Plataformas *Middleware* comerciales para la integración de flujos de vídeo bruto

Antonio Javier García Sánchez, Felipe García Sánchez, Francisco Javier Díaz Jiménez, Joan García Haro
Departamento de Tecnologías de la Información y las Comunicaciones. Universidad Politécnica de Cartagena
Campus Muralla del Mar. C/ Dr. Fleming s/n.
30202 Cartagena (Murcia)
Teléfono: 968 32 65 38 Fax: 968 32 59 77
E-mail: {antoniojavier.garcia, felipe.garcia, fjavier.diaz, joang.haro}@upct.es

Resumen. *El documento realiza un estudio sobre el uso de distintos componentes COTS (Comercial Off-The-Shelf) dirigido al desarrollo de aplicaciones que hagan posible la integración de flujos de vídeo. El objetivo es presentar tres plataformas de integración (CORBA, JavaRMI y .Net Remoting) que permitan construir aplicaciones y servicios distribuidos para transmitir y recibir flujos de vídeo sin compresión. CORBA y dentro de este, aquellos ORB's que incorporen entre sus servicios el denominado A/V Stream, hace posible la implementación de este tipo de aplicaciones. JavaRMI y más recientemente .Net Remoting son dos plataformas que permiten el desarrollo de aplicaciones distribuidas multimedia, pero con limitaciones propias: en el caso de JavaRMI, el lenguaje de programación (Java) y en el caso de .Net Remoting, el sistema operativo (Windows).*

1 Introducción

Tradicionalmente, las imágenes de vídeo tanto en Internet como en redes privadas son transmitidas en formato comprimido según alguno de los estándares existentes (por ejemplo, MPEG) [1]. Las imágenes transmitidas no son iguales a las que se capturan en origen, pudiendo perder sus características naturales (definición, color, etc.). La información total transmitida es menor que la enviada sin comprimir, consiguiendo reducir el ancho de banda de transmisión necesario, retardos, etc.

Existen entornos o aplicaciones en las cuáles no resulta conveniente la utilización de vídeo comprimido. En aplicaciones militares o médicas, la pérdida de un píxel puede suponer fallos notables y hasta catastróficos. Dentro de las primeras destacan, las salidas de imagen de un periscopio o del visor de un carro de combate. Respecto a las médicas, la supervisión de una operación por parte de un especialista externo en la que se necesite tener la máxima definición posible, o conocer el color exacto en cada punto.

Otro campo de aplicación es la tele-vigilancia. La vigilancia externa del núcleo de una central nuclear, o la seguridad en el almacenamiento de armamentos nucleares, químicos o biológicos. En todos los casos se puede estar buscando un pequeño deterioro, y por tanto, se deben evitar compresiones y pérdidas de información. De igual forma, es interesante poder realizar esta vigilancia de forma remota para evitar que una fuga perjudique al personal encargado.

La gran cantidad de información que se maneja en este tipo de transmisiones de vídeo, hace que se suela utilizar como soporte de transmisión infraestructura dedicada (RGB) hacia una estación de control cercana, donde un operador es el encargado de visualizar estas operaciones.

Sin embargo, la evolución de las tecnologías de red como es el caso de la familia Ethernet, permite la utilización de mayores anchos de banda y en consecuencia, la integración de flujos de transmisión elevados utilizando la red local que posea una instalación. Además, la integración de este tipo de flujos proporciona información sobre el comportamiento de éste y otros tipos de redes (Wireless, Giga-Ethernet, etc.) donde se les dé soporte y además, permite comprobar el límite que ofrecen dichas tecnologías.

2 Plataforma Middleware

A la hora de seleccionar una plataforma de comunicaciones para la transmisión de información de vídeo bruto se ha tenido en cuenta que las aplicaciones desarrolladas sean distribuidas, de tal manera que el servidor (facilita el servicio) y el/los cliente/-s (consumen ese servicio) se encuentre/-n en equipos distintos y conectados de manera remota.

La implementación a través de herramientas comerciales de integración como son los sistemas distribuidos *middleware*, proporciona un *software* que facilita y simplifica la construcción de aplicaciones/servicios distribuidos, proporcionando mecanismos que los componentes distribuidos utilizan para comunicarse e interaccionar a través de la red.

El middleware es una capa situada entre la aplicación y la capa de transporte que independiza las distintas aplicaciones software de los PC´s (clientes y

servidores) de la plataforma de comunicaciones subyacente. El programador puede proceder sin tener en cuenta la tecnología concreta de comunicaciones empleada, ni el lenguaje de programación utilizado en la capa de aplicación.

Entre los diferentes tipos de paquetes de software que cumplen en menor o mayor medida estas propiedades pueden incluirse el *middleware* basado en intermediarios o *brokers* (ORB, *Object Request Broker*). Dentro de dichos paquetes de software destacan tres: CORBA (*Common Object Request Broker Addressing*) [2], JavaRMI y .NET Remoting.

2.1 CORBA

Desde el comienzo como estándar de integración, CORBA ha soportado interoperatividad entre sistemas heterogéneos, proporcionando una comunicación flexible, y una base para el desarrollo de entornos distribuidos.

Sin embargo, los requerimientos que exigen las aplicaciones distribuidas multimedia (gestión de QoS, tiempo real, manejo de flujos continuos, etc.) hacen que la política tradicional de *request-reply* de CORBA no consigue los resultados deseados.

Los servicios CORBA de flujos de audio y vídeo soportan la transmisión de flujos de vídeo en tiempo real, así como un marco de trabajo (*framework*) que facilita la creación, envío y recepción de este tipo de flujos entre dos o más dispositivos conectados a una red. Por ello, de entre los distintos CORBA's comerciales (MICO, Orbit, ACE-TAO [3], etc.) se eligen aquéllos que proporcionan este tipo de servicios.

El funcionamiento de este servicio inicia con la captura de vídeo por una cámara y recogido por la aplicación *Sender*, que lo envía sin comprimir a la aplicación *Receiver*. Para ello y utilizando el protocolo IIOP (*Internet Inter ORB protocol*), el receptor crea una instancia de un objeto y, de forma transparente, el ORB localiza la implementación correspondiente al objeto referenciado y devuelve una referencia al objeto remoto ubicado en el servidor. La aplicación receptora invoca el método y el receptor envía los parámetros a través de la red, que llegan a la aplicación servidora y son entregados a la aplicación. Una vez realizado el establecimiento y configuración de la conexión, la aplicación servidora comienza a enviar los paquetes de datos de las imágenes al receptor. Véase fig. 1.

CORBA ofrece otras características como es el comportamiento en tiempo real, operaciones *multicast*, calidad de servicio, etc. Además, las aplicaciones pueden ser desarrolladas en multitud de lenguajes de programación (C++, Java, Ada, etc.).

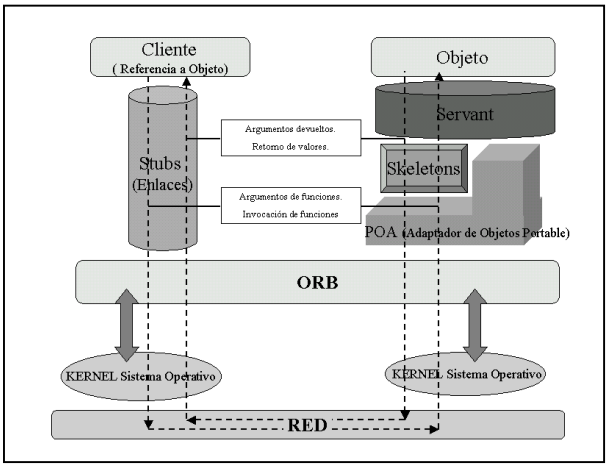

Fig. 1: *Arquitectura CORBA*

2.2 JavaRMI

El sistema de Invocación Remota de Métodos (RMI) de Java creado por *Sun Microsystems*, permite a un objeto que se está ejecutando en una Máquina Virtual Java (VM) llamar a métodos de otro objeto que está en otra VM diferente.

La implementación RMI está construida sobre tres capas. La primera de ellas es la capa *Stub* (cliente) o *Skeleton* (servidor). Esta capa intercepta las llamadas a métodos realizadas por el cliente y las redirige al servicio RMI remoto. La capa *Remote Reference Layer* interpreta y dirige las referencias hechas desde el cliente al objeto del servicio remoto. Por último, la capa de transporte está basada en conexiones TCP/IP entre máquinas de una red de comunicaciones. Véase fig. 2.

Usando esta arquitectura de capas, se puede reemplazar una de ellas sin afectar a otras. Por ejemplo, la capa de transporte puede ser sustituida por una capa UDP/IP sin afectar a las capas superiores.

Las aplicaciones pueden utilizar uno de los dos mecanismos para que los clientes obtengan referencias a objetos remotos, por una parte el servidor puede registrar sus objetos remotos con la facilidad de nombrado de RMI *rmiregistry* o puede pasar y devolver referencias de objetos remotos como parte de su operación normal. Una vez que el cliente obtiene la referencia al objeto, RMI proporciona el

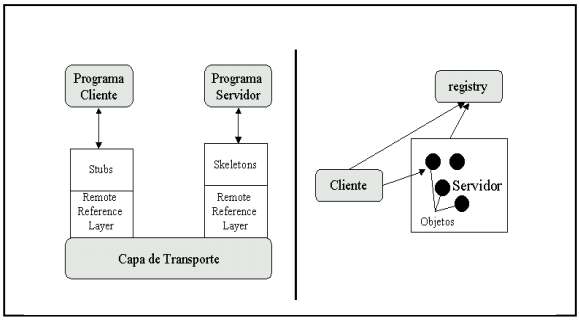

Fig. 2: *Arquitectura Java RMI*

mecanismo por el que se comunican y se pasan los paquetes de datos de vídeo del servidor al cliente [4].

2.3 .NET Remoting

Creado por Microsoft, consiste en una serie de librerías que permite a los programadores construir de una forma rápida aplicaciones distribuidas. Está escrito de acuerdo al Common Language Specification (CLS) y define un lenguaje intermedio (IL), que es equivalente al código objeto obtenido de un compilador convencional. Los programas y componentes escritos en una variedad de lenguajes diferentes pueden ser compilados por IL y linkados juntos. Microsoft ha implementado como lenguajes compatibles a IL, C#, Visual Basic (VB) y C++. Una vez obtenido el código IL linkado y usando el compilador *Just in Time* (JIT) se obtiene el ejecutable binario [5].

Las características más importantes al desarrollar una aplicación con .Net Remoting son las siguientes:

- El canal de comunicación usado por la aplicación para llamar a objetos remotos. Éste puede ser HTTP o TCP y ambos utilizan el protocolo SOAP (*Simple Object Access Protocol*) para la comunicación.
- La selección de la aplicación servidora para el objeto remoto. Ésta puede ser una aplicación consola, aplicación Windows o Servicio Windows.
- El modo de activación del objeto remoto. El objeto puede ser creado una única vez y todos los clientes se comunican con él, o crear una nueva instancia de la clase cada vez que haya una llamada hecha por un cliente.
- La forma de obtener la información del objeto remoto por parte del cliente. Se utiliza una clase *proxy* desde el objeto remoto. El *proxy* es incluido en el proyecto del cliente y usado como si fuera un tipo .Net local (internamente las llamadas del cliente se enrutan al objeto remoto).

A la hora de realizar una aplicación multimedia, la aplicación servidora registra un canal y un puerto donde son escuchadas las llamadas entrantes. La lectura de un fichero de configuración con parámetros como el modo de activación, hace que la información del objeto remoto sea recuperada (la aplicación servidora puede instanciar al objeto cuando lo requiera). La aplicación cliente también registra el mismo canal y crea una nueva instancia de la clase remota. El sistema *remoting* maneja las llamadas para la nueva instancia proporcionando un objeto *proxy* en lugar del actual objeto. Este objeto es creado inmediatamente en la primera llamada que realiza el cliente (el entorno *remoting* lo realiza automáticamente). Cuando los clientes llaman a un método del objeto *proxy*, éste referencia los valores del objeto remoto, y el servidor envía los paquetes de vídeo a través del canal seleccionado [6].

3 Conclusiones

La utilización de vídeo bruto unido al uso de un *middleware* proporciona un marco para el estudio del comportamiento de redes de comunicaciones con elevadas tasas de utilización [7].

Son muchas las ventajas que comporta la utilización de un *middleware*, algunas de ellas son:

- Los códigos del cliente y servidor están aislados de las capas de transporte y protocolos de comunicación inferiores.
- La gestión de conexión es transparente al usuario.
- Respecto a CORBA, el cliente y servidor se comunican sin importar el lenguaje de implementación utilizado ni la arquitectura del hardware.
- Las aplicaciones servidoras son transportables, es decir, pueden migrar sin cambios de una máquina a otra.

Por último se han descrito tres ORB's (CORBA, JavaRMI y .Net Remoting). Dentro de éstos, CORBA es el más flexible siendo a su vez el más complejo de aprender e implementar.

Estos tres *middlewares* presentados en este estudio, no son independientes, al contrario, los tres son enlazables por medio protocolo IIOP (RMI-IIOP o IIOP.Net).

Referencias

[1] S. Bottazi, S. Casselli, M. Regiani, M. Amoretti, "A software framework based on Real-Time CORBA for telerobotics systems", IROS 2002-b, International Conference on Intelligent Robots and Systems, pp. 985-992, Lausanne, Switzerland.

[2] Henning and Steve Vinoski, "Advanced CORBA programming with C++", Prentice-Hall 2000.

[3] Douglas C. Schmidt, Computer and Science Engineering of Washington University, "http://www.cs.wustl.edu/~schmidt/TAO.html".

[4] William Grosso, "JavaRMI", O'Reilly 2001.

[5] David Conger, "Remoting with C# and .NET", Wiley Publishing, Inc 2003.

[6] Ingo Rammer, "Advanced .NET Remoting", Ed: Apress 2002.

[7] Felipe García Sánchez, Antonio Javier García Sanchez, Juan Ángel Pastor Franco, "Use of COTS for Raw Video Streams Integration", IEEE PACRIM, pp. 671-672, University of Victoria, Canada, 28-30, August, 2003.