Desarrollo De Un Módulo Que Implemente Las Funcionalidades Del Protocolo RTPS Para Ser Utilizado en Aplicaciones de Tiempo Real

**Andrés X. RUBIO1, Alejandra B. TELLO2, Xavier A. CALDERÓN3**

*Facultad de Ingeniería Eléctrica y Electrónica, Escuela Politécnica Nacional*

1[andresrubiop@msn.com](mailto:andresrubiop@msn.com)

2[alejitat\_28@hotmail.com](mailto:alejitat_28@hotmail.com)

[3xavier.calderon@epn.edu.ec](mailto:3xavier.calderon@epn.edu.ec)

*Resumen*—El Proyecto de titulación comprende el desarrollo de un módulo que implementa funcionalidades del protocolo RTPS para aplicaciones distribuidas de tiempo real.

El módulo contempla una implementación básica del DDS, RTPS, y la interacción de los mismos, para el funcionamiento adecuado del middleware.

Primeramente se presenta el estado actual de los middleware de comunicaciones de tiempo real; realizando un breve estudio de cada tecnología incluyendo al middleware DDS[[1]](#footnote-1). Además se analiza las características y funcionalidades más específicas definidas en el estándar publicado por la OMG[[2]](#footnote-2) sobre DDS y RTPS.

Posteriormente se definen los requisitos necesarios para integrar el protocolo RTPS con el middleware DDS, realizando previamente un análisis de los **diferentes** paquetes RTPS[[3]](#footnote-3).

Se presenta un breve resumen del diseño que permite la interacción entre DDS y RTPS, el cual posteriormente es implementado.

Finalmente se presenta un ambiente de pruebas donde una implementación del protocolo es probada dentro de cuatro computadores que intercambian información.

# Introducción

En la actualidad los sistemas distribuidos de tiempo real se encuentran en una etapa de desarrollo donde se puede tomar varios caminos, es decir diferentes arquitecturas y tecnologías, de las cuales el protocolo de comunicación es parte vital en cada una de ellas, por lo que se requiere que al implementar la comunicación exista una correcta interacción entre el sistema distribuido y el protocolo de comunicación.

El presente proyecto se enfoca en diseñar, implementar y probar, el protocolo de comunicaciones RTPS, el cual es parte del middleware DDS.

# Marco Teórico

## Middleware de Comunicaciones

El Middleware es una capa intermedia de software, el cual se encarga de simplificar el manejo y la programación de aplicaciones, tratando de mantener la complejidad de redes y sistemas heterogéneos transparentes al usuario, por lo que se ha convertido en una herramienta esencial para el desarrollo de sistemas distribuidos.

Los Middleware de comunicaciones, son una abstracción de los detalles de bajo nivel relacionados con la distribución y la comunicación, el cual proporciona las bases para el desarrollo de Middlewares de alto nivel. Este maneja internamente los detalles del proceso de interconexión entre nodos que por lo general incluyen las siguientes características básicas:

* Direccionamiento o asignación de identificadores a entidades con la finalidad de indicar su ubicación.
* Marshalling[[4]](#footnote-4) o la transformación de los datos en una representación adecuada para la transmisión sobre la red.
* Envío o la asignación de cada solicitud a un recurso de ejecución para su procesamiento.
* Transporte o establecimiento de un enlace de comunicaciones para el intercambio de mensajes entre redes vía unicast o multicast.

En la Figura 1, se aprecia los servicios básicos que provee un Middleware.

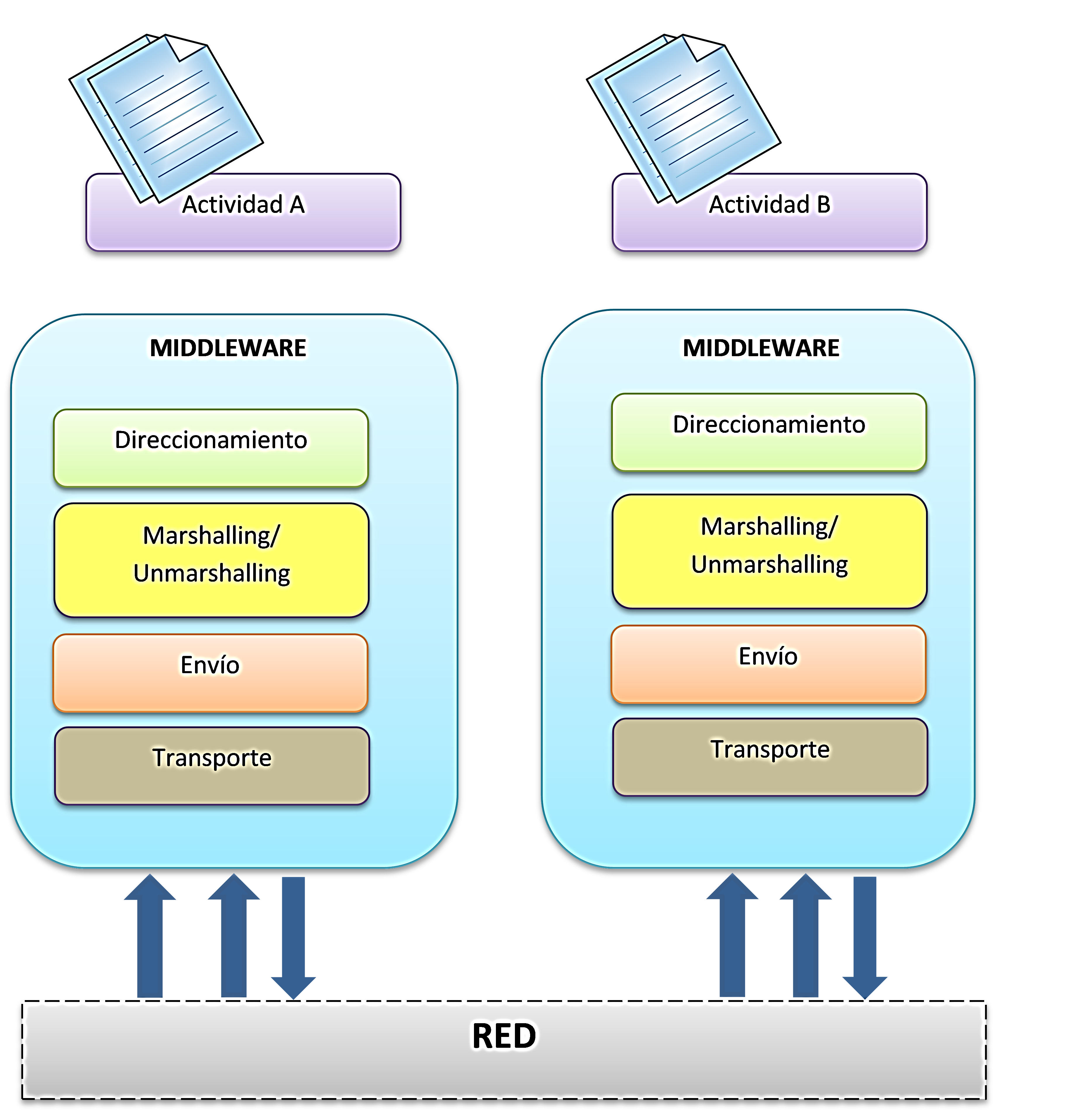


Figura . Servicios básicos provistos por el Middleware de distribución [1]

## Middlewares de Tiempo Real

Un sistema de tiempo real se define como un tipo especial de sistema cuya corrección lógica se basa tanto en la exactitud como también en la disminución de retardos en la información. En los sistemas de propósito general, el uso de la tecnología de Middlewares tiene como objetivo facilitar la programación de aplicaciones distribuidas. Con este fin, el Middleware proporciona una abstracción de alto nivel de los servicios ofrecidos por los sistemas operativos, sobre todo los relacionados con la comunicación.

Los desarrolladores son responsables de definir que parte de la aplicación puede ser accesible de forma remota, mientras el Middleware establece y gestiona transparentemente la comunicación entre los nodos del sistema distribuido. Además, los sistemas en tiempo real también se benefician de estas abstracciones de alto nivel.

### CORBA y RT-CORBA[[5]](#footnote-5)

CORBA [2] es un Middleware basado en el modelo de sistema distribuido DOM, el cual utiliza el paradigma Cliente-Servidor y cuya característica principal es facilitar la interoperabilidad entre aplicaciones heterogéneas[[6]](#footnote-6). Esta arquitectura está integrada por los siguientes componentes:

* Object Request Broker (ORB), representa el núcleo del Middleware y es responsable de coordinar la comunicación entre los nodos cliente y servidor.
* Interfaces del Sistema, estas consisten en un conjunto de interfaces agrupadas en función de su ámbito de aplicación.

A continuación se presenta en la Figura 1 una visión general de la arquitectura CORBA.

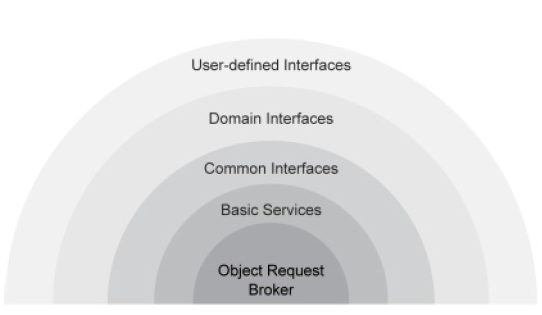


Figura . Arquitectura CORBA [1]

RT-CORBA nace a partir de las falencias de CORBA para tiempo real, las cuales se concentran en creación y destrucción de entidades en tiempo real, en los mecanismos de sincronización para controlar el acceso a recursos compartidos, y en los mecanismos para controlar el grado de concurrencia durante la ejecución de las llamadas remotas.

### The Ada Distributed System ANNEX [3]

Es un estándar internacional que incluye un anexo dedicado al desarrollo de aplicaciones distribuidas, la principal fortaleza de DSA es que el código fuente está escrito sin tener en cuenta de si va a ser ejecutado en una plataforma distribuida o en un solo procesador.

En el lenguaje de programación Ada cada parte de la aplicación se asigna de forma independiente a cada nodo la cual es llamada partición. Las particiones se comunican entre sí mediante el intercambio de datos a través las RPC y de los objetos distribuidos. La DSA define dos tipos de particiones: activos, los cuales pueden ser ejecutados en paralelo uno con otro; y pasivos, los cuales son particiones sin una tarea o hilo de control.

Los componentes de alto nivel del modelo de distribución de la DSA están ilustrada en la Figura 3 la cual representa el diagrama de secuencia de una llamada remota sincrónica entre dos particiones.

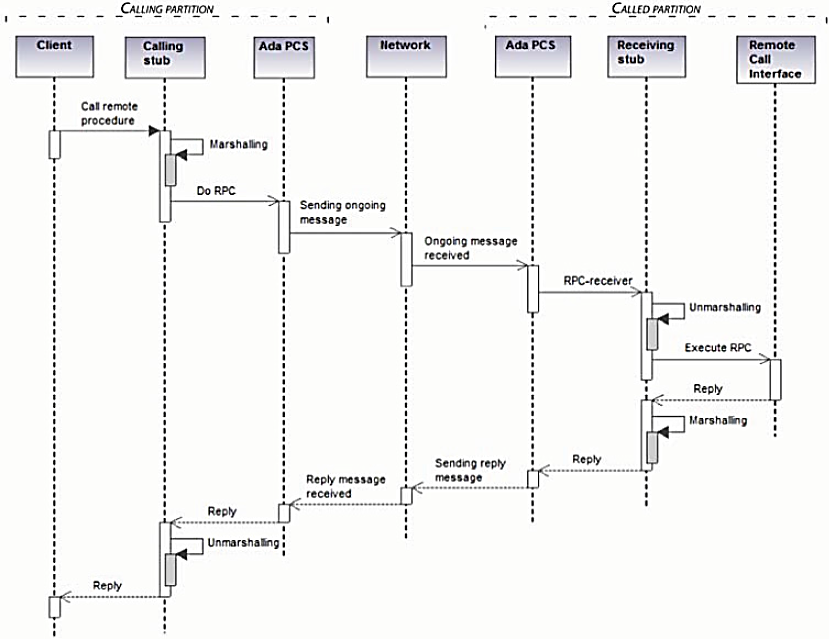


Figura . Diagrama de secuencia de una llamada remota síncrona [1]

### The Distributed Real-Time Specification for Java

Java fue diseñado inicialmente como un lenguaje de programación para sistemas de propósito general y por lo tanto tiene varios inconvenientes para el desarrollo de aplicaciones previsibles, especialmente en aspectos relacionados con la gestión de los recursos internos como la memoria o la programación del procesador [4]. Para los sistemas distribuidos de tiempo real, uno de los trabajos de investigación más notable es la Distributed Real-Time Specification for Java o DRTSJ [5], que integra las dos tecnologías existentes de Java:

* Real-Time Specification for Java [6], el cual logró el soporte en tiempo real a través de nuevas bibliotecas, un mejor mecanismo de Java, y una Máquina de Java en tiempo real.
* Remote Method Invocation [7], el cual define un modelo DOM basado en objetos Java que definen una nueva interfaz, llamada remota, permitiendo la diferenciación de objetos distribuidos de los locales. La arquitectura RMI se muestra en la Figura 4, que representa el diagrama de secuencia de una llamada remota asincrónica entre un cliente y un servidor.

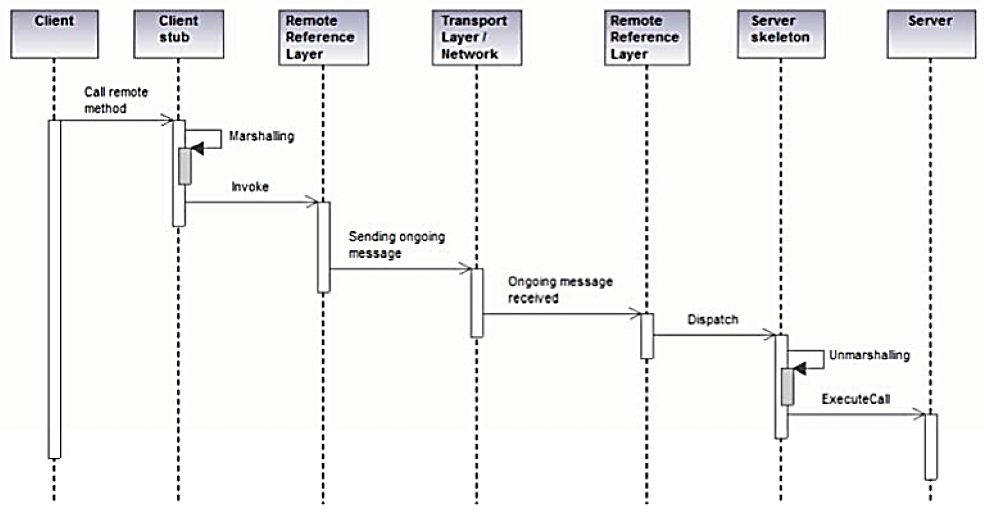


Figura . Diagrama de secuencia de una llamada remota asíncrona [1]

* Cliente Stubs o Proxy y Servidor Skeleton, los cuales representan la interfaz entre la capa aplicación y el resto del sistema RMI.
* Remote reference Layer, encargado de la semántica de las invocaciones remotas.
* Capa Transporte, utilizada para establecer las conexiones y gestionar los detalles de comunicación de bajo nivel.

### The Data Distriution Service for Real-Time Systems

El DDS [8] facilita el intercambio de datos en sistemas distribuidos a través del paradigma publicador-suscriptor. Una arquitectura publicador suscriptor promueve un bajo acoplamiento en la arquitectura de datos y es flexible y dinámica; es fácil de ser adaptada y extendida en sistemas basados en DDS, esta se conecta a los generadores de información anónima (el publicador) con los consumidores de información (el suscriptor).

Una visión general de la arquitectura se muestra en la Figura 5, está describe dos niveles de interfaces y un nivel de comunicaciones:

* Un nivel bajo denominado data-centric Publish-Subscribe o DCPS, el cual está orientado a la entrega eficiente de información adecuada a los destinatarios correctos.
* Un nivel opcional alto denominado data-local reconstruction layer o DLRL, el cual permite una integración simple a la capa aplicación.
* El nivel de comunicaciones se denomina DDS Interoperability Wire Protocol, el cual opera con el Protocolo RTPS.

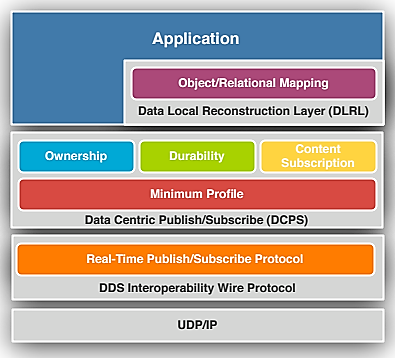


Figura . Arquitectura del Middleware DDS [9]

## Real-Time Publish-Subscribe Protocol

RTPS fue específicamente desarrollado para soportar los requerimientos únicos de los sistemas de datos distribuidos. La comunidad de automatización industrial define los requerimientos para un protocolo de Publicación-Suscripción que trabaje con DDS.

Las principales características del protocolo RTPS son:

* Rendimiento y Calidad de Servicio, los que permiten tener comunicación segura entre el Publicador y Suscriptor.
* Tolerancia a fallos para permitir la creación de redes sin puntos de fallos
* Extensibilidad para permitir que le protocolo sea extendido y mejorado con nuevos servicios.
* Conectividad Plug and Play para que las nuevas aplicaciones y servicios estén automáticamente descubiertos.
* Configurabilidad para permitir el balanceo de requerimientos para la confiabilidad y la puntualidad de cada entrega de datos.
* Capacidad para permitir que los dispositivos implemente un subconjunto del protocolo y que aun así participen en la red.
* Escalabilidad para sistemas que potencial mente escalen en redes extensas.
* Seguridad de tipo de datos para prevenir errores en la programación de aplicaciones.

En la Figura 6 se puede observar la interacción del protocolo RTPS con DDS por medio de los diferentes submódulos que lo componen.

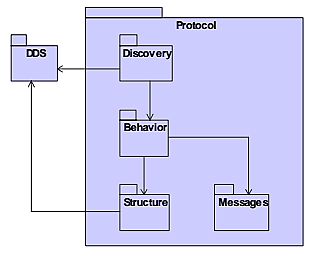


Figura . Módulos RTPS [1]

# Análisis de Requerimientos

Los requerimientos necesarios para soportar el protocolo RTPS con el Middleware DDS, son los siguientes:

* Se deberá implementar los componentes básicos que conforman el DDS, tales como el publicador, suscriptor y el topic.
* Se deberá implementar los mecanismos y técnicas para el alcance de la información, es decir se deberá organizar los datos dentro de cada dominio.
* Se deberá implementar los mecanismos de Lectura y Escritura de datos, y el ciclo de vida de los Topic.
* Se deberá implementar los componentes necesarios del protocolo RTPS y el mecanismo de descubrimiento proporcionado por el mismo.

A continuación se presenta una breve explicación sobre el contenido de cada requisito.

### Módulo DDS

Dentro del módulo DDS se encuentra al Publicador, al Suscriptor, y al Topic.

* El **Publicador** será el responsable de la distribución de datos, podrá publicar de los diferentes tipos de datos. Un *DataWriter* se encargará de comunicar a un publicador la existencia de datos de un tipo dado.
* El **Suscriptor** será el responsable de recibir los datos publicados y podrá recibir y despachar datos. Para acceder a los datos recibidos, la aplicación deberá utilizar un *DataReader* adjunto al suscriptor.
* Un **Topic** representará la unidad de información que puede ser producida o consumida; estará compuesta por un tipo, un nombre único y un conjunto de políticas de calidad de servicio, como se muestra en la Figura 7

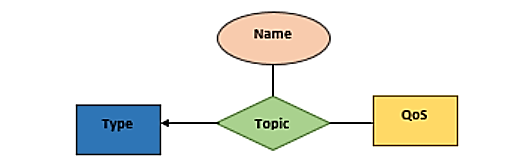
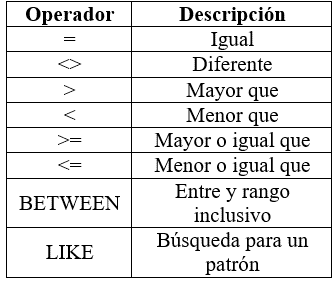


Figura . Objeto Topic y sus componentes. [9]

### Mécanismo y Técnicas Para el Alcance de la Información

Los dominios y las particiones serán la manera de organizar los datos. El Topic DDS permitirá crear *topics*, que limitará los valores que pueden tomar sus instancias. Al suscribirse a un *topic*una aplicación, sólo recibirá, entre todos los valores publicados aquellos valores que coincidan con el filtro del *topic*. Los operadores de los filtros y condiciones de consultas se muestran en la Tabla 1.

Tabla . Operadores para Filtros DDS y Condiciones de Consulta



Estos, limitarán la cantidad de memoria utilizada por el Middleware.

### Lectura y Escritura de Datos

#### Escritura de Datos

Escribir datos dentro del DDS será un proceso simple ya que solo se deberá llamar al método *write* del *DataWriter*. Este permitirá a una aplicación establecer el valor de los datos para ser publicados bajo un determinado Topic.

El ciclo de vida del *topic-instances*, podrá ser manejado implícitamente a través de la semántica implicada por el *DataWriter*, o podrá ser controlada explícitamente por la API *DataWriter*. La transición del ciclo de vida de *topic-instances*puede tener implicaciones en el uso de recursos locales y remotos.

#### Lectura de Datos

El DDS tendrá dos mecanismos diferentes para acceder a los datos, cada uno de ellos permitirá controlar el acceso a los datos y la disponibilidad de los mismos. El acceso a los datos se realizará a través de la clase *DataReader* exponiendo dos semánticas para acceder a los datos: *read* y *take*.

* La semántica del *read*implementado por el método *read*, dará acceso a los datos recibidos por el *DataReader*sin sacarlo de su caché local. Por lo cual estos datos serán nuevamente legibles mediante una llamada apropiada al *read*.
* La semántica del *take* implementado por los métodos *take*que permitirá acceder a los datos recibidos por el *DataReader*para removerlo de su caché local.

#### Notificaciones

Una forma de coordinar con DDS será tener un sondeo de uso de datos mediante la realización de un *read*o un *take*de vez en cuando. El sondeo podría ser el mejor enfoque para algunas clases de aplicaciones, el ejemplo más común es en las aplicaciones de control. En general, las aplicaciones podrán ser notificadas de la disponibilidad de datos o tal vez esperar su disponibilidad. El DDS apoyará la coordinación por medio de *waitsets*y los *listeners*.

* Los Waitsets, proporcionan un mecanismo genérico para la espera de eventos. Uno de los tipos soportados de eventos son las condiciones de lectura, las cuales podrán ser usadas para esperar la disponibilidad de los datos de uno o más *DataReaders*.
* Los Listeners, aprovechan al máximo los eventos planteados por el DDS y asincrónicamente notificar a los *handler*(controladores) registrados. Por lo tanto, si se quiere un *handler*sea notificado de la disponibilidad de los datos, se deberá conectar el *handler*apropiado con el evento *on\_data\_available*planteado por el *DataReader*.

### Módulo RTPS

Los módulos RTPS están definidos por la PIM. La PIM describe el protocolo en términos de una “máquina virtual.” La estructura de la máquina virtual está construida por clases, las cuales están descritas en el módulo de estructura, además este incluye a los *endpoints* de los *Writer* y *Reader*. Estos extremos se comunican usando los mensajes descritos en el módulo de mensajes. También es necesario describir el comportamiento de la máquina virtual, por medio del módulo de comportamiento, en el cual se observa el intercambio de mensajes que debe tomar lugar entre los extremos. Por último se encuentra el protocolo de descubrimiento usado para configurar la máquina virtual con la información que esta necesita para comunicar a los pares remotos, este protocolo se encuentra descrito en el módulo de descubrimiento.

#### Módulo Estructura

El propósito principal de este módulo es describir las clases principales que serán utilizadas por el protocolo RTPS, como se puede observar en la Figura 8.

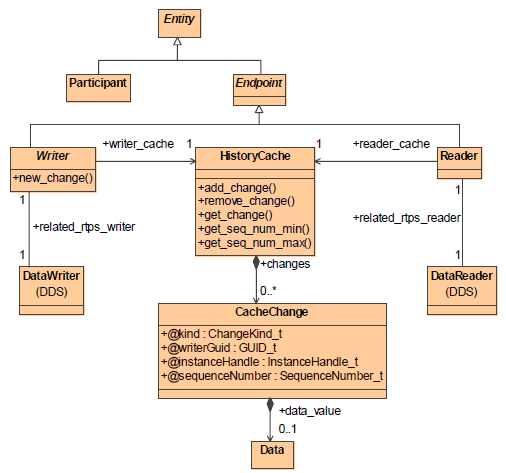


Figura . Módulo Estructura

#### Módulo Mensajes

Este módulo describe la estructura y contenidos lógicos globales de los mensajes que se intercambian entre los puntos finales del *Writer* RTPS y los puntos finales del *Reader* RTPS. Los mensajes RTPS son de diseño modular y se pueden ampliar fácilmente para apoyar tanto nuevas características del protocolo, así como extensiones específicas del proveedor.

#### Estructura general del Mensaje RTPS

Consta de una cabecera RTPS de tamaño fijo, seguido de un número variable de Submensajes RTPS. Cada submensaje a su vez consta de un *SubmessageHeader* y un número variable de *SubmessageElements.* Esto se muestra en la Figura 9.

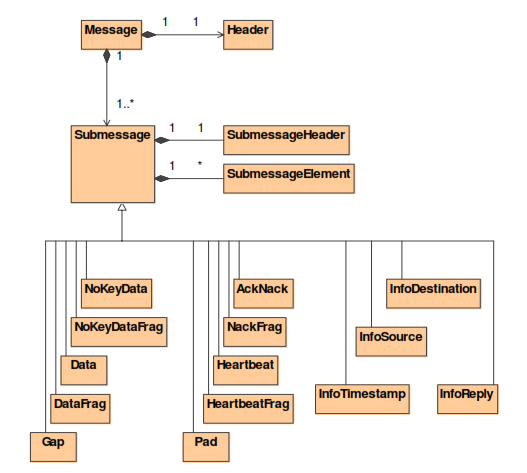


Figura . Estructura del mensaje RTPS

#### Elementos del submensaje RTPS

Cada mensaje RTPS contiene un número variable de submensajes RTPS. Cada submensaje RTPS a su vez, se construye a partir de un conjunto de bloques llamados *SubmessageElements*. El RTPS versión 2.2 define los siguientes elementos: submensaje *GuidPrefix, entityId, sequenceNumber, SequenceNumberSet, FragmentNumber, FragmentNumberSet, VendorID, ProtocolVersion, LocatorList, TimeStamp, Count, SerializedData* y *ParameterList*.

#### Submensajes RTPS

El protocolo RTPS de la versión 2.2 define varios tipos de submensajes. Se clasifican en dos grupos: EntitySubmessages e Interpreter-Submessages.

El *Entity submessage* se dirige a una Entidad RTPS. El *Interpreter-Submessages* modifica el estado del receptor RTPS y proporcionará el contexto que ayuda a los procesos posteriores del *Entity submessage*.

Las entidades del submensaje son:

* El submensaje *Data*, contiene información sobre el valor de un objeto fecha de la aplicación. Los submensajes de datos son enviados por el *Writer* a un *Reader*.
* El *DataFrag*, equivale a los datos, pero sólo contiene una parte del valor (uno o más fragmentos). Permite que los datos se transmitan como varios fragmentos para superar las limitaciones de tamaño de mensajes de transporte.
* El *HeartBeat*, describe la información que está disponible en un *Writer*. Los mensajes *HeartBeat* son enviados por un *Writer* a uno o más *Reader*.
* El *HeartbeatFrag*, sirve para los datos fragmentados, describe que fragmentos están disponibles en un Writer. Los submensajes *HeartbeatFrag* son enviados por un *Writer* a uno o más *Reader*.
* El *GAP*, describe la información que ya no es relevante para el *Reader*. Los mensajes *GAP* son enviados por un *Writer* a uno o más *Reader*.
* El *AckNack*, proporciona información sobre el estado de un *Reader* a un *Writer*. Los mensajes *AckNack* son enviados por un *Reader* a uno o más *Writer*.
* El *NackFrag,* proporciona información sobre el estado de un *Reader* a un *Writer*, específicamente los fragmentos de información que siguen perdidos en el *Reader*. Los submensajes *NackFrag* son enviados por un *Reader* a uno o más *Writer*.

Los submensajes de interpretación son:

* El *InfoSource,* proporciona información acerca de la fuente de donde se originaron los Entity Submessage posteriores. Este submensaje se utiliza principalmente para la retransmisión de los submensajes RTPS.
* El *InfoDestination*, proporciona información sobre el destino final de los submensajes que le acompañan. Este submensaje se utiliza principalmente para la retransmisión de submensajes RTPS.
* El *InfoReply*, proporciona información sobre donde responder a las entidades que figuran en submensajes posteriores.
* El *InfoTimestamp*, proporciona una marca de tiempo a los submensajes que le acompañan.
* El *Pad,* se utiliza para agregar relleno a un mensaje, siempre y cuando sea necesaria la alineación de la memoria.

### Módulo Comportamiento

Una ver que el *Writer* RTPS sea asociado a un *Reader* RTPS, es responsabilidad de ambos, asegurarse que los cambios en el *CacheChange* que existen en la *HistoryCache* de los diferentes *Writers* sean propagados a la *HistoryCache* de los diferentes *Readers*.

Este módulo describe como los pares de *Writer y Reader* RTPS asociados deben comportarse para propagar los cambios en el *CacheChange*. Este comportamiento está definido en términos de los mensajes intercambiados usando a los mensajes RTPS.

#### Requerimientos Generales

Los siguientes requerimientos aplican a todas las entidades RTPS.

* Todas la comunicaciones deberán tomar lugar usando mensajes RTPS, es decir que ningún otro mensaje que no está definido en los mensajes RTPS puede ser usado.
* Se debe implementar un *Message Receiver* RTPS, es decir que para interpretar a los submensajes RTPS se deberá usar esta implementación.
* Las características de tiempos en todas las implementaciones deben ser configurables.
* Se debe implementar el protocolo de descubrimiento denominado *Simple Participant and Endpoint Discovery Protocols*, es decir a los protocolos de descubrimientos que cubre el estándar.

### Módulo Descubrimiento

El módulo descubrimiento definirá el protocolo de descubrimiento RTPS. El propósito del protocolo de descubrimiento permitirá que cada *participante* RTPS descubra otros relevantes *participantes* y sus *endpoint*. Una vez que el *endpoint* ha sido descubierto, las implementaciones podrán configurar *endpoint* locales para establecer la comunicación.

La especificación RTPS divide al protocolo de descubrimiento en dos protocolos independientes:

* *Participant* Discovery Protocol (PDP)
* *Endpoint* Discovery Protocol (EDP)

Un PDP especifica como los participantes se descubren entre sí en la red. Una vez que dos *participantes* se han descubierto, intercambian información sobre los *endpoint* que los contienen utilizando un EDP. Aparte de esta relación de causalidad, ambos protocolos se pueden considerar independientes.

A fin de la interoperabilidad, todas las implementaciones RTPS deben proporcionar al menos los siguientes protocolos de descubrimiento:

* Simple Participant Discovery Protocol (SPDP), el objetivo es descubrir la presencia de otros participantes en la red y sus propiedades
* Simple Endpoint Discovery Protocol (SEDP), define la información intercambiada requerida entre dos *participantes* para descubrir *Writer* y *Reader Endpoint.*

Ambos son protocolos básicos de descubrimiento que bastan para pequeñas redes de mediana escala. Los PDP adicionales y EDP que están orientados hacia las redes más grandes se pueden añadir a las futuras versiones de la especificación.

Finalmente, el rol de un protocolo de descubrimiento será proporcionar información sobre *endpoint* remotos descubiertos.

## Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

# Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

## Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

## Units

* Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as “3.5-inch disk drive.”
* Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
* Do not mix complete spellings and abbreviations of units: “Wb/m2” or “webers per square meter,” not “webers/m2.” Spell units when they appear in text: “...a few henries,” not “...a few H.”
* Use a zero before decimal points: “0.25,” not “.25.” Use “cm3,” not “cc.” (*bullet list*)

## Equations

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in (1), using a right tab stop. To make your equations more compact, you may use the solidus ( / ), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in

*a**b*    

Note that the equation is centered using a center tab stop. Be sure that the symbols in your equation have been defined before or immediately following the equation. Use “(1),” not “Eq. (1)” or “equation (1),” except at the beginning of a sentence: “Equation (1) is ...”

## Some Common Mistakes

* The word “data” is plural, not singular.
* The subscript for the permeability of vacuum **0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter “o.”
* In American English, commas, semi-/colons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
* A graph within a graph is an “inset,” not an “insert.” The word alternatively is preferred to the word “alternately” (unless you really mean something that alternates).
* Do not use the word “essentially” to mean “approximately” or “effectively.”
* In your paper title, if the words “that uses” can accurately replace the word using, capitalize the “u”; if not, keep using lower-cased.
* Be aware of the different meanings of the homophones “affect” and “effect,” “complement” and “compliment,” “discreet” and “discrete,” “principal” and “principle.”
* Do not confuse “imply” and “infer.”
* The prefix “non” is not a word; it should be joined to the word it modifies, usually without a hyphen.
* There is no period after the “et” in the Latin abbreviation “et al.”
* The abbreviation “i.e.” means “that is,” and the abbreviation “e.g.” means “for example.”

An excellent style manual for science writers is [7].

# Using the Template

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Authors and Affiliations

The template is designed so that author affiliations are not repeated each time for multiple authors of the same affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization). This template was designed for two affiliations.

### For author/s of only one affiliation (Heading 3): To change the default, adjust the template as follows.

#### Selection (Heading 4): Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select “1 Column” from the selection palette.

#### Deletion: Delete the author and affiliation lines for the second affiliation.

### For author/s of more than two affiliations: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the “Columns” icon from the MS Word Standard toolbar and then select “1 Column” from the selection palette.

#### Highlight author and affiliation lines of affiliation 1 and copy this selection.

#### Formatting: Insert one hard return immediately after the last character of the last affiliation line. Then paste down the copy of affiliation 1. Repeat as necessary for each additional affiliation.

#### Reassign number of columns: Place your cursor to the right of the last character of the last affiliation line of an even numbered affiliation (e.g., if there are five affiliations, place your cursor at end of fourth affiliation). Drag the cursor up to highlight all of the above author and affiliation lines. Go to Column icon and select “2 Columns”. If you have an odd number of affiliations, the final affiliation will be centered on the page; all previous will be in two columns.

## Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include ACKNOWLEDGMENTS and REFERENCES, and for these, the correct style to use is “Heading 5.” Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract,” will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1,” “Heading 2,” “Heading 3,” and “Heading 4” are prescribed.

## Figures and Tables

### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1,” even at the beginning of a sentence.

1. Table Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. *(Table footnote)*
2. Example of a figure caption. *(figure caption)*

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization,” or “Magnetization, M,” not just “M.” If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization (A ( m(1),” not just “A/m.” Do not label axes with a ratio of quantities and units. For example, write “Temperature (K),” not “Temperature/K.”

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g.” Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

# Referencias

|  |  |
| --- | --- |
| [1] | H. Pérez y J. J. Gutiérrez, «A survey on standards for real-time distribution middleware,» *ACM Computing Surveys,* vol. 46, nº 49, p. 39, Marzo 2014. |

1. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi resolution TIFF or EPS file with all fonts embedded) because this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

1. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
2. I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
3. K. Elissa, “Title of paper if known,” unpublished.
4. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
5. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
6. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

1. DDS, Data Distributed System [↑](#footnote-ref-1)
2. OMG, Object Management Group [↑](#footnote-ref-2)
3. RTPS, Real-Time Publish-Suscribe Protocol [↑](#footnote-ref-3)
4. Marshalling, es un mecanismo ampliamente usado para transportar objetos a través de una red. [↑](#footnote-ref-4)
5. RT-CORBA, CORBA de Tiempo Real [↑](#footnote-ref-5)
6. Aplicaciones Heterogéneas, se refiere a las aplicaciones codificadas en diferentes lenguajes de programación, ejecución en diferentes plataformas y/o las implementaciones de Middlewares desarrolladas por diferentes empresas. [↑](#footnote-ref-6)