IX Jornadas de Ingeniería Telemática

JITEL 2010

Universidad de Valladolid, del 29 de Septiembre al 1 de Octubre de 2010

Editores:

Yannis Dimitriadis

María Jesús Verdú Pérez

Contenido

Artículos

S1A. Artículos con sello de interés para Telecom I+D (I)

S1B. Artículos con sello de interés para Telecom I+D (II)

Análisis de QoS para una Plataforma Distribuida de Telefonía IP

Jenifer Murillo, José Mª Saldaña, Julián Fernández-Navajas, José Ruiz-Mas, Eduardo Viruete, José I. Aznar
Grupo de Tecnologías de las Comunicaciones (GTC) Instituto de Investigación en Ingeniería de Aragón (I3A)
Dpto. IEC. Edificio Ada Byron. CPS Univ. Zaragoza
50018 Zaragoza, España
e-mail: {jenifer.murillo, jsaldana, navajas, jruiz, eviruete, jiaznar}@unizar.es

***Resumen*—En los últimos años, muchas empresas están cambiando sus soluciones de telefonía tradicionales por otras nuevas que utilizan centralitas *software*, en las que un único PC actúa como nodo central en un sistema completo de telefonía IP, utilizando Internet para realizar llamadas. Las soluciones propietarias suelen utilizar elementos locales que implementan esquemas de control de admisión, pero las centralitas *software* a menudo carecen de mecanismos para proporcionar Calidad de Servicio (QoS, *Quality of Service*) al tráfico de voz. En este artículo se presenta un Control de Admisión de Llamadas (CAC, *Call Admission Control*) para este tipo de sistemas. Para ello se incluye un agente local en cada sucursal de la empresa, que sólo acepta nuevas llamadas si estima para ellas una calidad aceptable. La primera evaluación del sistema ha sido realizada en una plataforma de pruebas basada en virtualización. En un primer paso las llamadas son simuladas, y posteriormente se realizan con tráfico real.**

***Palabras Clave*—CAC; centralita *software*; QoS; virtualización; VoIP**

I. Introducción

En los últimos años, muchas empresas están cambiando sus viejos sistemas de telefonía por otros que utilizan Voz sobre IP (VoIP, *Voice over IP*), que permite realizar llamadas telefónicas utilizando redes de datos. Este cambio ha impulsado la aparición de centralitas *software* que utilizan un simple PC para actuar como nodo central en un sistema completo de telefonía IP, ofreciendo los servicios propios de una centralita. Este PC puede utilizar también algunas tarjetas especiales para conectarse a la RTC. Las soluciones de centralita *software* están popularizándose en Pequeñas y Medianas Empresas (PYMES), para evitar el coste de un sistema propietario.

VoIP es un servicio en tiempo real que tiene que funcionar sobre una red que inicialmente fue diseñada para servicios *best-effort*. Pero los usuarios demandan una Calidad de Servicio (QoS, *Quality of Service*) similar a la de los sistemas telefónicos tradicionales. Esto ha propiciado la búsqueda de soluciones que permitan añadir control de calidad a las redes IP. El sobredimensionado de las redes se usa a menudo para resolver este problema, pero el continuo crecimiento del tráfico y el desarrollo de nuevos servicios hacen que sea una propuesta demasiado simple, y también muy cara. Existen soluciones más inteligentes que pueden ser aplicadas tanto en el plano de datos como en el de control [1]. Entre estas últimas se utiliza mucho el Control de Admisión de Llamadas (CAC, *Call Admission Control*), un sistema que puede rechazar nuevas llamadas que no van a recibir una QoS mínima o que van a disminuir la de las que ya están en curso, evitando así la degradación del servicio de todas las llamadas del sistema.

A menudo, las empresas tienen más de una sucursal y pueden utilizar los sistemas de telefonía IP para evitar el alto coste de las líneas dedicadas. Así, parte del tráfico VoIP ha de ser soportado por una red *best-effort*. Algunos sistemas propietarios de telefonía IP utilizan un elemento en cada sucursal para proporcionar QoS a las comunicaciones. Pero con frecuencia las soluciones de centralita *software* carecen de garantías de calidad para las comunicaciones.

En este artículo se presenta un CAC para un sistema de telefonía IP basado en una centralita *software* (Fig. 1). Se ha añadido un agente local en cada sucursal para interceptar los mensajes de señalización y, dependiendo de la decisión de admisión, reenviar el mensaje al destino o enviar un mensaje de no disponibilidad a la centralita. Así, el CAC se integra con facilidad en el sistema de telefonía, sin que la centralita ni los terminales VoIP necesiten modificación alguna. La introducción de un nuevo agente en cada sucursal puede verse como una desventaja, pero como veremos, es un elemento simple que no requiere gran capacidad de procesado, así que puede ser fácilmente incluido como un proceso dentro de un servidor ya existente. De todos modos, se estudiará el agente local como un elemento diferenciado.

Antes de implementar el sistema en entornos reales, es muy recomendable validar primero la solución en un entorno controlado, como una plataforma de pruebas, donde pueda ser instalado y evaluado. Las plataformas basadas en virtualización permiten desplegar un escenario de red completo en una única máquina física. Además, de este modo las aplicaciones multimedia y las pilas de protocolos pueden ser implementaciones reales. Por otra parte se pueden ahorrar costes, ya que es posible ejecutar muchas máquinas virtuales dentro de una única máquina física. La técnica de virtualización se ha utilizado para construir diversas plataformas: algunas son grandes infraestructuras como *PlanetLab* [2], mientras que otras son pequeñas [3], [4]. El simulador de red *ns-3* [5] permite integrar máquinas virtuales, que pueden ejecutarse sobre sus dispositivos y canales.

La organización del artículo es la siguiente: la sección II habla sobre trabajos relacionados y el uso del control de admisión. La arquitectura del sistema se presenta en la sección III. La siguiente sección se centra en la plataforma de pruebas. La sección V presenta los resultados obtenidos. La última sección detalla las conclusiones del presente trabajo.

II. Trabajos relacionados

Existe una amplia variedad de protocolos que se utilizan para la señalización en sistemas de VoIP. H.323, IAX (*Inter-*

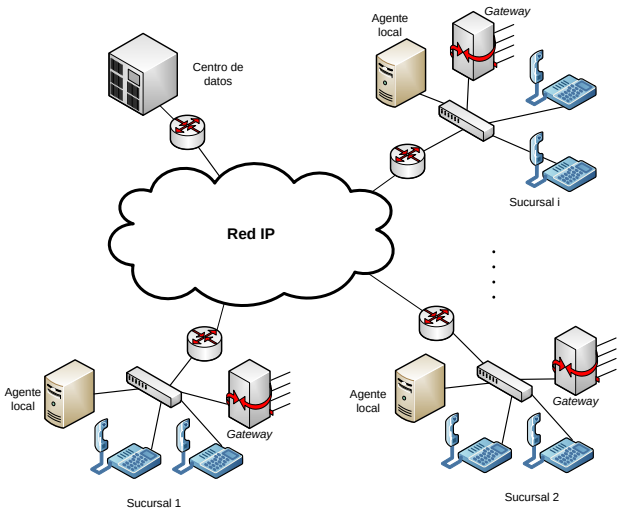

Fig. 1. Sistema de telefonía IP

Asterisk eXchange), MGCP (*Media Gateway Control Protocol*) y SIP (*Session Initiation Protocol*) son algunos de los más utilizados. El análisis realizado en este artículo está basado en SIP, ya que es un protocolo abierto y muy utilizado en redes IP. SIP también ha sido adoptado por el 3GPP como el protocolo de señalización para IMS (*IP Multimedia Subsystem*) [6]. Otra ventaja de SIP es que no sólo puede utilizarse para administrar sesiones VoIP, sino que también puede usarse para otros servicios. Existen centralitas *software* de código abierto y sistemas comerciales de VoIP que utilizan SIP [7] y permiten que el sistema CAC pueda ser fácilmente integrado en ellos.

La recomendación RFC 3261 para SIP incluye el concepto de *proxy* SIP, un elemento que puede concentrar o redirigir tráfico, añadiendo escalabilidad al sistema, ya que transfiere carga computacional del núcleo de la red a los extremos. Entre las soluciones de centralita *software* destaca Asterisk, desarrollada por Digium. Actúa como *back to back user agent*, realizando dos comunicaciones, una desde el origen hasta la centralita y otra desde la centralita hasta el destino y uniendo las dos. Asterisk soporta SIP y otros protocolos de señalización.

Pero actualmente no existe una solución completa para añadir QoS a una centralita *software* [8]. En redes IP se puede utilizar DiffServ (*Differentiated Services*), que modifica el campo ToS (Tipo de Servicio, *Type of Service*) de la cabecera IP para clasificar el tráfico. Pero no existe un acuerdo entre los diferentes proveedores de redes, aplicaciones, etc. para unificar dicha clasificación. De este modo el envío extremo a extremo no tiene garantías de QoS. Otra opción es el uso de IntServ (*Integrated Services*), que reserva recursos para los flujos de datos. Esto conlleva problemas de escalabilidad y requiere que todos los *router* intermedios implementen este protocolo, algo que no es posible en la práctica en muchos casos.

La existencia de un elemento local en cada sucursal es importante porque la centralita no dispone de información acerca del tráfico de cada sucursal. Algunas soluciones propietarias añaden un elemento en cada sucursal, por ejemplo las soluciones de CISCO utilizan un elemento denominado *Call-Manager* que interopera con el *Gatekeeper* que se encuentra en el nodo central [9].

En los últimos años se han estudiado y desarrollado muchos sistemas de control de admisión. En [10] se puede encontrar un estudio sobre diferentes métodos, incluyendo comparativas y clasificaciones. Las soluciones basadas en el almacenamiento de información sobre los recursos reservados en la red presentan problemas de escalabilidad y dificultades de implementación, mientras que las soluciones basadas en medidas de las condiciones de la red son más adecuadas para adaptarse al estado de la red.

Algunos sistemas CAC utilizan el modo basado en parámetros. Para ello, es necesario realizar algunas medidas durante la configuración inicial del sistema. El tráfico interferente debe ser medido para obtener el número máximo de llamadas que podrían establecerse simultáneamente. Los sistemas basados en medidas realizan estimaciones cuando una llamada va a establecerse, pero su desventaja radica en que añaden más retardo, ya que normalmente se implementa una fase de prueba previa al establecimiento de la llamada; las medidas activas añaden sobrecarga al sistema y presentan una alta dependencia de la congestión instantánea de la red.

En [11] se presentó un sistema CAC basado en parámetros. En él se añade QoS en entornos CISCO, siendo H.323 el protocolo inicial, y se incluye un *Integration Component* para interceptar la señalización. Como nuestro sistema está basado en SIP, el uso de *proxy* SIP evita este elemento, interceptando la señalización de las llamadas de una forma natural.

La principal novedad del presente estudio es que el sistema utiliza un protocolo abierto como SIP, y *software* libre para el elemento central y también para los agentes que están a cargo del CAC en cada sucursal. Estas características permitirán el estudio y las medidas, ya que la documentación y el código fuente están disponibles. El sistema evaluado es similar a los esquemas propietarios, pero con muchos parámetros que podrán ser modificados y estudiados para obtener resultados comparativos: diferentes *codec*, esquemas de CAC, influencia de los *buffer* de los *router*, etc.

Nuestro sistema también puede utilizarse para reducir costes en empresas multinacionales, ya que podemos conseguir que las llamadas internacionales tengan un coste local. Para ello, como muestra la Fig. 2, en lugar de realizar estas llamadas directamente desde el *gateway* local, se divide la llamada en dos tramos: el primero utiliza VoIP, vía Internet, hasta la sucursal destino, y un segundo tramo llega hasta el terminal destino con el coste de una llamada local.

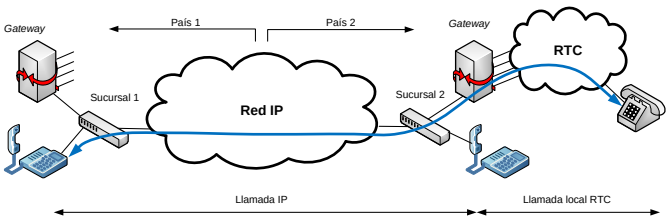

Fig. 2. Llamada internacional

III. Arquitectura del sistema

A. Descripción general del escenario

Como hemos dicho, el sistema de telefonía IP se ha diseñado para una empresa con varias sucursales en distintas localizaciones de diferentes países. Para reducir costes de administración, es deseable tener el plan de numeración sólo en la centralita, y no distribuido en las sucursales. Además, se utiliza Internet para enviar el tráfico telefónico entre las sucursales, en lugar de utilizar líneas dedicadas. Hemos asumido que el sistema no utiliza ningún protocolo de reserva de recursos, y podemos asumir también que el tráfico VoIP es el único tráfico en tiempo real que se trata de forma especial.

En este escenario se utiliza un sistema CAC basado en parámetros, en el que se realizan varias medidas de tráfico al inicio de la configuración, para asignar un número máximo posible de llamadas simultáneas en cada sucursal. El principal objetivo es asegurar un valor mínimo de QoS para las llamadas, al coste de tener que rechazar algunas de ellas.

Se ha introducido un elemento para implementar el sistema CAC en cada sucursal: un agente local. Su arquitectura puede verse en la Fig. 3. Este elemento juega un papel fundamental en la elección de la mejor ruta para realizar las llamadas telefónicas, en base a las medidas de QoS y a las tarifas. A continuación se explican los elementos que lo componen:

1) Un *proxy* SIP procesa la señalización de las llamadas telefónicas para implementar el mecanismo del CAC, basado en información almacenada en la base de datos. La decisión de CAC se basa en el estado actual de las sesiones VoIP, teniendo en cuenta las tarifas y las líneas disponibles en los *gateway*. El *proxy* SIP no requiere una gran capacidad de procesado. En [12] se realizaron algunas medidas del comportamiento del *proxy*, comprobando que un único *proxy* puede administrar 2.000 mensajes SIP por segundo. El tráfico que estamos considerando en este artículo es mucho menor.
2) El módulo contador se encarga del control de líneas libres y ocupadas en el *gateway*. También realiza un conteo del número de llamadas telefónicas cursadas en la sucursal en cada momento. Teniendo en cuenta esta información, las tarifas telefónicas y el número de líneas disponibles en el *gateway*, el agente local rellena la tabla de decisiones en la que se basa el sistema CAC.

B. Funcionamiento del sistema CAC

La señalización SIP se envía desde el teléfono IP al agente local, y posteriormente a la centralita. En ella, de acuerdo al plan de numeración, se establece otra llamada hasta el terminal destino, y las dos llamadas se unen. La comunicación de voz transmite tráfico RTP, y se ha configurado el sistema para que este tráfico se envíe directamente entre los teléfonos, y no a través de la centralita.

Como toda la señalización pasa a través del agente local, éste puede tomar decisiones sobre futuras peticiones de establecimiento y llevar la cuenta del número de líneas ocupadas en el *gateway*. En caso de que no haya que rechazar la llamada, el agente local únicamente retransmite los mensajes de señalización. Las llamadas internas a la sucursal son administradas directamente por el agente local, sin necesidad de intervención de la centralita, de modo que no están afectadas

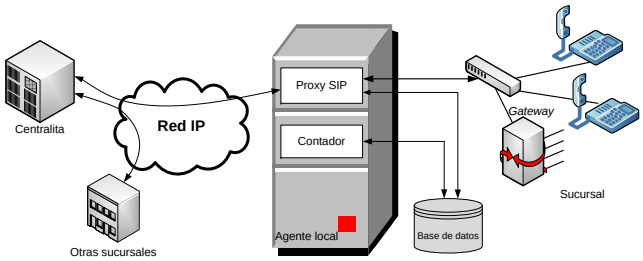

Fig. 3. Arquitectura del agente local

por el sistema CAC. Las llamadas entre terminales de distintas sucursales pueden ser rechazadas por el CAC, por falta de QoS en la ruta entre dichas sucursales o por la indisponibilidad del terminal destino. Cuando se rechaza una llamada, se envía un mensaje SIP a la centralita del tipo *480 Temporarily Unavailable*.

Las llamadas cuyo destino es la RTC que se realizan a través del *gateway* pueden ser redirigidas si no existen líneas disponibles. En este caso el agente actúa como *redirect server*, redirigiendo la llamada a otra sucursal (si es posible del mismo país) que tenga líneas disponibles para establecer la comunicación. El *redirect server* envía un mensaje 3XX informando sobre una ruta alternativa (Fig. 4). El agente que actúa como *redirect server* ya no tomará parte en esta llamada. Para evitar bucles en la señalización hay que elegir un valor adecuado en el campo *MAX_Forwards* de la cabecera.

Veamos un ejemplo simplificado para ilustrar el ahorro en costes y el incremento del grado de servicio de las llamadas a la RTC al utilizar redirecciones. Los teléfonos de cada sucursal generan tráfico hacia la RTC (AP_i) y hacia la otra sucursal a través de la red IP (AI_{ij}). El tráfico de desbordamiento (AO_i) representa las llamadas que no pueden ser cursadas por el *gateway* local. N_i representa el número de líneas del *gateway*, y M_i el máximo número de llamadas que pueden ser aceptadas por el sistema CAC. Para ilustrar las ventajas de compartir los *gateway* se muestra un ejemplo con dos sucursales (Fig. 5). Sea γ(A,N) la probabilidad de bloqueo del sistema con tráfico A y N servidores, P_{bRTC} y P_{bIP} las probabilidades de bloqueo de los *gateway* y del sistema CAC respectivamente. En caso de no compartir los *gateway*, la probabilidad de bloqueo en la sucursal 1 será:

$$P_{bRTC} = \gamma(AP_1, N_1) \tag{1}$$

$$P_{bIP} = \gamma(AI_{12} + AI_{21}, M_1) \tag{2}$$

Si los dos *gateway* se comparten entre las dos sucursales, el número de circuitos que se pueden utilizar para la conexión a la RTC crece a N_1+N_2. Así que las probabilidades de bloqueo que se obtienen son:

$$P_{bRTC} = \gamma(AP_1 + AP_2, N_1 + N_2) \tag{3}$$

$$P_{bIP} = \gamma(AI_{12} + AI_{21} + AO_1 + AO_2, M_1) \tag{4}$$

Podemos hacer una simplificación llamada *Erlang Fixed Point* que se usa para redes grandes con poca probabilidad de pérdidas. Asume que todos los tráficos tienen distribuciones

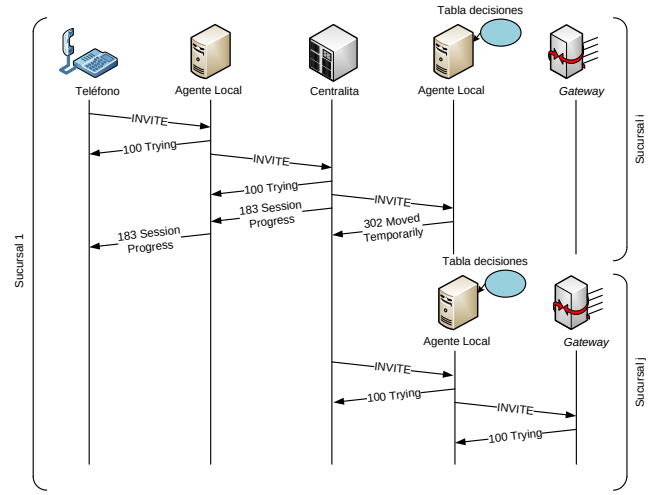

Fig. 4. Llamada redirigida por el sistema CAC

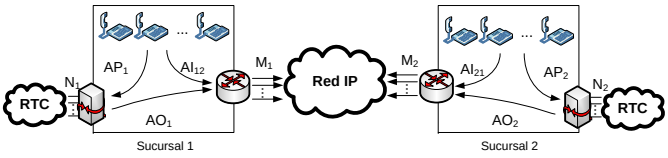

Fig. 5. Líneas de *gateway* compartidas

de probabilidad de Poisson. Así que la fórmula Erlang B (denotada como Er_b) podría utilizarse como la función γ para calcular las probabilidades de bloqueo. Utilizando esta simplificación, la probabilidad de bloqueo de los *gateway* mejora, ya que (3) será menor que (1), si suponemos que las dos sucursales tienen cantidades de tráfico y número de líneas similares. Por otra parte, (4) será mayor que (2), ya que el número de llamadas utilizadas por la red IP crecerá. Como hemos definido un número máximo de llamadas en nuestro sistema, éste rechazará llamadas en más casos. Lógicamente, la simplificación asumida podría ser muy severa, de manera que se necesitarían más cálculos y simulaciones para obtener de manera correcta γ para todos los casos. Hemos utilizado esta simplificación para ilustrar las ventajas de compartir recursos entre diferentes sucursales.

Dejando a un lado el ejemplo, se puede concluir que nuestro sistema permite intercambiar la probabilidad de bloqueo entre los *gateway* y la red IP. Pero si no se desea incrementar la probabilidad de bloqueo, existen dos soluciones: reducir el tráfico interferente o incrementar el ancho de banda, que será probablemente más económico que incrementar el número de líneas del *gateway*.

IV. Plataforma de pruebas

En esta sección se explica resumidamente la plataforma de pruebas [13] en la que se ha implementado el sistema. Se ha buscado un diseño adecuado para el sistema de telefonía IP, que emule condiciones reales y permita pruebas y medidas con flexibilidad.

A. Simulación, entorno real o emulación virtual

A la hora de construir una plataforma de pruebas existen varias opciones. Las herramientas de simulación son una de ellas, y de hecho, ya han sido utilizadas para el estudio de otros sistemas CAC [14], [15]. Su ventaja es que evitan las limitaciones electrónicas reales de los dispositivos, pero su inconveniente radica en que los protocolos tienen unas implementaciones por defecto que no tienen por qué coincidir con las implementaciones reales, de modo que sería necesario desarrollarlas para el simulador. Algunas herramientas disponibles son OPNET, OMNET++ y ns-2.

Por otra parte, la plataforma de pruebas podría implementarse también con máquinas reales. Sin embargo, el coste *hardware* sería elevado debido al gran número de elementos que forman el escenario. La virtualización consiste en ejecutar varias máquinas, cada una con su propio sistema operativo, sobre el *hardware* real de una máquina física. Algunos estudios [16] han utilizado la virtualización para minimizar costes y optimizar el control de la plataforma. Por ejemplo, *User Mode Linux* (UML) se ha utilizado en implementaciones de algunos emuladores, como vBET [17]. Los nodos virtuales se conectan mediante una red emulada que se ejecuta bajo el *driver* de la tarjeta de red. Este concepto encaja con el entorno de pruebas que queremos utilizar, permitiendo usar aplicaciones comerciales para la centralita *software*, *proxy* SIP y *softphone* que proporcionan realismo al escenario.

En lo que se refiere a la escalabilidad, la configuración de la red permite ampliar la plataforma con más máquinas físicas, en caso de que aumente la necesidad de cálculo o si existen más nodos. Además, se garantiza la repetibilidad de las pruebas, ya que utilizando una única máquina física se consigue un entorno aislado y controlable.

B. Selección de la tecnología de virtualización

Considerando todos los esquemas de virtualización existentes, se ha seleccionado la paravirtualización. La ventaja que proporciona es que la velocidad de ejecución conseguida es cercana a la de los esquemas de no virtualización. El único problema que presenta es que requiere modificaciones dentro del sistema operativo del cliente para evitar que ninguna instrucción tenga que ser ejecutada con privilegios, aunque esto no supone un inconveniente en nuestro caso, puesto que al usar Linux el problema queda reducido a una nueva compilación.

La solución escogida ha sido Xen, ya que algunos estudios comparativos de plataformas de virtualización [18] muestran que es una herramienta apropiada en términos de *overhead*, linealidad y aislamiento entre las máquinas virtuales. El rendimiento de las comunicaciones medido para un escenario compuesto por 10 máquinas virtuales fue de 93 MB/s entre pares.

Estas características son muy interesantes para el rendimiento de la plataforma, ya que se desea tener un entorno controlable en el que todas las máquinas virtuales compartan los recursos equitativamente.

C. Emulación de la red

Como el sistema se ha emulado utilizando máquinas virtuales, hemos utilizado el comando de Linux *brctl* para construir *bridge* que conecten las máquinas. Hemos creado dos redes: una de control y otra de pruebas. La red de control utiliza la interfaz *eth0* de cada máquina virtual. Como se puede ver en la Fig. 6, todos los interfaces de control están conectados al *bridge xenbr0*. Esta red se utiliza para controlar,

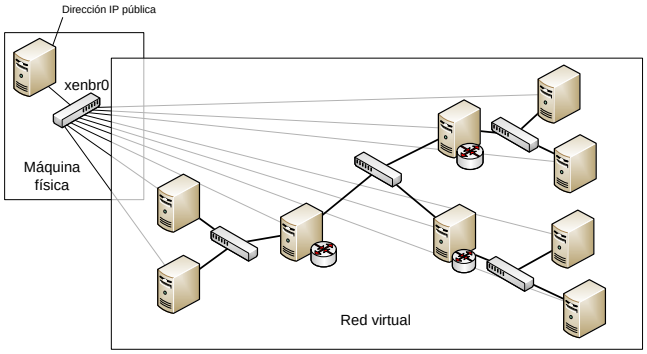

Fig. 6. Máquina física y red virtual de la plataforma

medir y configurar el sistema utilizando el protocolo SSH, evitando interferir con la red de pruebas. La máquina principal tiene una dirección IP pública que permite controlarla de forma remota. La red emulada se implementa utilizando las interfaces de red *eth1* y *eth2*.

El problema del uso de *bridge* emulados es que trabajan a velocidad de procesador, y carecen de los retardos y anchos de banda que tienen habitualmente las redes. Así que se ha utilizado la herramienta de Linux *Traffic Control (tc)* para añadir limitaciones de ancho de banda en los enlaces de acceso a cada sucursal, y la herramienta *Netem* [19] para añadir retardos en la red. *tc* tiene en cuenta las cabeceras de nivel 2, así que el límite del ancho de banda incluye en este caso los bytes de la cabecera ethernet.

Para realizar la sincronización del sistema se ha utilizado NTP, de manera que una de las máquinas virtuales tiene instalado un servidor NTP y en las demás se ha utilizado la opción de Xen *independent_wallclock*. De esta manera se ha conseguido que la sincronización sea muy precisa (aproximadamente 1 ms), permitiendo la obtención del *One Way Delay* (OWD, retardo extremo a extremo en un sentido) y del *Round Trip Time* (RTT).

D. Características de la máquina física

La máquina en la que se ha implementado la plataforma utiliza el Sistema Operativo CentOS 5. La versión del núcleo es la 2.6.18-8.1.15. Dispone de un procesador Core 2 Duo a 2.40 GHz, 2MB de Cache nivel 2, y 4GB de memoria RAM. Las máquinas virtuales también trabajan con CentOS 5. La versión de Xen es la 3.03-25.0.4.

Durante las pruebas se ha monitorizado la utilización de la CPU, para evitar la influencia de la carga del procesador en las medidas. La utilización nunca ha excedido el 10%.

E. Herramientas software

En esta sección se van a explicar las herramientas *software* utilizadas para implementar el sistema CAC en la plataforma. Para ahorrar carga computacional, sólo se han utilizado herramientas de línea de comandos.

En primer lugar, se requiere un *proxy* SIP instalado en el agente local. La herramienta seleccionada debe incluir la opción *redirect server* y la posibilidad de ser adaptable para que las decisiones de CAC puedan ser implementadas. También debe ser capaz de acceder a información externa situada en la base de datos. Se ha seleccionado OpenSIPS 1.4, un proyecto derivado de OpenSER, ya que proporciona funcionalidades de *register server, location server, proxy server y redirect server*. La baja carga computacional y la posibilidad de añadir y eliminar funcionalidades de forma modular son también características interesantes. La configuración del *proxy* SIP se realiza con un lenguaje de programación de alto nivel. También se dispone de acceso a bases de datos MySQL.

La centralita utilizada es Asterisk 1.6, una solución interesante por su flexibilidad, actualizaciones y licencia de distribución GNU-GPL. Por último, se eligió el *softphone* de línea de comandos PJSUA 1.0, también utilizado para emular los *gateway*, ya que no se usan conexiones reales a la RTC .

V. Pruebas y resultados

A. Automatización de realizaciones

Se ha realizado una batería de pruebas para caracterizar el comportamiento del sistema. Se ha construido un escenario que contiene un centro de datos y cuatro sucursales, cada una con una conexión de subida de 1 Mbps, conectadas con una red IP. Para emular la Hora Cargada, en primer lugar se ha generado *offline* con Matlab una hora de llamadas, haciendo uso de distribuciones de probabilidad para generar los instantes de las llamadas y su duración.

Posteriormente se emula dicha realización utilizando una herramienta de automatización de aplicaciones llamada *Expect*, aprovechando que el *softphone* usado es de línea de comandos. Como se utiliza emulación, el tiempo de la ejecución es tiempo real, así que se requiere una hora para cada ejecución. Así, el tráfico del sistema es real, no simulado. El tráfico SIP se genera en los *softphone* PJSUA, mientras que el tráfico RTP se genera con D-ITG [20], por razones prácticas. Durante la ejecución se intenta evitar cualquier cálculo innecesario. Los ficheros *log* del tráfico RTP y parte de la señalización SIP son almacenados en la base de datos, de manera que al finalizar la realización se puedan procesar *offline* para obtener resultados. El procesado del tráfico RTP se realiza con la aplicación ITG-Dec del generador D-ITG, que calcula el OWD, las pérdidas y otros parámetros.

El escenario utilizado para las medidas se puede ver en la Fig. 7. El tráfico interferente tiene la siguiente distribución: el 50% de los paquetes son de 40 bytes, el 10% de 576 bytes y el 40% de 1.500 bytes, todos ellos a nivel IP [21]. Este tráfico se ha generado para saturar el enlace de acceso de cada sucursal. Se ha utilizado UDP en lugar de TCP, para evitar el control de flujo que TCP realiza por defecto. Así, el tráfico interferente siempre es el mismo, y no se adapta al ancho de banda disponible, haciendo que el sistema trabaje siempre en el peor caso.

El tráfico SIP se dirige en primer lugar al *proxy*, después a la centralita y por último al *proxy* destino. El tráfico RTP se envía directamente entre los *softphone*. Cada flujo RTP tiene un ancho de banda de 24 kbps a nivel IP, o 29,6 kbps a nivel ethernet. Se ha utilizado el *codec* G.729, con dos muestras por paquete, que implica un retardo de paquetización de 25 ms, incluyendo un *look-ahead* de 5 ms.

Como se ha dicho, los *router* de cada sucursal tienen una conexión en el enlace de subida de 1 Mbps. Esto se ha conseguido gracias a la herramienta *tc*, utilizando una cola *token bucket* que limita el ancho de banda a nivel 2. El

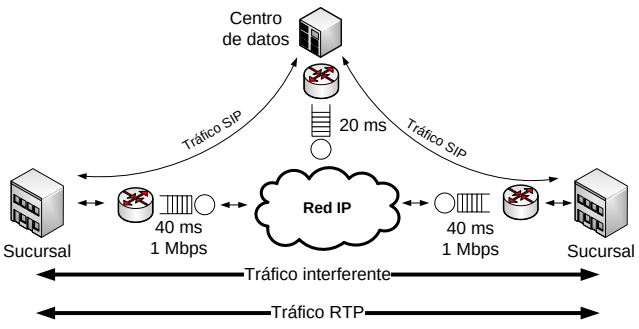

Fig. 7. Esquema de medidas

retardo de los paquetes RTP no debería superar el valor de 150 ms, como recomienda la ITU [22]. Como la red IP emulada introduce un retardo medio de 40 ms, el tamaño del *buffer* se ha calculado para que los paquetes permanezcan encolados un tiempo máximo de 50 ms. En el centro de datos se ha introducido un retardo medio de 20 ms.

B. Resultados

En primer lugar se ha medido el retardo de establecimiento de las llamadas para distintos valores de tráfico interferente. Los resultados se pueden ver en la Tabla V-B. Se compara la situación en la que los *softphone* están directamente registrados en la centralita (sin CAC), con la del comportamiento del sistema CAC, utilizando los *proxy*. Se observa un retardo adicional cuando se introduce el sistema CAC, debido a la presencia de los *proxy* como elementos intermedios, aunque los valores obtenidos no son excesivos y se pueden asumir. Si los enlaces están saturados, el retardo adicional es prácticamente igual para ambos casos, ya que depende mucho del retardo de encolado.

También se han realizado pruebas para obtener los límites de comportamiento del sistema. La existencia de las colas juega un papel muy importante en dicho comportamiento. En [23], los autores sugieren el uso de *buffer* pequeños. En este trabajo hemos supuesto que se puede limitar el tamaño de las colas de los *router*. En la Fig. 8 se observa que cuando el tráfico se acerca al límite del ancho de banda (1 Mbps), los paquetes comienzan a ser descartados, siendo los de mayor tamaño los que se descartan en mayor medida. Esto empieza a ocurrir cuando el número de flujos RTP es 7, ya que tenemos 800 kbps de tráfico interferente y más de 200 kbps de tráfico RTP. La Fig. 9 muestra que los paquetes pequeños mantienen su ancho de banda, mientras que los paquetes grandes son descartados en mayor número, ya que permanecen demasiado tiempo encolados. El hecho de que el tráfico total supere ligeramente el valor de 1 Mbps se debe a la tolerancia de la herramienta *tc*.

Este resultado muestra que el tráfico RTP está protegido frente al descarte debido a su pequeño tamaño. Así que limitar el número máximo de llamadas simultáneas es también importante para evitar la degradación del resto de tráfico de la sucursal. Esto significa que podemos utilizar el sistema CAC no sólo para proteger el tráfico RTP frente al descarte, sino también para evitar la degradación del tráfico interferente.

Las Fig. 10, 11 y 12 muestran el comportamiento del sistema en términos de retardo medio, paquetes perdidos y

Tráfico interferente	850 kbps	900 kbps	950 kbps	1000 kbps
No CAC	83 ms	87 ms	87 ms	199 ms
CAC	175 ms	183 ms	184 ms	205 ms

Tabla I
RETARDO DE ESTABLECIMIENTO DE LAS LLAMADAS

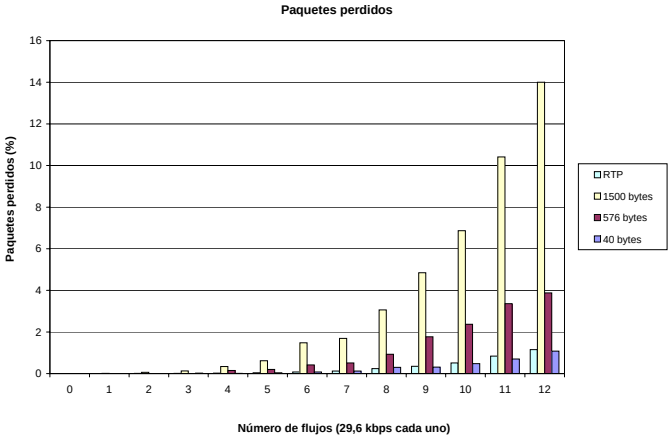

Fig. 8. Paquetes perdidos en el sistema con 800 kbps de tráfico interferente

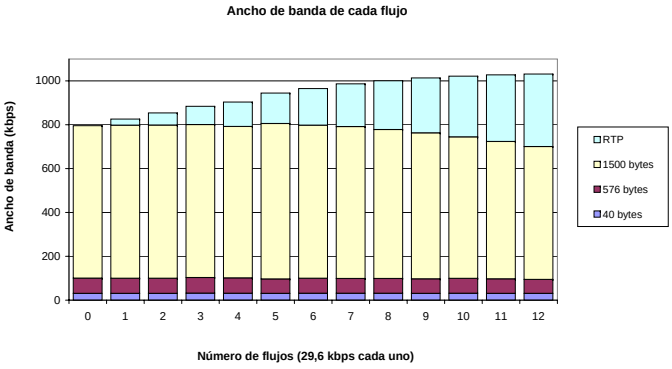

Fig. 9. Anchos de banda en el sistema con 800 kbps de tráfico interferente

jitter. Estas curvas muestran el límite superior para estos parámetros en el caso de tener el máximo número de llamadas posibles con el sistema CAC. Los valores se representan en función del tráfico interferente, de manera que cada gráfica tiene un momento diferente en el cual el tráfico total supera el límite del ancho de banda, y comienza la degradación del sistema. Si representamos estos parámetros en función del tráfico total, las gráficas se acercan entre sí (Fig. 14, 15 y 16). Las Fig. 13 y 17 muestran el comportamiento en términos del MOS. Estos valores se calculan según la recomendación G.107 [24], partiendo de la Relación Señal a Ruido (SNR, *Signal to Noise Ratio*) y teniendo en cuenta diferentes factores que reducen la calidad de la señal. Para obtener estos valores hemos utilizado una aplicación desarrollada por la ITU [25].

En la Fig. 12 observamos el comportamiento del sistema en términos del *jitter*. Para medirlo se ha utilizado el IPDV (*Instantaneous Packet Delay Variation*). Presenta un máximo, y decrece según el acceso se va saturando. La razón es que los paquetes grandes comienzan a ser descartados en mayor porcentaje, y estos paquetes son la principal causa del *jitter*. Los valores de *jitter* se mantienen por debajo de 12 ms.

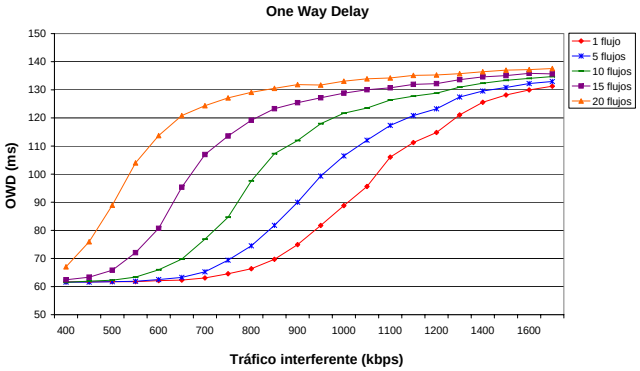

Fig. 10. *One Way Delay* en función del tráfico interferente

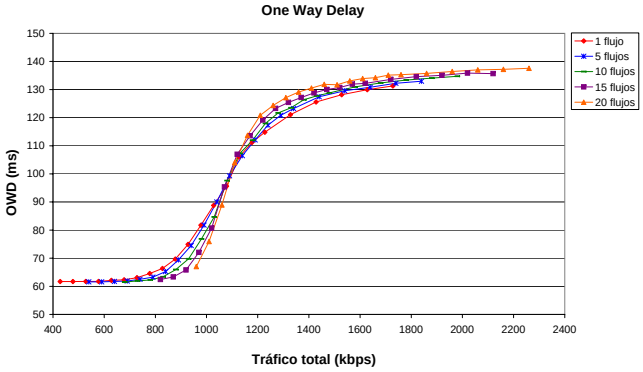

Fig. 14. *One Way Delay* en función del tráfico total

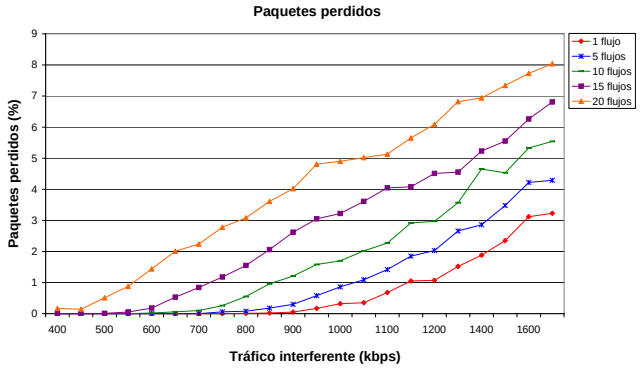

Fig. 11. Paquetes perdidos en función del tráfico interferente

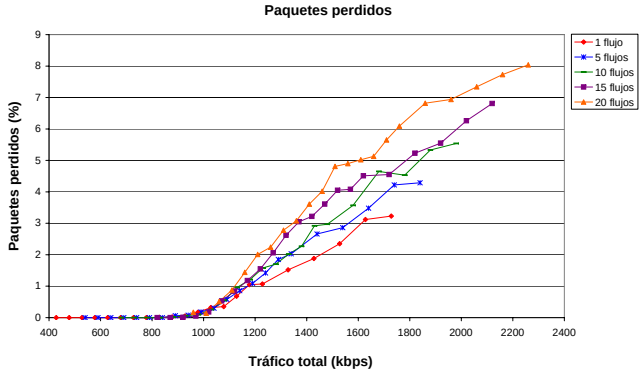

Fig. 15. Paquetes perdidos en función del tráfico total

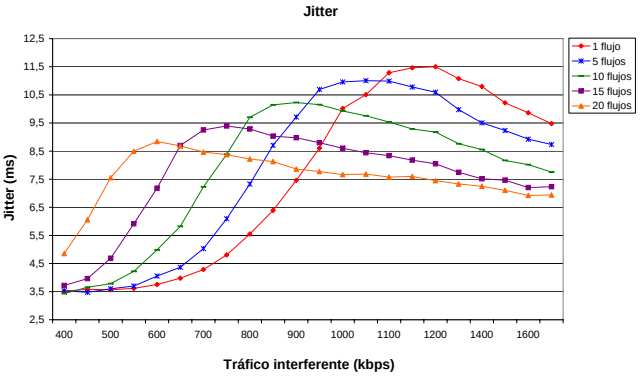

Fig. 12. *Jitter* en función del tráfico interferente

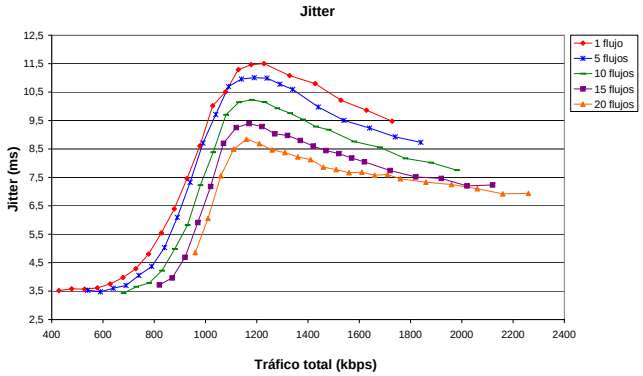

Fig. 16. *Jitter* en función del tráfico total

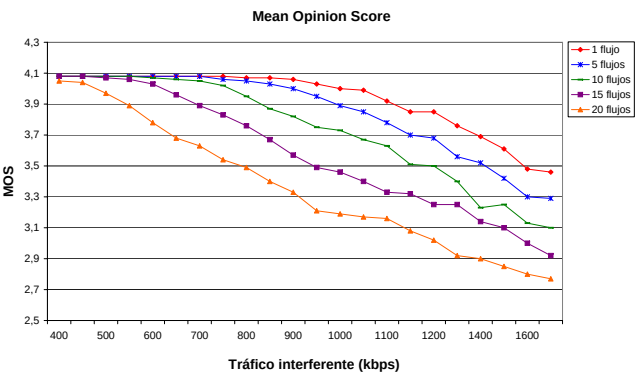

Fig. 13. MOS en función del tráfico interferente

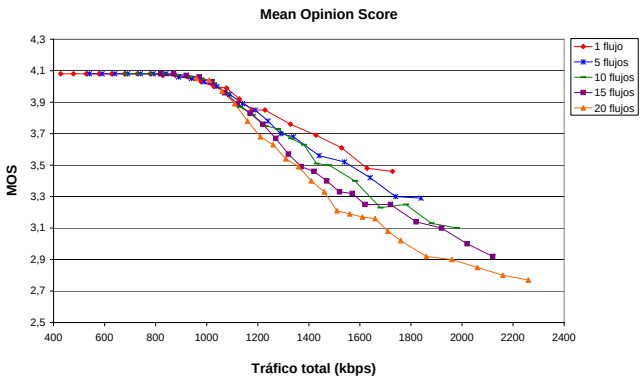

Fig. 17. MOS en función del tráfico total

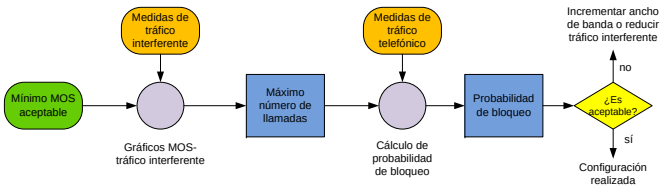

Fig. 18. Diagrama de flujo para configurar el sistema CAC

C. Configuración del sistema CAC

Se ha podido comprobar que existe un compromiso entre la QoS de las llamadas y la probabilidad de admisión del sistema. Por tanto, los pasos para configurar el sistema CAC son (Fig. 18): en primer lugar, se mide el tráfico interferente y el tráfico telefónico de la sucursal en la Hora Cargada; después, se decide un valor mínimo de MOS que sea aceptable para las llamadas y utilizando la Fig. 13 se obtiene el máximo número de llamadas simultáneas. Entonces, usando un método tradicional, como por ejemplo las tablas Erlang, se calcula la probabilidad de rechazo del sistema. Si este valor resulta inaceptable, la solución pasa por reducir el tráfico de fondo o incrementar el ancho de banda.

VI. Conclusiones y líneas futuras

Se ha integrado un sistema CAC con una centralita *software* para añadir QoS a un sistema de telefonía IP que trabaja con SIP. El sistema utiliza un *proxy* SIP para aceptar o rechazar las llamadas dependiendo de unos parámetros establecidos en el tiempo de configuración.

El sistema se ha implementado en una plataforma de pruebas basada en virtualización. Cada elemento del sistema se traslada a una máquina virtual. Se han añadido retardos en la red y anchos de banda para obtener una situación más realista.

Existe una primera fase en la que las llamadas se simulan, realizándose después con tráfico real. Las medidas muestran que el sistema no introduce retardos que podrían afectar la calidad experimentada por los usuarios. Actualmente se está añadiendo al sistema un modo basado en medidas, instalando un subsistema de medidas en el agente local, que modifica el máximo número de llamadas dependiendo de las condiciones de la red.

También se puede concluir que nuestro sistema permite intercambiar la probabilidad de bloqueo entre los *gateway* y la red IP. Pero si no se desea incrementar la probabilidad de bloqueo, existen dos soluciones: reducir el tráfico interferente o incrementar el ancho de banda, que será probablemente más económico que incrementar el número de líneas del *gateway*.

Agradecimientos

Este trabajo está parcialmente financiado por el proyecto RUBENS, del proyecto EUREKA CELTIC (código EU-3187 CP5-020) Europeo, el proyecto TSI-020400-2008-020 del subprograma AVANZA I+D, del Ministerio Español de Industria, Turismo y Comercio, el proyecto Cheque Tecnológico 2009/2010, de la Agencia Aragón I+D, del Gobierno de Aragón, y el proyecto Cátedra Telefónica, de la Universidad de Zaragoza.

Referencias

[1] X. Chen, C. Wang, D. Xuan, Z. Li, Y. Min, W.Zhao, "Survey on QoS Management of VoIP" , *In Proc. of the 2003 International Conference on Computer Networks and Mobile Computing, IEEE Computer Society.*

[2] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services", *in Proc. of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, 2004.

[3] P. K. Biswas, C.Serban, A. Poylisher, J. Lee, S. Mau, R. Chadha, C. J. Chiang, R. Orlando, K. Jakubowski, "An integrated testbed for Virtual Ad Hoc Networks", *5th International Conference on Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops Tridentcom 2009*, pp.1-10, 2009.

[4] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, Ch. Barb, A. Joglekar, "'An integrated experimental environment for distributed systems and networks"', in Proc. 5th symposium on Operating systems design and implementations, Boston, 2002.

[5] T. R. Henderson, M. Lacage, G. F. Riley, "Network Simulations with the ns-3 Simulator", *demo paper at ACM SIGCOMM'08*, ago. 2008.

[6] 3GPP TS 24.228 v5.15.0, Signalling flows for the IP multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP), Stage 3, R5, sep. 2006.

[7] SIP: Measurement-Based Call Admission Control for SIP, http://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ftcacsip.pdf.

[8] M. Ahmed and A. Malik Mansor, "CPU dimensioning on performance of Asterisk VoIP PBX", 11th Communications and Networking Simulation Symposium (CNS 2008), Ottawa, abr. 2008.

[9] VoIP Call Admission Control, http://www.cisco.com/en/US/docs/ios/solutions_docs/voip_solutions/CAC.pdf

[10] R. Solange Lima, P. Carvalho and V. Freitas, "Admission Control in Multiservice IP Networks: Architectural Issues and Trends", IEEE Communications, Vol. 45 No. 4, abr. 2007, 114-121.

[11] S. Wang, Z. Mai, D. Xuan, W. Zhao, "Design and implementation of QoS-provisioning system for voice over IP", Parallel and Distributed Systems, IEEE Transactions on, vol.17, no.3, pp. 276-288, mar.2006.

[12] S. Wanke, M. Scharf, S. Kiesel and S. Wahl, "Measurement of the SIP Parsing Performance in the SIP Express Router", Proc. 13th Open Eur. Summer School and IFIP TC6.6Workshop (EUNICE 07) Enschede, Neth., 2007.

[13] J. Saldaña, E. Viruete, J. Fernández-Navajas, J. Ruiz-Mas, J. I. Aznar, "Hybrid Testbed for Network Scenarios". SIMUTools 2010, the Third International Conference on Simulation Tools and Techniques. Torremolinos, Malaga (Spain), mar. 2010.

[14] E. Alipour, K. Mohammadi, "Adaptive Admission Control for Quality of Service Guarantee in Differentiated Services Networks", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.6, jun. 2008.

[15] H. Tran, T. Ziegler, F. Ricciato, "QoS Provisioning for VoIP Traffic by Deploying Admission Control. Architectures for Quality of Service in the Internet", Springer Berlin/Heidelberg, Vol. 2698/2003, pp. 1084-1085.

[16] J. Zhou, Z. Ji, R. Bagrodia, "TWINE: A Hybrid Emulation Testbed for Wireless Networks and Applications". Proc. IEEE INFOCOM 2006.

[17] X. Jiang, D. Xu, "vBET: a vm-based emulation testbed", Proc. of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research (MoMeTools03). New York, NY, USA: ACM Press, 2003, pp. 95-104.

[18] B. Quetier,V. Neri, F. Cappello, "Selecting a Virtualization System For Grid/P2P Large Scale Emulation", In Proc. of the Workshop on Experimental Grid testbeds for the assessment of large-scale distributed applications and tools (EXPGRID'06), Paris, France.

[19] S. Hemminger, "'Network Emulation with NetEm", Proc Linux Conference AU, Canberra, 2005.

[20] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, G. Ventre, "D-ITG Distributed Internet Traffic Generator" QEST , IEEE Computer Society, pp. 316-317, 2004.

[21] Cooperative Association for Internet Data Analysis "NASA Ames Internet Exchange Packet Length Distributions".

[22] One-way transmission time (recommendation G.114). International Telecommunication Union (ITU), feb. 1996.

[23] D. Wischik, N. McKeown, "Part I: buffer sizes for core routers". SIGCOMM Comput. Commun. Rev. 35, 3 (jul. 2005), 75-78.

[24] "'The E-model, a computational model for use in a transmission planning"', ITU-T Recommendation G.107, Mar. 2003

[25] http://www.itu.int/ITU-T/studygroups/com12/emodelv1/calcul.php