Arquitectura de Pasarela Residencial Orientada a la Autoconfiguración

Jaime García*, Iván Vidal*, Francisco Valera* y Arturo Azcorra*†
*Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
Avda. de la Universidad 30, 28911, Leganés - Madrid, España
Email: {jgr, ividal, fvalera, azcorra}@it.uc3m.es
†IMDEA Networks
Avda. Mediterraneo 22, 28918, Leganés - Madrid, España

***Abstract*— Los dispositivos de comunicación de datos mejoran sus funcionalidades día tras día. Con cada nuevo equipo, el usuario debe aprender a configurarlos, administrarlos, cargar nuevas actualizaciones y cuando se requiere una nueva funcionalidad no soportada por dicho equipo, cambiarlo por uno nuevo. Hoy en día, configurar un modem-router xDSL es una tarea complicada para un usuario inexperto y varios proveedores ya optan por la configuración remota. Conforme estos equipos se vuelvan más complejos, esta última funcionalidad será más y más demandada, aunque los usuarios con más experiencia querrán tener la posibilidad de realizar una configuración local. Este artículo propone una arquitectura general de una pasarela residencial con la flexibilidad suficiente como para cargar y descargar módulos individuales que desempeñen funcionalidades muy dispares. Esta arquitectura está especialmente pensada para implantarse en una red de siguiente generación, aunque puede emplearse en cualquier tipo de red.**

I. Introducción

Los hogares se están volviendo más inteligentes, los servicios más complejos, los dispositivos más diferentes y numerosos y los usuarios más exigentes. La llave para abrir entornos residenciales a este nuevo conjunto de requisitos se encuentra en las pasarelas residenciales (RGW a partir de ahora), responsables de conectar los hogares con las redes de acceso y de proveer un marco de trabajo donde los servicios se puedan desarrollar y configurar de acuerdo a las características del usuario.

Los RGWs actuales (muchos de ellos son de hecho más 'routers' que 'gateways') de hecho ya han incrementado sus capacidades para incluir varios interfaces (Ethernet, Wireless, PLC, USB, etc.), para proveer servicios de triple-play, la división del enlace en varios canales lógicos, etc. Esto implica un avance notorio en comparación con los modem-routers de ADSL actuales que fueron, y aún son, ampliamente utilizados, pero que no son lo suficientemente flexibles para instalarse en un entorno multi-servicio y multi-proveedores. Normalmente son soluciones cerradas para un proveedor en particular.

Este artículo presenta una arquitectura general para la configuración y control de los RGWs cuyo objetivo principal es el de permitir a diferentes agentes de configuración (ACs) instalarse en los RGW para configurar sus parámetros, basándose en las primitivas de control implementadas por dicho AC. Una de las peculiaridades más importantes de esta propuesta es que el interfaz ofrecido por la capa de control hacia los ACs es común. No se impone la utilización de ninguna tecnología en particular para la implementación de estos ACs mientras se respete la definición de la interfaz. Por ejemplo, un AC puede ser una aplicación hecha en Java, en C, un bundle de OSGi, un servidor web, etc. Será el responsable de interpretar mensajes de diferentes protocolos como SIP, SNMP, TR-069, RTCP, etc. y basado en la información obtenida de la interacción con la Capa de Configuración usando un interfaz común. En este artículo se presenta una descripción exhaustiva de un AC SIP, como el punto de configuración automático, ya que ha sido el protocolo elegido para el control de sesión en varias propuestas de arquitectura en Redes de Siguiente Generación (por ejemplo, en TISPAN NGN [1]).

El resto del artículo se organiza de la siguiente forma. La Sec. II presenta un resumen de las diferentes iniciativas que existen en la actualidad para promover un estándar de pasarela residencial. En la Sec. III se define y explica la arquitectura propuesta para una pasarela residencial con configuración automática. Esta arquitectura será posteriormente validada en la Sec. IV. Posteriormente, se introducirán algunos ejemplos de Agentes de Configuración en la Sec. V y en la Sec. VI se concluye el artículo resumiendo las aportaciones más importantes propuestas en el mismo.

II. Estado del arte de las RGWs

Esta sección repasa la situación actual de estándares y arquitecturas de las RGWs, la necesidad de crear una RGW de configuración automática en entornos inteligentes de siguiente generación y una pequeña introducción de la propuesta que será más detallada en las siguientes secciones.

II-A. Situación actual

Actualmente las comunidades científicas y de la industria han puesto especial interés en el hogar inteligente y especialmente en la RGW, como principal dispositivo en ese entorno. Existen muchas iniciativas para estandarizar una arquitectura de una RGW, centrándose en diferentes puntos pero con el mismo objetivo. Por ejemplo, el HGI (Home Gateway Initiative) es un foro abierto en Diciembre de 2004 con el objetivo de generar una especificación de una RGW. El HGI recoge en

el documento [2] una propuesta completa y exhaustiva de una arquitectura para la RGW. Además, el DSL Forum recoge en [3] una amplia y completa lista de parámetros que extienden el TR-069 [4] para contemplar una RGW. El UPnP (Universal Plug and Play) Forum también ha generado propuestas para la RGW [5]. Además de estos organismos de estandarización, existen otras propuestas fruto de proyectos de investigación como MUSE, ASTRALS y MEDIANET. Por ejemplo, en MUSE se creó un grupo entero de trabajo para diseñar la arquitectura de una RGW.

II-B. Descripción del problema

Como se ha comentado anteriormente, existen varias iniciativas encaminadas a estandarizar una arquitectura de una RGW, usando diferentes nombres y a veces diferentes expresiones para definir lo que debe ser una RGW. Este último punto es muy importante ya que diferentes organismos de fabricantes/proyectos/estandarización tienen su propia visión sobre la funcionalidad de una RGW. La RGW básica es sólo un equipo de conexión entre la red del hogar y la de acceso. Varias funcionalidades pueden ser añadidas a esta configuración básica como la QoS, capacidades multicast, administración local y/o remota, funcionalidades UPnP, NAT (Network Address Translation), etc. También es posible añadir servicios a la RGW como domótica, media servers (para almacenamiento, trans-codificación, etc.), eCare, web proxy, control parental, etc. convirtiendo a la RGW en una pasarela de servicios.

Un punto importante a tener en cuenta al diseñar una arquitectura es su extensibilidad. La extensibilidad puede ser definida como la capacidad de incrementar la funcionalidad sin cambiar la arquitectura principal. Aunque esta definición puede ser demasiado flexible, ya que de hecho existen dispositivos que pueden aumentar su funcionalidad, como por ejemplo con actualizaciones de firmware, insertando una nueva tarjeta inteligente, etc. es una terminología ampliamente utilizada.

Hoy en día, varias RGWs tienen esta característica de extensibilidad. Por ejemplo, una RGW puede ser extendida incluyendo nuevas capacidades IMS actualizando el firmware. Este comportamiento tiene varias desventajas:

- Es común que solo el fabricante del producto tenga la descripción completa del hardware y software, por lo que es el único capaz de crear nuevas versiones de su firmware. Esto es claramente un problema ya que los usuarios desean una rápida adopción de nuevas funcionalidades.
- ¿Cómo se realiza esta actualización? El usuario puede descargarla manualmente y reiniciar el dispositivo. Por otro lado, el propio proveedor de transporte puede realizar esta actualización y reiniciar el dispositivo (esto puede interferir con el proceso normal del usuario). Ya que el firmware es una pieza de software monolítica, el proceso de actualización finaliza con un reinicio de la RGW.
- Otras compañías de software no tienen la posibilidad de crear nuevos módulos con ciertas funcionalidades para la RGW, por lo que el usuario se ve forzado a utilizar siempre el software del propio fabricante. Aunque hoy en día las RGWs sean pequeñas y con un reducido conjunto de funcionalidades, las pasarelas de siguiente generación serán pequeños ordenadores con grandes capacidades, por lo que debería ser posible añadir y eliminar módulos de diferentes compañías.
- La Internet tradicional está cambiando, y debido a nuevas leyes o necesidades, el modelo clásico, donde un cliente tiene un sólo ISP y un par de proveedores de servicios a lo sumo, cambiará en el futuro [6]. Para estos nuevos escenarios, un único punto de acceso a la RGW no seguirá siendo válido, ya que varias entidades pueden instalar y configurar su propio software [7].

Debido a todo esto, una arquitectura monolítica tradicional no parece ser recomendable y sí una basada en múltiples capas para crear una *RGW dinámicamente extensible*. Con una arquitectura en capas sería posible pedir, cargar y descargar módulos desarrollados para realizar ciertas funcionalidades. La petición puede ser realizada por el usuario utilizando un interfaz web, por otro módulos o por un proveedor de servicio o de transporte.

Además, aún con un buen método para instalar nuevas funcionalidades en la RGW, es también importante administrar todos los procesos en ejecución para incrementar el rendimiento y para prevenir posibles problemas. Con un diseño adecuado, sería posible compartir funcionalidades entre módulos y proteger accesos restringidos al núcleo de la pasarela.

Con respecto a las iniciativas descritas en II-A, y hasta donde conoce el autor, ninguna de ellas proponen los cambios explicados aquí para crear una arquitectura de RGW dinámicante extensible.

II-C. Posibles soluciones

Es importante no confundir los términos diseño de una arquitectura con su posterior implementación. Una arquitectura puede definir diseños de bloques funcionales que, posteriormente, serán implementados con una tecnología dada. Este artículo propone una arquitectura de RGW que permite proveer un mecanismo general para automatizar el proceso de configuración. La implementación de la arquitectura propuesta se discutirá posteriormente en la sección de validación (Sec. IV). Sin embargo, debido a la flexibilidad de esta arquitectura, otras alternativas son también válidas para la implementación. Por ejemplo, la plataforma de servicios *OSGi* [8] es un entorno de ejecución Java para la creación de componentes software, pensado especialmente para RGWs. Con este entorno de ejecución, los componentes pueden ser cargados y descargados sin realizar un reinicio. Estas aplicaciones serán multiplataforma (Java) y los programadores no deben preocuparse de la comunicación entre procesos ya que se ejecutan sobre OSGi. Otra posibilidad es la de utilizar *Máquinas Virtuales* para separar varios contextos de diferentes proveedores de servicio en un mismo equipo. De todas formas, se debe proveer de un mecanismo que permita la inter-comunicación entre máquinas virtuales con el núcleo de la RGW y resolver

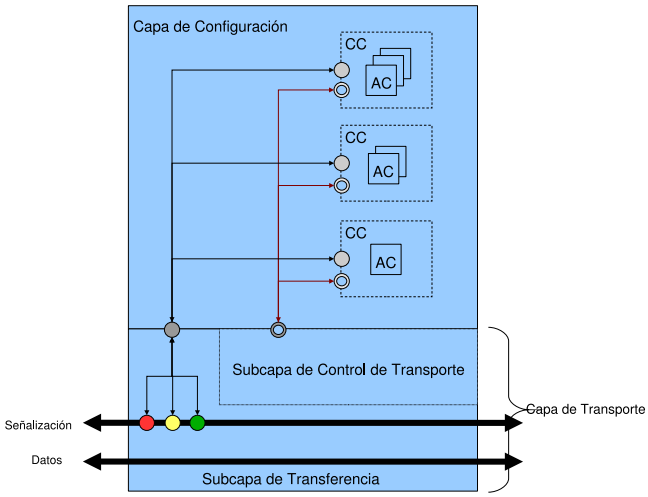

Fig. 1. Arquitectura propuesta

posibles conflictos, por lo que aún existen varios problemas a resolver.

III. Arquitectura propuesta

La arquitectura que se propone en este artículo (Fig. 1), representa la capa de configuración y control de la pasarela residencial. Una definición más completa de la arquitectura está fuera de los objetivos de este artículo (más detalles pueden encontrarse en [2]).

El objetivo principal de esta arquitectura es el de facilitar la actualización de la pasarela con nuevas funcionalidades (o mejorar las que ya existen) sin intervención alguna por parte de los usuarios. Las funcionalidades vienen proporcionadas por diferentes módulos (programas, que reciben el nombre de Agentes de Configuración, AC) que deben instalarse y ejecutarse en la pasarela residencial y que son capaces de configurar sus diferentes parámetros operativos.

Por ejemplo, si la pasarela no es capaz de interpretar mensajes de SIP (Session Initialization Protocol) y el usuario quiere subscribirse a un servicio de voz sobre IP es probable que, en cualquier caso, la pasarela tenga que ser capaz de ofrecer mecanismos para que los mensajes SIP pasen a través del NAT, asumiendo que el cliente no tenga disponible en su terminal mecanismos como STUN (mediante una pasarela de nivel de aplicación, ALG, por ejemplo). En el mismo escenario, es también posible que el usuario quiera que la misma pasarela configure los diferentes flujos para conservar la calidad de las sesiones de voz sobre IP (basándose por ejemplo en la información proporcionada por los mensajes de SIP) o que la pasarela monitorice los flujos RTP o RTCP para ser capaz de reaccionar en el caso de detectar algún problema.

Todas estas cosas podrían ser hechas de forma automática en la pasarela con el AC adecuado tan pronto como el proveedor del servicio de voz sobre IP contratado lo instale (el procedimiento de instalación se detallará posteriormente).

La arquitectura propuesta se ha dividido en dos capas: la *Capa de Configuración* y la *Capa de Transporte* (esta última dividida en la *Subcapa de Control de Transporte* y la *Subcapa de Transferencia*).

La Subcapa de Transferencia es responsable de los típicos mecanismos de reenvío de datos incluyendo funcionalidades como la clasificación de tráfico, gestión de colas, conformado de tráfico, NAPT, etc. (y por supuesto funcionalidades de encaminamiento o conmutación). Esta capa la configura la Subcapa de Control de Transporte que tiene acceso directo a diferentes parámetros como el tamaño de los buffers, el número máximo de flujos permitidos, etc.

Y finalmente en la Capa de Configuración es donde los *Agentes de Configuración (ACs)* se gestionan. Esta capa es responsable de los procedimientos de instalación y desinstalación, de crear un entorno de ejecución adecuado y de asociar los diferentes ACs en *Contextos de Configuración* (CC en la figura) cuando es necesario que exista cooperación entre los agentes.

Dichos Agentes de Configuración son las entidades responsables de contactar con la Subcapa de Control de Transporte para configurar la pasarela residencial (o simplemente para leer la información de diferentes parámetros). Aunque esto es al final una de sus principales funciones, los ACs son capaces de hacer muchas más cosas y de hecho son programas independientes que forman parte de la pasarela.

Hay dos tipos de ACs definidos en esta arquitectura de configuración (aunque en una arquitectura global, esta idea puede extenderse a más tipos):

- Los *Agentes de Configuración de Señalización*, son responsables de procesar los mensajes correspondientes a un determinado protocolo. Esto puede implicar desde simplemente observar el tráfico y obtener estadísticas a reconfigurar la pasarela en base a dichas observaciones o a modificar el plano de señalización, etc. En general habrá un AC por protocolo de señalización. De acuerdo al ejemplo de SIP que se ha comentado, un AC de SIP podría por ejemplo implementar funcionalidades de NAT traversal (mediante un ALG) modificando los mensajes SIP cuando sea necesario, podría actuar de Back-to-Back User Agent (B2BUA) para dar soporte a terminales SIP en entornos IMS/TISPAN, podría deducir (y posteriormente configurar) la calidad de servicio requerida a partir de la información intercambiada sobre los codec disponibles, etc.
- *Agentes de Configuración de Aplicación*. Estos agentes son responsables de la gestión de aplicaciones y servicios. No procesan mensajes pertenecientes a protocolos específicos pero son capaces de configurar la pasarela residencial cuando así lo requiere algún servicio. Un ejemplo de un AC de aplicación puede ser un servicio de tele medicina en el que la pasarela residencial tenga un determinado dispositivo médico conectado y este AC sea el responsable de configurar la Capa de Transferencia para enviar la información médica (alarmas, medidas, etc.) hacia el servidor del hospital utilizando un flujo de alta prioridad (ver [9] para más detalles).

El procedimiento de instalación de los diferentes ACs

depende de la implementación. La idea es que la instalación se inicie como consecuencia de una orden del usuario o del operador, la intervención de un determinado servicio, automáticamente cuando la pasarela detecta que se necesita, etc. El mecanismo para descargar el AC en la pasarela y la tecnología específica para codificar el AC se dejan abiertos. Puede desarrollarse en cualquier lenguaje, ser una aplicación independiente, un módulo Java de una plataforma como OSGi, un agente ejecutado en una plataforma de agentes inteligentes o móviles, varias de estas opciones al mismo tiempo, etc. En la sección IV se detalla la implementación propuesta.

En cualquier caso, para conseguir una comunicación adecuada entre las diferentes capas hay diferentes cosas que deben respetarse independientemente de la tecnología específica elegida para implementación y que constituyen la interfaz entre las capas de la arquitectura.

La Subcapa de Control de Transporte ofrece una interfaz común (conjunto de primitivas) a la Capa de Configuración. De esta manera no importa lo heterogénea que sea la tecnología de los ACs puesto que la configuración de la pasarela residencial se hace de la misma forma en todos los casos. De nuevo la implementación de esta interfaz es abierta (puede hacerse mediante interfaces Java, un lenguaje XML como el que se describe en el ejemplo de la sección IV-C, etc.). Esta interfaz permite que la Capa de Configuración escriba o lea un gran número de parámetros (ancho de banda disponible por interfaz, asociaciones del NAT, tamaño de una cola determinada, etc.) y ejecute cualquier tipo de acción (reinicio, restaurar una configuración, etc.).

La Subcapa de Transferencia y la de Control de Transporte están relacionadas puesto que esta segunda es responsable de la configuración de los recursos que utiliza la primera.

Y finalmente hay otra relación existente entre la Subcapa de Transferencia y la Capa de Configuración: los ACs reciben tramas de la Subcapa de Transferencia (y pueden también enviar tramas hacia ella). Es importante destacar que estas tramas deben ser encapsuladas (tuneladas) al ser transmitidas entre estas capas puesto que es importante, no ya solo tener los datos del mensaje, sino la trama completa incluyendo las cabeceras originales de nivel 2 y 3.

La última propuesta relevante de esta arquitectura es la decisión de instalar tanto la capa de Configuración como la Subcapa de Control de Transporte a nivel de aplicación, dejando únicamente la Subcapa de Transferencia para que sea ejecutada a nivel de kernel (o incluso a nivel de hardware). Aunque esto es más una decisión de implementación que de diseño, es importante destacarlo, porque parte de la flexibilidad funcional que ofrece esta arquitectura se perdería en el caso de que la opción de desarrollo final de la misma fuese otra.

Una desventaja de esta alternativa pueden ser las prestaciones obtenidas. Más que el tiempo de procesamiento, que probablemente sea despreciable, el problema es el tiempo que se tarda en enviar las tramas desde el nivel de enlace al nivel de aplicación y después de procesarlas enviarlas de nuevo al nivel de comunicación (Subcapa de Transferencia). Esto es algo que deberá ser validado una vez que la arquitectura sea instanciada e implementada.

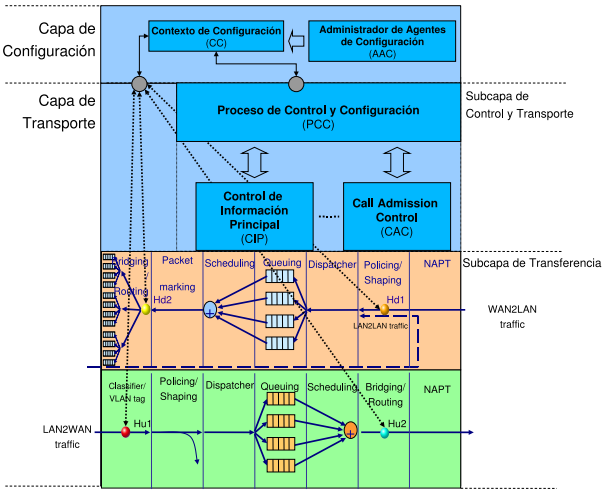

Fig. 2. La arquitectura implementada

Hay sin embargo importantes ventajas asociadas a este diseño:

- Los ACs pueden ser instalados de forma automática y sencilla sin la intervención del usuario (nada que ver con el típico mecanismo de instalación de firmware actual).
- Los ACs son fáciles de mantener: el proveedor puede actualizar el software de la pasarela residencial en cuanto exista una nueva versión del mismo (incluso la pasarela podría actualizase automáticamente si se programa para ello).
- El desarrollo y mantenimiento del software es más sencillo si se hace a nivel de aplicación que a nivel de kernel.
- La implementación de los ACs no es dependiente del kernel por lo que puede ser portada y reutilizada en diferentes plataformas.

Aunque la flexibilidad proporcionada por esta arquitectura es útil cuando se integra en un entorno multiproveedor (cada uno puede elegir su propia implementación de su AC), las implicaciones de este escenario son complejas desde el punto de vista de la seguridad, gestión, etc. y están fuera del ámbito de este artículo. Se han estudiado diferentes soluciones en [7].

IV. Validación de la Arquitectura

IV-A. Arquitectura del prototipo

La Fig. 2 presenta una instanciación de la arquitectura general introducida en la Sec. III. Se presentará en las siguientes sub-secciones un prototipo que sigue esta instanciación. Como se explicará posteriormente, la Subcapa de Transferencia fue implementada a nivel de kernel de Linux utilizando el software router modular Click! [10] mientras que la Subcapa de Control de Transporte y la Capa de Configuración en la capa de aplicación se implementaron con Java.

IV-B. Validación de la Capa de Configuración

Siguiendo la filosofía de arquitectura flexible descrita en la Sec. III, se ha utilizado Java para implementar la Capa de

Tamaño de paquete	Max. throughput
100 bytes	11.9 Mbps
200 bytes	23.7 Mbps
400 bytes	46.4 Mbps
800 bytes	89.8 Mbps
1400 bytes	95 Mbps

TABLE I
MAX. THROUGHPUT VS TAMAÑO DE PAQUETE

Tamaño de paquete	Retardo hook	Retardo directo	Retardo cont.
100 bytes	0.676 ms	0.595 ms	81 μs
200 bytes	0.726 ms	0.635 ms	91 μs
400 bytes	0.790 ms	0.712 ms	78 μs
800 bytes	0.953 ms	0.875 ms	78 μs
1400 bytes	1.203 ms	1.047 ms	156 μs

TABLE II
RETARDO VS TAMAÑO DE PAQUETE

Configuración. Por lo tanto, cada AC será una aplicación Java, lo mismo que el Administrador de Agentes de Configuración (AAC en la Fig. 2) la cual es la entidad responsable en el registro de ACs (ver Sec. III). Este bloque es capaz de instalar los diferentes ACs que conforman los Contextos de Configuración. Como ya se mencionó en la Sec. III, la flexibilidad de esta implementación a nivel de aplicación tiene el problema del retardo introducido debido al paso de mensajes de señalización desde la Subcapa de Transferencia a la Capa de Configuración. Es también importante comentar, que este retardo depende considerablemente de la implementación final. Por ejemplo, como la Subcapa de Transferencia se ha implementado con Click!, que se ejecuta a nivel de kernel, es necesario un mecanismo para extraer o copiar tramas de Click! y enviarlas al AC correspondiente a nivel de aplicación. Este mecanismo es el denominado *hook* en la Sec. III.

Para obtener el retardo introducido por el mecanismo de comunicación entre la Subcapa de Transferencia y la Capa de Configuración, primero es necesario cargar el sistema con tráfico de tasa constante. Para esto se utilizó la herramienta Iperf [11], con la que se generó tráfico UDP, con el fin de conseguir el máximo throughput admisible por el sistema frente al tamaño de paquete. Para esta prueba, un AC fue creado y registrado para extraer todo el tráfico UDP con destino al puerto 5001 de un hook de upstream (Hu1 en la Fig. 2). Para cada trama recibida, el AC sólo tiene que reinyectar la trama en el mismo hook. La Tabla I recoge los resultados obtenidos.

Se pueden extraer dos resultados importantes de esta prueba: como era de esperar, el tamaño de paquete es un parámetro importante ya que, para paquetes pequeños, el throughput es menor que para los mayores, y el throughput prácticamente se duplica cada vez que se duplica el tamaño de paquete. La principal conclusión es que la implementación puede ser considerada válida, ya que sólo se extraerán tramas de señalización y obviamente estas tramas de señalización tendrán un pequeño throughput agregado.

Otra validación importante es el retardo impuesto por esta solución. Para calcular este retardo, se realizaron dos experimentos utilizando la aplicación ping: en el primero, el ping atraviesa la Subcapa de Transferencia (retardo directo), pero en el segundo el mensaje se extrae de la dirección de upstream y es reinsertado otra vez en el hook (retardo hook). Si estas dos medias se restan es posible estimar el retardo completo introducido por este mecanismo de comunicación. La Tabla II muestra los resultados, donde los valores importantes se recogen en la última columna, donde se representa el retardo debido al cambio de contexto (retardo cont.).

Es importante resaltar que el retardo de cambio de contexto es independiente del tamaño del paquete y del orden de los micro-segundos, un valor razonable para las tramas de señalización.

IV-C. Validación de la Subcapa de Control de Transporte

Para la implementación del prototipo de la RGW, la Subcapa de Control de Transporte se implementó utilizando tres bloques funcionales:

- El Proceso de Control y Configuración (PCC), además de proveer un interfaz común a la Capa de Configuración y de cargar y descargar módulos en esta subcapa (el CAC y el CIP, por ejemplo), el PCC tiene una tercera funcionalidad relacionada con el CAC que será discutida posteriormente en dicho módulo.
- Control de Información Principal (CIP) es el único módulo en toda la arquitectura con los permisos para poder modificar la Subcapa de Transferencia. Otros módulos pueden leer, escribir, modificar, registrar o eliminar objetos de la Subcapa de Transferencia almacenados en la MIB, pero la correcta traducción entre la MIB y la Subcapa de Transferencia se realiza en el CIP. El CIP debe asegurar la integridad de esta MIB y controlar accesos múltiples. Para la implementación de este prototipo, la MIB es un documento en XML que define todos los objetos disponibles en la Subcapa de Transferencia.
- El Call Admission Control (CAC) tiene una única (aunque compleja) funcionalidad: aceptar o no un nuevo flujo con una determinada prioridad. Para esto, debe tener en cuenta los flujos instalados y el que se pide insertar para ejecutar el algoritmo del CAC. La salida de este algoritmo es binario: se acepta o no.
  El algoritmo del CAC debe tener en cuenta el algoritmo de scheduling, número de colas, tamaño de las colas, máximo tamaño de ráfaga y jitter para una prioridad dada. Con estos parámetros y lo pedido por el nuevo flujo, el algoritmo del CAC aceptará o no ese nuevo flujo basado en la prioridad y ancho de banda requerido.
  Existen algunos casos especiales donde el CAC debería ser desactivado. Este no es un comportamiento normal y tiene que ser considerado como una situación crítica. Por ejemplo, imaginemos una llamada de emergencia cuando no hay recursos en la capa de transferencia. En ese caso, el CAC rechazaría la conexión y el usuario

se vería forzado a cerrar otras conexiones (la TV, por ejemplo). Para una llamada de emergencia, esto no es aceptable y se deben aportar otros mecanismos.

Para esta implementación se define un nuevo flag para una regla de flujo denominado *unavoidable*. El módulo CAC siempre acepta todas las reglas que lleven este flag activo, independientemente de la prioridad o el ancho de banda pedido. Está claro que este tipo de flujos pueden desestabilizar el sistema, ya que la QoS no se puede seguir garantizando mientras exista una regla unavoidable activa (el sistema no necesariamente tendrá que ser inestable, pero puede serlo). En otras palabras, si existen reglas unavoidable insertadas, ante la llegada de una nueva petición al CAC, este las procesará normalmente. El sistema sólo será inestable si aceptando un nuevo flujo unavoidable, los recursos aceptados son mayores que los existentes.

Algo importante es decidir quién puede insertar una regla de flujo unavoidable. Como se ha comentado anteriormente, este debe ser un caso excepcional y sólo eventos importantes pueden originar este tipo de flujos. En el ejemplo del SIP AC implementado, este puede reconocer llamadas de emergencia (112 es el número de emergencia Europeo) por lo que puede activar el flag unavoidable a esos flujos. Esta funcionalidad ha sido probada y validada en varias pruebas, donde una RGW sin recursos disponibles recibía una llamada SIP de emergencia. En ese caso, la llamada se establecía con éxito. Pero a veces esto no es suficiente, ya que puede suceder que la llamada, aunque se establezca, comparta recursos con otros flujos lo que haga ininteligible la conversación. En estos casos, puede ser necesario parar flujos previos para dejar más recursos a los unavoidable.

Cuando un AC quiera insertar un flujo tipo alarma con unos requisitos altos de ancho de banda, este activará el flag unavoidable y además el flag *freeze*. El algoritmo es el siguiente:

- El PCC recibirá una petición de inserción de nuevo flujo con los flags unavoidable y freeze activos.
- El PCC envía el flujo al CAC filtrando la etiqueta freeze.
- El CAC acepta el flujo.
- El PCC activa el modo freeze e incrementa el contador freeze en una unidad.
- EL PCC inserta el nuevo flujo y detiene los flujos avoidables (los flujos no se eliminan).

El CAC no maneja la etiqueta freeze y sólo el PCC lo hará. El pseudo-código para peticiones al PCC sería el siguiente:

```
IF Flow has to be added THEN
   IF Flow does not have unavoidable
      flag set THEN
      IF FreezeMode is set THEN
         RETURN
      ENDIF
   ELSE
      IF Flow has freeze flag set THEN
        SET avoidable flag in Flow
        IF CAC(Flow) returns accepted THEN
           CALL MIBC.write(Flow)
           RETURN
        ELSE
           SET unavoidable flag in Flow
           SET FreezeMode
           ADD Flow to freeze array
           DEACTIVATE all avoidable Flows
        ENDIF
      ENDIF
   ENDIF
   IF CAC(Flow) returns accepted THEN
      CALL MIBC.write(Flow)
   ENDIF

ELSE IF Flow has to be removed THEN
  IF (Flow has freeze flag set) AND
     (FreezeMode is set) THEN
     REMOVE Flow from freeze array
     IF freeze array is empty
        UNSET FreezeMode
        SET AvoidableFlows
     ENDIF
  ENDIF
  CAC(Flow)
  CALL MIBC.remove(Flow)
ENDIF
```

Otra importante contribución al módulo CAC es la *promoción provisional*. Después de que una inserción de flujo sea aceptada, un AC puede pedir una promoción provisional al PCC (algunos ejemplos se describen en V-A). El CAC subirá la prioridad al flujo a la siguiente prioridad más alta si existen recursos suficientes, anotando tanto la nueva prioridad como la prioridad con la que fue aceptado el flujo. No existe un número máximo de promociones. Cuando el CAC reciba una nueva petición de inserción de flujo, si no hay recursos disponibles, intentará decrementar la prioridad de un flujo previamente promocionado. Si aún así no hay recursos, el CAC seguirá haciendo la misma operación hasta poder aceptar el flujo. Si no es posible aceptar ese nuevo flujo, se retomará la configuración previa (realmente, los flujos no se cambian hasta que el algoritmo del CAC converja a un estado estable).

Cuando el CAC necesita decrementar la prioridad de un flujo promovido, debe elegir un flujo candidato. Existen varias formas de hacer esto, aunque para este prototipo se decidió elegir al flujo más promocionado (el flujo con mayor diferencia entre su prioridad actual y la prioridad aceptada). Cuando dos o más flujos están empatados, el flujo candidato será el más antiguo (con respecto al instante de su promoción).

V. Casos de uso de Agentes de Configuración

V-A. ACs de Señalización

V-A.1. NAT-STUN: Este AC actúa como un servidor STUN [12] [13] para eliminar los problemas producidos por los NATs. Algunas aplicaciones incluyen en sus PDUs direcciones IPs y/o puertos de transporte para realizar sus funcionalidades (SIP, por ejemplo). Este tipo de aplicaciones tienen varios problemas cuando se sitúan detrás de de NATs y

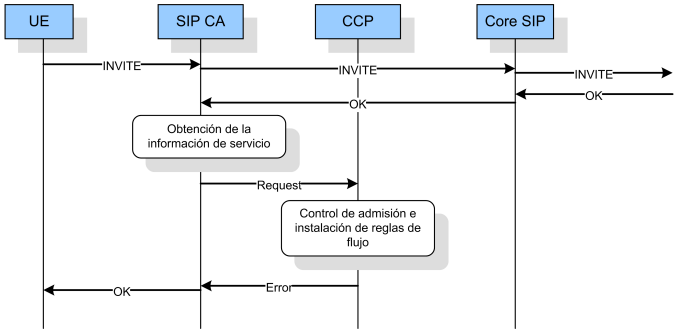

Fig. 3. Establecimiento de Sesión

normalmente no funcionarán, por lo que otros protocolos son necesarios. En este caso, STUN se usa para el descubrimiento de NATs y para obtener las direcciones IP y puertos públicos. Con esta información, las aplicaciones pueden utilizar las IPs y puertos descubiertos en vez de las locales.

Este servidor STUN puede acelerar la conexión cuando la RGW está directamente conectado a la Internet pública. En otro caso, el AC NAT-STUN deberá reenviar la petición STUN a un servidor STUN público. Para más detalles ver [14].

V-A.2. SIP: El AC SIP procesa todas las tramas de señalización SIP intercambiadas entre el terminal de usuario y los SIP proxies que se acceden desde el entorno residencial. La Subcapa de Transferencia en la RGW se configura para redirigir todas las tramas SIP recibidas tanto en downstream como en upstream hacia el AC SIP.

Durante el establecimiento de sesión SIP, después de recibir cada mensaje SIP conteniendo una respuesta SDP, el AC SIP derivará cierta información de servicio de la respuesta SDP y su correspondiente oferta SDP (esta oferta, de acuerdo al modelo Oferta/Respuesta de SDP [15] debió ser recibida en un mensaje SIP previo). La información de servicio describe los diferentes media streams (audio o vídeo, por ejemplo) que serán intercambiados durante la sesión multimedia. En concreto, para cada media stream, la información de servicio contiene los parámetros que definen los flujos asociados con el stream, en ambas direcciones de la comunicación. Estos parámetros incluyen, para cada flujo, la dirección IP destino y puerto, los requisitos de ancho de banda (opcional), el tipo de datos (vídeo o audio por ejemplo) y los formatos de flujos aceptados (por ejemplo, los codecs en caso de usar RTP). Con esta información, el AC SIP generará un conjunto de reglas de flujos que se instalarán en la RGW con el fin de dar soporte de QoS a las sesiones multimedia asociadas.

La Fig. 3 muestra un ejemplo donde se establece una sesión multimedia y la Subcapa de Transferencia se configura de forma automática para proveer la QoS requerida por la sesión.

Si la RGW está situada en una red IMS o TISPAN, se puede aprovecha de ello. En IMS y TISPAN las funcionalidades de control de sesión se basan en SIP [16], SDP [17] y el modelo Oferta/Respuesta de SDP [15]. De esta forma, el soporte de SIP introducido por el AC SIP en la RGW es suficiente para integrar la RGW en un entorno residencial de redes de siguiente generación, como el actualmente se está desarrollando por el grupo ETSI TISPAN group [1]. Sin embargo, en un contexto IMS/TISPAN, dependiendo de la política del operador, es posible que tráfico reciente (es decir, tráfico intercambiado antes de la finalización de la conexión) sea rechazado. En este caso, se necesita emplear un mecanismo de reserva y autorización [18] en la RGW. Este esquema se ha implementado de la siguiente forma:

- Cuando el AC SIP le envía al PCC los flujos que debe instalar en el RGW, incluye con cada regla una indicación para el bloque de Clasificador para que filtre las tramas de dichos flujos.
- Cuando el AC SIP recibe una respuesta SIP OK para una petición INVITE, contacta con el PCC para autorizar la reserva de recursos. La puerta se abre entonces en la Subcapa de Transferencia para todos los flujos pertenecientes a dicha sesión.

De esta forma, el AC SIP puede ser inicializado en un mecanismo de reserva y autorización, para cubrir los escenarios propuestos en IMS/TISPAN donde el tráfico reciente se impide en la propia red de acceso.

En el caso en el que la RGW esté integrado en una red basada en IMS, y existan terminales no IMS entre los equipos del usuario, se puede utilizar el bloque **B2BUA**. En este caso, el AC SIP tendrá que comportarse como un B2BUA, adoptando el rol de Servidor de Agente de Usuario (como se especifica en [16]) desde el punto de vista del terminal, y el rol de Cliente de Agente de Usuario, desde el punto de vista del core IMS. De esta forma, el AC SIP estará a cargo de realizar las funciones de control de sesión en nombre del terminal no IMS, y al mismo tiempo contactando con el PCC para instalar nuevos flujos.

En el caso que se detecte una llamada de emergencia proveniente de la red del usuario, las demandas de QoS de dicha llamada deben ser garantizadas incluso si no existen recursos disponibles en ese momento. Por lo tanto, cuando el AC SIP procesa el establecimiento de una llamada de emergencia, la información de servicio de dicha llamada se deriva de la sesión como se explicó anteriormente, pero el flujo será marcado con el flag unavoidable.

V-A.3. Pasarela de nivel de aplicación: Este AC provee mecanismos específicos de SIP para solventar los problemas originados por el NAT en SIP. El AC ALG recibe mensajes SIP y, después de examinarlos, le pide al PCC NAT bindings (o asociaciones de puertos) que son necesarios para modificar las direcciones IPs y puertos que vienen en el mensaje SIP, por direcciones públicas. De esta forma, las direcciones IPs internas y los puertos dentro de los mensajes SIP y en la carga SDP se modificarán por los bindings que se asignan por el block NAPT (utilizando el PCC como intermediario), garantizando que tanto el que inicia la sesión como el destinatario utilizarán la información correcta para el envío posterior del tráfico de sesión.

El AC ALG típicamente será instalado por el AAC en el mismo Contexto de Configuración que el resto de ACs que lo contactarán pidiendo su servicio. Por lo tanto, si el bloque

NAPT se instala en la Subcapa de Transferencia y se utiliza SIP para proveer la configuración automática de QoS en el entorno residencial, el AAC instalará una instancia del AC ALG en el mismo contexto que el AC SIP.

V-A.4. TR-069: TR-069 [4] es el protocolo de administración estandarizado por el DSL Forum para acceder a los parámetros de una RGW de forma remota. Debido a la arquitectura propuesta y a la implementación del prototipo, otro tipo de procedimiento de configuración, como por ejemplo configuración por web, puede acceder a la RGW junto al TR-069 al mismo tiempo, garantizando la consistencia de la MIB (el módulo PCC es transaccional).

V-A.5. UPnP: Este AC tiene dos interfaces: actuará como un punto de encuentro para todos los servicios disponible en la RGW (realmente otras aplicaciones ACs) y como un servicio (device en la terminología UPnP) para los puntos de control de usuarios (client en terminología UPnP). Un AC, por ejemplo un IPTV AC, debe registrarse con el UPnP si existe. La capa de device se configura para publicar este servicio cuando un punto de control lo contacta, redirigiendo todas las comunicaciones.

V-B. ACs de Aplicación

V-B.1. eCare: El equipo médico del usuario contacta al eCare AC para enviar, de forma periódica, los datos obtenidos. Este AC debe procesar los datos y generar los informes que se enviarán posteriormente a los servidores de los hospitales para almacenar los datos históricos de sus pacientes. el eCare AC tiene varias funcionalidades, y es uno de los ACs con permisos para generar reglas unavoidable y freeze.

V-B.2. Vídeo bajo demanda: Este AC provee de un interfaz web al usuario. Tan pronto se pide un vídeo por este medio, el AC contacta con otro VoD AC situado en otro RGW en modo overlay network. Un servidor de vídeo es otro miembro de la overlay network por lo que, si el vídeo no está disponible en ningún RGW, el propio servidor se encargará de servir dicho vídeo.

VI. Conclusiones

En este artículo se ha propuesto una arquitectura genérica para una pasarela residencial que cubren las capas de configuración y transporte. Esta arquitectura permite la actualización y configuración automática de la RGW basado en *Agentes de Configuración* flexibles. Como conclusión general se puede afirmar que la funcionalidad de la Subcapa de Transferencia puede ser realizada a nivel de kernel o en hardware, mientras que la Capa de Configuración y la Subcapa de Control de Transporte pueden implementarse a nivel de aplicación.

Esta arquitectura genérica se instanció en un prototipo de RGW, donde se realizaron varias pruebas de rendimiento. Se ha demostrado que implementar los Agentes de Configuración a nivel de aplicación no supone un severo impacto en el ancho de banda o en el retardo ya que todo el tráfico que va a ser procesado por los ACs es de señalización. Además, se ha detallado un mecanismo completo que soporta la instalación automática y manual de ACs a nivel de configuración. Con respecto a la Subcapa de Control de Transporte, se propuso un mecanismo de control de admisión (CAC) incluyendo grandes mejoras a sus funcionalidades. Para la Subcapa de Transferencia, se presentó una arquitectura basada en bloques que permite implementar funcionalidades de control de QoS basado en diferenciación de clases de tráfico.

Finalmente, se han incluido varios casos de uso para mostrar las posibilidades de la arquitectura. Se puso especial énfasis en la configuración automática basada en el protocolo SIP.

Agradecimientos

Este artículo ha sido parcialmente financiado por la Comisión Europea a través del proyecto MUSE (IST-026442), y del MEC español a través del proyecto CONPARTE (TEC2007-67966-C03-03/TCM).

References

[1] ETSI-TISPAN, “TISPAN (Telecoms & Internet converged Services & Protocols for Advanced Network).” [Online]. Available: http://www.etsi.org/tispan/

[2] H. G. I. HGI, “Home gateway requirements: Residential profile.” 2007.

[3] DSLForum, “TR-098 Amendment 1: Internet Gateway Device Data Model for TR-069,” December 2006.

[4] D. Forum, “TR-069 Amendment 1. CPE WAN Management Protocol. ,” 2006.

[5] UPnP, “Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0.” 2001.

[6] T. Monath, “Business role models for bb access,” in *BB Europe*, December 2004, brugge, Belgium.

[7] S. Royon, Y.; Frenot, “Multiservice home gateways: business model, execution environment, management infrastructure,” *Communications Magazine, IEEE*, vol. 45, no. 10, pp. 122–128, October 2007.

[8] OSGI, “OSGi Service Platform Release 4,” October 2007, http://www2.osgi.org/Release4/Download.

[9] V. Ribeiro, V. Pinto, J. Wellen, W. Hellenthal, W. van Willigenburg, F. Valera, I. Vidal, J. Garcia, and M. I. nez, “A European high speed Access Platform and Residential Gateway,” in *BB Europe*, December 2007, antwerp, Belgium.

[10] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, “The Click Modular Router Project.” Internet, May 2006, http://www.read.cs.ucla.edu/click/.

[11] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, “Iperf,” http://dast.nlanr.net/Projects/Iperf/.

[12] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs),” RFC 3489 (Proposed Standard), Mar. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3489.txt

[13] R. Mahy, D. Wing, J. Rosenberg, and C. Huitema, “Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs),” Internet Draft, February 2006.

[14] J. Garcia, F. Valera, I. Vidal, and A. Azcorra, “A broadcasting enabled residential gateway for next generation networks,” in *2nd IEEE International Workshop on Broadband Convergence Networks (BcN 2007)*, Munich, Germany, May 2007.

[15] J. Rosenberg and H. Schulzrinne, “An Offer/Answer Model with Session Description Protocol (SDP),” RFC 3264 (Proposed Standard), Jun. 2002.

[16] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol,” RFC 3261 (Proposed Standard), Jun. 2002, updated by RFCs 3265, 3853, 4320.

[17] M. Handley, V. Jacobson, and C. Perkins, “SDP: Session Description Protocol,” RFC 4566 (Proposed Standard), Jul. 2006.

[18] TISPAN, “ETSI ES 282 003 V1.1.1: “Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Resource and Admission Control Sub-system (RACS); Functional Architecture.”,” March 2006.