DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

PROYECTO FIN DE CARRERA

Soporte al descubrimiento y autoconfiguración de dispositivos y servicios basados en UPnP sobre una pasarela residencial

AUTOR: Sergio Morcuende Parrilla
TUTOR: Julio A. Cano Romero

Leganés, 2011

Resumen del proyecto

Hoy en día los ordenadores personales desempeñan cada vez más actividades cotidianas, esto hace que los consumidores busquen soluciones de conectividad entre el ordenador personal y los productos de electrónica de consumo. El desarrollo de nuevas tecnologías y su interacción con las redes de datos, están propiciando el hecho de que los equipos interconectados en red, no sean otros que los que hoy pueblan las casas modernas: tostadoras, microondas, televisores, cadenas de música, sistemas de alarmas, termostatos, llaves de paso del gas o el agua, y más, sólo que ahora estos dispositivos cuentan con mecanismos de control más sofisticados que regulan su funcionamiento de manera automática e incluso remota. Estos dispositivos están teniendo un crecimiento considerable y una curva ascendente en sus aplicaciones que es imparable. En muchos entornos se habla ya de viviendas domóticas o inteligentes teniendo en cuenta múltiples versiones y matices.

Paralelamente a este hecho, el éxito que han tenido las redes con arquitectura TCP/IP en el mundo (gracias a que es la tecnología por antonomasia de Internet), ha supuesto la proliferación de redes de banda ancha domésticas para el acceso a Internet. Esto último junto a la demanda de ubiquidad de múltiples ordenadores personales, telefonía móvil y dispositivos dentro del entorno domótico, están forzando la fusión de las tecnologías usadas para las redes de comunicaciones privadas (utilizadas solo para el entorno domótico) con puertas de enlace a Internet o pasarelas residenciales, debido a la tendencia que existe en un futuro de utilizar las redes de datos IP para conectar cualquier dispositivo electrónico. Estos elementos integradores, como son las pasarelas residenciales, facilitan precisamente la interconexión de estas plataformas en el hogar y a la vez permite su interconexión con el exterior, con redes públicas como Internet. Una de las iniciativas más aceptadas por los fabricantes y desarrolladores de software, como tecnología de integración, son los sistemas basados en OSGi (*Open Service Gateway Initiative*).

Una de las tecnologías que hace posible la creación de un entorno domótico y que además se puede desplegar sobre redes IP, es la tecnología UPnP (*Universal Plug and Play*), tecnología en la que se basa este proyecto y que ofrece grandes expectativas para el concepto de "Hogar digital". Con UPnP, la pasarela residencial es capaz de descubrir y registrar dinámicamente recursos y servicios dentro del entorno domótico.

En este proyecto se ha desarrollado una extensión de la tecnología UPnP dentro de sistemas basados en OSGi, para comunicar dispositivos y servicios entre dos pasarelas residenciales a través de redes públicas como Internet. Se aborda por tanto la problemática que surge al intentar extender la tecnología UPnP a redes externas debido al direccionamiento *unicast* de Internet. Como resultado final de las simulaciones, el software implementado es capaz de instalarse en cualquier pasarela residencial basada en OSGi, y acceder a dispositivos y servicios dentro de la red privada que se interconecta a otra pasarela reciproca a través de Internet.

Palabras clave: UPnP, IGD, OSGi, Domótica, Pasarela Residencial.

Abstract

Nowadays the personal computers carry out more and more activities daily, this causes that the consumers look for solutions of connectivity between the personal computer and products of consumption electronics. The development of new technologies and its interaction with the data networks, are causing the fact that the equipment interconnected in network, is not other than those that today populates the modern houses: toasting, microwaves, televisions, chains of music, alarm systems, thermostats, stop cocks of the gas or the water, and more, only than now these devices count on control mechanisms more than sophisticated which regulate his operation of automatic and even remote way. These devices are having a considerable growth and an ascending curve in their applications that are unstoppable. In many environments are already talking about home automation or smart houses, taking into account multiple versions and nuances.

Parallel to this fact, success that has had the networks with architecture TCP/IP in the world (thanks to the fact that it is the technology by excellence of Internet), has supposed the proliferation of domestic broadband networks for the access to Internet. This latest next to the demand of ubiquity of multiple PCs, mobile phones and devices within the environment automation, is forcing the fusion of the technologies used for the private communications networks (used only for the environment automation) with gateways to the Internet or residential gateways, to the trend in the future to use IP data networks to connect any electronic device. These integrating elements, such as residential gateways, indeed facilitate the interconnection of these platforms in the home and simultaneously it allows his interconnection with the outside, with public networks like Internet. One of the initiatives more accepted by the manufacturers and developer of software as technology integration systems are based on OSGi (Open Service Gateway Initiative).

One of the technologies that enable the creation of an automation environment and also can be deployed over IP networks is the UPnP technology (Universal Plug and Play), technology that underlies this project and that offers great expectations for the concept of “Digital Home”. With UPnP, the residential gateway is able to discover and dynamically register resources and services within the automation environment.

In this project an extension of the UPnP technology has been developed within OSGi-based systems, to communicate devices and services between two residential gateways to public networks like the Internet. It therefore addresses the problem that arises when trying to extend the UPnP technology to external networks due to the unicast address of the Internet. Like final result of the simulations, the deployed software can be installed in any OSGi-based residential gateway and access devices and services within the private network to another gateway interfaces between them through Internet.

Keywords: UPnP, IGD, OSGi, Smart house, Residential Gateway

Índice de contenido

Índice de Figuras

Índice de Tablas

Índice de Cuadros de texto

Capítulo 1

1. Introducción y objetivos del proyecto

Este capítulo sirve de presentación general del proyecto fin de carrera. Primero se expondrá una breve introducción para situar al lector en el marco tecnológico en el que se mueve este proyecto. Posteriormente se hablará sobre las motivaciones que han dado lugar al mismo y los objetivos marcados para su desarrollo, y finalmente se describirá la estructuración de esta memoria.

1.1. Introducción

A comienzos de este nuevo siglo, se va ha visto un desarrollo imparable de todo tipo de nuevos dispositivos electrónicos debido al imparable desarrollo de la computación electrónica y las telecomunicaciones en los últimos años. Esto se traduce en todo un mercado de productos electrónicos de consumo en los hogares y empresas, ya sean sistemas audiovisuales, de comunicación o automatismos. Si se analizan los factores humanos que propician este hecho, pueden aparecer conceptos como: confort, seguridad, eficiencia energética, y como no, progreso tecnológico; que se contagia a otros ámbitos de nuestro entorno, como puede ser la arquitectura.

Existe un aumento de las personas que usan, de manera directa o indirecta, dispositivos controlados por pequeños sistemas informáticos y lo más curioso es que cada vez estos sistemas se vuelven más autónomos. El hecho de que estos dispositivos sean capaces de realizar tareas por si mismos reaccionando a su entorno, los ha colgado, poco a poco, la etiqueta de "inteligentes". Es decir, empezaron a ser altamente automatizados por medio de la integración de todos sus sistemas. Un claro ejemplo son los nuevos electrodomésticos que incluyen más automatización electrónica, incluso pequeños sistemas informáticos en su interior. Todo ello viene ligado siempre además a intentar reducir el consumo energético y aumentar la eficiencia que genera el hecho de producir estos sistemas.

Todo lo anteriormente mencionado junto a una creciente proliferación de tecnologías de red cada vez más rápidas, sin olvidar el uso de Internet, hace que se planteen nuevas posibilidades de interconexión entre estos dispositivos, ampliando las capacidades antes aisladas en labores específicas e intentando mejorar su eficiencia.

Si se extrapolan estas nuevas posibilidades en el intento de integrar este progreso

tecnológico en el campo de la arquitectura, aparece el concepto de "Edificio Inteligente" [1]. Este concepto no es nuevo, surgió a mediados de la década de 1980 y ello atrajo la atención de constructores de edificios y del mercado inmobiliario. Esta nueva propuesta, en principio, integró todos los aspectos de comunicación dentro del edificio, centralizó los sistemas de seguridad e íntegro el control de algunos sistemas como el control de temperatura del edificio. En la actualidad, al estudio de edificios inteligentes se le llama "Inmótica" [2] y engloba muchos más aspectos como: la gestión integral de los sistemas básicos de un edificio (agua, electricidad, gas,…), sistemas de seguridad conectados con el exterior, control de iluminación, comunicaciones y entretenimiento, etc.

Hoy en día, el edificio inteligente es entendido como una instalación que puede adaptarse a nuevas necesidades, y es capaz de interactuar con todos los elementos instalados en él, con el objetivo de maximizar el ahorro y la eficiencia energética, la accesibilidad, la seguridad, el confort, etc. En definitiva, la Inmótica ya no se percibe como algo futurista sino como un nuevo servicio en alza con grandes perspectivas. Esto supone todo un horizonte de posibilidades que poco a poco está viendo la luz.

Si se limita el uso de estas tecnologías a los entornos domésticos o viviendas, aparecen conceptos hoy en día como, domótica [3] o 'entorno domótico', 'casa inteligente' (*smart house*), 'sistemas domésticos' (*home systems*) o 'automatización de viviendas' (*home automation*). El hecho de que existan diferentes nomenclaturas, es debido a que han surgido de avances tecnológicos distintos atendiendo a diferentes necesidades. Aunque, poco a poco, estos avances tienden hacia la denominación de "hogar inteligente" o "entorno domótico", en los que prima la interoperabilidad de las diferentes tecnologías existentes, en un intento de crear estándares internacionales.

Una definición más técnica del concepto de hogar inteligente sería: «*conjunto de servicios de la vivienda garantizado por sistemas que realizan varias funciones, los cuales pueden estar conectados entre sí y a redes interiores y exteriores de comunicación. Gracias a ello se obtiene un notable ahorro de energía, una eficaz gestión técnica de la vivienda, una buena comunicación con el exterior y un alto nivel de seguridad*» [4].

Para que un hogar suficientemente automatizado pueda ser considerado inteligente, debe sumar sistemas basados en las nuevas tecnologías de la información. Por tanto, una casa inteligente es aquélla cuyos elementos o dispositivos están integrados y automatizados a través de una red de datos y que a través de otro dispositivo remoto o interno, pueden modificar sus estados o realizar ciertas acciones cuando han detectado cambios en su propio entorno. Este funcionamiento suele estar gestionado por una unidad central que, en función de una determinada programación, tratará y elaborará la información recibida ofreciendo una respuesta que se podrá manifestar de diversas formas en el entorno.

Las diferentes iniciativas realizadas para configurar la red doméstica o entorno domótico, han llevado a los largo del tiempo a la creación de una amplia gama de tecnologías de comunicación diferentes y en su mayoría incompatibles debido al tipo de arquitectura y los protocolos de comunicaciones que usan. Ejemplos de estas tecnologías son por ejemplo: *HomePNA*, *X10*, *LonWorks*, *CEBus*, *WLAN*, *Bluetooth*, *HAVi*, *Bacnet*, *Konnex*, *SCP* y *HAPI*. Además de estas, existen otras tecnologías de más alto nivel que permiten el descubrimiento automático y dinámico de dispositivos, y que

además se puede desplegar sobre redes IP, como es el caso de *Jini* y *UPnP* (*Universal Plug and Play*). Esta última tecnología nombrada fue impulsada por Microsoft, y es en la que se basa este proyecto ya que ofrece grandes expectativas para el concepto de "Hogar digital" [5].

Para conseguir el concepto de Hogar Digital, y debido a las incompatibilidades que pueden existir entre las diferentes tecnologías de red existentes en la vivienda, es necesario un elemento integrador, no solo para que interoperen entre ellas, sino también con el exterior de la vivienda[1]. Por ello es aconsejable también introducir el concepto de "Pasarela Residencial" [6] o "Pasarela de servicios", ya que va a suponer una parte esencial del proyecto.

Una Pasarela Residencial es un dispositivo que conecta las infraestructuras de telecomunicaciones (datos, control, automatización,...) del hogar digital a una red pública de datos, como por ejemplo Internet. La Pasarela Residencial normalmente combina las funciones de enrutador/hub, de modem con acceso a Internet para varios ordenadores, de cortafuegos, de servidor de aplicaciones de entretenimiento como vídeo/audio bajo demanda, de comunicaciones como *VoIP* (telefonía sobre Internet) o incluso de telecontrol como la domótica. De esta manera se permite la conectividad total de los hogares con el mundo exterior para poder controlar de forma remota, desde cualquier parte, electrodomésticos, sistemas de seguridad, de gestión energética, equipos de electrónica de consumo como vídeos y televisores, ordenadores personales y muchos más.

Debido a la diversidad de tecnologías existentes en los hogares, se hace evidente la necesidad de un estándar común que establezca un marco de trabajo sobre el que las compañías puedan desarrollar servicios y aplicaciones para las pasarelas, sin tener que preocuparse por aspectos de bajo nivel como tecnologías de red o hardware, en este ámbito residencial nace *OSGi* (*Open Service Gateway Initiative*). La plataforma de servicios OSGi es complementaria a la mayoría de los estándares o iniciativas actuales, trabaja con varios estándares de acceso a dispositivos.

Para terminar esta introducción, un ejemplo de una posible configuración estándar de un hogar inteligente, a día de hoy, contaría con un sistema compuesto por: uno o varios ordenadores conectados mediante un sistema de red, sensores de temperatura (exterior e interior), detectores/medidores de humo, gas y agua, video porteros, sensores magnéticos para puertas y ventanas, detectores de presencia, mandos a distancia y emisores-receptores de señal, sistemas de audio/video, electrodomésticos en red, etc. Todo ello conectado a través de una pasarela residencial, con un posible software integrado de gestión de dispositivos con pequeños *displays* por toda la vivienda que dan información y permiten interacción del usuario, a través de un interfaz gráfico.

[1] **Nota para el lector:** En diversas partes del documento cuando se habla de entorno inteligente, Hogar Digital, entorno domótico, etc., se alude a entornos domésticos para que el lector tenga una idea más clara del concepto, pero estos términos se pueden extender a cualquier tipo de entorno o construcción.

1.2. Objetivos del proyecto

La tecnología UPnP originariamente se pensó desarrollar únicamente dentro del entorno residencial o doméstico. Debido a las nuevas necesidades del usuario que surgen al avanzar la tecnología, ya sea por que nacen aplicaciones que necesitan gestionarse de manera remota o simplemente midiendo el nivel de comodidad, nos encontramos con la limitación actual de usar esta tecnología fuera del entorno residencial donde está desplegada.

El objetivo de este proyecto se centra en solucionar este problema que se plantea al intentar extender la tecnología UPnP a redes externas fuera del entorno residencial, generalmente Internet, debido al direccionamiento IP que existe actualmente. Dicho esto, aclarar que este proyecto no se centra en que tipos de dispositivos UPnP puedan existir, más bien, en definir la infraestructura necesaria y la tecnología para que se puedan desplegar éstos y puedan prestar sus servicios sobre pasarelas residenciales.

El diseño de la infraestructura se basa en un modelo de utilización de la tecnología UPnP para el descubrimiento y autoconfiguración de dispositivos y servicios residenciales haciendo uso de un tipo de dispositivo ya existente en UPnP llamado IGD (Internet Gateway Device) que sirve de puente entre el entorno residencial y la red externa. En pocas palabras, el objetivo del proyecto consiste en poder comunicar dispositivos entre dos entornos residenciales con direccionamiento IP privado a través de redes públicas.

En el siguiente capítulo se hablará más detalladamente sobre la problemática actual del direccionamiento de redes IP en Internet y se hará un análisis del estado de madurez de la tecnología UPnP.

1.3. Contenido de la memoria

A continuación se expone de manera resumida como está estructurada la memoria de este proyecto.

Después de este capítulo de introducción inicial al proyecto, le sigue el segundo capítulo, que se ha centrado en describir el estado del arte, es decir, hacer un repaso de la tecnología existente implicada en la realización del proyecto. Dentro de este capítulo se aborda todo lo que concierne a las líneas principales del proyecto, como son el estándar OSGi para pasarelas residenciales, y los protocolos de comunicaciones que entran en juego para el funcionamiento de una red basada en tecnología UPnP, así como el funcionamiento de este en profundidad, ya que es el eje fundamental en torno al que gira el proyecto. Finalmente, se expone un resumen del estudio realizado sobre las posibles formas de extender la tecnología UPnP, para solventar los problemas de direccionamiento de paquetes que surgen al interconectar redes privadas basadas en esta tecnología, con redes públicas como Internet.

La solución escogida es analizada en profundidad en el tercer capítulo, presentando el modelo propuesto de desarrollo. Este modelo consiste en un diseño

abstracto de diferentes módulos, que compondrán el software que posteriormente será implementado a través de la pasarela residencial.

El capítulo cuarto se dedica a la implementación de la plataforma, se detalla cómo se ha desarrollado el código de los modelos expuestos en el capítulo anterior, explicando cómo se ha conseguido integrar la tecnología UPnP en la pasarela residencial basada en OSGi y los mecanismos para comunicar las dos subredes privadas.

El capítulo quinto es el capítulo que contiene el manual de ejecución de las simulaciones, se explican en él todos los detalles referentes a la forma de instalación y ejecución del software que implementan la funcionalidad UPnP en la pasarela residencial, así como las pruebas realizadas que demuestran su correcto funcionamiento en un entorno simulado de dos subredes que se comunican a través de una red pública como Internet.

El capítulo sexto trata sobre las observaciones más importantes que se han contemplado en la elaboración de este proyecto y que han surgido a medida que se han ido completando las diferentes fases de desarrollo. Por ello, existe también un apartado en este capítulo donde se plantean posibles trabajos futuros o líneas de investigación para mejorar el modelo propuesto en este proyecto, solventando problemas que no se contemplan en los objetivos del proyecto.

El capítulo séptimo hace referencia a la planificación y el presupuesto general del proyecto, donde se indicarán las distintas fases del desarrollo de la plataforma, tanto las que inicialmente estaban planificadas como las posibles modificaciones de la planificación original, junto con los costes que han supuesto su realización.

Esta memoria también cuenta con un Glosario de términos y una bibliografía donde se puede ampliar y contrastar la información que en este documento se contempla. Las referencias bibliográficas estarán indicadas en el texto con un número entre paréntesis, sirva a modo de ejemplo [1]. La memoria también se acompaña con anexos, donde se puede ampliar de manera rápida la documentación del proyecto.

Capítulo 2

2. Estado del arte

En este capítulo se expone una síntesis del estudio previo de documentación sobre las tecnologías implicadas en este proyecto, elaborado antes de proceder a la realización del mismo. Este estudio está enfocado en cuatro grandes bloques principales que se presentan en forma de diferentes apartados resumidos a continuación.

En el apartado 2.1 se hará una introducción sobre el concepto de domótica y los factores que conllevan la integración de sistemas informatizados e interconectados dentro del hogar, de esta forma se intenta establecer un punto de referencia al lector sobre el uso de pasarelas residenciales dentro de las viviendas.

De esta manera entramos en el mundo de las pasarelas residenciales con un estándar muy conocido en el mercado como es *OSGi* [7] y punto clave para el desarrollo del proyecto. En el apartado 2.2 se presentará la síntesis realizada sobre las tecnologías en las que está basada OSGi, y como está formada su arquitectura.

En el apartado 2.3, se abordará el funcionamiento de la tecnología UPnP [8], tecnología como eje principal de este proyecto. Posteriormente se hará un estudio de las características de UPnP, indicando sus ventajas y limitaciones. Finalmente se hará un breve repaso de otras tecnologías similares a UPnP que se pueden desplegar sobre pasarelas residenciales comerciales. Dentro de este bloque también hay que prestar especial atención, en el capítulo 2.4, sobre el modelo del IGD (*Internet Gateway Device*) como dispositivo UPnP que ofrece la funcionalidad de gestionar la frontera con la red externa. El IGD constituye un elemento básico en el funcionamiento del modelo que posteriormente se implementara.

Finalmente en el apartado 2.5 se ofrece un resumen de las investigaciones realizadas sobre las posibles formas de extender UPnP a redes externas utilizando el soporte que aportan las pasarelas residenciales, con el objetivo final de proponer una línea de desarrollo viable como solución a la problemática que existe al extender UPnP a redes como Internet.

2.1. Introducción

Ya se ha hablado del término “Domótica” o “Casa Inteligente” en el capítulo de introducción, si se analizan más en detalle los factores que motivan el hecho de que haya varias líneas de investigación en este campo, se puede llegar a hacer un estudio más exhaustivo del actual despliegue que tienen estos términos en la sociedad. Por tanto, ¿qué características conllevan estos términos?:

- ***Integración:*** Todo el sistema funciona bajo el control de un sistema informático. De esta manera, los usuarios no tienen que estar pendientes de los diversos equipos autónomos. El sistema se rige con su propia programación, evitando dificultades de interconexión entre equipos de distintos fabricantes, etc.

- ***Interrelación:*** Una de las principales características que debe ofrecer un sistema domótico es la capacidad para relacionar diferentes elementos y obtener una gran versatilidad y variedad en la toma de decisiones. Así, por ejemplo, es sencillo relacionar el funcionamiento del aire acondicionado con el de otros electrodomésticos, con la apertura de ventanas, o con que la vivienda esté ocupada o vacía, etc.

- ***Facilidad de uso:*** Con una sola mirada en los interfaces visuales del sistema, el usuario está completamente informado del estado de su casa. Y si desea modificar algo, solo necesitará pulsar un reducido número de teclas o solo hacer un clic con el ratón. Así, por ejemplo, la simple observación de la pantalla nos dirá si tenemos correo pendiente de recoger en el buzón, las temperaturas dentro y fuera de la vivienda, si está conectado el aire acondicionado, cuando se ha regado el jardín por última vez, si la tierra está húmeda, si hay alguien en las proximidades de la vivienda, etc.

- ***Control remoto:*** Las mismas posibilidades de supervisión y control disponibles localmente, pueden obtenerse mediante cualquier conexión a Internet desde otro sistema en cualquier lugar del mundo. De gran utilidad será en el caso de personas que viajan frecuentemente, o cuando se trate de residencias de fin de semana, etc.

- ***Fiabilidad:*** Los sistemas informáticos actuales son máquinas muy potentes, rápidas y fiables. Si añadimos la utilización de un Sistema de Alimentación Ininterrumpida, ventilación forzada, batería de gran capacidad que alimente periféricos, apagado automático de pantalla, etc. Se debe disponer de una plataforma ideal para aplicaciones domóticas capaces de funcionar muchos años sin problemas.

- ***Actualización:*** La puesta al día del sistema es muy sencilla. Al aparecer nuevas versiones y mejoras sólo es preciso cargar el nuevo programa en su equipo. Toda la lógica de funcionamiento se encuentra en el software y no en los equipos instalados. De este modo, cualquier instalación existente puede beneficiarse de las nuevas versiones, sin ningún tipo de modificación.

Estos factores suponen las bases para una nueva generación de todo tipo de

dispositivos de naturaleza muy diferente. Algunas de las áreas principales de la domótica son:

- ***Automatización y Control*:** incluye el control de la iluminación, climatización, persianas y toldos, puertas y ventanas, cerraduras, riego, electrodomésticos, suministro de agua y gas, etc.

- ***Seguridad***: incluye alarmas de intrusión, alarmas personales y alarmas técnicas (incendio, humo, agua, gas, fallo de suministro eléctrico,…).

- ***Telecomunicaciones*:** incluye transmisión de voz y datos en redes locales para compartir acceso de banda ancha a Internet, y el intercambio de recursos entre todos los equipos. Además permite disfrutar de nuevos servicios como "Telefonía sobre IP" (*VoIP*) y "Televisión digital" (*DTV*).

- ***Audio y video*:** incluye la distribución de imágenes de video capturadas con cámaras dentro y fuera de la casa, a la red local y a través de Internet. Se podría incluir aquí también el concepto de "Ocio Digital" [9] que abarca todo lo relacionado con el entretenimiento como los videojuegos, el *multi-room* y el "Cine En Casa".

La domótica facilita una infraestructura para nuevos servicios muy interesantes para usuarios y proveedores de servicios. Además la integración de los sistemas permite que servicios tradicionales sean ofrecidos por nuevos actores y utilizando nuevas soluciones tecnológicas. Por ejemplo, existen nuevos sistemas que permiten al propio usuario el control de intrusión con detectores de movimiento, video-vigilancia y control remoto de las aplicaciones del hogar en tiempo real. Antes, este tipo de servicios sólo eran ofrecidos por empresas homologadas de seguridad pero ahora, aprovechando la potencia de la banda ancha y las nuevas tecnologías, pueden ser ofrecidos por un proveedor, utilizando Internet o un operador de telecomunicaciones, a través de una red móvil. El tradicional servicio ofrecido por una empresa de seguridad probablemente se va a seguir llamando servicio de seguridad, pero cuando está ofrecido por un proveedor de servicios de banda ancha puede pasar a ser llamado 'servicio control del hogar' o algo parecido.

Existen, y se están desarrollando, un gran número de servicios que se pueden realizar o crear gracias a la domótica. Aunque la lista de servicios que se va a mencionar a continuación, de ninguna manera pretende ser completa, está claro que destaca unos ejemplos de servicios muy interesantes para distintos tipos de usuarios y proveedores de servicios:

- **Automatización de eventos:** Gracias a la integración de sistemas en el hogar se puede hacer una gestión mejor y más personalizada. Esto permite ahorrar dinero, tener mejor confort y mejorar la seguridad. La programación horaria permite que la iluminación y la climatización sean utilizadas de una manera óptima adaptadas al estilo de vida y costumbres de la familia. La simulación de presencia permite hacer creer que la casa está habitada aunque estemos fuera durante el día o de vacaciones.
- **Avisos:** Dispositivos como el vídeo portero digital permiten, a través de

interfaces telefónicas, desviar todas las llamadas al portero o a los teléfonos de la casa o a algún un teléfono móvil externo multimedia por ejemplo. De esta forma estaríamos en todo momento informados de la llegada/salida de terceros o la ausencia de actividad en un determinado intervalo de tiempo. La medición de forma remota a través de Internet de consumos de agua, gas, electricidad, etc. Avisos de intrusión de los sistemas de seguridad de la casa.

- **Inteligencia Artificial**: Una característica muy importante de las casas inteligentes es que deben tener la flexibilidad para asumir modificaciones de manera conveniente y económica. Desde el punto de vista computacional, una "Casa Inteligente" sugiere la presencia de sistemas basados en técnicas de inteligencia artificial, sistemas distribuidos, capaces de:
  - Tomar las decisiones necesarias en un caso de emergencia.
  - Predecir y autodiagnosticar los fallos que ocurran dentro de la casa.
  - Tomar las acciones adecuadas para resolver dichos fallos en el momento adecuado.
  - Monitorear y controlar las actividades y el funcionamiento de las instalaciones de la casa.

Como se dijo en la introducción, es necesario centralizar todas las comunicaciones entre los distintos sistemas y dispositivos, no solo de forma interna, sino también con redes externas a la vivienda. Para controlar de una única manera los dispositivos, simplificar las tareas de administración, monitorizar los recursos, es necesaria la aparición del concepto de pasarela residencial, para unificar y homogeneizar todas las redes existentes dentro y fuera de la casa, así como el acceso a Internet.

Para que realmente un sistema, catalogado como Pasarela Residencial, tenga cierto éxito o alcance una implantación masiva, el usuario tiene que sentir que realmente los servicios de valor añadido son útiles y aportan valor, confort y tranquilidad en su modo de vida. Por ello las pasarelas deberían tener las siguientes características:

- **Instalación sencilla**. La instalación debe ser sencilla y la configuración rápida y asequible (*Plug&Play*). Igualmente, la asignación y especificación de las funciones que puede hacer cada dispositivo domótico o electrodoméstico debería ser automática.

- **Telecarga de software**. El proveedor de servicios, o directamente el usuario, bajo supervisión del proveedor, deberían ser capaces de actualizar o telecargar nuevos servicios, además de configurarlos remotamente.

- **Soporte para redes**. Las Pasarelas Residenciales deberían tener interfaces que permitan conectar redes de datos de banda ancha (>10Mbps) con tecnologías como la tradicional *Ethernet* o con las nuevas tecnologías "sin cables" como *HomePlug* [10], *HomePNA* [11] o *WLAN* [12].

- **Cortafuegos y capacidad de construir VPNs** (*Virtual Private Network*). Deben ofrecer servicios de protección de los datos y seguridad contra los ataques de los

hackers, impidiendo el acceso de estos a los ordenadores o equipos en red de la vivienda. Deberían ser capaces de formar redes privadas virtuales entre estos equipos.

- **Capacidad para soportar múltiples servicios**. Con suficiente memoria, capacidad de procesamiento y un sistema operativo robusto y multitarea, las pasarelas residenciales deberán ser capaces de ejecutar múltiples aplicaciones concurrentemente, donde cada una de ellas se corresponderá a un "servicio electrónico"[2] (*e-service)* diferente. La conexión de banda ancha será compartida entre todos estos servicios con la multiplexación de datos, ya sea a nivel IP o nivel de aplicaciones.

- **Monitorización y configuración a través de interfaces Web.** Tanto de forma local como de forma remota, el usuario podrá acceder, mediante un navegador Web, al sistema operativo de la Pasarela Residencial para cambiar su configuración, borrar aplicaciones (servicios) o supervisar su estado. Para ello las pasarelas tendrán empotrados pequeños servidores HTTP.

2.2. OSGi

Como principal tecnología, que cumple con los requisitos para ser el software de excelencia dentro de una pasarela residencial, es inevitable hablar de OSGi. Originariamente las siglas **OSGi** (***O**pen **S**ervices **G**ateway **I**nitiative*) nacieron como una iniciativa abierta a partir de 1999, para definir especificaciones abiertas de software que permitieran diseñar plataformas compatibles que pudieran proporcionar múltiples servicios sobre un entorno residencial. Fue pensado principalmente para su aplicación en redes domésticas y por ende en domótica, pero posteriormente el uso de OSGi se ha extendido a otros ámbitos.

Actualmente las siglas de OSGi se han quedado obsoletas, ya que ahora se denomina Alianza OSGi (*OSGi Alliance*) y está formada por un consorcio mundial de compañías expertas e innovadoras en tecnología. Las diferentes compañías colaboran de una forma igualitaria a través de sus beneficios, experiencia de sus profesionales y foros. Su objetivo es promover la adopción de la tecnología OSGi para asegurar la interoperabilidad de aplicaciones y servicios y su gestión por la red. Para ello ofrecen al mercado especificaciones, aplicaciones de referencia, conjuntos de pruebas y certificación para ser un marco de referencia en la industria.

La Alianza y sus miembros han especificado una plataforma de servicios basados en Java que pueden ser gestionados de forma remota. El núcleo principal de la especificación reside en un *framework* que define un modelo de gestión de ciclo de vida de aplicaciones, un registro de servicios, un entorno de ejecución y modularización. De acuerdo con este marco, se han definido una gran cantidad de capas de OSGi, de API's

[2] Hace referencia a los servicios electrónicos de valor añadido como puede ser el comercio electrónico, telemedicina y teleasistencia, tele-educación, etc.

(*Application Programming Interface*), y de servicios.

El ámbito de aplicaciones de las distintas especificaciones de OSGi que van apareciendo a lo largo del tiempo cada vez es más amplio, debido a la presencia en *OSGi Alliance* de nuevos miembros que se mueven en diferentes mercados tales como, fabricantes de hardware o PC's, empresas de desarrollo de software, de sistemas de gestión corporativos, operadores de telecomunicaciones, compañías eléctricas, industria automovilística, mercado de dispositivos móviles, etc. El éxito del framework OSGi reside en ofrecer una arquitectura completa extremo-a-extremo, que cubra todas las necesidades del proveedor de servicios, del cliente y de cualquier dispositivo instalado en el entorno, facilitando la instalación y funcionamiento de múltiples servicios en un solo dispositivo.

A esta plataforma le han puesto el nombre de "Pasarela de Servicios" en su modalidad genérica, pero en el ámbito de las viviendas y pequeños negocios, se la conoce con el nombre de “Pasarela Residencial”. En la figura 1 se puede observar un esquema general de la arquitectura de una pasarela, cuyos elementos se explican más adelante.

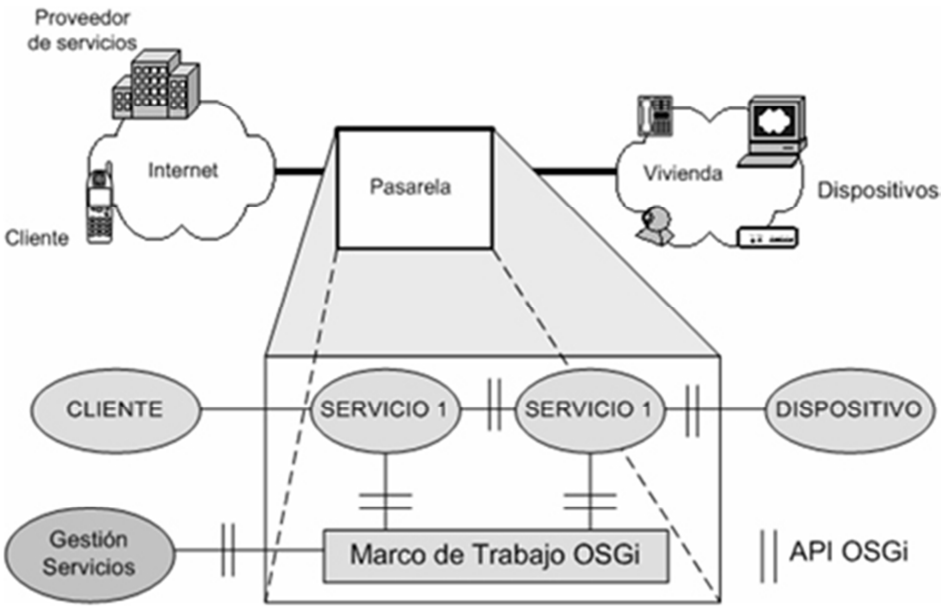

Figura 1: OSGi como pasarela de servicios [13]

La plataforma de OSGi es complementaria a la mayoría de estándares e iniciativas actuales. Esto es debido a que la especificación OSGi está orientada a la capa de aplicación y se complementa con cualquier iniciativa de las capas inferiores, en las que se incluyen protocolos de comunicación que utilizan muchas de las redes domóticas existentes. Si existe una especificación Java, como por ejemplo Jini [14], la plataforma OSGi puede usarla. También existen estándares que no están basados en Java que pueden funcionar sobre OSGi mediante módulos puente.

2.2.1. Arquitectura

Como ya se ha mencionado en la introducción, la arquitectura de OSGI [15] proporciona un marco de trabajo java de uso general, llamado *framework*, seguro y administrado que soporta el despliegue dinámico de aplicaciones conocidas como *Bundles* o módulos. Estos bundles pueden exportar servicios o ejecutar procesos, y asumir que sus requerimientos son administrados por el framework. Además, un bundle puede tener su propio *classpath* (librerías), de modo que puede brindar servicios como unidad independiente, si es lo que se desea. Todo esto se estandariza de tal manera, que cualquier bundle OSGi válido puede ser instalado, en teoría, sobre cualquier framework OSGi válido. OSGI intenta solventar de esta forma los problemas del tradicional *classloader* (cargador de clases) de la máquina virtual Java (*JVM*) y de los servidores de aplicaciones Java. En OSGI, cada módulo tiene su propio *classpath* separado del resto de los demás módulos.

Framework OSGi

El framework proporciona a los desarrolladores un entorno orientado a servicios y basado en componentes, ofreciendo estándares para manejar los ciclos de vida de los módulos o bundles instalados en él, los cuales se ejecutan todos sobre una máquina virtual Java. De ahí que el framework OSGi sea portable, ya que podrá instalarse en cualquier plataforma que tenga instalada una máquina virtual. Proporciona también un módulo de registro de servicios, componentes fundamentales también en esta tecnología. La arquitectura de servicios de OSGi se basa en un número de capas como representa la figura 2.

Figura 2: Arquitectura software OSGi [16]

La funcionalidad del framework se divide en las siguientes capas:

- **Capa de módulos**: Define el modelo de modularización. Este módulo establece las reglas para el intercambio de paquetes Java entre los bundles.

- **Capa de ciclo de vida**: Gestiona el ciclo de vida de un bundle dentro del framework, sin tener que detener la JVM.

- **Capa de servicios**: La capa de servicios proporciona un modelo de programación dinámico para los desarrolladores de bundles, simplificando el desarrollo y despliegue de módulos a través del desacople de la especificación de servicio de su implementación. El registro de servicios de esta capa proporciona un sistema para compartir objetos entre bundles, además de definir el estado de actividad de los servicios y la versión implementada.

- **Entorno de ejecución**: Define la especificación mínima para que exista un entorno de ejecución OSGi.

- **Capa de seguridad**. Adicionalmente se define una capa de seguridad que se integra verticalmente con las demás capas.

Bundle

En el entorno OSGi, los bundles son las únicas entidades para desplegar aplicaciones Java. Un bundle comprende clases Java y otros recursos que juntos pueden proporcionar funciones a los usuarios finales y proporcionar componentes a otros bundles, llamados servicios. El bundle en sí, es una aplicación Java empaquetada en un fichero *JAR*, que se despliega en una plataforma OSGI. Cada bundle contiene:

- Fichero de metadatos organizado por pares de "*nombre clave: valor"*, donde se describen las relaciones del bundle con el mundo exterior a él, como por ejemplo, el nombre del bundle, la versión, paquetes que importa o exporta (servicios), etc. Este fichero viene dado con el nombre MANIFEST.MF y se encuentra ubicado en el directorio META-INF.

- Clase Activadora: Es necesario cuando se despliega un bundle dentro de una plataforma OSGI definir una clase especial que actúa como activador del bundle, y es el punto de entrada de su ciclo de vida controlado por el propio framework. El desarrollador podrá controlar el ciclo de vida de su bundle, implementado una serie de interfaces de OSGI que contiene métodos como *'start'* (se invoca para arrancar el bundle) y *'stop'* (se invoca al pararlo).

El ciclo de vida de un bundle se puede ver en la figura 3. La transición de unos estados a otros en algunos casos es transparente para el programador, ya que es el mismo framework el que realiza la función dependiendo de los errores que internamente se puedan producir al pasar de un estado a otro.

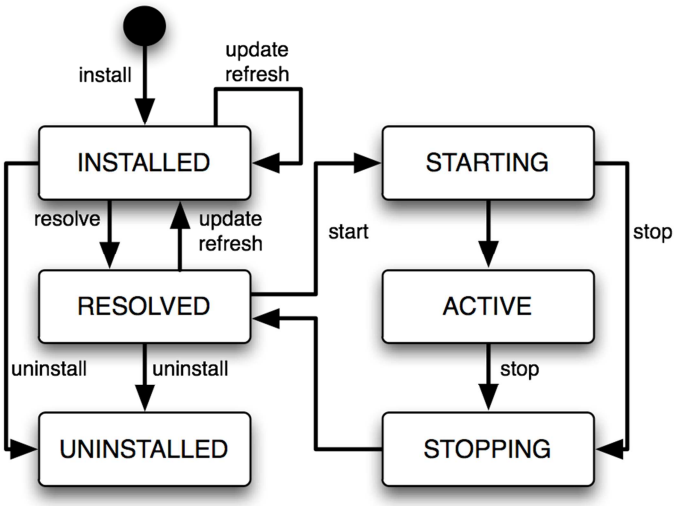

Figura 3: Ciclo de vida de un Bundle [17]

El framework OSGi en sí mismo es un bundle más que se carga al iniciar el sistema. A través de este bundle, el framework proporciona una serie de servicios básicos para que sean utilizados por otros bundles. El framework OSGi puede descargar e instalar bundles cuando se necesitan, y desinstalarlos cuando ya no se necesitan, todo ello en tiempo de ejecución sin necesidad de reiniciar el sistema entero. Todo esto permite la creación de arquitecturas muy flexibles.

Servicios

La plataforma OSGi es un ambiente donde los bundles pueden publicar, compartir y utilizar servicios. Estos servicios son objetos que proporcionan aplicaciones o algún tipo de funcionalidad. Cada bundle se encarga de registrar en el módulo de servicios del framework sus propios servicios, otro bundle podría por ejemplo, buscar un determinado servicio en el sistema y utilizarlo. La semántica y sintaxis de un servicio se especifica en una interfaz Java. En él, se declaran los métodos públicos que ofrece el objeto que representa el servicio. Diferentes bundles pueden implementar el mismo servicio.

Se puede ver en la figura 4 como estaría desplegado un framework OSGi completo sobre una plataforma de servicios genérica. La plataforma estaría compuesta, desde el nivel más bajo al nivel más alto; por el hardware seguido del sistema operativo que lo gestiona, así como los recursos software del ordenador; sobre el sistema operativo y para que pueda ejecutarse el framework OSGi debe existir un entorno de ejecución Java que proporciona la JVM. De esta forma sería posible la ejecución de bundles en el sistema que puedan registrar y hacer uso de servicios, notad incluso que los propios bundles podrían interactuar con aplicaciones instaladas en el S.O. o incluso librerías nativas de él.

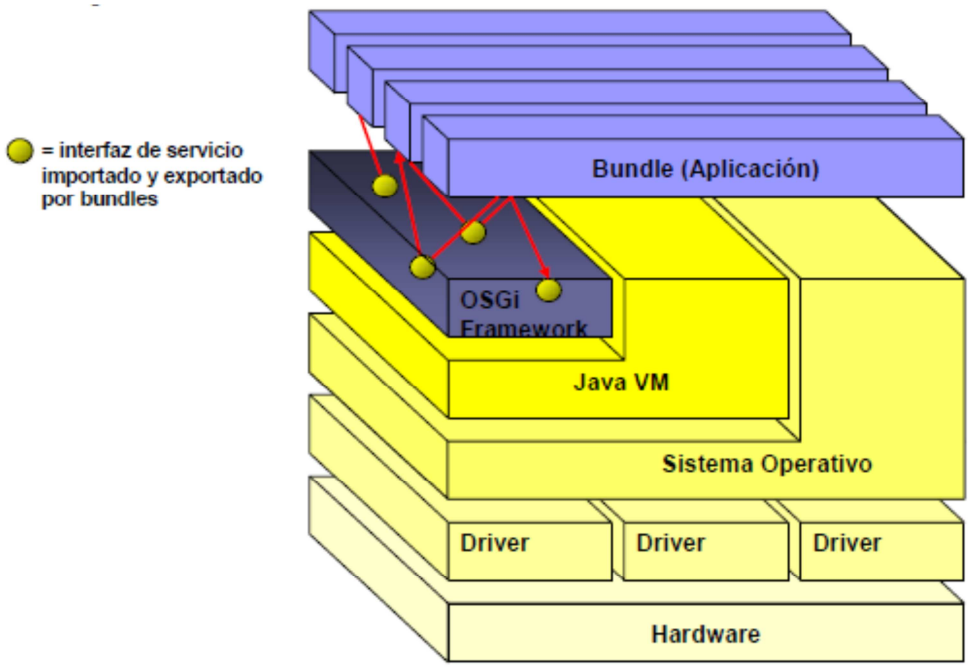

Figura 4: Plataforma OSGi [18]

2.2.2. Características

Los puntos donde se vuelcan todos los esfuerzos de mejora de las especificaciones OSGi son:

- **Estandarización**. Para que los fabricantes de equipos y los proveedores de servicios tengan una plataforma común sobre la que ofrecer sus servicios e impedir que un único fabricante monopolice el mercado.

- **Tecnología abierta**. No se define ninguna arquitectura de red domótica ni obliga al uso de una tecnología concreta, ni ningún protocolo. Cualquier empresa puede apostar por introducir su propia tecnología al producto final guardándose que sea compatible con las API's predefinidas.

- **Orientado a servicios**. Se pretende crear una plataforma que sea capaz de procesar y tratar de forma correcta toda la información necesaria para proporcionar servicios de comunicaciones, de entretenimiento, de telecontrol o domótica, y de seguridad. Por lo tanto, la especificación OSGI debe tener los interfaces adecuados para soportar todos estos servicios sin incompatibilidades además de permitir gestionarlos de forma adecuada (incluso de forma remota).

- **Dinámica y modulable**. Permite configuración para adaptarse a las necesidades del usuario con el paso del tiempo. Facilita la modularización de una aplicación en componentes más pequeños y manejables.

- **Portable**. La tecnología tiene que ser independiente de la plataforma hardware

de forma que pueda funcionar con múltiples soluciones software. No solo la plataforma hardware, sino también debe ser independiente de las tecnologías de redes de datos y control de las viviendas, teniendo en cuenta la variedad de hogares y edificios donde este tipo de pasarelas pueden ser instaladas, esta iniciativa no se acoge a una única tecnología de conexión en red. Su objetivo es definir un interfaz común para todas ellas, dejando la responsabilidad a los fabricantes de construir los controladores adecuados para cada una de ellas.

- **Métodos de acceso**. La idea es que la pasarela OSGi sea capaz de acceder al mundo exterior (redes de datos tipo Internet) usando cualquiera de las tecnologías disponibles actualmente. La tendencia es volcar todos los esfuerzos en tecnologías de acceso de banda ancha con conexión permanente a Internet. Destacan el ADSL, el modem de Cable, o inalámbricas UMTS, LMDS. Esta tendencia hacia la banda ancha es debido, además del aumento de la implantación, a la mejora sustancial en la "Calidad del Servicio", parámetro muy de moda últimamente y que será un requisito necesario para el éxito del mercado de los servicios electrónicos.

- **Escalable**. La administración y operación del parque de pasarelas, que podría llegar a millones de abonados, debe ser flexible, personalizable y escalable acorde a las nuevas necesidades del proveedor del sistema.

- **Segura**. Se define una arquitectura software que proporciona una alta seguridad e integridad para que los proveedores puedan ofrecer múltiples servicios sobre la misma plataforma sin interferirse unos con otros.

- **Fiable**. La pasarela debe funcionar 24 horas al día, sin caídas del sistema por descuidos o provocadas malintencionadamente.

2.2.3. Especificaciones

La alianza OSGi ha ido definiendo a lo largo del tiempo diferentes especificaciones que han sido usadas para crear varias implementaciones del framework OSGi. Estas especificaciones técnicas crean un estándar abierto para desarrollar aspectos como: descarga de software, gestión del ciclo de vida de una aplicación, seguridad de la pasarela, acceso a dispositivos, gestión de recursos y funciones necesarias para la administración remota de la pasarela. La plataforma OSGi contiene, además del propio núcleo del framework, un conjunto de API's básicas para el desarrollo de servicios estándar del framework definidas en los documentes publicados por el consorcio.

Las especificaciones son fácilmente aplicables, porque la plataforma se basa en una pequeña capa que permite a múltiples componentes basados en Java, cooperar en una simple máquina virtual. Además se han definido especificaciones adaptadas a cada escenario donde se vayan a implementar. Existen diferentes versiones (*releases*) de las especificaciones de OSGi que van desde el R1 (*Release 1*) al R4 [19].

Implementaciones de OSGi

Las actuales plataformas que están certificadas por la alianza OSGi para último *release R4 V4.2* son:

- *Makewave Knopflerfish Pro 3.1* [20]
- *ProSyst Software mBedded Server 7.0* [21]
- *HitachiSoft SuperJ Engine Framework V4* [22]
- *Apache Felix Framework Distribution 3.0.8* [23]

Existen más desarrollos que implementan versiones anteriores del framework OSGi (R4, R3 y R2), como por ejemplo, Eclipse Equinox 3.2 [24] del que se habla a continuación y que implementa la R4. Además hay muchas otras implementaciones no certificadas, sobre todo en el mundo OpenSource, como Newton Project [25] o Concierg [26].

Eclipse Equinox 3.2

Esta implementación de la especificación OSGi no es solo una plataforma donde desplegar bundles, al estar integrado dentro del entorno de desarrollo Eclipse [27], provee toda una plataforma para el desarrollo de sistemas en OSGi. En realidad, en si el propio IDE (*Integrated Development Enviroment*) Eclipse está basado en OSGi. Aunque Equinox comenzó como un proyecto para sustituir el *plugin* original de tiempo de ejecución de Eclipse en la versión 3.0, las posteriores adaptaciones realizadas en ese tiempo dieron forma a un nuevo proceso en la especificación OSGi incorporando muchas de ellas en la versión R4 de OSGi. Desde entonces Equinox ha sido la implementación de referencia para esta tecnología.

Una de las razones por las que eclipse se ha extendido tanto, es por su carácter modular, ofreciendo al desarrollador una integración completa al desarrollo de bundles, especialmente al desarrollo sobre su propia implementación (en Equinox los bundles reciben el nombre de *plugins*). Es decir, que el propio Eclipse provee de un entorno de ejecución, a modo de framework para los bundles. Se pueden encontrar referencias a tutoriales sobre la creación de *plugins* en la bibliografía de este documento [28].

2.3. UPnP

La tecnología UPnP (*Universal Plug & Play*), originalmente creada por Microsoft en el año 1999, es una iniciativa de la industria diseñada para permitir la conectividad sencilla y robusta entre la electrónica de consumo, aparatos inteligentes y dispositivos móviles de diferentes proveedores.

La arquitectura UPnP se está difundiendo mucho ya que cuenta con el respaldo de la compañía de software más comercial del mundo: Microsoft, por lo que su

popularidad puede ser alta aunque su funcionamiento todavía no esté trabajando en todo su potencial. UPnP está ahora bajo el control del *UPnP Forum* [29], una organización independiente que cuenta hoy en día con más de 870 miembros de sectores tecnológicos diversos.

Los grupos de trabajo del *UPnP Forum*, han desarrollado estándares para varios tipos de dispositivos de red, como impresoras, cámaras, escáneres, equipos de red, etc. Con estos estándares, consiguen extender la funcionalidad UPnP a dispositivos que anteriormente no eran de fácil conexión/configuración y simplificar sobre todo la implementación de dispositivos de red. Se impulsa así el desarrollo de un ecosistema de apoyo a los dispositivos UPnP.

Gracias también a estos desarrollos de UPnP, se están ofreciendo nuevos modelos de uso para los usuarios. Este es el caso de los adaptadores de medios digitales (*Digital Media Adapter - DMA*) que son dispositivos que concentran flujos multimedia (audio, imagen, video y datos) de diferentes dispositivos y sirven como centro de entretenimiento en el hogar. Estos sistemas harán posible que todo el contenido multimedia en el hogar pronto se agregará en un único almacén virtual de contenido a disposición de cualquier dispositivo que este dentro de la red UPnP.

El desarrollo de la tecnología UPnP está aprovechando la actual tendencia que hay en la sociedad a instalar redes domésticas y redes de pequeñas empresas y edificios comerciales, y aparte de esto, la proliferación de dispositivos electrónicos que son dotados por los fabricantes de software lo suficientemente complejo, como para ser considerados dispositivos inteligentes. Estos dos factores son propicios para la creación de oportunidades para nuevos productos y funcionalidades basadas principalmente en la conectividad natural y transparente entre estos dispositivos. Es por ello que, desde su introducción en 1999, la tecnología UPnP ha ido ganando el apoyo de la industria como el estándar preferido para el control de dispositivos en redes domésticas (y pequeñas oficinas) basadas en IP. Por tanto esta tecnología nos permite proporcionar un marco de informática distribuida basada en tecnologías Web para redes pequeñas.

Muchas empresas y organizaciones de normalización ven UPnP como la base tecnológica del hogar digital, que permite innovadores modelos de uso, mayores niveles de automatización, y una integración más fácil de dispositivos de diferentes fabricantes.

2.3.1. Arquitectura UPnP

Su arquitectura de software abierta define una serie de protocolos y procedimientos comunes que garantizan la interoperabilidad sobre dispositivos conectados a una misma red local. Esta arquitectura se construye sobre protocolos basados en la arquitectura TCP/IP [30], y es independiente de estos. Es por ello que la tecnología UPnP tiene muy fuertes raíces en las tecnologías Web estándar. Por tanto cualquier dispositivo que soporte esta tecnología de red, podrá desplegar tecnología UPnP para anunciar y ofrecer sus servicios a los clientes de la red.

La arquitectura UPnP soporta el trabajo de una red sin configurar y automáticamente detecta cualquier dispositivo que puede ser incorporado a esta, obtiene

su dirección IP, un nombre lógico, informando a los demás de sus funciones y capacidad de procesamiento, y le informa, a su vez, de las funciones y prestaciones de los demás dispositivos.

La "arquitectura de dispositivos UPnP" (*UDA - UPnP Device Architecture*) [31] es el núcleo sobre el que están construidos los productos y sus especificaciones de servicio. Define las descripciones de los dispositivos y servicios UPnP, denominados "Protocolos de control de dispositivo" (*DCP - Device Control Protocols*) [32], y explica la estructura básica que todos los dispositivos UPnP tienen que seguir en su funcionamiento.

2.3.1.1. Componentes de una red UPnP

La *UDA* define dos tipos de agentes: los "Puntos de Control" (*Control Point*) y los "Dispositivos" (*Device*). Además los dispositivos pueden ofrecer una serie de "Servicios" que los dotan de funcionalidad y pueden modificar el estado de esos servicios a través de "Acciones". A continuación se explicará de forma general en que consiste cada entidad.

Puntos de Control

Un punto de control es una entidad en la red que invoca la funcionalidad de un dispositivo. En analogías cliente/servidor, el punto de control es el cliente y el dispositivo es el servidor. Los puntos de control pueden invocar acciones relativas a los servicios, facilitando la entrada y la recepción de parámetros y, posiblemente, un valor de retorno.

Dispositivos

Un dispositivo UPnP puede ser identificado de forma lógica, como un único tipo de dispositivo. Este puede representar a un dispositivo real o a un módulo de una aplicación que actúa como dispositivo virtual. Cada tipo de dispositivo viene definido en la especificación por:

- Un identificador UDN (*Unique Device Name*).
- Un identificador de tipo de dispositivo (Device Type)
- Una cantidad indeterminada de servicios.

Toda esta información está contenida en un documento XML que además contiene la información de descripción de sí mismo, así como el fabricante, modelo y número de serie. Este XML está incluido en la implementación del dispositivo.

Un dispositivo UPnP también puede encapsular otros dispositivos (dispositivos embebidos). Lo lógico es, que el diseñador de flexibilidad en la composición de los dispositivos a la hora de determinar la forma de usarlos en un entorno.

Servicios

Un servicio de un dispositivo debe proporcionar una funcionalidad real. Cada servicio en un dispositivo UPnP puede contener cualquier número de "Acciones". Además cada servicio puede poseer un conjunto de "Variables de estado" (nombre y tipos de datos) que definen la situación del mismo y que pueden ser modificadas por las acciones de un servicio o simplemente por el funcionamiento interno del dispositivo.

Todos los servicios UPnP están definidos por una tabla de estado, el servidor de control y el servidor de eventos. La tabla de estado mantiene el estado del servicio y lo cambia cuando sea requerido. El servidor de control recibe las solicitudes, las realiza y manda el resultado como respuesta al usuario. El servidor de eventos anuncia a la comunidad de las acciones que deben efectuar cuando cambie el estado de un servicio. Cada servicio UPnP viene definido en la especificación por:

- Un identificador USN (*Unique Service Name*).
- También tiene un identificador de tipo de servicio (Service Type).
- Un identificador URI (Uniform Resource Identifier) que identifica de forma exclusiva el servicio entre todos los demás servicios de un dispositivo.

Normalmente se definen los tipos de servicios por un comité de trabajo del Foro UPnP. Los servicios que un dispositivo debe implementar son determinados por el tipo de dispositivo.

Acciones

Las acciones son las encargadas de modificar el valor de las variables de estado de un dispositivo, y por consiguiente el estado de un dispositivo o acceder a cierta información sobre él.

Cada acción tiene un identificador único y un conjunto de parámetros de entrada que se pasan como argumento. Este argumento tiene un nombre, un valor y una dirección. La dirección puede ser de entrada o de salida (pero no ambos), dependiendo de si el argumento es pasado por la acción o es devuelto por ella. Cada acción también puede tener uno o varios valores de retorno que proporciona el resultado de la acción.

Variables de estado y eventos

Cada variable de estado tiene un nombre, un tipo, un valor por defecto, y un valor actual. También cuenta con un conjunto de tipos de datos permitidos para describir la gama de los valores de las variables admisibles. Cualquier variable de estado puede desencadenar los eventos de cambios de estado, cuando se producen estos acontecimientos está determinada por la ejecución de un servicio. Cada argumento de entrada de una acción se asocia con una de las variables de estado del servicio, denominado argumento conexo de la variable de estado. Los puntos de control se pueden suscribir para recibir las notificaciones (eventos) de cambio de las variables de estado de un servicio. Cuando el servicio detecta un cambio en una variable de estado, este avisa a cualquier punto de control que este registrado para ver esos cambios.

Por tanto un punto de control descubre dispositivos, invoca acciones en un servicio de un dispositivo, y se adhiere a las notificaciones de eventos. Un dispositivo, por el contrario, puede tener alguna respuesta al invocar sus acciones, envía eventos cuando las variables de estado cambian, y puede soportar una interfaz web para el control administrativo.

2.3.1.2. Como trabaja UPnP (Estados de la comunicación)

Cada fase o estado por el que pasa un dispositivo UPnP está relacionada con uno u otro protocolo de red y utiliza alguna de las tecnologías que aparecen descritas en el Anexo A.1 de este documento.

Dentro del funcionamiento de la tecnología UPnP podemos encontrar varias etapas o estados en la comunicación: Direccionamiento, Descubrimiento, Descripción, Control, Eventos y Presentación. En la figura 5 se muestra gráficamente el orden que siguen estas fases. Se irán ahora exponiendo cada una de ellas explicando de manera general como operan.

Figura 5: Esquema de las fases de comunicación en UPnP

Direccionamiento

Es el proceso en el que un dispositivo adquiere automáticamente una dirección única IP que las entidades pueden utilizar para comunicarse con él. Es el primer paso en el funcionamiento de un dispositivo UPnP, aunque en realidad es un aspecto intrínseco de los sistemas que funcionan sobre tecnología de comunicaciones IP. Por ello es necesario que la red soporte tecnología IP en el nivel de red de su torre de protocolos.

Descubrimiento

El mecanismo de descubrimiento de dispositivos UPnP permite a los dispositivos enviar anuncios de su presencia a todos los demás dispositivos y puntos de control de la red, haciéndoles saber de la disponibilidad de sus servicios. La fase de descubrimiento de UPnP, utiliza el protocolo SSDP (*Simple Service Discovery Protocol*). Por tanto, un dispositivo UPnP es como un mini servidor de red que puede ser descubierto y controlado por un punto de control UPnP. El proceso de descubrimiento permite a los puntos de control encontrar los dispositivos y servicios de interés y

recuperar información acerca de ellos.

El dispositivo incluye en su anuncio y/o en las respuestas al proceso de descubrimiento, el identificador del dispositivo (o servicio del dispositivo) que se está anunciando, así como una dirección URL donde se encuentra su documento de descripción XML.

Algunos aspectos de UPnP exigen que el remitente se comunique con varios destinatarios. A tal fin, UPnP se basa en una variante de HTTPU que se envía a través de multidifusión (*multicast*), llamado HTTPMU. En la mayoría de los casos, la respuesta de los mensajes se envía directamente a la fuente mediante HTTPU (HTTP sobre UDP).

El proceso de descubrimiento puede estar iniciado tanto a través del punto de control como por el dispositivo. Existen rutinas que hacen que se ejecute esta acción una vez se ha inicializado el dispositivo (o punto de control), y periódicamente de modo que el sistema sea robusto ante nuevas incorporaciones de dispositivos. Por tanto, el proceso se puede desencadenar de dos maneras:

- **Si el que inicia el proceso de descubrimiento es el punto de control**, este construye un paquete SSDP y lo envía sobre HTTPMU de tal manera que llegue a todos los dispositivos. Dicho paquete realiza una petición de anuncio a los dispositivos. El mensaje que se envían aparece en el cuadro de texto 1.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
```

Cuadro de texto 1: Mensaje M-Search

Cuando el dispositivo detecta este mensaje, construye un paquete SSDP con su URL, donde recuperar su descripción (campo '*Location*'), junto con información adicional de interés y lo envíe a la dirección *unicast* donde se encuentra el punto de control mediante un paquete HTTPU. El paquete de respuesta del dispositivo se muestra en el cuadro de texto 2.

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until
advertisement expires
DATE: when response was generated
EXT:
LOCATION: URL for UPnP description for root device
SERVER: OS/version UPnP/1.0 product/version
ST: search target
USN: advertisement UUID
```

Cuadro de texto 2: Paquete de respuesta al M-Search

- **En el caso de que el dispositivo sea el que inicie el descubrimiento,** este será el paquete mandado, mostrado en el cuadro de texto 3, utilizando de nuevo HTTPMU.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement
expires
LOCATION: URL for UPnP description for root device
NT: search target
NTS: ssdp:alive
SERVER: OS/version UPnP/1.0 product/version
USN: advertisement UUID
```

Cuadro de texto 3: Mensaje NOTIFY

Una vez llega este paquete al punto de control, éste realiza un registro del nuevo dispositivo y de su URL a partir de los campos del paquete SSDP. A partir de este momento se inicia la etapa de descripción UPnP. Cada vez que el punto de control realice una comunicación con dicho dispositivo lo hará a través un socket que construya a partir de la ya mencionada URL.

La figura 6 muestra de forma gráfica el proceso completo de descubrimiento en UPnP.

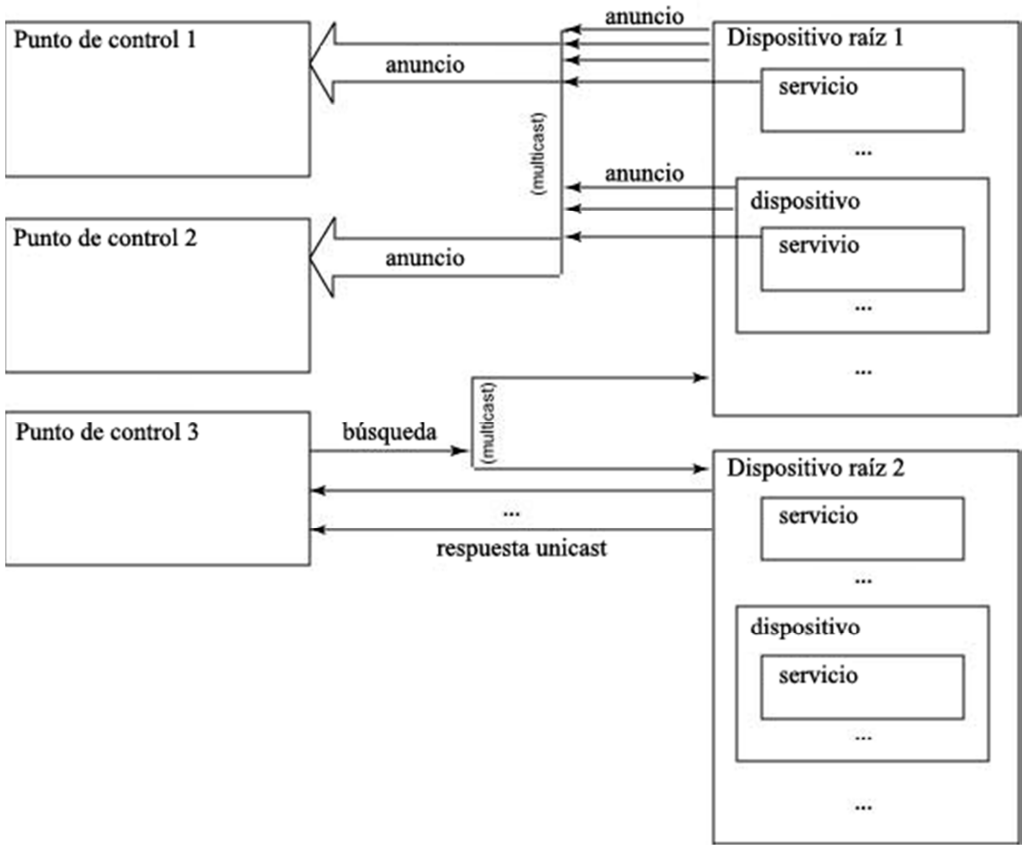

Figura 6: Proceso de descubrimiento en UPnP [33]

Descripción

Las descripciones de los productos y sus servicios se incluyen en el documento de descripción basado en XML, también llamado DCPD (*Device Control Protocol Document*). Esta descripción está dividida en dos partes, la descripción del dispositivo y

una o más descripciones de los servicios. Estas descripciones incluyen la información del dispositivo, como nombre, fabricante, marca, modelo y número de serie, así como una lista de dispositivos empotrados si existiera. La descripción de cada servicio incluye la lista de acciones y argumentos para cada acción, así como la lista de variables de estado.

Los documentos de descripción que recuperan los puntos de control son analizados para entender todo acerca de cada uno de los dispositivos. Por cada dispositivo y servicio que sea anunciado a la red habrá asociado una URL con la dirección de su documento de descripción. Todos los servicios que figuran en un dispositivo mantienen tres tipos de URL's que proporcionan la información necesaria a los puntos de control:

- ***DescriptionURL***: muestra la ubicación de puntos de control para recuperar el documento de descripción del servicio. A veces incluso después de reconocer un tipo de dispositivo UPnP, un punto de control descubre los documentos de descripción del servicio. Dado que algunas interfaces de servicios son opcionales, no todos los vendedores podrán aplicar una determinada funcionalidad. Por ejemplo, una impresora con UPnP habilitado puede que no admita la impresión en color se puede optar por no aplicar medidas para ajustar propiedades de color.

- ***ControlURL***: es la URL donde los puntos de control mandan las solicitudes de control del servicio(s). Un dispositivo UPnP especifica un proveedor para cada dispositivo.

- ***EventSubURL***: es donde los puntos de control deben mandar solicitudes para suscribirse a los eventos. Hay un *EventSubURL* para cada servicio en un dispositivo. Si este campo aparece vacío en el documento de descripción, es que el servicio no tiene gestión de eventos.

Una vez se dispone de la notificación del dispositivo, el punto de control inicia la fase en la que recopilará toda la información relativa al mismo. Dichos datos vendrán contenidos dentro del archivo de descripción XML que el dispositivo mandará al punto de control bajo demanda. Este proceso se realiza con sencillas peticiones HTTP GET, que realiza el punto de control a la URL indicada por el dispositivo en la fase de descubrimiento. El dispositivo que recibe esta petición, devuelve una respuesta con su XML de descripción. El punto de control forma un árbol con la información recogida de los documentos de descripción. En los Anexos A.2 y A.3, se muestra el aspecto que tienen los documentos XML de descripción de un dispositivo y de un servicio respectivamente. El punto de control recibe registra tanto el dispositivo como todas sus acciones además de las URL's mencionadas anteriormente, a partir de la cuales accederá tanto al control de éste y de todos sus servicios, como al control de eventos.

La figura 7 ilustra las medidas adoptadas de manera genérica por un punto de control UPnP para descubrir las capacidades de un dispositivo.

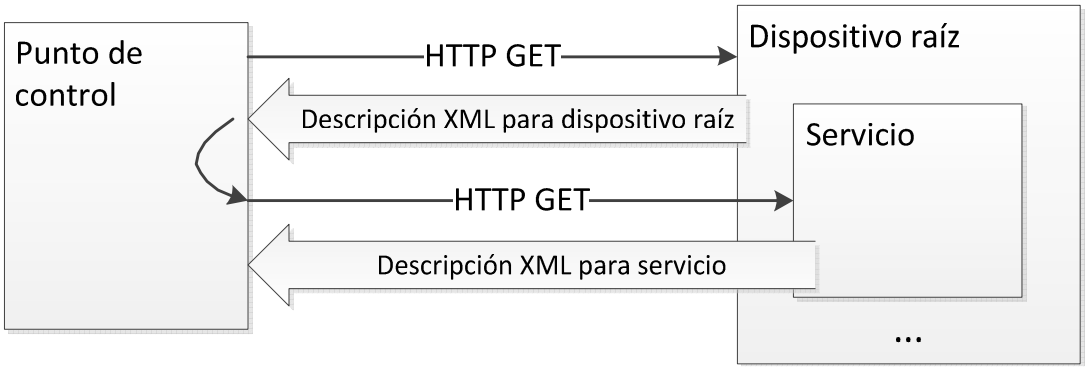

Figura 7: Recuperación de descripciones de dispositivos y servicios.

Control

En la fase de control de UPnP, los puntos de control pueden invocar las acciones de los servicios de los dispositivos. Cuando un servicio recibe un mensaje de control, el dispositivo actúa en consecuencia sobre ese mensaje. UPnP se basa en el protocolo SOAP (*Simple Object Access Protocol*) para el control del dispositivo.

Una vez registrado un dispositivo dentro del punto de control, se pueden hacer llamadas a dada uno de sus servicios de modo que éste nos devuelva el resultado de dicha ejecución. Para llevar a cabo esta acción es preciso contar con la dirección URL adecuada en la que se encuentra escuchando tal servicio, que se encuentra almacenada en el punto de control tras la etapa de 'descripción'. De este modo, la secuencia de pasos necesarios para llevar a cabo esta acción es la siguiente:

1. Envío por parte del punto de control al dispositivo del paquete encargado de realizar la llamada de alguna de las acciones de uno de sus servicios. Para tal tarea, el punto de control rellenará los argumentos de salida dentro de la petición SOAP que utiliza para comunicarse. Finalmente dicho paquete SOAP (mostrado en el cuadro de texto 4), se encapsula bajo una petición HTTP y se envía hacia la dirección y puerto correspondiente del dispositivo.

```
POST path of control URL HTTP/1.1
HOST: host of control URL:port of control URL
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:service:serviceType:v#actionName"
<?xml version="1.0"?>
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
<u:actionName xmlns:u="urn:schemas-upnp-org:service:serviceType:v">
<argumentName>in arg value</argumentName>
other in args and their values go here, if any
</u:actionName>
</s:Body>
</s:Envelope>
```

Cuadro de texto 4: Mensaje SOAP de control

2. El dispositivo, que recibe la petición correspondiente, se encarga de llevar a cabo las operaciones solicitadas y devuelve por el mismo socket la contestación SOAP (mostrada en el cuadro de texto 5) encapsulada en una respuesta HTTP.

```
HTTP/1.1 200 OK
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version
<?xml version="1.0"?>
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
<u:actionNameResponse xmlns:u="urn:schemas-upnp-
org:service:serviceType:v">
<argumentName>out arg value</argumentName>
other out args and their values go here, if any
</u:actionNameResponse>
</s:Body>
</s:Envelope>
```

Cuadro de texto 5: Respuesta al mensaje SOAP

La figura 8 ilustra de manera esquemática como sería el proceso de invocación de acciones de un punto de control sobre un dispositivo.

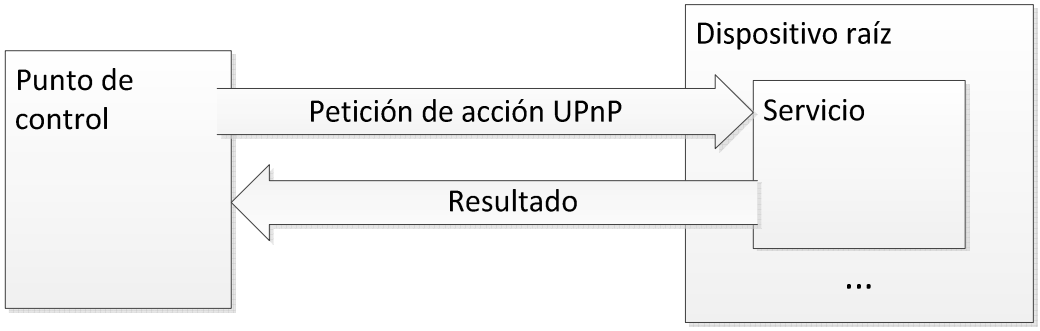

Figura 8: Proceso de control en UPnP.

Eventos

UPnP emplea un sistema de notificación de eventos mediante un modelo editor/suscriptor en el que los puntos de control pueden suscribirse a los eventos enviados por un servicio, y los servicios son capaces de publicar las notificaciones de eventos a los suscriptores. Un evento es un mensaje de un servicio a los puntos de control suscritos. Los eventos mantienen informados a los puntos de control de los cambios en el estado asociado a un servicio. Estos mensajes se expresan también en formato XML y usando GENA (*General Event Notification Architecture*).

Un punto de control interesado en recibir la notificación de cambios de variable de estado se adhiere a un evento de origen mediante el envío de una solicitud de suscripción que incluye: el servicio de interés, una URL a la que enviar los eventos y una suscripción para la notificación de eventos.

Presentación

La presentación es una fase opcional en el diseño de los dispositivos UPnP, donde estos pueden llegar a proporcionar una interfaz HTML para permitir el control manual y la supervisión del mismo por parte de un usuario o administrador. Cada dispositivo UPnP que implemente esta funcionalidad debe contener por tanto un servidor Web HTTP que pueda proporcionar una página web para clientes basados en navegador.

En esta figura 9 se muestra de forma completa la pila de protocolos que utiliza la tecnología UPnP en cada una de sus fases en la comunicación.

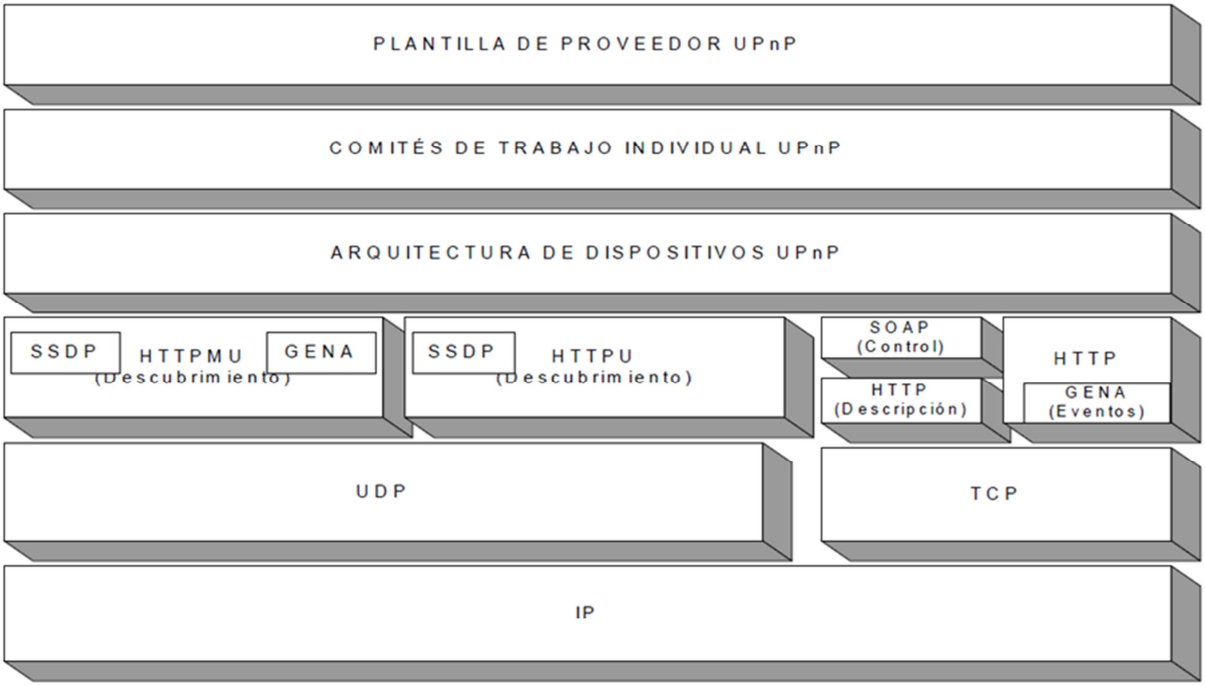

Figura 9: Pila de protocolos UPnP [34]

2.3.2. Características de UPnP

En este apartado se analizan cuáles son las características generales de UPnP, teniendo en cuenta sus prestaciones y sus limitaciones. Además se hace una comparativa con otras tecnologías existentes en el mercado que pueden solventar problemas similares que también se plantean resolver con UPnP. Por último se muestra un breve repaso de la presencia en el mercado de certificaciones comerciales que están basadas de una u otra forma en la tecnología UPnP.

2.3.2.1. Prestaciones

Las prestaciones técnicas más importantes de la arquitectura UPnP que se pueden exponer aquí son:

- ***Independencia de medios y dispositivos.*** Los dispositivos de una red UPnP funcionar sobre cualquier medio físico de comunicación, incluyendo radiofrecuencia, línea telefónica, líneas de baja tensión, Ethernet e IEEE 1394. En otras palabras, cualquier medio que puede ser utilizada para los dispositivos de red, puede permitir UPnP. Aunque no solo es independiente del medio físico, desde el punto de vista de los protocolos y las conexiones posibles a comunicarse con UPnP, soporta cualquier tecnología estándar sobre la que se pueda desplegar redes IP, o incluso a otro tipo de redes que no son IP mediante el uso de una pasarela que nos permita adaptar los distintos tipos de conexiones y la conversión entre ellas. Esto lo hace apropiado para usos en domótica, ya que con UPnP se pueden interconectar dispositivos de naturaleza muy diversa. La única preocupación podría ser que los medios de comunicación en los que se apoya utilizan el ancho de banda necesario para el uso previsto.

- ***Independencia de plataforma.*** Para construir dispositivos en esta tecnología se puede usar cualquier sistema operativo (Windows, Linux,…) y cualquier lenguaje de programación (Java, C++,…).

- ***Tecnologías basadas en internet.*** UPnP aprovecha los protocolos de comunicaciones estándares como la arquitectura TCP/IP, y metalenguajes como XML. El uso de estas tecnologías ayuda a garantizar la interoperabilidad entre las implementaciones de proveedores, ya que su uso está muy extendido por todo el mundo. Esta prevalencia se asegura de que hay un gran número de personas con conocimientos en la aplicación y el despliegue de soluciones basadas en estos protocolos.

- ***Usabilidad.*** La tecnología UPnP proporciona un importante beneficio de usabilidad. Ya se están desarrollando nuevos modelos de uso en actividades de apoyo en el hogar. En el futuro, los dispositivos de interfaz de usuario se adaptarán más a las actividades de los usuarios, ya que se desplazan más allá de la simple interfaz de teclado y ratón.

- ***Extensible.*** UPnP no especifica las API's que usarán las aplicaciones, permitiendo a los fabricantes de dispositivos que soportan esta tecnología, poder implementar servicios de valor añadido.

- ***Configuración 'cero' y Descubrimiento Automático.*** UPnP permite implícitamente la noción de una plataforma de hogar digital, en la que gran parte de la complejidad de la instalación, configuración y uso de dispositivos en red puede ser ocultado a los usuarios de la tecnología UPnP. Esto supone una abstracción que aporta coherencia a las islas, antes separadas, de la conectividad en el hogar y es capaz de enlazar las diferentes redes en el hogar en una única red lógica programable de dispositivos. Un dispositivo UPnP tiene la ventaja de poder dejar una red automáticamente y sin problemas, sin dejar detrás información estado al tratarse de una red dinámica.

- ***Estandarización.*** Al igual que la creación de estándares de Internet, la iniciativa UPnP implica una colaboración de múltiples proveedores y fabricantes bajo una serie de especificaciones comunes. Dado que los desarrolladores de software ya están familiarizados con conceptos básicos de la red y protocolos de programación, la inserción de tecnología UPnP en sus nuevos productos es muy sencilla.

2.3.2.2. Limitaciones

A continuación se expondrán una serie de limitaciones que cabe destacar sobre la arquitectura propuesta por UPnP, que aún quedan por resolver y que pueden afectar a su aplicabilidad universal. Se han separado en tres grandes ámbitos dentro del escenario en el que se mueve UPnP, que vienen definidas como limitaciones de compatibilidad, limitaciones de extensibilidad y limitaciones de seguridad.

Limitaciones de compatibilidad

Aunque se ha impulsado mucho la tecnología UPnP en los últimos años, hoy en día aún muchos dispositivos de red como pasarelas de servicios (*Gateways*) de Internet e impresoras en red, todavía requieren de instalación y configuración manual. Este hecho viene dado porque tradicionalmente los periféricos de red no han sido fáciles de instalar. Los últimos estándares como USB (*Universal Serial Bus*) y Plug-and-Play han mejorado la situación para que los dispositivos se detecten e instalen automáticamente los controladores de dispositivo, pero la transición a nuevas tecnologías como UPnP aún no se ha consolidado.

La gran dificultad de esta tecnología es que necesita que haya acuerdos en cada subsector (sistemas de impresión, almacenamiento, sector audiovisual, etc.) para consensuar las interfaces de programación de aplicaciones, ya que existen distintas implementaciones que complican la interoperabilidad de los dispositivos. Esto se puede derivar en problemas de formatos, distintos *codecs* utilizados, etc.

Algunos protocolos que utiliza UPnP, como HTTPU y HTTPMU, incluso no están estandarizados y se especifican sólo en un borrador del IETF [35] que expiró en 2001. De hecho, con el fin de implicar aún más el desarrollo de estos API's, se integró en sus inicios al grupo de trabajo HomeAPI [36], que venía trabajando desde 1997 con el objetivo de crear especificaciones y desarrollar un conjunto de servicios e interfaces de programación orientados hacia la automatización y control de las viviendas.

También una de las críticas es que UPnP es demasiado "abierto", esto quiere decir que se permite que existan en el mercado dispositivos *"UPnP compliant"*[3] que no son interoperables entre sí. Más adelante se verá que la tecnología DLNA (*Digital Living Network Alliance*) [37] trata de resolver esta problemática.

[3] Se refiere a dispositivos que intentan adaptar algunas características de UPnP sin seguir por completo la especificación.

Por otra parte, hay otras tecnologías en el mercado que tienen un nivel de madurez mayor, como es el caso de Jini y DPWS (*Device Profile for Web Services*) [38] , que también ayudan a frenar el despliegue de UPnP.

Limitaciones de extensibilidad

Desde el punto de vista técnico, la principal limitación de UPnP es que es una arquitectura de software que permite únicamente la transmisión de datos entre dispositivos IP. Debido a que en esencia está basado en SOAP, para utilización de dispositivos no IP tenemos que recurrir al SCP (*Simply Control Protocol*), esta limitación no es tan importante si tenemos en cuenta que la tendencia futura sea la integración de redes IP en todos los ámbitos.

La siguiente limitación que se plantea la tecnología UPnP es que inicialmente solo fue pensada para desplegarla sobre redes locales, pero las nuevas aplicaciones requieren extender el uso a otras redes. Es en este punto donde UPnP presenta algunos problemas para comunicar dispositivos a través de cortafuegos y enrutadores. Esto es debido a que no hay en Internet un servicio de descubrimiento adecuado, ya que el tráfico *multicast* es filtrado en estos equipos que son frontera con Internet para evitar saturar la red. Es en este factor donde se centran los esfuerzos de este proyecto para conseguir paliar esta limitación.

Empieza a ser una necesidad también, la creación de estándares para la personalización y caracterización de los dispositivos en la red del hogar en función del usuario. No solamente para aprovechar sus posibilidades sino poder adaptarse a los gustos y perfiles de los usuarios. Actualmente, el uso de los dispositivos UPnP es genérico, el dispositivo no se adapta al perfil del usuario como personas mayores, usuarios expertos, etc.

Limitaciones de seguridad

El tema de la seguridad es otra de las limitaciones de UPnP que no se abordaron al principio de su desarrollo, por lo que aún queda mucho que avanzar en este punto. La protección de los contenidos todavía se encuentra sin resolver satisfactoriamente.

El principal problema de UPnP es que no contiene ningún mecanismo de autentificación, por lo que todos los elementos de la red esencialmente se "fían" unos de otros. Esto puede representar un problema importante en muchos casos, ya que la protección de los contenidos todavía se encuentra sin resolver satisfactoriamente, por lo que los fabricantes de productos compatibles implementan a veces sus propios mecanismos de seguridad. Estos pueden ser potencialmente incompatibles con los de otros fabricantes. Existe un problema añadido, y es que UPnP no tiene un protocolo de autenticación ligero, mientras que los protocolos de seguridad disponibles son complejos. Como resultado, un dispositivo que soporta UPnP muchas veces lo mantiene desactivado de forma predeterminada como medida de seguridad.

En el apartado 2.5.1.2 se analizan posibles vulnerabilidades que pueden surgir al exponer una red UPnP local a una red externa como Internet.

2.3.2.3. Comparativas

Además de UPnP se están desarrollando otras tecnologías, relacionadas conceptualmente en mayor o menor medida con la tecnología UPnP y que pueden incluso competir con ella. Aquí se expondrán algunas de ellas:

Jini

Una de las arquitecturas que quizá pueda competir más con UPnP es **Jini,** debido a que persiguen los mismos objetivos. Es una tecnología, desarrollada por Sun Microsystems [39], aprovechando la experiencia y muchos de los conceptos en los que está inspirado, el lenguaje Java y, sobre todo, en la filosofía de la JVM. Por lo tanto, Jini puede funcionar sobre potentes estaciones de trabajo, en PC's, en pequeños dispositivos o en electrodomésticos. Gracias al lenguaje Java, la compatibilidad y la seguridad están garantizados. Jini proporciona un mecanismo sencillo para que diversos dispositivos conectados a una red puedan colaborar y compartir recursos sin necesidad de que el usuario final tenga que configurar dicha red.

La arquitectura está totalmente distribuida, ningún dispositivo hace el papel de controlador central o maestro de la red, todos pueden hablar con todos y ofrecer sus servicios a los demás. No es necesario el uso de un punto de control que controle a los dispositivos conectados a la red. Igualmente, Jini puede funcionar en entornos dinámicos donde la aparición o desconexión de dispositivos sea constante. No se puede decir en este sentido que sea una ventaja frente a UPnP, ya que en la tecnología UPnP se puede implementar a la vez el 'rol' de cliente/servidor en un mismo dispositivo.

Desde su lanzamiento y presentación en el año 1999 por Sun Microsystems, la tecnología Jini no está teniendo el éxito que se esperaba de ella. De hecho, la propia Sun así lo ha reconocido. Algunos fabricantes de dispositivos achacan este fracaso a la actitud que mantiene Sun respecto a los derechos sobre el lenguaje Java y su máquina virtual. Aunque cualquier fabricante puede usar el lenguaje Java en infinidad de aplicaciones, realmente sólo Sun o alguna empresa autorizada puede desarrollar la JVM. En este aspecto Jini está en clara desventaja frente a UPnP, ya que el despliegue de UPnP no se limita al lenguaje Java y podría estar presente en plataformas en las que no existe JVM. Pese a que Jini es un competidor de UPnP, se han creado puentes entre estas dos tecnologías para que interoperen entre sí.

DLNA (Digital Living Network Alliance)

Tecnología destacada actualmente. Esta certificación actualmente la componen más de 250 compañías, entre ellas algunas de gran renombre, como Acer, Motorola, Toshiba, Philips, Samsung, Matsushita, Hewlett-Packard, Sony, Microsoft, Intel, Pioneer, Nokia,…. Este grupo de empresas acordaron la creación de una especie de estándar compatible para todos sus sistemas, con el fin de compartir un objetivo común hacia la interoperabilidad de dispositivos conectados en las redes residenciales en las que los contenidos digitales de la electrónica de consumo se compartan dentro y fuera del hogar. Esto supone una gran ventaja a la hora de sacar productos compatibles al mercado.

DLNA utiliza un subconjunto de UPnP, que se centra en las especificaciones de dispositivos de audio y video - *UPnP AV* [40], permitiendo localizar y enlazar el resto de dispositivos compatibles en la red local. A lo que hay que añadir que no se define una interfaz para el usuario, de forma que cada dispositivo ofrece el suyo propio, lo que crea cierta complicación en ese sentido a los usuarios menos aventajados.

De alguna manera DLNA solventa el problema de seguridad de UPnP ya que hace uso del sistema de protección de datos DRM, por lo que algunos contenidos no podrán leerse desde otros dispositivos. DLNA también trata de resolver la problemática que presenta UPnP en cuanto a dispositivos de distintos fabricantes, ya mencionada anteriormente en sus limitaciones, debido a que UPnP es demasiado "abierto"**.**

DPWS (Device Profile for Web Services)

DPWS es una especificación que define un perfil de la arquitectura basada en servicios Web (*Web Services*) que permiten mensajería basada en transacciones, descubrimiento, descripción y gestión de eventos de dispositivos de red con limitaciones de recursos. Fue publicada en mayo de 2004 y su objetivo es similar a UPnP, a pesar de que emplea un modelo de servicios Web.

DPWS es parte de la tecnología Windows Rally [41]. Una librería del cliente DPWS, conocida como *WSDAPI,* es una parte de Windows 7, y los dispositivos con DPWS activado aparecen de forma automática en el explorador de red de Windows 7. Y a partir de la versión 2.5, DPWS también es parte de .NET Framework Micro [42], por lo que se puede proporcionar o consumir funcionalidad DPWS en dispositivos pequeños.

Debido a que aprovecha los últimos estándares del W3C [43] y del comité de servicios Web de OASIS [44], DPWS tiene la ventaja sobre UPnP de usar las modernas versiones de WSDL, XML y SOAP, ya que para conocer la tecnología UPnP hoy en día, se requiere el conocimiento de una variedad de diferentes protocolos más antiguos. DPWS permite que los desarrolladores de aplicaciones se centren exclusivamente en los problemas de aplicación en lugar de la comunicación a bajo nivel, la configuración y los problemas de conectividad. Aunque DPWS es una especificación relativamente nueva, las tecnologías 'Web Services' se están convirtiendo en el paradigma de computación predominante en el futuro previsible. Como principales implementaciones de DPWS están SOA4D [45] y WS4D [46].

HAVi (Home Audio/Video Interoperativity)

HAVi [47] ha sido desarrollado para cubrir las demandas de intercambio de información entre los equipos de audio y video digitales de las viviendas actuales. Es independiente del firmware usado en cada uno de los equipos, de hecho, HAVi tiene su propio sistema operativo (independiente del hardware y de la función del equipo) que ha sido especialmente diseñado para el intercambio rápido y eficaz de grandes paquetes de datos de audio y video (*streaming*).

Se ha escogido al estándar IEEE 1394 [48] (*FireWire*) como soporte físico de los paquetes de datos. Este estándar es capaz de distribuir al mismo tiempo diversos paque-

tes de datos de audio y video entre diferentes equipos de una vivienda, además de todos los paquetes de control necesarios para la correcta distribución y gestión de todos los servicios. Además el estándar HAVI ha definido (y aún está haciéndolo con otras tecnologías) un conjunto de normativas para interactuar con otras iniciativas de normalización en el mundo de Hogar Digital. Por ejemplo, HAVI define un módulo con API's que permiten a las aplicaciones formar parte de los protocolos internet más comúnmente utilizados como HTTP, FTP, POP3,…. También se ha creado un puente que permite a las aplicaciones HAVI interactuar con redes basadas en estándares como Jini o UPnP.

A simple vista HAVI parece que podría representar el despegue de las tecnologías de *webcasting* [49] y la televisión interactiva a gran escala, sin embargo, cuenta con dos limitaciones importantes: Escaso número aún de fabricantes que lo soportan; y una limitación a la capa física representada por el IEEE 1934, e interoperabilidad incierta con tecnologías inalámbricas, actualmente más solicitadas.

OpenDomo

El Proyecto OpenDomo [50] un proyecto de desarrollo libre que pretende crear un sistema de control domótico usable y seguro. Actualmente es un proyecto en activo desarrollo, y aunque todavía se encuentra en una fase temprana, ya ofrece los servicios básicos de todo sistema de control domótico. Intenta unificar las diferentes tecnologías existentes en el mundo de la domótica, como UPnP, X10, EIB,…, con el protocolo de comunicaciones más usado en la actualidad TCP/IP.

IGRS (Intelligent Grouping and Resource Sharing)

Iniciativa china fundada en julio de 2003. El grupo de trabajo de IGRS [51] es una organización de estándares y la alianza de la industria en conjunto establecida y promovida por las empresas, incluyendo Lenovo, TCL y Konka Hisense, y el número de sus miembros ha aumentado desde entonces a 115. Su objetivo principal es desarrollar un estándar que permita que los dispositivos 3C puedan trabajar juntos sin problemas y permitir a los consumidores disfrutar de interconexión sin fisuras de sus dispositivos y proporcionar servicios y aplicaciones adicionales para sus dispositivos ya existentes. La tarea central de IGRS es personalizar y promover el estándar colaborativo 3C, del que China posee sus propios derechos independientes de propiedad intelectual.

Este estándar ofrece una cierta garantía en cuanto a mecanismos de seguridad, debido al interés de que prolifere una amplia variedad de dispositivos inteligentes conectados con 3C para compartir recursos y colaboración. De alguna manera IGRS trata de una mezclar los conceptos de UPnP y DPWS.

DHWG (Digital Home Working Group)

DHWG [52] es una organización sin ánimo de lucro compuesta por más de 17 compañías, que trata de simplificar la compartición de contenidos digitales entre dispositivos móviles, dispositivos de electrónica de consumo y ordenadores en red. Este grupo tiene la meta común de, establecer una plataforma de interoperabilidad basada en estándares industriales abiertos y proporcionar normas de diseño técnico que las

empresas podrán utilizar para desarrollar productos para el hogar digital, capaces de compartir contenidos mediante redes cableadas o inalámbricas en la casa.

Los estándares industriales solos no siempre aseguran la interoperabilidad. El marco de interoperabilidad y normas de diseño técnico establecidos por el DHWG proporcionan la base para el desarrollo de productos y soluciones de múltiples marcas que funcionarán mejor juntos. Las normas de diseño del DHWG emplean estándares bien conocidos y establecidos tales como IP, UPnP y WiFi entre otros formatos comunes. Estas normas evolucionarán con el tiempo para incorporar versiones emergentes o posteriores de estándares existentes. El formato de interoperabilidad se conseguirá al requerir ciertos formatos que cumplen criterios específicos. El formato tiene que ser un estándar abierto que haya sido ratificado formalmente por la organización de estándares internacionalmente reconocida e IP tiene que licenciarse bajo términos razonables no discriminatorios. En base a estas normas, el grupo promueve un conjunto de programas incluyendo la certificación, logotipos de compatibilidad, marketing y promoción que estarán disponibles para los miembros del DHWG.

Los productos basados en normas crearán nuevas oportunidades de negocio para las industrias de dispositivos informáticos, móviles y de electrónica de consumo, acelerando así el desarrollo y entrega, por parte de las compañías asociadas, de productos audiovisuales en red incluyendo hardware, software, aplicaciones y componentes. De esta forma se ofrece a los consumidores una mayor seguridad de conectividad entre dispositivos, lo que permitirá a los clientes disfrutar fácilmente de contenidos y servicios.

Conclusiones

La tecnología actual también cuenta con una serie de inconvenientes:

- **Actualmente existen distintas implementaciones que complican la interoperabilidad de los dispositivos** que pueden derivar en problemas de formatos, distintos *codecs* utilizados, etc.
- L**a protección de los contenidos todavía se encuentra sin resolver satisfactoriamente.**
- Empieza a ser una necesidad la **creación de estándares para la personalización y caracterización de los dispositivos en la red del hogar en función del usuario**. No solamente para aprovechar sus posibilidades sino poder adaptarse a los gustos y perfiles de los usuarios. Actualmente, el uso de los dispositivos es genérico, el dispositivo no se adapta al perfil del usuario como personas mayores, usuarios expertos, etc.
- **Puede ser interesante seguir la evolución de IGRS** dado el enorme potencial del mercado chino en cuanto a electrónica de consumo. El estándar IGRS se espera que sea una norma básica y fundamental para el desarrollo futuro de la industria de la información. Después de ser aprobado como un estándar internacional, la forma de integrar aún más los recursos industriales e innovadores, es lanzar más productos IGRS y servicios y establecer la cadena

industrial basándose en la cadena de valor. China junto con otros países está luchando por la normalización internacional. Con IGRS se intenta implementar la estrategia "*go out, bringing in*" (el estándar para el mundo). La ISO y organismos de estandarización extranjeros claves, intentan fortalecer la cooperación y el intercambio de normas fundamentales estableciendo estrechos lazos.

2.3.2.4. Presencia en el mercado

La presencia de UPnP en el mercado es bastante reciente, cuando otros sistemas, ya tienen una representación más consolidada como es el caso de Jini de Sun Microsystems, el cual se encuentra presente desde hace 2 años con la consecuente experiencia tecnológica. Aún con estos indicios, UPnP cuenta con cerca de 700 empresas trabajando con este protocolo, lo que hace prever que el estándar que impulso el gigante informático Microsoft, llegará a asentarse como uno de los actores principales del Hogar Digital. El tiempo nos indicará el resultado en este caso.

El ejemplo más claro del uso de la tecnología UPnP en el mercado es la certificación **DLNA**. Está haciendo posible la interoperabilidad de muchos nuevos dispositivos pensados para ser conectados en las redes domésticas en las que los contenidos digitales de la electrónica de consumo se compartan dentro y fuera del hogar. Algunas de estas nuevas funcionalidades y servicios que ofrecen estos dispositivos, que ya existen en el mercado, son:

- Sistemas de control remoto, que surgen al automatizar algunas aplicaciones domesticas (aire acondicionado/calefacción, iluminación, seguridad/vigilancia, electrodomésticos, dispositivos multimedia,…).

- La nueva era del entretenimiento Digital: La tecnología UPnP abre un abanico de posibilidades en el que la música y otros contenidos de entretenimiento digital son accesibles desde diferentes dispositivos en el hogar sin tener en cuenta los medios donde se almacenan. Algunas de estas aplicaciones, que podemos ver ya en los hogares son: la interacción con juegos online basados en Internet, la reproducción de contenidos multimedia a través de la TV u otros dispositivos servidores de contenidos, que se puedan encontrar en diferentes puntos del inmueble. Por ejemplo, es una realidad ya el uso de cámaras fotográficas digitales con interfaces WiFi capaces de poder conectarse a una red que soporte UPnP, de esta forma la cámara subirá las imágenes a una ubicación de almacenamiento central en el hogar automáticamente, y las imágenes estarán disponibles para su visualización en cualquier dispositivo capaz de visualizar imágenes. Es el caso de dispositivos como los Media-Center que permiten una plataforma de almacenamiento e intercambio de medios digitales a todos los dispositivos conectados a él.

- Compartir información entre dispositivos y con la Web. Esto incluye el intercambio estructurado y seguro de datos digitales para apoyar el comercio electrónico.

- Sistemas de teleasistencia y telemedicina. Ayuda a las personas discapacitadas para interactuar con el entorno que les rodea a través de dispositivos de acceso personal que responden a sus necesidades.

Aun así, UPnP continúa siendo desarrollado activamente. A finales de 2008, el Foro UPnP ratifico como sucesor de UPnP 1.0, UPnP 1.1. Todo esto a pesar de que el estándar DPWS era un candidato favorito a sucesor de UPnP, pero UPnP 1.1 fue seleccionado por el Foro.

2.4. Tecnología IGD de UPnP

IGD son las siglas de **Internet Gateway Device** (Dispositivo de puerta de enlace a Internet) que designan a esta especificación de dispositivo UPnP. Debido a la importancia que tiene este dispositivo en el desarrollo del proyecto, se va a explicar en detalle el funcionamiento de algunos de sus servicios que se utilizarán en la plataforma.

El concepto de IGD proviene, como su nombre indica de dotar de funcionalidad UPnP a los *Gateways*[4] o pasarelas, ya que es el principal elemento basado en IP para el funcionamiento de una red de comunicaciones en la que están presentes diferentes tecnologías a nivel físico y de enlace.

El gateway es normalmente un equipo configurado para dotar a las máquinas de una red local (***LAN*** - *Local Area Network*), conectadas a él, de un acceso hacia una red exterior, generalmente realizando para ello operaciones de traducción de direcciones IP (***NAT***: *Network Address Translation*) [53]. Esta capacidad de traducción de direcciones permite aplicar una técnica llamada enmascaramiento IP, usado muy a menudo para dar acceso a Internet a los equipos de una red de área local compartiendo una única conexión a Internet (***WAN*** - *Wide Area Network*), y por tanto, una única dirección IP externa. En entornos domésticos normalmente se usan los módems ADSL (o módems cable) como gateways para conectar la red local doméstica con la red que es Internet, si bien esta puerta de enlace no conecta dos redes con protocolos diferentes, sí que hace posible conectar dos redes independientes (LAN y WAN) haciendo uso de la técnica ya mencionada NAT. Se abordara en el punto 2.5 que peculiaridades tiene esta técnica NAT y cómo afecta al intentar extender UPnP más allá de las interfaces WAN.

Por tanto, la idea del IGD en el UPnP Forum fue usar este dispositivo como "borde" de interconexión entre un dispositivo UPnP residencial de la LAN y la WAN, que prevé la conexión a Internet. El gateway puede estar físicamente implementado como un dispositivo dedicado, o modelado como un conjunto de dispositivos UPnP y servicios en un PC. La mayoría de los enrutadores y cortafuegos llevan implementado UPnP en su software y se exponen a sí mismos como IGD's, permitiendo que cualquier controlador local de UPnP pueda llevar a cabo una variedad de acciones como:

[4] El Gateway es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino.

- La recuperación de la dirección IP externa del dispositivo.
- Enumerar, añadir y eliminar las asignaciones de puertos[5]. Mediante la adición de un puerto mapeado, un controlador UPnP detrás del IGD puede comunicar una dirección externa a un cliente interno.
- Inicio compartido y configurable del acceso a datos de Internet entre los dispositivos conectados en la red residencial.
- Enriquecer la experiencia del usuario final de dispositivos habilitados para UPnP:
    - Proporcionar el estado y los eventos de conexiones
    - Control de inicio y terminación de las conexiones
- Gestión de configuración de servicios de host:
    - DHCP, DNS dinámico (DDNS)
    - Preservar la capacidad de dispositivos non-UPnP para iniciar y/o compartir el acceso a Internet.

2.4.1. Arquitectura

La especificación del IGD encierra todos los subproductos y servicios para el DCP del IGD. Esta versión del DCP no cubre las redes de pequeñas empresas. No se recomienda el descubrimiento y el acceso a estos servicios desde fuera de la red casera, a menos que haya una adecuada autenticación, autorización y mecanismos de control de acceso en los dispositivos, más allá de lo que actualmente se especifica en el marco de la arquitectura UPnP.

El perfil del IGD de UPnP se aplica en muchos enrutadores de banda ancha y módems de cable o ADSL. El IGD es un contenedor que está compuesto por subdispositivos. Muchos de estos subdispositivos no son más que contenedores de uno o más servicios.

La figura 10 que aparece continuación es una ilustración conceptual de un IGD genérico que consta de uno o más Interfaces WAN y LAN físicos.

[5] El mapeo de puertos, es una técnica que se aplica como una extensión del mecanismo NAT. Consiste en configurar la puerta de enlace de tal manera que se mapeen puertos de origen y destino para enviar todos los paquetes recibidos en un puerto particular a un equipo específico de la red interna.

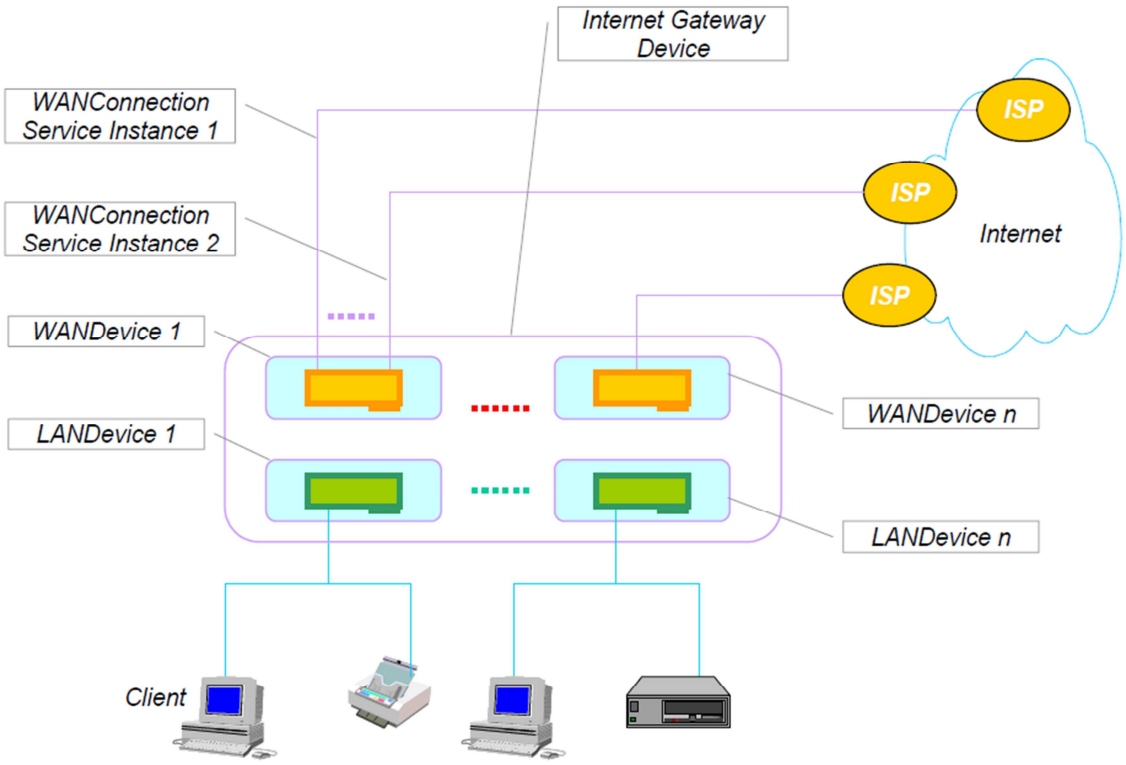

Figura 10: Modelo conceptual de un IGD [54]

Requisitos para el *IGD*

- El IGD necesita soportar al menos un interfaz físico WAN para conectarse a Internet y un interfaz físico LAN para conectar a la red residencial. Una tarjeta de red puede acoger las interfaces WAN y LAN en el mismo interfaz físico de red.

- La conectividad en el lado WAN debe permitir que los nodos de la LAN puedan acceder a los recursos de Internet. Por tanto un gateway que puede contener la funcionalidad de IGD, debe soportar módems y/o conexiones de módem a un ISP (Proveedor de servicios de Internet).

- El IGD necesita tener una IP direccionable desde la LAN residencial en todo momento para ser compatible con UPnP.

La figura 11 ilustra de forma conceptual la jerarquía de dispositivos y servicios en un IGD. Un módem físico en el lado de la WAN se modela con una instancia ***WANDevice*** y una interfaz LAN con una instancia ***LANDevice***. Dependiendo de las capacidades hardware del gateway, es posible tener más de una instancia WANDevice y/o LANDevice en la actual implementación del IGD. A su vez estos dispositivos raíz encapsulan una serie de dispositivos y servicios. Si un IGD proporciona múltiples interfaces físicas WAN UPnP a los clientes, cada uno de estos por lo general se incluirá en el documento de descripción del dispositivo WANDevice.

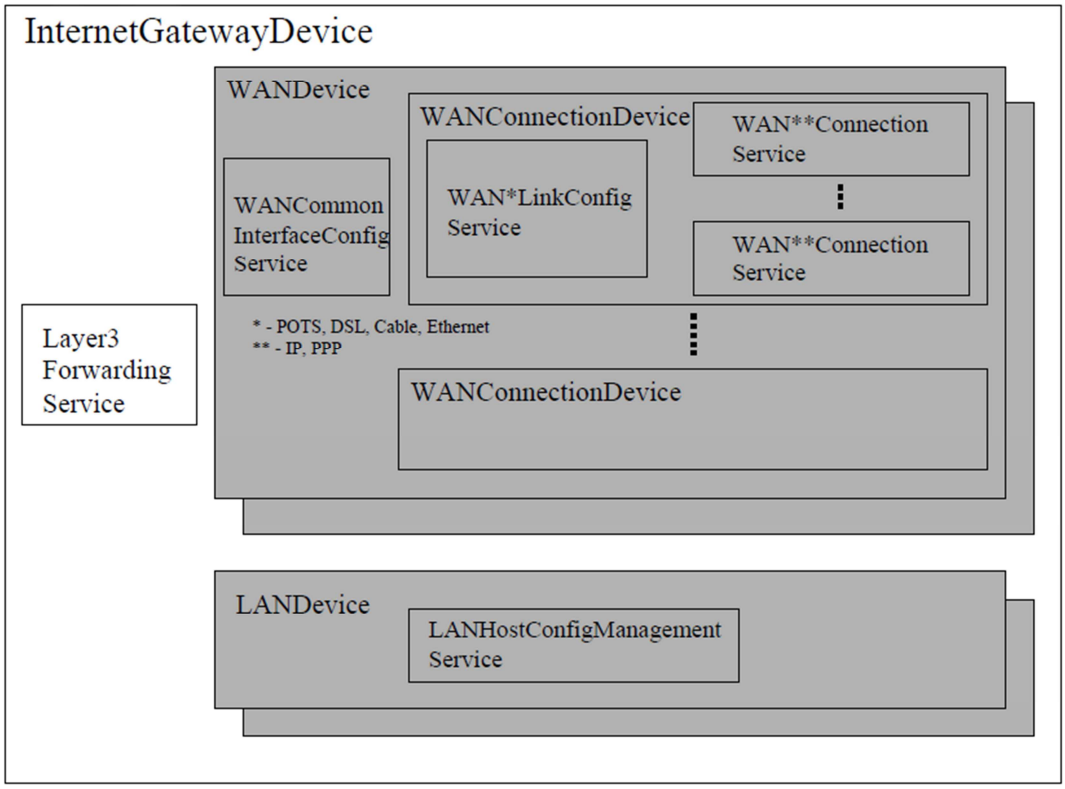

Figura 11: Jerarquía de servicios y dispositivos de un IGD [55]

Es necesario recordar que este documento es meramente orientativo, y sirve para dar una mejor idea global de la estructura de un IGD. Por tanto en este apartado se limitará a dar una breve descripción del funcionamiento solo de los servicios y dispositivos del IGD que se usarán en la implementación del proyecto. Tampoco se incluyen los esqueletos XML de descripción de los dispositivos y servicios. Para un mejor conocimiento de este dispositivo, se recomienda acudir a la especificación propuesta por el UPnP Forum sobre el IGD [56].

En concreto este apartado se centra en el dispositivo interno que tiene WANDevice llamado ***WANConnectionDevice*** y en algunos de los servicios que ofrece.

WANConnectionDevice

Cada WANDevice tiene uno o más subdispositivos WANConnectionDevice que se encargan de encapsular un enlace y un tipo específico de conexión en una interfaz WAN. La mayoría de los tipos de interfaces WAN pueden ser modelados por una sola instancia de WANConnectionDevice.

El DCP del WANConnectionDevice especifica dos conjuntos de servicios, que combinados en este subdispositivo, permiten la distinción entre protocolos de acceso a Internet que suelen ser independientes de los tipos de módems utilizados y la configuración de conexión que son específicos para cada tipo de módem. Esto permite facilitar el modelado de los distintos escenarios de conexión en un enlace WAN.

En concreto para la implementación del proyecto solo se manejarán servicios relacionados con el tipo de protocolo de acceso a la WAN. Las conexiones sobre una interfaz WAN, independientes del tipo de módem utilizado, pueden ser de tipo *PPP*[6] o *IP*, estas instancias corresponden a los modelos de servicio: *WANIPConnection y WANPPPConnection*.

A continuación se irán explicando brevemente cada uno de los servicios de conexión que existen en la especificación del dispositivo, limitándose el documento a enumerar las variables de estado, si pueden desencadenar eventos o no, así como también las acciones que son capaces de realizar.

Servicio WANIPConnection

Este tipo de servicio permite a un punto de control configurar y controlar las conexiones IP sobre la interfaz WAN de un IGD. Cualquier tipo de interfaz WAN que puedan soportar una conexión IP puede utilizar este servicio. El servicio WANIPConnection proporciona conectividad IP con un ISP para los clientes de la LAN. En el Anexo A.5, se puede observar las variables de estado que tiene este servicio. Para este proyecto solo se van a utilizar la funcionalidad de algunas de estas variables, por ello se va a dar una pequeña descripción de uso de las más relevantes:

- **ExternalIPAddress**: IP externa usada por NAT para configurar la conexión.
- **PortMappingNumberOfEntries**: Esta variable indica el número de entradas que existen en la tabla NAT para una conexión.
- **PortMappingEnabled**: Esta variable permite activar o desactivar el mapeo dinámico o estático de la NAT.
- **PortMappingLeaseDuration**: Esta variable determina el tiempo de vida en segundos de una entrada en la tabla NAT. El valor 0 indica un mapeo estático.
- **RemoteHost**: Esta variable representa la fuente de entrada de paquetes IP. En la mayoría de los casos no se usa esta variable (*String* vacío), ya que corresponde a la IP externa del gateway.
- **ExternalPort**: Puerto externo en el que el gateway "escuchará" peticiones de conexión para reenviar el trafico al correspondiente *InternalPort* en un *InternalClient.*
- **InternalPort**: Esta variable representa el puerto en el *InternalClient* al que el gateway debe enviar las solicitudes de conexión. El valor 0 no está permitido. Algunas implementaciones de NAT no permiten valores diferentes para *ExternalPort* e *InternalPort*.
- **PortMappingProtocol**: Esta variable representa el protocolo de la asignación de puertos. Los valores posibles son TCP o UDP.
- **InternalClient**: Direccion IP de un cliente interno de la LAN externa usada por NAT para configurar la conexión. Su valor es obligatorio y es posible establecer el valor de *broadcast* (255.255.255.255) para mapeos UDP. Esto permite a múltiples clientes NAT usar el mismo puerto simultáneamente.
- **PortMappingDescription**: Es una representación en *String* de un mapeo de puertos y es aplicable tanto para mapeo dinámico como estático.

[6] Acrónimo de *Point to Point Protocol*, es un protocolo de nivel de enlace estandarizado por el IETF [81].

La lista de acciones completa que provee este servicio, se pueden consultar en el Anexo A.5. Las acciones más relevantes son mostradas más en detalle a continuación, mostrando los argumentos de entrada y salida que tienen y las variables de estado a las que afecta cada acción.

Acciones más relevantes del servicio WANIPConnection

AddPortMapping

Esta acción crea un nuevo mapeo de puertos o sobrescribe una asignación existente con el mismo cliente interno. Si el par de valores *ExternalPort* y *PortMappingProtocol* ya está asignado a otro cliente interno, se devuelve un error[7]. Una vez añadido el mapeo en la tabla NAT, todo el tráfico enviado al puerto indicado en *ExternalPort* de la interfaz WAN del Gateway es reenviado a la IP del cliente interno indicada en *InternalClient* por el puerto configurado en *InternalPort*. En la tabla 1 se muestran los argumentos asociados a esta acción.

Argumento	Tipo de argumento	Variable de estado modificada
NewPortMappingIndex	Entrada	PortMappingNumberOfEntries
NewRemoteHost	Entrada	RemoteHost
NewExternalPort	Entrada	ExternalPort
NewProtocol	Entrada	PortMappingProtocol
NewInternalPort	Entrada	InternalPort
NewInternalClient	Entrada	InternalClient
NewEnabled	Entrada	PortMappingEnabled
NewPortMappingDescription	Entrada	PortMappingDescription

Tabla 1: Argumentos de la acción AddPortMapping

DeletePortMapping

Con esta acción se eliminan entradas en la tabla NAT. Por cada entrada que es borrada, la tabla es compactada y la variable *PortMappingNumberOfEntries* es decrementada. Si los argumentos de entrada no coinciden con ningún mapeo existente se devuelve un mensaje de error. En la tabla 2 se muestran los argumentos asociados a esta acción.

Argumento	Tipo de argumento	Variable de estado modificada
NewPortMappingIndex	Entrada	PortMappingNumberOfEntries
NewRemoteHost	Entrada	RemoteHost
NewExternalPort	Entrada	ExternalPort
NewProtocol	Salida	PortMappingProtocol
NewInternalPort	Salida	InternalPort

[7] Los mensajes de error en UPnP se devuelven como respuesta al mensaje SOAP, mediante el fichero XML correspondiente donde se indica el tipo de error recibido. Para más información sobre los tipos de mensajes de error que existen, se puede acudir a la referencia de la especificación de la arquitectura UPnP.

NewInternalClient	Salida	InternalClient
NewEnabled	Salida	PortMappingEnabled
NewPortMappingDescription	Salida	PortMappingDescription

Tabla 2: Argumentos de la acción DeletePortMapping

GetExternalIPAddress

Para obtener el valor de la IP externa a la conexión instanciada se usa esta acción. En la tabla 3 se muestran los argumentos asociados a esta acción.

Argumento	Tipo de argumento	Variable de estado modificada
NewExternalIPAddres	Salida	ExternalIPAddres

Tabla 3: Argumentos de la acción GetExternalIPAddress

Servicio WANPPPConnection

Este servicio es muy similar a *WANIPConnection* y permite a un punto de control configurar y controlar las conexiones PPP de la interfaz WAN de un IGD. Cualquier tipo de interfaz WAN que pueda soportar una conexión PPP puede utilizar este servicio.

En general, las conexiones a Internet se crean desde una interfaz WAN del IGD a un ISP. Sin embargo, una implementación puede soportar conexiones PPP que estén puenteadas o que sean retransmitidas (como en el caso de algunos módems xDSL) a través del IGD.

El servicio ***WANPPPConnection*** proporciona conectividad PPP con un ISP, para los clientes de la red LAN. Se pueden representar en varias instancias de este servicio sesiones PPP de varios usuarios (nombre de usuario/contraseña) con la misma conexión a un ISP, todas ellas definidas en un mismo *WANConnectionDevice*. A diferencia de *WANIPConnection*, un servicio *WANPPPConnection* normalmente exige, a priori, configuración. Debido a que este servicio comparte casi las mismas variables de estado y acciones que el anterior servicio, y las variables adicionales que propone este servicio son irrelevantes para este proyecto, se obviará en este documento esta información. Para más información sobre este servicio, se puede recurrir a las referencias bibliográficas sobre el IGD [57].

2.5. Extender UPnP a redes externas

La ubicuidad crea la demanda de acceso de banda ancha para la conexión a Internet entre varios ordenadores y dispositivos en el interior de la casa u oficina, y no solo eso, también se demanda el acceso a estos dispositivos de forma remota desde cualquier punto geográfico que tenga acceso a Internet.

La extensión de UPnP a redes externas de banda ancha, proporcionan un puente entre el hogar y los vastos recursos y servicios disponibles en otras redes principalmente Internet. Se abrirá por tanto un nuevo horizonte en el mercado, de nuevas necesidades y demandas de productos, debido al mayor conocimiento tecnológico que los potenciales usuarios poseen y a una serie de factores económicos (mayor nivel adquisitivo) y sociológicos (menos tiempo libre, ausencia de los miembros de la familia de su hogar durante la mayor parte del día), que favorecen la implantación de soluciones de gestión remota de aplicaciones a un precio asequible.

Aunque la tecnología UPnP aún no está muy consolidada en el mercado, ya que todavía queda mucho que avanzar en lo relativo a la integración en dispositivos y sistemas, se espera que esta tecnología tenga una gran aceptación en cuanto a funcionalidades que no solo limiten su actuación en redes locales, sino que además extienda sus funcionalidades a redes WAN externas. Este hecho será posible debido a la aparición de nuevas versiones en las especificaciones y nuevas tecnologías, tanto dentro de la arquitectura UPnP como en otros posibles estándares compatibles con UPnP.

Algunas aplicaciones, que ya se han mencionado en la introducción de este capítulo (televigilancia, telemedicina, telemedida, teleducación, comercio electrónico, video/audioconferencia, difusión de audio/video bajo demanda, control domótico remoto,…), están teniendo gran eco en el mercado y pueden ser motivo de extensión de la tecnología UPnP a redes externas debido al el desarrollo que suponen para la sociedad. Estos servicios propuestos potencialmente pueden tener una mayor demanda. También es cierto que surgirán aplicaciones que requieran un mayor ancho de banda. Por ello muchas empresas están trabajando en desarrollar tecnologías como UPnP, ya que lo ven como una inversión de nuevas aplicaciones en la Internet del futuro.

2.5.1. Problemas a resolver

Para analizar los problemas que surgen al intentar extender UPnP a redes externas, se distinguen tres ámbitos de observación: Análisis de problemas desde el punto de vista de extensibilidad/escalabilidad, donde se examina el hecho que supone extender UPnP más allá de la red privada; Análisis de los problemas de seguridad; Otros problemas secundarios que pueden surgir derivados de los primeros.

2.5.1.1. Problemas de extensibilidad/escalabilidad

Tradicionalmente el direccionamiento IP en Internet ha estado limitado por los rangos de direcciones disponibles en la versión 4 del protocolo IP. Es importante aquí la distinción entre direcciones IP privadas y direcciones IP públicas.

Cuando se habla de direcciones públicas en Internet suelen ser direcciones IP fijas que mantienen siempre su valor. Los servidores de correo, DNS, FTP públicos y servidores de páginas web necesariamente deben contar con una dirección IP fija, ya que de esta forma se permite su localización en la red.

Hay ciertas direcciones IP que no están asignadas públicamente en Internet, y que se denominan direcciones privadas. Las direcciones privadas pueden ser utilizadas por los equipos que usan el mecanismo NAT de traducción de direcciones para conectarse a una red pública (a través de un enrutador IP), o por los equipos que no se conectan a Internet. Es importante recordar que en una misma red IP no pueden existir dos direcciones iguales, pero sí se pueden repetir en dos redes privadas que no tengan conexión entre sí o que usen la técnica NAT.

Estos rangos de direcciones privadas se suele asignar de forma dinámica en las redes privadas (protocolos como DHCP [58] tienen esta función), es decir, son asignadas a los equipos en el momento en que lo precisan para conectarse a la red. Estas direcciones pueden usarse libremente y en la cantidad que se quiera dentro de una red privada. Esto es necesario debido al progresivo agotamiento de las direcciones IPv4, de esta manera se pueden reutilizar direcciones IP. Por ello hay muchos investigadores que se están centrando en el desarrollo de la nueva versión IPv6, que se supone que solventará muchos de estos problemas de direccionamiento. Si el lector quiere ampliar más en profundidad esta problemática debe acudir a las referencias de IPv4 que aparecen en la bibliografía [59].

De hecho aún existen algunos problemas que surgen en la migración de IPv4 a IPv6, que en principio debería haber sido transparente en la capa de aplicación en cualquier plataforma. El principal problema es que hay redes IPv4 que aún no tienen soporte para IPv6 y tienen que interactuar entre ellas. Sin embargo, los entornos IPv6 son inevitables en un futuro, debido al agotamiento de las direcciones IPv4 públicas, por lo que es necesario que cada vez más dispositivos estén preparados para ello.

El hecho de que las direcciones IPv4 públicas sean limitadas motivo el uso tan extendido de la técnica NAT en las redes LAN (y en cualquier tipo de subred privada que tenga acceso directo a una red pública). Es muy normal que el mecanismo NAT este implementado en todos los enrutadores IP que puedan soportar una interfaz WAN. Su uso más común es permitir utilizar direcciones IP privadas y aun así proveer conectividad con el resto de Internet. Por tanto el mecanismo NAT es el método por el que una dirección de IP se convierte en otra, y su introducción en IPv4 se debe a la necesidad de resolver dos problemas:

- La escasez de direcciones en el espacio de direccionamiento de IPv4.
- La interconexión de redes que no siguen el RFC 1918 [60] con redes públicas.

Las técnicas NAT suelen ser necesarias típicamente para aplicaciones de red cliente-cliente, especialmente despliegues P2P y VoIP. Existen muchas técnicas, pero no un único método que funcione en cada situación ya que el comportamiento de NAT no está estandarizado. Muchas técnicas requieren la ayuda de un servidor que sea capaz de enrutar direcciones IP públicas. Algunos métodos utilizan el servidor sólo cuando se establece la conexión, mientras que otros se basan en la retransmisión de todos los datos a través de él, los cuales añaden costos de ancho de banda y aumentan la latencia, en detrimento de la información multimedia en tiempo real.

Tradicionalmente, los enrutadores NAT presentan muchas limitaciones cuando trabajan con aplicaciones de múltiples usuarios a través de Internet. Los usuarios están

normalmente limitados al acceso de una sola estación cuando el enrutador utiliza ese tipo de software. Algunos programas de mensajería instantánea, juegos en red, sistemas de asistencia remota y un buen número de clientes P2P usan acciones, que se definen en los servicios *WANIPConnection* y *WANPPPConecction* del IGD, para facilitar la utilización de la red. Con estos servicios el IGD trabaja en torno a un problema fundamental del mecanismo NAT: 'no se puede utilizar un puerto predefinido fácilmente si se utiliza este mecanismo, ya que si más de un programa tiene la necesidad de usar dicho puerto, a menos que se use algo como un servidor Proxy[8], aparecen problemas de conexión'.

Por esta razón, muchos programas asignan dinámicamente un puerto para evitar conflictos con otro software. Para resolver este problema se debe hacer uso de las acciones de estos servicios del IGD (*AddPortMapping*, *DeletePortMapping*) que proveen también de mapeo dinámico de puertos en la tabla NAT. Lo ideal es que un programa que utilice una conexión a Internet mapee un puerto y lo elimine cuando no lo vaya a necesitar más. Muchas pilas UPnP sólo se han probado con los programas que se comportan así, por lo que muchos errores pasan desapercibidos.

Otro de los problemas importantes se debe a que originariamente UPnP surge como una arquitectura destinada a redes locales (LAN), donde el problema del direccionamiento IP no existe. Pero a la hora de utilizar ciertos protocolos que usa UPnP en redes públicas como Internet, como es el caso de HTTPMU, para la fase de descubrimiento que usa direcciones IP *multicast* [61], estos paquetes son eliminados, limitados o simplemente no soportados por algunos enrutadores de la red por razones de seguridad, complejidad, escalabilidad,….

2.5.1.2. Problemas de seguridad

La seguridad es un inconveniente importante que surge a la hora de intentar exponer redes con soporte para UPnP a redes externas, sobre la que ya se han presentado algunas limitaciones que tiene UPnP en el apartado 2.3.2.2. Este apartado se centra en analizar los principales problemas que surgen en este ámbito.

Cuando se analizan problemas de seguridad, tener en cuenta el componente humano es esencial, ya que el comportamiento humano es impredecible y gracias a él surgen los dilemas de inseguridad en las redes de comunicaciones. Por tanto siempre que se intenta resolver un problema de seguridad, no solo basta analizar protocolos, habrá que analizar en el peor de los casos, los efectos que puedan influir de forma negativa en nosotros debido a actos ajenos; y esto dependerá siempre de cada aplicación. Por ello es importante destacar el hecho de que se está realizado un considerable esfuerzo para diseñar mecanismos de seguridad que pueden ser desplegados y gestionados por los propios consumidores (es decir, los usuarios sin ningún conocimiento de redes o la seguridad de la red).

En un principio la tecnología UPnP se diseñó sin tener en cuenta ningún modelo de seguridad debido a que fue pensada para redes locales privadas, en las que en

[8]Sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

principio se puede asumir que sus usuarios son completamente confiables. Los nuevos modelos de uso que los usuarios demandan de una tecnología como UPnP, requiere de nuevos mecanismos de seguridad y modelos de autenticación e identificación de usuarios.

Es importante analizar los problemas de seguridad que se plantean en el IGD, un punto crítico en el diseño de la plataforma, en concreto se hace un análisis de los servicios que se van a utilizar: *WANIPConnection* y *WANPPPConnection*. Una vez que la red UPnP queda expuesta a una red pública, existen vulnerabilidades debido a que no posee, por defecto, ningún tipo de autentificación. Algunos posibles ataques que se pueden producir en estos casos:

- Un dispositivo con UPnP activado, puede controlar otros dispositivos UPnP tales como cortafuegos y enrutadores, de forma automática y sin autenticación, por tanto puede modificar otros parámetros en dispositivos compatibles. Mediante mensajes de control SOAP puede cambiar configuraciones de cualquier dispositivo. Por ejemplo, los programas de Adobe Flash son capaces de generar un tipo específico de solicitud HTTP. Esto permite que un enrutador que tiene activado por defecto el dispositivo UPnP IGD, sea controlado por un sitio web malicioso cuando algún usuario lo visita, devolviendo el enrutador información de su configuración sin ningún tipo de autorización. Una vez que el atacante ha conseguido información, puede modificar la configuración de conexión de nuestro enrutador con su ISP, o abrir puertos para acceder a otros equipos en la LAN.
- Otro ejemplo serían programas como Messenger de Microsoft que utilizan el IGD para abrir automáticamente puertos en el enrutador para dar servicio a aplicaciones como: Transferencia de Archivos, Webcam, Audio/Video Conferencia. Estos flujos de datos pueden ser interceptados por un tercero.
- También se le suma el factor de que muchos enrutadores permiten el uso de esta tecnología a través de redes inalámbricas, con los posibles daños que puede suponer en redes inalámbricas abiertas.

Si las nuevas infraestructuras de red adoptan un modelo basado en la seguridad, que se centre en el control de acceso, la protección proactiva y la respuesta dinámica, se podrán incorporar por ejemplo, en una red UPnP extendida, servicios multimedia en tiempo real. De esta forma se establecerán los cimientos en lo relativo a la seguridad para casi todo tipo de aplicaciones o servicios futuros.

El éxito o fracaso de poder desplegar UPnP a redes externas, depende en gran medida por tanto, de la implementación de mecanismos de seguridad acorde a lo mencionado anteriormente.

2.5.1.3. Otros problemas

UPnP reutiliza muchas ideas de la web, sin embargo, las interacciones entre los puntos de control (clientes) y dispositivos pueden ser significativamente diferentes de las interacciones de los navegadores Web y servidores. UPnP no solo se limita a

recuperar (y predecir) contenido web como hacen los navegadores, sino que necesitan estar en constante interacción con dispositivos y saber en cada momento donde se pueden localizar los recursos de estos. Cuando se trata de dispositivos móviles en redes externas, estos no siempre tienen la misma dirección IP, no siempre pertenecen a la misma red. Por tanto el problema surge debido al URI[9] (*Uniform Resource Identifier*) que debe utilizarse para hacer referencia a los recursos alojados en dichos dispositivos móviles. Ya no es válido el hacer uso de nombres de host en el URI. Los dispositivos deben estar dispuestos a modificar la URI de los contenidos y las descripciones para adaptarla a tecnología Web y mantener los recursos localizables a pesar de la ubiquidad de los dispositivos. Sería necesario en estos casos implementar en cada dispositivo un sistema que detectara el cambio de IP y cambiara la URL de su documento de descripción.

Otro problema son las configuraciones manuales de los enrutadores exigidas por los usuarios finales. Este problema se traduce en: El aumento de las llamadas de soporte técnico a los proveedores de servicios, proveedores de dispositivos cliente y los vendedores de juegos con funcionalidades online. Muchos de los puertos que se configuran de forma manual en los enrutadores se dejan abiertos, siendo un peligro en la seguridad.

El último punto o dilema a tener en cuenta es si los navegadores web pueden o no ser excelentes medios para proporcionar interfaces de usuario de tecnología UPnP. En la práctica, actualmente esto rara vez se hace. Los navegadores estándares no son una plataforma de aplicación muy atractiva debido a la dificultad de añadir protocolos, aplicaciones lógicas sofisticadas, servicios distribuidos, etc. Tecnologías como Java, AJAX, etc.; son interesantes tecnologías que pueden conducir a que los navegadores puedan actuar con más flexibilidad ante esta plataforma de computación distribuida como es UPnP.

2.5.2. Iniciativas actuales

Actualmente las iniciativas existentes sobre extender la tecnología UPnP más allá de la red privada, parten de la premisa de extender aplicaciones que surgen en el hogar u oficina como es la difusión de contenido multimedia local y la gestión domótica en general, hacia redes como Internet.

La mayoría de estas nuevas iniciativas para interconectar dispositivos a redes de comunicaciones compaginan varias tecnologías existentes aparte de UPnP como Jini, X10, Longworks, HAVI…, que están más especializadas en una u otra parte de la comunicación. Algunas iniciativas ya se han mostrado en el apartado 2.3.2.3, al comparar de forma general UPnP con otras tecnologías. También es cierto que muchas de estas iniciativas intentan conseguir una convergencia de redes heterogéneas, incluidas las de acceso inalámbrico, de área amplia (WAN), de área local (LAN) y redes de telefonía móvil. Esto propiciaría aún más el desarrollo de aplicaciones que usen UPnP, ya que hay una convergencia de redes públicas y privadas, y no existirían muchos

[9] Un URI es una cadena corta de caracteres que identifica inequívocamente un recurso. Normalmente estos recursos son accesibles en una red o sistema.

de los problemas que hemos comentado anteriormente.

Estos sistemas pueden ser desarrollados íntegramente por tecnología UPnP y son compatibles con las pasarelas residenciales, de hecho, las alternativas actuales para gestionar estas redes son mucho mayores y se han extendido a redes externas. Algunos ejemplos son: las interfaces vocales accesibles mediante las redes telefónicas, interfaces Web accesibles a través de Internet o la Intranet (basados en redes privadas virtuales), mensajes móviles que permiten informar al usuario de determinados eventos o incidencias, etc.

Pero el desarrollo aun va más allá, cuando en la vivienda entra en juego una pasarela de servicios, en vez de conectar el sistema de control centralizado a las redes telefónicas y de datos externas, es dicha pasarela la que hace de intermediaria.

A continuación se enumeran algunas de estas iniciativas que se utilizan para extender la funcionalidad de UPnP.

OSGi + UPnP

Como ya se ha comentado anteriormente en el apartado 2.2, el hecho de que OSGi sea ampliamente usado en las pasarelas residenciales confiere una flexibilidad en este sentido muy grande al combinarla con UPnP. Actualmente los proyectos que se basan en esta combinación, pueden ofrecer servicios desde redes externas a la pasarela residencial que haya en una plataforma que soporta tecnología UPnP. De esta manera la tarea del punto de control de descubrir dispositivos fuera de la LAN, podría ser cedida a los bundles del propio framework OSGi. Algunos proyectos que ya están en marcha son:

- *Arquitectura multimedia UPnP AV gestionada con un Gateway OSGi*

  Desde que la Alianza OSGi incluyo el módulo de servicio UPnP en su Release 3, ha seguido desarrollando aplicaciones para este estándar emergente en la electrónica de consumo. Como ya se ha mencionado en este documento, el servicio OSGi UPnP mapea dispositivos de una red UPnP a su registro de servicios. Alternativamente también puede mapear servicios OSGi a la red UPnP.

  En concreto la alianza OSGi está impulsando la arquitectura multimedia que ofrece UPnP. La clave de estos sistemas multimedia en una red doméstica, es la interoperabilidad entre los diferentes tipos de dispositivos multimedia. Otro factor clave es como proporcionar, para los proveedores de contenidos multimedia externos al entorno doméstico, la forma de prestar el servicio multimedia desde el exterior de una casa. Por ello, impulsado por la plataforma OSGi, se está desarrollando esta arquitectura multimedia UPnP AV con un gateway doméstico, que ofrece interoperabilidad entre los dispositivos multimedia dentro del hogar, y una poderosa herramienta para los proveedores de contenidos multimedia para proporcionar servicios multimedia inteligentes para los usuarios en el hogar. El cumplimiento de las especificaciones del estándar OSGi, permite a los usuarios descargar servicios bajo demanda de cualquier proveedor de servicios o contenidos, siendo la pasarela la que gestione la instalación y configuración de estos servicios sin interferir con el resto.

- *Proyecto de Wen-Wei Lin y Yu-Hsiang Sheng: Usar OSGi UPnP y Zigbee para proporcionar ubicuidad inalámbrica en entornos de telemedicina*

Con este proyecto [62], estos dos investigadores han diseñado un sistema para poner en contacto a los dispositivos portátiles médicos con un gateway del centro de salud a través de *ZigBee*[10] [63], ya que es una tecnología de bajo coste y provee larga duración a las baterías de los dispositivos, y además es un protocolo de transmisión inalámbrica de fácil instalación.

Combinando estas tres tecnologías (OSGi, UPnP, Zigbee), el sistema convierte los dispositivos médicos Zigbee en dispositivos UPnP que puedan ser descubiertos y configurados automáticamente. Para gestionar el software en el gateway del centro de salud se utiliza OSGi con *TR-069*[11], por lo que el proveedor de servicios puede mantener el sistema y reparar posibles problemas sin necesidad de destinar recursos en el lado del usuario.

Por otra parte, para mejorar la usabilidad, se implementa una interfaz de usuario de TV, aprovechando el beneficio de la MHP (*Multimedia Home Platform*). El usuario puede controlar los dispositivos a través de un controlador remoto del televisor, y los resultados de las mediciones que aparecen en la pantalla serán más legibles por las personas con problemas de visión.

- *Felix UPnP*

El proyecto *Félix UPnP* [64] proporciona una implementación de la especificación OSGi UPnP (versión 1.1) como se describe en el *OSGi Service Compendium* (Release 4) [65]. La especificación se implementa en el paquete *org.apache.felix.upnp.basedriver* y viene con otros paquetes, que se han desarrollado para facilitar la escritura y verificación de código de UPnP.

La especificación define un componente *UPnP Base Driver* que actúa como puente entre el software de redes UPnP y OSGi. Los desarrolladores escribiendo código UPnP, no necesitan interactuar directamente con el driver a través de alguna API. El driver trabaja en segundo plano mediante la exportación de los servicios registrados como dispositivos UPnP, y registrando servicios de los dispositivos descubiertos en redes UPnP. Sin embargo, el proyecto de Félix UPnP ha definido algunas interfaces adicionales, por lo que un conocimiento básico de la forma en que funciona UPnP Base Driver es útil y ayudará a los desarrolladores a escribir su código.

[10] Es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (WPAN).
[11] El Technical Report 069 es un estándar técnico del Broadband Forum conocido como CPE WAN Management Protocol (CWMP). Define un protocolo para el mantenimiento remoto de los dispositivos del usuario final.

UPnP + IPv6

El *UPnP Forum* dió soporte a la tecnología UPnP para las nuevas redes IPv6. Si se utiliza UPnP en una red IPv6, según las características de esta nueva versión del protocolo IP, no haría falta utilizar elementos como la NAT para poder comunicar dos entornos residenciales. Esto es debido a que ya no haría falta utilizar direcciones privadas para los dispositivos UPnP.

Existen diferentes iniciativas o estudios de investigación publicados sobre como desplegar redes UPnP sobre IPv6 completamente funcionales. De manera general, estos documentos plantean como objetivo principal la configuración completa de un entorno todo-IPv6 (*all-IPv6*); y posteriormente se plantean soluciones a los diferentes problemas en una implementación UPnP, donde un punto de control desplegado en una red IPv4 no puede comunicarse con dispositivos UPnP en una red IPv6 o viceversa. Algunas de estas iniciativas se enumeran a continuación:

- *Running UPnP under the IPv6 protocol* [66]. Este *paper* define una librería completa desarrollada en Java sobre sistemas operativos Linux, que resuelve los problemas que surgen en una red UPnP sobre IPv6.

- *UPnP Extensions for Public IPv4 Sharing in IPv6 Environment* [67]. Esta iniciativa propone una extensión del IGD de UPnP para solventar de manera eficiente el problema de los fallos de reserva de puertos en entornos de acceso a una red IPv6 que adopta un mecanismo público de compartición IPv4, por ejemplo, una dirección IPv4 es compartida por múltiples IGD's con rangos de puertos independientes.

- *UPnP IPv4/IPv6 bridge for home networking environment* [68]. En este trabajo se propone un puente UPnP IPv4/IPv6 para que los puntos de control y los dispositivos puedan interoperar entre los protocolos IPv4 e IPv6, es decir, que puedan comunicarse entre sí, aunque algunos de ellos soporten solo IPv4 y otros soporten IPv6.

- *6-Plug* [69]. El sistema 6-Plug permite un acceso IPv6 seguro a los dispositivos UPnP del entorno residencial.

DLNA + IMS

Existe una iniciativa que pone en juego la tecnología DLNA, ya comentada en este documento, con las redes IMS (*IP Multimedia Subsystem*) [70]. Debido a que DLNA se basa en UPnP, presenta los mismos problemas que éste a la hora de utilizarlo fuera de la red doméstica.

IMS ha surgido como un fuerte competidor para la gestión de los servicios multimedia en redes de telecomunicaciones. IMS proporciona autenticación, autorización y otras funciones de seguridad, así como la calidad del servicio. Estos servicios se implementan normalmente en servidores de alta disponibilidad y fiabilidad. IMS ha encontrado usos múltiples fuera de los servicios VoIP, a los que originalmente

fue destinado, por ejemplo en mensajería, información de presencia e IPTV.

El hecho de combinar la tecnología DLNA con la infraestructura IMS permite extender el acceso a los servicios fuera del entorno doméstico, permitiendo las funciones de seguridad necesarias para autenticar y autorizar a los dispositivos en la red e introduciendo el concepto de 'perfil de usuario'. Aparece el dispositivo DLNA-IMS Gateway como elemento puente para gestionar la comunicación de las dos tecnologías combinadas.

Iniciativas de seguridad

En este apartado se contemplan algunas de las iniciativas que se están llevando a cabo en cuanto a solventar los problemas de seguridad, mencionados anteriormente, en redes con tecnología UPnP.

Desde el *UPnP Forum* se decidió crear una especificación de seguridad que implementara un servicio UPnP dentro de los dispositivos, llamado servicio *Device Security* [71]. Este modelo de seguridad se estableció poco tiempo después de lanzar la versión 1.0 de la tecnología, en concreto en el año 2003. Este modelo de seguridad se centra en el funcionamiento básico de UPnP: El protocolo SOAP. El hecho de que no se hayan implantado los mecanismos de seguridad en muchos dispositivos UPnP, es debido a que las versiones de la especificación UPnP que se han utilizado para implementar su software son anteriores a los *releases* de los nuevos DCP's que han ido apareciendo a lo largo del tiempo, y no contenían especificaciones de seguridad.

También existe una iniciativa no estándar llamada UPnP-UP (*UPnP - User Profile*) [72], que propone una extensión de los dispositivos UPnP para permitir mecanismos de autenticación y autorización de usuario para sus aplicaciones. La idea es habilitar perfiles de usuario personalizado que soporten los servicios en redes con tecnología UPnP. El principal objetivo es modificar la actual especificación UPnP lo menos posible, hasta conseguir estos objetivos, y mantener la compatibilidad hacia atrás con las versiones anteriores del protocolo.

Es importante destacar aquí también la iniciativa de seguridad que se propone en la tecnología 6-plug mencionada anteriormente. Su modelo de seguridad se basa en el protocolo SSL.

2.5.3. Línea de Investigación

El hecho de poder interconectar islas de redes con soporte para tecnología UPnP, a través de redes como Internet, llevaría a crear toda clase de aplicaciones y nuevos sistemas. La línea de investigación a seguir, si se fija el lector en las tecnologías existentes que intentan extender funcionalidades UPnP a redes externas, se centraría en crear redes UPnP adaptadas a los métodos de direccionamiento IP que existen a través de Internet, siempre haciendo transparente al usuario todo el proceso de intercambio de información para descubrir nuevos dispositivos fuera del entorno doméstico o

residencial en el que se encuentra.

De esta forma se crea un nuevo modelo de utilización de dispositivos UPnP, pudiendo acceder a los dispositivos que están en un entorno doméstico desde otro entorno, por ejemplo el entorno empresarial, conectado a Internet en una localización geográficamente diferente o viceversa y todo lo que este hecho conlleva. El punto de control, de cualquiera de esas redes, debería ser capaz de descubrir estos dispositivos remotos como si se tratasen de dispositivos de su misma red local.

Siguiendo la base creada con estos objetivos comentados y en la documentación recogida en este documento, se van a exponer ahora varias posibles líneas de desarrollo destacando las limitaciones y ventajas que tienen frente a las otras expuestas en este punto.

Extender protocolos de descubrimiento UPnP actuales

Es interesante el hecho de ahondar más en las tecnologías de descubrimiento que tiene UPnP e intentar llevarlas más allá de la red local, lamentablemente no hay protocolo todavía que demuestre ser adecuado en entornos domésticos, redes empresariales y la Internet pública a la vez. Hoy en día, hay muchos y diferentes protocolos de descubrimiento y en todos ellos se utilizan diferentes tecnologías y lenguajes de descripción.

Probablemente no pueda existir un solo protocolo simple de descubrimiento, que sería adecuado para todos los tipos y tamaños de redes, pero sería muy valioso si se pudiera normalizar el lenguaje de descripción y el lenguaje de consulta.

Sin embargo, el hecho de utilizar simplemente tecnología UPnP, es una gran ventaja debido a que los fabricantes de dispositivos UPnP simplemente tienen que seguir unas especificaciones concretas a la hora de implementar la tecnología en ellos.

UPnP con soporte IPv6

En la actualidad, la razón de no centrarse en el soporte que proporciona el *UPnP Forum* sobre redes IPv6 es debido a que aún muchos de los dispositivos UPnP existentes, como ya se ha comentado en las iniciativas actuales, no tienen implementada esta funcionalidad. También es pronto para desplegar dispositivos UPnP de un entorno doméstico a una red como Internet con IPv6, debido a que las especificaciones de seguridad en UPnP deben mejorar mucho aún y aún no está maduro el despliegue de este tipo de redes en Internet debido a los problemas mencionados con IPv6.

Combinar UPnP con otra tecnología

Ya se ha visto que existen tecnologías que combinadas con UPnP, es decir, creando puentes entre ambas tecnologías, son capaces de aprovechar ventajas que por sí solas no tienen. Una buena línea de investigación se encuentra realizando alguna extensión sobre alguna de las arquitecturas de dispositivos como el IGD comentado en este documento, ayudado por tecnologías como OSGi, que se pueden desplegar

fácilmente sobre pasarelas residenciales para poder interconectar, en este caso, pasarelas de redes privadas diferentes con tecnología UPnP a través de redes públicas como Internet.

Conclusión

Las ventajas que presenta utilizar OSGi como tecnología de soporte para extender UPnP son muchas, debido a su uso cada vez mayor en aplicaciones que requieren hacer usos de servicios de Internet.

De tal forma que desplegando la funcionalidad de los puntos de control sobre pasarelas basadas en OSGi, y mediante el diseño de un mecanismo de descubrimiento que utilice protocolos (y arquitecturas de red) soportados en Internet, se pueda desplegar este sistema en redes con una localización geográfica diferente. De esta forma, un punto de control puede acceder de forma inmediata a los servicios y acciones de los dispositivos de la red remota. Todo ello simplemente con la configuración básica de conocer la dirección IP pública de la pasarela residencial, con la que se quiere interconectar.

Habría que prestar especial atención también al tema de la seguridad al intentar extender estas redes UPnP a redes públicas, ya que pueden aparecer nuevas vulnerabilidades y amenazas.

Capítulo 3

3. Diseño de la solución propuesta

En este capítulo se expone el modelo definitivo propuesto para cumplir los requisitos de este proyecto.

Primero se muestra una pequeña introducción, presentando el modelo propuesto recurriendo a las líneas de investigación abiertas para enlazar de nuevo con los objetivos del proyecto y establecer un punto de referencia al lector.

Posteriormente en el apartado 3.2, se procede a plantear de manera general los diferentes módulos que comprenden la plataforma, analizando más en detalle cada elemento clave dentro de ella en los siguientes apartados. Así como, el diseño del proceso de intercambio de mensajes entre estos elementos en el apartado 3.4.

Se destaca la función del IGD dentro de la plataforma en el apartado 3.5, ya que es la piedra angular para el funcionamiento de la misma, explicando de forma detallada cual es el su uso exacto dentro del sistema.

Por último se presentara el modelo final de la plataforma completa desplegada a través de una red pública y se analizara mediante un nivel de abstracción conceptual como interactúa extremo a extremo.

3.1. Introducción al modelo propuesto

Tras presentar en el apartado 2.5.3 varias líneas de investigación posibles para cumplir con los objetivos del proyecto, como conclusión se ha optado por combinar la tecnología UPnP con otra tecnología con gran auge entre las pasarelas residenciales como es OSGi. La combinación de estas dos tecnologías no es suficiente para cumplir todos los objetivos del proyecto, es necesario además crear un sistema de soporte de los mecanismos de descubrimiento y de descripción de UPnP, para poder proveer de interconectividad ambas pasarelas.

Para facilitar el despliegue del modelo de plataforma desarrollada para el proyecto se ha usado un entorno de ejecución de servicios OSGi. Este hecho por si solo facilita muchísimo su integración en el entorno residencial o empresarial, ya que estos sistemas existen actualmente en el mercado.

Este modelo cumple con los objetivos que se plantean resolver, contemplando la posibilidad de dar soporte a los dispositivos UPnP de cada red local[12] para poder prestar sus servicios a otra red local diferente, a través de una red pública. Este hecho se ha conseguido mediante una extensión del estándar UPnP desplegado sobre una pasarela de servicios basada en OSGi.

Es importante destacar, que este proyecto solo abarca la posibilidad de extender la funcionalidad de dispositivos UPnP entre dos redes locales conociendo, a priori, sus direcciones IP públicas que permiten su acceso desde la red que las interconecta. Con esta premisa se quiere dejar claro al lector, que el objetivo de este trabajo no es el de la seguridad. Todos los mecanismos de seguridad que han sido implementados en esta plataforma, proveen un soporte adicional a los objetivos inicialmente marcados del proyecto. Se analizará en detalle, en el capítulo de observaciones, que motivaciones existen para llevar acabo implementaciones de seguridad en la plataforma.

A continuación se muestra en la figura 12 el modelo general, a un nivel de abstracción máximo, que representa la plataforma completa.

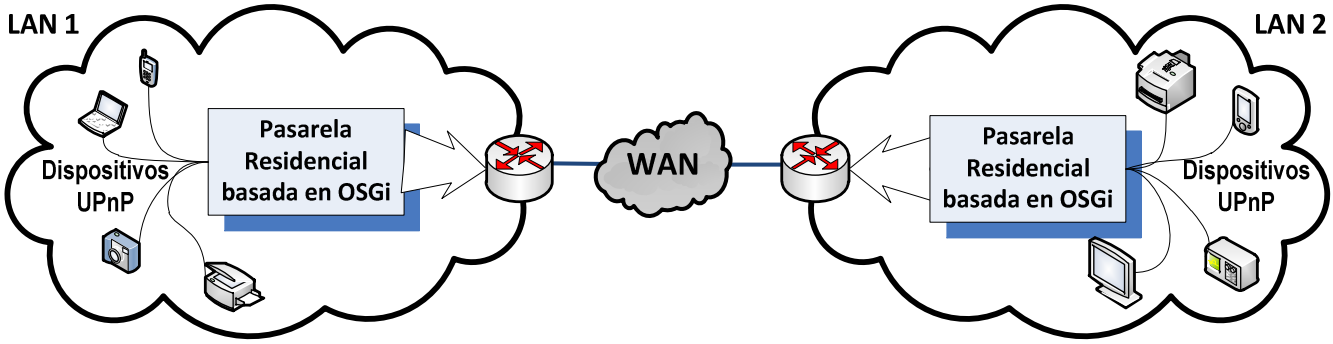

Figura 12: Modelo general de la plataforma

En la figura están representadas las dos subredes privadas conectadas a una red pública como puede ser Internet. Los sistemas que conforman la plataforma en cada red local están formados por una pasarela residencial o de servicios, basadas en un framework OSGi, y un enrutador de trafico IP como interfaz de conexión con el exterior. Estos dos sistemas pueden incluso estar físicamente en la misma máquina, aunque se ha optado por separarlos conceptualmente para explicar mejor su funcionamiento. La plataforma de una de las redes privadas con respecto a la otra, es simétrica. Por tanto las dos pasarelas pueden ofrecer la misma funcionalidad simultáneamente una vez que se ponen en funcionamiento.

En cada red local, podrán existir cualquier número de dispositivos compatibles con el estándar UPnP. Además debe existir como mínimo un IGD (basado en UPnP) en la subred, ya que es un agente fundamental en la plataforma e interviene activamente en su funcionamiento. En el caso del IGD, puede o no estar implementado dentro del

[12] Se usará indistintamente tanto el concepto de red local (LAN), como el concepto de red privada para referirnos al mismo elemento dentro de la plataforma, ya que esta red solo es accesible a través de, al menos, una dirección IP pública.

enrutador (que es lo normal), o como un dispositivo más en la red local.

También el sistema implementado puede coexistir con otros puntos de control que existan en la red local. De hecho, cuando se publiquen dispositivos UPnP externos dentro de la red local, los puntos de control locales los registrarán como si estuviesen en la misma subred, el funcionamiento será transparente para ellos.

Para que no exista confusión para el lector, cuando se habla de dispositivos internos, nos estamos refiriendo a dispositivos UPnP que están presentes físicamente dentro de la red local, de la misma forma cuando se habla de dispositivos externos, nos estamos refiriendo a dispositivos que se están ejecutando en la otra red local y a los conseguimos acceder de forma remota.

Toda la funcionalidad de soporte para el descubrimiento de dispositivos desde fuera de la red local, estará implementada dentro de la misma pasarela residencial. Este hecho proporciona algunas ventajas desde el punto de vista sistemático y de despliegue, que se comentarán más adelante al analizar en detalle las aplicaciones instaladas en la pasarela.

3.2. Diseño de la pasarela residencial

Ahora se va a proceder a analizar en detalle los módulos internos que están instalados en la pasarela residencial.

Cabe destacar que, los módulos que se ejecutan en la pasarela van a lidiar en todo momento con la información, tanto de los dispositivos internos de la subred, como con la información que les llegue de dispositivos externos a través de la interfaz WAN del enrutador.

El hecho de que se despliegue la plataforma sobre una pasarela residencial facilita mucho la instalación de este sistema, teniendo en cuenta que la pasarela residencial (como dispositivo) está presente por defecto en muchas infraestructuras de red existentes. De esta forma, no es necesario tener ninguna máquina adicional para desplegar el software de la plataforma.

En la pasarela van a interactuar dos entidades o actores, que van a ser las que dotarán de funcionamiento a la plataforma:

- **Punto de control**: Uno de los módulos corresponde a un punto de control UPnP adaptado para poder ejecutarse dentro de la pasarela como un bundle. Este punto de control será el responsable de iniciar el proceso de descubrimiento para los dispositivos UPnP que existen dentro de la LAN, aunque este hecho lo podría desencadenar otro punto de control existente o incluso los mismos dispositivos anunciándose en la red. Pero la función principal es la de interactuar con el IGD para abrir puertos en el enrutador, facilitando el intercambio de mensajes entre el interior y el exterior de la subred.

- **Virtual Device**: El otro módulo se denomina Dispositivo Virtual o *Virtual Device* y es donde se concentra toda la funcionalidad realmente del proyecto. Este módulo ha sido diseñado para satisfacer los requisitos que se plantean en el proyecto y concentrar toda la funcionalidad posible, dejando al punto de control las funciones de una red UPnP. En el siguiente apartado se explica en detalle su funcionalidad.

Estos dos módulos descritos utilizan servicios OSGi para comunicarse entre sí. En concreto la comunicación con el Virtual Device se presenta en forma de notificaciones por parte del punto de control, con las cuales realiza las acciones pertinentes en cada momento. En la figura 13 se puede apreciar una representación de los módulos que componen la plataforma y que se ejecutan dentro del framework OSGi de la pasarela.

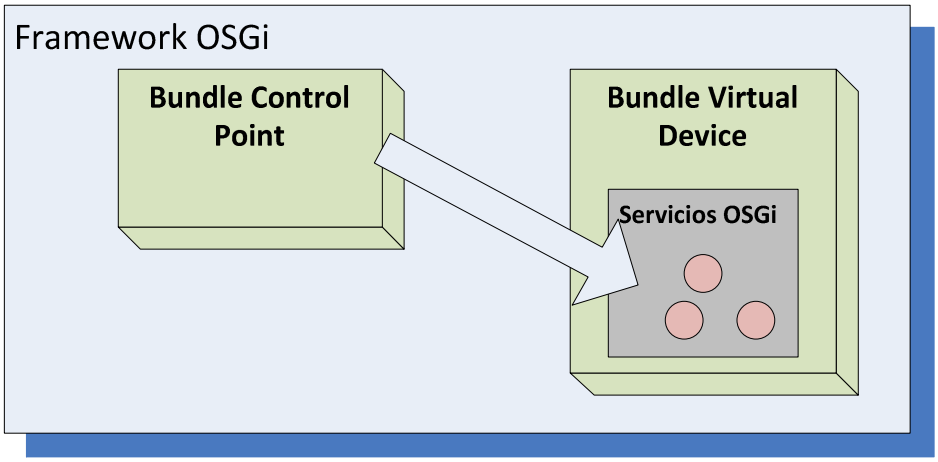

Figura 13: Componentes internos de la pasarela

3.3. Uso del IGD dentro de la plataforma

El uso del IGD en este proyecto es muy importante, debido al hecho que supone intentar que exista comunicación con un dispositivo IP concreto, que está dentro de una red local, desde la red WAN. Su uso en este proyecto es obligado debido a que este dispositivo fue diseñado para que exista conectividad IP entre la red privada y la pública, y los requisitos del proyecto lo requieren. Si al arrancar la plataforma no se encuentra ningún IGD implementado en la red local, esta no funcionará. Esto es muy importante a la hora de intercambiar mensajes en la fase de exportar dispositivos, ya que sin el IGD no sería posible, según este modelo.

El Virtual Device es el encargado de hacer de cliente del IGD para crear las traducciones de red (NAT) oportunas en el enrutador. Dicho de otra forma, es el responsable de gestionar la asignación de puertos en el enrutador que da acceso a la red WAN, utilizando el IGD. De hecho, se hace uso del punto de control, que se ejecuta en la pasarela, para acceder a la información del dispositivo y mandar a través del Virtual Device peticiones SOAP de control sobre el IGD.

Por tanto, para cada dispositivo que exista localmente, se creará utilizando el IGD, un mapeo NAT en el enrutador. Este mapeo consiste en asociar una dirección IP y puerto local, donde "escucha" el dispositivo UPnP, con la dirección IP del interfaz WAN y el mismo número de puerto interno, que tiene asociado el dispositivo, como puerto externo (debido a que algunas implementaciones de NAT en los enrutadores solo soportan esta configuración). De esta manera se permite que haya conectividad con los dispositivos desde la otra pasarela a través de la red pública.

Además el propio Virtual Device debe mapearse en el enrutador así mismo para poder ser accedido desde el exterior por el otro Virtual Device de la pasarela gemela. Esto es debido a que se ejecutan dentro de la red local con una dirección privada. Este hecho es muy importante ya que los Virtual Device de cada pasarela se intercambian la información de los dispositivos que existen en su red local.

Por motivos de seguridad, estos mapeos son dinámicos de manera que se crean en el momento en el que se hace uso de ellos, y se eliminan cuando no se están usando.

Otro uso que hace el Virtual Device también del IGD, es el de obtener la dirección IP de la interfaz WAN y así poder usarla como información para posteriormente hacer el mapeo de cada dispositivo.

3.4. Virtual Device

El Virtual Device es el nombre con el que se ha bautizado este módulo de la plataforma. Es el principal elemento de la plataforma y se encarga de gestionar todo el proceso de soporte para extender los dispositivos hacia la otra pasarela. Sus funciones son:

- Mapear, con ayuda del IGD, los dispositivos internos en la tabla NAT del enrutador.
- Tener una lista de los dispositivos internos y externos que son detectados.
- Publicar los dispositivos internos, cuando la otra pasarela lo solicite.
- Publicar los dispositivos remotos recuperados en la LAN donde se encuentra.
- Iniciar el proceso para recuperar la información de descripción de los dispositivos remotos de la otra subred.

Estas funciones se deben separar en dos procesos distintos dentro del Virtual Device. Por una parte existe la función encargada de exportar los posibles dispositivos internos de la LAN hacia la otra pasarela. En paralelo a esta, existe la función de importar dispositivos externos desde la otra pasarela. Esto es debido a que en cada red local está implementado el mismo software y cada plataforma funciona de forma recíproca, estando preparada para gestionar tanto dispositivos externos como para exportar dispositivos internos. Es decir, que cuando en una pasarela se están importando dispositivos, lo que está haciendo la otra pasarela es exportarlos hacia ésta y viceversa. En todo momento, existe un intercambio de información entre el Virtual Device y el punto de control que se ejecuta dentro de la pasarela, para dar soporte a estas dos

funcionalidades. A continuación, se explica de manera más detallada estos dos procesos dentro del Virtual Device.

3.4.1. Exportar dispositivos internos

El Virtual Device a través del punto de control, con el que coexiste en la pasarela, es capaz de detectar los dispositivos UPnP que se anuncian en la LAN y almacenar su información de descripción en una lista de "dispositivos internos". A la vez que se almacena esta información, el Virtual Device hace uso del IGD a través del punto de control para abrir puertos en el enrutador.

El Virtual Device, al iniciar su ejecución, permanece escuchando posibles mensajes de búsqueda (*Search Request*), en uno de los puertos de la interfaz de red de la pasarela, que provienen de la otra red privada. Cuando recibe algún mensaje de búsqueda, es en este momento cuando se procede a exportar las respuestas con la información de localización de los dispositivos internos a la otra LAN.

Para crear los mensajes de respuesta que viajan por la red pública, se usa la lista de dispositivos internos del Virtual Device. Los datos que se extraen de esta lista contienen información útil para la entidad gemela en la otra pasarela, como por ejemplo el puerto en el que está mapeado el dispositivo interno en el enrutador. A continuación, se va explicar qué tipo de información contiene esta lista.

Lista de dispositivos internos

En esta lista o tabla virtual de dispositivos internos se almacenan, durante el funcionamiento del Virtual Device, toda la información relacionada con los dispositivos UPnP que existen y están activos en la red local donde se ejecuta la plataforma. Cada dispositivo almacenado en la lista contiene información del identificador único (USN) y un campo que indica si es un dispositivo raíz o no; además se almacena también la URL que tiene el dispositivo configurada, donde se puede recuperar su documento de descripción. Esta URL, mediante un proceso de traducción de direcciones, es modificada por la IP y puerto públicos en la información contenida en los mensajes de respuesta que se envían a través de la WAN.

Si se quiere almacenar la información de un dispositivo que ya está en la lista, debido a que tiene el mismo identificador que una de las entradas en la tabla, la información de éste se obvia. Si el sistema tiene constancia de que un dispositivo se ha desconectado, el Virtual Device procede a eliminar la información de éste de la tabla para liberar recursos.

En la figura 14, se puede ver en forma de diagrama esquemático, cuál sería el proceso de almacenamiento de información de localización de dispositivos internos, para el proceso de exportar dispositivos, en el Virtual Device.

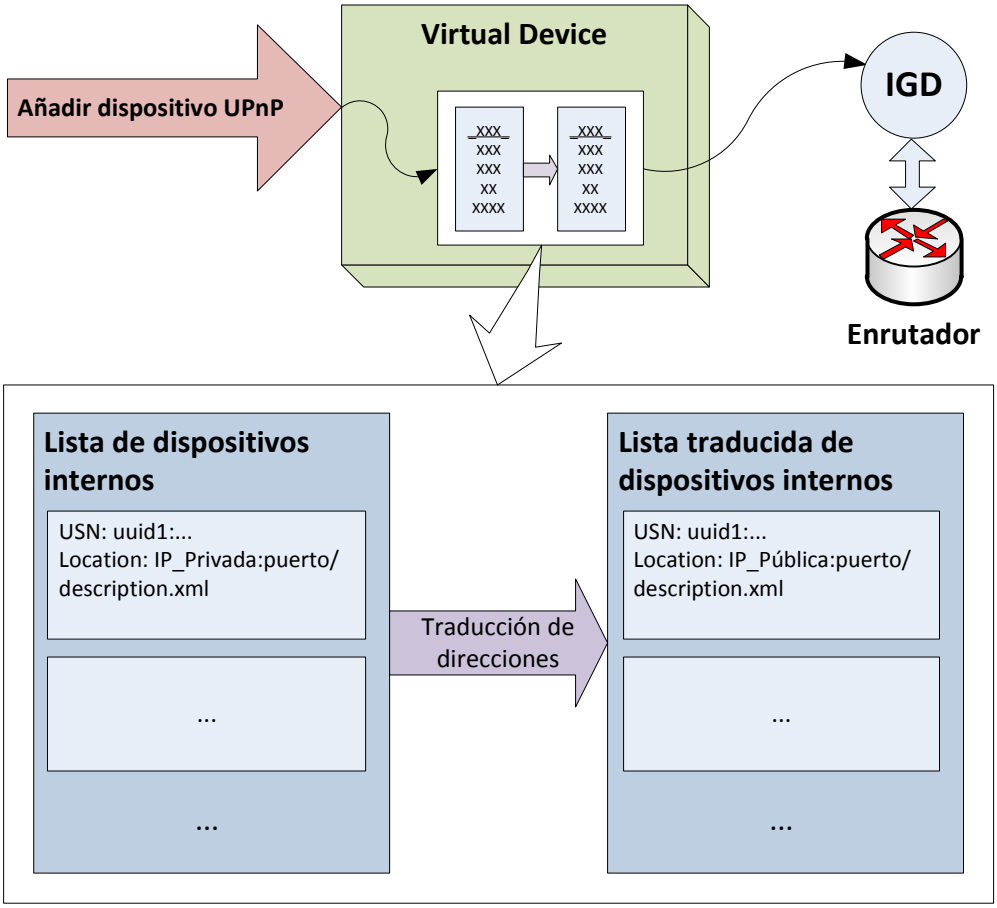

Figura 14: Proceso de traducción de direcciones de dispositivos internos

Una vez que el Virtual Device remoto recibe los dispositivos externos, desencadena su fase de descripción UPnP a través de la red pública. En este momento, los Virtual Device de cada extremo desempeñan el papel de "*man in the middle*"[13]. Esto es necesario en la etapa de descripción, debido a que los descriptores XML contienen un campo llamado *URLBase* que un punto de control toma como URL raíz ("*http://direccionIPv4:puerto"*) para acceder a las direcciones relativas de los demás recursos del dispositivo. Por tanto, el problema surge debido a que esta URL, que indica la máquina donde se ejecuta el servidor HTTP del dispositivo, corresponde a su dirección y puerto local de la red privada.

Es necesario usar de intermediarios a los Virtual Device durante la petición del descriptor XML de cada dispositivo UPnP raíz, por parte de los puntos de control de la otra pasarela. Esto se justifica porque es en este primer mensaje donde entra en juego el campo *URLBase*. Por tanto, cuando se solicite un descriptor XML por parte de un Virtual Device remoto, el Virtual Device local debe ser capaz de recuperar el descriptor del dispositivo interno correcto, y **modificar el campo *URLBase*** en su descriptor XML con la IP y puerto público desde el que es accesible, a través de la red pública. A partir de este momento, el descriptor modificado puede ser devuelto al Virtual Device remoto

[13] Es una técnica usada mucho por los hackers, y consiste en que una entidad, dentro de una red, adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido modificado.

para que lo haga llegar a su red local.

Por tanto, cuando un punto de control externo intente acceder a los servicios internos (y los subdispositivos) de un dispositivo concreto, éste debe contener el valor correcto en su campo *URLBase* almacenado para ese dispositivo. De esta manera, se soluciona esta limitación para que exista comunicación con el dispositivo de la otra pasarela, es aquí donde entra en juego la entidad traductora NAT, para redireccionar los mensajes a las direcciones privadas. En la figura 15, se puede ver de forma gráfica el proceso explicado.

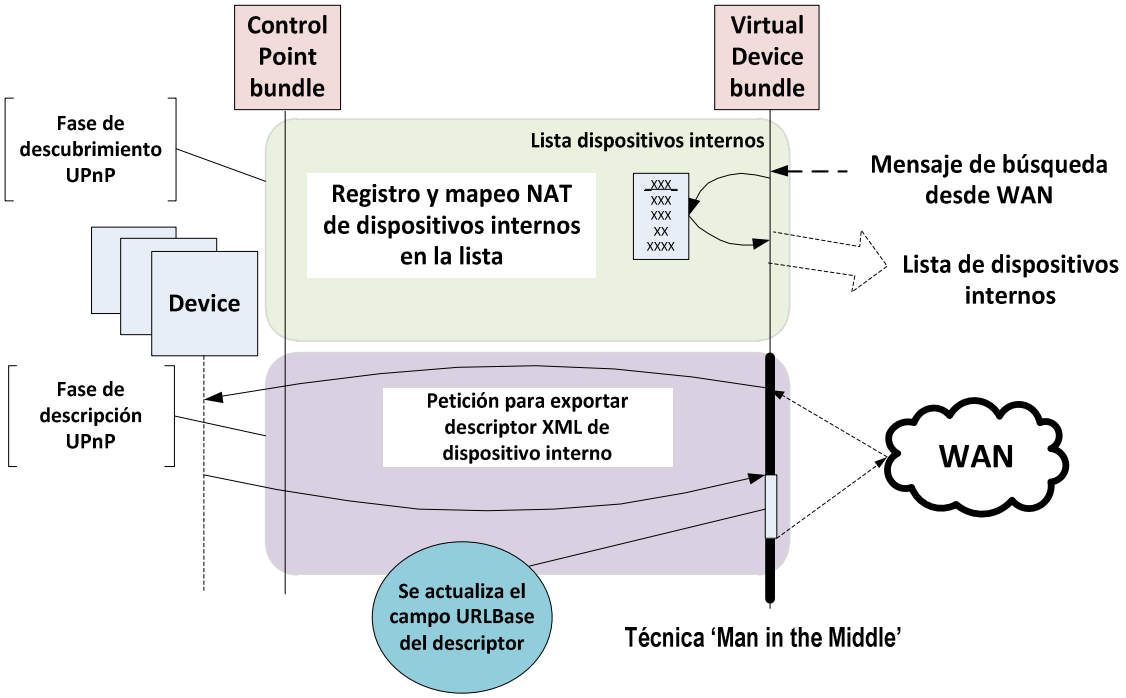

Figura 15: Proceso de exportación de dispositivos UPnP

3.4.2. Importar dispositivos externos

Además de exportar dispositivos, el Virtual Device puede importar dispositivos que no se encuentran físicamente en la LAN. Cada vez que una de las plataformas quiere importar dispositivos externos, propaga hacia la otra pasarela, a través de la red externa, cualquier mensaje de descubrimiento que se lance dentro de la red local donde se está ejecutando. El hecho de que el Virtual Device pueda localizar la descripción de los dispositivos externos y transmitirla hacia su red local, en cierta medida, es análogo al proceso de descubrimiento de UPnP y transparente a éste.

El Virtual Device es capaz de capturar la información que proviene de la otra pasarela, cuando ésta responde al mensaje de descubrimiento. Estos mensajes de respuesta corresponden a la lista de dispositivos internos que tenga el Virtual Device del otro extremo en un momento dado. Con la información que el Virtual Device extrae de cada posible mensaje de respuesta, crea una "lista de dispositivos externos", necesaria

para que éste pueda intervenir en la fase de descripción UPnP de los dispositivos externos. A continuación, se va explicar qué tipo de información contiene esta lista.

Lista de dispositivos externos

Al igual que existe una lista de dispositivos internos, el Virtual Device crea una lista de dispositivos externos donde almacena la información que proviene de los mensajes de respuesta de la otra LAN, como es el identificador único (USN) de dispositivo y su localización. Es importante destacar que en su localización, es decir, la dirección donde se encuentra su documento de descripción y el dispositivo en sí, ya está modificada por la IP y puerto externo de la otra red local. El motivo es debido a que ya está hecha la traducción de direcciones (mapeo de puertos) en el enrutador remoto.

La información almacenada en esta lista es refrescada cada vez que se reciben mensajes de respuesta a través de la WAN, esto supone no mantener información de dispositivos que se hayan desconectado en la pasarela remota.

Para que la información de los dispositivos externos llegue a los puntos de control internos de la LAN, el Virtual Device lo que hace es crear mensajes de notificación de presencia (NOTIFY) por cada dispositivo externo detectado, emulando el comportamiento de un dispositivo UPnP que se anuncia por sí mismo, como si de la misma red local se tratase. Estos NOTIFY están modificados de tal forma que, cuando los puntos de control de la red local intentan acceder a las descripciones de estos dispositivos externos, hagan las peticiones HTTP al propio Virtual Device local. Esto es debido a que las direcciones de los documentos de descripción, de los dispositivos que provienen de la otra pasarela, contienen las direcciones locales aún y son inaccesibles desde la red local.

Es en este punto, al comienzo de la etapa de descripción de los dispositivos externos, cuando el Virtual Device usa la técnica de "*man in the middle*", como ya se ha comentado en el proceso de exportación, para solicitar el descriptor remoto del dispositivo UPnP raíz a la otra pasarela.

Una vez que el Virtual Device consigue el documento XML de descripción de un dispositivo externo determinado, lo que hace éste es devolverlo al punto de control que lo ha solicitado previamente mediante un mensaje de respuesta HTTP que contiene el descriptor actualizado. Con esto se consigue que todo el proceso sea transparente para el punto de control de la LAN. De esta manera queda ligada la fase de descripción UPnP con el mecanismo de traducción de mensajes que se propone en el modelo.

3.5. Diálogo entre el punto de control y el Virtual Device

Dentro del framework OSGi, se establece un diálogo entre los bundles que

modelan la funcionalidad tanto del punto de control como del Virtual Device. El termino diálogo, se refiere conceptualmente al intercambio de mensajes e información entre estas dos entidades, que es necesario para el correcto funcionamiento de la plataforma. Esto significa que debido a su naturaleza de bundles, a veces se hace uso de sus servicios para intercambiar o notificar algún tipo de información. Este apartado, por tanto, se centra en definir la información que tienen que contener cada mensaje que se intercambia y cuál es su función.

Para explicar mejor el tipo de mensajes que existen entre las dos entidades, se diferencia entre el proceso de importar dispositivos y el de exportar dispositivos.

Intercambio de información en el proceso de exportar dispositivos

Cada vez que el punto de control de la pasarela almacena la información de descripción de un nuevo dispositivo UPnP (detectado dentro de la LAN), éste lo notifica al Virtual Device para que lo almacene en su lista de dispositivos internos. A su vez, el punto de control también notifica al Virtual Device cuando un dispositivo UPnP manda un mensaje de desconexión, para eliminarlo de la lista de dispositivos internos. Todos los mensajes mencionados hasta ahora se envían durante la fase de descubrimiento de UPnP. Es decir, que la información sobre la localización de los documentos de descripción de los dispositivos internos, debe estar disponible en el Virtual Device si se requiere exportar dispositivos a la otra LAN.

Mediante la figura 16, se muestra de forma gráfica la información intercambiada en el proceso de exportación de dispositivos, dentro de la pasarela, para la fase de descubrimiento UPnP.

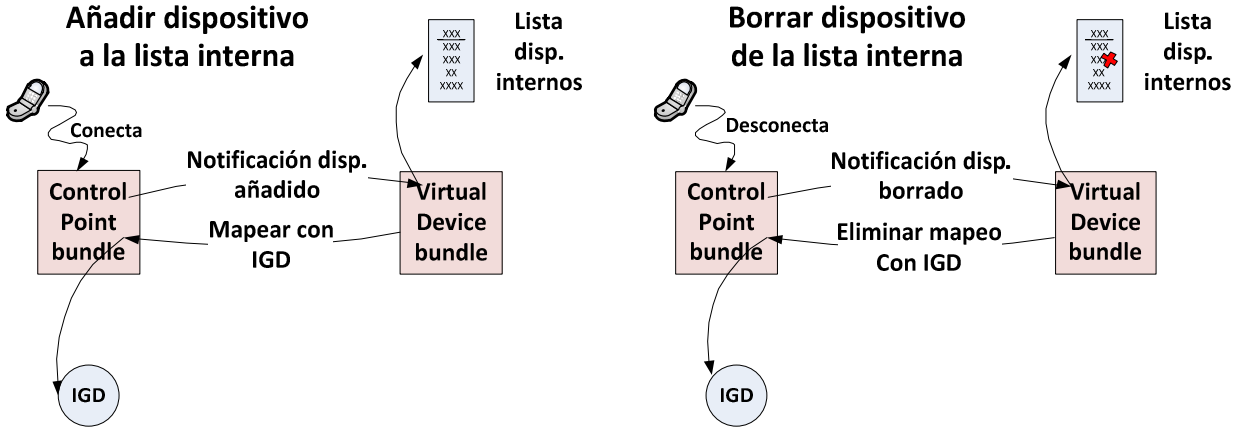

Figura 16: Diálogo entre C.P. y V.D. para agregar dispositivos internos a la lista

Intercambio de información en el proceso de importar dispositivos

El proceso de intercambio de mensajes para importar dispositivos es más complejo, y abarca las fases de descubrimiento y de descripción UPnP. En este caso no solo se hacen uso de los servicios entre bundles, sino que también el Virtual Device mandará mensajes UPnP destinados a puntos de control emulando el comportamiento de

un dispositivo UPnP común. Por este motivo, entre otros, se justifica la presencia de un punto de control dentro de la pasarela (ya que debe haber como mínimo uno para que el proceso funcione).

El diálogo se desencadena en la fase de descubrimiento UPnP, cuando el punto de control de la pasarela notifica al Virtual Device que ha mandado un mensaje de búsqueda (M-SEARCH), de nuevos dispositivos UPnP en la LAN. Esta notificación es necesaria ya que es importante que el Virtual Device replique este mensaje de búsqueda a la WAN, para importar dispositivos externos.

En el momento en el que la otra pasarela responde y empiezan a llegar las respuestas al mensaje de descubrimiento, todo el intercambio de información posterior, entre el punto de control y el Virtual Device, se hace a través de mensajes UPnP que cualquier punto de control común puede interpretar. De esta forma, pueden acceder a los dispositivos externos cualquier punto de control en la LAN. Lo que el Virtual Device está haciendo a partir de este momento es comportarse como un agente intermediario, utilizando la información de localización y descripción de los dispositivos de la otra subred como a se ha explicado en el apartado anterior.

El proceso para que exista conectividad UPnP completa finaliza, según el modelo, cuando el Virtual Device les devuelve a los puntos de control "virtualmente" el descriptor de cada dispositivo externo que han solicitado. Este documento de descripción ha sido modificado durante el proceso, actualizando algunos de sus campos, como ya se ha mencionado. Mediante la figura 17, se muestra de forma gráfica la información intercambiada en el proceso de importación de dispositivos dentro de la pasarela.

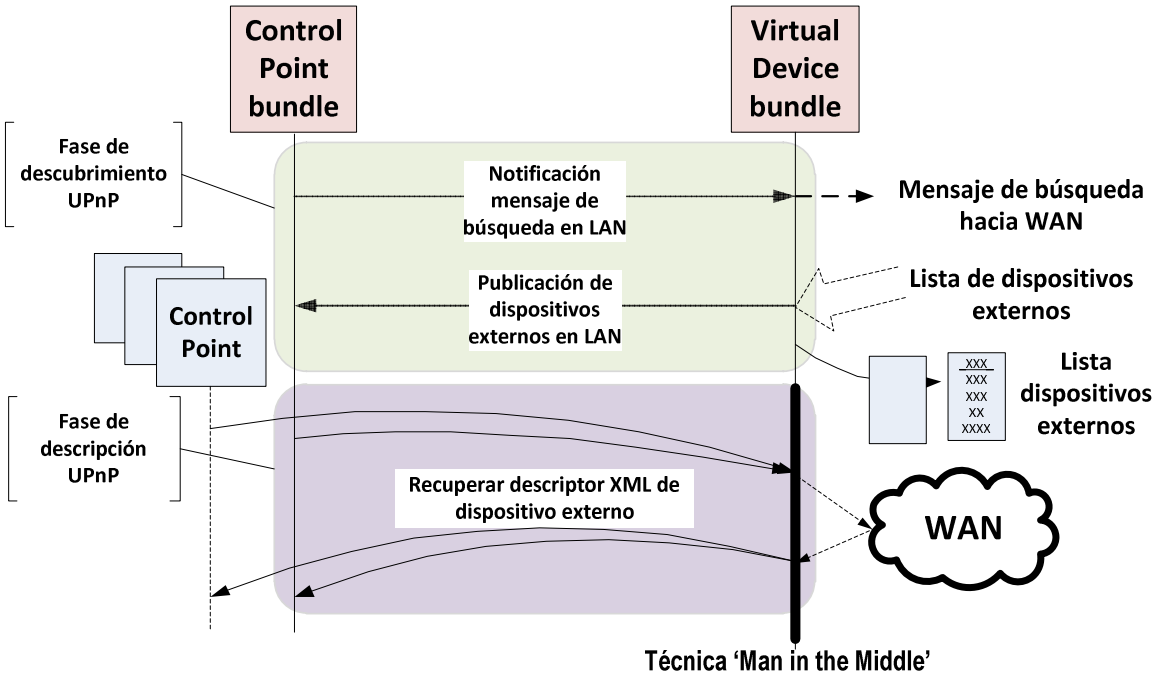

Figura 17: Diálogo entre C.P. y V.D. en el proceso de importación de dispositivos

3.6. Comunicación del Virtual Device con la pasarela gemela

El hecho de que se puedan importar y exportar dispositivos de una pasarela a otra, es debido al diálogo que se establece entre las dos implementaciones del Virtual Device. Toda esta comunicación sigue la analogía cliente/servidor, es decir, que cada Virtual Device se comporta de forma dual dependiendo de si realiza peticiones de cierta información, o es requerida cierta información de él. Para que un Virtual Device pueda iniciar su servicio de importación de dispositivos, a través de la red pública, su entidad análoga debe tener preparado el servicio de exportación en la otra pasarela, por lo que se deben haber establecido todos los mecanismos para que exista conectividad.

Hay que tener en cuenta que la información que se intercambia a través de las interfaces WAN, es visible a cualquier usuario malintencionado que pueda hacer un uso inapropiado de esta información. En vista de esta vulnerabilidad, el modelo de seguridad que se propone consiste en utilizar el protocolo SSL [73] en la información intercambiada a través de la red pública. Se ha intentado implementar este mecanismo de seguridad en las conexiones en las que se tiene control directo, pero sobre el resto de conexiones (como son la fase de control y eventos) no se tiene control directo sobre lo que se envía o recibe.

A continuación, se explicará de forma separada cual es el proceso que sigue el Virtual Device, tanto para el mecanismo de exportación como para el mecanismo de importación de dispositivos, a través de la red WAN. De esta forma, el lector tendrá una idea más clara del proceso.

Comunicación durante el proceso de exportación de dispositivos

En la fase de exportación de dispositivos, cada Virtual Device estará escuchando posibles mensajes de búsqueda de dispositivos. Al recibir este mensaje significa que la otra pasarela está intentando importar dispositivos a la red local a la que pertenece. En este momento, el Virtual Device lo que hace es devolver una copia de la lista de dispositivos internos que tiene. Esta lista contiene la información necesaria, para que el Virtual Device que lo recibe, pueda realizar correctamente el proceso de importación de dispositivos externos en su red local.

Comunicación durante el proceso de importación de dispositivos

El proceso de importación de dispositivos en una de las redes locales, se desencadena como consecuencia de la exportación de dispositivos en la otra red local. Por tanto, una vez que el Virtual Device consigue la información de los dispositivos de la otra red, éste pasa a ser intermediario en la fase de descripción UPnP, traduciendo los mensajes que intervienen en este proceso entre las dos redes locales. Todo el proceso de traducción de mensajes, se hace a través de los dos Virtual Device debido a que implementan toda la seguridad de la plataforma. De esta forma, cualquier red UPnP donde se despliegue será compatible con la plataforma. En la figura 18 se puede ver, en forma de diagrama de secuencia, cuál sería el esquema de comunicación a través de la

red pública durante todo el proceso.

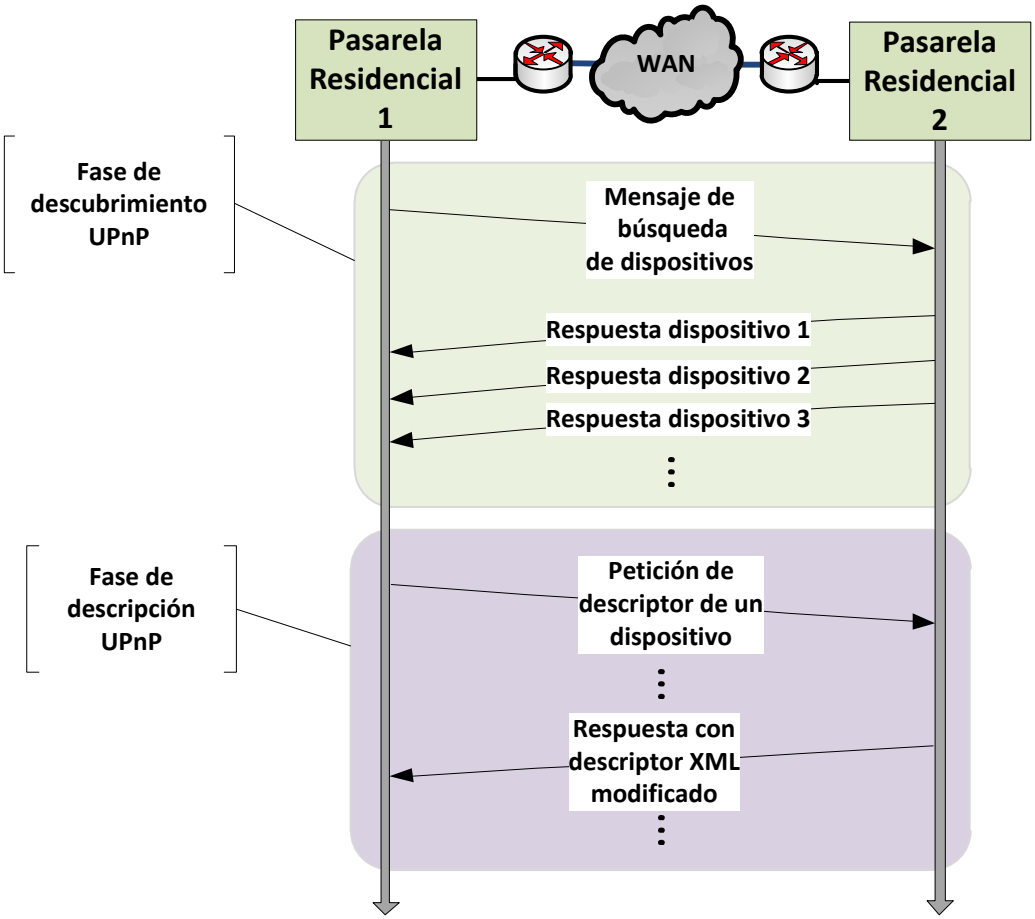

Figura 18: Diálogo WAN

Capítulo 4

4. Implementación

En este capítulo se aborda como ha sido desarrollada la aplicación en toda su fase de implementación.

El capítulo comienza con una breve introducción donde se describen, de manera general, los criterios elegidos a la hora de implementar la plataforma de soporte. Se describen también las herramientas software usadas en el entorno de desarrollo. Para terminar este apartado de introducción, por último, se describen las fases de desarrollo software que componen el proceso de implementación.

Posteriormente se profundiza en cada una de estas fases. Todo ello haciendo una síntesis de su arquitectura de clases y métodos, mostrando los principales aspectos más relevantes desde el punto de vista de su programación.

Finalmente, se muestra el despliegue completo de la plataforma, para su posterior puesta en marcha, mostrando los diagramas de secuencia de los procesos de mayor relevancia entre todos los que suceden durante su ejecución.

4.1. Introducción

El proceso de implementación de la plataforma, se ha realizado en base a los criterios establecidos por las distintas tecnologías que se han ido perfilando en el diseño del modelo propuesto. Se ha querido en todo momento seguir la filosofía del estándar UPnP, utilizando tecnologías TCP/IP para el funcionamiento de la plataforma.

La utilización de Java como lenguaje de programación ha permitido cumplir varios de los requisitos del sistema descritos con anterioridad (multiplataforma, amplia gama de librerías, interfaces gráficos,…), además como ya hemos visto en el estado del arte, OSGi está implementado en este lenguaje. Para la implementación y el desarrollo de extensiones de los elementos UPnP, se ha utilizado la implementación que propone Satoshi Konno, dentro de su proyecto *Cybergarage* [74], usando su librería desarrollada en lenguaje java *CyberlinkForJava* (ver Anexo A.4).

En este proyecto, se ha utilizado como entorno de desarrollo integrado para su implementación, el IDE de código abierto Eclipse 3.6 “Helios”. Este IDE proporciona en conjunto todas de herramientas necesarias para el desarrollo de aplicaciones en Java.

Además sirve como entorno de pruebas, ya que se ha usado su *plugin* Eclipse Equinox (ya mencionado en el capítulo 2) para desarrollar y probar bundles en un framework OSGi controlado, facilitando así la rápida depuración de errores.

Se ha pretendido implementar una estructura lógica lo más sencilla e intuitiva posible, que facilite no sólo su compresión sino también la integración de mayores capacidades en un futuro. Además se ha decidido que la estructura funcional del código corresponda, en la medida de lo posible, a la jerarquía de paquetes creados. Se ha utilizado el idioma inglés en toda la programación. Esto viene motivado principalmente por el carácter global de las tecnologías en las que se basa la aplicación, así como por guardar una mayor coherencia con la utilización de librerías Java definidas en este idioma.

Para facilitar la compresión de este capítulo, se han utilizado varios diagramas de clases que sintetizan de forma gráfica las características y arquitectura del sistema diseñado. Estos diagramas nos dan una idea de cómo está estructurada la programación de cada módulo.

En las próximas líneas, se muestra las distintas fases de desarrollo que comprenden cada una de las entidades mencionadas en el diseño de la plataforma, cada cual representa una unidad funcional dentro de la misma. Mostradas en orden de desarrollo son estas:

1. **Fase de desarrollo del bundle `org.cybergarage.stack`**: Creación de un bundle "repositorio" con la librería *CyberLinkForJava*, como repositorio de paquetes necesarios para la ejecución del punto de control y el Virtual Device (ya que usa también características de esta librería). Este bundle no tiene ciclo de vida, simplemente exporta dentro del framework los paquetes que contiene, para que otros bundles los utilicen.
   No se va a entrar más en detalle en la creación de este bundle, debido a la simplicidad que entraña este proceso. Sin embargo, la documentación su API, puede ser consultada en las referencias de este documento [75].

2. **Fase de desarrollo del bundle `org.uc3m.sergiopfc.controlpoint`**: Consiste en una adaptación de la implementación del punto de control propuesto en el proyecto *CyberGarage*, para que sea posible su despliegue dentro de un framework OSGi.

3. **Fase de desarrollo del bundle `org.uc3m.sergiopfc.virtualdevice`**: Creación de la entidad "Virtual Device" como bundle, para su ejecución dentro del framework OSGi y la implementación de los servicios asociados a sus funciones.

En cada una de estas fases, el código generado ha sido sometido a un proceso de pruebas funcionales y de integridad, realizando en todo momento depuración de errores para comprobar el correcto funcionamiento del sistema.

4.2. Adaptación del punto de control UPnP a bundle de OSGi.

Para crear del punto de control, se ha utilizado la implementación que propone a modo de ejemplo Satoshi Konno, usando su librería Cyberlink para lenguaje Java englobada dentro del proyecto Cybergarage.

El hecho de elegir esta implementación, se debe a que respeta mucho la especificación de UPnP. Además provee una interfaz gráfica muy útil para su uso, que proporciona información al usuario sobre los dispositivos UPnP registrados y los mensajes que son capturados por el punto de control. Además se pueden invocar acciones de control y eventos desde la propia interfaz gráfica. Para ello define un conjunto de métodos y clases perfectamente estructurado.

Es importante también destacar que la interfaz gráfica que provee esta implementación nos servirá también para monitorizar el estado de la pasarela en todo momento.

El proceso de integración del punto de control en el framework es sencillo, simplemente se procedió a transformar la aplicación Java normal en un bundle OSGi para poder ser integrada en una plataforma OSGi. Para ello, se utilizó el API de OSGi que existe en Eclipse Equinox, que actualmente es la especificación *OSGi R4 core framework*.

4.2.1. Arquitectura de clases

En este apartado se describen el conjunto de paquetes y clases que componen el punto de control integrado en la pasarela. También, por su importancia, se analizan algunos métodos definidos en estas clases.

En la tabla 4, se puede observar cómo está formado el conjunto de paquetes y clases de este módulo. En la columna de la derecha, aparece el nombre del paquete raíz primero, y debajo los paquetes contenidos en éste. En la columna de la izquierda, se muestran las clases contenidas en cada paquete. Aparecen marcadas las clases más relevantes para entender su funcionamiento dentro del módulo.

Paquete	**Clases contenidas**	
`org.uc3m.sergiopfc.controlpoint`	**`ControlPointActivator`**	
Paquetes contenidos		

`org.uc3m.sergiopfc.controlpoint.upnp`	`ActionDialog`
	`ActionPane`
	`ActionTable`
	`ArgumentTable`
	`CtrlPoint`
	`CtrlPointPane`
	`DeviceTable`
		`ServiceTable`
		`StateVariablePane`
		`StateVariableTable`
		`TableComp`
		`TableModel`
		`TreeComp`
		`TreeNode`

	IconTable MenuBar ServicePane	

Tabla 4: Composición del bundle ControlPoint

En la figura 19 se muestra el diagrama de clases del punto de control. Aparecen en otro color las clases que se han añadido o modificado de la implementación original de Cybergarage.

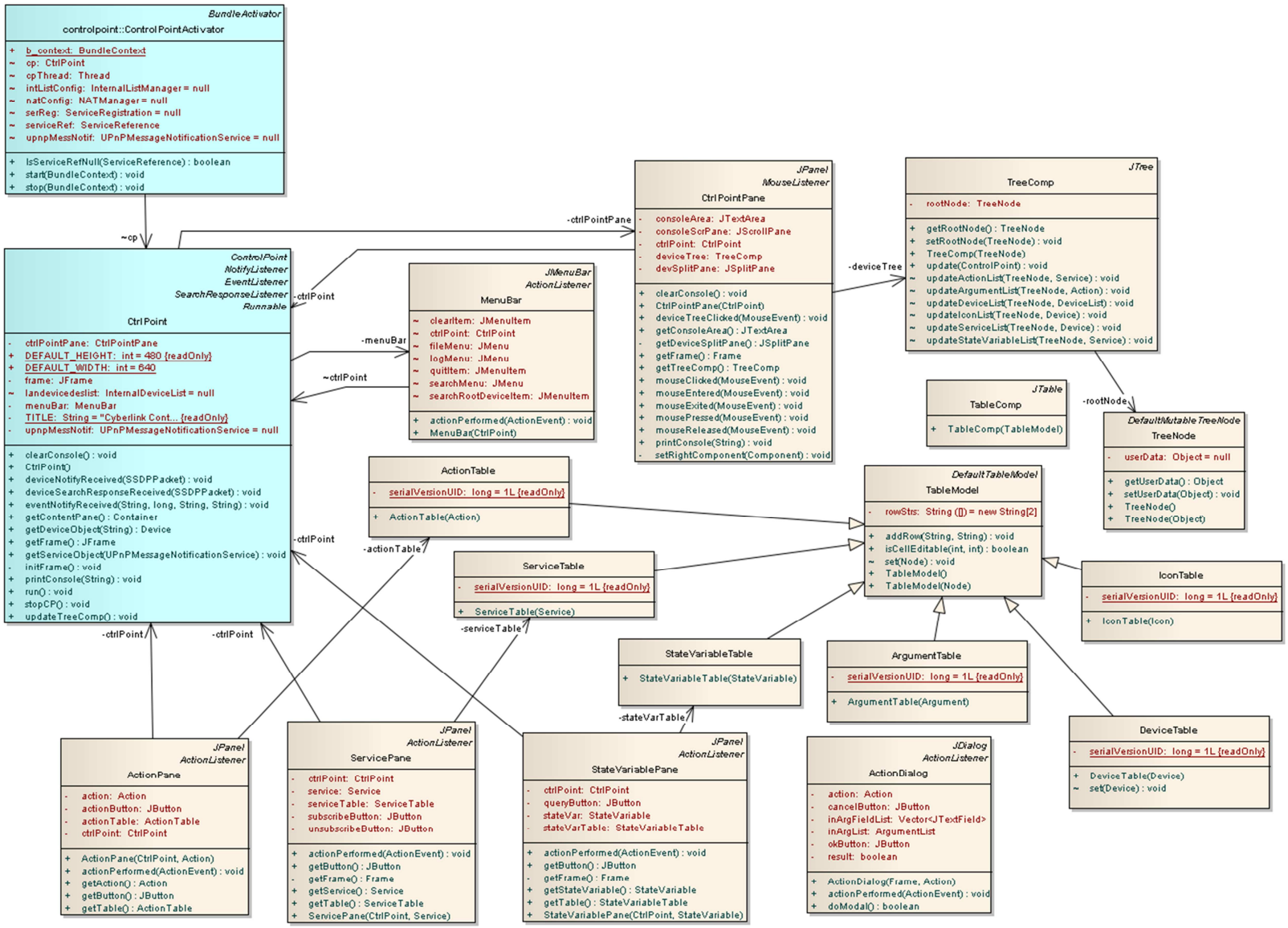

Figura 19: Diagrama de clases del Control Point

A continuación, se describe el contenido de cada paquete en los que se ha subdividido este módulo. Dentro de cada paquete, se muestra de forma más detallada el código de programación de las clases más relevantes.

Paquete org.uc3m.sergiopfc.controlpoint (paquete raíz)

Este es el paquete principal que contiene las clases del bundle que implementa la funcionalidad de punto de control UPnP. Este paquete raíz engloba otro paquete como se ve en la tabla anterior.

Clases

- ***ControlPointActivator***: Es la única clase existente que hay en este nivel del paquete, y es el punto de inicio de la ejecución de este módulo. Esta clase define el ciclo de vida del bundle Control Point y consigue la referencia de los servicios OSGi que va a consumir este módulo. Su función principal consiste en iniciar la ejecución de la funcionalidad del bundle punto de control en la red local y suministrar las referencias a los servicios OSGi a la clase *CtrlPoint*.

Paquete org.uc3m.sergiopfc.controlpoint.upnp

Este paquete contiene todas las clases que implementan el punto de control propuesto por Satoshi Konno. Solo se ha modificado la clase principal *CtrlPoint*, que gestiona la funcionalidad UPnP y la interfaz gráfica. Las demás clases se han dejado intactas, ya que son clases auxiliares usadas por la clase principal.

Clases

- ***CtrlPoint***: Es la clase principal desde la que se ha gestionado toda la comunicación con el Virtual Device, todas las notificaciones de sucesos en la red UPnP (como aparición de nuevos dispositivos, mensajes de búsqueda lanzados,…) se lanzan desde esta clase hacia el Virtual Device. Esto es posible a través de la referencia a un servicio OSGi que tiene asociada esta clase, de esta manera puede lanzar la funcionalidad de asociada a este servicio.

  Es importante destacar que la funcionalidad de este servicio OSGi, es la de notificar al Virtual Device los sucesos en la red UPnP en la fase de descubrimiento. Esto se consigue gracias a los propios escuchadores o *listeners*, que tiene implementado el punto de control para actualizar el estado de los dispositivos que tiene registrados. De esta manera, se ha conseguido reutilizar esta funcionalidad de captura de paquetes UPnP de la red y adaptarla a las necesidades de la plataforma. Este servicio de notificación tiene tres tipos de métodos para notificar sucesos al Virtual Device: notificación de mensajes de búsqueda, notificación de dispositivos que se anuncian a la red y notificación de dispositivos que se desconectan de la red UPnP. La clase que representan este servicio OSGi, junto con sus métodos, se mostrara en la implementación del Virtual Device.

4.3. Implementación del Virtual Device

La fase de implementación del Virtual Device es la más larga y compleja, ya que tiene casi toda la carga funcional del proyecto.

Este módulo al igual que el punto de control, también se ejecuta como un bundle dentro de la pasarela. Registra su propio servicio OSGi para poder, a través de él, establecer un diálogo con el punto de control. El servicio OSGi que registra es este: *UPnPMessageNotificationService*. La función que tiene este servicio, como ya se ha comentado en el punto de control, es la de notificar al Virtual Device los sucesos UPnP que ocurren dentro de la LAN a la que pertenece. Se muestra en detalle más adelante.

Como ya se ha comentado en la introducción a este capítulo, para la implementación de estos servicios OSGi asociados a este bundle se ha utilizado también el bundle repositorio, ya que se hace uso de algunas de las funciones de la librería Cybergarage.

4.3.1. Arquitectura de clases

En este apartado se describen el conjunto de paquetes y clases que componen el Virtual Device integrado en la pasarela. También, por su importancia, se analizan algunos métodos definidos en este módulo.

En la tabla 5 se puede observar cómo está formado el conjunto de paquetes y clases de este módulo. El significado de la tabla es el mismo que el de la tabla 4. Aparecen marcadas las clases más relevantes para entender su funcionamiento dentro del módulo.

Paquete	Clases contenidas
org.uc3m.sergiopfc.virtualdevice	VirtualDeviceActivator **VirtualDeviceManager** ExternalDevice ExternalDeviceList InternalDevice InternalDeviceList
Paquetes contenidos	
org.uc3m.sergiopfc.virtualdevice.service	**DeviceExportService** **DeviceImportService** **UPnPMessageNotification-Service** **NatManager** **InternalListManager**
org.uc3m.sergiopfc.virtualdevice.wan	DescriptionRequestWAN DescriptionResponseWAN SearchRequestWAN **SSLServer** **SSLClient**
org.uc3m.sergiopfc.virtualdevice.lan	**VirtualDescripRequest** VirtualDescripResponse VirtualNotify

Tabla 5: Composición del bundle ControlPoint

En la figura 20 se muestra el diagrama de clases del Virtual Device.

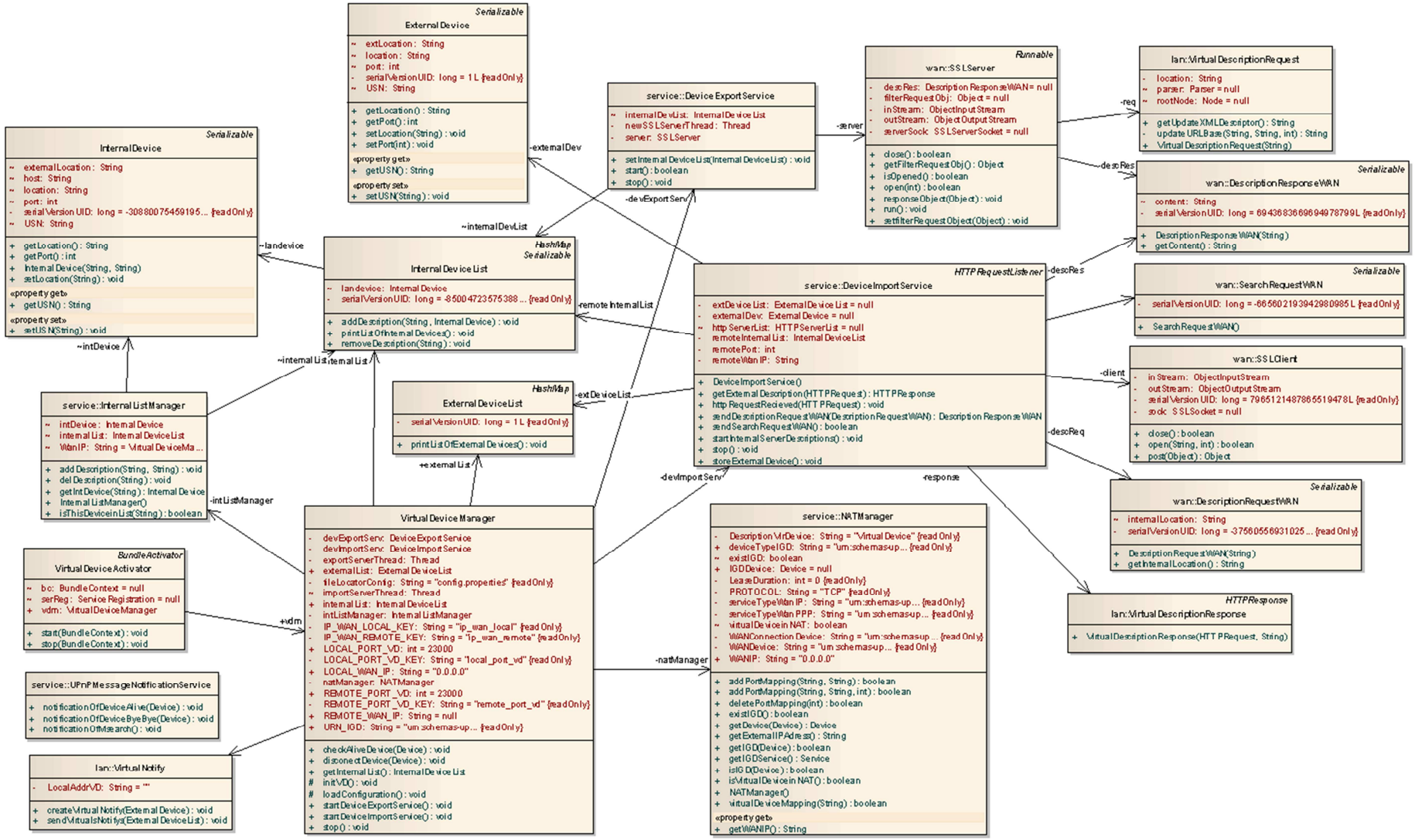

Figura 20: Diagrama de clases del Virtual Device

A continuación se describe el contenido de cada paquete en los que se ha subdividido este módulo. Dentro de cada paquete se muestra de forma más detallada el código de programación de las clases más relevantes.

Paquete org.uc3m.sergiopfc.virtualdevice (paquete raíz)

Este es el paquete principal que contiene la clase activadora del bundle, cuya función es la de gestionar su ciclo de vida, y la clase *VirtualDeviceManager* que gestiona todos los elementos funcionales del Virtual Device. Además contiene los elementos necesarios para crear las listas interna y externa de dispositivos.

Dentro de este paquete raíz, se pueden encontrar los paquetes *services*, *wan* y *lan* como se ve en la tabla anterior. El hecho de encontrar las clases de este módulo separadas en estos paquetes, viene propiciada por la estructuración del código para su mejor entendimiento por parte de los programadores. De esta manera se consigue separar las clases desde el punto de vista estructural y funcional, agrupando las clases que están relacionadas en función de su papel dentro del sistema.

Clases

- ***VirtualDeviceActivator***: Es la clase que gestiona la puesta en marcha del bundle Virtual Device, controlando su ciclo de vida. Es la encargada también de registrar el servicio OSGi `UPnPMessageNotificationService` que va a ser usado por el bundle Control Point. Desde esta clase también se inicia el *VirtualDeviceManager*.

- ***VirtualDeviceManager***: Esta es la clase más importante en el Virtual Device, es la encargada de gestionar toda la funcionalidad del mismo, gestionando por separado diferentes servicios internos dentro de él. Entre sus funciones está la de cargar la configuración inicial del Virtual Device, e iniciar y parar los servicios de importación (`DeviceImportService`) y exportación (`DeviceExportService`) de dispositivos.

  La carga de configuración se hace a través del método `loadConfiguration()`, desde un archivo de texto plano que se encuentra dentro del mismo bundle llamado *config.properties* (el contenido de este archivo se comentará más en detalle en el apartado 5.2). Las variables se cargan en un objeto *Properties* desde la URL del fichero de configuración. Este objeto representa parejas de datos "clave=valor" que se recuperan del archivo. Las variables son las siguientes:

  - *LOCAL_PORT_VD* = Puerto local abierto en el enrutador para comunicarse desde el exterior con el Virtual Device.
  - *REMOTE_PORT_VD* = Puerto remoto abierto en la otra LAN para comunicarse con el Virtual Device de la otra pasarela.
  - *LOCAL_WAN_HOST* = IP de la interfaz WAN que tiene asignada el enrutador de la red local. Se hace uso de esta variable si el IGD no es capaz de conseguir esta IP.
  - *REMOTE_WAN_HOST* = IP de la interfaz WAN de la LAN remota.

Esta clase también se encarga de recibir las notificaciones que le llegan del punto de control a través de su servicio OSGi, y de esta forma gestionar la lista interna de dispositivos (a través de `InternalListManager`) y la NAT del enrutador (a través de `NatManager`). Lo más relevante de esta clase es precisamente el manejo de estas notificaciones por parte del Virtual Device. El proceso es sencillo, cuando el servicio OSGi captura una notificación, éste llama a alguno de los métodos del *VirtualDeviceManager*.

Si la notificación recibida, es la de un dispositivo UPnP que manifiesta su presencia en la red, se hace uso del método `checkAliveDevice(Device dev)` para registrarlo en el Virtual Device y mapearlo en el enrutador. Desde este método también se inicia el proceso de exportación de dispositivos (mediante el método `startDeviceExportService()`), cuando se mapea por primera vez el Virtual Device en la NAT del enrutador. En el cuadro de texto 6, se puede ver el método encargado de llevar a cabo estas funciones.

```
// Chequea la info. del disp. para añadirlo y mapearlo si es necesario
public static void checkAliveDevice(Device dev) {

      if(dev==null){
            System.out.println("No se puede acceder al objeto Device");
            return;
      }
      SSDPPacket ssdpPack = dev.getSSDPPacket();
      String usn = ssdpPack.getUSN();
      String location = ssdpPack.getLocation();
      String serverInfo = ssdpPack.getServer();

      if (serverInfo.contains("VDServer")){
            System.out.println("Es un dispositivo de la otra LAN");
            return;
      }

      //Se comprueba que el disp. no esté en la lista y no sea un IGD
      if (!(natManager.isIGD(dev))){
            if (!(intListManager.isThisDeviceinList(usn))){
                  //Almacenar dispositivo interno en la lista
                  intListManager.addDescription(usn,location);
                  //MAPEO NAT Dispositivo
                  natManager.addPortMapping(ssdpPack.getLocalAddress(),
                        dev.getModelDescription(),
                        HTTP.getPort(location));
            }
      }else{//Es un IGD. Comprobamos si hay registrado alguno en V.D.
            if(!(natManager.existIGD())){
                  natManager.getIGD(dev);
                  //Comprobamos si el V.D. esta mapeado en la tabla NAT
                 if(natManager.virtualDeviceMapping(
                        ssdpPack.getLocalAddress()))
                     //Inicio de servicio exportador de dispositivos
                     startDeviceExportService();
            }
      }
}
```

Cuadro de texto 6: Método checkAliveDevice()

Si la notificación recibida, es la de un dispositivo UPnP que se desconecta de la red, se hace uso del gestor de la lista interna de dispositivos y del gestor de NAT nuevamente para eliminarlo del Virtual Device y eliminar el mapeo de puertos en el enrutador. En el cuadro de texto 7 se puede ver el método encargado de llevar a cabo estas funciones.

```
// Chequea la info. del disp. para eliminarlo del Virtual Device
public static void disconectDevice(Device dev) {

	if(dev==null){
		System.out.println("No se puede acceder al objeto Device");
		return;
	}
	SSDPPacket ssdpPack = dev.getSSDPPacket();
	String usn = ssdpPack.getUSN();
	String location = ssdpPack.getLocation();

	//Se comprueba que el disp. esté en la lista y no sea un IGD
	if (!(natManager.isIGD(dev))){
		if (intListManager.isThisDeviceinList(usn)){
			//Eliminar dispositivo interno en la lista
			intListManager.delDescription(usn);
			//Eliminar Mapeo NAT del enrutador
			natManager.deletePortMapping(HTTP.getPort(location));
			}
	}else
		System.out.println("Se está desconectando un IGD");
}
```

Cuadro de texto 7: Método disconectDevice()

El servicio de importación de dispositivos, es iniciado desde el método `startDeviceImportService()`, cuando se recibe una notificación de mensaje de búsqueda en la LAN. El proceso es el siguiente: se manda una petición de búsqueda de dispositivos a la otra pasarela, posteriormente se almacena la información de los dispositivos que se recibe de la otra pasarela y se desencadena la etapa de descripción de los dispositivos externos.

- ***ExternalDevice***: Esta clase representa, como objeto, la información que se recupera de un dispositivo externo concreto, a través de la interfaz WAN. Almacena la información del USN del dispositivo y la localización de su documento de descripción.

- ***ExternalDeviceList***: La lista de dispositivos externos viene representada por esta clase, contiene un conjunto de objetos *ExternalDevice*.

- ***InternalDevice***: Al igual que *ExternalDevice* contiene la información de un dispositivo externo, la clase *InternalDevice* almacena la información que se recupera de un dispositivo dentro de la LAN. Almacena la información del USN del dispositivo y la localización de su documento de descripción.

- ***InternalDeviceList***: La lista de dispositivos internos viene representada por esta clase, contiene un conjunto de objetos *InternalDevice*. Esta lista se envía a la pasarela remota.

Paquete org.uc3m.sergiopfc.virtualdevice.service

Este paquete contiene todas las implementaciones de los servicios internos que gestiona el *VirtualDeviceManager* para el correcto funcionamiento de la plataforma.

Clases

- ***DeviceExportService***: Esta clase implementa el servicio de exportación de dispositivos del Virtual Device. Su función es la de iniciar el servidor de escucha de mensajes (representado por la clase *SSLServer*) procedentes de la interfaz WAN, en un hilo de ejecución paralelo a la ejecución del programa principal.

- ***DeviceImportService***: Esta clase implementa el servicio de importación de dispositivos del Virtual Device. Su función es la de proveer al *VirtualDeviceManager* de mecanismos para iniciar el proceso de importación de dispositivos, a través de la interfaz WAN. Con esta clase se pueden gestionar todas las peticiones y respuestas que se generan en el proceso de importación. Estos posibles mensajes petición/respuesta, están representados por clases Java que almacenan cierta información, dependiendo del tipo de entidad que representan. En la tabla 6, se pueden ver la relación de estas clases con la función que desempeñan en cada etapa del proceso.

	Petición	**Respuesta**
Etapa de descubrimiento	`SearchRequestWAN`	`InternalDeviceList`
Etapa de descripción	`DescriptionRequestWAN`	`DescriptionResponseWAN`

Tabla 6: Relación de entidades petición/respuesta

Los mensajes petición/respuesta que se envían durante la etapa de descubrimiento, sirven para recuperar información de los dispositivos de la LAN remota. Se gestionan a través de los métodos: `sendSearchRequestWAN()` y `storeExternalDevice()`. El primer método sirve para enviar una petición de búsqueda de dispositivos externos (`SearchRequestWAN`) a través de la interfaz WAN (para ello se usa *SSLClient*). La lista de dispositivos que se recibe como respuesta es almacenada, en la lista de dispositivos externos, a través del método *storeExternalDevice*.

Y los mensajes petición/respuesta que se envían durante la etapa de descripción, sirven para obtener los descriptores, ya traducidos, de los dispositivos externos. Como soporte para gestionar las peticiones de descriptores, por parte de los puntos de control, se hace uso del *listener* de paquetes HTTP `HTTPRequestListener`, que ya está implementado en la librería CyberLinkForJava. De esta forma,

es posible capturar estos mensajes en el Virtual Device y devolver una respuesta virtual al punto de control correspondiente, previa modificación del descriptor. Se usan los métodos `sendDescriptionRequestWAN` y `getExternalDescription`, respectivamente. El proceso para obtener las descripciones externas se desencadena cuando se recibe una petición http en el Virtual Device, es aquí cuando entra en juego el método *getExternalDescription*. Este método contiene a su vez una llamada al método *sendDescriptionRequestWAN* internamente, para hacer las peticiones de descripción a través de la WAN y capturar la respuesta (*DescriptionResponseWAN*) de la otra pasarela que contiene el documento de descripción modificado. Una vez que el Virtual Device ha conseguido el descriptor solicitado, éste lo reenvía hacia su LAN, emulando el comportamiento del dispositivo como si se tratase del dispositivo externo. En el cuadro de texto 8, se muestra el código del método *getExternalDescription*.

```
//Método getExternalDescription
public HTTPResponse getExternalDescription(HTTPRequest httpReq){
      VirtualDescripResponse response = null;
      String HOST = httpReq.getHost();
      //Accedemos a la info del disp. al que se realiza la petición
      for( Iterator<String> it
            = extDeviceList.keySet().iterator(); it.hasNext();) {
            String key = (String)it.next();

            ExternalDevice extDev = extDeviceList.get(key);

            if (HOST.contains(String.valueOf(extDev.getPort()))){

                  DescriptionRequestWAN descReq =
                     new DescriptionRequestWAN(extDev.getLocation());
                  //Enviamos petición de descriptor a la LAN remota
                  DescriptionResponseWAN descRes =
                              sendDescriptionRequestWAN(descReq);
              //Se captura el contenido (XML) de la respuesta obtenida
                  if (descRes != null){
                        String content = descRes.getContent();
                        if (content.equals(""))
                              return null;
                        response =
                          new VirtualDescripResponse(httpReq, content);
                       //Se devuelve el descriptor encapsulado en una
                       //respuesta HTTP
                        return response;
                  }
            }
      }//for
      return response;
}
```

Cuadro de texto 8: Método getExternalDescription

- ***UPnPMessageNotificationService***: Esta clase implementa el servicio OSGi que registra el Virtual Device. Contiene tres métodos correspondientes a los tres tipos de notificaciones que se pueden generar en el punto de control (añadir dispositivo, borrar dispositivo y notificación de búsqueda).

- ***NatManager***: Es la clase que gestiona el IGD registrado para crear o eliminar mapeos de dispositivos (incluso del propio Virtual Device) en el enrutador. También, otra de sus funciones es tener registrado un IGD, para comprobar que la gestión de puertos sea posible en todo momento. Como soporte a la plataforma, esta clase obtiene y mantiene la información de la IP de la interfaz WAN del enrutador igualmente. El proceso para mapear con el IGD, es posible gracias a la librería Cybergarage que provee mecanismos para crear acciones de control. A modo de ejemplo, se puede ver en el cuadro de texto 9, el proceso para crear una petición SOAP que añade un mapeo de puertos con el IGD.

```
public boolean addPortMapping(String intClient, String description, int
port){

      //Se mapean con el mismo puerto en las interfaces LAN y WAN,
     //para evitar problemas con algunas implementaciones NAT
      int NATPort=port;

      if (!existIGD){
            System.out.println("No se puede hacer NAT, no hay IGD");
            return false;
      }

      if(description==DescriptionVirDevice){
            System.out.println("Añadiendo mapeo NAT de V.D.");
            NATPort=VirtualDeviceManager.LOCAL_PORT_VD;
      }else
            System.out.println("Añadiendo mapeo NAT de dispositivo.");

      // Mapeo de puertos
      Action ac = getIGDService().getAction("AddPortMapping");
      ac.getArgument("NewRemoteHost").setValue(WANIP);
      ac.getArgument("NewExternalPort").setValue(NATPort);
      ac.getArgument("NewProtocol").setValue(PROTOCOL);
      ac.getArgument("NewInternalPort").setValue(NATPort);
      ac.getArgument("NewInternalClient").setValue(intClient);
      ac.getArgument("NewEnabled").setValue(1);
      ac.getArgument("NewPortMappingDescription").setValue(description);
      ac.getArgument("NewLeaseDuration").setValue(LeaseDuration);
      if(ac.postControlAction()){
            System.out.println("....Mapeo realizado con éxito");
            return true;
      }else{
            System.out.println("...Error al mapear este dispositivo");
            return false;
      }
```

Cuadro de texto 9: Método addPortMapping

- ***InternalListManager***: Esta clase representa el gestor de la lista de dispositivos internos (*InternalDeviceList*). Es usado por el *VirtualDeviceManager* para gestionar esta lista y ofrece las funciones de añadir, borrar, obtener un elemento o comprobar si está en lista.

Paquete org.uc3m.sergiopfc.virtualdevice.wan

En este paquete se pueden encontrar todas las clases que implementan la funcionalidad necesaria para establecer las comunicaciones a través de la red WAN. En otras palabras, este paquete contiene las clases encargadas de crear los mensajes que se intercambian las dos pasarelas y establecer el mecanismo para poder enviar y recibir estos mensajes a través de la red pública. Este mecanismo está basado en sockets TCP, que envían y reciben flujos de datos que corresponden a objetos java serializados[14].

Clases

- *SearchRequestWAN*: Esta clase define un mensaje de petición de búsqueda de dispositivos externos. La clase no contiene ningún método debido a que solo es usada como objeto serializado, y se detecta como tal al comparar el flujo de bytes recibido en la otra pasarela con la información del objeto Java que representa.

- *DescriptionRequestWAN*: Esta clase define la petición de descripción de un dispositivo externo. Es utilizada para almacenar cierta información (como la localización interna de un dispositivo) y ser transmitida como un objeto serializado hacia la otra pasarela.

- *DescriptionResponseWAN*: Para generar la respuesta a la petición *DescriptionRequestWAN*, se utiliza esta clase también como objeto serializado para transmitirlo por la WAN. Contiene el documento XML con su campo *URLBase* modificado.

- *SSLServer*: Es la implementación de un servidor basado en sockets TCP. Es usada por el servicio de exportación de dispositivos para responder a posibles peticiones de la otra pasarela (tanto en la etapa de descubrimiento como en la de descripción) a través de la red pública. Estos mensajes se manifiestan en forma flujos de bytes, que corresponden a objetos serializados que se envían desde la otra pasarela. Además se usa el soporte que provee Java para enviar y recibir información cifrada sobre la capa de protocolo SSL. El puerto configurado donde escucha las peticiones este servidor, es el que previamente se carga desde el archivo de configuración. Lo más relevante de esta clase es el comportamiento del servidor cuando detecta un flujo de bytes en su canal de entrada. Después de analizar el objeto Java que representa, ejecuta un método llamado `responseObject` y dependiendo del valor de dicho objeto envía una respuesta u otra.

- *SSLClient*: Esta es la clase que interactúa con *SSLServer,* ya que implementa un cliente basado en sockets TCP. Es usada por el servicio de importación de dispositivos para mandar peticiones a la otra pasarela a través de la red pública. Estas peticiones se manifiestan en forma flujos de bytes, que corresponden a objetos serializados que se envían. Los envíos de objetos se realizan a través del método `post(Object)` que devuelve a su vez el objeto de respuesta que recibe de la

[14] La serialización de un objeto consiste en obtener una secuencia de bytes que represente el estado de dicho objeto. Esta secuencia puede utilizarse de varias maneras (puede enviarse a través de la red, guardarse en un fichero para su uso posterior, utilizarse para recomponer el objeto original,…) [82].

otra pasarela por el flujo de entrada del socket. Esta clase sirve tanto para mandar mensajes de descubrimiento, como peticiones de descriptores externos. Este cliente TCP también usa cifrado SSL en sus comunicaciones, debido a que el servidor *SSLServer* lo soporta.

Paquete org.uc3m.sergiopfc.virtualdevice.lan

Dentro de este paquete se encuentran las clases que representan los posibles mensajes que se pueden transmitir a través de la LAN. Estos mensajes tienen que ser mensajes UPnP válidos, ya que en este caso el Virtual Device tiene que interactuar con los puntos de control y dispositivos dentro de la red local. Para ello se ha usado librería Cybergarage, ya que provee de mecanismos para crear este tipo de paquetes UPnP.

Clases

- *VirtualDescripRequest*: Con esta clase se consigue recuperar el descriptor XML de un dispositivo, y actualizar su campo *URLBase* para exportarlo a la pasarela remota, solventando la problemática ya comentada con este campo. En concreto, esta clase se usará en el proceso final de exportación de dispositivos en el momento que se solicite una descripción de un dispositivo externo desde la red WAN. Para recuperar la descripción de un dispositivo concreto y parsear su contenido XML, se hace uso de la clase *Parser*. La información del XML es almacenada en un objeto de la clase *Node.* Ambas clases mencionadas se encuentran en la librería CyberLinkForJava (para más información, véase API de CyberLinkForJava). Mediante el método `getUpdateXMLDescriptor()` se consigue acceder al campo *URLBase* del XML para modificarlo por la IP y puerto públicos (si el XML no tiene este campo, éste se inserta). Una vez modificado, se devuelve el descriptor modificado como una cadena de texto plano, que será encapsulada en la respuesta *DescriptionResponseWAN*. En el cuadro de texto 10 se puede ver el código de este método.

```
//Método encargado de devolver el descriptor actualizado
public String getUpdateXMLDescriptor(){

      int port = HTTP.getPort(location);
      String newUrlBase = "http://"+
                  VirtualDeviceManager.LOCAL_WAN_IP+":"+port;
      Node node = rootNode.getNode("URLBase");

      if (node!=null){
            node.setValue(newUrlBase);
            return rootNode.toString();
      }

      node = new Node("URLBase");
      node.setValue(newUrlBase);
      int index = 1;

      if(!rootNode.hasNodes())
            index = 1;
```

```
        rootNode.insertNode(node, index);

        return rootNode.toString();

}
```

Cuadro de texto 10: Método getUpdateXMLDescriptor

- *VirtualDescripResponse*: Esta clase sirve para responder a las peticiones HTTP de documentos de descripción, que envían los puntos de control al Virtual Device, cuando intentan recuperar la descripción de un dispositivo externo. Para ello se ha usado la clase *HTTPResponse* (véase API CyberLinkForJava), que crea un mensaje de respuesta HTTP al cual se le incluye como contenido, un documento XML de descripción en su interior. Además se añade al paquete de respuesta el campo "*Server:VDServer",* para evitar que el *VirtualDeviceManager* añada el dispositivo externo como interno en la LAN. Esta clase no implementa ninguna funcionalidad añadida a la creación de respuestas a las peticiones HTTP GET con la librería Cybergarage, pero se ha separado en una clase diferente para dejar el código lo más limpio y entendible posible.

- *VirtualNotify*: Esta clase se usa para desencadenar la etapa de descripción de los dispositivos externos, ya que provee soporte para generar mensajes NOTIFY UPnP en la LAN. Se envía un NOTIFY por cada dispositivo externo que haya en la lista. Su funcionamiento se basa en la implementación de la clase *Device* (véase API de CyberLinkForJava), por lo que hace uso de alguna de sus clases como son `SSDPNotifySocket` y `SSDPNotifyRequest`. El hecho de que tenga la nomenclatura de "virtual", quiere decir que el Virtual Device está emulando el comportamiento del dispositivo externo, para forzar a que se inicie un proceso en la comunicación.

4.4. Despliegue de la plataforma en el framework OSGi

A continuación, se muestra el comportamiento de la plataforma una vez desplegados los distintos módulos, anteriormente descritos en el framework OSGi de la pasarela. Para ello se hará uso de diagramas de secuencia. Estos diagramas muestran de forma esquemática la secuencia de invocación de métodos que realiza el sistema para la ejecución de una funcionalidad concreta.

Para explicar mejor el funcionamiento de la plataforma, se distinguen dos diagramas de secuencia. Uno de ellos corresponde al proceso de exportación de dispositivos y el otro al proceso de importación. Estos dos procesos se ejecutan de forma paralela en cada plataforma y son complementarios, es decir, que el proceso de importación de una pasarela requiere del proceso de exportación de la pasarela remota a la que se conecta.

Una vez se inicia la ejecución de los bundles de la plataforma, es el

VirtualDeviceManager el encargado de crear todas las instancias necesarias para que la plataforma funcione (como son las lista interna/externa de dispositivos, el gestor de la lista interna y el gestor de NAT). Además se encarga también de ejecutar tanto el proceso de importación como el de exportación y de gestionar la comunicación con el bundle Control Point.

A continuación, se explican estos dos procesos a través de sus diagramas de secuencia, mostrando los recursos software que están siendo usados en cada momento. Estos recursos software, que aparecen en los diagramas, corresponden a las llamadas a los métodos que intervienen en cada punto de la ejecución, además de algunas anotaciones que ayudan a entender el estado interno.

4.4.1. Despliegue del proceso de exportación de dispositivos

Este proceso se inicia una vez arrancada la aplicación en la pasarela, cuando se detecta un IGD y se mapea el Virtual Device en la tabla NAT del enrutador. En este momento, el Virtual Device está preparado para servir peticiones al proceso de importación de la pasarela remota. Este proceso se limitará por tanto a ser un servidor de peticiones por parte de la otra pasarela.

Si no existe un IGD en el sistema, no se podrá crear la lista de dispositivos internos y el Virtual Device no estará accesible desde la red WAN. Por tanto, el sistema de exportación de dispositivos no estará operativo.

En la figura 21 aparece el diagrama de secuencia de este proceso, que se explica más adelante.

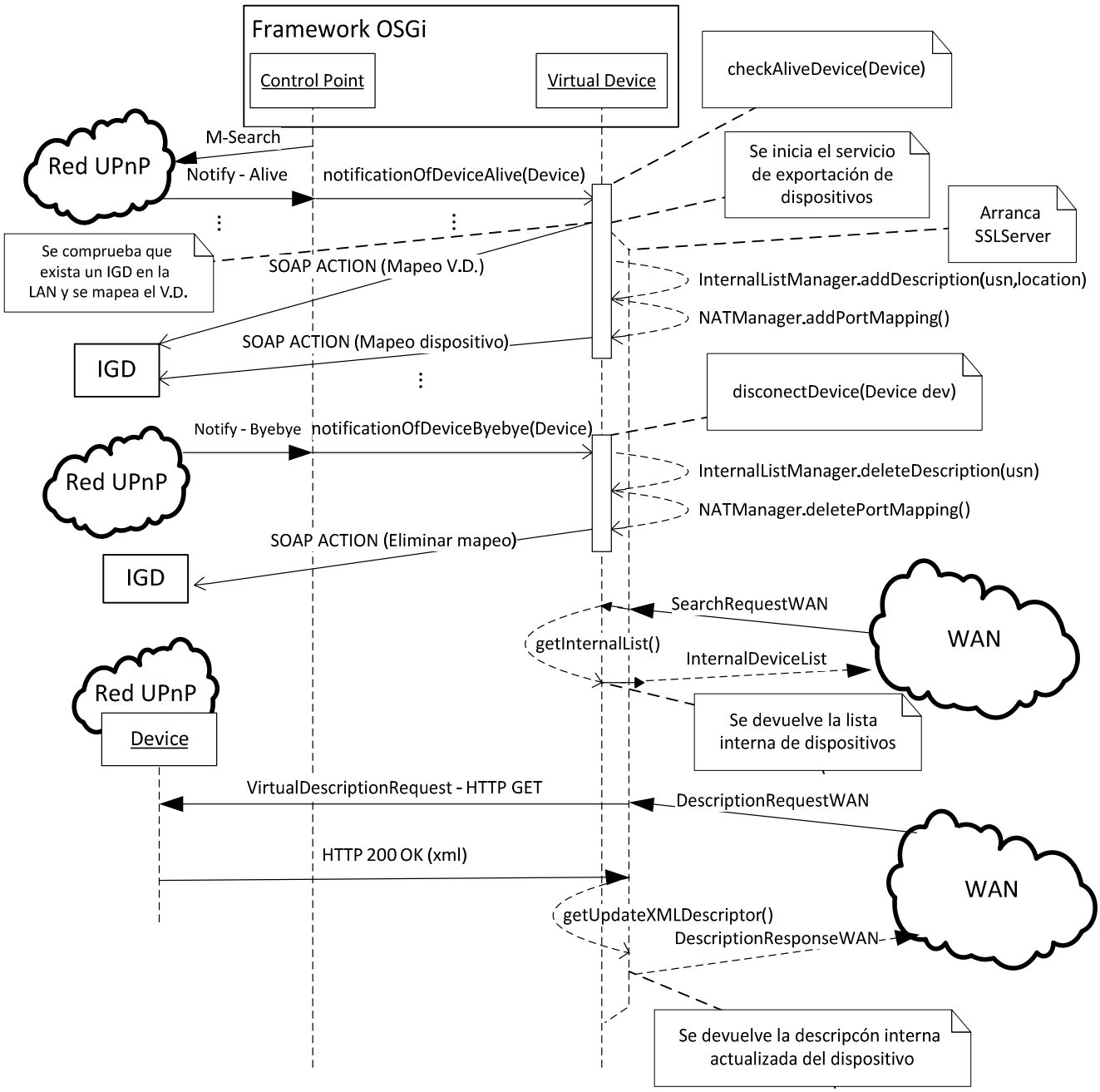

Figura 21: Diagrama de secuencia del proceso de exportación de dispositivos

Una vez puesto en marcha la plataforma, el Virtual Device está preparado para recibir las notificaciones del punto de control, y de esta forma gestionar la lista interna de dispositivos (a través de `InternalListManager`) y la NAT del enrutador (a través de `NatManager`).

Las notificaciones de presencia de dispositivos UPnP (`notificationOfDeviceAlive(Device dev)`) son gestionadas por el *VirtualDeviceManager* desde su método `checkAliveDevice(Device dev)`. Para añadir un nuevo dispositivo a la lista de dispositivos internos, primero se analiza si es un dispositivo de la otra LAN, para evitar añadirlo como interno. Se comprueba con el

método `natManager.isIGD(dev)`, si el dispositivo detectado es un IGD para ver si existe alguno registrado en el Virtual Device, si no es así, se procede a registrar el IGD y mapear el Virtual Device en el enrutador. Es en este punto en concreto cuando comienza a ejecutarse la secuencia de este proceso, con la invocación del método `startDeviceExportService()` por parte del *VirtualDeviceManager*. Esto es debido a que se necesita que esté mapeado en el enrutador el Virtual Device para recibir conexiones desde la WAN. Importante destacar en este punto, que los mismos IGD no se almacenan como dispositivos internos (no se envían a la otra pasarela) por motivos de seguridad. El método encargado de añadir el dispositivo (si no está en la lista) es `addDescription(String usn, String location)` de la clase *InternalListManager*, y el método `addPortMapping(String intClient,String description)` de la clase *NATManager* para mapear puertos con el IGD. El diagrama de secuencia de este método se muestra a continuación en la figura 22.

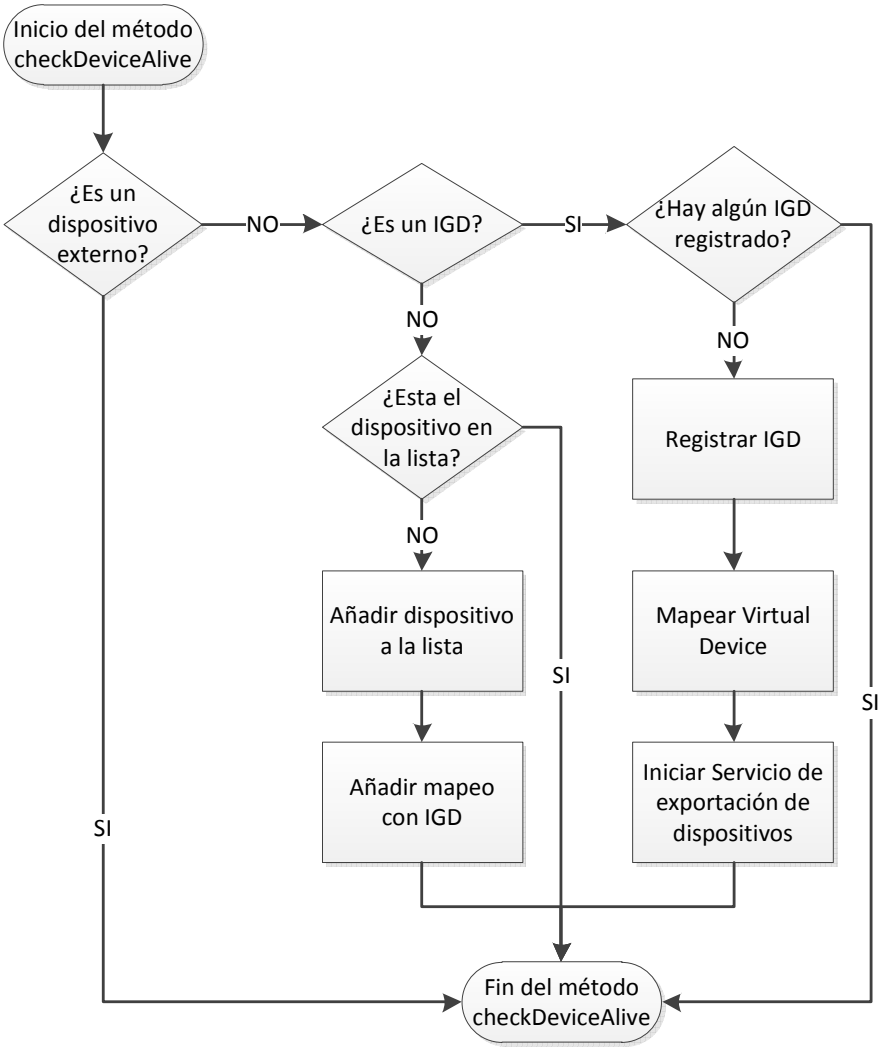

Figura 22: Diagrama de flujo del método checkAliveDevice

En el caso de las notificaciones de desconexión de un dispositivo, se usa el método `disconectDevice(Device dev)` de VirtualDeviceManager. Mediante `delDescription(String usn)` de la clase *InternalListManager* se elimina de la lista

interna, y `deletePortMapping(String intClient,String description)` de la clase *NATManager* para eliminar mapeo de puertos con el IGD.

Una vez que se ha hecho la llamada al método `startDeviceExportService()`, se crea un objeto de la clase *DeviceExportService* que se inicia con el método start(). Esto provoca la creación de un hilo de ejecución para la clase *SSLServer*, que será la encargada de gestionar las peticiones que reciba de la clase *SSLClient*. Este hilo se mantendrá abierto, a partir de este momento, durante toda la ejecución del programa. Hay que remarcar este hecho, ya que la plataforma tiene que estar preparada en todo momento para recibir las peticiones de importación de dispositivos de la otra pasarela. A partir de este momento, este proceso se limitará a responder a estas peticiones. Se pueden distinguir dos tipos de peticiones:

- Petición de búsqueda de dispositivos: En este caso se recibe un objeto *SearchRequestWAN* y se devuelve un objeto *InternalDeviceList* para que la pasarela remota tenga una copia de la lista interna.

- Petición de descriptor de un dispositivo concreto: Lo que está intentando la otra pasarela es acceder a la localización de un dispositivo interno. Aquí entra en juego la técnica *"man in the middle"* y el servicio de exportación de dispositivos recupera el descriptor XML correspondiente, mediante la clase *VirtualDescriptionRequest*, actualizando su campo *URLBase* con el método `getUpdateXMLDescriptor()` para devolverlo a la otra pasarela.

4.4.2. Despliegue del proceso de exportación de dispositivos

Este proceso, como se muestra más adelante en su diagrama de secuencia, se inicia automáticamente cuando se lanza un mensaje de búsqueda de dispositivos en la LAN, de esta manera se unifican los procesos de búsqueda interna y externa de dispositivos de la red interna UPnP.

Hay que tener en cuenta también para este proceso, que es necesaria la existencia de un IGD en su red UPnP local para el funcionamiento correcto del sistema. Esto es debido, igual que en el proceso de exportar dispositivos, a que el Virtual Device necesita estar mapeado en el enrutador para capturar las respuestas a las peticiones que haga a la otra pasarela. Por tanto, en este proceso el Virtual Device se comporta como un mero cliente de peticiones.

De manera gráfica en la figura 23 se puede ver el diagrama de secuencia completo de este proceso, detallando posteriormente algunos de los métodos que entran en juego.

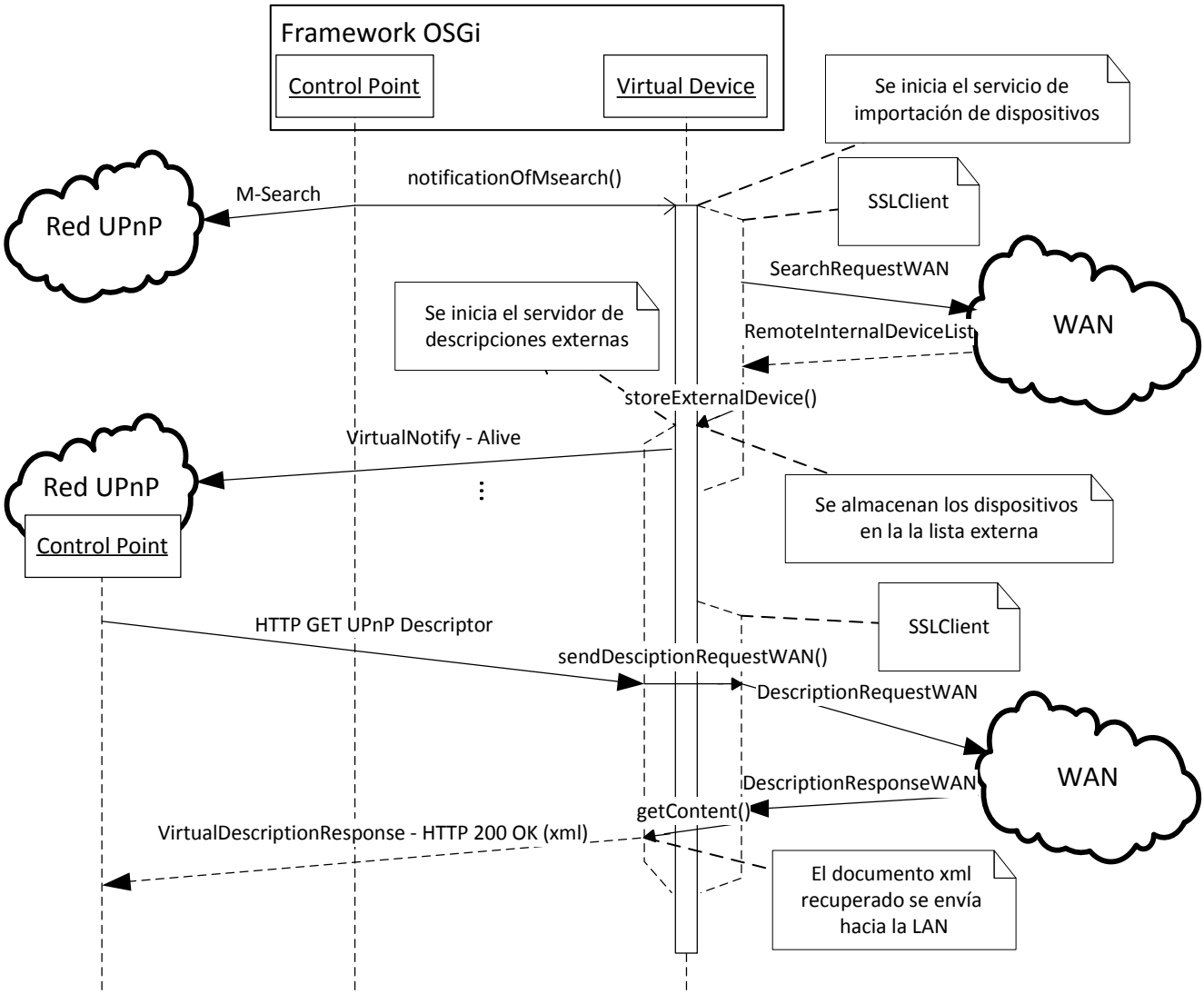

Figura 23: Diagrama de secuencia del proceso de importación de dispositivos

Hay que tener en cuenta en este proceso que, para todas las peticiones que se manden a la WAN se hará uso de la clase *SSLClient*.

Se iniciará automáticamente el servicio de importación de dispositivos (`DeviceImportService`), con una notificación de mensaje de búsqueda en la LAN. El proceso es el siguiente: se manda una petición de búsqueda de dispositivos (*SearchReqestWAN*) a la otra pasarela, posteriormente se almacena la información que se recibe de la otra pasarela en forma de lista interna remota. Esta lista interna se transforma en una lista externa de dispositivos mediante el método `storeExternalDevice()`.

A partir de este momento, se desencadena la etapa de descripción de los dispositivos externos que gestionará la clase *DeviceImportService*. Se usa de nuevo en este punto la técnica "man in the middle", para obtener el descriptor actualizado de la otra pasarela, siendo transparente para la red UPnP interna. El proceso es el siguiente:

1. Se extrae la información de la lista externa y se crean servidores http para escuchar las peticiones de documentos de descripción por parte de los puntos de control, esto se hace con el método `startInternalServerDescriptions()`. Para detectar estas peticiones en el Virtual Device, se hace uso de la implementación del listener de paquetes http (*HTTPRequestListener*). La misma clase *DeviceImportService* actuará de escuchadora gestionando los eventos con el método `httpRequestRecieved(HTTPRequest httpReq)`.

2. Se lanzan a la LAN un mensaje virtual NOTIFY por cada dispositivo externo en la lista, para desencadenar el proceso de descripción en los puntos de control. El término virtuales, se recuerda al lector, que es debido a que el Virtual Device está actuando como si fuera un conjunto de dispositivos anunciándose en la red. Estos mensajes se crean con la clase *VirtualNotify*, de tal manera que los puntos de control hagan sus peticiones HTTP a los servidores abiertos en el Virtual Device.

3. Se escuchan las posibles peticiones que realicen los puntos de control, para devolverles el descriptor solicitado. Puede haber varios puntos de control en la LAN, incluido el que se ejecuta en la pasarela, aunque solo este escenificado el proceso de petición de uno de ellos. Una vez recibido una petición se ejecuta el método `getExternalDescription(httpReq)` para enviar una petición *DescriptionRequestWAN* mediante `sendDescriptionRequestWAN(descReq)` que obtendrá una respuesta *DescriptionResponseWAN* de la otra pasarela con el contenido XML actualizado del descriptor (este contenido se puede recuperar con su método `getContent()`). Para propagar este descriptor hacia la LAN, se usa un objeto *VirtualDescriptionResponse* que genera un *HTTPResponse* con el contenido que obtiene de *DescriptionResponseWAN* y se envía como respuesta al punto de control que lo ha solicitado.

Capítulo 5

5. Simulación de la plataforma

En este capítulo se abordan los procedimientos que se han realizado para poner en marcha la plataforma, con el objeto de valorar la respuesta del sistema en un escenario real de ejecución y solventar posibles errores que puedan surgir por agentes externos (como pueden ser dispositivos UPnP) que interactúan con la plataforma, que de otra forma pasarían desapercibidos. De esta forma se puede comprobar su viabilidad a la hora de desplegarlo en un entorno cotidiano de la vida real.

Como primer paso se analiza el escenario de pruebas usado en el laboratorio, basado en el IDE Eclipse, y se analizan los requisitos hardware y software que deberían cumplirse para el correcto funcionamiento de la plataforma.

Posteriormente se hará un breve guía de configuración de la plataforma, para poder desplegarla correctamente en cualquier escenario.

Por último, se presentan las simulaciones llevadas a cabo tanto con el framework OSGi en el que se ha desarrollado la aplicación, como en otras distribuciones software de frameworks OSGi.

5.1. Escenario de pruebas

Uno de los primeros pasos en la elaboración de un proyecto es definir el entorno de desarrollo en el que se va a trabajar. A continuación, se enumeran las herramientas que han sido utilizadas durante todo el proceso de pruebas de la plataforma.

Se han utilizado equipos informáticos para simular las pasarelas residenciales comerciales, ya que no se disponía de este tipo de equipos especializados. Cada uno de estos equipos y sus características se muestran en la tabla 7.

Equipos informáticos	Sistema operativo	Procesador	Memoria RAM
Equipo Portátil (Simulador pasarela 1)	Windows XP Pro	Pentium M 1.7 GHz	1GB
Equipo Sobremesa (Simulador pasarela 2)	Ubuntu 10.04	Pentium 4 2,66 GHZ	512MB

Tabla 7: Equipos informáticos usados en la plataforma

El hecho de estar la plataforma albergada dentro de un framework OSGi, ha facilitado mucho su despliegue en estos equipos, ya que simplemente se necesita como requisito tener instalado la máquina virtual Java en las máquinas para que hacer funcionar el framework OSGi sobre ellas. En concreto, se ha usado las versiones de JDK 1.6 de Java, tanto para el equipo Windows como para el equipo Ubuntu. Como peculiaridad también se pone de manifiesto el carácter multiplataforma que tiene esta aplicación, debido a los distintos sistemas operativos que están presentes. Este hecho viene motivado por el uso de Java donde esta característica es intrínseca a su tecnología.

En cuanto al despliegue de equipos de red para montar las dos redes locales, se ha contado con dos enrutadores IP, que aparecen en la tabla 8, donde se muestran sus características. Y se ha contado con la infraestructura de red del laboratorio que provee conexión a Internet a través de interfaces WAN Ethernet, manteniendo direcciones IPv4 públicas en esta interfaz.

Equipos de red	Procesador	Memoria RAM	Memoria Flash
Enrutador IP Linksys WRT54G v.1.1. (LAN 1)	BCM4710 125 MHz	16MB	4MB
Enrutador IP Linksys WRT350N v.1.1. (LAN 2)	BCM4785 300 MHz	32MB	8MB

Tabla 8: Equipos de red usados en la plataforma

Para el escenario de pruebas software se ha utilizado como entorno de desarrollo integrado el *Eclipse Helios*, como ya se ha comentado anteriormente. Este IDE provee un plugin llamado Eclipse Equinox (en su versión 3.6.1), que provee una implementación del *Realease* 4 de OSGi. A través de este IDE, se ha facilitado al programador todas las herramientas para el desarrollo y prueba de bundles en Java (llamados plugins en Equinox), facilitando así la rápida depuración de errores.

El escenario UPnP se ha simulado dentro de los mismos equipos informáticos, es decir, que los mismos equipos han actuado como pasarelas OSGi y como dispositivos UPnP. Para crear el escenario UPnP, se han utilizado las siguientes implementaciones que cumplen con el estándar UPnP:

- Control Point UPnP: versión 1.5 de CyberLink para lenguaje Java. Se trata de una aplicación de código abierto desarrollada en el marco del proyecto CyberLinkForJava (véase Anexo A.4).
- IGD: Se ha utilizado la implementación interna que tienen los enrutadores IP del IGD de UPnP. La versión que implementan es la 1.0.
- Dispositivos UPnP de prueba: Se ha usado los dispositivos de ejemplo desarrollados con la librería Java CyberLink propuestos por el equipo de Cybergarage. También se han usado algunas implementaciones de dispositivos desarrolladas en código abierto, como los que se proponen

desde el proyecto *Developer Tools for UPnP Tech* [76].

A continuación en la figura 24, se muestra de forma gráfica como está desplegado el escenario de pruebas, según lo comentado en las anteriores líneas.

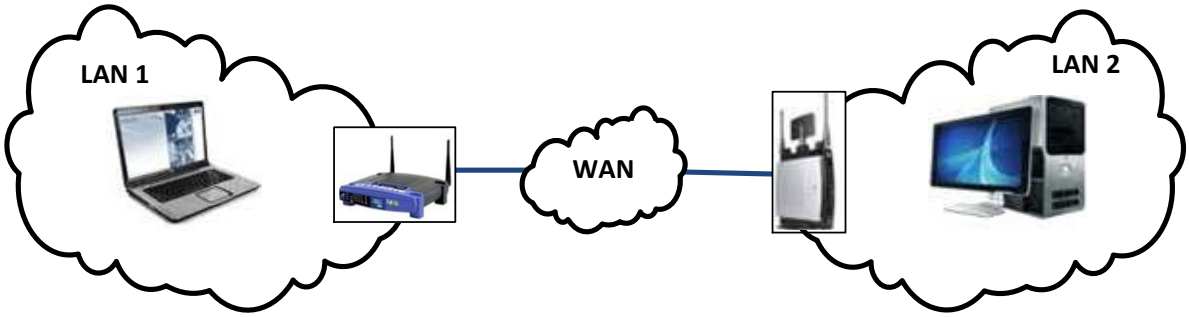

Figura 24: Despliegue del escenario de pruebas

Otras herramientas usadas para la el desarrollo de la plataforma son estas:

- *WireShark 1.4.2*: como analizador de tráfico IP de protocolos. Se ha utilizado para el análisis y monitorización de las comunicaciones de red entre los dispositivos UPnP.

- Herramientas de edición de diagramas UML: ArgoUML (en su versión 0.32.1) y Microsoft Visio 2010, para generar los diagramas de clases y los diagramas de secuencia.

5.1.1. Requisitos hardware y software

En cuanto a los requisitos de funcionamiento de la plataforma, no han sido requeridos como objetivos dentro del proyecto, pero se establecen algunas recomendaciones en este punto. No se han tenido en cuenta factores como retardos de transmisión en la comunicación, ya que el propósito de las pruebas no incluye la valoración ni del tipo ni del estado de la red.

Por supuesto, sí que habría que contar con el requisito de que UPnP debe estar habilitado en los sistemas de la LAN y configurados los firewall para que permita su uso, se comentará sobre esto en el apartado 5.2.

En todo caso, para conocer que equipos cumplirán los requisitos para soportar la plataforma, se deberá tener en cuenta factores como el tipo de dispositivos que van a ser desplegados en la red UPnP, o el tipo de red WAN y LAN sobre la que se despliega.

En cuanto a los requisitos mínimos hardware se desconoce las características que debe tener el equipo que despliegue la pasarela residencial, debido al desconocimiento de equipos hardware comerciales que soporten frameworks OSGi. En todo caso, el equipo no tendría que ser muy potente. Simplemente debería poder ejecutar el

framework OSGi sobre una máquina virtual Java sin problemas de recursos. Precisamente el hecho de usar un tipo pasarela u otro determinará los requisitos mínimos. Por ejemplo, un sistema informático con un mínimo de 256MB de RAM y procesador de 800MHz, debería bastar, incluso se podrían usar menores prestaciones si es un sistema dedicado.

En cuanto a los requisitos mínimos software, normalmente el sistema operativo del equipo depende mucho del fabricante, pero hay otros factores importantes a tener en cuenta como la implementación del framework OSGi.

5.2. Configuración de la plataforma

Antes de desplegar la plataforma, es necesario realizar una configuración previa de los equipos de red que intervienen en ella. Los enrutadores IP que intervienen en la plataforma deben permitir que exista comunicación entre los dos Virtual Device de cada pasarela.

Debido a que normalmente la pasarela residencial se sitúa dentro de la LAN, los enrutadores IP tienen que redireccionar tráfico procedente de su interfaz WAN hacia la dirección privada donde se encuentra el Virtual Device de la plataforma. Este hecho presenta un problema, ya que por defecto la mayoría de los enrutadores IP tienen activados mecanismos de seguridad (*firewalls*) para evitar intrusiones no deseadas desde su interfaz WAN a los equipos de su LAN. Solo se mantienen abiertos ciertos puertos (y se permite cierto tipo de tráfico) por defecto en su interfaz WAN. Por lo que sería necesaria una configuración de seguridad de estos equipos de red.

En el caso del escenario desplegado para realizar las pruebas, ha sido necesario modificar la configuración del cortafuego, mediante una opción llamada *DMZ (DeMilitarized Zone)* [77] que posibilita el acceso desde la interfaz WAN mediante NAT a cualquier puerto abierto en una dirección privada de la LAN. Este hecho en sí provoca vulnerabilidad en el host que tenga asignada esa IP interna, a menos que se use un cortafuego apropiado en él.

Otro factor a tener en cuenta para configurar sería el cortafuego interno que tenga la máquina donde se ejecuta la pasarela en la LAN, al igual que el enrutador, sería necesario configurarlo para permitir recibir conexiones desde el exterior, limitadas estas a los puertos requeridos en la plataforma.

Fichero de configuración

En cuanto a la configuración interna de la plataforma de soporte, se hace uso del archivo de configuración del Virtual Device, mencionado en el capítulo de implementación, para configurar parámetros como direcciones IP y puertos. A continuación, se muestra, en el cuadro de texto 11, el aspecto de este archivo de configuración llamado *config.properties*. En el contenido del archivo se muestra una pequeña descripción y el nombre correspondiente de las variables clave (*key*) dentro del código.

```
################################################################
# Archivo de configuración específica para el Virtual Device
################################################################

#Descripción – String key
#key=value

#Direccion IP WAN de la subred remota - REMOTE_WAN_HOST
ip_wan_remote=163.117.141.50

#Direccion IP WAN de la subred local - LOCAL_WAN_HOST
ip_wan_local=163.117.141.37

#Puerto abierto en VD remoto - REMOTE_PORT_VD
remote_port_vd=23000

#Puerto abierto en VD local - LOCAL_PORT_VD
local_port_vd=23000
```

Cuadro de texto 11: Archivo de configuración

Este archivo de configuración debe situarse dentro del contenedor del mismo bundle del Virtual Device, para que el *classloader* pueda cargar su localización durante la ejecución. En concreto, la ruta escogida para almacenar este archivo es *./properties/config.properties* (en algunas implementaciones de frameworks solo es accesible desde el directorio raíz).

La configuración de los interfaces IP WAN de las redes deben conocerse a priori para incluirlas en este archivo de configuración. Aunque como mínimo bastaría con incluir la IP de la interfaz WAN remota, porque la dirección de la interfaz WAN local se obtiene mediante el IGD (se debe incluir en aquellas implementaciones de IGD que devuelven “0.0.0.0” como valor de IP externa). Hay que destacar que los valores de esta direcciones IP deben ser direcciones IPv4 públicas y corresponder con las direcciones WAN de las máquinas de cada LAN donde está desplegada la plataforma.

Los puertos usados para abrir las comunicaciones en los Virtual Device pueden tener cualquier valor dentro del rango permitido que no esté reservado en el sistema. Obviamente se debe conocer la configuración en las dos pasarelas para crear los archivos de configuración.

5.3. Simulación

La parte de simulación es muy importante para el desarrollo del proyecto y forma parte activa de él, durante todo el proceso de implementación se ha llevado a cabo la simulación de cada módulo por separado, y posteriormente de toda la plataforma integrada. Hay que tener en cuenta que las simulaciones se han realizado en una infraestructura de red real, con conexión a Internet como red pública. Los objetivos que se persiguen con estas pruebas son:

- Determinar la viabilidad de la plataforma dentro de un framework OSGi.
- Comprobar los posibles inconvenientes que conlleva desplegar la plataforma a través de una infraestructura de red IPv4 real.
- Simular mediante dispositivos UPnP, y comprobar que realmente se han cumplido los objetivos de soporte del proyecto.

Se ha simulado en un principio a través del framework Equinox integrado en Eclipse, para el desarrollo de pruebas de software funcionales en cada módulo y posteriormente pruebas de integración dentro del framework.

Para probar las ventajas que presenta desarrollar aplicaciones sobre un framework OSGi, se ha optado por probarlo en otras implementaciones de framework que existen actualmente y comprobar que realmente existe compatibilidad completa.

5.3.1. Simulación del framework OSGi con Eclipse Equinox

Hay que tener en cuenta que el framework Equinox usado está integrado dentro del IDE Eclipse, por lo que se evitan muchas configuraciones de librerías y del propio framework, debido a la herramienta de desarrollo de bundles (*plugins*) que tiene. En este caso no es necesaria la empaquetación de bundles dentro de archivos jar.

En el espacio de trabajo tenemos tres *plugins* que corresponden a los tres módulos comentados en el capítulo anterior, bundle respositorio, controlpoint y virtualdevice. A la hora de desplegarlos en el framework, se debe establecer un orden de carga mediante prioridades o niveles. El primer bundle que se tiene que cargar, es el bundle repositorio, debido a que contiene las librerías usadas por los otros bundles. Posteriormente el bundle Virtual Device debe ejecutarse antes que el bundle controlpoint en el framework OSGi, ya que éste es consumidor de servicios del Virtual Device. Para ello se establecen en la configuración del framework, el nivel de arranque (*start Level*) de cada bundle, para que se carguen en este orden. Suponiendo que se quiera automatizar el arranque de la plataforma al arrancar el framework.

Tras comprobar el funcionamiento correcto dentro del entorno de desarrollo se ha querido usar el framework OSGi Equinox *standalone* (framework Equinox ejecutado de manera independiente fuera del IDE Eclipse) [78]. El mayor inconveniente que surge es el hecho de que el framework precisa algunas librerías que hay que cargar manualmente en el contexto de ejecución del framework para que se ejecuten sin problemas los bundles. Esto no pasa en el entorno de desarrollo, ya que internamente Eclipse resuelve muchas de estas librerías cuando se ejecuta dentro del propio entorno.

A continuación se muestra el procedimiento de despliegue del framework OSGi Equinox 3.6.2, fuera del entorno de desarrollo:

- Se comprueba primera las librerías Java que se cargan al iniciar el framework de esta manera, analizando las posibles dependencias con los bundles de la plataforma que no se resuelven. Es necesario indicar estas dependencias, con las librerías Java insertadas manualmente en los archivos MANIFEST de los bundles que lo requieran, para que accedan correctamente a estos repositorios. Las dependencias de librerías de J2SE que ha sido necesarias cargar manualmente en este caso, han sido: *javax.swing, javax.swing.table*, *javax.swing.tree*, cuyo requerimiento lo solicita el bundle Control Point; *javax.net*, *javax.net.ssl*, como paquetes necesarios para el Virtual Device.
- Se obtienen los bundles empaquetados en archivos .jar, para cargarlos en el framework. Los bundles obtenidos tienen esta denominación en sus archivos:
  - "org.cybergarage.stack.jar"
  - "org.uc3m.sergiopfc.virtualdevice.jar"
  - "org.uc3m.sergiopfc.controlpoint.jar"
- Se pueden crear entradas en el archivo de configuración del framework para que se carguen automáticamente estos bundles, situados previamente en el mismo directorio que el bundle del sistema, *org.eclipse.osgi_3.6.2.R36x_v20110210.jar*. En este archivo, llamado *config.ini*, se puede indicar el orden de carga de los bundles, su localización está en el directorio "*./configuration/"*. A continuación se muestra el contenido del archivo de configuración, en el siguiente cuadro de texto 12.

```
osgi.bundles=org.cybergarage.stack@2:start,
org.uc3m.sergiopfc.virtualdevice.jar@3:start,
org.uc3m.sergiopfc.controlpoint.jar@4:start
eclipse.ignoreApp=true
```

Cuadro de texto 12: Contenido del archivo config.ini

- Una vez arrancado el framework, se comprueba siguiendo estos pasos que el despliegue en el framework es el correcto:
  1. La carga de configuración correcta desde el bundle Virtual Device, mediante un mensaje por consola de los valores cargados.
  2. Registro del servicio OSGi de notificación por parte del Virtual Device.
  3. Carga del bundle Control Point y subscripción al servicio de notificación del Virtual Device. Esto provocará el inicio del proceso de descubrimiento de dispositivos UPnP internos y la visualización de la interfaz gráfica del punto de control.
  4. En este punto debería iniciarse el servicio de exportación de dispositivos una vez se haya mapeado el Virtual Device con un IGD.
  5. Se comprueba el funcionamiento de la plataforma, mandando un mensaje de búsqueda desde el punto de control, para desencadenar el

proceso de importar dispositivos externos, comprobando en la interfaz gráfica del punto de control que realmente aparecen los dispositivos de la otra pasarela.

A continuación se muestra en la figura 25, una captura de pantalla de la consola del framework OSGi Equinox, junto con el Control Point, donde aparece el proceso de arranque y configuración. Además aparecen todos los mensajes de depuración según el proceso explicado para el proceso de exportación de dispositivos.

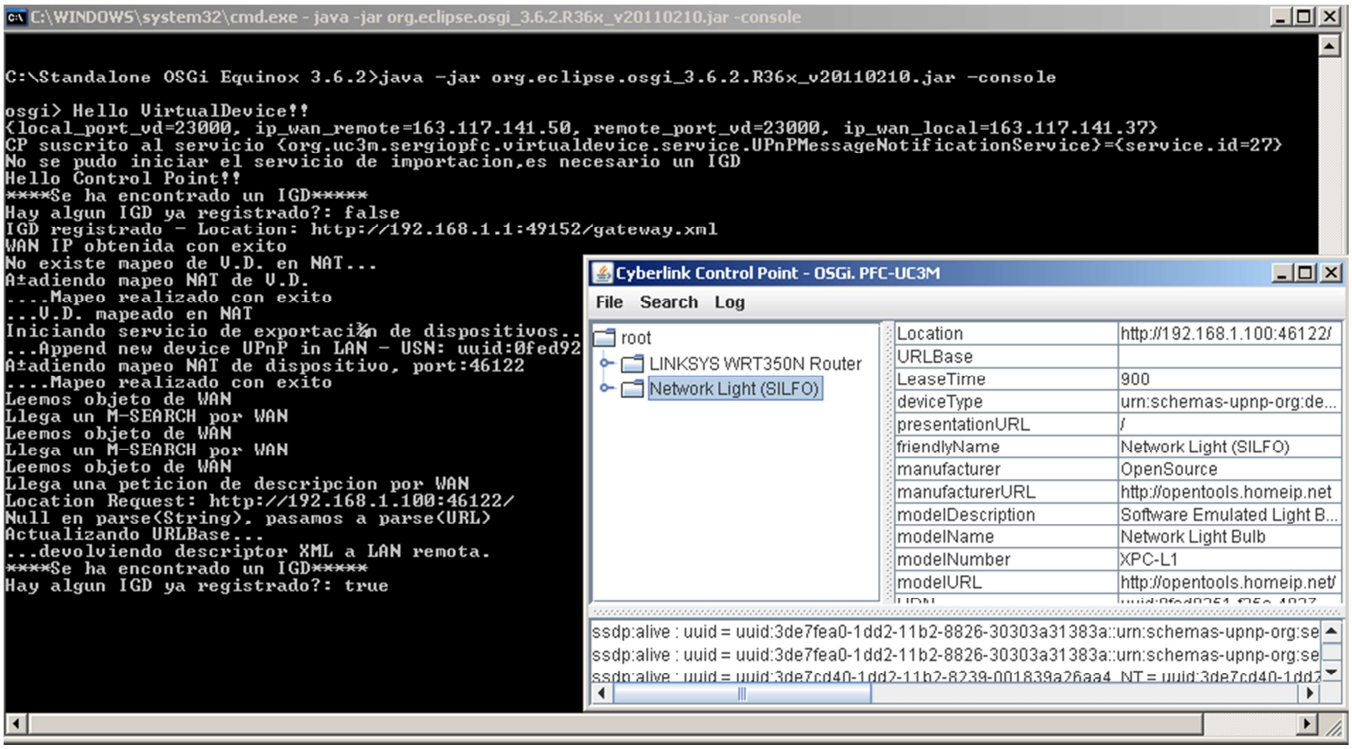

Figura 25: Consola framework Equinox 1

En la figura 26, como complementación de la anterior, se muestra otra captura de pantalla de la consola junto con el Control Point, donde aparece el proceso de importación de dispositivos.

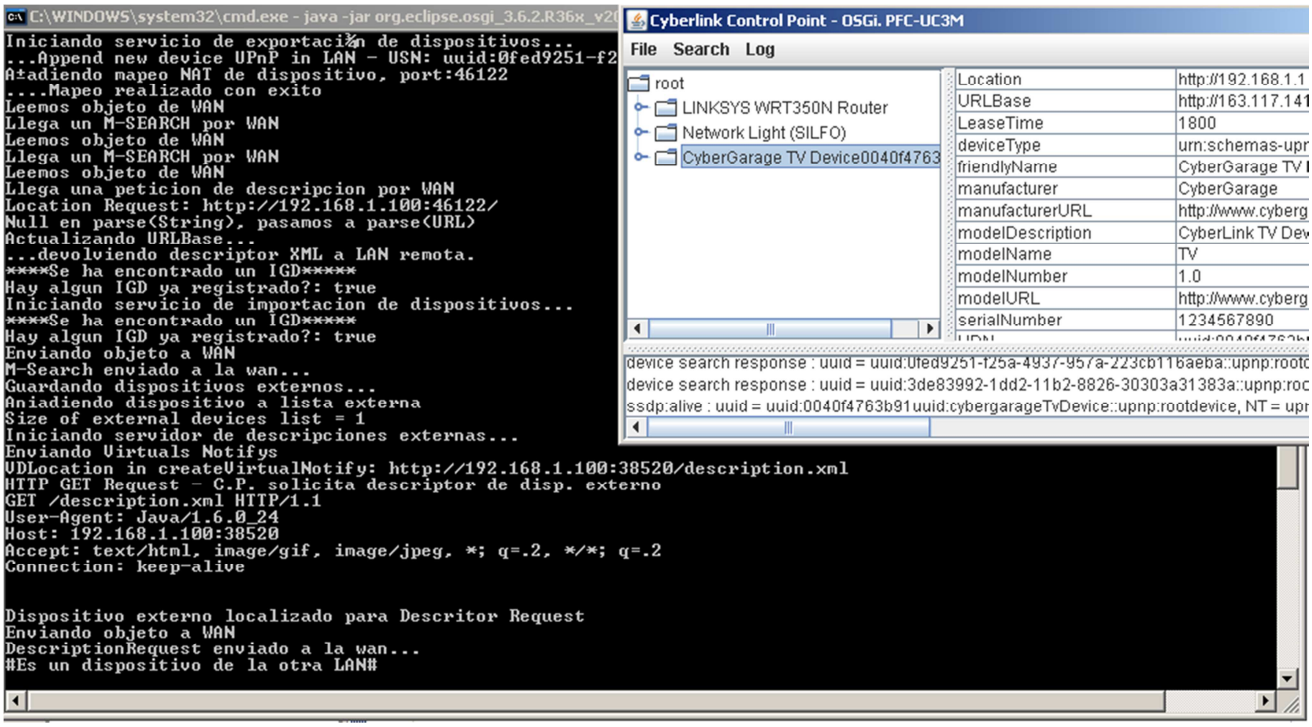

Figura 26: Consola framework Equinox 2

5.3.2. Simulación con otros frameworks OSGi

Para comprobar el carácter multiplataforma que tiene OSGi, se ha querido simular la plataforma con otra implementación OSGi como es Knopflerfish, en su última versión 3.1.0.

El procedimiento para desplegar el sistema en este framework OSGi, es el mismo que el proceso empleado para el framework OSGi Equinox. En este caso, ha sido necesario también indicar en los archivos MANIFEST de los bundles, los paquetes a importar para que accedan correctamente a estos repositorios. Son los mismos paquetes Java a importar, que en el caso del framework OSGi. Los bundles utilizados tienen la misma denominación, en sus archivos .jar también.

Al igual que en el caso de Equinox, se puede modificar el archivo de configuración del framework Knopflerfish, para que la plataforma se carge al arrancar el framework. Para ello es necesario modificar el archivo *init.xargs*, añadiendo las siguientes entradas mostradas en el cuadro de texto 13.

```
-initlevel 7
#PFC Cybergarage Stack
-install pfc/org.cybergarage.stack.jar
##Other jars
#
-startlevel 7
# Start of these bundles are delayed since this makes start
```

```
# order dependencies much easier
# Other jars
#
#PFC Cybergarage Stack
-start pfc/org.cybergarage.stack.jar
#PFC Bundles Platform
-initlevel 8
-install pfc/org.uc3m.sergiopfc.virtualdevice.jar
-startlevel 8
-start pfc/org.uc3m.sergiopfc.virtualdevice.jar
-initlevel 9
-install pfc/org.uc3m.sergiopfc.controlpoint.jar
-startlevel 9
-start pfc/org.uc3m.sergiopfc.controlpoint.jar
```

Cuadro de texto 13: Archivo de configuración del framework Knopflerfish

A continuación se muestra, en la figura 27, una imagen de la consola del framework Knopflerfish, que en este caso sí que provee una interfaz gráfica, con los mensajes de depuración del proceso de arranque y exportación de dispositivos.

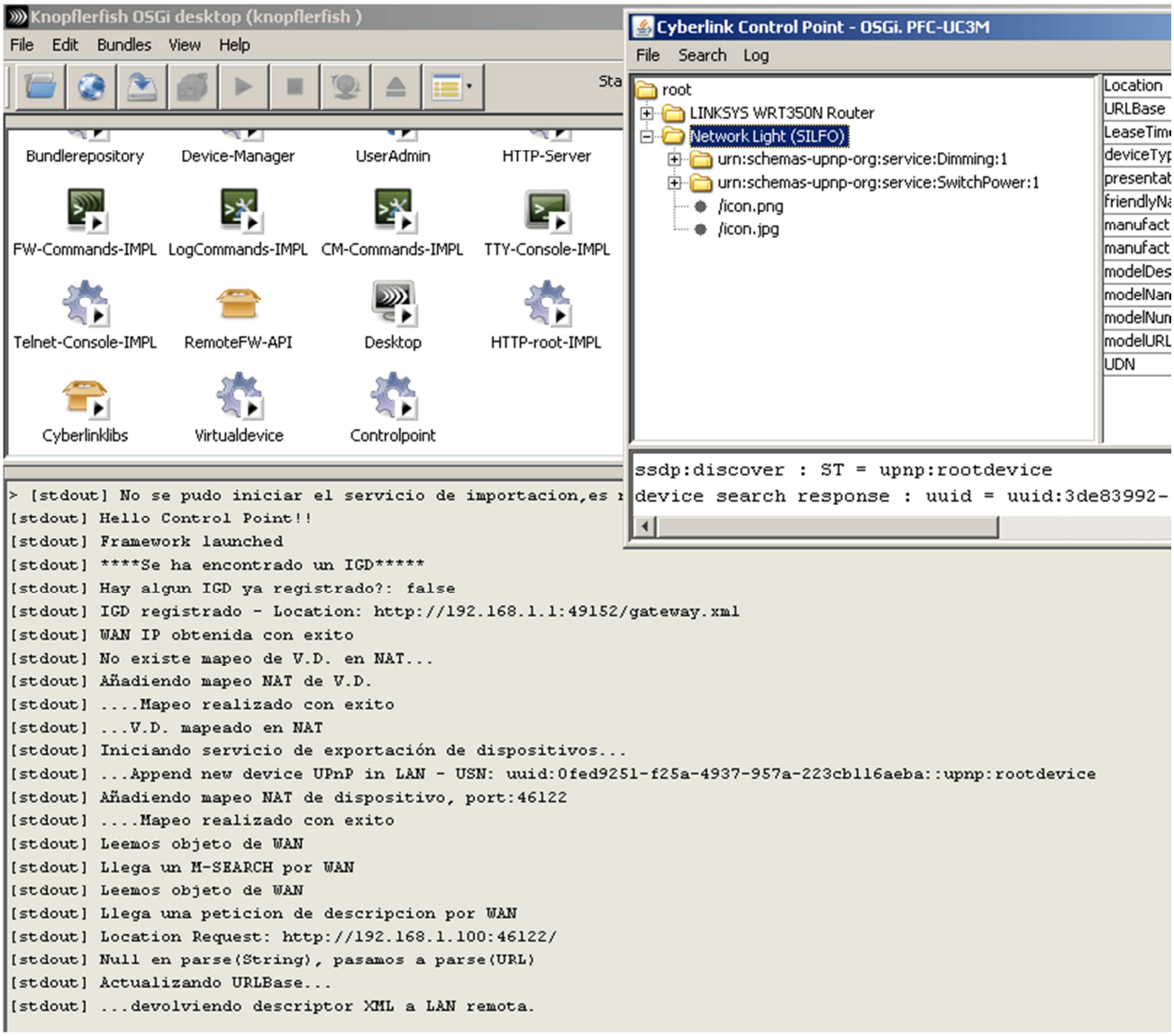

Figura 27: Consola framework Knopflerfish 1

Y en la figura 28, se muestra el proceso de importación de dispositivos.

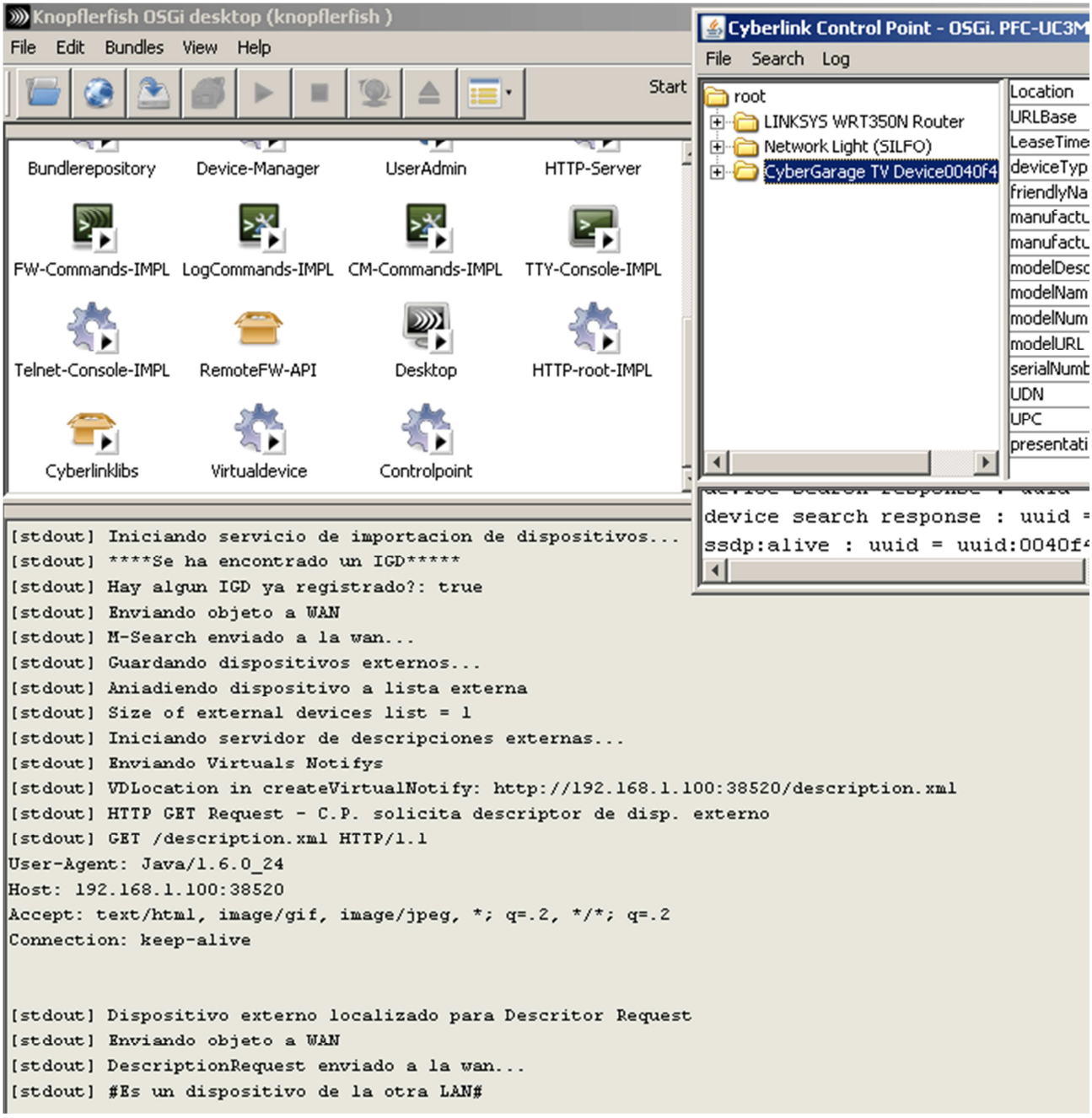

Figura 28: Consola framework Knopflerfish 2

Capítulo 6

6. Observaciones

En este capítulo se recogen las observaciones más relevantes obtenidas durante la realización de este proyecto.

En primer lugar, se detallan las principales conclusiones tras realizar este proyecto, basadas en el grado de cumplimiento de los objetivos planteados. Tras esto, se exponen las limitaciones que surgieron durante el desarrollo de la plataforma, así como, las limitaciones que presenta el modelo final desplegado que escapan a los requisitos del proyecto. Por último, se proponen algunas recomendaciones para una futura mejora de la plataforma, basadas sobre todo en las limitaciones que se mencionan.

6.1. Conclusiones

Tras la realización de este proyecto se ha llegado a las siguientes conclusiones:

- El uso tecnologías como UPnP, va a suponer todo una revolución en el futuro de los entornos digitales aplicados a viviendas (y a edificios en general). El hecho de instalar/configurar sistemas y dispositivos de forma transparente al usuario, y las ventajas que todo ello conlleva, proporcionarán una nueva forma de interactuar con los distintos dispositivos electrónicos. Se ha demostrado además, que UPnP es una tecnología fiable y muy versátil, capaz de realizar todas las tareas deseables para el funcionamiento del Hogar Digital.
- Estas tecnologías son hoy en día desconocidas para la mayoría de las personas, por lo que el mercado que éstas abarcan es muy pequeño. El objetivo que se debería marcar, es dar a conocer todas las ventajas que ofrece, facilitando el uso de estándares en el mercado (en entornos domóticos sobre todo) para su rápido despliegue en la sociedad. Un primer paso sería crear mecanismos de compatibilidad entre las tecnologías existentes. El hecho de crear facilidad de uso al usuario, es uno de los puntos fuertes que por los que apostar. Un punto importante a destacar, por el que estas tecnologías presentan muchas ventajas y pueden ser precisadas en las construcciones del futuro, son la gestión energética y los sistemas de seguridad, aparte de servicios de valor añadido como la teleasistencia y la telemedicina.
- Gracias a plataformas de soporte como la que se propone en este proyecto, se

puede extender este tipo de tecnologías, gracias a Internet, sobre cualquier parte. Aunque según el estudio realizado, se puede decir que no existe una única solución para implementar la plataforma, se puede afirmar que se han cumplido los objetivos fijados.

- Hay que destacar en este punto la tecnología OSGi para las pasarelas residenciales, ya que ha facilitado muchísimo el despliegue de la plataforma, a pesar de que no se ha utilizado todo su potencial. Se podría haber simplificado un poco más la implementación de la plataforma utilizando alguno de los servicios que se proponen en las ultimas *releases* de OSGi (como son los servicios http y upnp).
- En todo momento se diseñó la plataforma pensando en la especificación UPnP, esto permite que exista compatibilidad con todo dispositivo que soporte esta certificación. Se ha comprobado que algunas implementaciones de dispositivos UPnP no contienen en sus descriptores XML el campo *URLBase*. Por ello, se ha diseñado la plataforma de tal manera que sea insertado este campo en ese caso.
- El hecho de usar conexiones TCP, para enviar la información entre pasarelas, nos ofrece fiabilidad en la recepción de la información, ya que se da prioridad a que la petición llegue (generalmente no es un factor crítico el que la petición se retrase).
- Se decidió también que los mismos IGD que intervienen en la plataforma, no se enviaran a la otra pasarela como los demás dispositivos UPnP por motivos de seguridad. Aunque si el entorno de despliegue lo requiere, se puede modificar esta restricción.
- Se ha intentado implementar mecanismos de seguridad (como SSL) en las conexiones, a través de la WAN, en las que se tiene control e intervienen en los procesos de la plataforma. Hay que destacar que la implementación de seguridad no es un objetivo que se contemple en el proyecto, por ello no se ha perseguido dar soporte completo de seguridad en toda comunicación WAN.
- Gracias a herramientas como *Wireshark* y *Developer Tools for UPnP*, el proceso de implementación y testeo ha sido mucho más rápido.

6.2. Limitaciones

No hay que olvidar que todo desarrollo, presenta siempre algunas limitaciones que se escapan a los objetivos de un proyecto. Algunas limitaciones o vulnerabilidades más importantes de la plataforma se comentan en las siguientes líneas:

- La limitación más importante viene de la mano de la seguridad en la comunicación de la plataforma a través de la red WAN. Si bien se ha intentado implementar seguridad en las comunicaciones que se controlan en la plataforma, quedaría por resolver la vulnerabilidad que supone transmitir información sin cifrar en los mensajes UPnP que no se controlan por parte

del Virtual Device a través de la WAN. Como son los mensajes de control y eventos UPnP.

- Otra limitación importante es el hecho de que este proyecto intenta dar soporte a UPnP sobre la versión 4 del protocolo IP. Esta versión está en declive, como consecuencia del agotamiento de direcciones IPv4 y el posterior cambio a la versión 6 del protocolo IP. En principio, con la nueva versión del protocolo IP se resuelven los problemas que intenta solventar la plataforma, debido a las limitaciones de la versión anterior del protocolo IP ya comentadas. Este hecho supondría el desuso de esta plataforma de soporte.
- El hecho de usar librerías externas dentro de los bundles de la plataforma, supone un proceso previo de verificar si los frameworks OSGi sobre los que se va a desplegar la plataforma cargan por defecto las librerías requeridas. Todo ello dependerá de la JVM que este desplegada en la pasarela residencial correspondiente y de la versión del framework. En el caso de que algunas de las librerías externas no se carguen por defecto, es necesario cargarlo manualmente en el bundle repositorio e indicarlo en su MANIFEST.
- El hecho de usar SSL como mecanismo de cifrado de la comunicación, puede dar problemas de compatibilidad entre plataformas que no soporten, por defecto, los mismos certificados de seguridad.
- Existe la posibilidad de que haya un conflicto de puertos cuando se ejecutan las mismas implementaciones de dispositivos UPnP en cada LAN y en la misma máquina donde se encuentra la pasarela.
- Se han realizado las pruebas de soporte sobre dispositivos no limitados (que no tienen problemas con su rendimiento), esto podría presentar un inconveniente en caso de querer ejecutar la plataforma sobre dispositivos más limitados, como dispositivos móviles, al no poseer JVM sino CVM o KVM las cuales no dan soporte a aplicaciones J2SE.
- La carencia de una interfaz gráfica de configuración quizás haga la plataforma menos atractiva, en lo relativo a la configuración rápida de la misma, ya que actualmente la modificación de la configuración requiere de un proceso de reinstalación del bundle. Cualquier característica que indebidamente limita la capacidad de un dispositivo para diferenciarse no tendrá gran aceptación popular.

6.3. Trabajos futuros

Basándonos en las limitaciones anteriores y en otros aspectos funcionales, se proponen algunas recomendaciones para una mejora de la plataforma. Se presentan en forma de posibles líneas futuras de trabajo, partiendo del desarrollo de esta plataforma. Ejemplo de estas futuras líneas de trabajo serían:

- Una de las principales líneas de trabajo correspondería a dar soporte completo de seguridad a la plataforma, usando en la medida de lo posible los mecanismos de seguridad que propone el *UPnP Forum - DeviceSecurity*

Service. De esta manera conseguiríamos una plataforma segura y orientada a autenticación del usuario.

- El hecho, comentado anteriormente, de que la plataforma este diseñada para dar soporte en redes IPv4, daría pie a una línea de desarrollo de adaptación a las nuevas redes IPv6. Habría que mantener la filosofía actual de la plataforma, aunque se simplificaría mucho su implementación.
- Se podría extender también el carácter punto a punto que tiene la plataforma, en la que solo configuramos una plataforma remota a la acceder, por un carácter multipunto, configurando una lista de redes LAN remotas de las importar y exportar dispositivos externos desde la misma pasarela. Gracias a la tecnología OSGi sería muy fácil creando, por ejemplo, varios bundles Virtual Device por cada pasarela remota.
- Implementar un sistema de configuración rápida del sistema, quizás mediante un interfaz de usuario, evitando de esta forma la limitación mencionada. Aquí también entraría en juego la fase de presentación UPnP como un interesante punto de innovación. La clara separación entre la programación de servicios en UPnP y la interfaz de usuario en la descripción de dispositivos hace que sea sencillo y cómodo el administrar este tipo de arquitecturas de red para los usuarios. Además la interfaz de usuario de un producto es muy importante para las empresas que diseñan dispositivos electrónicos de consumo.
- Llevar la plataforma a sistemas y dispositivos más limitados como *PDA's*, *SmartPhones*, *PocketPC's*, etc. Esto es posible gracias a la especificación R4 de OSGi en la que se definen perfiles de ejecución del framework para dispositivos limitados como CDC en J2ME.
- Se podría crear compatibilidad con los dispositivos DLNA, que al parecer está obteniendo más soporte comercial que UPnP. Debido a que esta tecnología está basada en UPnP, sería sencillo crear soporte para extenderla de igual manera hacia redes externas.
- Otra tecnología a la que podría darse soporte, junto con los dispositivos UPnP sería Jini, gracias a que existe un módulo en OSGi que podría usarse para consigue una interoperación entre las dos tecnologías.

Capítulo 7

7. Planificación y presupuesto

Este capítulo se centra en describir cual ha sido el proceso de planificación del proyecto, un factor muy importante a la hora de desarrollar con éxito un proyecto. De igual manera, acorde con esta planificación se ha establecido un presupuesto en base a la duración y recursos de esta planificación. Este presupuesto muestra una memoria resumida de los costes y gastos generados de la realización del mismo.

7.1. Planificación

Se han utilizado, para este capítulo, herramientas de gestión de proyectos para estructurar la planificación en tareas simples con una duración estimada de cada una, mostrando también la desviación con respecto a la planificación inicial.

Primero se muestra una descripción general de las fases principales por las que pasa proyecto. Estas fases se desglosan más adelante en tareas, que se mostrarán en un diagrama de Gantt.

Fases del Proyecto

La realización del proyecto se puede separar en varias fases, aunque algunas de ellas se han realizado en paralelo. En las próximas líneas se enumeran las distintas fases con una breve descripción del propósito de cada una:

- **Fase 1**. Documentación sobre las posibles tecnologías y herramientas implicadas en el proyecto. Marco Teórico.

  En esta fase se realiza un estudio previo sobre los objetivos del proyecto y se analizan las posibles tecnologías que entran en juego, para adquirir, si es necesario, los conocimientos técnicos para el desarrollo de la aplicación.

  Duración: 160 horas.

- **Fase 2**. Diseño del modelo propuesto.

  En esta fase se realiza el diseño del modelo posteriormente a implementar. Se basa en hacer un estudio, en base a la documentación inicial recogida, de los posibles modelos a llevar a cabo, escogiendo el más conveniente para su

implementación.

Duración: 56 horas.

- **Fase 3**. Implementación del modelo

  En esta fase se realiza la implementación propiamente dicha de la plataforma de soporte, en base al modelo realizado en la fase previa. Las tareas que se definen en esta fase suponen la implementación de cada una de las unidades funcionales de la aplicación.

  Duración: 176 horas.

- **Fase 4**. Desarrollo del entorno de pruebas.

  Una vez obtenido los prototipos implementados de los módulos de la plataforma, se realiza un montaje de un entorno de pruebas. En este entorno se desarrollan toda una fase de pruebas, a la que se somete la plataforma, para comprobar el correcto funcionamiento de la aplicación en los posibles escenarios.

  Duración: 160 horas.

- **Fase 5**. Elaboración de la memoria del proyecto.

  La elaboración de la documentación recogida a lo largo del proyecto, no es una fase en sí, ya que es una acción que se desarrolla en paralelo a las demás. Pero se ha optado por incluirla como una fase, ya que consume tiempo y recursos.

  Duración: 352 horas.

Planificación mediante Diagrama de Gantt

A continuación, se muestra mediante un diagrama de Gantt en la figura 29, cada una de las tareas que se han realizado agrupadas en las distintas fases que definen la consecución de este proyecto. Esta planificación corresponde a las fases que inicialmente se crearon para gestionar el proyecto. Además se muestra en el mismo diagrama, en rojo, las tareas que han sido quitadas o añadidas, y que han supuesto un desvió en la planificación inicial del proyecto.

Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin	Predeces
1	**DOCUMENTACIÓN SOBRE EL PROYECTO**	**30 días**	**lun 06/09/10**	**vie 15/10/10**	
2	Analisis del problema que se plantea en el proyecto	2 días	lun 06/09/10	mar 07/09/10	
3	Documentación sobre UPnP	8 días	mar 07/09/10	jue 16/09/10	2
4	Sintesis de ideas y funcionamiento UPnP-IGD aplicado al proyecto	3 días	jue 16/09/10	lun 20/09/10	3
5	Documentación sobre el Framework de OSGi - Equinox	7 días	lun 20/09/10	mar 28/09/10	4
6	Integración de UPnP en Framework OSGi	5 días	mié 29/09/10	mar 05/10/10	5
7	Documentación sobre direccionamiento IPv4 y mecanismo NAT	5 días	lun 11/10/10	vie 15/10/10	6
8	**DISEÑO DEL MODELO PROPUESTO**	**10 días**	**lun 18/10/10**	**vie 29/10/10**	**1**
9	Diseño de la pasarela de soporte	2 días	lun 18/10/10	mar 19/10/10	7
10	**Diseño de los componentes del framework de OSGi**	**9 días**	**mar 19/10/10**	**vie 29/10/10**	**9**
11	Diseño del Control Point	3 días	mar 19/10/10	jue 21/10/10	9
12	Diseño del IGD (Tarea eliminada, implement. IDG de router)	0 días	jue 21/10/10	jue 21/10/10	11
13	Diseño del Virtual Device	5 días	lun 25/10/10	vie 29/10/10	12
14	**IMPLEMENTACIÓN DEL PROYECTO**	**33 días**	**lun 01/11/10**	**mié 15/12/10**	**8**
15	Preparación del IDE Eclipse Equinox	5 días	lun 01/11/10	vie 05/11/10	13
16	Implementación de los Devices UPnP de test (Tarea eliminada)	0 días	vie 05/11/10	vie 05/11/10	15
17	**Despliegue del framework de OSGi**	**24 días**	**jue 11/11/10**	**mar 14/12/10**	**16**
18	Implementación del bundle - Control Point	7 días	jue 11/11/10	vie 19/11/10	16
19	Implementación del bundle - IGD (Tarea eliminada)	0 días	vie 19/11/10	vie 19/11/10	18
20	Implementación del bundle - Virtual Device	10 días	lun 22/11/10	vie 03/12/10	19
21	Extensión funcional a etapa de descripción UPnP	7 días	vie 03/12/10	lun 13/12/10	20
22	**ENTORNO DE EJECUCION**	**40 días**	**jue 16/12/10**	**mié 09/02/11**	**14**
23	Preparación del escenario de pruebas	7 días	jue 16/12/10	vie 24/12/10	21
24	Integración de software en el entorno de simulacion	15 días	lun 27/12/10	vie 14/01/11	23
25	Simulación en otros frameworks OSGi (Tarea nueva)	10 días	vie 14/01/11	jue 27/01/11	24
26	Pruebas y conclusión	10 días	jue 27/01/11	mié 09/02/11	25
27	**MEMORIA DEL PROYECTO**	**78 días**	**mié 10/11/10**	**vie 25/02/11**	
28	Redactar: Introducción	2 días	mié 10/11/10	jue 11/11/10	1
29	Redactar: Estado del arte	23 días	jue 11/11/10	lun 13/12/10	7
30	Redactar: Analisis/Diseño del escenario experimental	18 días	lun 13/12/10	mié 05/01/11	8
31	**Redactar: Implementacion/Despliegue**	**13 días**	**jue 06/01/11**	**lun 24/01/11**	**14**
35	Redactar: Entorno de ejecución del proyecto/Demo	14 días	vie 28/01/11	mié 16/02/11	22
36	Redactar: Pruebas y conclusiones / Resultados	6 días	mié 16/02/11	mié 23/02/11	26
37	Redactar: Presupuesto y planificación. Anexos	3 días	mié 23/02/11	vie 25/02/11	36

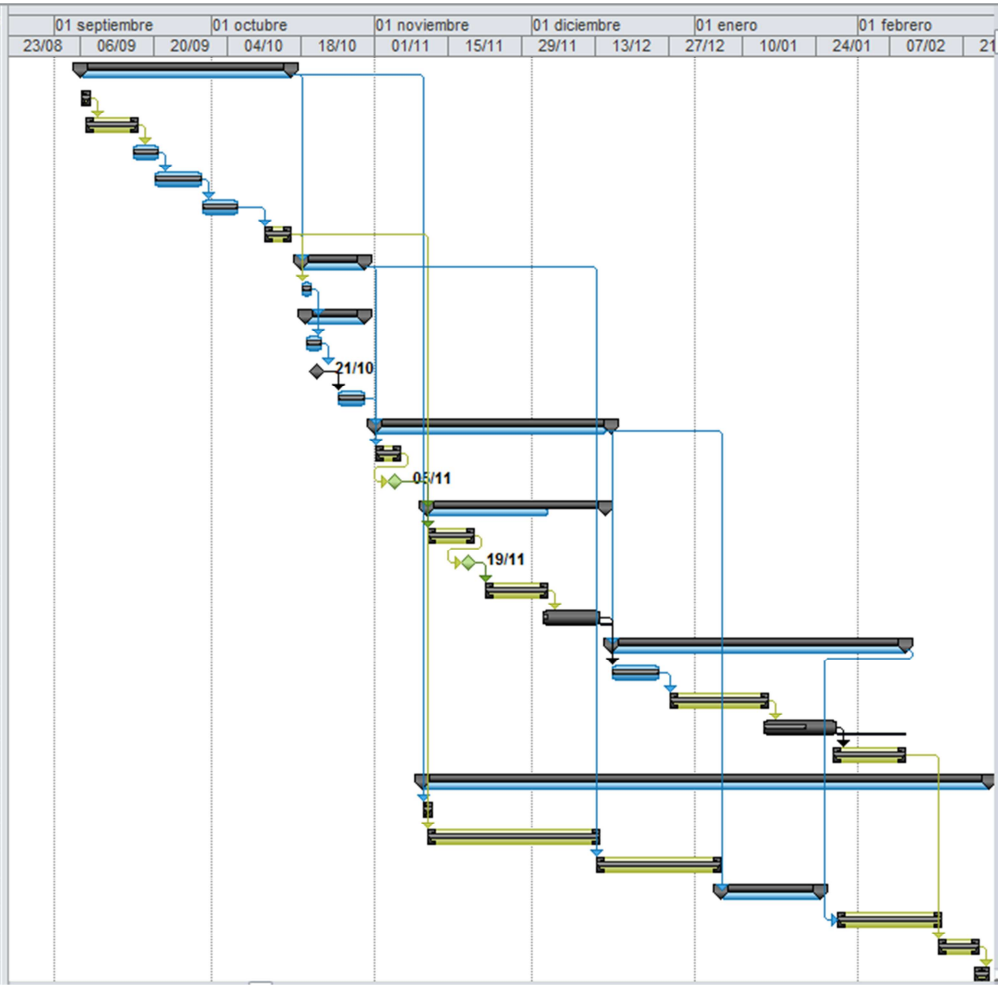

Figura 29: Diagrama de Gantt

7.2. Presupuesto.

Para la elaboración del presupuesto se han considerado aquellos costes que están relacionados con la utilización de infraestructuras y equipos, y los que provienen de la mano de obra para la realización del proyecto.

Costes materiales

Son los derivados de la creación del entorno de trabajo donde se ha desarrollado y probado la aplicación, incluyendo los costes indirectos estimados por el consumo energético, material fungible y comunicaciones. Se han considerado únicamente los equipos que han sido necesarios en este proceso aunque se hayan utilizado un número mayor para las pruebas. Estos costes se muestran en la tabla 9.

Concepto	Precio (€)	Periodo de amortización (meses)	Periodo de uso (meses)	Importe (€)
Ordenador portátil (Windows XP)	600	36	6	166,67
Ordenador sobremesa (Ubuntu 10.01)	500	36	6	83,34
Router Linksys WRT54G	52	36	6	8,67
Router Linksys 350N	153	36	6	25,5
Infraestructura red (2 IP´s públias)	280	36	6	46,67
Costes indirectos				230
TOTAL				**560,85**

Tabla 9: Costes materiales

Costes de mano de obra

En este proyecto ha intervenido un único Ingeniero Técnico de Telecomunicaciones para realizar el conjunto de tareas definidas en las distintas fases del proyecto. El coste derivado de la mano de obra, se resume en la tabla 9.

Responsable	Horas	€/hora	Importe (€)
Ingeniero Técnico de Telecomunicaciones	904	15	**13560**

Tabla 10: Costes mano de obra

Presupuesto total

En la tabla 11, se resume el presupuesto total para la realización del proyecto.

Concepto	Importe (€)
Costes materiales	560,85
Costes mano de obra	13560
Total base imponible	14120,85
IVA (18%)	2541,75
Total	**16662,6**

Tabla 11: Presupuesto total

Instalación y configuración

Adicionalmente el cliente puede requerir los servicios de instalación y puesta en marcha de la plataforma de soporte en sus pasarelas residenciales. Si el cliente no dispone del equipo pasarela residencial, se suministra tal equipo. En la tabla 12, se muestra el presupuesto adicional que supone la instalación en una pasarela existente. Y en la tabla 13, se muestra el presupuesto adicional que supone el despliegue de la plataforma, si el cliente necesita la instalación de una pasarela residencial.

Concepto	Importe (€)
Costes mano de obra (Técnico de redes)	40
Costes de desplazamiento	20
Total base imponible	60
IVA (18%)	10,8
Total	**70,8**

Tabla 12: Presupuesto adicional de puesta en marcha

Concepto	Importe (€)
Pasarela residencial (2Wire 3800HGV-B Gateway)	300
Costes mano de obra (Técnico de redes)	40
Costes de desplazamiento	20
Total base imponible	360
IVA (18%)	64,8
Total	**424,8**

Tabla 13: Presupuesto adicional de puesta en marcha con instalación de pasarela

Glosario de términos

Glosario de términos

API	Application Programming Interface
ARP	Address Resolution Protocol
CE	Consumer Electronics
CP	Control Point
DCP	Device Control Protocol
DHCP	Dynamic Host Configuration Protocol
DMA	Digital Media Adapter
DNS	Domain Name System
DSL	Digital Subscriber Line
GENA	General Event Notification Architecture
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPMU	HTTP Multicast over UDP
HTTPU	HTTP (unicast) over UDP
IGD	Internet Gateway Device
IP	Internet Protocol
ISP	Internet Service Provider
NAT	Network Adress Translation
NT	Notification Type
NTS	Notification Sub Type
OSGi	Open Services Gateway Initiative
PPP	Point-to-point Protocol
RPC	Remote Procedure Call
SC	Security Console
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
SSL	Secure Socket Layer
SST	Service State Table
ST	Service Type
TCP	Transmission Control Protocol
UDA	UPnP Device Arquitecture
UDN	Unique Device Name
UDP	User Datagram Protocol
UPC	Universal Product Code
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
USN	Unique Service Name
UUID	Universally Unique Identifier
XML	Extensible Markup Language

Referencias bibliográficas

Fuentes bibliográficas

1. **Pérez Guzmán, Julio César - Edificios inteligentes.** Monografías.com. [En línea] http://www.monografias.com/trabajos15/edific-inteligentes/edific-inteligentes.shtml.
2. **Inmótica, Concepto de.** Wikipedia. [En línea] http://es.wikipedia.org/wiki/Inm%C3%B3tica.
3. **Domótca, Concepto de.** Portal Web sobre Domótica. [En línea] http://www.domotica.net.
4. **Barzanallana, Rafael - Curso de introducción a la domótica.** [En línea] Marzo de 2009. http://www.wikilearning.com/curso_gratis/domotica_automatizacion_de_viviendas-introduccion/3063-1.
5. **Hogar Digital, Concepto de.** Wikipedia . [En línea] http://es.wikipedia.org/wiki/Hogar_digital.
6. **Pasarela Residencial, Introducción al concepto de.** Portal CasaDomo. [En línea] http://www.casadomo.com/noticiasDetalle.aspx?c=49&idm=60.
7. **OSGi, Web oficial de.** [En línea] www.osgi.org/.
8. **UPnP Forum, Web oficial del.** [En línea] www.upnp.org/.
9. **Martino, Mariano - Ocio Digital.** comunidadconecta.ning.com. *Blog post.* [En línea] Julio de 2010. http://comunidadconecta.ning.com/profiles/blogs/ocio-digital-nuevos-entornos.
10. **HomePlug, Web oficial de.** [En línea] http://www.homeplug.org/.
11. **HomePNA, Web oficial de.** [En línea] http://www.homepna.org/.
12. **grupo de trabajo del estándar IEEE 802.11, Web oficial del.** [En línea] http://www.ieee802.org/11/.
13. **Figura 1, Osgi como pasarela de servicios.** CasaDomo. [En línea] http://www.casadomo.com/noticiasDetalle.aspx?c=49&idm=60.
14. **Jini, Web oficial de.** jini.org. [En línea] http://jini.org.
15. **Montero, Roberto - Tutorial sobre OSGi.** [En línea] http://www.javahispano.org/contenidos/archivo/14922/OSGI_Roberto_Montero.pdf.
16. **Figura 2, Arquitectura de software OSGi.** osgi.org. [En línea] http://www.osgi.org/About/WhatIsOSGi.
17. **Figura 4, Ciclo de vida de un bundle.** springsource.org. [En línea] Fuente: http://static.springsource.org.
18. **Figura 5, Plataforma OSGi.** blog.ethomasjoseph.com. [En línea] http://blog.ethomasjoseph.com.
19. **OSGi, Releases de.** OSGi Alliance. [En línea] http://www.osgi.org/Specifications/HomePage.
20. **Knopflerfish, Web oficial de.** [En línea] http://www.knopflerfish.org/.
21. **Prosyst, Web oficial de.** [En línea] http://www.prosyst.com/.
22. **HitachiSoft SuperJ Engine Framework, Web oficial de.** [En línea] http://www.hitachisoft.com.
23. **Apache Felix Framework, Web oficial de.** [En línea]

http://felix.apache.org/site/index.html.
24. **desarrollo del framework Equinox, Web oficial del.** www.eclipse.org. [En línea] http://www.eclipse.org/equinox/.
25. **Newton Project, Desarrollo del proyecto .** [En línea] http://newton.codecauldron.org/.
26. **Concierg, Desarrollo de.** Sourceforge. [En línea] http://concierge.sourceforge.net/.
27. **Eclipse, Entorno de desarrollo.** [En línea] www.eclipse.org.
28. **Vogel, Lars - OSGi with Eclipse Equinox.** www.vogella.de. [En línea] http://www.vogella.de/articles/OSGi/article.html.
29. **Forum, UPnP.** [En línea] http://www.upnp.org/.
30. **Arquitectura TCP/IP, Tutorial sobre la.** Network Working Group. *ietf.org.* [En línea] Enero de 1991. http://tools.ietf.org/html/rfc1180.
31. **Aruitectura de dispositivos UPnP, Especificación de la.** [En línea] http://upnp.org/index.php/sdcps-and-certification/standards/device-architecture-documents/.
32. **DCP's, Especificación de los.** [En línea] http://upnp.org/sdcps-and-certification/standards/sdcps/.
33. **Figura 7, Proceso de descubrimiento UPnP.** www.upnp.org. [En línea] http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.
34. **Figura 12, Pila UPnP.** Microsoft.com. [En línea] http://technet.microsoft.com/en-us/library/bb457049.aspx.
35. **HTTPU y HTTPMU, Borrador de la especificaciones.** ietf.org. [En línea] Diciembre de 1999. http://tools.ietf.org/html/draft-goland-http-udp-00.
36. **HomeAPI, Web oficial de.** [En línea] http://www.homeapi.org/.
37. **DLNA, Web oficial de.** [En línea] http://www.dlna.org/.
38. **DPWS, Especificación de.** Oasis.org. [En línea] Julio de 2009. http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01.
39. **Sun MicroSystem, Web oficial de.** Oracle. [En línea] http://www.oracle.com/us/sun/index.html.
40. **arquitectura UPnP AV, Especificación de la.** UPnP Forum. [En línea] http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1-20020625.pdf.
41. **Windows Rally, Información sobre.** [En línea] http://msdn.microsoft.com/en-us/windows/hardware/gg463018.
42. **.NET Framework Micro, Web oficial de.** Microsoft. [En línea] http://www.microsoft.com/netmf/default.mspx.
43. **W3C, Estandares del.** [En línea] http://www.w3c.es/estandares/.
44. **Servicios Web Oasis, Comité de.** www.oasis-open.org. [En línea] http://www.oasis-open.org/committees/tc_cat.php?cat=ws.
45. **SOA4D, Web de desarrollo de.** [En línea] https://forge.soa4d.org/.
46. **WS4D, Web oficial de.** [En línea] http://www.ws4d.org/.
47. **Teirikangas, Jussi - Documentación sobre Havi.** Helsinki University of Technology. [En línea] http://www.csun.edu/~andrzej/COMP529-S05/papers/jussi_teirikangas.pdf.
48. **IEEE 1394, Información sobre el estándar.** Wikipedia. [En línea] http://es.wikipedia.org/wiki/IEEE_1394.
49. **Webcasting, Información sobre el término.** Wikipedia. [En línea] http://en.wikipedia.org/wiki/Webcast.
50. **OpenDomo, Web oficial de.** [En línea] http://es.opendomo.org/.
51. **Alianza IGRS, Sitio Web de la.** [En línea] http://www.igrs.org/indexen.aspx.
52. **DHWG, Web oficial del.** [En línea] www.dhwg.org/.

53. **NAT, Terminología y consideraciones del mecanismo.** Network Working Group - RFC 2663. *ietf.org.* [En línea] Agosto de 1999. http://tools.ietf.org/html/rfc2663.
54. **Figura 13, Modelo conceptual del IGD.** [En línea] http://upnp.org/specs/gw/igd1/.
55. **Figura 14, Jerarquía de servicios y dispositivos de un IGD.** [En línea] http://upnp.org/specs/gw/igd1/.
56. **IGD de UPnP, Especificación del.** www.upnp.org. [En línea] http://upnp.org/specs/gw/UPnP-gw-InternetGatewayDevice-v1-Device.pdf.
57. **WANPPPConection, Especificación del servicio.** [En línea] http://www.upnp.org/specs/gw/UPnP-gw-WANPPPConnection-v1-Service.pdf.
58. **DHCP, Especificación de.** Network Working Group - RFC 2131. *ietf.org.* [En línea] http://www.ietf.org/rfc/rfc2131.txt.
59. **IPv4, Agotamiento de direcciones.** Wikipedia. [En línea] http://es.wikipedia.org/wiki/Agotamiento_de_las_direcciones_IPv4.
60. **RFC 1918, Address Allocation for Private Internets.** [En línea] http://www.faqs.org/rfcs/rfc1918.html.
61. **IP, Especificación de la tecnologia multicast de.** Network Working Group - RFC 1112. *ietf.org.* [En línea] Agosto de 1989. http://www.ietf.org/rfc/rfc1112.txt.
62. **Lin, Wen-Wei y Sheng, Yu-Hsiang.** "Using OSGi UPnP and Zigbee to Provide a Wireless Ubiquitous Home Healthcare Environment". *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008.* The Second International Conference, UBICOMM '08.
63. **tecnologia Zigbee, Especificación de la.** [En línea] http://www.zigbee.org/Specifications.aspx.
64. **Felix UPnP, Documentación sobre.** Apache Felix. [En línea] http://felix.apache.org/site/apache-felix-upnp.html.
65. **OSGi Service Release 4, Especificación del.** Osgi.org. [En línea] http://www.osgi.org/Release4/HomePage.
66. **Madrid, N.M., y otros.** "Integration of an advanced emergency call subsystem into a car-gateway platform". Design, Automation & Test in Europe Conference & Exhibition, Septiembre de Abril 2009.
67. **Zhao, Xiaoyu, Wang, Lan y Gu, Daqing.** "UPnP Extensions for Public IPv4 Sharing in IPv6 Environment". *Networking and Services (ICNS).* 2010 Sixth International Conference on, Marzo de Marzo 2010.
68. **Li, Chung-Sheng, Huang, Yueh-Min y Chao, Han-Chieh.** "UPnP IPv4/IPv6 bridge for home networking environment". Consumer Electronics, IEEE Transactions on, Noviembre de November 2008.
69. **Ortiz, F. y Palet, J.** "IPv6 for Home Automation". *Applications and the Internet Workshops.* Saint Workshops 2005. The 2005 Symposium on, Enero de 2005.
70. **IMS, Especificación de la tecnología.** 3GPP. [En línea] http://www.3gpp.org/article/ims.
71. **seguridad de UPnP, Especificación de.** upnp.org. [En línea] http://upnp.org/specs/sec/security/.
72. **Sales, T., y otros.** "Towards the UPnP-UP: Enabling User Profile to Support Customized Services in UPnP Networks". *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008.* UBICOMM '08. The Second International Conference on ,, 2008.
73. **TLS/SSL, Especificaión del protocolo.** Network Working Group - RFC 2246. *ietf.org.* [En línea] 1999. http://www.ietf.org/rfc/rfc2246.txt.
74. **Konno, Satoshi.** Proyecto Cybergarage. [En línea] http://www.cybergarage.org.
75. **CyberLink, API online de.** [En línea]

http://cgupnpjava.sourceforge.net/javadoc/html/.
76. **Developer Tools for UPnP, Web oficial de.** [En línea]
http://opentools.homeip.net/dev-tools-for-upnp.
77. **DMZ, Concepto de.** Wikipedia. [En línea]
http://es.wikipedia.org/wiki/Zona_desmilitarizada_(inform%C3%A1tica).
78. **Framework OSGi Equinox Standalone, Tutorial sobre el.** eclipse.org. [En línea]
http://www.eclipse.org/equinox/documents/quickstart-framework.php.
79. **Libreria KXML2, Repositorio de la.** SourceForge KXML. [En línea]
http://kxml.sourceforge.net/kxml2/.
80. **apache Xerces, Web del proyecto.** Apache. [En línea] http://xerces.apache.org/.
81. **Protocolo PPP, Especificación del.** Network Working Group - RFC 1661. *ietf.org.* [En línea] Julio de 1994. http://tools.ietf.org/html/rfc1661.
82. **Serializacion en Java, Documentación sobre la.** Oracle. [En línea]
http://java.sun.com/developer/technicalArticles/Programming/serialization/.

Anexos

Anexo A

A.1. Protocolos y tecnologías usados en UPnP

Las tecnologías y protocolos de comunicaciones utilizados y en los que se basa UPnP se resumen en el resto de este anexo. Para más información sobre estas tecnologías se debe recurrir a las referencias de este documento.

TCP/IP (*Transmission Control Protocol / Internet Protocol*)

Conjunto de protocolos a nivel de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. En concreto TCP/IP hace referencia a los dos protocolos más importantes que la componen. Esta pila de protocolos sirve como base para el resto de los protocolos que constituyen UPnP. Los dispositivos UPnP pueden utilizar muchos de los protocolos en la pila TCP/IP, como TCP, UDP, IGMP, ARP, HTTP, DHCP, DNS. Estos protocolos y servicios se utilizan para proporcionar lo necesario para una red IP sobre la que se despliega UPnP. Con su uso se asegura la anteriormente mencionada interoperabilidad en diferentes medios físicos e independencia de las plataformas.

HTTP, HTTPU, HTTPMU (*HyperText Transfer Protocol*)

HTTP define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición se lo conoce como agente del usuario. A la información transmitida se la llama recurso y se la identifica mediante una URL. Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc. Todos los aspectos de UPnP se construyen en la parte superior de HTTP o sus variantes.

HTTPU (y HTTPMU) son variantes de HTTP para enviar mensajes definidos en la parte superior de UDP/IP en lugar de TCP/IP. Estos protocolos son utilizados por SSDP, que se describe a continuación. El formato de mensaje básico usado por estos protocolos se adhiere a el de HTTP y es necesaria tanto para las comunicaciones *multicast* como *unicast*, y cuando la entrega de mensajes no requiere la sobrecarga asociada con la fiabilidad. Algunas de las explicaciones de los protocolos de nivel superior y el funcionamiento de UPnP asumen un conocimiento básico del protocolo

HTTP.

SSDP (*Simple Service Discovery Protocol*)

Como su nombre lo indica, define la forma de servicios de red que se pueden descubrir en la red. SSDP se basa en HTTPU y HTTPMU y define los métodos, tanto para un punto de control para localizar recursos de interés en la red, como a dispositivos para anunciar su disponibilidad en la red. Al definir el uso de solicitudes de búsqueda y la presencia de anuncios, SSDP elimina los gastos generales que serían necesarias si sólo uno de estos mecanismos se utiliza. Como resultado, cada punto de control en la red tiene la información completa de la red estatal de mantenimiento, mientras que el tráfico de la red baja. Del mismo modo, un dispositivo al ser conectado a la red, envía múltiples anuncios SSDP publicitando los servicios que soporta.

Tanto los anuncios de presencia como los mensajes de respuesta *unicast* de un dispositivo contienen una URL a la ubicación del documento de descripción del mismo, que tiene información sobre el conjunto de servicios soportados por él.

Además de las capacidades de descubrimiento, SSDP también proporciona notificación de desconexión de la red para un dispositivo y el servicio(s) asociado(s) e incluye el tiempo de caché (timer) para purgar la posible información desactualizada.

DHCP (*Dynamic Host Configuration Protocol*)

DHCP es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente. Está basado en el protocolo de transporte inseguro UDP.

SOAP (*Simple Object Access Protocol*)

SOAP define el uso de XML y HTTP para ejecutar llamadas a procedimientos remotos en entornos descentralizados y distribuidos. Se está convirtiendo en el estándar para la comunicación basada en RPC (*Remote Procedure Call*) a través de Internet. Al hacer uso de la infraestructura de Internet existente, puede trabajar eficazmente con los *firewalls* y *proxies*. SOAP también puede hacer uso de SSL (*Secure Sockets Layer*) para la seguridad y el uso de la conexión HTTP de gestión de instalaciones, con lo que la comunicación distribuida a través de Internet es tan fácil como acceder a páginas web.

Como una llamada a procedimiento remoto, UPnP utiliza SOAP para entregar mensajes a los dispositivos de control y devolver los resultados o errores de nuevo a los puntos de control. Cada solicitud de control de UPnP es un mensaje SOAP que contiene la acción de invocar, junto con un conjunto de parámetros. La respuesta es un mensaje que contiene la condición satisfactoria (si procede) de la respuesta, valor de retorno y parámetros.

GENA (*General Event Notification Architecture*)

GENA fue definido para proporcionar la capacidad de enviar y recibir notificaciones mediante HTTP a través de TCP/IP y UDP *multicast*. A su vez, define los conceptos de subscriptores y editores de notificaciones.

El formato de GENA en UPnP sirve para crear la presencia de anuncios que se envían utilizando SSDP y para proporcionar la capacidad de señalizar los cambios en el servicio de eventos de UPnP. Un punto de control interesado en recibir notificaciones de eventos se suscriben a un tipo de eventos concreto mediante el envío de una solicitud que incluye el servicio de interés, una dirección para enviar los acontecimientos y una suscripción para la notificación de eventos. La suscripción debe ser renovada periódicamente para seguir recibiendo las comunicaciones, y también puede ser cancelado utilizando GENA. Siempre hay garantía en UPnP de que un mensaje de eventos se entrega a un suscriptor, ya que GENA utiliza TCP para el transporte.

XML (*Extensible Markup Language*)

XML es un metalenguaje extensible de etiquetas desarrollado por el W3C. Dicho de otro modo, es el formato universal para datos estructurados en la Web, es una manera de poner casi cualquier tipo de datos estructurados en un archivo de texto. XML se parece mucho a HTML en la medida en que utiliza etiquetas y atributos. En realidad, es bastante diferente en cuanto a su significado, ya que sus etiquetas y atributos no están definidos a nivel mundial, sino que se interpretan en el contexto de su uso. Estas características hacen de XML un buen ajuste para el desarrollo de esquemas para diversos tipos de documentos.

XML es una parte fundamental de la utilizada en el dispositivo UPnP y descripciones de servicios, mensajes de control y eventos.

A.2. Documento XML de descripción de un dispositivo UPnP

En el cuadro de texto 14, se muestra un descriptor XML genérico de un dispositivo UPnP.

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>URL base para todas las urls relativas</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:TipoDeDispositivo:version</deviceType>
    <friendlyName>nombre corto</friendlyName>
    <manufacturer>nombre del fabricante</manufacturer>
    <manufacturerURL>url del fabricante</manufacturerURL>
    <modelDescription>nombre largo y descriptivo del dispositivo</modelDescription>
    <modelName>nombre del modelo</modelName>
    <modelNumber>numero del modelo</modelNumber>
    <modelURL>url al sitio del modelo</modelURL>
    <serialNumber>numero de serie del fabricante</serialNumber>
    <UDN>identificador unico y universal del dispositivo</UDN>
    <UPC>codigo universal del producto</UPC>
    <iconList>
      <icon>
        <mimetype>formato imagen</mimetype>
        <width>pixeles horizontales</width>
        <height>pixeles verticales</height>
        <depth>profundidad del color</depth>
        <url>URL de localizacion del icono</url>
      </icon>
        ...
    </iconlist>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:TipoServicio:version</serviceType>
        <serviceId>urn:upnp-org:serviceId:IDServicio</serviceId>
        <SCPDURL>URL para la descripcion del servicio</SCPDURL>
        <controlURL>URL para control</controlURL>
        <eventSubURL>URL para eventos</eventSubURL>
      </service>
        ... otros servicios definidos por el Foro UPnP
        ... otros servicios de valor añadido creados por el fabricante
    </serviceList>
    <deviceList>
      ... otros dispositivos embebidos definidos por el Foro UPnP
      ... otros dispositivos embebidos añadidos por el fabricante
    </deviceList>
  </device>
</root>
```

Cuadro de texto 14: Documento XML de descripción de dispositivo UPnP

A.3. Documento XML de descripción de un servicio UPnP

En el cuadro de texto 15, se muestra un descriptor XML genérico de un servicio UPnP.

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0>
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <serviceStateTable>
    <stateVariable sendEvents="yes"> (yes: si se envía evento cuando cambia)
      <name>nombre variable</name>
      <dataType>tipo de datos</dataType>
      <allowedValueList>
        <allowedValue>valor enumerado</allowedValue>
          ... otros valores definidos por el Foro UPnP
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>nombre variable</name>
      <dataType>tipo de datos</dataType>
      <defaultValue>valor por defecto</defaultValue>
      <allowedValueRange>
        <minimum>valor minimo permitido</minimum>
        <maximum>valor maximo permitido</maximum>
        <step>incremento</step>
      </allowedValueRange>
    </stateVariable>
      ... otras variables de estado definidas por el Foro UPnP
      ... otras variables de estado definidas por el fabricante
  </serviceStateTable>
  <actionList>
    <action>
      <name>nombre accion</name>
      <argumentList>
        <argument>
          <name>nombre parametro</name>
          <direction>direccion: entrada (in) o salida (out)</direction>
          <relatedStateVariable>variable de estado relacionada</relatedStateVariable>
        </argument>
          ... otros argumentos definidos por el Foro UPnP
    </action>
      ... otras acciones definidas por el Foro UPnP
      ... otras acciones definidas por el fabricante
  </actionList>
</scpd>
```

Cuadro de texto 15: Documento XML de descripción de servicio UPnP

A.4. CyberLinkForJava

CyberLinkForJava hace referencia a una librería de desarrollo en el lenguaje Java, que nace dentro del proyecto 'CyberGarage' que empezó a desarrollarse por Satoshi Konno en el 2002. Es una implementación de código abierto de la tecnología UPnP y ofrece el soporte para crear dispositivos y puntos de control, ya que es capaz de controlar los protocolos en los que se basa UPnP automáticamente. Esta librería puede trabajar en cualquier plataforma que soporte la ejecución de una JVM. Actualmente ha salido la versión 2.0 de esta librería.

Existen otras implementaciones de CyberLink para otros lenguajes como 'C', 'C++', 'Objetive C' y 'Perl'. Además existen implementaciones de dispositivos creadas como ejemplo dentro del repositorio de CyberLink, así como también una implementación de un punto de control. En este caso nos centraremos en la librería desarrollada en Java, ya que es el lenguaje en el que se desarrollara la plataforma en este proyecto.

A continuación se explicará el contenido de la librería CyberLink, indicando que parámetros generales hay que tener en cuenta y son modificables en el código del paquete. Posteriormente se proporcionara una visión de la estructura de clases necesarias para la creación de dispositivos y puntos de control de UPnP.

Configuración

Para el correcto funcionamiento de la librería, es importante también importar una librería adicional junto con la librería CyberLink, llamada *KXML2* [79] para parsear los documentos XML que incluyen los mensajes SOAP de UPnP y los ficheros de descripción de los dispositivos. Además hay otra librería necesaria para crear, manejar y validar documentos XML llamada *Apache Xerces* [80].

Antes de empezar a desarrollar dispositivos propios con la librería CyberLink, el desarrollador debe tener en cuenta en qué escenario va a desplegar esos dispositivos. Por ejemplo, CyberLinkForJava soporta la versión 6 del protocolo IP (*IPv6*), además de la muy conocida y usada en el mundo IPv4, cuando se programe con estas librerías se podrá indicar si se quieren usar solo interfaces IPv6. En el ámbito de este proyecto nos centraremos en redes IPv4, que son las más extendidas en el mundo y debido a que aún le queda mucho camino al despliegue de IPv6.

Otra característica modificable de CyberLink es el alcance local de la técnica de *multicasting* para algunos de los protocolos que utiliza UPnP. En el caso de redes IPv6, las técnicas multicast no presentan las mismas limitaciones que la versión 4 de IP.

Paquetes más relevantes de la librería para este proyecto

A continuación se comentarán aquellos aspectos de la lógica de la librería que resultan interesantes para este proyecto y que son usados para implementar alguna funcionalidad.

- *org.cybergarage.http*: para la creación y configuración de los servidores HTTP necesarios en la comunicación, así como los mensajes tipo de solicitud y respuesta HTTP, y toda la funcionalidad asociada a este protocolo.

- *org.cybergarage.net*: para la configuración de los parámetros de red.

- *org.cybergarage.soap*: para la implementación del protocolo SOAP para la invocación remota de acciones.

- *org.cybergarage.upnp*: cubre toda la lógica para la creación de puntos de control, dispositivos y servicios, además de la implementación de los mecanismos de descubrimiento (SSDP), de control (SOAP) y eventos (GENA) que especifica UPnP.

- *org.cybergarage.util*: contiene un conjunto de clases e interfaces que son utilizadas como herramientas para la implementación del código del resto de paquetes.

- *org.cybergarage.xml*: para dar soporte al tratamiento, análisis y generación de documentos en formato XML que se utilizan para la creación de mensajes en varios de los protocolos UPnP, así como los descriptores de dispositivos y servicios.

Creación de dispositivos con esta librería

Documento de descripción del dispositivo y sus servicios

El paso previo a la implementación del dispositivo es la creación de los documentos XML descriptores tanto del dispositivo como de los servicios que contiene. Estos ficheros serán cargados al inicio de la comunicación para estar disponibles en la petición de los mismos por parte del Punto de Control durante el proceso de descubrimiento.

El campo *URLBase* del descriptor del dispositivo no debe figurar en este ya que es generado dinámicamente cuando se recibe la petición del punto de control para obtener este documento.

Crear Dispositivos

Existe una clase *Device* (Dispositivo) principal que puede identificar a un dispositivo raíz, almacenando en listas los posibles dispositivos embebidos y servicios que contiene. A la hora de crear un dispositivo con esta librería, se le puede asignar directamente un documento XML que se haya creado anteriormente para que sea su documento de descripción. Solo los dispositivos diseñados como raíz pueden ser activados usando el método *start()* de la clase Device, para iniciar todo el proceso de comunicación del dispositivo UPnP. La invocación de este método desencadena la creación de un servidor HTTP por interfaz, para poder recibir los mensajes de los distintos protocolos UPnP (SOAP, GENA y HTTP GET para la obtención de los

documentos de descripción) y dos sockets, uno *unicast* y otro *multicast*, para el protocolo SSDP. Estos servidores http coexisten en tiempo de ejecución, debido a que se ejecutan en varios hilos paralelos para el envío de peticiones, la recepción de los mensajes y la generación de las correspondientes respuestas a los puntos de control, conforme a lo especificado para cada protocolo mencionado. Cada vez que se recibe una solicitud HTTP enviada por el Punto de Control, se clasifica según el tipo de mensaje UPnP sea. Esto se realiza en el método *httpRequestReceived*() de Device.

A continuación se muestran en la tabla 14, una serie de parámetros por defecto con los que se crea el dispositivo y que son modificables a priori de la ejecución del dispositivo.

Parámetro	Valor por defecto	Método	Detalle
HTTP Port	4004	setHTTPPort()	Puerto http usado
Descripción URI	/descripción.xml	setDescriptionURI()	URI del dispositivo raíz
Lease time	1800	setLeaseTime()	Tiempo entre réplicas de mensajes (milisegundos)

Tabla 14: Parámetros por defecto de un dispositivo en Cyberlink

A la hora de anunciarse el dispositivo en la red UPnP mandará cada cierto tiempo mensajes *'ssdp::alive'* mientras siga en ejecución. Cuando el dispositivo se para usando el método *stop()* también lo notificara a la red con un mensaje *'ssdp::byebye'*. Según la implementación de la librería estos mensajes SSDP se pueden publicar en cualquier momento usando los métodos *announce()* y *byebye()*. En el momento en que recibe un mensaje de descubrimiento *'m-search'* del punto de control, el dispositivo manda una respuesta automáticamente al punto de control. Todos estos mensajes comentados se envían por duplicado separados temporalmente por el parámetro *Lease_time*. Así mismo se usa la misma clase para crear posibles dispositivos embebidos. Son accesibles en todo momento por el dispositivo raíz, mediante el método *getDeviceList()* en forma de lista conjunta o *getDevice()* para buscar un dispositivo en concreto (por nombre o UDN).

Crear servicios y variables de estado

Al igual que se pueden crear dispositivos a partir de la clase '*Device'*, también se pueden crear servicios, variables de estado, acciones e incluso argumentos para esas acciones, a partir de clases Java 'esqueleto' ya existentes en esta librería, véase clases como '*Service'*, '*Action'*, '*StateVariable'* o '*Argument'*.

Se usa el método *getServiceList()* de la clase Device, para acceder a los servicios que implementa el dispositivo. Se usa el método *getActionList()* del servicio para conseguir su lista de acciones, y el método *getServiceStateTable()* para las variables de estado.

Se puede buscar un servicio en concreto dentro del dispositivo a través del identificador del servicio usando *getService()*. Lo mismo pasa con *getAction()* para el caso de las acciones y *getStateVariable()* para el caso de una variable de estado en concreto.

Crear mecanismos de control y eventos

En esta librería, el mecanismo para recibir invocaciones de control de acciones desde los puntos de control, se hace a través de la interfaz '*ActionListener*'. El dispositivo (o servicio) que implemente este escuchador tiene que implementar el método '*actionControlReceived()*', que tiene la acción que ha provocado el evento, como parámetro de entrada. La acción es interpretada y el método devuelve la variable booleana '*true*' para indicar que la petición es válida. Se genera una respuesta *UPnPError* al punto de control automáticamente, cuando la acción es inválida (el valor de retorno es false o el dispositivo no tiene interfaz implementada). Esta clase UPnPError se utiliza además para generar otro tipo de errores que surgen al poner en funcionamiento la red. Por defecto es *INVALID_ACTION* pero se puede modificar con el respectivo método de la clase.

También Cyberlink da soporte para la implementación del mecanismo de control, *Query for variable*, para preguntar el valor de una variable de estado de un servicio, pero este mecanismo fue deprecado por el Foro UPnP. Si se desea averiguar que métodos y clases define Cyberlink para este mecanismo, se puede consultar en la guía de programación de esta librería que se encuentra disponible en la web de Cybergarage.

Todos estos escuchadores de eventos para que funcionen deben añadirse a los propios dispositivos o servicios con el método '*setActionListener*()' o '*setQueryListener*()' dependiendo de qué tipo de escuchador se trate.

El punto de control puede subscribirse para la recepción de eventos del dispositivo. Es el dispositivo y no el punto de control quien maneja los mensajes de subscripción automáticamente. A su vez, el dispositivo también maneja los mensajes que recibe periódicamente para renovar esta subscripción, tal y como especifica el estándar UPnP para este mecanismo de eventos. Para que el dispositivo establezca o actualice el valor de una variable de estado, se debe utilizar el método setValue() de la clase StateVariable. Si las modificaciones de esta variable de estado deben ser informadas mediante este mecanismo de eventos, se enviará el correspondiente evento a los subscriptores.

Diagrama de clases

En la figura 30, se muestra el diagrama de clases que proporciona CyberLink para crear tus propios dispositivos UPnP.

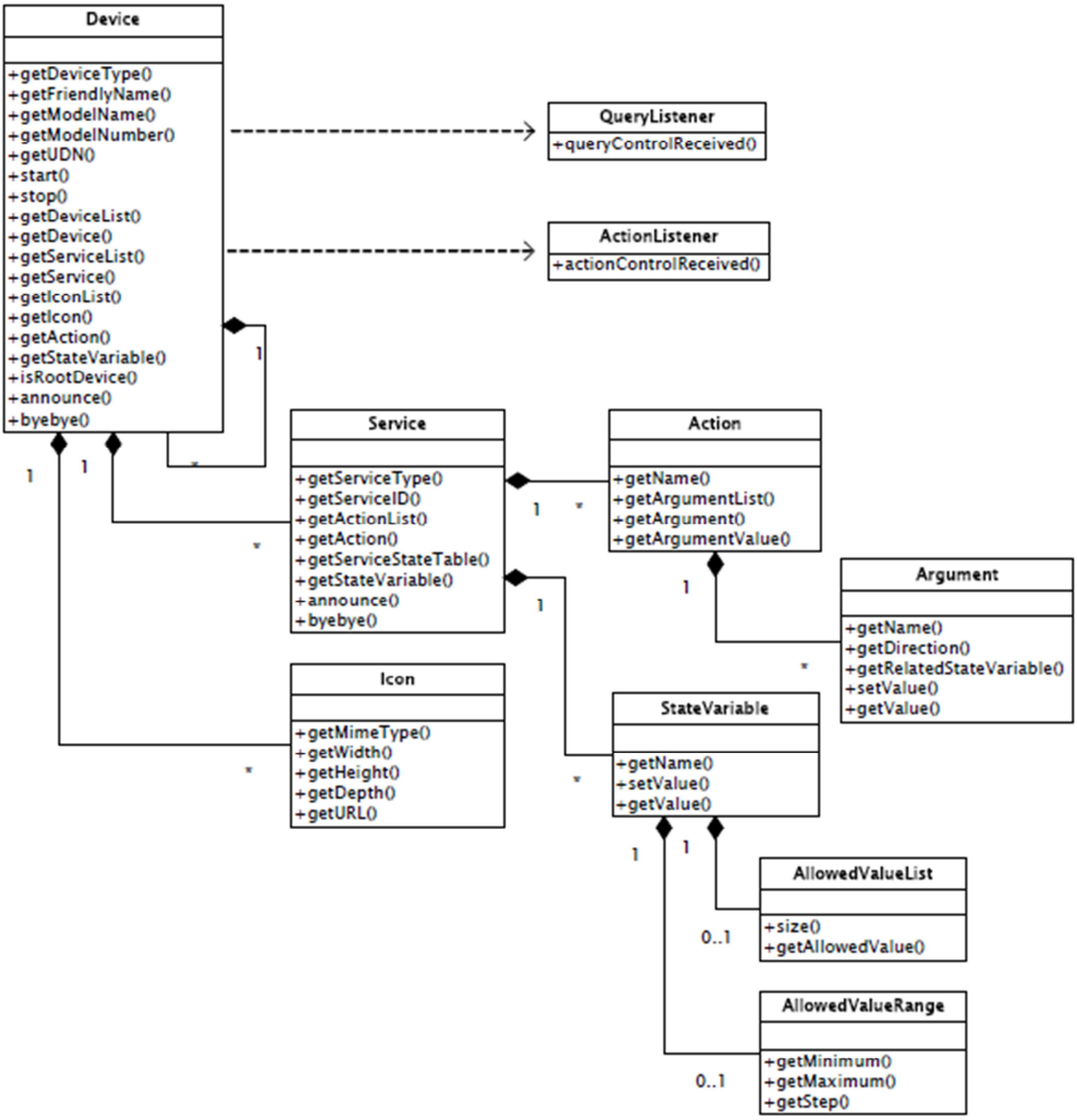

Figura 30: Diagrama de clases CyberLink para dispositivos

Fuente: http://sourceforge.net/projects/cgupnpjava/files/clinkjava-doc/2.0/

Creación de Puntos de Control con esta librería

Crear Puntos de Control

Para crear un punto de control UPnP, se debe crear una instancia de la clase *ControlPoint*. Se usa el método *start()* de la clase para activarlo. Es este método el que inicia todo el proceso de comunicación del dispositivo UPnP, enviando automáticamente a la red mensajes para iniciar la fase de descubrimiento de UPnP. El punto de control tiene varios procesos activos, y devuelve una respuesta automática cuando un dispositivo UPnP manda un mensaje al punto de control o se anuncia a la red.

Esto es debido a que el punto de control tiene servidores HTTPU y HTTMU internos implementados para manejar mensajes SSDP.

A continuación en la tabla 15, se muestran una serie de parámetros por defecto con los que se crea el punto de control y que son modificables a priori de su ejecución.

Parámetro	Valor por defecto	Método	Detalle
HTTP Port	8058	setHTTPPort()	Puerto http usado para suscribir eventos
Suscription URI	/eventSub	setEventSubURI()	URI para recibir eventos de suscripción
SSDP Port	8008	setSSDPPort()	Puerto para recibir respuestas SSDP
Search Response	3	setSearchMx()	Máxima espera para búsqueda de dispositivo

Tabla 15: Parámetros por defecto de un punto de control en Cyberlink

El punto de control recibe eventos de notificación de los dispositivos UPnP cada vez aparecen y desaparecen estos en la red. Aunque no es necesario implementar estos *listeners* de eventos de notificación, ya que los mensajes SSDP son 'escuchados' en todo momento en la red, se puede hacer uso de estos eventos implementando la interfaz *NotifyListener*. Así mismo se usa el método *search()* de la clase *ControlPoint* para actualizar la lista de dispositivos. Los dispositivos raíz descubiertos son añadidos automáticamente a la lista del punto de control, y se puede recibir la respuesta mediante la interfaz *SearchResponseListener*.

Control de eventos

Como ya se ha mencionado antes, el punto de control puede mandar mensajes de control de acción a los dispositivos que implementan escuchadores de estos eventos. Para enviar el mensaje de control de acción, se usa el método *setArgumentValue()* y el método *postControlAction()* de la clase *Action*. Se debe poner el valor de la acción a todos los argumentos de entrada (ignorándose los parámetros de salida si aparecen al enviar la acción), los argumentos de salida se pueden capturar.

El punto de control puede se puede suscribir también a eventos que se produzcan en los dispositivos, como el cambio de alguna variable de estado. Se usa el método *suscribe()* de la clase *ControlPoint* para suscribirse a un evento (este método devolverá una variable booleana dependiendo del éxito de la suscripción). El escuchador tiene que implementar la interfaz *EventListener*, y definir el método *eventNotifiReceived()*. También es posible conseguir el identificador de la suscripción y el *tiemout*. Cuando un dispositivo usa el método *setValue()* en una variable de estado, automáticamente se manda un mensaje con el nuevo estado a los subscriptores.

Diagrama de clases

En la figura 31, se muestra el diagrama de clases que proporciona CyberLink para crear tus propios puntos de control de UPnP.

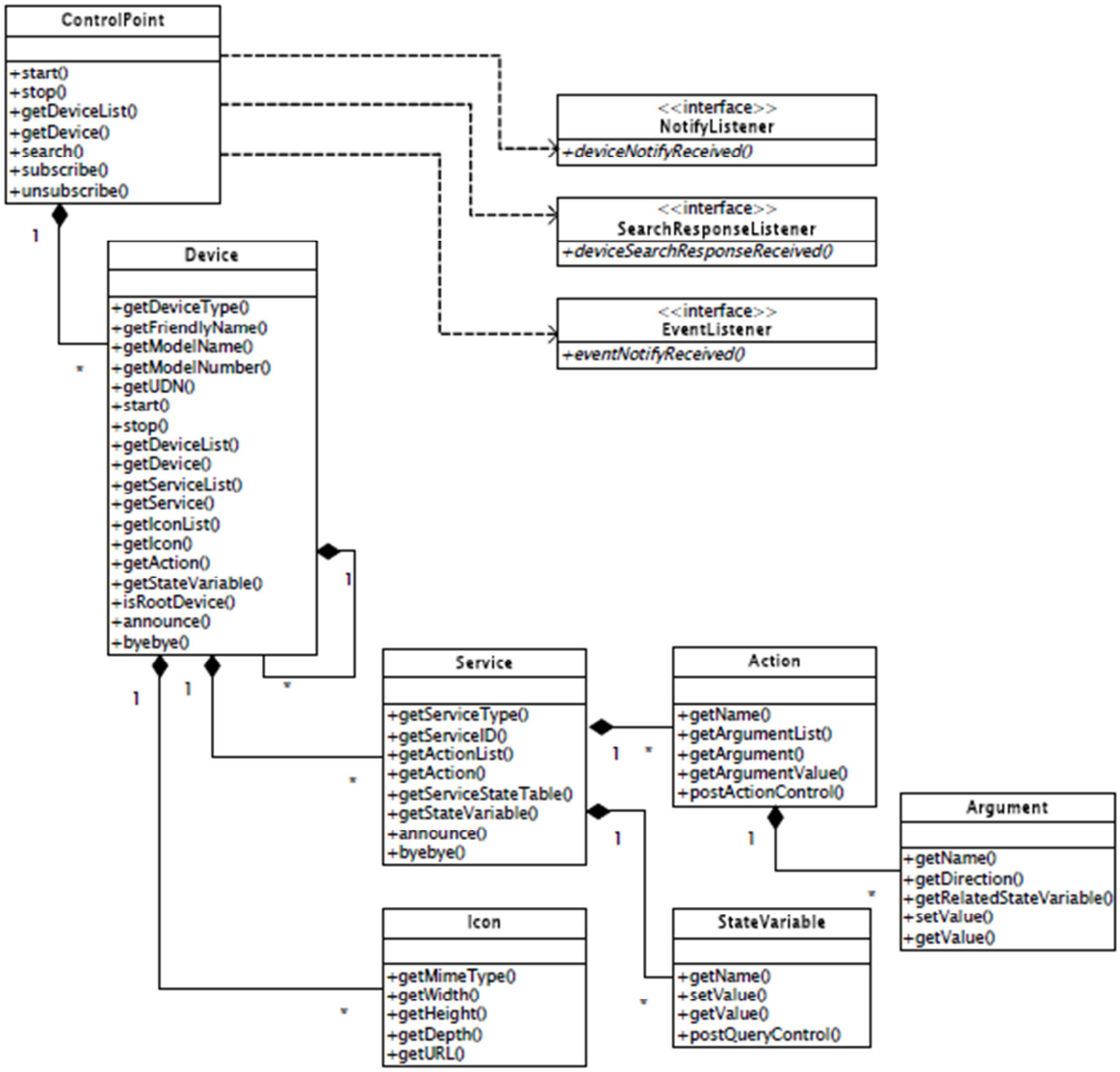

Figura 31: Diagrama de clases CyberLink para Puntos de Control

Fuente: http://sourceforge.net/projects/cgupnpjava/files/clinkjava-doc/2.0/

A.5. Variables de estado y acciones del servicio WANIPConnection

En la tabla 16, se puede observar las variables de estado que tiene este servicio, indicando el tipo de dato y valor permitido, valor por defecto y si es una variable requerida (u opcional). Están marcadas en la tabla las variables más relevantes para este proyecto.

Nombre de variable	Req. / Opc.	Tipo de dato	Valor permitido	Valor por defecto
ConnectionType	R	String	Depende de PossibleConnectionTypes	No especificado
PossibleConnectionTypes	R	String	Mirar especificación	No especific.
ConnectionStatus	R	String	Mirar especificación	No especific.
Uptime	R	ui4	No definido	No especific.
LastConnectionError	R	String	Mirar especificación	No especific
AutoDisconnectTime	O	ui4	>=0	No especific
IdleDisconnectTime	O	ui4	>=0	No especific.
WarnDisconnectDelay	O	ui4	>=0	No especific.
RSIPAvailable	R	boolean	0,1	No especific.
NATEnabled	R	boolean	0,1	No especific.
ExternalIPAddress	R	String	String de tipo “x.x.x.x”	String vacío
PortMappingNumberOfEntries	R	ui2	>=0	No especific.
PortMappingEnabled	R	boolean	0,1	No especific.
PortMappingLeaseDuration	R	ui4	0 a valor máximo de ui4	No especific.
RemoteHost	R	String	String de tipo “x.x.x.x”	Sting vacío
ExternalPort	R	ui2	Entre 0 y 65535	No especific.
InternalPort	R	ui2	Entre 1 y 65535	No especific.
PortMappingProtocol	R	String	Mirar especificación	String vacío
InternalClient	R	String	String de tipo “x.x.x.x”	String vacío
PortMappingDescription	R	String	No definido	String vacío

Tabla 16: Variables de estado del servicio WANIPConnection

A continuación se muestra en la tabla 17, una lista de las acciones especificadas, en la versión 1.0 del IGD, para este servicio. Aparecen marcadas las acciones más relevantes para este proyecto.

Acciones	
Obligatorias de implementar	Opcionales de implementar
SetConnectionType GetConnectionTypeInfo RequestConnection ForceTermination GetStatusInfo GetNATRSIPStatus GetGenericPortMappingEntry GetSpecificPortMappingEntry **AddPortMapping** **DeletePortMapping** **GetExternalIPAddress**	RequestTermination SetAutoDisconnectTime SetIdleDisconnectTime SetWarnDisconnectDelay GetAutoDisconnectTime GetIdleDisconnectTime GetWarnDisconnectDelay

Tabla 17: Acciones para el servicio WANIPConnection

A.6. Variables de estado y acciones del servicio WANPPPConnection

En la tabla 18, se puede observar las variables de estado que tiene este servicio, indicando el tipo de dato y valor permitido, valor por defecto y si es una variable requerida (u opcional). Están marcadas en la tabla las variables más relevantes para este proyecto.

Nombre de variable	Req. / Opc.	Tipo de dato	Valor permitido	Valor por defecto
ConnectionType	R	String	Depende de PossibleConnectionTypes	No especificado
PossibleConnectionTypes	R	String	Mirar especificación	No especific.
ConnectionStatus	R	String	Mirar especificación	No especific.
Uptime	R	ui4	No definido	No especific.
LastConnectionError	R	String	Mirar especificación	No especific
AutoDisconnectTime	O	ui4	>=0	No especific
IdleDisconnectTime	O	ui4	>=0	No especific.
WarnDisconnectDelay	O	ui4	>=0	No especific.
RSIPAvailable	R	boolean	0,1	No especific.
NATEnabled	R	boolean	0,1	No especific.
ExternalIPAddress	R	String	String de tipo “x.x.x.x”	String vacío
PortMappingNumberOfEntries	R	ui2	>=0	No especific.
PortMappingEnabled	R	boolean	0,1	No especific.
PortMappingLeaseDuration	R	ui4	0 a valor máximo de ui4	No especific.
RemoteHost	R	String	String de tipo “x.x.x.x”	Sting vacío
ExternalPort	R	ui2	Entre 0 y 65535	No especific.
InternalPort	R	ui2	Entre 1 y 65535	No especific.

PortMappingProtocol	R	String	Mirar especificación	String vacío
InternalClient	R	String	String de tipo “x.x.x.x”	String vacío
PortMappingDescription	R	String	No definido	String vacío
UpstreamMaxBitRate	R	ui4	>=0	No especific.
DownstreamMaxBitRate	R	ui4	>=0	No especific.

Tabla 18: Variables de estado para el servicio WANPPPConnection

A continuación se muestra en la tabla 19, una lista de las acciones especificadas, en la versión 1.0 del IGD, para este servicio. Aparecen marcadas las acciones más relevantes para este proyecto.

Acciones	
Obligatorias de implementar	Opcionales de implementar
SetConnectionType GetConnectionTypeInfo RequestConnection ForceTermination GetLinkLayerMaxBitRates GetStatusInfo GetNATRSIPStatus GetGenericPortMappingEntry GetSpecificPortMappingEntry **AddPortMapping** **DeletePortMapping** **GetExternalIPAddress**	ConfigureConnection RequestTermination SetAutoDisconnectTime SetIdleDisconnectTime SetWarnDisconnectDelay GetPPPEncryptionProtocol GetPPPCompressionProtocol GetPPPAuthenticationProtocol GetUserName GetPassword GetAutoDisconnectTime GetIdleDisconnectTime GetWarnDisconnectDelay

Tabla 19: Acciones para el servicio WANPPPConnection