuvo en cuenta para las futuras mejoras (sección 6).

2.4. Tolerancia a fallos

Al igual que RT-EP, AR-TP tolera las pérdidas de paquetes y fallos de las estaciones. Cada máquina que envía el testigo vigila las comunicaciones siguientes en la red. Si la máquina de la cual se espera respuesta contesta, es decir, si envía a su vez un paquete cualquiera, su estado es considerado como correcto. En caso contrario, la máquina originaria reenvía el testigo. Si se llega a un número máximo de retransmisiones sin que el siguiente nodo reenvíe el testigo, se considera que la estación ha caído y el próximo paquete informará sobre ello para que las demás máquinas la supriman del anillo.

Type		Flags		Packet	Token	Failed	Slot	Enqueued	Slots		
PT	V	*Reserved*	FS	number	master	station	index	messages	$(P, ID)_1$	...	$(P, ID)_n$
5 bits	3 bits	*7 bits*	1 bit	2	2	2	2	2	4	...	4

Figura 1: Paquete de testigo (tamaños en octetos)

Type		Current	Packet	Priority		Channel	Length	Info
PT	V	slot	number	Normal	Congestion			
5 bits	3 bits	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	0–1488 bytes

Figura 2: Paquete de información

Durante la fase de transmisión la tolerancia a fallos se hace más compleja y distribuida aunque —para no introducir demoras en el envío de información— en esta fase no se eliminan estaciones del anillo en ningún caso. No obstante, para que la actividad continúe incluso en caso de fallo, las estaciones controlan el buen funcionamiento del protocolo. Si no se produce ningún incidente, cada estación envía sus mensajes pendientes cuando el turno contenido en el campo *Current Slot* es justo el anterior al esperado para enviar (confirmación implícita). Pero si dicho mensaje no llega, los nodos con turnos pendientes terminan enviando igualmente los mensajes correspondientes al expirar un tiempo límite establecido para este fin.

Otro contratiempo que podría tener lugar es que no funcionara adecuadamente la estación coordinadora. En ese caso la próxima vuelta de arbitraje no comenzaría nunca, y para evitar esta situación las estaciones que no tienen turnos pendientes (porque ya los han utilizado o porque no los tuvieron en ningún momento), quedan a su vez controlando que comience una nueva fase de arbitraje. Si esta fase no empieza tras concluir el tiempo límite de espera, alguna de dichas estaciones procederá a comenzarla por sí misma.

3. Comportamiento temporal

Suponiendo un número de nodos igual a M, la duración del ciclo k-ésimo en AR-TP viene descrito por la ecuación 1 —no se han tenido en cuenta posibles caídas de nodos ni retransmisiones— donde t_{token} es el tiempo de transmisión del testigo, t_{delay} es el retardo entre paquetes de control, t'_{delay} es el tiempo transcurrido desde que la coordinadora recibe el testigo hasta que envía el permiso, $t_{permission}$ es el tiempo de transmisión del paquete de permiso, t''_{delay} es el retardo entre mensajes de información, n_k es el número de mensajes enviados en la fase de transmisión k-ésima, $Msg_{i,k}$ es el tiempo de transmisión del mensaje i-ésimo del ciclo k, y W el retardo que se realiza cuando termina una vuelta de arbitraje sin ganadora alguna registrada.

El retardo entre paquetes de control no consta únicamente del tiempo de cómputo, sino que el envío del testigo se retrasa un determinado periodo configurable para reducir la carga de CPU del sistema, y para que todas las estaciones puedan procesar el testigo aunque tengan velocidades distintas.

Una de las principales ventajas de AR-TP respecto a su antecesor RT-EP es un mayor aprovechamiento del ancho de banda, al ser posible el envío de hasta n mensajes por fase de transmisión. La eficiencia aumenta al ser mayor la relación entre mensajes de información y paquetes de control. Pero este aumento en el ancho de banda tiene un precio puesto que cuanto mayor sea el tiempo de la fase de transmisión, peor será el máximo tiempo de bloqueo.

El *tiempo de bloqueo* en AR-TP será el intervalo de tiempo en el peor caso que un nodo tiene que esperar desde que genera un mensaje hasta que puede transmitirlo, sin contar la posible interferencia del mensaje (definida posteriormente). Por lo tanto, el valor de n no será arbitrario sino que existe un compromiso entre este número y el tiempo máximo de bloqueo.

$$Cycle_k = \overbrace{(t_{delay} + t_{token}) \cdot M}^{fase\ de\ arbitraje} + \begin{cases} W & n_k = 0 \\ \overbrace{t'_{delay} + t_{permission} + \sum_{i=1}^{n_k} \left(t''_{delay} + Msg_{i,k}\right)}^{fase\ de\ transmision} & n_k > 0 \end{cases} \quad (1)$$

El *mejor tiempo de respuesta* del protocolo se da cuando el penúltimo nodo de la fase de arbitraje quiere enviar un mensaje de tamaño mínimo y de prioridad mayor a las registradas, justo antes de pasar el testigo a la coordinadora.

$$R_{best} = t_{token} + t_{delay} + Msg_{min}$$

donde Msg_{min} es el tiempo de transmisión de un mensaje de información de tamaño mínimo. Pero para el análisis de la planificación de un sistema de tiempo real lo que interesa es acotar el tiempo en el peor caso. Para un mensaje genérico μ_i, éste será:

$$R_i = Q_i + Tr$$

donde Q_i es el *tiempo de encolamiento* —máximo tiempo que un mensaje puede estar encolado antes de ser enviado— y Tr es la duración máxima de la fase de transmisión:

$$Tr = \left(t''_{delay} + Msg_{max}\right) \cdot n$$

Despreciando la influencia del valor n en la duración de la fase de arbitraje, que es mínima como se verá más adelante en la sección 4, se puede modelar el tiempo de dicha fase como

$$Ar = (t_{delay} + t_{token}) \cdot M + t'_{delay} + t_{permission}$$

El valor del tiempo de encolamiento Q_i consta de tres componentes: el tiempo de bloqueo, la duración de la fase de arbitraje y la interferencia del mensaje. El tiempo máximo de bloqueo viene dado por:

$$B = Ar + \text{máx}(Tr, W)$$

Ocurre cuando el primer nodo produce un mensaje a la vez que está transmitiendo el testigo. Dada esta situación, la estación debe esperar la vuelta de arbitraje entera y la duración máxima de la fase de transmisión (o el retardo W, si éste fuera más largo que dicha fase de envío).

La interferencia de un mensaje se define como el número de ciclos que debe esperar la estación antes de enviarlo debido a que hay mensajes más prioritarios. Como ya se demostró en el pasado [2], es posible calcular dicha interferencia en AR-TP:

$$Q_i = B + \left\lfloor \frac{\sum_{j \in hep(i)} \left\lceil \frac{Q_i}{T_j} \right\rceil}{n} \right\rfloor (Ar + Tr) + Ar$$

donde $hep(i)$ es el conjunto de mensajes con mayor o igual prioridad que μ_i, y T_j es el tiempo mínimo entre llegadas del mensaje j (μ_j). La interferencia depende a su vez de ella misma, con lo que se tendría que resolver la relación recurrente para obtener el valor final. Se aprecia fácilmente que este tiempo de bloqueo depende en gran medida del número de mensajes por fase de transmisión (n), del tiempo de retardo (t_{delay}) y del tamaño máximo del mensaje de información (Msg_{max}). El usuario del protocolo puede ajustar según sus necesidades los dos primeros valores. Nótese que aumentando n se incrementará el tiempo de respuesta del mensaje más prioritario pero tenderá a reducirse (de forma discontinua y no monótona) el de los mensajes menos urgentes.

4. Evaluación

Este segundo prototipo también se ha implementado en el lenguaje de programación Ada 2005, adhiriéndose en concreto al perfil de Ravenscar [4]. Ada dispone de mecanismos específicos para la programación de sistemas de tiempo real, incluyendo cláusulas de represen-

Cuadro 1: Métricas de AR-TP (μs)

Tamaño mensaje (octetos)	n	Procesamiento testigo (empírico) Nodo 1	Nodo 2	Nodo 3	Duración fase arbitraje Empírico	Teórico
75	1	5.40	3.39	1.45	670	420.48
	3	5.77	3.16	1.90	676	420.48
	5	6.34	3.31	2.20	795	420.48
500	1	5.40	3.13	1.45	663	420.48
	3	5.76	3.10	2.87	684	420.48
	5	6.12	3.26	1.60	736	420.48
1500	1	5.40	3.13	2.13	677	420.48
	3	4.33	2.39	1.13	673	420.48
	5	9.57	9.86	10.95	750	420.48

tación, manejo de interrupciones, gestión del tiempo, diversas políticas de planificación y sincronización libre de inversión de prioridad.

El perfil de Ravenscar es un subconjunto estándar del lenguaje diseñado para aplicaciones de alta integridad. Anteriormente los perfiles para sistemas críticos (*safety critical*) eran puramente secuenciales, mientras que Ravenscar al ser concurrente facilita enormemente el diseño y mantenimiento. El perfil permite realizar un análisis temporal de la aplicación a la vez que la implementación del núcleo de ejecución es certificable y muy eficiente. Gracias al determinismo conseguido, los sistemas desarrollados conforme al perfil pueden someterse a los estrictos procesos de certificación requeridos por los sistemas críticos de tiempo real.

Para que la evaluación fuera lo más precisa posible, las pruebas se hicieron en un entorno muy controlado: el *run-time* de Ada no ejecuta sobre un sistema operativo sino directamente sobre máquina desnuda, concretamente sobre el núcleo de ejecución ORK [5]. Además, las servicios de red eran llamadas directas al controlador de la tarjeta Ethernet.

El sistema estaba formado por un Pentium a 133 MHz (primer nodo del anillo), un Pentium II a 233 MHz (segundo nodo), y un Pentium III a 500 MHz (tercer nodo). A su vez se disponía de una máquina adicional —otro Pentium III— en la red para observar la comunicación mediante un analizador de protocolos. Dichos computadores estaban conectados en estrella mediante un concentrador (*hub*) Ethernet a 100 Mbps (red completamente independiente).

Cuando el prototipo está en modo de depuración es posible extraer diversos tiempos tales como la duración de una llamada al controlador de red, el tiempo de cómputo de diferentes trozos de código, o la duración de la diferentes fases. El cuadro 1 muestra algunas métricas para distintas configuraciones de AR-TP. En ellas se varía el tamaño máximo de los mensajes de información entre 75 y 1500 bytes y el número de mensajes por ciclo toma los valores 1, 3 y 5. El retardo utilizado entre testigos (para coordinar las diferentes estaciones) fue de 100 μs y se tomaron los resultados obtenidos por el primer nodo en el anillo.

Nótese que cuando el número mensajes por ciclo es igual a uno el protocolo se comporta como RT-EP y como se aprecia en el cuadro existe un pequeño incremento en el tiempo de cómputo de procesamiento del testigo para valores mayores. Así que se puede inferir que AR-TP consigue una mayor capacidad de transmisión que RT-EP a costa de un leve incremento en la sobrecarga de implementación del protocolo.

Otra métrica interesantes del protocolo es el tiempo de la fase de arbitraje, que a su vez depende del tiempo de procesamiento del testigo. En el cuadro se muestra el teórico, que es el resultado de enviar 4 mensajes de 64 bytes a 100 Mbps y realizar 4 esperar de 100 μs, y el medido empíricamente. La diferencia entre ambos es de aproximadamente 250 μs que

es atribuible a las 4 operaciones de recepción, procesado y transmisión de los paquetes. Nótese que son aproximadamente 65 μs por nodo y que aunque pueda parecer elevado incluye operaciones como reconocimiento y procesado de la interrupción y accesos al dispositivo físico que son lentas bien porque provocan muchos fallos de cache al cambiar radicalmente el contexto, bien por el acceso a registros de dispositivos.

Nótese que a mayor número de mensajes por ciclo mayor tamaño del testigo, sin embargo no influye en el tiempo de transmisión ya que en todos los casos estudiados no supera el tamaño de una trama de Ethernet mínima. Por lo tanto el incremento en la fase de arbitraje es pequeño ya que es debido al tiempo de comprobación del testigo. Es decir, con un bajo incremento de la fase de arbitraje se consigue multiplicar el ancho de banda respecto a RT-EP. Igualmente, se aumenta la escalabilidad ya que para mandar el mismo número de mensajes de información hay que transmitir muchos menos paquetes de arbitraje. Sin embargo hay que hacer notar que este incremento en el número de mensajes de información transmitido en cada ciclo aumenta el tiempo de bloqueo. Por ello el número de mensajes por fase de transmisión debe ser elegido considerando las restricciones temporales.

5. Cómo ajustar los parámetros

Gracias al múltiple envío de mensajes por ciclo de transmisión, se consigue un mayor aprovechamiento del ancho de banda. No obstante, se genera una relación directa entre dicho número de mensajes por ciclo y el tiempo máximo de bloqueo. Por lo tanto, aunque el número de mensajes por ciclo n y el tamaño máximo de los mensajes de información no son parámetros temporales, influyen directamente en las métricas del protocolo. Los retardos t_{delay} entre paquetes también afectan el peor tiempo de respuesta, pero en menor medida.

Una posible estrategia para ajustar los parámetros del protocolo comenzaría comprobando cuál es el tamaño máximo de los mensajes de información de la aplicación concreta. Si hay mensajes demasiado grandes convendría segmentarlos para no incrementar mucho el tiempo de bloqueo. Después, hay que ajustar el retardo entre mensajes según las velocidades de los nodos del sistema (la CPU más lenta) y los recursos computacionales que puedan dedicarse al protocolo (regular el flujo de paquetes para reducir carga de la CPU) frente al resto del código de tiempo real.

Una vez obtenidos los tiempos de transmisión (Msg_{max}, t_{token} y $t_{permission}$) y los retardos entre paquetes (t_{delay}, t'_{delay} y t''_{delay}) se irá calculando el peor tiempo de respuesta de *sólo* el mensaje más prioritario para $n = 1, 2, 3 \ldots$ Cuando el bloqueo sea demasiado grande y se pase de su plazo temporal, se irá reduciendo n hasta que se cumplan los plazos de *todos* los mensajes de prioridad inferior.

Por último, habría que escoger el retardo W. Un valor alto puede reducir considerablemente la sobrecarga, y mientras sea menor o igual que Tr no se incrementará el tiempo de bloqueo ni por tanto el tiempo de respuesta. De todas formas, un valor de W bajo, aunque no influya en el cálculo del peor tiempo de respuesta, sí que mejorará el tiempo de respuesta medio. En cualquier caso, el valor de W debería ser igual o superior al valor de t_{delay}.

En resumen, tanto el número de paquetes como los retardos descritos deben ser establecidos tomando como punto de partida las restricciones temporales específicas de cada aplicación. Existen otros parámetros de tiempo configurables que van a determinar en qué momento se considera que la no ocurrencia de un suceso supone un mal funcionamiento del protocolo, así como el número de reenvíos máximos antes de proceder a la eliminación de alguna estación. Dichos parámetros también deberán ser ajustados por el usuario acorde al entorno y los requisitos de cada aplicación.

6. Trabajo futuro

El protocolo puede seguir evolucionando ya que el testigo puede aprovecharse para ofrecer nuevos servicios. En concreto, existen dos nuevas funcionalidades incluidas en la especificación de AR-TP que aún no han sido im-

plementadas: un mecanismo para el control de congestión, y el asentimiento de mensajes.

Para el mecanismo de *control de congestión*, cada mensaje de información debe tener asociadas dos prioridades. Una de ellas para ser utilizada por defecto y otra que sería usada sólo en caso de congestión (*Normal Priority* y *Congestion Priority* en la figura1). Se considera que la red está congestionada desde el momento en el que el número de mensajes pendientes de envío supera un determinado umbral O_1), hasta que dicho número se reduce respecto a otro umbral O_2. Cada mensaje ha de tener asociado un plazo temporal, así que aquellos mensajes menos importantes irán caducando en la cola mientras se envían los más críticos.

El *asentimiento de mensajes* permitiría saber si los mensajes de información han sido recibidos correctamente. Esto podría conseguirse añadiendo un nuevo campo en el testigo de forma que las estaciones que recibieron algún mensaje durante una fase de transmisión puedan señalar en la siguiente fase de arbitraje que el mensaje fue recibido correctamente. Al aprovechar el testigo para este fin apenas se incrementa la sobrecarga introducida por el protocolo.

7. Conclusiones

AR-TP es un protocolo de comunicaciones de investigación para sistemas distribuidos de tiempo real estricto para Ethernet basado en paso de testigo y prioridades fijas. Es posible realizar un análisis temporal de la red, y gracias al múltiple envío de mensajes por ciclo de transmisión se consigue una mayor planificabilidad y ancho de banda que RT-EP. Pero se genera una relación directa entre dicho número de mensajes por ciclo y el tiempo máximo de bloqueo. Este compromiso tendrá que ser evaluado por los usuarios del protocolo, siguiendo por ejemplo la estrategia planteada sobre el ajuste de los parámetros configurables.

Un prototipo de AR-TP ha sido implementado aprovechando las ventajas de Ada 2005 y respetando el perfil de Ravenscar para conseguir un mayor determinismo temporal. Gracias a este prototipo se ha realizado ciertas modificaciones a la especificación original de AR-TP para poder mejorar la implementación del mismo. Asimismo, se han propuesto ciertas líneas futuras de investigación para mejorar y añadir nueva funcionalidad al protocolo.

Referencias

[1] Martínez, J.M., González Harbour, M.: RT-EP: A fixed-priority real time communication protocol over standard ethernet. In Vardanega, T., Wellings, A., eds.: Reliable Software Technologies — Ada-Europe 2005. Volume 3555 of LNCS., Springer-Verlag (2005)

[2] Urueña, S., Zamorano, J., Berjón, D., Pulido, J.A., de la Puente, J.A.: Schedulability analysis of AR-TP, a Ravenscar compliant communication protocol for high-integrity distributed systems. In: 14th International Workshop on Parallel and Distributed Real-Time Systems, Island of Rhodes, Greece (April 2006)

[3] Urueña, S., Zamorano, J., Berjón, D., Pulido, J.A., de la Puente, J.A.: The arbitrated real-time protocol (AR-TP): A Ravenscar compliant communication protocol for high-integrity distributed systems. In Pinho, L.M., González Harbour, M., eds.: Reliable Software Technologies — Ada-Europe 2006. Volume 4006 of LNCS., Springer Berlin / Heidelberg (2006) 215–226

[4] Burns, A., Dobbing, B., Vardanega, T.: Guide for the use of the Ada Ravenscar profile in high integrity systems. Technical Report YCS-2003-348, University of York (2003)

[5] de la Puente, J.A., Ruiz, J.F., Zamorano, J., García, R., Fernández-Marina, R.: ORK: An open source real-time kernel for on-board software systems. In: DASIA 2000 — Data Systems in Aerospace, Montreal, Canada (May 2000)