Sistema para la distribución de objetos en entornos publisher/subscriber

Alejandro Ariel Pettinelli

Trabajo final de la materia Sistemas Distribuidos I
Facultad de Ingeniería de la Universidad de Buenos Aires
alejandroarielp@gmail.com

RESUMEN

En este artículo se propone un modelo para el diseño y desarrollo de una plataforma de bajo costo para la distribución de objetos en entornos publisher/subscriber, donde la capacidad de procesamiento es limitada. Este trabajo es el resultado de un exhaustivo análisis sobre modelos de transmisión de objetos y/o eventos existentes en la actualidad, destacando las ideas que son de utilidad y descartando aquellas que no aportan beneficios en el ambiente para el que fue ideado.
El sistema permite la distribución de objetos pertenecientes a clases previamente definidas por el usuario. Su arquitectura es un MOM orientado a objetos**,** basado en una Red Lógica constituida por nodos que forman una red escalable y transparente ante el cliente, en lo que a la mecánica de distribución de objetos se refiere. Este encapsulamiento favorece a que las cuestiones referentes al ruteo de los mensajes y al control de errores sean más eficientes.
Finalmente, los resultados mencionados en las conclusiones pudieron ser validados satisfactoriamente mediante un prototipo desarrollado en C++ y desplegado en un entorno LINUX, en los laboratorios de la facultad.

Palabras Claves: Publisher/Subscriber, redes peer-to-peer, rendezvous.

I. INTRODUCCIÓN

El objetivo de este trabajo es tener una idea sobre los lineamientos generales que se deben tener en cuenta al momento de desarrollar un *middleware*[1] para la distribución de objetos en entornos publish/subscribe[2], de bajo costo tanto operativo como de desarrollo.

Tal como es el caso de Hermes[3] o Siena[4], muchas implementaciones tratan de mostrar una red lógica a la cual le llegan los datos de sus clientes (denominados eventos), que luego se transforman en mensajes dentro del middleware y son transportados por el mismo. Particularmente, Hermes es un middleware de estas características, cuya arquitectura general se basa en una Red Lógica de Ruteo (*Overlay Routing Network*), formada por nodos de distribución de eventos que brindan conectividad a los clientes y reenvían los datos que éstos publican. La finalidad de esta red es, primero, brindar transparencia en la transmisión de datos y, en segunda instancia, no utilizar mensajes de difusión (*Broadcast*) para la notificación masiva. Éste último factor hace de este conocido middleware una solución tan escalable que permite su despliegue sobre Internet.

El manejo de eventos en Hermes requiere tres niveles de abstracción, lo cual puede ser contraproducente cuando el sistema debe desplegarse en contextos con escasa capacidad de procesamiento, como ser sensores de bajo consumo en aplicaciones industriales, aplicaciones en celulares, etc., donde lo importante es generar el mensaje y que la red se encargue de su distribución de forma transparente.

En otras aplicaciones, tales como Control de Acceso o prevención de spam en entornos Publisher/Subscriber [5, 6], el concepto de "evento" es indistinto, ya que la finalidad del trabajo no depende de este mecanismo y, si bien la capacidad de procesamiento es suficiente, el uso de este middleware puede ser contraproducente, en lo que a cuestiones de performance se refiere. La mecánica de herencia de eventos y el filtrado por atributos son otras funcionalidades de esta plataforma que pueden resultar innecesarias en estos ámbitos.

Partiendo de estas ideas se diseñó un *middleware* de tipo MOM (*Message Oriented Middleware*)[7] para la distribución de objetos basado en una Red Lógica de Interconexión de tipo peer-to-peer. En esta plataforma, los Clientes Publicadores (*Publishers*) registran sus clases de las cuales, posteriormente, publicarán los objetos. Por otro lado, los Clientes Subscriptores (*Subscribers*) reciben los objetos publicados por los anteriores. La idea de "transparencia", a la hora de la transmisión de los objetos, habla claramente del concepto de MOM.

Este tipo de middleware se ve como una cola de transmisión, de la cual se obtiene la información que se desea recibir y se colocan los datos a transmitir, dejando todo el trabajo a las capas inferiores. Este tipo de operatoria requiere de persistencia de los datos en del middleware, lo cual favorece a las cuestiones de tolerancias a fallos.

La comunicación entre los clientes se diseñó mediante una Red Lógica de Interconexión, formada por nodos (*Brokers*) interconectados entre sí, a modo de brindar una capa de abstracción para la distribución de los objetos. Dentro de la plataforma, los nodos son las entidades que permiten el acceso de los clientes al servicio. Éstos permiten desde la conexión de los distintos clientes a la Red (mediante los Nodos Internos periféricos de la misma) hasta la redirección (ruteo), dentro de la Red, de los objetos publicados hacia los clientes asociados a ellos, mediante los Nodos Internos y los Nodos de Encuentro (*Rendezvous Brokers*). Más adelante se verá que el uso de los Nodos de Encuentro ayuda a evitar el uso de mensajes de tipo *Broadcast*, permitiendo, de esta manera, una amplia escalabilidad del middleware.

Las restricciones impuestas por el contexto de trabajo al alcance del middleware, conllevan a una notoria reducción de los costos de implementación y procesamiento, siendo esta reducción una de las principales ventajas de este proyecto.

A continuación se detalla la arquitectura del middleware, luego se describe la Red Lógica de Interconexión (sección 3), posteriormente se detalla la mecánica de identificación de Nodos y Ruteo de mensajes (secciones 4 y 5) y, finalmente, se desarrolla el concepto de los Nodos de Encuentro, en la sección 6.

II. ARQUITECTURA GENERAL

En una primera visión general, la idea principal es brindar simplicidad a la hora de transmitir y recibir los objetos. El middleware brinda al cliente una simple API (*Application Programming Interface*), la cual permite al cliente ingresar a la Red Lógica, registrar una nueva clase de objeto, que pueda registrarse para publicar

objetos de una clase ya registrada, subscribirse a una cierta clase y, finalmente, transmitir y recibir objetos de dichas clases. Los clientes acceden a la Red Lógica a través de un Nodo Interno periférico de la misma. La transmisión y recepción de los objetos, por ser un mecanismo similar a una cola, garantiza la entrega de los objetos en orden y acota el tiempo que el cliente se encuentra bloqueado durante dicha operación.

Los Nodos de la Red Lógica de Interconexión se encuentran interconectados entre sí de forma arbitraria, pueden ser de dos tipos. Los Nodos Internos, que son los encargados de brindar conectividad a los clientes a la Red Lógica, redirigir los mensajes y formar caminos virtuales entre los clientes. Sólo aquellos nodos que se encuentran en la periferia de la Red Lógica son aquellos que brindan conectividad. Por el otro lado, los Nodos de Encuentro son nodos internos especiales de la Red Lógica cuya funcionalidad está orientada a la distribución de los mensajes y a formar Puntos de Encuentro entre los mensajes publicados y los subscriptores (se describe con más detalle en la sección 6). Para cada clase registrada existe un único Nodo de Encuentro asociado para su distribución y, para ubicarlo, se utiliza una función de hash que determina el identificador de nodo a partir de su nombre de clase (véase figura 1).

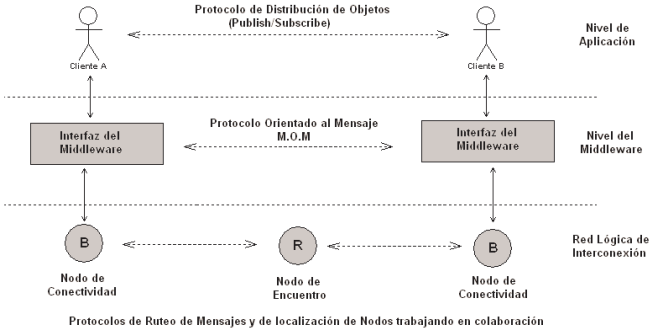

FIGURA 1
ARQUITECTURA GENERAL DEL MIDDLEWARE

Esta abstracción en una Red Lógica de Interconexión, oculta al cliente tanto la mecánica de envío de mensajes así como quién es el responsable de dicha tarea. Además, esta abstracción facilita la implementación de sistemas sobre cualquier tipo de sistema de comunicación existente (IP, X.25, etc.).

III. RED LÓGICA DE INTERCONEXIÓN

La Red Lógica de Interconexión es la capa que engloba toda la funcionalidad de ruteo de los objetos de los clientes. Está formada por Nodos Internos (caracterizados por un identificador) los cuales son los encargados del ruteo de los mensajes. Éstos se interconectan arbitrariamente y van conformando de forma dinámica la estructura de la red mencionada, trabajando sobre cualquier tipo de capa de red estándar (IP, X25, etc.). Una posible topología de la Red Lógica se muestra en la Figura 2.

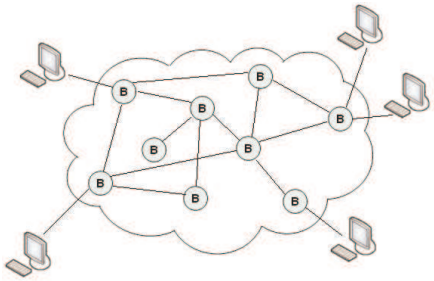

FIGURA 2
POSIBLE TOPOLOGÍA DE LA RED LÓGICA.

El ruteo necesita un protocolo que mantenga actualizados los datos de una topología de red cambiante. Existen protocolos de ubicación de objetos como Prasty[8] que incluyen la mecánica de actualización de la estructura de la red o se pueden utilizar las ideas de protocolos de ruteo ya existentes como RIP u OSPF[9]. Debe tenerse en cuenta que dicho protocolo de ruteo puede afectar (reducir) considerablemente el ancho de banda disponible. Además de las tareas mencionadas, los Nodos Internos tienen la labor de brindar conectividad a los clientes de la red y de trazar "caminos virtuales" entre éstos y los Nodos de Encuentro. En la sección 5 se describirán en detalle las decisiones tomadas acerca del ruteo de mensajes en este trabajo.

Otros protocolos de tipo multicast o Servicios de notificación de eventos [10,11] son modelos válidos para determinar la localización de los Nodos, pero el uso de mensajes de tipo *Broadcast,* para la difusión del estado lógico de la red, reduce la escalabilidad de esta misma. Una solución más eficiente, y con la misma funcionalidad, es la del uso de Nodos de Encuentro.

Estos Nodos tienen la finalidad de concentrar los mensajes de registro, publicación y suscripción para una clase de objeto registrada en el middleware. Cuando los clientes solicitan realizar operaciones sobre los objetos, el middleware recibe los datos y le solicita a su Nodo Interno el identificador del Nodo de Encuentro asociado a esa clase de objeto. Con esta información, el middleware arma y transmite el mensaje, a través de los Nodos de Interconexión, hasta su Nodo de Encuentro, el cual lo diseminará a todos los clientes suscriptos. El Nodo que le brinda conectividad al cliente consigue el identificador del Nodo de Encuentro por medio de una función de hash utilizando, como parámetro, el nombre de la clase asociada al objeto que el middleware le indica.

IV. IDENTIFICACIÓN DE NODOS

La idea general para la identificación de nodos y el ruteo de los mensajes adoptada en este proyecto es similar a la que utiliza Prasty[8]. Cada nodo de la red está identificado con un número de 128 bits (denominado IDNodo), elegido al azar por el mismo nodo cuando se registra dentro de la Red Lógica la primera vez. El espacio de identificadores posibles (debido al gran tamaño del mismo) sumado a la uniformidad de distribución de claves, que ofrecen las funciones hash, reducen en una alta probabilidad la posibilidad de que se produzcan colisiones de número de IDNodo.

En principio, todo Nodo nuevo debe conectarse a la Red Lógica a través de un Nodo existente en la red, ubicado en la periferia de la misma. En el momento de registrarse, el Nodo nuevo elige un número X al azar, generalmente obtenido mediante un hash criptográfico con algún parámetro propio (su nombre, etc.) y le envía dicha información a su Nodo de conexión, para que le devuelva sus tablas de ruteo y así inicializar su estado. A continuación, el Nodo que brinda acceso agrega a este Nodo nuevo en su información de ruteo, notifica a sus vecinos la novedad y así continúa la diseminación de la información de ruteo en la red. Dicha información permite mantener actualizada la Tabla Hash Distribuida (DHT).

Si el nodo desea retirarse de la Red Lógica, notifica a sus vecinos y, si es un Nodo de Encuentro, entrega toda la información acerca de esta tarea al Nodo vecino con IDNodo más cercano. Esta notificación de salida hace que toda la red quede informada y actualice sus tablas. Para que los Nodos estén informados del estado de los enlaces con sus vecinos y del estado de estos, se necesita de algoritmos del tipo Keep-alive[12].

V. Ruteo de Mensajes

El ruteo de los mensajes es una tarea relativamente simple en este contexto. Para ello, los Nodos tienen una tabla de ruteo, la cual indica el nodo siguiente en el camino en base al IDNodo indicado.

Cuando no se dispone de un camino especificado en la misma, la política asumida por el Nodo es enviárselo al Nodo vecino con número de identificación más cercano al de destino y asentarlo en su tabla. Este protocolo asegura una cantidad de saltos *O(logN)* para que un mensaje llegue a su destino, en condiciones normales. Así, la misma red va creando los caminos de ruteo para los mensajes, tema que será de mucha ayuda a la hora de registrar nuevas clases de objetos a ser transferidos.

Nótese aquí la independencia que brinda la red de aplicación de la red física donde se despliega, pudiendo, inclusive, aprovecharse de funcionalidades de la red de comunicaciones con la que se esté trabajando, como, por ejemplo, el uso de protocolos de ruteo en IP, los cuales reducen enormemente los costos de implementación y, de ninguna manera, afectan a la funcionalidad brindada al Cliente de la Red.

VI. Puntos de Encuentro

En esta sección, se detallará la principal ventaja de este artículo, en lo que a escalabilidad se refiere, que son los Nodos de Encuentro (*Rendezvous Brokers*).

Los Clientes Publicadores registran nuevas clases de objetos o se registran para enviar objetos de una cierta clase ya existente. Luego publican objetos de esa clase, que son recibidos por los Clientes Subscriptores adheridos a la misma. Como puede observarse, existe una limitada cantidad de tipos de mensajes asociados a una cierta clase en particular. La solución trivial sería utilizar mensajes tipo *Broadcast* para que todos estuvieran actualizados de las nuevas clases registradas o de los nuevos clientes asociados a una clase existente, pero esta idea atenta contra la escalabilidad del middleware, ya que existen redes donde no existen los mensajes de este tipo.

El Nodo de Encuentro es una entidad dentro de la Red Lógica que tiene como finalidad concentrar y administrar los mensajes de una cierta clase registrada. Por lo

tanto, existe un único Nodo de Encuentro asociado a cada clase registrada por los usuarios, localizado por la función de hash distribuida. De esta manera se obtiene el máximo nivel de abstracción provisto por la red, ya que los clientes no se refieren a orígenes o destinatarios de los objetos, sino que se refieren a clases de objetos a transferir.

Los Nodos de Interconexión, en base al IDNodo del Nodo de Encuentro asociado a la clase referida, generan caminos virtuales, dentro de la Red lógica, desde los usuarios hasta él. El Nodo de Encuentro administra el contenido de todos los mensajes referidos a la clase que le corresponde (registración de nueva clase, registración de un Cliente publicador o de un Cliente Subscriptor). De esta manera, el Nodo de Encuentro sabe a que clase esta asociado, quienes son sus publicadores y quienes sus suscriptores. Su único trabajo, de ahora en más, es recibir y enviar a quien le corresponda, según se puede ver en la Figura 3.

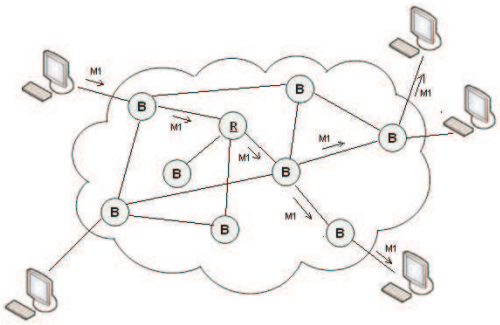

FIGURA 3
DISTRIBUCIÓN DE UN OBJETO DENTRO DEL MIDDLEWARE

VII. CONCLUSIONES

La simplicidad que existe detrás del concepto peer-to-peer llevó a los diseñadores de software a desplegar sistemas de estas características en ámbitos cada vez más remotos. Sin dudas, ésta fue la principal causa de la evolución de los sistemas y middlewares de este tipo en el ambiente de desarrollos de sistemas distribuidos. Día a día pueden observarse nuevos proyectos relacionados con este tipo de sistemas, los cuales intentan focalizarse en dar solución a problemas en contextos acotados, esa fue la idea tras la cual se trabajó durante el diseño de este proyecto.

Durante el transcurso de este trabajo se describieron las decisiones generales para el diseño y desarrollo de un middleware para la distribución de objetos en entornos Publish/Subscribe. El principal aporte fue el limitar el alcance del middleware, basado principalmente en el contexto donde puede ser desplegado, pero sin perder de vista los avances ya realizados en otros proyectos, de modo tal de obtener una solución de bajo costo, tanto de desarrollo como operativo, y de una fortaleza similar a la de otros proyectos de mayor envergadura.

La principal ventaja de este middleware es la escalabilidad. La arquitectura transparente de su Red Lógica de Interconexión no solo permite un crecimiento

horizontal, en lo que a cuestiones de despliegue se refiere, sino también vertical, ya que permite seguir generando niveles de abstracción por encima del middleware, a modo de llegar a un plataforma orientada a eventos para la distribución de objetos, compatible con los clientes de menor capacidad de procesamiento.

El prototipo desarrollado y desplegado en el laboratorio de la facultad permitió corroborar los resultados de este trabajo, permitiendo el correcto intercambio de objetos entre sistemas desarrollados en PCs con capacidad de procesamiento limitado (procesadores Pentium de 166 Mhz con 128 MB de memoria RAM, por ejemplo), sin que la performance de la Red Lógica se viera afectada.

VIII. REFERENCIAS

[1]. Wikipedia Middleware Definition. http://en.wikipedia.org/wiki/Middleware.

[2]. Wikipedia Publish/Subscribe Definition. http://en.wikipedia.org/wiki/Publish/subscribe.

[3]. P. R. Pietzuch and J. Bacon. *Hermes: distributed event-based middleware architecture*. Proc. 1st International Workshop on Distributed Event-Based Systems, 2002.

[4]. A. Carzaniga, J. Deng, and A. L. Wolf. *Fast Forwarding for Content-Based Networking*. Technical report, Dept. of Computer Science, Univ. of Colorado, Nov. 2001.

[5]. András Belokosztolszki, David M. Eyers, Peter R. Pietzuch, Jean Bacon, Ken Moody. *Role-Based Access Control for Publish/Subscribe Middleware Architectures*. Second International Workshop on Distributed Event-Based Systems (DEBS'03).

[6]. Sasu Tarkoma. *Preventing Spam in Publish/Subscribe*. Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06).

[7]. A, Tanenbaum, M. van Steen. *Distributed Systems. Principles and Paradigms*. Prentice Hall. 2002. Cap 2.

[8]. A. Rowstron and P. Druschel. *Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer Systems*. In Proc. of Middleware 2001, Nov. 2001.

[9]. William Stallings. *Data and Computer Communications*. Pretince Hall. 2006. Séptima Edición. Cap. 19.

[10]. G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. *An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems*. In ICDCS, 1999.

[11]. A. Carzaniga, D. S. Rosenblum, and A. L.Wolf. *Design and Evaluation of a Wide-Area Event Notification Service*. ACM Trans. on Computer Systems. 2001.

[12]. Richard Stevens. *TCP/IP Illustrated, Vol. 1*. Addison - Wesley. 2000. Cap. 23.