Demostrador de una arquitectura de videoconferencia en la Web 2.0

J. Cervino (jcervino@dit.upm.es), J. Salvachúa (jsr@dit.upm.es), P. Rodríguez (prodriguez@dit.upm.es), G. Huecas (gabriel@dit.upm.es), J. Quemada (jquemada@dit.upm.es)

Dit ETSIT. Universidad Politécnica de Madrid

***Abstract* — En este demostrador presentaremos el funcionamiento de un sistema de colaboración multimedia basado en la utilización de clientes ligeros desplegados en la *web*. El objetivo de este sistema es definir una nueva arquitectura de conferencias en *Internet* centrada en la facilidad de uso y de instalación, dotando de conectividad total al usuario final con el resto de participantes. En este documento presentaremos el diseño de la solución y describiremos en qué consistirá la demostración, que mostrará los resultados que verifican la validez de este concepto.**

I. Introducción

Hasta ahora las aplicaciones multimedia distribuidas en la red de redes (Internet) son de una complejidad de desarrollo muy importante, a la dificultad intrínseca de cualquier aplicación distribuida en Internet, hay que sumar los requisitos de retardo propios de las aplicaciones multimedia en tiempo real donde el tiempo de respuesta es muy importante de cara a obtener un resultado satisfactorio. Por todo lo anterior, este tipo de aplicaciones generalmente quedaban reducidas a un entorno bastante restringido con gran ancho de banda y capacidad de proceso del servidor. Sin embargo, ante el constante aumento del ancho de banda disponible y de capacidad de proceso de las máquinas, estas utilidades han ido ampliando su interés de cara a un público más general lo que, a su vez, ha fomentado el desarrollo de nuevas plataformas para su desarrollo que disminuyen su complejidad y aumentan la facilidad de instalación y uso.

A partir de experiencias previas se pretende conseguir una aplicación de videoconferencia independiente del sistema operativo y que alcance resultados parecidos reduciendo sensiblemente la complejidad de desarrollo y de la arquitectura tanto del servidor como del cliente, aliviando las dificultades para desarrolladores y usuarios.

Como experiencia previa a este trabajo existen otros trabajos del mismo grupo de investigación, todos ellos basados en la creación de herramientas de colaboración multimedia. Estos sistemas son las versiones 1.0 y 2.0 de la herramienta *Marte*. La primera versión de *Marte* se desarrolló en el año 2004 como una alternativa a los servicios de colaboración multimedia que existían en aquel momento, los cuales imponían requisitos muy fuertes de computación a las máquinas que los albergaban; el objetivo principal fue crear una plataforma orientada al usuario medio de Internet utilizando poco consumo del ancho de banda y de procesador sin necesidad de que el usuario tuviera conocimientos avanzados de comunicaciones en Internet; para ello se utilizó el protocolo de inicio de sesión (SIP). *Marte* 2.0 fue una continuación de la primera versión con varias mejoras, entre las que destacaban la posibilidad de crear múltiples salas de conferencia, comunicación a través de NAT y de *Firewalls*, etc. Las aplicaciones cliente de estos sistemas (que eran sistemas cliente-servidor) estaban hechas para la plataforma *Microsoft Windows* y programadas en .NET.

El problema al que nos enfrentamos en las versiones anteriores de *Marte* no era el ahorro de ancho de banda mediante el uso de complejas codificaciones y la complejidad de los protocolos de nivel de transporte, si no que la aplicación cliente resultante no siempre funcionaba debido a motivos de conectividad (por la existencia de *Firewalls* muy restrictivos) o de instalación (porque sólo era compatible con sistemas *Windows*, y era muy complicado portarlo a otros sistemas), por lo que en la siguiente versión sacrificaríamos si fuese necesario este ahorro e incluso la calidad de los datos multimedia en favor de la conectividad. La idea que surge a partir de *Marte* 2.0 es conseguir un sistema cliente/servidor en el que la mayor carga de trabajo resida en el servidor mientras que los clientes sean muy ligeros en cuanto a computación y, por otra parte, se intenta lograr éxito en la conectividad entre todos los clientes en el mayor número de escenarios posible. Para conseguir esta conectividad nos apoyamos como explicaremos en el artículo en el buque insignia de la conectividad que hoy en día arrasa en todo *Internet*: la *web 2.0*.

El documento se ha organizado como sigue: en la sección II describimos la arquitectura general del sistema introduciendo los objetivos principales del proyecto, en la sección III pasamos a los detalles de los diferentes elementos, en la sección IV explicamos las conclusiones del trabajo y se comentan los posibles trabajos futuros.

II. Objetivos de Marte 3.0

A la vista de los problemas encontrados en las versiones anteriores de *Marte*, se comenzó a diseñar una nueva arquitectura más sencilla que basada en los puntos que iremos explicando en este apartado.

Entorno para *web* 2.0: El primer objetivo fue encontrar un entorno de desarrollo que permitiera crear esta arquitectura de conferencias multimedia de forma que fuera capaz de ejecutarse en el mayor número de escenarios posible. Por eso centramos la base de nuestro estudio no en el servidor, ni en la tecnología del servidor, sino en los distintos clientes y en las

posibilidades que ofrecía cada uno de cara a videoconferencias. Para ello la solución más interesante que encontramos fue la de crear un cliente accesible vía *web* y que pudiera ser ejecutado en el mayor número de casos posible, incluyendo la adptación sencilla de dispositivos móviles. Para ello la tecnología *Flash de Adobe* [11] presenta un entorno de carácter abierto desde hace ya varios años que permite crear aplicaciones web de forma sencilla, pero con capacidades multimedia muy potentes. Este entorno se llama *Adobe Flex*.

El segundo objetivo fue enriquecer la arquitectura lo máximo posible añadiendo servicios al audio y el vídeo. Para ello entre los flujos también deberíamos incluir el de compartición de escritorio, de forma que fuera posible crear salas de conferencia en las que uno o varios usuarios pudieran mostrar las ventanas que había en el escritorio. Este servicio es muy útil en conferencias en las que existe un ponente que quiere hacer presentaciones ayudándose de transparencias o diapositivas. La tecnología que se escogió para este propósito fue la computación de red virtual (VNC [7]) éxito en las versiones anteriores debido a que es una tecnología muy madura que da muchas posibilidades. Sin embargo esta tecnología de por sí propone conexión punto a punto, sin utilizar servidores o *proxies*. Como en anteriores ocasiones nosotros utilizamos un *proxy* intermedio que nos permite ahorrar ancho de banda en los clientes y lograr éxito en la mayor número de escenarios de conexión.

La idea de utilizar un cliente web permite acercar este servicio al usuario medio de Internet, el cual podría tener conocimientos muy escasos sobre las aplicaciones de videoconferencia. Por lo que un requisito importante fue el de crear una interfaz sencilla e intuitiva al usuario, basada en el hecho de que crear, eliminar y unirse a conferencias sea una tarea rápida y directa y el hecho de hablar con los demás usuarios de cada sala de conferencia no requiera conocimientos multimedia avanzados.

El último objetivo es herencia de la primera versión de *Marte*, que se basaba en la creación de un sistema flexible y escalable. Para ello la mejor manera es crear una aplicación abierta, que pueda ser fácilmente modificada y extendida sin ningún tipo de impedimento por licencias restrictivas.

Como resultado de la consecución de estos objetivos nos encontramos con una arquitectura totalmente nueva de Marte, caracterizada por el modelo cliente/servidor. El servidor utilizado será abierto y de carácter centralizado por el que pasará cada flujo de información multimedia de las conferencias. Este servidor se utilizará como indicador de presencia de los clientes, proxy de las sesiones VNC, servidor de autenticación, y distribuidor de los datos de vídeo y audio. La parte cliente será una aplicación del mundo de la *web* 2.0, ejecutable en la mayoría de navegadores *web* y de sistemas operativos, con la idea de ser un cliente ligero y muy sencillo de utilizar aprovechando las posibilidades de *Flex* para crear interfaces de usuario.

III. Implementación de Marte 3.0

La arquitectura del servidor está basada en Red5, un servidor *Flash* de código abierto que nos permite, entre otras cosas, distribuir contenidos multimedia a través del protocolo RTMP o de su variante sobre HTTP, RTMPT.

Red5 se sostiene en la estructura MINA (*Multi-purpose Infrastructure for Network Applications* [13]) de *Apache* que permite desarrollar aplicaciones de red altamente escalables basándose en parte en la arquitectura SEDA [3] (*Staged Event-Driven Architecture*).

Como se menciona anteriormente, *Red5* utiliza el protocolo RTMP (o RTMPT) para la distribución de contenidos multimedia. El objetivo de este protocolo es que los clientes hechos con la tecnología *Flash* (en su momento también de *Macromedia*) pudieran enviar y recibir datos en tiempo real con un mínimo de garantía desde y hacia el servidor *Flash Media Server*. Una de las características que permite esto es la posibilidad de tunelar el tráfico para que los datos viajen como si fueran el cuerpo de mensajes HTTP, de forma que no sean tan fácilmente visibles en dispositivos como *firewalls* y gestores de tráfico que son configurados en la frontera de las redes empresariales y personales para filtrar contenidos de este tipo. Esta última característica facilita enormemente el uso del servicio de videoconferencia en ambientes muy restringidos sin, en principio, tener que implementar cambios importantes en el cliente ni en el servidor.

En lo que concierne a *Marte* 3.0 se ha aprovechado las posibilidades que ofrece *Red5* para crear nuevas aplicaciones de servidor y se han implementado las partes necesarias para cumplir con los requisitos expuestos por las versiones anteriores de *Marte* con ciertas variaciones en algunos casos para adaptarse a la nueva situación. Como resultado el servidor almacena la información de los usuarios en LDAP, mantiene un servicio de presencia de los clientes conectados y las capacidades de cada cliente (si tienen el vídeo, el audio o el escritorio compartidos). La mayor parte de esta comunicación se realiza mediante llamadas a métodos remotos desde los clientes aprovechando las capacidades para ello ofrecidas por *Red5*, las respuestas del servidor son enviadas como objetos serializados e interpretadas por el cliente como sea conveniente.

Mención aparte merece al solución de escritorio compartido adoptada. Para ella se optó por utilizar VNC *Reflector* [16] en el servidor. VNC *Reflector* es un servidor especial de VNC que actúa como *proxy* entre un servidor de VNC y los clientes, evitando efectivamente el problema planteado, ya que los clientes pasan a conectarse, a la hora de compartir el escritorio, directamente a la misma dirección en la que se encuentra el Red5. Esto obliga a crear una capa entre la aplicación de servidor existente en *Red5* y dicho VNC Reflector para arrancarlo con los parámetros adecuados (los puertos de a los que deben conectarse el cliente que comparte su escritorio y el resto) e informar sobre los servidores existentes y las correspondencias necesarias, esta capa actúa como indica la figura 1.

La arquitectura del cliente la hemos creado basándonos en el concepto de aplicaciones ricas de Internet, que fue descrito en [4], y actualmente está teniendo mucho auge en Internet. Según este artículo una tecnología de cliente rica debe cumplir con los siguientes aspectos: debe proveer un entorno de ejecución eficiente que permita además de ejecutar el código, tratar contenidos y comunicar el cliente con aplicaciones externas, debe integrar en un entorno común estos contenidos, comunicaciones e interfaces (no como el actual HTML), permitir que el usuario interactúe con modelos de objetos extensibles (mejorando lo que nos proporciona *Java Script* y DHTML), facilitar un desarrollo rápido de las aplicaciones a través del uso de componentes reutilizables, comunicarse con servidores de aplicaciones a través de los servicios de datos y de la web, permitir a las aplicaciones ejecutarse en los estados conectado y desconectado, y ser fácilmente desarrollable en múltiples plataformas y dispositivos.

De todos los marcos de desarrollo de aplicaciones RIA existentes en la actualidad en nuestro caso optamos por utilizar *Adobe Flex*, ya que es un marco de desarrollo de código abierto que permite crear flujos de vídeo y audio desde el navegador de forma muy sencilla, tanto para el desarrollador como para los usuarios finales.

La arquitectura del cliente, que podemos ver en la figura 2, se puede dividir en cuatro módulos que se diferencian en el propósito de cada uno de ellos: Módulo del control de sesión, de control de los flujos multimedia y del escritorio compartido.

El primero de ellos es el que contempla todas las clases encargadas de enviar y recibir mensajes al servidor Red5 manteniendo el estado de sesión y datos sobre los usuarios conectados, las salas disponibles y el modo de presentación que hay en la sala en la que el usuario está conectado. El control de los flujos multimedia se hace utilizando el API que ofrece el marco de desarrollo Flex. El módulo del escritorio compartido se divide en dos partes: una es la implementación del cliente VNC que es producto de un proyecto de software libre llamado *FlashVNC*; la biblioteca es un cliente completo de VNC para la versión 3.3 del protocolo. Éste permite recibir la pantalla del escritorio compartido y enviar eventos del ratón y del teclado al servidor VNC. La segunda parte es la que hace de servidor VNC, que la hemos realizado con tecnología Java porque no es posible realizar capturas de la pantalla de la máquina en la que se está ejecutando la aplicación Flex.

Por encima de cada módulo se han creado interfaces definidas en MXML para mostrar al usuario la información necesaria y recibir las órdenes de éste por medio de eventos del teclado y ratón.

IV. Conclusiones

Como comentamos en la introducción el objetivo final de esta implementación era obtener un sistema de videoconferencia con un diseño fácil, con una interfaz sencilla para el usuario final y que pudiera ser utilizado en la mayoría de sistemas operativos sin necesidad de instalaciones complejas. Durante el estudio inicial comprobamos que la mejor herramienta para lograrlo era la tecnología *Flex*, por que ha sido creada por una empresa dedicada históricamente a la fabricación de sencillas herramientas de diseño *web* (antes como *Macromedia* y ahora como *Adobe*). Hemos visto como uno de los principales compromisos de *Adobe*, que es la seguridad en este tipo de aplicaciones, nos ha impedido desarrollar ciertas funcionalidades como la compartición de escritorio por lo que tuvimos que acudir a la tecnología *Java*, por lo que el resultado ha sido el que buscamos inicialmente. Comparando los sistemas anteriores con la nueva versión, comprobamos que la utilización de *Adobe Flash* presenta, por el momento, una ligera pérdida de calidad de video, pero que se ve ampliamente compensada por el hecho de no tener que instalar ningún tipo de aplicación y poder accede desde cualquier rincón de Internet.

Por ello este proyecto es un buen punto de partida para futuras investigaciones que busquen objetivos similares. El punto en contra es que en cuanto la complejidad aumenta y se quiere seguir teniendo compatibilidad con otros sistemas surgen problemas que hacen que el desarrollador se tenga que plantear el resolverlo utilizando otras tecnologías de la *web* 2.0 o incluso fuera de este escenario.

Durante la celebración del congreso, se plantea la posibilidad de realizar una demostración práctica del funcionamiento del sistema. Para ello en la figura 3 vemos las posibilidades que ofrece una arquitectura de este tipo para su utilización desde dispositivos móviles, demostrando que es posible conseguir la conexión desde un amplio conjunto de dispositivos.

Con los resultados obtenidos se han iniciado investigaciones para resolver temas más concretos del proyecto, como pueden ser la mejora del servicio de escritorio compartido para que viaje con el resto de información dentro de peticiones HTTP. O la inclusión de nuevos servicios como son la pizarra compartida o la mensajería instantánea. Otro punto de interés sería el aumento de compatibilidad con otros sistemas de videoconferencia presentes en Internet, como pueden ser *Jingle*, SIP, etc. Además la facilidad de creación de interfaces de *Flex* hace que sea posible la búsqueda de nuevos modelos de interfaz basados en la *web* 2.0 para el control de conferencias de vídeo. Por último también se está trabajando actualmente en la creación de distintos tipos de interfaces para la gestión y consulta de videoconferencias, como son los servicios *web* (*Web Services*) y *Rest*.

Agradecimientos

Agradecemos al CDTI y al Ministerio de Industria por la financiación otorgada. Asimismo agradecemos a todos los participantes en los diversos proyectos de codigo libre que nos han dado soporte para conseguir que este sistema se ha podido llevar a cabo.