SciVerse ScienceDirect
Disponible en www.sciencedirect.com

Revista Iberoamericana de Automática e Informática industrial 9 (2012) 406–416 www.elsevier.es/RIAI

Módulo Distribuido de Subasta Java sobre CORBA de Tiempo Real

P. Basanta-Val*, M. García-Valls, H. Morillas-Rodrigo y J. Cano-Romero

Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, Avda. Universidad nº 30, 28911, (Leganés) Madrid, España.

Resumen

El uso de infraestructuras middleware que permitan intercomunicar diferentes nodos de un sistema puede aliviar aspectos importantes relacionados con el coste de desarrollo y mantenimiento de dichos sistemas en entornos flexibles. En este contexto el artículo presenta una arquitectura desarrollada dentro del contexto de los sistemas de subasta distribuidos que tienen ciertos requisitos de tiempo real que han de ser satisfechos. Dicha arquitectura implementa un sistema de puja doble (CDA) para un sistema de tiempo real industriales. El artículo presenta tanto la arquitectura del sistema con sus diferentes actores así como una evaluación en una plataforma real utilizando middleware de tiempo real ya existente.

Palabras Clave:
Sistema de Subasta, Informática Industrial, Middleware, Tiempo Real, RT-CORBA, Real-time Java

1. Introducción

El advenimiento de Internet a principios de los noventa dio lugar a nuevos modelos de negocio que antes no eran posibles. Las barreras físicas y temporales asociadas a los procesos de negocio habituales se vinieron abajo, y uno de los ejemplos más característicos de esta nueva economía fueron los portales de subastas online (como por ejemplo eBay.com). El modelo es sencillo pero novedoso: el portal se encarga de poner en contacto a particulares y empresas y de gestionar las subastas. Son dichos clientes (particulares y empresas) los que poseen los bienes y servicios. El sitio web sólo se encarga de la gestión, no realiza ninguna acción física sobre el intercambio de productos: un modelo de negocio puramente virtual.

La mayor parte de los subastadores que existen funcionan de forma electrónica cerrando operaciones automáticamente, cuando se dan las condiciones necesarias para ello, como ocurre cuando el comprador y el vendedor ofrecen el mismo precio (Cliff, 2003). Otra característica que tienen es requerir un cierto tiempo respuesta máximo en el proceso de subasta desde el momento en el que se da la orden hasta el momento en que ésta se realiza. Existe también la posibilidad de que dichos sistemas ofrezcan diferentes tipos de servicio con diferentes calidades de servicio según la cantidad pagada al proveedor del servicio.

El tema de la creación de subastadores ha suscitado interés investigador tanto desde el punto de vista arquitectónico, en relación con las diferentes arquitecturas existentes, como a la hora de soportar dichos sistemas. Así, la literatura existente cubre tanto aspectos tecnológicos como puede ser una implementación (Aleksy, Korthaus, Schader, 2001) mediante middleware CORBA (Siegel, 1999) como otros más teóricos relacionados con los modelos formales que implementan los diferentes algoritmos de subasta existentes en las plataformas actuales (Vytelingum, 2006), (Klemperer, 1999).

Dichos subastadores electrónicos, bien conocidos en el mundo Internet y con arquitecturas y estrategias bien definidas, tienen también cabida en el sistema informático de una planta industrial (Bussmann and Schild, 2000). La subasta se puede incorporar como un módulo avanzado relacionado con la gestión de recursos y facturación realizada en una planta (Figura 1). De esta manera se puede realizar por un proceso de subasta tanto la provisión de bienes como la propia venta de los productos producidos de forma puramente electrónica. La integración de este módulo de subasta permite una gestión más ágil y eficiente de la producción, que puede ser regulada, a partir de flujos de oferta y demanda surgidos del subastador. Tanto proveedores como demandantes pueden tener un acceso ágil y actualizado a las diferentes ofertas existentes en el entorno industrial. La subasta también se puede aplicar sobre el propio sistema de producción que negocia los

* Autor de correspondencia.
Correos electrónicos: pbasanta@it.uc3m.es (Pablo Basanta Val), mvalls@it.uc3m.es (Marisol García Valls), hector28933@gmail.com (Héctor Morillas Rodrigo) y jcano@it.uc3m.es (Julio Ángel Cano Romero)
URL: www.it.uc3m.es/drequiem/

http://dx.doi.org/10.1016/j.riai.2012.09.009

diferentes elementos de la línea de producción. La aplicación a un entorno industrial donde existen plazos que han de ser cumplidos fuerza a que los tiempos del subastador estén en el rango de los pocos milisegundos a los pocos segundos (dependiendo del elemento subastado).

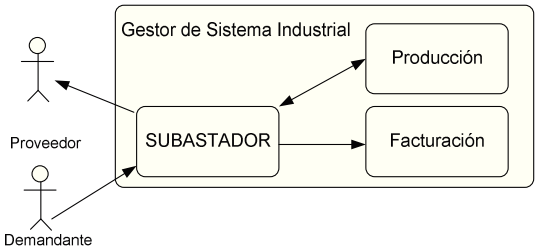

Figura 1: Integración de la tecnología de subasta dentro de un software de sistema industrial de orden superior

En este marco de la contribución realizada por este artículo es la de proveer un subastador sencillo e integrable dentro del entorno industrial distribuido. La contribución se centra en los aspectos de diseño e implementación del subastador. En el bajo nivel está tanto en la elección un algoritmo de subasta como en su posterior implementación en una plataforma de tiempo real. El evaluarlo va a permitir decidir sobre el rendimiento esperable de la plataforma escogida en la implementación.

En la implementación se escoge la tecnología RT-CORBA (Schmidt and Kuhns, 2000) y en especial el middleware Java RTZen (Raman *et al.*, 2005) para realizar dicha arquitectura. La ventaja que tiene adherirse al estándar RT-CORBA es la de disfrutar de una serie de mecanismos genéricos que permiten controlar de forma estándar la calidad de servicio. Por último, utilizar Java permite reducir (Phipps, 1999) los tiempos de desarrollo de los algoritmos de subasta.

Los resultados de rendimiento descritos en este artículo (sobre RTZen) son también interesantes para la comunidad investigadora Java de tiempo real (Basanta-Val and Anderson, 2012) pues les permite ver el rendimiento que RTZen puede ofrecer a aplicaciones distribuidas. Las técnicas utilizadas son también de utilidad para mejorar arquitecturas de subasta ya descritas con nuevas características de tiempo real. Una de ellas es la ya mencionada arquitectura para la realización de subastas en CORBA (Aleksy, Korthaus, Schader, 2001). La incorporación de las técnicas descritas en este artículo dentro de dicha arquitectura permitirá un mayor control sobre los recursos empleados en las comunicaciones extremo a extremo, y puede mejorar sus tiempos de respuesta.

El resto del artículo se organiza tal y como sigue. La Sección 2 introduce el modelo de subasta electrónica de forma general. Dicho modelo será la base sobre la cual se articulará la Sección 3 que aborda el diseño de alto nivel del subastador. Dicho diseño se complementa en la Sección 4 con el modelo middleware que se ha elegido y la estructura de la base de datos empleada. La Sección 5 se refiere a su despliegue en un entorno real y los resultados obtenidos de su evaluación. La Sección 6 estudia la aplicación de otros modelos middleware alternativos al modelo de subasta propuesto. Finalmente, la Sección 7 concluye y expone nuestro trabajo en curso referente a dicho sistema de subasta.

2. La subasta

En un sentido meramente abstracto una subasta es un mecanismo para la interacción comercial (Klemperer, 1999; Cliff, 2003). La clave se encuentra en el matiz que introduce la palabra interacción, ya que conlleva que las partes implicadas tienen que comunicarse de alguna forma para llegar a un acuerdo comercial. Esta interacción se especifica mediante tres aspectos distintos como son: (1) las reglas de intercambio entre las partes, las (2) estrategias de los participantes y (3) las reglas de finalización de la subasta. Las reglas de intercambio se tienen que acordar antes del inicio de la subasta, y a lo largo de ésta, los participantes van modificando su estrategia, y finalmente las reglas de finalización determinan cuándo termina la subasta.

Una de las tipologías más comunes a la hora de clasificar los tipos de subastas es según qué participantes en la subasta son los autorizados a modificar sus ofertas apareciendo la *subasta ascendente* o inglesa, la *descendente* o holandesa y la *subasta continua*. En la subasta inglesa, los compradores modifican sus ofertas al alza por un objeto hasta que el resto de compradores se retira. En la subasta holandesa, el vendedor va reduciendo el precio inicial pedido hasta que un comprador detiene la puja y acepta la oferta. Y por último en la continua doble, los vendedores y compradores varían el valor de su oferta o puja, respectivamente, hasta que ambas convergen y se realiza la venta. Típicamente ésta última se usa en mercados financieros o de materias primas. En el mundo anglosajón se conoce como subastas CDA (Continuous Double Auction) (Vytelingum, 2006).

Las subastas CDA son las que van a centrar el interés del modelo desarrollado, ya que son los clientes del sistema (vendedores y compradores) los que definen el comportamiento del sistema en sí, mientras que el sistema central que administra las transacciones (ofertas aceptadas), se limita a registrar cuándo una oferta de compra y una oferta de venta cuadran y se cierra un trato, y ese es el tipo de comportamiento que se está buscando en este trabajo.

Para comprender mejor cómo funciona un sistema de subasta CDA, se puede ver la siguiente secuencia de ofertas y pujas (Tabla 1):

Tabla 1: Pasos dados en una subasta CDA

Paso	Oferta vendedor	Puja Comprador
1	1200 €	500 €
2	1200 €	800 €
3	1000 €	900 €
4	950 €	950 €

En el último paso, después de que el vendedor fuera rebajando la cantidad que pedía y el comprador fuera aumentando el precio que estaba dispuesto a pagar, se llegó a un momento de convergencia y se cerró la transacción.

2.1 Marco Histórico de los Subastadores

Los primeros sistemas de subastas en Internet pueden encontrarse en sistemas por correo electrónico y grupos de noticias alrededor de 1988 (Cliff, 2003). Con el desarrollo de Internet a principios de los 90 empezaron a surgir formas más sofisticadas de subastas online. Al principio simplemente se publicaban los listados de productos con información multimedia añadida en una página web y las subastas se seguían realizando por correo electrónico. El siguiente paso fue el uso de formularios para recolectar las ofertas.

Ebay surgió en 1995 y fue uno de los primeros y más conocidos sitios de subastas en Internet, y poco tiempo después surgieron competidores como Ubid, Onsale y Z-auctions. Se pudo ver por aquella época como se empezaba a dividir el mercado de las subastas en línea en función del tipo de participantes.

En el terreno de las subastas entre empresas, una empresa como Fastparts.com ya ofrecía servicios de subastas basados en boletines en el año 1991 y en el año 1996 ya creó un sitio web. A la vez que se incrementaba la popularidad de los sitios web que gestionaban subastas, surgieron compañías de software que ofrecían productos para que los clientes montaran sus propios servicios de subastas. Inicialmente fueron compañías independientes como Opensite, Trading Dynamics, Moai o los ya reseñados FreeMarkets, que surgieron a mediados de los 90.

Con el paso del tiempo las grandes compañías de software para empresas empezaron a adquirir algunas de las empresas independientes mencionadas y otras a crear sus propias herramientas de subastas. El primer camino fue tomado por ejemplo por Siebel (ahora parte de Oracle) cuando adquirió Opensite y Ariba adquirió Trading Dynamics. En lo que respeta a IBM, en 1998 ya empezó a trabajar en su propia herramienta de subastas.

Por otro lado, también hay que reseñar los sistemas online de subastas construidos sobre redes propias al margen de la WWW, típicamente como una extensión de los sistemas bursátiles, como fue por ejemplo la creación por parte del mercado de valores americano NASDAQ del sistema Small Order Execution System.

2.2 Estrategias de Subasta CDA

A la hora de modelar el comportamiento de los compradores y los vendedores hay que tener en cuenta dicho flujo continuo de información y realizar suposiciones que permitan tomar decisiones en un momento determinado en función de la memoria del proceso y de ciertos límites impuestos de antemano al comportamiento del agente (i.e., una entidad con capacidad de participación en la subasta).

Hay tres principios básicos que definen el comportamiento de un agente. (1) Un agente necesita información sobre sí mismo y sobre su entorno para tomar decisiones informadas. (2) Un agente rara vez tiene todos los recursos necesarios para realizar un análisis completo de su información. Y por último, debido a la capacidad de análisis limitada, (3) un agente necesita utilizar técnicas de tanteo y aproximación para tomar sus decisiones.

A partir de estos preliminares, en la literatura sobre el comportamiento de los agentes en subastas CDA, se plantean dos aproximaciones básicas. La primera es el agente de inteligencia mínima ZIP (Zero Intelligence Plus) y la segunda es el agente GD (Gjerstad-Dickhaut) (Vytelingum, 2006).

Dentro de este contexto, el agente ZIP (Zero Intelligence Plus) parte del concepto de grito. Un vendedor o un comprador, al entrar en el mercado lanza una oferta que considera muy beneficiosa para él, pero que puede estar muy lejos de lo que se considera del punto de equilibrio entre comprador y vendedor. De esta forma se realiza un primer tanteo del mercado. Si hay alguien dispuesto a aceptar la oferta, el beneficio es muy alto. En caso contrario tendrá que ir reduciendo el beneficio de la oferta.

Por otro lado, el modelo GD permite, añadiendo memoria a largo plazo, un análisis de la tendencia del mercado. Se almacena un historial de ofertas y transacciones, y se clasifican en función de su aceptación. Para precio p el algoritmo almacena el número de ofertas aceptadas con precio mayor que p; el total de ofertas realizadas con un precio mayor que p y el total de ofertas denegadas con un precio menor que p.

Ambos modelos de negociación (GD y ZIP) necesitan cierta infraestructura mínima que les permita realizar compras y ventas. Dicha infraestructura básica es la que se ha desarrollado como base nuclear del sistema de subastas que pasa a examinarse a continuación.

3. Modelo Desarrollado

El sistema es un sistema distribuido de subasta de productos marcados con etiquetas. Toda la gestión gira alrededor de dichas etiquetas, las cuales identifican a todos los productos que se puedan negociar. Por tanto, su objetivo es permitir a un conjunto de compradores y vendedores realizar pujas sobre una serie de productos, en un entorno distribuido; esto es, con acceso remoto al proceso de compra y venta.

La modalidad de puja implementada es la puja doble (CDA) en la que los compradores y los vendedores van modificando sus ofertas en función de la información de la que disponen, que en la arquitectura desarrollada es transparente en el sentido de que sabrán el precio mínimo que ofrecen los vendedores y el precio máximo que ofrecen los compradores.

Además, se realizará un sistema lo más neutral posible desde el punto de vista de la descripción cualitativa de los productos ofrecidos. La descripción de identidad de los productos tendrá una forma genérica adaptable a cualquier tipo de mercado, para lo cual dicha identidad se reducirá a una etiqueta única y una posible descripción para mejor identificación del producto.

3.1 Actores Involucrados

Subiendo a una visión más global de la descripción realizada en el apartado anterior, se puede separar en dos grupos las funcionalidades necesarias para realizar subastas:

- *Acceso a una subasta*, que permite a compradores y vendedores empezar a realizar ofertas.
- *Proceso de negociación* en una subasta, que culmina con los compradores y vendedores llegando a un acuerdo final tras una serie de pujas.

El objetivo tras esta división es llegar a un modelo de objetos que permita describir a los elementos involucrados en una subasta: *compradores y vendedores* por un lado, y los *productos* que estos negocian por otro lado (Figura 2).

- En primer lugar se necesita identificar en una entidad lógica a los compradores y vendedores, para lo cual se parte del principio de que alguien que puede comprar también puede vender. De esta forma se puede encapsular la identidad de los compradores y vendedores en un objeto único, el `comerciante`. Además, se podrá ejecutar las operaciones necesarias para solicitar el acceso a las subastas y para realizar ofertas de compra y venta.
- A continuación, la identificación de los productos disponibles y la gestión de las pujas que se realizan sobre ellos se encapsulará en una única entidad, el objeto etiqueta, dado que cada producto se identifica de forma única con una `etiqueta`.
- Finalmente, es necesario un tercer elemento que valide la identidad de los objetos comerciante y etiqueta, además de

poner ambos conjuntos de elementos en contacto entre sí, para que los procesos de subasta puedan comenzar y finalizar. Esta entidad se denomina `intermediario`.

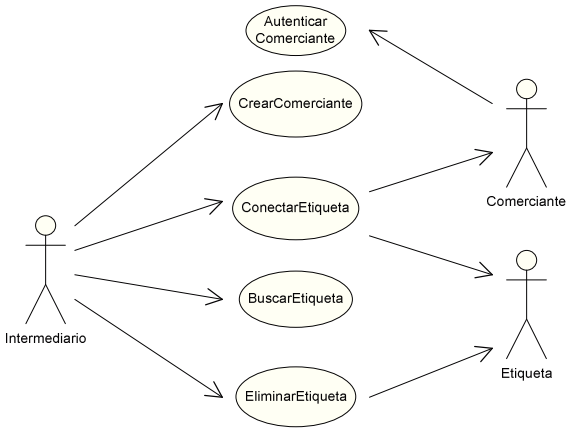

Figura 2: Casos de uso relacionados con el acceso a una subasta

3.2 Diseño Relacionado con la Inicialización del Sistema de Subasta

Los objetos remotos definidos en el apartado anterior interaccionan entre sí para controlar el acceso de los clientes al sistema de subastas. A través del objeto intermediario se realiza la autenticación de los clientes y el acceso de éstos a las diferentes etiquetas por las que pueden pujar. En segundo plano, a través del almacenamiento persistente que ofrece la base de datos, se garantiza la coordinación entre el proceso de autenticación de clientes y la conexión a una de las etiquetas, así como la desconexión. Como se puede ver en la Figura 2, el intermediario tiene cuatro casos de uso, uno para crear un objeto comerciante (*autenticándolo*) y otros tres para manejar el acceso a las etiquetas: *búsqueda*, *conexión* y *desconexión*.

El proceso completo de inicialización comprende los pasos de creación de referencia del intermediario y de referencia del comerciante tal y como se muestra en la Figura 3 y la Figura 4. En estos diagramas se ve el ciclo completo de la gestión de las etiquetas que va desde la búsqueda de la etiqueta en la base de datos hasta la conexión con la etiqueta. Dicha conexión es necesaria para poder realizar pujas, ofertas y consultas sobre la etiqueta, la cual simboliza el ítem por el que se quiere pujar. Finalmente el cliente también puede eliminar la etiqueta lo que anula su utilización en el sistema de subasta.

En estos hay que tener en cuenta una serie de restricciones de bajo nivel relativas a lo que es el proceso de autenticación:

- Si la autenticación es exitosa, se creará un objeto de servidor representando la identidad del comerciante, esto es, el cliente que se ha conectado al sistema.
- A la hora de buscar etiquetas no se crean objetos de servidor, sino que simplemente se realiza una búsqueda en base de datos.
- La conexión con una etiqueta supone buscar el objeto de servidor que lo representa y si no existe crear dicho objeto. Tras ello, se introducirá la identidad del comerciante en la lista de clientes autorizados a realizar pujas sobre esa etiqueta.
- La desconexión elimina la asociación entre el objeto comerciante el objeto etiqueta, eliminando ésta si no existen más comerciantes conectados.

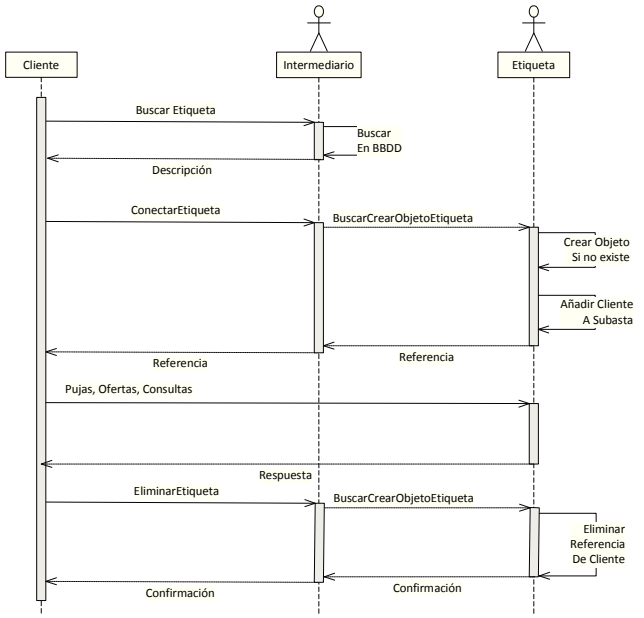

Figura 3: Diagrama de operaciones de acceso al sistema de subastas

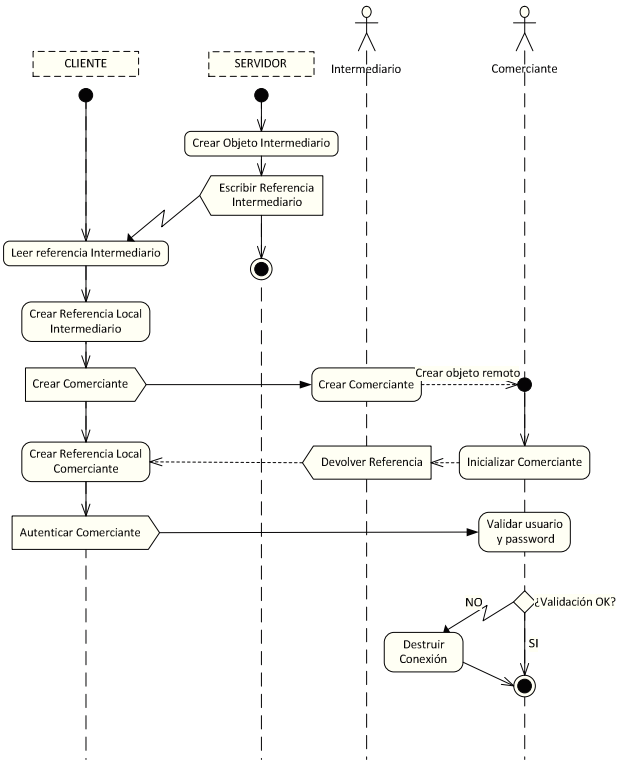

Figura 4: Detalle técnico de la inicialización del cliente y la creación del objeto intermediario

3.3 Diseño Relacionado con el Proceso de Subasta

Una vez inicializada la conexión al sistema, el cliente puede participar en subastas, siguiendo la secuencia de operaciones de subasta que son las que permiten que los diferentes actores puedan llegar a acuerdos sobre sus transacciones basadas en etiquetas.

Dichas operaciones necesarias para participar en una subasta se reducen a pujas y consultas de ventas y de compras, tal y como detalla la Figura 5. En ella se ve que hay dos pujas: una asociada a la venta y otra asociada a la compra. Además completando estas dos operaciones existen otras dos que permiten realizar consultas tanto de compra como de venta.

Es de resaltar que durante estas operaciones se utiliza el concepto etiqueta que centraliza las operaciones sobre un producto. Las cuatro operaciones requieren el identificador de etiqueta para poder operar. Por último la identidad de los comerciantes se persiste a lo largo de la sesión de puja, de forma también subyacente a los casos de uso explicitados.

El proceso de subasta, en sí, se resume en tres capacidades, desde el punto de vista de los comerciantes, que se pueden enunciar como reglas de negocio:

- Los comerciantes pueden consultar la oferta más baja del momento y la puja más alta del momento también, consiguiendo información del estado de la subasta.
- Un comerciante o vendedor puede realizar una oferta para el precio que decida libremente. En caso de haber un comprador para ese precio, ésta se hará efectiva.
- Un comerciante comprador puede realizar una puja de compra por el precio que decida libremente, y en caso de haber un vendedor para el precio dado, ésta se hará efectiva.

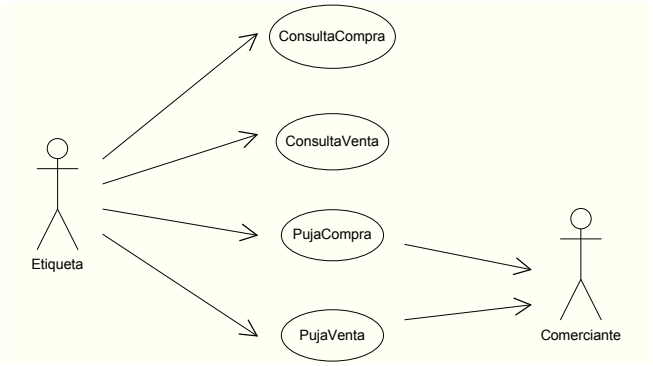

Figura 5: Casos de uso relacionados con el proceso de subasta

De esta forma, los comerciantes tendrán una información transparente y en tiempo real del estado del mercado y podrán tomar decisiones de compra y venta bajo la responsabilidad de que al hacer una oferta o una puja si efectivamente alguien la acepta, se hará efectiva. Lo que se corresponde con el modelo de puja doble en el que se basa el sistema de subasta propuesto.

3.4 Algoritmos de Subasta para la Puja Doble (CDA)

Los algoritmos que se ejecutan dentro de una etiqueta para valorar las distintas ofertas de compra y venta que hacen los comerciantes, son similares y se diferencian principalmente en el umbral de decisión. Para una oferta de compra, la transacción se efectúa si existe una oferta de venta inferior y, a la inversa, una oferta de venta se hace efectiva si hay un comprador con una oferta superior.

Asimismo, se contempla un tiempo límite por el cual una oferta se sostiene, devolviendo la expiración del tiempo como una excepción al comerciante que ha hecho la oferta.

La Figura 6 nos muestra el proceso de puja de compra en más detalle. En primer lugar el algoritmo comprueba si el usuario está registrado o no, siendo expulsado de la puja si no está autenticado adecuadamente. Se comprueba que la puja es un valor positivo, mayor que cero, si no lo es la puja también se cancela pasando esta vez por un estado de puja insuficiente. Pasadas estas dos comprobaciones se ve si hay o no otras ofertas en el mercado. En caso de haberlas, se comprueba si la oferta es mayor o menor que la del sistema. Si es menor, no se podrá cerrar el trato y el sistema volverá a salir por puja insuficiente. Si la oferta es mayor que la del sistema se establece una oferta de compra que significará que el comprador podrá adquirir (si el vendedor así lo desea) el bien ofertado.

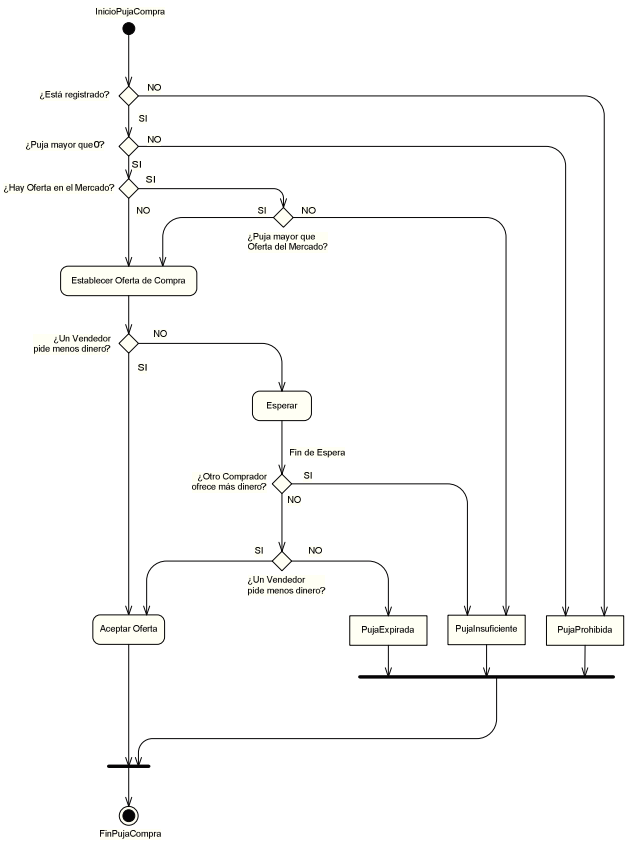

Figura 6: Algoritmo de procesado de una puja de compra

Una vez establecida la oferta de compra queda por determinar si el sistema aceptará dicha oferta o no. Para ello hay que chequear la cantidad que pide el vendedor; si ésta es menor que la que se oferta, se acepta la puja, lo que hace que se cierre el trato. En el caso de que el vendedor pida más dinero el cliente esperará un cambio en el sistema. Los eventos que pueden ocurrir están relacionados con la llegada de una puja mayor al sistema y que el vendedor modifique su precio. En el primer caso si hay otro comprador que ofrezca más, éste expulsará al comprador del sistema. En el caso de que el vendedor cambie su oferta y pida menos dinero del ofertado, el algoritmo cerrará el trato. Por último, puede pasar que venza el plazo máximo asignado para la

realización de la operación de subasta, caso en el que el sistema irá al estado de puja expirada.

Completando el algoritmo de la puja de compras, está la puja de venta (Figura 7). Dicho algoritmo es ejecutado cuando un comerciante desea vender un producto. Al igual que en el caso anterior, el algoritmo comienza expulsando a un usuario que no se encuentre registrado en el sistema o intente vender un artículo por un valor negativo de euros. Tras ello se comprueba si hay alguna oferta en el mercado para dicho producto, si es así se comprueba si se puede realizar la operación. Para ello es necesario que el precio que pide el vendedor sea menor que el de la oferta del mercado. En dicho caso, se establece la operación de venta que será materializada cuando un vendedor ofrezca más dinero.

Al igual que en el caso anterior, en el caso de que no haya vendedores que ofrezcan lo que pide el cliente, se esperará un cambio en el sistema. Dicho cambio vendrá dado por la aparición de un nuevo comprador que podrá provocar una situación de puja insuficiente o por la aceptación de una oferta. También podrá venir derivado del vencimiento del temporizador del sistema, lo que hará que la puja expire.

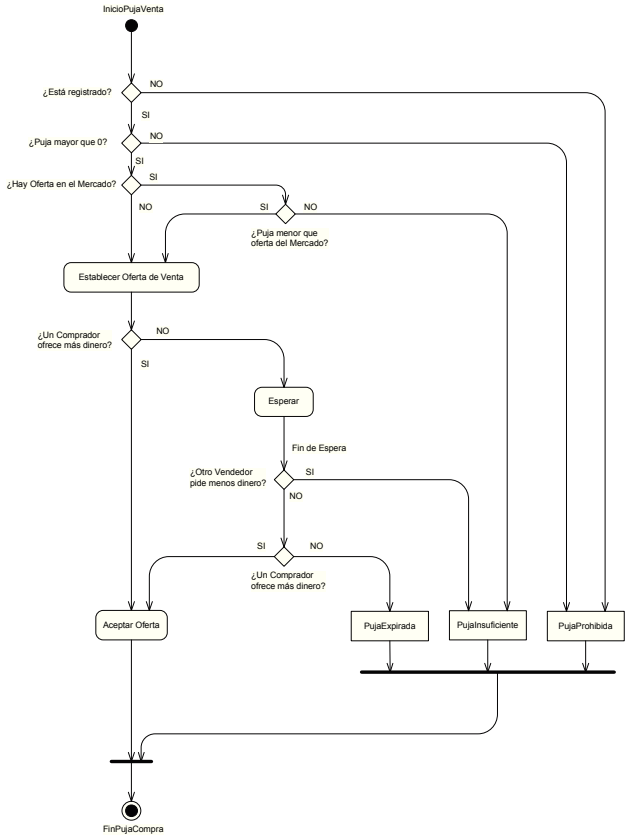

Figura 7: Diagrama de actividad del algoritmo de procesado de una puja de venta

4. Detalles específicos de la implementación

La sección anterior se ha referido al modelo de subasta en si y aporta un modelo que puede ser desplegado en diferentes sistemas finales. Esta sección particulariza dicho modelo y lo ciñe a un middleware en concreto: RTZen.

RTZen (Raman *et al.*, 2005) es un middleware Java de tiempo real que soporta el desarrollo de aplicaciones compatibles con RT-CORBA (Schmidt and Kuhns, 2000). Su enfoque es altamente conservador y se enfoca en la optimización del middleware, dejando de lado aspectos más relacionados con el impacto en las interfaces remotas (como podría ser la eliminación del recolector de basura de la invocación remota mediante nuevas modelos de programación). Desde el punto de vista de este sistema dos son las características más importantes que posee. La primera es que ofrece el mecanismo de calidad de servicio definido en RT-CORBA y segunda es que su diseño facilita el reemplazo de una máquina virtual y la utilización de otra diferente.

4.1 Modelo de Prioridades Adoptado por el Sistema de Subastas

En el modelo propuesto para las subastas cada operación de compra-venta tendrá asociada una prioridad RT-CORBA. Dicha prioridad servirá para establecer la calidad de servicio que tendrá dicha operación. De esta manera operaciones de compra con una mayor prioridad se ejecutarán siempre antes que las de baja prioridad, siendo este el corazón del mecanismo de calidad de servicio soportado. Este mecanismo también resulta básico si se quiere obtener tiempos de respuesta acotados para las aplicaciones pues la prioridad junto a los plazos de las diferentes aplicaciones puede ser utilizada para realizar análisis de planificación de sistema.

El estándar RT-CORBA define dos tipos de modelos de prioridades(Schmidt and Kuhns, 2000): *definidas por cliente* y *definidas por el servidor*. Ambos modelos tienen sus ventajas y desventajas. Las prioridades controladas por el servidor permiten ajustar los recursos globales y repartirlos entre los distintos clientes conectados. Por otro lado, si los clientes pueden controlar su prioridad pueden ajustar los recursos del servidor en función de sus propios recursos: si un cliente tiene problemas para procesar la información que recibe puede ralentizar sus invocaciones para ajustar la carga de extremo a extremo. Ambos modelos pueden ser utilizados al definir el modelo de calidad de servicio de una subasta sin que haya un gran impacto en el modelo de negociación.

Desde una perspectiva práctica (esto es, la implementación real) se han utilizado prioridades propagadas por el cliente. El principal motivo es que RTZen no soporta de forma estable las prioridades controladas por el servidor pero soporta las propagadas por el cliente correctamente. Concretamente se define un modelo de prioridades mediante pistas (*lanes*), esto es, franjas de prioridades que permiten agrupar los hilos que ejecutan las prioridades en conjuntos. Lo que da por resultado el que aparezcan tres grupos de prioridades: *alta*, *baja* y *media*.

4.2 Interfaz IDL Encargada del Proceso de Subasta

En CORBA la interacción entre entidades se modela en un lenguaje llamado IDL. La Figura 8 muestra en lenguaje IDL las interfaces de los objetos CORBA creados para la implementación del sistema de subasta y los tipos de datos y excepciones asociados a dichos mecanismos. Estos métodos se agrupan en tres interfaces, una por cada actor del proceso de subasta, y definen los tipos de datos utilizado por los principales entes de la subasta. De todos ellos, los más importantes son **`pujaVenta`** y **`pujaCompr`**, las cuales generan excepciones (puja prohibida, puja insuficiente o puja expirada)

en el caso de que no se puedan cruzar las operaciones descritas.

```
    # Excepciones
     exception AutenticacionException{};
     exception ObjetoNoExisteException{};
     exception PujaInsuficiente{};
     exception PujaProhibida{};
     exception PujaExpirada{};

     # Tipos de Datos
     typedef sequence<string> ListaString;
     typedef sequence<long> ListaInt;

     # Interfaces
     interface Intermediario
      string crearComerciante();
      ListaString buscarEtiqueta(
                  in ListaString palabrasClave)
                   raises(ObjetoNoExisteException);
      string conectarEtiqueta(
                  in string nombreEtiqueta,in string
                  nombreComerciante,
                  in string id_sesion)
                  raises(ObjetoNoExisteException);
      void eliminarEtiqueta(in string nombreEtiqueta,
                  in string id_sesion)
                  raises(ObjetoNoExisteException);

     interface Comerciante
      string autenticar(in string nombre,in string password)
                  raises( AutenticacionException );

     interface Etiqueta
      long consultaCompra(in string id_sesion);
      long consultaVenta(in string id_sesion);
      string pujaCompra(in long in, string id_sesion,in long
time_out)
                  raises (
PujaProhibida,PujaInsuficiente,PujaExpirada );
      string pujaVenta(in long precio,in string id_sesion,in
long time_out)
                  raises (
PujaProhibida,PujaInsuficiente,PujaExpirada );
```

Figura 8: Interfaces IDL

4.3 Gestión de la Persistencia y Modelo de la Base de Datos

La información persistente, que se necesita para mantener la arquitectura, se reduce a mantener la información relativa a los comerciantes, la información que describe una etiqueta y el registro de las transacciones realizadas, tal y como se muestra en la Figura 9. Hay por tanto tres tablas: una para almacenar transacciones que almacena el valor de los intervinientes en la operación; otra para identificar a los comerciantes con su clave; y una tercera que describe una etiqueta con un nombre.

Dichas tablas se relacionan mediante claves foráneas que permiten a la transacción mantener una referencia con el comerciante y con la etiqueta.

4.4 Despliegue de la Aplicación de Subasta

Además de las interfaces de los objetos CORBA y el modelo de datos persistente, también es necesario tener una idea de la localización física de los distintos elementos de la aplicación, que se exponen en el diagrama de despliegue de la Figura 10. Este tipo de despliegue coincide con los utilizados en aplicaciones empresariales donde la lógica del cliente es ligera y se reduce a temas de presentación y donde casi toda la lógica de negocio está en el servidor, lo que minimiza los costes de desarrollo y mantenimiento en sistemas abiertos con diferentes tipos de clientes.

Como se puede observar, se tiene un modelo cliente-servidor, que se comunica mediante el protocolo IIOP, que posibilita la propagación invocaciones de objetos remotos CORBA a través de TCP/IP.

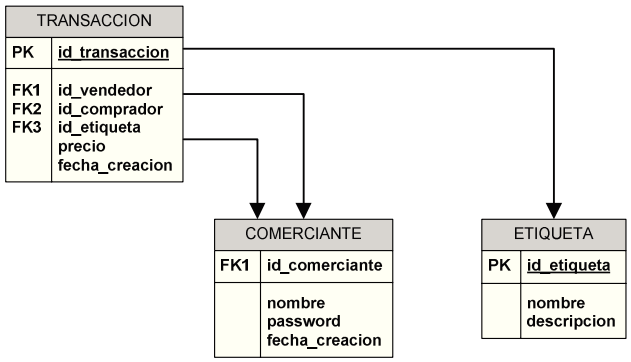

Figura 9: Modelo de información persistente: estructura de la base de datos

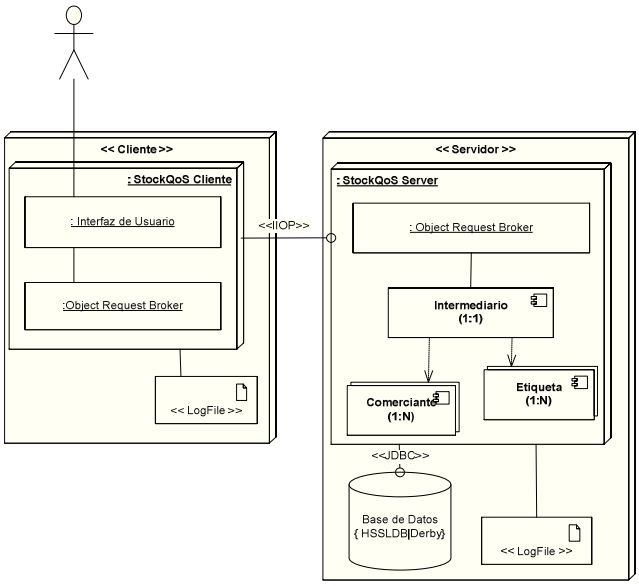

Figura 10: Despliegue típico de la arquitectura de subastas

Además se pueden ver los archivos de log que van registrando la actividad del servidor, el cliente y la base de datos. Dicha base de datos almacena la información persistente del servidor, en sus dos implementaciones posibles (HSQLDB y Apache Derby) comunicada con el servidor mediante un driver JDBC con el servidor.

Por último, destacar la interfaz con el usuario en el lado del cliente, la cual permite que éste interaccione de forma adecuada con el sistema mediante un bróker local que sirve de puente entre el lado cliente y el lado servidor.

5. Evaluación

A la hora de evaluar el rendimiento del sistema de subastas propuesto hay un aspecto fundamental a considerar: el

rendimiento de las invocaciones cliente-servidor, definidas éstas como tiempo que transcurre desde que se envía la petición hasta que se recibe la respuesta. Pero también es interesante obtener la función de distribución de probabilidades que muestre cómo se reparten dichos valores en los casos peores/mejores y promedio.

Otro aspecto importante a considerar en la evaluación empírica consiste en la identificación, en los escenarios de pruebas, de aquellas variables que pueden afectar a los resultados y que se pueden modificar de forma controlada.

Los experimentos realizados tienen por tanto como objetivo comprobar el rendimiento de la aplicación en función de los siguientes parámetros:

- Retardo introducido por la red de comunicaciones entre los clientes y servidores.
- Efecto del paralelismo y la ausencia de éste en la ejecución de la aplicación.
- Diferencias de rendimiento entre las distintas máquinas virtuales de Java disponibles en el mercado que resultan compatibles con RTZen.

Sobre una estructura básica donde hay un cliente y un servidor se instala cliente que realiza pujas (compra/venta) en una base de datos empotrada en memoria, donde existe un número bajo de pujas. Ambos procesos (compra/venta) ofrecen rendimientos similares (en tiempo de respuesta). Sobre esta infraestructura básica se varían tres parámetros:

- La prioridad a la que ejecuta cada cliente (que puede ser 1000 o 2000) y el número de clientes. El objetivo es ver que el comportamiento obtenido se corresponde con el comportamiento expulsivo típico de un sistema de tiempo real.
- La topología del sistema que puede estar desplegado en un único procesador o en un sistema distribuido con una red.
- La máquina virtual utilizada durante la ejecución.

La generación de peticiones se hace con trazas generadas de forma automática mediante scripts.

5.1 Pila de Componentes Software y Red Utilizada para la Comunicación

La aplicación se instalará sobre una pila de componentes, cuya configuración definirá nuestro escenario de pruebas, tal y como se puede ver en la Figura 11. Dicha pila tiene la máquina sobre la que se instala sistema Linux (el recomendado por RTZen) y una máquina virtual Java con soporte parcial (ofrecido por RTZen) a RTSJ (Bollella G. et al., 2001). Encima de esta máquina virtual está RTZen, elemento clave sobre el que se apoya el sistema de subastas.

En esta arquitectura uno de los elementos con los que se ha jugado son las diferentes máquinas virtuales de Java que dan soporte a RTZen. Las utilizadas en los experimentos se detallan en la Tabla 2. Las tres máquinas: la de IBM, Sun y Oracle (JRMC), funcionan correctamente cuando son utilizadas por RTZen para el despliegue de aplicaciones.

Teniendo en cuenta estas máquinas virtuales, se define un escenario distribuido con diferentes máquinas virtuales. En este caso la infraestructura de red utilizada es una red Ethernet (Figura 12). El sistema se encuentra virtualizado, lo que permite asignar una tarea a un único núcleo para los equipos que usen un sistema multicore. Las máquinas utilizadas son Intel para los Xeon 7300 (a 4 GHz y con 8 MB de cache) y la red utilizada de 1 Gigabit.

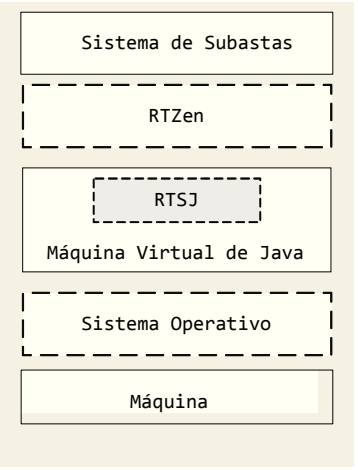

Figura 11: Pila software de despliegue de la arquitectura de subastas

Tabla 2: Máquinas Virtuales Utilizadas

Fabricante	Runtime Environment	Virtual Machine
Sun	Java(TM) SE Runtime Environment build 1.6.0_17-b04	Java HotSpot VM build 14.3-b01
Oracle	Java(TM) SE Runtime Environment build 1.6.0_11-b03	BEA JRockit build R27.6.3-40 (JRMC)
IBM	Java(TM) SE Runtime Environment build pxi3260sr5-20090529_04(SR5)	IBM J9 VM build 2.4 J2RE 1.6.0

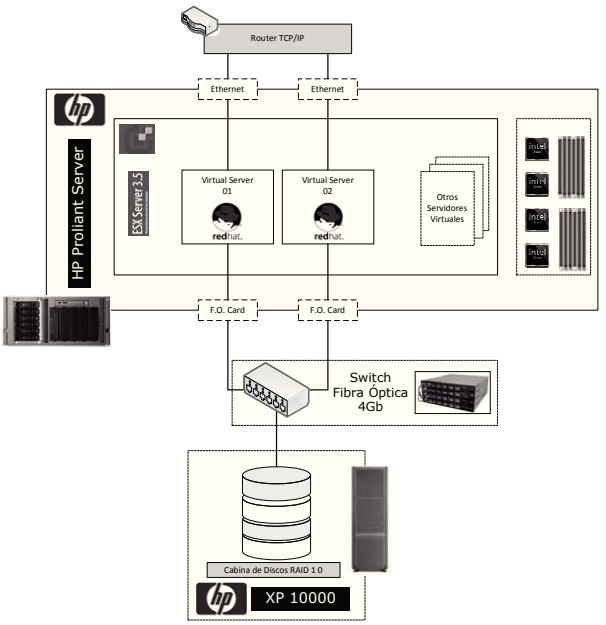

Figura 12: Escenario de pruebas utilizado

Para aislar el comportamiento de estas dos variables frente a los retardos introducidos por la lectura y escritura en disco, se utilizó un almacenamiento en una cabina de discos SAN, que utiliza fibra óptica como medio de transmisión y tiene una caché de 32 GB que amortigua por completo los posibles retardos por

escritura en disco al encolar todas las peticiones de escritura en memoria RAM.

5.2 Tiempo de Respuesta Extremo a Extremo

Aunque pueda parecer que la elección de la maquina virtual Java puede tener un impacto bajo o medio en el rendimiento final, esto no es así. El elegir una u otra infraestructura tiene un impacto decisivo en la operación a realizar que se puede elevar hasta en un 50 % (Figura 13) el tiempo de respuesta del sistema.

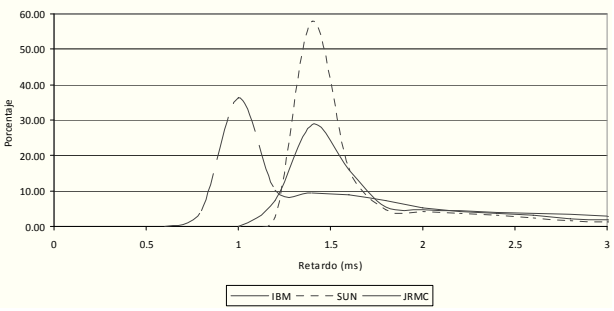

Figura 13: Dos clientes conectados a un servidor a través de una red Ethernet. Patrón de resultados.

Las máquinas virtuales de IBM y de Sun ofrecen un rendimiento peor que JRockit. Claramente, JRockit (JRMC) es bastante mejor y esta ventaja en rendimiento puede ser debida a que dicha máquina virtual reimplementa parte de las comunicaciones y de los mecanismos de serialización para que sean más eficientes. Es esta reimplementación la que reduce tan drásticamente los tiempos de respuesta del sistema.

Los resultados obtenidos (Figura 13) nos muestran que la infraestructura (RTZen) ofrece unos tiempos de respuesta que se encuentran en el rango de los milisegundos (0,9 ms para Jrockit, 1.3 ms para Sun JRTS y 1.4 ms para IBM J9). Desde un punto de vista práctico, este es el coste máximo que tendría el proceso de subasta y se corresponde en su mayor parte con la transferencia en la red y acceso a la base de datos en el lado del servidor.

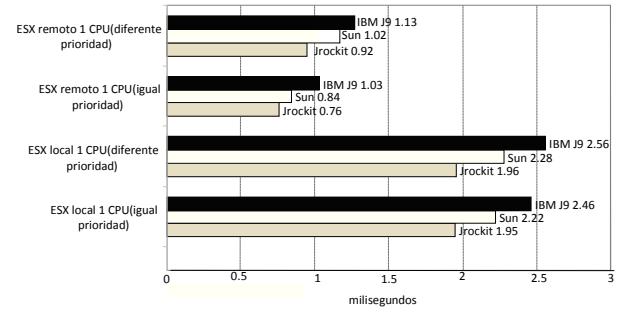

Figura 14: Resultados globales que dependen de la topología del sistema y de la prioridad empleada en las comunicaciones (se muestra la media)

Tal y como se muestra en la Figura 14, el rendimiento de las diferentes máquinas virtuales se mantiene y la máquina virtual que consume menos recursos es la de Oracle, seguida por la de Sun y la de IBM. Esta última es la más ineficiente para todas las configuraciones probadas en todos los experimentos.

Por último es de destacar un efecto (Figura 14) que ocurre muy a menudo en sistemas distribuidos: la paralelización parcial de la ejecución. A priori parece raro que el coste en distribuido sea menor que en centralizado (pues la red introduce retardos adicionales). Sin embargo el que exista una unidad de cómputo remoto acorta el coste de la ejecución notablemente, reduciendo notablemente el tiempo de respuesta. En este ejemplo dicho tiempo de respuesta pasa de estar alrededor de los dos milisegundos a bajar del milisegundo, lo que significa reducir a la mitad el tiempo de respuesta al introducir el elemento red.

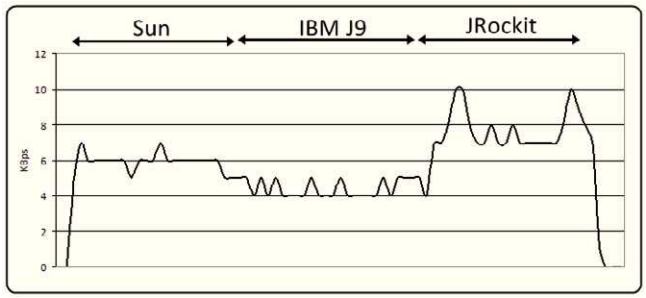

Figura 15: Consumo de Ancho de Banda de red realizado por las diferentes maquinas virtuales en el subastador

5.3 Consumo de Ancho de Banda

Se ha medido el consumo de ancho de banda realizado por cada una de las tres máquinas virtuales contempladas en la evaluación. Se ha visto que IBM J9 es la máquina virtual que consume un mayor ancho de banda (4 Kbps) y que JRrockit la que más consume (7 Kbps) mientras que Sun se queda en medio (6 Kbps). Dichas diferencias ponen de manifiesto que las tres máquinas virtuales usan diferentes implementaciones para las comunicaciones TCP/IP y/o la serialización de objetos utilizada por RTZen.

Finalmente, se ha medido el consumo de disco físico utilizado por la herramienta que soporta el almacenamiento de trazas de ejecución en disco. Los resultados obtenidos (Figura 16) no muestran gran diferencia entre el consumo de disco realizado por ninguna de las tres máquinas virtuales utilizadas en la implementación.

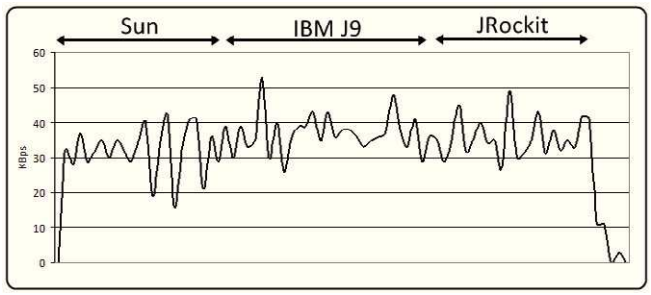

Figura 16: Consumo de almacenamiento en disco realizado por el almacenamiento de trazas de la aplicación

6. Otros middlewares aplicables

A la hora de comparar con diferentes alternativas del estado del arte se pueden analizar otras arquitecturas de tiempo real que podrían dar soporte al subastador. Por un lado nos podemos mover

por un eje arquitectónico y ver otras opciones a la hora de realizar el sistema de subastas que no estén tan altamente acopladas. Una solución es la utilización de JMS (Sun Microsystems, 2002) lo que permite aumentar la escalabilidad del sistema. Desde la óptica de nuestro sistema JMS podría sustituir a RTZen como elemento de comunicación. Pero si se hiciese ese cambio, se estarían sacrificando los mecanismos de calidad de servicio subyacentes a cambio.

También se podría optar por utilizar otras implementaciones de CORBA de tiempo real para C como es el caso de TAO (Krishna, Schmidt, Klefstad, 2004). La utilización del lenguaje del lenguaje de programación C hace más rápida la ejecución del subastador pero a cambio el esfuerzo necesario para implementar el subastador se hace mayor. En dicha implementación para C se podría reutilizar las interfaces diseñadas en este artículo.

Otra tecnología que se podría utilizar el modelo de eventos de tiempo real de TAO (Harrison, Levine, Schmidt, 1997). Dicho modelo ofrece predictibilidad en un modelo de comunicaciones basado en eventos. El mayor problema que presenta es el de no tener estandarizadas sus interfaces.

Otra clara opción pasa por la utilización de DDS (Data Distribution Service) (Pardo-Castellote, 2003) como mecanismo que permitiese desacoplar las comunicaciones entre diferentes entes del sistema. La ventaja sería doble: una arquitectura más distribuida y un modelo de calidad de servicio. Aún así, su modelo de programación no está tan bien estudiado desde el punto de vista de la programación como puede estarlo el modelo de invocación remota. Es de reseñar que algunos autores han intentado simplificar dicho modelo (Garcia-Valls *et al.*, 2010)(García-Valls, Basanta-Val, Estévez-Ayres, 2011; García-Valls, Rodríguez-López, Fernández-Villar, 2012; García-Valls, Alonso, de la Puente, 2012) preservando su principal ventaja: su completo modelo de calidad de servicio.

Sin salir de la dimensión de un middleware de tiempo real con soporte Java, otra opción sería la de utilizar DRTSJ (The Distributed Real-Time Specification for Java)(Anderson and Jensen, 2006). El principal problema existente en esa línea de trabajo es que actualmente no existen implementaciones para DRTSJ y sólo algunas implementaciones ofrecen soluciones no estándar (como por ejemplo (Basanta-Val, Garcia-Valls, Estevez-Ayres, 2010) o (Tejera, Alonso, de Miguel, 2007)). La principal ventaja que daría esta vía sería el poder utilizar abstracciones más potentes como por ejemplo sería la eliminación del recolector de basura en comunicaciones extremo a extremo (Basanta-Val, Garcia-Valls, Estevez-Ayres, 2010a) o modelos más elaborados para los hilos (Basanta-Val, Garcia-Valls, Estevez-Ayres, 2009)(Basanta-Val, Garcia-Valls, Estevez-Ayres, 2011), la serialización eficiente y el uso de protocolos avanzados de comunicación (Basanta-Val *et al.*, 2011).

Por último, se podría orientar la solución hacia las arquitecturas basadas en servicios ((Estévez-Ayres *et al.*, 2011; Xu, 2011; García-Valls, Rodríguez-López, Fernández-Villar, 2012)(García-Valls, M and Basanta-Val, P 2012)(Cucinotta *et al.*, 2009)(Basanta-Val, Garcia-Valls, Estevez-Ayres, 2012)). A este respecto es de destacar que la implementación diseñada sería reutilizable siempre y cuando se utilizase una implementación de servicio web que ofrezca soporte a implementaciones realizadas en CORBA. Aunque muy posiblemente los tiempos de respuesta se verían mermados por el procesado de mensajes y caberas en formato `xml`.

7. Conclusiones

En este trabajo se ha explorado el diseño e implementación de un subastador con capacidad básica de calidad de servicio. Se han definido el modelo de dicho subastador así como los algoritmos internos utilizados a la hora de realizar el mecanismo de puja. Los resultados empíricos han fijado unos tiempos de referencia para las operaciones de subasta que tienen una gran dependencia de la máquina virtual utilizada. El diseño del subastador es sencillo y es por tanto susceptible de ser modelado y utilizado en herramientas que simulen su funcionamiento.

Nuestro trabajo de bajo nivel continúa a partir de aquí en dos líneas diferentes que se complementan. En primer lugar estamos trabajando en versiones de tiempo real de las estrategias ZIP y GD que permitan establecer límites temporales en el proceso de subasta. Por otro lado, estamos evaluando la utilización de herramientas de modelado (Pasaje, Harbour, Drake, 2001) que permitan dimensionar adecuadamente el sistema de subastas descrito. También se está integrando en modelo propuesto con arquitecturas con acceso seguro desde Internet abordando aspectos de seguridad y capacidad para atravesar cortafuegos. Por último, en el alto nivel, se evalúa el modelo de subasta desarrollado dentro de la arquitectura Java desarrollada para sistemas industriales descrita en (Basanta-Val, Garcia-Valls, Estevez-Ayres, 2010b).

English Summary

Distributed Java Auction Module on real-time CORBA

Abstract

Using middleware infrastructures that enable communication among different nodes in a system may alleviate aspects related to management and development cost in flexible environments. In that context the article presents an auction architecture with certain real-time requirements that have to be satisfied. The architecture implements a continuous double action (CDA) algorithm for a real-time middleware. The article describes the architecture (actors and goals) and the results obtained by using common-off-the-shelf real-time middleware.

Keywords:

Action Systems, Real-time, Middleware, Industrial informatics, RT-CORBA, Real-time Java

Agradecimientos

Este trabajo ha sido realizado con el apoyo del proyecto iLAND (ARTEMIS-JU 100026) financiado por ARTEMIS JTU y el Ministerio de Industria, Comercio y Turismo español. También ha sido parcialmente financiado por ARTISTDesign NoE (IST-2007-214373) del 7º programa marco de la Unión Europea y por REM4VSS (TIN2011-28339) del Ministerio de Ciencia e Innovación. Los autores quieren también agradecer a los anónimos revisores y al editor asociado encargado de este artículo su aporte sobre el conjunto del artículo.

Referencias

Aleksy M, Korthaus A, Schader M (2001) A CORBA-based Architecture for Electronic Auction Applications.

Anderson JS, Jensen ED (2006) Distributed real-time specification for Java: a status report (digest), 3-9.

Basanta-Val P, Anderson JS (2012) Using Real-Time Java in Distributed Systems: Problems and Solutions. In: TH Toledano, AJ Wellings, eds. *Distributed and Embedded Real-Time Java Systems*.

Basanta-Val P, Garcia-Valls M, Estevez-Ayres I (2012) Enhancing OSGi with real-time Java support. *Software: Practice and Experience*, -(-), -,-.

Basanta-Val P, Garcia-Valls M, Estevez-Ayres I (2011) A Dual Programming Model for Distributed Real-Time Java. *Industrial Informatics, IEEE Transactions on*, 758.

Basanta-Val P, Garcia-Valls M, Estevez-Ayres I (2010) A neutral architecture for distributed real-time Java based on RTSJ and RMI, 1-8.

Basanta-Val P, Garcia-Valls M, Estevez-Ayres I (2009) Simple Asynchronous Remote Invocations for Distributed Real-Time Java. *Industrial Informatics, IEEE Transactions on*, 5(3), 289-98.

Basanta-Val P, García-Valls M, Fernandez-Gonzalez J, Estevez-Avres I (2011) Fine tuning of the multiplexing facilities of Java's Remote Method Invocation, 1260.

Basanta-Val P, Garcia-Valls M, Estevez-Ayres I (2010a) No-Heap remote objects for distributed real-time Java. *ACM Trans.Embed.Comput.Syst.*, 10(1), 1-25.

Basanta-Val P, Garcia-Valls M, Estevez-Ayres I (2010b) Towards a Cyber-Physical Architecture for Industrial Systems via Real-Time Java Technology. *Computer and Information Technology, International Conference on*, 0, 2341-6.

Bollella G. et al. (2001) The Real-Time Specification for Java.

Bussmann S, Schild K (2000) Self-organizing manufacturing control: an industrial application of agent technology, 87.

Cliff D (2003) Explorations in evolutionary design of online auction market mechanisms. *Electronic Commerce Research and Applications*, 2(2), 162-75.

Cucinotta T, Mancina A, Anastasi GF, Lipari G, Mangeruca L, Checcozzo R, Rusina F (2009) A Real-Time Service-Oriented Architecture for Industrial Automation. *Industrial Informatics, IEEE Transactions on*, 5(3), 267.

Estévez-Ayres I, García-Valls M, Basanta-Val P, Díez-Sánchez J (2011) A hybrid approach for selecting service-based real-time composition algorithms in heterogeneous environments. *Concurrency and Computation: Practice and Experience*, 23(15), 1816-51.

García-Valls M, and Basanta-Val P (2012) Usage of DDS Data-Centric Middleware for remote monitoring and control laboratories. *Industrial Informatics, IEEE Transactions on*, -.

García-Valls M, Rodríguez-López I, Fernández-Villar L (2012) iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems. *Industrial Informatics, IEEE Transactions on*, -.

Garcia-Valls M, Rodriguez-Lopez I, Fernandez-Villar L, Estevez-Ayres I, Basanta-Val P (2010) Towards a middleware architecture for deterministic reconfiguration of service-based networked applications, 1-4.

García-Valls M, Basanta-Val P, Estévez-Ayres I (2011) Real-time Reconfiguration in Multimedia Embedded Systems. *Consumer Electronics, IEEE Transactions on*, 1287.

García-Valls M, Alonso A, de la Puente J.A. (2012) A dual-band priority assignment algorithm for dynamic QoS resource management. *Future Generation Computer Systems*(6), 902-12.

Harrison TH, Levine DL, Schmidt DC (1997) The design and performance of a real-time CORBA event service, 184-200.

Klemperer P (1999) Auction Theory: a Guide to the Literature(2163).

Krishna AS, Schmidt DC, Klefstad R (2004) Enhancing Real-Time CORBA via Real-Time Java Features, 66-73.

Pardo-Castellote G (2003) OMG Data-Distribution Service: Architectural Overview, 200-6.

Pasaje JLM, Harbour MG, Drake JM (2001) MAST Real-Time View: a graphic UML tool for modeling object-oriented real-time systems, 245.

Phipps G (1999) Comparing observed bug and productivity rates for Java and C++. *Softw.Pract.Exper.*, 29(4), 345-58.

Raman K, 0001 YZ, Panahi M, Colmenares JA, Klefstad R, Harmon T (2005) RTZen: Highly Predictable, Real-Time Java Middleware for Distributed and Embedded Systems, 225-48.

Schmidt DC, Kuhns F (2000) An Overview of the Real-Time CORBA Specification. *IEEE Computer*, 33(6), 56-63.

Siegel J (1999) A Preview of CORBA 3. *IEEE Computer*, 32(5), 114-6.

Sun Microsystems (2002) Java Messaging System.

Tejera D, Alonso A, de Miguel MA (2007) RMI-HRT: remote method invocation - hard real time, 113-20.

Vytelingum P (2006) The Structure and Behaviour of the Continuous Double Auction.

Xu LD (2011) Enterprise Systems: State-of-the-Art and Future Trends. *Industrial Informatics, IEEE Transactions on*, 7(4), 630.