Diseño de una pasarela de acceso a sistemas propietarios de videoconferencia

D. Moreno, S. Pavón, G. Huecas, P. Rodríguez

***Resumen*— Los sistemas de videoconferencia han utilizado tradicionalmente protocolos propietarios que los impedían interoperar. Sin embargo, en los últimos años se está imponiendo la tendencia a usar protocolos abiertos para solucionar este problema. Este artículo describe la arquitectura de una pasarela genérica para acceder desde los clientes más típicos a los sistemas de videoconferencia propietarios ya existentes. Esta arquitectura se ha validado implementando una pasarela de acceso a Isabel, un sistema de videoconferencia con opciones avanzadas de colaboración.**

***Palabras clave*—. Asterisk, H.323, Isabel, Jabber, PBX (*Private Branch eXchange*), red de telefonía básica (*public switched telephone network*), SIP (*Session Initiation Protocol*), videoconferencia (*videoconference*), voz sobre IP (*voice over IP*), XMPP (*Extensible Messaging and Presence Protocol*).**

I. Introducción

En el campo de las aplicaciones de videoconferencia, el modelo dominante durante muchos años ha sido el que los programas de uso más extendido sólo se comunicaban entre productos del mismo fabricante a través de protocolos propietarios. Dentro de los programas de mensajería y voz sobre IP dirigidos al usuario final, tenemos ejemplos como Skype [14] y Windows Live Messenger [15], que utilizan protocolos propietarios de comunicación. La misma situación se repite en el entorno de la videoconferencia avanzada, esto es, programas de videoconferencia a los que se añaden funcionalidades de aplicaciones compartidas (muestra de presentaciones, escritorio compartido, pizarra...). Es el caso de ConferenceXP [16] y Adobe Connect [17], incapaces de interoperar con otras aplicaciones distintas a ellas. En ese modelo, el usuario de un determinado programa se encuentra aislado y es incapaz de comunicarse con los usuarios de otras aplicaciones de videoconferencia. En consecuencia, se forman redes aisladas y los usuarios han de emplear varios de estos programas, de manera simultánea, para estar alcanzables desde varias de estas redes.

Sin embargo, en la actualidad, asistimos a movimientos convergentes por parte de las grandes compañías de los productos de voz sobre IP. Para la superación del modelo anterior de redes aisladas, la tendencia es hacia la interoperabilidad. Por ejemplo, una de las empresas líderes en equipos de videoconferencia, Polycom [27], aparte de mantener H.323 [13], ha adoptado SIP en sus terminales. Por otro lado, la primera gran empresa en apostar por la apertura de su red de usuarios a otras redes fue Google, que al escoger como protocolo XMPP [6][7], unió a sus usuarios a la red Jabber [18]. Asimismo, especificó el estándar de las comunicaciones multimedia para dicha red, voz y vídeo, con Jingle [11] y mantiene sus planes para soportar en un futuro SIP [12][19]. Además, aunque aún no se haya materializado, Ebay, propietaria de Skype, y Google acordaron hacer compatibles sus productos. Por su parte, Microsoft y Yahoo comunicaron sus redes de usuarios tras hacer interoperables sus aplicaciones de mensajería y VoIP (voz sobre IP) [20]. Así, la tendencia, aunque lenta, es clara, ya que los usuarios no quieren depender de una única aplicación para sus comunicaciones digitales.

En paralelo a la evolución de las aplicaciones enfocadas al usuario final, y dentro del ámbito de la educación, en el Departamento de Ingeniería de Sistemas Telemáticos (DIT) de la Universidad Politécnica de Madrid (UPM) [21] se ha desarrollado, a lo largo de la última década, un sistema avanzado de videoconferencia llamado Isabel [1][2]. La plataforma Isabel posibilita, desde hace años, la realización de telecongresos, es decir, congresos distribuidos (Telecom I+D, InternetNG, GEANT2) en los que no es necesario reunir en un mismo auditorio a todos los ponentes y audiencias, sino que pueden estar muy distantes físicamente. Del mismo modo, a través de Isabel se practican teleclases en programas de educación a distancia, como CyberAula, y telereuniones en proyectos donde sus componentes no residen en la misma ciudad, como el grupo PROLEARN.

En su estado anterior a los desarrollos presentados, la aplicación Isabel funcionaba sobre Ubuntu GNU/Linux y sólo interaccionaba con otros terminales que también poseyeran Isabel. Ello traía consigo varias limitaciones. En primer lugar, al funcionar de manera exclusiva sobre GNU/Linux obligaba al usuario a tener instalado este sistema operativo, realidad que rara vez se daba. La solución pasaba por la instalación del sistema operativo, lo cual no es trivial. La segunda limitación era que para poder participar en una sesión de Isabel se

S. Pavón y G. Huecas imparten docencia en el Departamento de Ingeniería de Sistemas Telemáticos en la Universidad Politécnica de Madrid, Avda. de la Complutense s/n. Ciudad Universitaria, 28040 Madrid (correos e.: santiago@dit.upm.es; ghuecas@dit.upm.es).

D. Moreno realiza su tesis doctoral dentro del grupo de Internet de Nueva Generación del Departamento de Ingeniería de Sistemas Telemáticos de la Universidad Politécnica de Madrid (correo e.: dmoreno@dit.upm.es)

P. Rodríguez trabaja en el Departamento de Ingeniería de Sistemas Telemáticos con una beca de investigación (correo e.: prodriguez@dit.upm.es)

precisaba la aplicación instalada, ya que no se comunicaba con ningún otro cliente de videoconferencia de los entonces existentes pues Isabel emplea un protocolo propietario para la señalización de la videoconferencia denominado SeCo [5].

II. Objetivos

Con el futuro prometedor que presenta el mundo de las comunicaciones por IP sería un error que cualquier aplicación de videoconferencia, incluida Isabel, se quedase encerrada en sí misma, y sólo permitiese la comunicación entre usuarios de su misma red, existiendo, como ya hemos visto, multitud de usuarios con distintas aplicaciones.

De esta forma, el principal objetivo de nuestra investigación ha sido diseñar la forma en que una aplicación de videoconferencia que usa un protocolo propietario para sus comunicaciones, puede abrirse e interoperar con otras aplicaciones y redes de usuario. El requisito más importante del diseño fue conservar intacto el protocolo de comunicación interno de la aplicación, para no afectar a su funcionamiento. De esta manera, se ha procurado diseñar únicamente añadidos que aporten interoperabilidad sin sustituir ni interferir con la red de usuarios ya existente. Por tanto, la solución tenderá a situarse en la periferia de la red y adoptará la estructura de una pasarela de medios.

Cuando se quiere que una aplicación interaccione con otra, conviene focalizarse en los protocolos que utilizan y no en las aplicaciones en sí. De esta forma, teniendo como punto de referencia los protocolos a la hora de diseñar la compatibilidad, se estará ganando interoperabilidad con muchas aplicaciones al mismo tiempo. Con esta idea y dentro del ámbito de los objetivos de la investigación, se han fijado los protocolos abiertos con los que es más interesante ser compatible, por su grado de utilización. Los protocolos propietarios por su propia naturaleza fueron descartados. De esta forma, se establece como objetivo prioritario desarrollar una pasarela SIP ya que es el protocolo abierto más usado actualmente y en mayor expansión. Por otro lado, se estudió la posibilidad de interoperar con H.323, que a pesar de ser sustituido paulatinamente por SIP, muchos sistemas de videoconferencia por hardware, como los de la empresa Polycom, siguen usando este protocolo y muchos de sus usuarios no están dispuestos a renovar sus equipos. También se contempló el caso de XMPP/Jingle, por ser el futuro estándar de toda la red de usuarios Jabber.

Además, para terminar de abrir las posibilidades de comunicación de la aplicación se aportó interconexión con la red de comunicaciones más extendida del mundo, la Red de Telefonía Básica (RTB) [3]. De esta forma se consigue facilitar de forma definitiva el acceso a la red de usuarios, aunque con limitaciones evidentes de funcionalidad, intrínsecas al canal utilizado.

La arquitectura e ideas surgidas del diseño se implementaron en la aplicación de videoconferencia avanzada Isabel, con muy buenos resultados que validaron el éxito de la investigación.

III. Arquitectura

La primera posible arquitectura software que se plantea para cumplir los objetivos de la investigación es la más sencilla y resulta casi evidente. Se basa en el desarrollo independiente de distintas pasarelas, una por cada protocolo con el que se quiera comunicar la plataforma de videoconferencia. Cada pasarela actuaría como un cliente más dentro de cada una de las redes que quieren interconectar, obteniendo los flujos de señalización y multimedia de ambas redes y procediendo con la traducción de estos.

Esta solución tiene como ventaja la independencia entre pasarelas, gracias a la cual un mal funcionamiento de una no afectará a las demás. Pero aparte del bajo acoplamiento del código esta implementación no tiene más ventajas y sí unos grandes inconvenientes que la hacen desaconsejable. Habría que escribir mucho código, del cual una parte muy pequeña sería reusable de una pasarela a otra. Además, con cada nuevo protocolo con el que se quisiese interoperar habría que iniciar un nuevo desarrollo.

Esta arquitectura clásica es claramente mejorable y sobre la base de investigaciones previas [4] se ideó una segunda solución surgida del conocimiento del proyecto Asterisk [23]. Asterisk es una centralita por software con capacidades de voz sobre IP, que es capaz de conmutar videollamadas desde un protocolo a otro y con una MCU (Unidad de Control Multipunto) integrada. Aprovechando la capacidad de Asterisk, es posible diseñar una arquitectura en la que, con el desarrollo de una sola pasarela a la aplicación de videoconferencia, se diesen cabida a todos los protocolos de señalización soportados por Asterisk. De esta manera, únicamente habría que desarrollar una pasarela a un protocolo que entienda Asterisk, por ejemplo SIP, y se tendría una vía de comunicación abierta con la MCU de la centralita. El siguiente paso sería configurar todas las interfaces de Asterisk que nos interesen (por ejemplo RTB, H.323, SIP y Jingle) para que reenvíen todas las llamadas entrantes a la plataforma de videoconferencia a través de la pasarela SIP. La Fig. 1 muestra la arquitectura de acceso mediante Asterisk.

Este diseño cuenta con un desarrollo más rápido ya que sólo hay que desarrollar una pasarela y configurar adecuadamente Asterisk. Pero la gran ventaja es que cada vez que el proyecto Asterisk de cabida a un nuevo tipo de interfaz, automáticamente será soportado también por la aplicación de videoconferencia. No serán necesarios desarrollos adicionales en la pasarela y simplemente habrá que actualizar a las nuevas versiones de la centralita y realizar las configuraciones oportunas. Esto es así porque Asterisk es un software de código libre y está soportado por una gran comunidad formada por desarrolladores y empresas. Por otro lado, la comunicación con Asterisk se produce a través de un protocolo abierto y estandarizado, SIP, cuya base se sabe que no cambiará previsiblemente, con lo que los cambios que se produzcan en Asterisk no afectarán a la pasarela.

De todas formas, para no cerrar la vía a desarrollos futuros de pasarelas específicas a algún protocolo que no sea soportado por Asterisk, uno de los objetivos principales en el

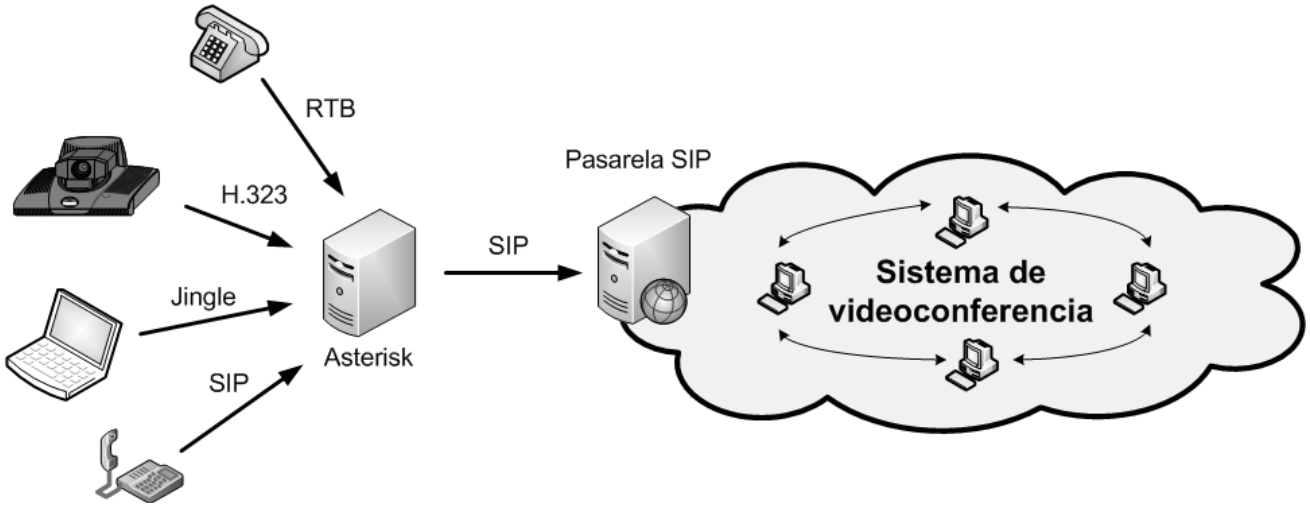

Fig. 1 Escenario con una pasarela SIP y Asterisk

diseño de la pasarela SIP es conseguir un API general de pasarelas. Es decir, conseguir la modularización suficiente para que cuando se quiera implementar otra pasarela baste con sustituir únicamente el módulo que entiende SIP, por otro módulo que entienda otro protocolo. Así, se conserva y aprovecha la mayor parte de la implementación.

En resumen, los dos grandes bloques de la arquitectura son la pasarela SIP y la centralita telefónica por software Asterisk.

A. Pasarela SIP

La pasarela SIP se divide en tres grandes bloques conceptuales: un simulador de clientes nativos de la videoconferencia empotrado dentro del programa de videoconferencia, un API general de pasarelas y un agente SIP como parte más externa de la aplicación. Podemos ver esta división en la Fig. **2**.

El simulador de clientes se encarga de lanzar clientes simulados que se conectan a la videoconferencia como si fueran clientes reales. El sistema de videoconferencia no diferencia los clientes reales de los simulados. Estos clientes, se crean o destruyen según lo pida el módulo adyacente, API General de Pasarelas, y serán los encargados de hablar el protocolo propietario de la videoconferencia.

Este módulo es el más dependiente de la aplicación de videoconferencia a la que estemos aportando interoperabilidad. Por tanto, los detalles de implementación sobre nuestra aplicación ejemplo Isabel, carecen de interés en la presente investigación. Sólo se comentará que no debe realizarse una simulación de todos los componentes que soporta un cliente normal de Isabel, sino que bastaría con los componentes de audio y vídeo, porque lo soportan la mayoría de protocolos con los que se quiere interactuar.

El módulo de simulación debe cubrirse con una capa que haga de interfaz unificado con las posibles pasarelas que se puedan desarrollar, sería el API general de pasarelas. El API debe ser lo más genérico posible para que permita interactuar en primera instancia con un módulo SIP, pero posibilite en un futuro el desarrollo de módulos con soporte para otros protocolos, pero sobre la misma base.

Una vez conseguido el API unificado quedaría desarrollar un módulo que pueda recibir llamadas SIP y gestionarlas: el agente SIP Sería necesario que se registrase en un servidor de registro para que pueda ser localizado a la hora de recibir llamadas, negocie los medios de la sesión (*codecs* o codificadores de audio y vídeo) y los puertos por donde irá la comunicación. Tras analizar las funcionalidades requeridas por este módulo se aprecia que son las mismas que tiene un agente SIP básico. Así, para el desarrollo de este agente SIP,

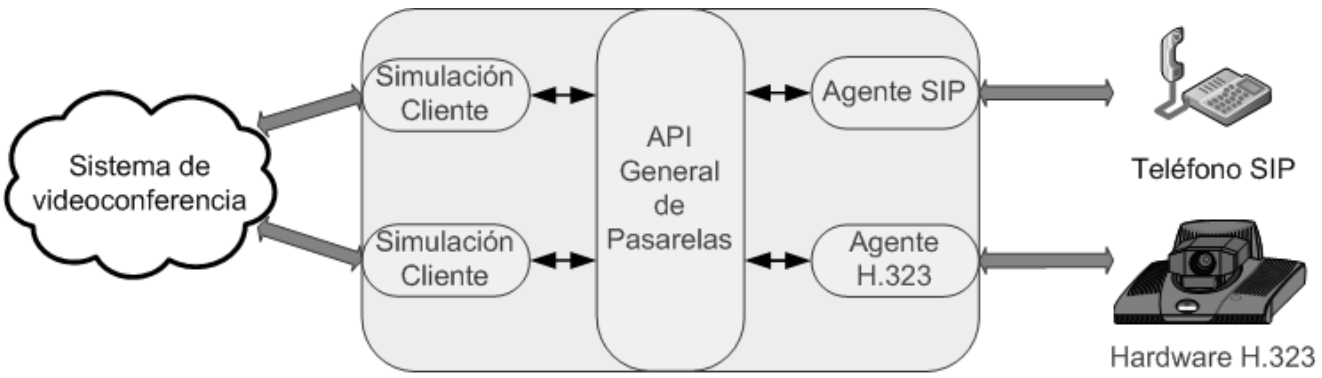

Fig. 2 Arquitectura general de la pasarela SIP

se adoptó, con las modificaciones necesarias, un cliente SIP existente y que se desarrolló en el DIT para el acceso al sistema de conferencias Marte [24].

Por último, cabe destacar una decisión de diseño importante, como es la elección del lenguaje de programación con el que se ha implementado el diseño propuesto. Por un lado, el código que se reutilizará del proyecto Marte está escrito en Java. Este hecho condiciona a que el código del módulo adyacente con el que interopera (API general de pasarelas) esté también en Java para facilitar en gran medida la comunicación e integración entre ambos módulos. En el otro extremo, tenemos el Simulador de clientes, que deberá integrarse a la perfección con módulos ya existentes de la plataforma Isabel. Dichos módulos con los que hay que integrarse están también escritos en Java. Si seguimos el mismo razonamiento y con el objetivo de facilitar al máximo la integración entre módulos se aprecia la conveniencia de escribir el simulador en Java. De esta manera, se concluye que el desarrollo íntegro de la pasarela SIP a Isabel conviene que sea en Java.

A continuación, entraremos en los detalles más interesantes de los dos bloques independientes de la aplicación de videoconferencia sobre la que estemos trabajando: el API General de Pasarelas y el agente SIP.

1) API General de Pasarelas

El API General de Pasarelas es el bloque central del diseño. Tiene el objetivo de ofrecer una interfaz unificada para el desarrollo de distintas pasarelas. Su comunicación con el simulador de clientes depende de la aplicación base, con lo que sus detalles carecen de interés. En cambio, su comunicación con el módulo SIP especificará el API hacia las distintas pasarelas que se puedan desarrollar.

Desde el módulo SIP hacia el API de Pasarelas existen dos llamadas, denominadas *addParticipant* y *removeParticipant*, avisos ambos del momento en que se recibe o termina una llamada desde el módulo SIP.

Desde la aplicación de videoconferencia, a través del API, hacia el módulo SIP hay tres llamadas. La primera, *kickoutParticipant*, echará a un participante de la conferencia. Las otras dos llamadas, *changeAudio* y *changeVideo*, comunican al módulo SIP los audios y vídeos de qué participantes tiene que aportar en cada momento a la conferencia.

2) Agente SIP

El bloque conceptual que representa el agente SIP es la parte más externa de la aplicación, es decir, es la interfaz con la que interactuaran directamente los clientes SIP externos que quieran unirse a una sesión Isabel. En primer lugar, trataremos cómo los clientes SIP son capaces de localizar al agente SIP, y por tanto, a la pasarela. En segundo lugar, analizaremos los componentes del agente SIP y las relaciones entre ellos.

a) Localización

En primer lugar, el agente SIP debe ser capaz de atender una petición de llamada que se origina con el mensaje INVITE proveniente del cliente SIP. Ese mensaje es el inicio de toda llamada SIP. Para que el proxy SIP del cliente llamante sepa llegar a la pasarela SIP, el módulo SIP de la pasarela debe previamente haberse registrado en un servidor de registro con alguna URI (*Uniform Resource Identifier*) [8] que le identifique.

De esta manera, como cualquier cliente SIP, al iniciar su ejecución, la pasarela debe registrarse en un servidor SIP con una cierta URI con la finalidad de ser localizable y alcanzable por otros terminales. Para esta función, durante el desarrollo se ha empleado el servidor SIP denominado SER (SIP Express Router) [25] licenciado bajo la GPL [26]. La dirección de la máquina en la que está el servidor SIP y la URI con la que se registra la pasarela SIP en el servidor, con la que más tarde los clientes SIP tendrán que contactar, es aconsejable que sean dos variables configurables en tiempo de ejecución.

Una vez que la pasarela se ha registrado en el servidor indicado ya es alcanzable a través de su URI. A partir de ese momento le llegarán los mensajes INVITE por parte de los terminales SIP.

b) Componentes

El agente SIP modificado tiene diversos componentes de los cuales el principal es el *Service Manager* que inicia y gestiona las operaciones del resto de componentes. De esta forma, el *Service Manager* es el componente que primero se ejecuta y durante su inicio arranca los componentes foco SIP, MCU y VNCaVideo. Su situación e interacción con el resto de componentes del módulo SIP se aprecia en la Fig. 3.

El foco SIP es el encargado de gestionar la entrada y salida de mensajes SIP. Al recibir la petición de llamada (mensajes INVITE) analiza dicha petición en busca de la información relevante, como es la dirección IP, los puertos a usar y los *codecs* de audio y vídeo que el cliente sugiere utilizar. Dicha información va contenida en los mensajes SDP [9], encapsulados dentro del mensaje INVITE del cliente. Una vez que el foco SIP ha extraído la información se la pasará al *Service Manager*.

El *Service Manager* deberá gestionar la respuesta al mensaje INVITE. Con los datos recibidos del foco conocerá con qué medios cuenta el cliente que quiere iniciar la videoconferencia. Se sabrá en qué puertos espera recibir los flujos RTP [10], tanto de audio como de vídeo, además de con qué *codecs* pueden estar codificados dichos flujos para poder ser comprendidos por el cliente. Para que la comunicación sea exitosa el cliente debe entender flujos RTP codificados con alguno de los *codecs* que soporta el módulo SIP. Esta lista de *codecs* vendrá impuesta por la capacidad de la MCU, como veremos más adelante. El *Service Manager*, sabiendo los *codecs* que soportan el cliente y la MCU, escogerá los más convenientes para establecer la videoconferencia y junto a los puertos a usar los enviará en el mensaje respuesta 200 OK.

La respuesta preparada por el *Service Manager* será enviada al cliente por el foco SIP, el cual mantiene el estado de las transacciones y diálogos.

Una vez terminada la negociación, el *Service Manager* configurará el componente MCU para que sea capaz de recibir los flujos RTP del nuevo cliente. Por otra parte, deberá informar de las nuevas conexiones y desconexiones de

usuarios SIP que se vayan produciendo a través del API General de Pasarelas. Desde el mismo módulo, la plataforma de videoconferencia ordenará al *Service Manager* qué vídeos y qué audios de los usuarios SIP se deben mandar en cada momento a la conferencia. El *Service Manager* recibirá estas órdenes y cambiará el comportamiento de la MCU según corresponda, ya que es la encargada de la gestión de los flujos multimedia de los clientes conectados a la pasarela.

La MCU es un componente con capacidad de gestionar flujos de audio y vídeo. Es capaz de realizar operaciones básicas con flujos RTP como son recibir y enviar flujos a unas direcciones IP y puertos determinados. Además, puede llevar a cabo operaciones mucho más complejas como son la suma de flujos y la recodificación, es decir, recibir un flujo codificado con una cierta norma, decodificarlo y codificarlo de nuevo para su envío siguiendo otra norma distinta.

Cada cliente SIP mandará un flujo RTP de audio y otro de vídeo a los puertos que se negociaron por SDP. En esos puertos, como ya hemos dicho se encontrará a la escucha la MCU, que aceptará los flujos y les dará un trato diferente según sean de audio o vídeo.

El componente VNCaVideo es iniciado por el *Service Manager* en el arranque de la pasarela SIP, junto con el foco SIP y la MCU. Este componente captura el escritorio de la aplicación Isabel, y genera un flujo de vídeo RTP con su contenido.

B. Central telefónica digital Asterisk

La solución propuesta, para conseguir la máxima interoperabilidad con otros protocolos, se basa en una arquitectura donde una máquina con Asterisk se conecta a la videoconferencia a través de la pasarela SIP. Asterisk debe ser instalado y configurado para que acepte llamadas desde SIP, H.323, XMPP/Jabber/Jingle y desde la Red de Telefonía Básica. Una vez recibidas las llamadas desde las distintas interfaces debe encaminarlas a la URI SIP que conecta con la aplicación de videoconferencia.

La interfaz SIP de Asterisk es la única que puede actuar como servidor, es decir, tiene funcionalidad de servidor de registro y MCU recodificadora de flujos. De esta forma, al incluir a Asterisk, podemos prescindir del servidor de registro SER.

En la interfaz H.323, Asterisk actúa como un cliente H.323 normal que necesita registrarse en un *gatekeeper* (servidor de registro de usuarios en terminología H.323), con lo que hay que recurrir a algún programa con esta función como el GNU Gatekeeper [22].

Con el protocolo XMPP/Jabber ocurre algo parecido a H.323, ya que Asterisk vuelve a actuar como cliente y no como servidor, necesitando un servidor Jabber externo. Este hecho no supone una gran dificultad ya que existen muchos servidores Jabber disponibles (Ejabberd, Openfire, OpenIM, etc).

Para la conexión con la Red de Telefonía Básica es necesario hardware especial, consistente en una tarjeta Digium con capacidad para al menos una línea telefónica.

C. Validación con clientes SIP

Una vez pasada la etapa de las pruebas unitarias y con los módulos probados por separado, se llega a la etapa de integración y a las pruebas de sistema, destinadas a validar la arquitectura final.

En estas pruebas ya se puede apreciar la funcionalidad completa de la aplicación. Para esto, se requerirán como mínimo tres ordenadores. El primer ordenador es un terminal Isabel normal, el segundo será el terminal Isabel que realice las funciones de pasarela SIP y un tercero irá ejecutando distintos clientes SIP por software, para comprobar su funcionamiento con el primer cliente Isabel, y con un teléfono hardware SIP.

En las pruebas se configuró la pasarela para que en la negociación SIP sólo admitiese dos *codecs* de audio (PCMU y GSM a 8 Khz). Esta decisión buscó la homogeneidad en las pruebas sin ser una gran limitación debido a que son los

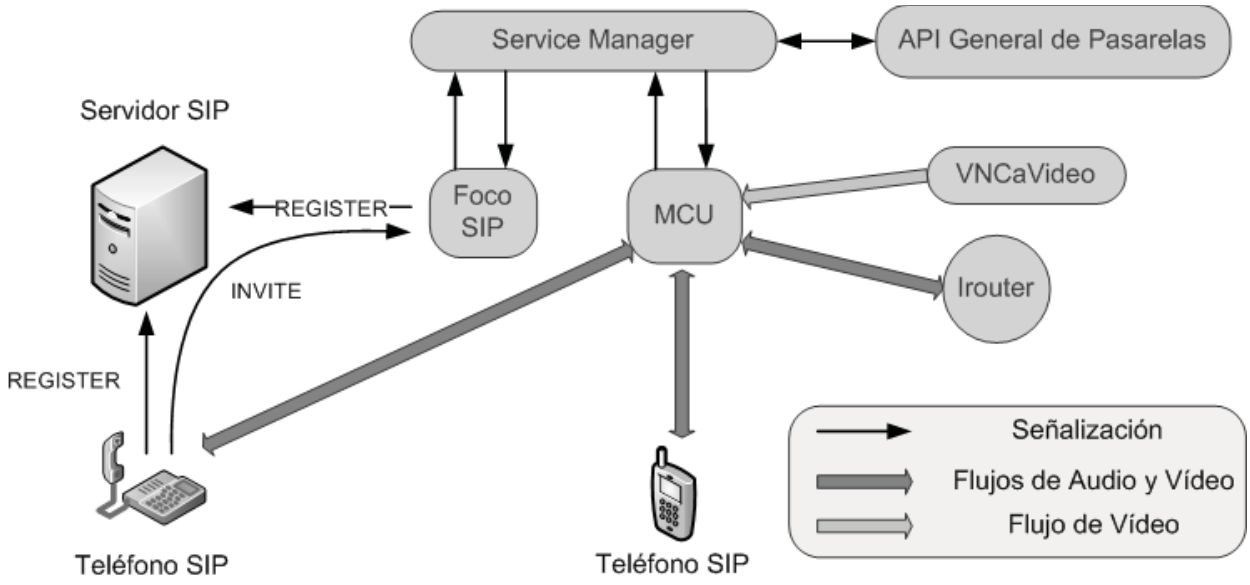

Fig. 3 Bloques del agente SIP

codecs más comunes. De esta forma, se tiene un escenario sencillo y controlado, con muchas posibilidades para las pruebas ya que es posible probar todos los programas telefónicos SIP o *softphones* que se quiera.

Por este escenario pasaron diversos programas, cada uno con sus resultados.

- *Microsoft Windows Messenger:* se comportó bien, funcionando sin problemas tanto el vídeo como el audio. Hubo que usar una versión concreta del programa, ya que a partir de esa versión dejó de soportar el protocolo SIP.
- *Marte 2.0:* también funcionó sin problemas.
- *SIP Communicator:* pese a estar en una etapa temprana de desarrollo y lejos de una versión estable, el audio funcionó.
- *WengoPhone:* esta aplicación utiliza para el vídeo H263+ y se dispuso de una versión alfa del software, lo que le impidió intercambiar flujos de vídeo con la pasarela. El audio funcionó bien.
- *Ekiga:* emplea únicamente el *codec* H.261 para codificar el vídeo, con lo que es imposible el intercambio de vídeo. El audio funcionó sin problemas.

Se probaron muchos más clientes como Gizmo, SJPhone, X-Lite... pero en ellos se encontraron pequeñas divergencias en la implementación del protocolo SIP que los hacían incompatibles.

IV. Conclusiones

Nuestro objetivo ha sido idear una arquitectura software capaz de aportar interoperabilidad con protocolos de comunicación abiertos a una aplicación de videoconferencia que use, internamente, un protocolo propietario. Dicha arquitectura se ha expuesto en el presente artículo y, además, se han facilitado algunos detalles de cómo se implementa en una plataforma real de videoconferencia avanzada llamada Isabel. El hecho de la implementación exitosa de la arquitectura propuesta valida, sin duda, las ideas propuestas.

Así, se han desarrollado las pasarelas oportunas que abran las puertas de Isabel al resto de aplicaciones de videoconferencia. Era objetivo prioritario desarrollar una pasarela SIP, ya que es el protocolo abierto más usado en la actualidad y en mayor expansión. También se tenía la intención de interoperar con H.323, pues es el único protocolo que comprenden muchos equipos antiguos de videoconferencia por hardware. Además, ser compatibles con XMPP/Jingle supone valor añadido al ser el futuro estándar de toda la red de usuarios Jabber.

Todos estos objetivos se han cumplido casi en su totalidad. Se ha desarrollado una pasarela SIP a Isabel que interactúa con clientes SIP, aunque deja algunos fuera que han realizado ciertas variaciones en la implementación del protocolo. Además, se ha diseñado una arquitectura que permite a un ordenador con Asterisk y la pasarela SIP de Isabel acceder a plataformas que se basen en H.323, como las Polycom, y a la red Jabber con su protocolo de VoIP Jingle.

Mediante este desarrollo, se han conseguido eliminar las antiguas limitaciones de Isabel al facilitar la comunicación con otros usuarios que no puedan, o no deseen, instalarse Isabel pero sí quieran participar en alguna de sus sesiones. Así, se ha conseguido incluir en las sesiones Isabel a participantes con independencia de su sistema operativo (GNU/Linux, MacOSX, Microsoft Windows) y plataforma (PC, *hardphone* o teléfono móvil).

V. Trabajos futuros

A la finalización de los trabajos que ha conllevado esta investigación es posible atreverse a sugerir mejoras a los mismos.

La solución escogida al problema planteado, con una pasarela SIP y un Asterisk trabajando conjuntamente, ha sido la óptima teniendo en cuenta la intención de tener un desarrollo lo más rápido posible, y poder disponer pronto de un sistema utilizable. Sin embargo, se podría plantear una arquitectura alternativa, con un Asterisk integrado totalmente en la aplicación de videoconferencia. Esta solución plantea utilizar la MCU de Asterisk integrada con el núcleo de Isabel y nos da una implementación mucho más compacta y un sistema total cerrado. En cambio, tiene como desventaja el aumento excesivo de la complejidad y el acoplo del código. Un diseño intermedio consistiría en desarrollar un canal específico, para que Asterisk entienda el protocolo de control empleado por la videoconferencia.

Ya a nivel funcional, se podría añadir una negociación de calidades en la recepción de los flujos multimedia procedentes de la pasarela, que ahora mismo no existe. Así, usuarios con poco ancho de banda podrían recibir un audio y un vídeo ajustado a sus necesidades y clientes, sin problemas de este tipo, podrían recibir un vídeo de alta calidad.

Otra línea de mejora, sería preparar el sistema completo para producción. Esta etapa incluiría la creación de scripts de instalación y personalización de los archivos de configuración, además del empaquetado para su fácil instalación y puesta en marcha.

Para finalizar con las sugerencias de mejora y complemento de la aplicación, habría que aumentar el número de clientes soportados por los distintos protocolos. Se ha comprobado en las pruebas de la presente investigación los problemas que sufren algunas aplicaciones para interactuar a través de Asterisk y de la pasarela. Habría que centrarse en una lista cerrada de programas, trabajar sobre ellos y analizar qué diferencias realizan en el protocolo.

Referencias

[1] T. de Miguel et al., “ISABEL - Experiment Distributed Cooperative Work Application over Broadband Networks”, Lecture Notes In Computer Science, vol. 868, pp. 353 - 362, 1994.

[2] T. de Miguel et al., “ISABEL: A CSCW Application for the Distribution of Events”, Lecture Notes In Computer Science, vol. 1185, pp. 137 - 153, 1996.

[3] Comisión del Mercado de las Telecomunicaciones, “Informe Anual 2006”, 2007.

[4] D. Moreno, "Desarrollo de pasarelas de acceso de sistemas de comunicaciones al entorno de colaboración Isabel", Proyecto Fin de Carrera de la ETSIT de la UPM, 2007.
[5] J. C. del Valle, "Desarrollo de un servicio de comunicaciones de control no bloqueante multiplataforma. Aplicación a la plataforma Isabel", Proyecto Fin de Carrera de la ETSIT de la UPM, 2007.
[6] P. Saint-Andre, "RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): Core", Request For Comments, IETF, 2004.
[7] P. Saint-Andre, "RFC 3921 - Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", Request For Comments, IETF, 2004.
[8] T. Berners-Lee, R. Fielding, L. Masinter, "RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax", Request For Comments, IETF, 2005.
[9] M. Handley, V. Jacobson, "RFC 2327 - SDP: Session Description Protocol", Request For Comments, IETF, 1998.
[10] H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications", Request For Comments, IETF, 1996.
[11] S. Ludwig et al. "XEP 0166 - Jingle", Proposed Standard, Jabber Software Foundation, 2008.
[12] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "RFC 3261 - SIP: Session Initiation Protocol", Request For Comments, IETF, 2002.
[13] Unión Internacional de las Telecomunicaciones (ITU), "Packet-based multimedia communication systems". Recomendación H.323, 1998.
[14] Página oficial de Skype, http://www.skype.com
[15] Página oficial de Windows Live Messenger http://get.live.com/messenger/overview
[16] Página oficial de ConferenceXP http://research.microsoft.com/conferencexp
[17] Página oficial de Adobe Connect http://www.adobe.com/es/products/connect
[18] Página oficial de la Jabber Software Foundation http://www.jabber.org
[19] Noticia, Google planea soportar SIP en Google Talk http://code.google.com/apis/talk/open_communications.html
[20] Noticia, acuerdo entre Microsoft y Yahoo para unir sus redes de mensajería http://www.microsoft.com/presspass/press/2005/oct05/10-12MSNYahooMessengerPR.mspx
[21] Página oficial de la Universidad Politécnica de Madrid http://www.upm.es
[22] Página oficial de GNU Gatekeeper http://www.gnugk.org
[23] Página oficial del proyecto Asterisk http://www.asterisk.org/
[24] Página oficial de Marte 2.0 http://marte.dit.upm.es
[25] Página oficial de SIP Express Router (SER) http://www.iptel.org/ser
[26] Licencia "GNU General Public License", http://www.gnu.org/copyleft/gpl.html
[27] Página oficial de Polycom http://www.polycom.com