3 Sistemas en tiempo real

Cada vez es más común la existencia de sistemas en tiempo real en el día a día. Son sistemas que requieren un tiempo de respuesta inmediato lo que obliga a diseñar sistemas con un tiempo de computación mínimo. En este proyecto se diseña una herramienta que se soporta sobre una arquitectura orientada a datos haciendo uso de uno de los protocolos estándar más conocido para este tipo de arquitecturas, el protocolo DDS.

3.1 Introducción

Un sistema en tiempo real (RTS) es aquel cuyo correcto funcionamiento depende de que las salidas se produzcan en un tiempo acotado despues de un determinado evento. En este aspecto diremos que los sistemas en tiempo real son sistemas reactivos con requisitos, es decir, sistemas que reaccionan ante determinados eventos en un tiempo acotado sin establecerse a priori ningún tipo de orden o periodicidad. Este tiempo no tiene porque ser necesariamente muy limitado, simplemente debe ser acotado el tiempo desde que se produce un evento y el tiempo de la respuesta provocada. De modo que uno de los aspectos que ha de ser muy cuidado en un sistema en tiempo real es el tiempo de reacción ante cada evento. En el momento en el que una tarea no se realiza en el tiempo acotado que se le ha asignado se considera que el sistema no responde correctamente. En estos sistemas se observa tanto la correcta funcionalidad de las aplicaciones como el tiempo de ejecución dedicado. Si una aplicación es funcionalmente correcta pero tarda

demasiado tiempo en realizar cierta respuesta se dirá que el sistema en tiempo real no funciona correctamente. En este sentido se pueden clasicar los sistemas en tiempo real en dos grupos:

- Hard RTS: Cuando el incumplimiento de un *deadline* implica un funcionamiento incorrecto. Un ejemplo de estos sistemas podra ser el sistema de frenado ABS o un marcapasos.
- Soft RTS: Cuando el incumplimiento de un *deadline* no implica funcionamiento incorrecto pero sí una degradación en la calidad de servicio. Una ejemplo de este tipo de sistemas sera una interfaz de usuario o un reproductor de DVD.

A pesar de esta clasificación, cuando se diseña un soft RTS se hace como si se tratase de un sistema hard RTS. La diferencia es que al realizar un hard RTS estos son sometidos a rigurosos test de carga, rendimiento, etc, que aseguren el correcto funcionamiento del hard RTS. En general, cuando se diseña un sistema de vuelo, estos sistemas son diseñados como hard RTS dado que un fallo temporal en la respuesta de un evento pude llevar a un desastre mortal al igual que ocurre con otros de sistemas críticos. No obstante, en el caso del diseño de sistemas de simulación aeroespacial, a pesar de ser sistemas en tiempo real, estos son diseñados como soft RTS, dado que un retraso ante un evento implicaría que el usuario no se sienta completamente inmerso pero no tendrá consecuencias mortales (no se trata de un sistema crítico). Los sistemas en tiempo real estan definidos por:

- Los **eventos externos** que puede atender.
- La **respuesta lógica** (o salida) que debe producir cada evento.
- Los **requerimientos temporales** de esas respuestas (*deadlines* relativos).

Para la especificación de los requerimientos temporales se suele hacer uso de diagramas de secuencia u otros modelos de computación y lenguajes de modelado. Para poder asegurar el cumplimiento de todos los requisitos de un sistema en tiempo real se suele hacer uso de sistemas operativos en tiempo real (RTOS) que se encarguen del *scheduling*, *dispatching* y gestion de memoria, propios de un sistema en tiempo real.

Los sistemas en tiempo real suelen hacer uso de ficheros (o configuraciones) que definan las calidades de los servicios del sistema (QoS). Es decir, configuraciones que

especifiquen cual ha de ser el comportamiento del sistema. Un incumplimiento de estos QoS deberá provocar un fallo en el sistema que indique cuál de las especificaciones indicadas en el QoS no se ha cumplido correctamente (y si es posible también se indica el motivo).

3.2 Sistemas distribuidos

Un sistema distribuido es todo sistema compuesto por varios nodos con capacidad de procesamiento que se presentan al usuario como un único sistema capaz de procesar grandes volúmenes de datos. Las características básicas de este tipo de sistemas son la transparencia para:

- **Acceso**: El objetivo es esconder las diferencias en la representación de los datos y en el acceso a los recursos.
- **Localización**: El objetivo es independizar a la aplicación cliente del nodo (o nodos) en el que se encuentran los datos solicitados.
- **Migración**: El objetivo es tener capacidad de mover los datos de un nodo a otro sin afectar a la aplicación cliente.
- **Relocalización**: El objetivo es esconder el hecho de que un recurso pueda cambiar de nodo en tiempo de ejecución.
- **Sincronización y concurrencia**: El objetivo es poder compartir un recurso entre varias aplicaciones cliente (consistencia de datos).
- **Replicación de recursos**: El objetivo es ofrecer la capacidad de copiar los recursos entre los diferentes nodos que forman el sistema distribuido sin afectar las aplicaciones cliente.
- **Fallo**: El objetivo es ocultar las aplicaciones cliente de cualquier fallo ocurrido en el sistema y tener la capacidad de recuperar el sistema a un estado consistente.
- **Persistencia**: El objetivo es ocultar al usuario de la forma en la que los datos son accedidos y almacenados (base de datos, disco duro, memoria, etc).

Estos sistemas por lo general presentan una gran capacidad de abstracción no obstante esta capacidad introduce una cierta latencia que puede convertirse en un problema en combinación con otros sistemas.

3.2.1 Sistemas distribuidos en tiempo real

Existen aplicaciones en tiempo real que acceden a recursos de un sistema distribuido que no requiere tiempo real, estos sistemas no son considerados como sistemas distribuidos en tiempo real, puesto que el tiempo real solo está en las aplicaciones cliente. De modo que, para considerar un sistema distribuido en tiempo real es necesario analizar ciertos aspectos de las características de estos sistemas para adaptarlos a un sistema en tiempo real.

Una aplicación en tiempo real debe ser predecible (o determinista), de modo que no exista ningún tipo de indeterminismo que deba ser resuelto en tiempo de ejecución. En general un sistema no determinista introduce ciertas latencias para resolver indeterminismos en ciertos puntos que pueden provocar un mal funcionamiento del sistema a causa del retraso de los *deadlines* de cada evento.

A diferencia de un sistema distribuido cualquiera, los sistemas distribuidos en tiempo real requieren una serie de limitaciones sobre las características que suelen tener estos sistemas en general. Un sistema distribuido es un sistema no determinista en el sentido de que abstrae a las aplicaciones cliente del funcionamiento de la red de nodos. Estos sistema suelen ofrecer un *middleware*, estandarizado por la OMG, para la abstraccion del sistema. Este *middleware* es el encargado de resolver los no determinismos del sistema como pueden ser la localizacion de recursos, la migracion, etc.

Como se indicó anteriormente, los sistemas distribuidos tienen una serie de características que dan cierto nivel de abstracciñon a costa de introducir una cierta latencia en el acceso a los datos. Por ese motivo, en los sistemas distribuidos en tiempo real se debe prescindir de caractersticas como:

- **Localización**: Los datos deben estar localizados en todo momento dentro de la red.
- **Migración**: Un sistema en tiempo real no permite el movimiento de recursos de forma abstracta (consecuencia directa de la falta de localización).

- **Relocalización**: Al igual que ocurre con la migración, los sistemas distribuidos en tiempo real no pueden tener la capacidad de mover los datos en tiempo de ejecución.

El *middleware* utilizado debe utilizar unas pautas mas estrictas que elimina los indeterminismos de estos sistemas. Ademas, otros aspectos de la distribución que son menos relevantes o no resueltos en tiempo real son:

- El servicio de nombres se puede sustituir por un sistema mas sencillo y normalmente cerrado.
- Las réplicas van mas dirigidas a la tolerancia a fallos. En tiempo real se utilizan otros mecanismos mas sencillos y predecibles.
  - Control de sobrepaso del tiempo de ejecución.
  - O simplicación de elementos en sistemas de seguridad crítica.
- Aspectos como la seguridad no son importantes al tratarse de sistemas normalmente cerrados. Sistemas de ficheros distribuidos.

Por otro lado, los aspectos importantes para tiempo real siguen siendo el modelo de concurrencia, las políticas de planificación y la sincronización que ahora se hace mas compleja si se accede a recursos compartidos remotos. Además aparecen nuevos aspectos a tener en cuenta como son la sincronización de relojes, el uso de redes de comunicaciones predecibles y los conceptos relativos a la calidad de servicio que cada vez se mezclan más con los de tiempo real.

3.3 Distribución de datos por DDS

DDS es un protocolo estándar definido por la OMG (Object Management Group), llamado *Data Distribution Service for real-time systems* (DDS), que proporciona una solución a la necesidad de tener un paradigma centrado en datos para sistemas distribuidos [24]. Introduce un Espacio Global de Datos virtual (*Global Data Space*), donde las aplicaciones pueden compartir información simplemente con leer o escribir objetos de datos a los que se les identifica con un tópico y/o una clave; procedimiento con el cual

las aplicaciones únicamente se reciben los datos que interesan sin más que subscribirse a dicho identificador de objeto de dato.

Utilizando DDS, se abstrae a los terminales de la red que los comunica, liberando al programador de tener en cuenta una topología de red determinada y haciendo irrelevante el número de equipos involucrados en el sistema, por lo que es altamente escalable.

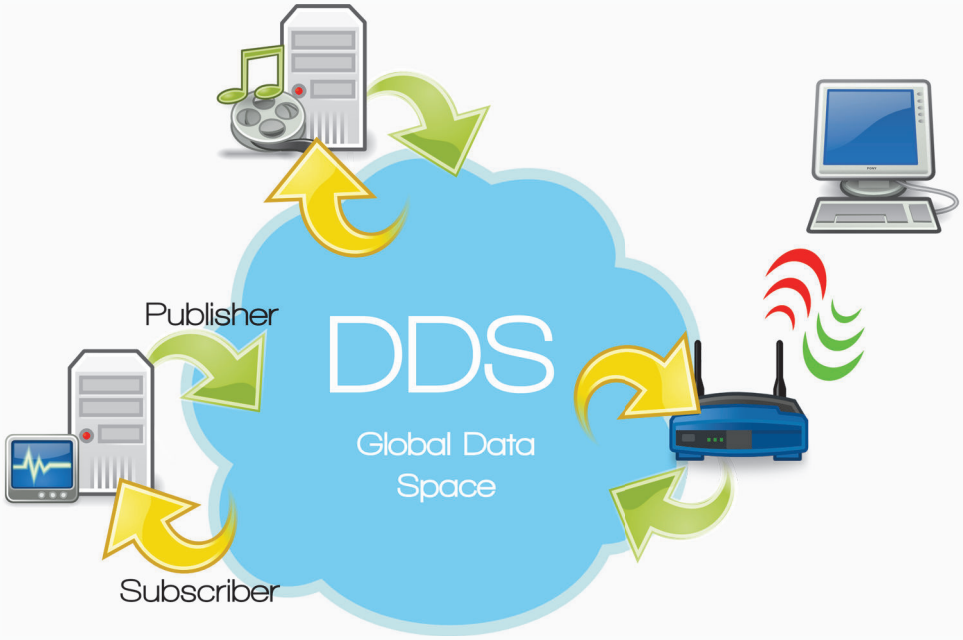

Figura 22: Funcionamiento de DDS.

Es considerado el primer protocolo que realmente implementa una comunicación distribuida en tiempo real dado que no solo pretende una comunicación desacoplada y descentralizada sino también una comunicación en tiempo real capaz de establecer unas especificaciones en las calidades de servicios formales propias de una aplicación en tiempo real. El objetivo principal del protocolo DDS es establecer un metodo eficaz de comunicación entre procesos que funcionan en tiempo real [25].

3.3.1 Arquitectura

DDS se define en base a dos niveles o capas, una primera llamada DCPS (*Data-Centric Publish-Subscribe*) de más bajo nivel dedicada a entregar eficientemente la información a los receptores correspondientes; y una capa superior llamada DLRL (*Data*

Local Reconstruction Layer) que permite la integración de DDS en la capa de aplicaciones.

Al usuario se presenta un sistema en el que todos los elementos principales se les denomina *participants* (participante en adelante), que son los que contribuyen a la red de datos. La forma en que pueden contribuir será recibiendo datos o enviándolos, por lo que cada participante podrá tener lo que se denomina *subscribers* y *publishers* respectivamente (subscriptor y publicador en adelante). Por debajo de este nivel se tiene entonces el tópico que será el identificador del tipo de dato de interés que se va a distribuir por la red. La tecnología empleada si bien se encarga de la comunicación de red, no dispone de la capacidad de encaminamiento, por lo que todos los equipos que participen deberán estar dentro de la misma red LAN (física o lógica), o en otras palabras, deben tener visibilidad de red.

Cada tipo de dato que se distribuya, tiene un tópico asociado que lo identifica unívocamente, y DDS da cierta libertad a la hora de asociarlos a los publicadores y subscriptores. Por ejemplo, se pueden tener todos los tópicos necesarios controlados por un único publicador, un publicador por cada tipo de dato, o cualquier combinación entre estos dos casos. En cualquier caso, cada tópico responderá ante un publicador o subscriptor, que a su vez lo engloba un participante.

Para las aplicaciones no existe ninguna red para comunicarse, puesto que DDS se encarga de que la comunicación sea transparente, y por tanto, supervisa la carga de transacciones, cuántos equipos están activos y a qué se han subscrito, etc. Es por ello que dispone de mecanismos para no colapsar la red, y uno de ellos consiste en no transmitir un tipo de dato si a éste no se ha subscrito nadie. Otros mecanismos quedan reservados al usuario como la elección de las calidades de servicio o QoS como se verá más adelante.

3.3.2 Implementaciones comerciales

Dado que DDS es un estándar, existen numerosas compañías que ofertan librerías para incorporar DDS. Se encuentran por ejemplo las versiones de MilSOFT DDS Middleware, PRISMTECH (OpenSplice), o la utilizada para llevar a cabo el presente proyecto: RTI DDS (Real-Time Innovations). También hay disponible una versión de software libre llamada OpenDDS.

Por lo general, las soluciones DDS (tal y como se define en el estándar) son descentralizadas o *peer-to-peer*. La comunicación entonces se efectúa mediante un protocolo

de descubrimiento vía multicast. Dos soluciones comerciales se desestiman por tanto ya que no cumplen el ser *peer-to-peer*:

- ▷ OpenSplice, porque necesita un demonio ejecutándose en cada nodo (aunque a partir de la versión 6 ya no es necesario).
- ▷ OpenDDS, ya que necesita de un demonio central.

Al final se eligió RTI DDS ya que daba facilidades frente a los demás en cuanto a licencias se refiere para trabajar en sistemas embebidos y sistemas operativos en tiempo real.

3.3.3 Librería CL

Como base para el proyecto, se desarrolló una capa software que envuelve la funcionalidad de DDS para facilitar su uso, ya que se encarga de automatizar el proceso de configuración de numerosos parámetros necesarios para realizar una comunicación con esta tecnología que para el caso del presente proyecto, siempre van a ser iguales.

Así, es una librería muy importante ya que agiliza el desarrollo de aplicaciones de alto nivel al abstraer al programador de un protocolo tan complejo como lo es DDS. En futuros apartados se detallará la librería CL, mostrando sus capacidades, arquitectura y diseño.

3.4 Conclusiones

Los sistemas en tiempo real se han ido integrando en la sociedad hasta tal punto que los tenemos presentes casi en cualquier momento: móviles, electrodomésticos, marcapasos y otras soluciones ortopédicas, sistemas de navegación (profesionales o personales para el día a día), etc.

La gran importancia de estos sistemas deriva en la contínua aparición de nuevos e innovadores desarrollos en este ámbito. Uno de ellos se unifica con los sistemas en tiempo real, ya que en la industria se suele presentar muy a menudo la necesidad de interconectar entre sí sistemas en tiempo real. Por ejemplo imaginar una planta solar, la cual necesita numerosos sensores, actuadores y controladores para supervisar en todo momento la potencia instantánea que se genera. Este tipo de planta abarca grandes

áres de terreno lo que supone un problema para las comunicaciones entre equipos y así controlar en tiempo real.

Una solución a este tipo de comunicaciones entre sistemas lo brinda DDS, que pone al alcance la posibilidad de comunicar sistemas heterogéneos e incluso en distintas redes si así se da el caso.