EUSKO JAURLARITZA
GOBIERNO VASCO

Programa Saiotek 2006

SMARTLAB

Entorno de Trabajo Inteligente Colaborativo y Programable

Mecanismo de Descubrimiento

Tecnológico Fundación Deusto
Teknologikoa Deustu Fundazioa

HISTORIAL DE CAMBIOS

Versión	Descripción	Autor	Fecha	Comentarios
V0.1	Añadido Capítulo 1 – Estado del Arte	Iker Larizgoitia	14/05/2007	Por aprobar
V0.2	Añadido Capítulo 2 – Arquitectura de Descubrimiento Añadido Capítulo 3 – Arquitectura de Descubrimiento Añadido Capítulo 4 – Arquitectura de Descubrimiento Añadido Capítulo 5 – Arquitectura de Descubrimiento	Iker Larizgoitia	04/06/07	Por aprobar
V1.0	Añadidas Referencias. Añadido Resumen. Corregido formato del documento.	Iker Larizgoitia	23/01/2008	Revisado por Diego López de Ipiña.

TABLA DE CONTENIDOS

TABLA DE FIGURAS

1 INTRODUCCIÓN

El presente documento contiene toda la información relacionada con el mecanismo de descubrimiento de servicios realizado para el proyecto Smartlab.

El documento se divide en 5 apartados siendo su contenido el siguiente:

- ***Estado del arte***: comprende el estudio de los mecanismos y protocolos de descubrimiento más habituales utilizados en entornos ubicuos, así como un análisis de las características principales que hay que tener en cuenta a la hora de diseñar un mecanismo de descubrimiento.

- ***Modelo de descubrimiento***: contiene el modelo general definido para el proyecto Smartlab, así como los requisitos que debe cumplir dicho modelo.

- ***Diseño general del Mecanismo de Descubrimiento***: en este apartado se definen los aspectos principales del protocolo de descubrimiento, así como los bloques funcionales y sus reponsabilidades.

- ***Diseño OSGi del Mecanismo de Descubrimiento***: siendo OSGi la plataforma elegida, este apartado define el diseño adaptado a las características de OSGi de cada uno de los bloques funcionales definidos en el apartado anterior.

- ***Escenario de funcionamiento***: aquí se define una secuencia completa de ejemplo del funcionamiento del mecanismo de descubrimiento.

2 ESTADO DEL ARTE

2.1 Descubrimiento en entornos Ubicuos

Uno de los grandes caballos de batalla de la Computación Ubicua es conseguir definir un mecanismo de descubrimiento que se adapte a las características particulares de este tipo de entornos. Sin embargo, las propias características de los entornos ubicuos, principalmente la heterogeneidad de dispositivos y el dinamismo extremo de los mismos, hacen que sea difícil diseñar un mecanismo que se adapte a todos los escenarios posibles.

El descubrimiento es el proceso por el cual una entidad (cliente) es notificado espontáneamente de la disponibilidad de servicios o dispositivos de una red (recursos) [1]. En otras palabras, el descubrimiento es el proceso por el cual se referencian dinámicamente recursos en una red. Estas referencias pueden ser usadas por los clientes para contactar con dichos recursos y ejecutar acciones.

Cuando un dispositivo entra en una red, tiene que registrarse en el servicio de descubrimiento. Esto puede implicar que contacte en un servicio de directorio o que simplemente envíe anuncios periódicos a la red. Durante este proceso, los recursos de alguna manera proveerán de descripciones de ellos mismos (mediante tipos y atributos) y de información necesaria para los clientes (IP, puertos, …). Con esta información se podrán llevar a cabo procesos de selección o filtrado de los recursos por parte de los clientes que les ayuden a elegir los más adecuados para sus intereses en cada momento.

En resumen, se puede decir que un mecanismo de descubrimiento contempla tres fases principales:

- *Descubrimiento*: cómo un cliente descubre servicios a su alrededor.
- *Anuncio de servicios*: cómo un servidor da a conocer sus servicios.
- *Invocación de Servicios*: cómo un cliente invoca los servicios descubiertos.

Los protocolos de descubrimiento convencionales contemplan las dos primeras y algunos definen también la invocación a servicios.

2.2 Protocolos de descubrimiento

En este apartado se comentarán brevemente las características de los protocolos de descubrimiento más utilizados actualmente. Analizando cada uno de ellos se quieren extraer los aspectos que hay que considerar a la hora de diseñar un protocolo de descubrimiento. Estas características servirán para contrastarlas después con los requisitos del mecanismo de descubrimiento para Smartlab y poder así valorar si alguno de los protocolos existentes cumplen dichos requisitos o si hay que diseñar un mecanismo nuevo que se adapte al ámbito del proyecto.

2.2.1 SSDP (Simple Service Discovery Protocol)

SSDP es el mecanismo utilizado por UPnP para el descubrimiento. Está definido sobre HTTP y utiliza URIs para la identificación de los servicios. Cada servicio es una unidad funcional que es anunciada mediante HTTPMU[1] y las respuestas van directamente a la URI anunciada. El mecanismo de búsquedas es simple, por tipo de dispositivo y no soporta búsquedas múltiples ni por atributos. Sin embargo, los dispositivos pueden proporcionar una URL donde se encuentra una descripción en XML del mismo.

2.2.2 Jini

Jini es una arquitectura de descubrimiento propuesta por Sun Microsystems basada totalmente en Java. En esta arquitectura existen unas entidades que actuan de servicio de directorio (Jini Lookup Services) los cuales son descubiertos a través de multicast por los diferentes dispositivos. Este directorio da servicios de registro, búsqueda y notificación de cambios. En el Lookup Service lo que se registra es en realidad un proxy del servicio a invocar. Esto implica que ambos extremos deben soportar Java. Por lo tanto, la búsqueda tiene como objetivo descargarse en la parte cliente dicho proxy para poder invocarlo después por RMI.

El mecanismo de búsquedas se basa en atributos de los objetos java, por lo que se limita en cierta manera la heterogeneidad de los servicios. La validez de los servicios se gestiona a través de un mecanismo de leasing por el cual los servidores deben renovarse en el Lookup Service periódicamente.

[1] HTTP over multicast User Datagram Protocol

2.2.3 Bluetooth SDP (Service Discovery Protocol)

Este mecanismo es el utilizado en la pila de protocolos Bluetooth para descubrir servicios. Por la propia naturaleza de Bluetooth, está pensado para redes de corto alcance y con restricciones de eficiencia.

Utiliza un mecanismo P2P para el descubrimiento, dividiéndolo en dos fases. En una primera se descubre el dispositivo en sí y en una segunda los servicios que contiene. Los dispositivos cuentan con dos estados (descubrible y no descubrible) pensamos para el ahorro energético.

Existen unos servicios predefinidos que siguen una descripciones estándar mediante atributos, con unos identificativos preasignados (UUID). Estos identificativos son los que se utilizan para la búsqueda de servicios.

2.2.4 SLP (Service Location Protocol)

SLP está pensado para descubrir servicios en redes IP, de forma descentralizada y extensible. Se basa en el uso de URLs para localizar los servicios. Basándose en ellas los clientes pueden conocer e invocar los servicios diponibles.

Utiliza multicast para descubrir servicios en la red local y el protocolo es suficientemente flexible para el uso opcional de un servicio de directorio. Si existe se pasan los mensajes por él y si no existe, la comunicación es entre los diferentes clientes y servidores.

Las búsquedas se realizan mediante consultas LDAP y admite la introducción de predicados simples. Las respuestas a la búsqueda contienen, entre otros, un parámetro de tiempo de vida, que sirve para gestionar el tiempo de validez de los servicios.

2.2.5 Salutation

Salutation es una arquitectura para el descubrimiento pensada para entornos muy dinámicos. Como particularidad, este mecanismo es independiente del protocolo de transporte concreto que se esté utilizando.

Salutation define una arquitectura con tres elementos:

- Functional Units: define los servicios en sí, estando algunos de los más comunes estandarizados.

- Salutation Manager: actúa de servicio de directorio y es capaz de comunicarse con otros Salutation Managers para optimizar los procesos de búsqueda.

- Transport Manager: permiten aislar al SLM del protocolo de transporte que se esté utilizando en cada momento.

A parte del descubrimiento, Salutation también define tres modos de acceso a los servicios los cuales son: permitir una comunicación nativa entre cliente y servidor, utilizar traductores entre ambos cuando no hablan el mismo protocolo o seguir el propio mecanismo que está definido en Salutation.

2.2.6 Descubrimiento en OSGi

Habiendo sido OSGi la plataforma elegida para el desarrollo de Smartlab, es conveniente analizar los mecanismos de descubrimiento que proporciona dicha plataforma. La arquitectura de OSGi se basa en la creación de componentes llamados bundles que se registran en un servicio de directorio donde el resto de componentes pueden buscarlos para ejecutarlos o componer nuevos servicios.

OSGi está pensado como plataforma de ejecución, por lo que la búsqueda con el Service Registry es dentro de la propia JVM. Existen definidos unos servicios estándar como es el DAS (Device Access Specification) que permite el descubrimiento y anuncio dinámico de dispositivos y de los servicios ofrecidos por éstos. En DAS se definen unos componentes que se encargan de descubrir dispositivos con el protocolo que corresponda y después se localiza el bundle correspondiente al dispositivo concreto para instalarlo en el Service Registry y que sea accesible al resto de componentes.

2.2.7 Protocolos de descubrimiento eficientes energéticamente

En el momento en que entran en juego dispositivos desplegados en el entorno, uno de los factores a tener en cuenta no sólo en el descubrimiento, sino en el funcionamiento global de los dispositivos es el ahorro de energía.

El ahorro de energía debe estar integrado en todas las funciones del dispositivo, siendo el descubrimiento una de ellas.

Normalmente, el principal objetivo que se persigue en comunicaciones para ahorrar energía es reducir la cantidad de datos transmitidos. La consecución de este objetivo pasa por el diseño de mensajes de protocolo lo más pequeños posibles y la aplicación de estrategias que minimicen el número de mensajes enviados y la periodicidad de los mismos.

A modo de ejemplo de este tipo de técnicas, en [2] definen unas fases intermedias donde los dispositivos no anuncian directamente sus servicios hasta que un potencial cliente no envía un mensaje de “deseo” de descubrimiento, ahorrando de esta manera el número general de mensajes enviados. Además, dividen las posibles opciones del protocolo en estados (ahorro de energía y normal) con el objetivo de optimizar el uso de energía de los dispositivos.

2.2.8 Nuevas tendencias en descubrimiento

Los protocolos analizados hasta ahora son un grupo significativo del panorama del descubrimiento en entornos ubicuos. Ninguno es la panacea y cada uno tiene sus ventajas e inconvenientes dentro del ámbito de aplicación para el que fueron pensados.

Dentro de los aspectos que abarcan estos protocolos, todavía hay ciertas problemáticas que no están del todo resueltas. A modo de resumen, los principales retos para los nuevos protocolos de descubrimiento son:

- Proveer infraestructura sin necesidad de infraestructura.
- Conectar islas de descubrimiento, juntando varias tecnologías.
- Mecanismos de búsqueda adecuados a entornos ubicuos, por ejemplo, aplicando semántica a la definición de los servicios.

2.3 Características de los protocolos de descubrimiento

Después de haber analizado los mecanismos de descubrimiento más comunes y sus características, se puede concluir que los aspectos a tener en cuenta a la hora de diseñar un mecanismo de descubrimiento son:

- **Fases del protocolo**

  A la hora de diseñar un mecanismo de descubrimiento, lo primero que hay que considerar es hasta qué fases de todo el proceso se contemplan en el propio protocolo de las tres principales (Descubrimiento, Anuncio e Invocación).

  El nivel de complejidad aumenta a medida que se incorpora una fase al protocolo.

- **Transporte**

  El transporte condiciona el tipo de protocolo de red en el que se operará. La gran mayoría de los protocolos funcionan a modo de middleware para servicios IP (utilizando multicast, unicast, http, o combinaciones). Otros como el de Bluetooth no van sobre IP, pero tienen sus propias restricciones. En protocolos más complejos se pueden adoptar modelos neutros de transporte.

  El tipo de transporte normalmente está condicionado por el tipo de dispositivos que el protocolo quiere ser capaz de descubrir.

- **Búsquedas**

  El mecanismo de búsqueda de los protocolos de descubrimiento permite filtrar los servicios u objectos que se buscan por ciertos criterios. De este modo se optimiza desde el propio protocolo el proceso de selección de un proveedor de servicios.

  En las búsquedas, lo más común es definir una serie de atributos de servicio (tipo, UUID, …) y lanzar las búsquedas por ellos. Utilizando otro tipo de mecanismos, por ejemplo LDAP, se pueden permitir búsquedas más complejas. Un nivel de sofisticación mayor, raramente contemplado en los protocolos actuales es el filtrado semántico de servicios.

  El mecanismo de búsqueda debe definir también todo el proceso de la misma: quién busca, qué se busca, quién filtra, cuántos responden….

- **Topología**

  Un servicio de directorio permite centralizar cierta información sobre los servicios y puede ayudar a mejorar el rendimiento en ciertas situaciones. Incluir un servicio de

directorio o utilizar un modelo más flexible como p2p dependerá de los objetivos del protocolo.

Normalmente se recomienda que el uso de servicios de directorio sea en entornos con tendencia estática, mientras que la versión p2p se ajusta a entornos muy dinámicos.

Otras alternativas que podrían considerarse son modelos mixtos que soporten entornos con y sin servicio de directorio.

- **Descripción de los servicios**

  A no ser que el mecanismo de descubrimiento sea muy simple, deberá haber un mecanismo que permita definir los atributos de un servicio. Estos atributos se podrán utilizar bien para seleccionar un servicio concreto, para obtener la forma de invocarlos, para realizar búsquedas y filtrarlos, etc.

  Las descripciones de los servicios podrían ir en pares atributo/valor, en XML, o incluso en forma semántica con OWL. Las descripciones de servicios pueden venir definidas en el propio protocolo (como usa SDP o UPnP), pero siguiendo una taxonomía flexible, que permita la introducción de nuevos dispositivos/servicios.

- **Seguridad**

  La seguridad depende de las funciones del protocolo (quién puede descubrir, quién puede anunciarse, quién puede invocar…). Se suelen diseñar mecanismos ad-hoc incrustados en el protocolo, o se puede aprovechar los mecanismos de capas inferiores si existen (HTTP, Bluetooth).

- **Alcance del descubrimiento**

  Esta característica tiene que ver con el espacio físico donde se es capaz de descubrir. El ámbito puede ser de corto alcance, a nivel de red local, red extensa, red con movilidad. En función del elegido habrá que tener en cuenta aspectos de direccionamiento y el protocolo de transporte.

- **Invocación de los servicios**

  Una vez que se ha descubierto un servicio, es necesario saber cómo invocarlo. Algunos protocolos obligan a un tipo concreto (UPnP es bajo HTTP/SOAP y Jini es mediante RMI). También se puede incluir en la descripción algún tipo de enlace al mecanismo de invocación (al estilo de WSDL). Otra opción es simplemente no contemplar la invocación y permitir que sea de cualquier tipo.

- **Tiempo de vida de los servicios**

  En un entorno dinámico de descubrimiento es muy importante que en todo momento se garantice que los servicios descubribles estén operativos. Esto se resuelve mediante mecanismos de tiempo de vida (leasing) o mediante comprobaciones de estado (heartbeat). La principal cuestión a considerar es sobre quién recae la responsabilidad de controlar los servicios y si dicho mecanismo formará parte del protocolo o no.

- **Infraestructura**

  Ciertos enfoques pueden obligar a que existan en el sistema otros mecanismos adicionales como pueden ser DHCP, DNS, Proxys….en cualquier caso habrá que considerar qué prerrequisitos tiene el mecanismo de descubrimiento para poder funcionar correctamente.

- **Gestión de energía**

  En un entorno con dispositivos de recursos limitados (sobre todo la batería) es recomendable que los protocolos con los que trabaje tengan esto en cuenta y estén optimizados para minimizar los tiempo de operación y el tráfico generado. La estrategia común es descomponer el protocolo en fases y que se realicen las de más peso únicamente si son verdaderamente necesarias.

3 MODELO DE DESCUBRIMIENTO

A partir de las características de los mecanismos de descubrimiento detectadas en el apartado anterior, para el proyecto Smartlab se va a diseñar un protocolo simple de descubrimiento, seleccionando las características que más interesen en función de los objetivos del proyecto.

3.1 Descripción general del descubrimiento

La arquitectura del proyecto Smartlab se basa en un servidor OSGi donde se alojarán los bundles necesarios para controlar y gestionar los diferentes dispositivos. El objetivo del mecanismo de descubrimiento es recuperar del entorno los bundles necesarios para controlar los dispositivos. A nivel general, cada dispositivo dispondrá de un bundle con el que poder controlarlo. El objetivo es descubrir este bundle de control, descargarlo al servidor central y arrancarlo para que los servicios que proporciona estén disponible al resto de la plataforma. Hay que tener en cuenta también que dichos dispositivos serán dinámicos, por lo que debe haber algún mecanismo de control de ciclo de vida de los mismos.

3.2 Requisitos de descubrimiento

A partir de las características definidas en el apartado de estado del arte, a continuación resumimos los requisitos que debe cumplir el protocolo de descubrimiento para el proyecto Smartlab. (Ver página siguiente)

Criterio	*Descripción*
Fases del Protocolo	De las fases principales se contemplarán: *Anuncio*: el cliente se anunciará cuando esté disponible. *Descubrimiento*: el servidor lanzará peticiones de búsqueda en intervalos de tiempo configurables. *Invocación*: la invocación de los diferentes servicios estará delegada en el Bundle que se instale en la plataforma. De esta manera el mecanismo de invocación puede adaptarse al dispositivo a controlar.
Transporte	Los mensajes del protocolo irán sobre Multicast y Unicast IP. Se valorará más adelante la inclusión de otros protocolos de transporte como Bluetooth o Zigbee.
Búsquedas	Por la naturaleza del descubrimiento, en la que el servidor debe obtener todos los recursos disponibles del entorno, no se incluirán búsquedas semánticas en el propio protocolo.
Topología	Aunque a efectos prácticos se configure la plataforma como un único servidor central, el protocolo será totalmente distribuido.
Descripción de los servicios.	Los servicios que se proporcionen en cada bundle deben estar descritos de alguna manera para poder después razonar sobre ellos. Esta descripción deberá usar un lenguaje estándar basado en XML.
Seguridad	Es necesario definir un mecanismo de acceso seguro y autenticado desde fuera hacia el servidor OSGi.
Alcance del Descubrimiento	El ámbito necesario del descubrimiento será únicamente local.
Tiempo de vida de los servicios	El entorno será dinámico, por lo que hace falta un mecanismo para asegurar que los servicios siguen vivos y disponibles. Dicho mecanismo puede darse a dos niveles, uno a nivel del dispositivo que anunció el bundle y otro a nivel interno del bundle, que debe controlar la disponibilidad de los servicios que registra en la plataforma.
Infraestructura	La red sobre la que irán los controladores de los dispositivos será IP, por lo que necesitarán de una infraestructura de asignación de direcciones. Además, el descubrimiento está orientado para entornos OSGi, por lo que los diferentes mecanismos se basarán en su filosofía de componentes.

4 DISEÑO GENERAL DEL MECANISMO DE DESCUBRIMIENTO

A la hora de diseñar el mecanismos de descubrimiento tenemos que tener en cuenta que la plataforma OSGi está orientada a componentes y servicios. Analizando el proceso de descubrimiento deseado y descomponiendo las actividades básicas, podemos concluir que hay cuatro bloques operacionales:

- ***Descubrimiento***: es el mecanismo de descubrimiento en sí. Su responsabilidad principal es la implementación del protocolo de descubrimiento a nivel de los mensajes de red necesarios (búsquedas, anuncios, leasing…). Este bloque obtendrá la información de los posibles bundles a ser instalados en el servidor.

- ***Instalación***: este bloque será el encargado de decidir si un bundle, llamémosle *candidato* debe ser instalado en el sistema o no. Para ello utilizará la información que le proporcione el mecanismo de descubrimiento, en forma de metadatos del propio bundle. Los bundles en OSGi pueden definir una serie de dependencias para su correcto funcionamiento. El mecanismo de instalación será responsable también de proporcionar cuando sea posible las correspondientes dependencias de cada bundle.

- ***Almacenamiento***: una vez descargado un bundle e instalado en el sistema, puede ser conveniente cachearlo para en un futuro acceder a él sin tener que descargarlo otra vez del dispositivo. Este mecanismo de almacenamiento actuará de caché de los bundles que se hayan descubierto para ahorrar después tráfico y tiempo si el dispositivo reaparece en el entorno.

- ***Configuración:*** los bundles son en realidad ficheros .jar precompilados, por lo que cualquier información dinámica de configuración deben ser capaces de obtenerla por otros medios. El mecanismo de configuración deberá proporcionar a los bundles un entorno dinámico donde poder recuperar información que necesiten preconfigurar de antemano.

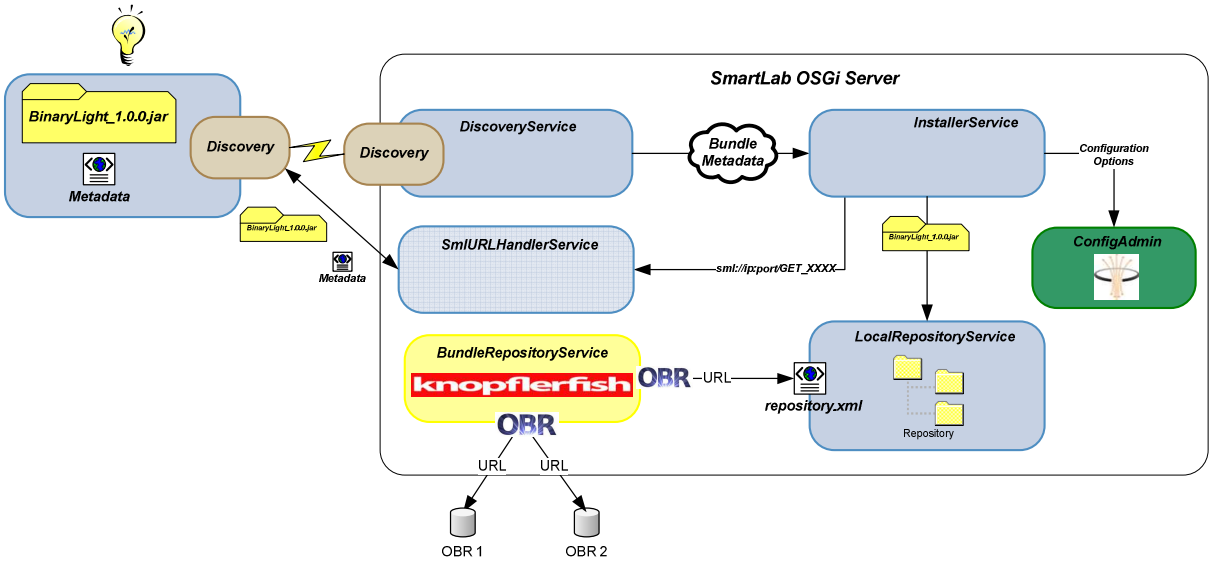

Figura 1 .- Bloques funcionales del protocolo de descubrimiento siguiendo el modelo de componentes de OSGi

A continuación se expondrá el diseño de cada uno de los bloques funcionales:

4.1 Descubrimiento

4.1.1 Protocolo

El mecanismo de descubrimiento para Smartlab se basará en un protocolo simple multicast que permitirá, como se ha comentado anteriormente, anunciar, descubrir, obtener meta información de los bundles o servicios y descargar el correspondiente bundle.

Los mensajes del protocolo de descubrimiento son los siguientes:

- ***SEARCH***

  El servidor envía periódicamente este mensajes MULTICAST UDP para sondear los bundles que se encuentran disponibles en el entorno.

- ***RESPONSE <BundleName> <Version>***

  Cada dispositivo que implementa este mecanismo de descubrimiento responderá al mensaje de SEARCH, enviando por UNICAST el nombre del Bundle y la versión correspondiente. Este par de valores son necesarios para identificar de forma

unívoca un bundle. En función de esta respuesta el servidor decidirá si solicitar más datos al cliente o no.

- ***ANNOUNCE <BundleName> <Version>***

  El dispositivo envía este mensaje MULTICAST UDP para anunciar su existencia. El servidor, al recibir este mensaje decidirá si solicitar más datos al cliente o no.

  Este mensaje se utiliza también para controlar el tiempo de vida de los dispositivos descubiertos. A modo de mecanismos heartbeat, cada vez que se recibe este mensaje en el servidor se resetea su tiempo de vida. Si pasa el tiempo establecido sin recibir este mensaje, se desinstala el bundle correspondiente.

- ***GET_METADATA***

  El servidor, una vez conocida la dirección del cliente podrá solicitar sus metadatos en cualquier momento. El dispositivo deberá contestar con un fichero XML con la definición del bundle y sus servicios.

- ***GET_JAR_FILE***

  Asimismo, una vez conocida la dirección del cliente, el servidor podrá solicitar la descarga del bundle para instalarlo en el sistema.

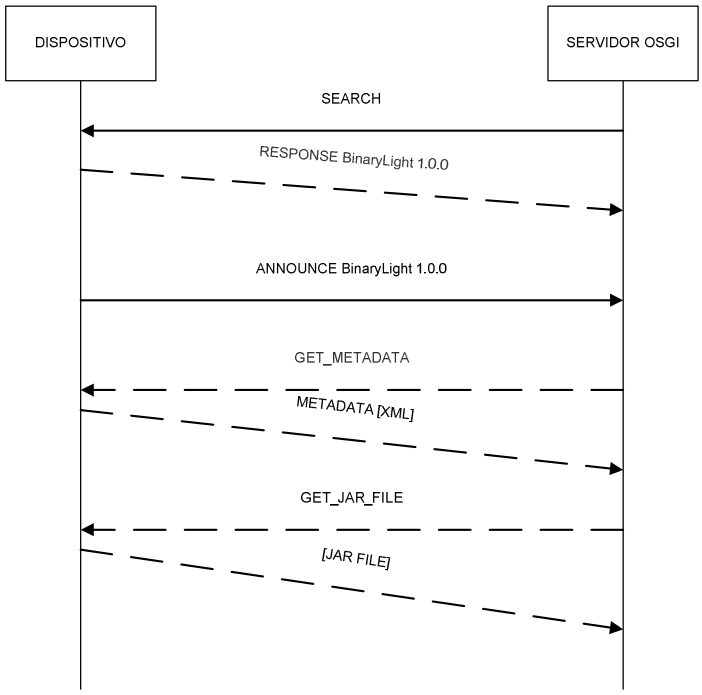

Figura 2 .- Mensajes del protocolo de descubrimiento

4.1.2 Tiempo de vida de los servicios

Los servicios que se instalan con la descarga de cada nuevo bundle descubierto tienen que tener algún mecanismo que asegure que están disponibles en un momento dado. Si bien parte de esa responsabilidad puede recaer en la implementación del propio bundle, obligándole a que compruebe con su correspondiente dispositivo que éste se encuentra operativo, dentro del propio mecanismo de descubrimiento también se puede incluir un sencillo sistema de leasing por el cual si no se recibe un mensaje de renovación cada cierto tiempo, se considera que el bundle que se instaló previamente ya no está operativo.

Una forma sencilla de añadir esta funcionalidad al protocolo que hemos diseñado es utilizar los mensajes de ANNOUNCE como renovaciones del tiempo de vida de un bundle. Los clientes que sigan el protocolo de Smartlab tendrán que anunciarse cada cierto tiempo a la red mediante este mensaje multicast. Cada vez que el servidor reciba un mensaje de ANNOUNCE de un bundle que estaba ya previamente instalado, se renueva su tiempo de vida. De esta manera se asegura que por lo menos la entidad o dispositivo que envió el bundle está activo. Esto es así ya que se considera que la entidad que proporciona el bundle normalmente estará ligada a los servicios que proporciona, por lo que si deja de responder significará que los servicios instalados ya no están disponibles.

4.1.3 Diseño en OSGi

El mecanismos de descubrimiento trasladado al modelo de componentes de OSGi se convierte en un Bundle con un único componente que implementa el protocolo anterior. Una vez que este componente obtiene los datos de un bundle candidato, se lo comunica al servicio de instalación para que lo procese.

Por lo tanto, la responsabilidad de este componente es:

- Implementar el protocolo de descubrimiento, recibiendo anuncios y lanzando búsquedas periódicas.
- Crear candidatos de bundles para pasárselos al servicio de instalación y que él decida si instalarlos o no.
- Gestionar el tiempo de vida de los dispositivos.

4.1.4 URL Service

Cuando el servicio de instalación reciba los datos de un bundle candidato, para poder decidir deberá ser capaz de obtener sus metadatos y también obtener el jar correspondiente. Estos dos procesos son dependientes del protocolo, por lo que el nivel de acoplamiento entre el servicio de instalación y el de descubrimiento sería demasiado alto.

Para desacoplar ambos procesos, se ha optado por definir un manejador de URLs para el protocolo utilizado en este proyecto (URLs que comienzan por sml://). De esta manera, el servicio de instalación podrá obtener tanto los metadatos como el fichero jar a través de una URL, sin conocer los mecanismos internos de su obtención.

De esta manera se consigue desacoplar ambos procesos y permitir la inclusión en un futuro de otros mecanismos de descubrimiento adicionales. También el propio uso del protocolo se vuelve más flexible, pudiendo cambiar el mecanismo de obtención de los metadatos o el fichero de forma independiente si así se desea.

4.2 Instalación y Almacenamiento

Uno de los problemas de la instalación en un entorno de componentes es la resolución de las dependencias de los mismos. Este proceso es de sobra conocido en los entornos OSGi y existe una alternativa para resolverlo de forma automática.

Aunque no está dentro de la especificación, las diferentes implementaciones de OSGi tienen mecanismos de gestión de bundles en forma de repositorios que llaman OBR (OSGi Bundle Repository). Un OBR no es más que un repositorio donde se colocan los diferentes bundles disponibles y después se crea un XML con la descripción de cada uno y sus dependencias. El servicio OBR debe ser capaz de gestionar diferentes URLs apuntando a dichos XMLs y ser capaz de resolver las dependencias entre los bundles que estén descritos en ellas. Es decir, si se solicita la instalación de un bundle, sus dependencias se podrán instalar automáticamente si se encuentran en alguno de los repositorios que gestione el OBR.

Siguiendo esta estructura es como se ha diseñado el mecanismos de almacenamiento, el cual proporciona un repositorio local donde se van almacenando los bundles descargados. Este repositorio es capaz de actualizarse dinámicamente, y generar su correspondiente XML describiendo sus bundles.

La funcionalidad básica del OBR ya viene implementada como servicio en la distribución de Knopflerfish, por lo que se ha adaptado para el mecanismo de instalación.Por lo tanto, el mecanismo de Instalación decide si instalar o no un determinado bundle y para ello colabora con el de Almacenamiento, que en realidad en un OBR local que se actualiza con cada nuevo bundle que recibe.

4.3 Configuración

Como los bundles se instalan automáticamente y su función principal, en la mayoría de los casos será gestionar un dispositivo o servicio remoto, casi con total seguridad serán necesarios ciertos parámetros dinámicos (ips, puertos,…) para los servicios que registre el bundle.

Con el objetivo de configurar los servicios, en los metadatos del bundle deberán estar definidos los parámetros correspondientes. Estos parámetros se cargarán en el Configuration Admin service de OSGi. Este servicio almacena configuraciones en base al PID del servicio o componente que se registra.

5 DISEÑO OSGI DEL MECANISMO DE DESCUBRIMIENTO

A continuación hacemos un breve resumen del diseño UML correspondiente a cada uno de los servicios que se han definido para el mecanismo de descubrimiento.

La arquitectura de descubrimiento se compone de 3 servicos OSGi principales (más sus clases relacionadas), cada uno con una responsabilidad bien definida: Descubrimiento (DiscoveryService),Instalación (InstallerService) y Almacenamiento(LocalRepositoryService). Para la función de configuración definida anteriormente se utilizará el servicio OSGi del ConfigAdmin.

5.1 DiscoveryService

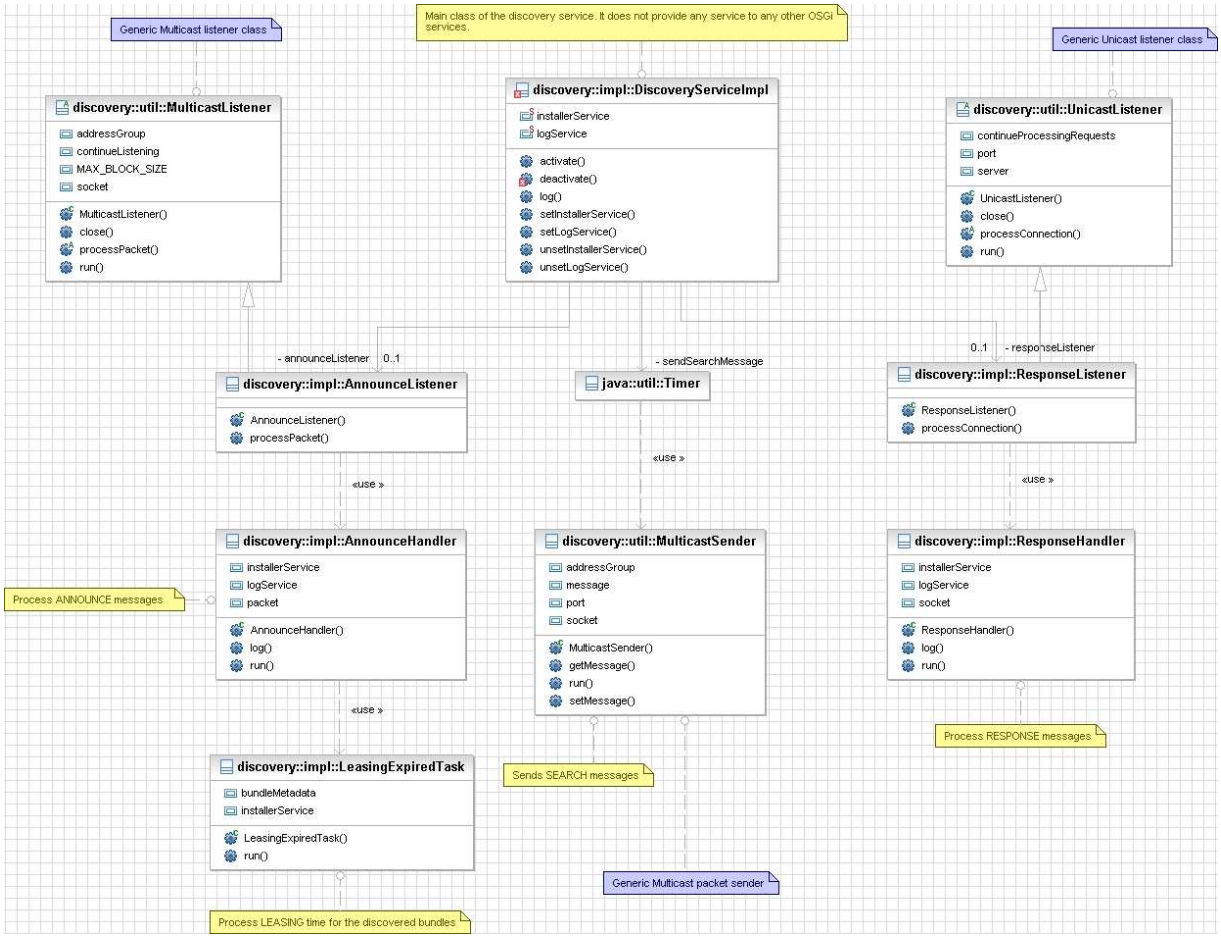

Figura 3.- Diagrama de clases del DiscoveryService

El servicio de Descubrimiento es el encargado de implementar el protocolo de descubrimiento de Bundles definido anteriormente. La implementación del protocolo está separada en un componente OSGi independiente para poder añadir otro mecanismo diferente en un futuro.

La implementación del intercambio de mensajes se ha llevado a cabo mediante unas clases genéricas que permiten enviar y recibir tanto mensajes Unicast como Multicast y definiendo los Handlers necesarios para los mensajes del protocolo:

- *AnnounceHandler*: implementa la recepción multicast de los anuncios de los dispositivos a descubrir.

- *MulticastSender*: clase genérica que envía mensajes multicast a un grupo. Se encapsula en un *Timer* para enviar periódicamente los mensajes de búsqueda.

- *ResponseHandler*: implementa la recepción unicast de las respuestas a los mensajes de búsqueda.

LEASING

El mecanismo de leasing está incluido en el propio procesamiento de los mensajes de ANNOUNCE. Existe un Timer por cada bundle descubierto que se va reseteando en función de los mensajes de announce recibidos. Si el Timer llega a expirar se ejecuta la tarea LeasingExpiredTask que lo que hace es eliminar el bundle del sistema (manteniendo el fichero descargado en el repositorio local).

URL Handler

El fichero del bundle y los metadatos de cada bundle se pueden descargar a través de dos comandos del protocolo que se envían por unicast al dispositivo remoto. Sin embargo, obligar a usar este protocolo resta flexibilidad al sistema, por lo que la obtención del fichero .jar y de la descripción XML del Bundle se lleva a cabo a través de URLs. De esta manera el sistema es mucho más flexible pudiendo incorporar nuevos protocolos en el futuro. Lo único que hay que hacer es registrar en OSGi un nuevo manejador de URLs para nuestro protocolo concreto. En este caso la URL se forma de la siguiente manera:

- sml://ipaddress:port/GET_JAR : para obtener el fichero .jar del cliente.

- sml://ipaddress:port/GET_METADATA : para obtener los metadatos del cliente.

El diseño en OSGi se muestra en la siguiente figura:

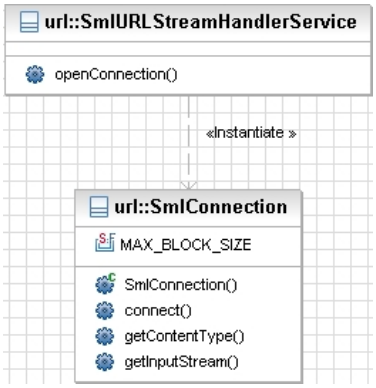

Figura 4 .- Diseño del manejador de URL para Smartlab.

Para que sea efectivo en OSGi, lo único que hay que hacer es registrar esta clase bajo el interfaz URLStreamHandlerService, lo que hará que automáticamente cualquier URL que empiece por sml:// será gestionada por estas clases.

DEPENDENCIAS

El servicio de descubrimiento necesita del InstallerService para proporcionarle los BundleMetadata y que él decida el proceso de instalación adecuado.

En resumen, el mecanismo de descubrimiento implementa el protocolo definido y genera objetos BundleMetadata con la información necesaria para que posteriormente el servicio de instalación decida y actúe en función de la información descubierta.

5.2 InstallerService

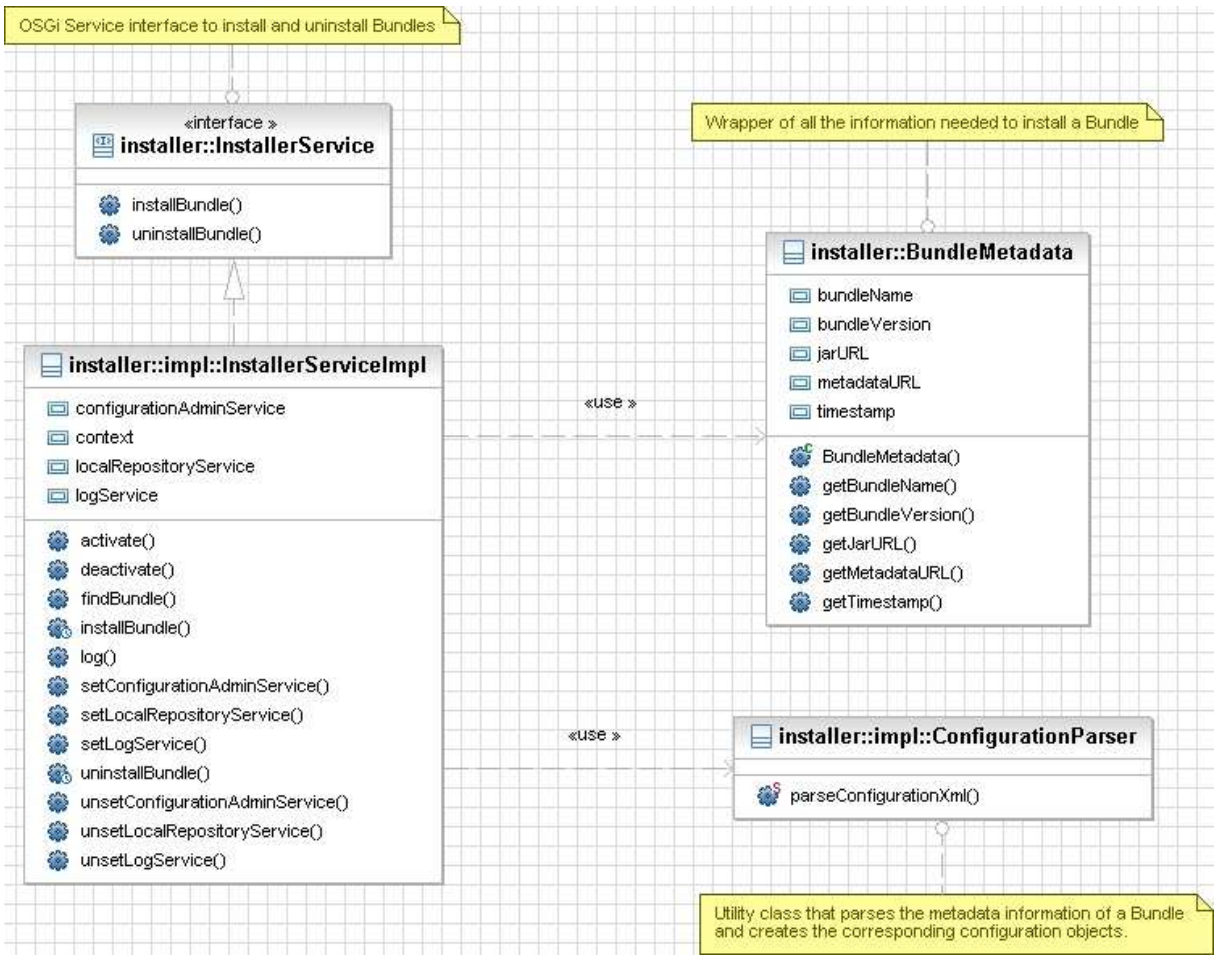

Figura 5 .- Diseño UML de InstallerService

Este servicio tiene la responsabilidad de installar Bundles en el sistema a partir de los datos necesarios contenidos en el objeto BundleMetadata. El proceso de instalación recibirá un objeto de este tipo, comprobará si está previamente instalado en el sistema y en caso contrario procederá a descargar el .jar a partir de la URL. También accederá a los metadatos del Bundle, que principalmente será su descripción y la información de configuración.

METADATOS

Esta información se encuentra descrita en un XML en la forma de pares atributo-valor, que el instalador parseará convenientemente. Los atributos de configuración serán por cada servicio que se instale en OSGi (un bundle puede instanciar varios servicios), por lo que el XML de configuración tiene que tener expresividad suficiente para indicar el nombre del

servicio y los atributos que le corresponden. De esta manera el servicio de instalación puede registrar de antemano la configuración a partir del nombre del servicio (aprovechando así el funcionamiento del ConfigAdmin de OSGi). Un ejemplo posible de XML que se puede utilizar es el siguiente:

```
<metadata name='smartlab.RemoteDisplay'>
      <property name='displayIPAddress' value='20.3.4.21'/>
      <property name='displayPort' value='5050' />
</metadata>
```

DEPENDENCIAS

Este servicio necesita del LocalRepositoryService para poder cachear en local los bundles que se vayan descargando y ahorrar así tiempo y recursos para cargas futuras.

5.3 LocalRepositoryService

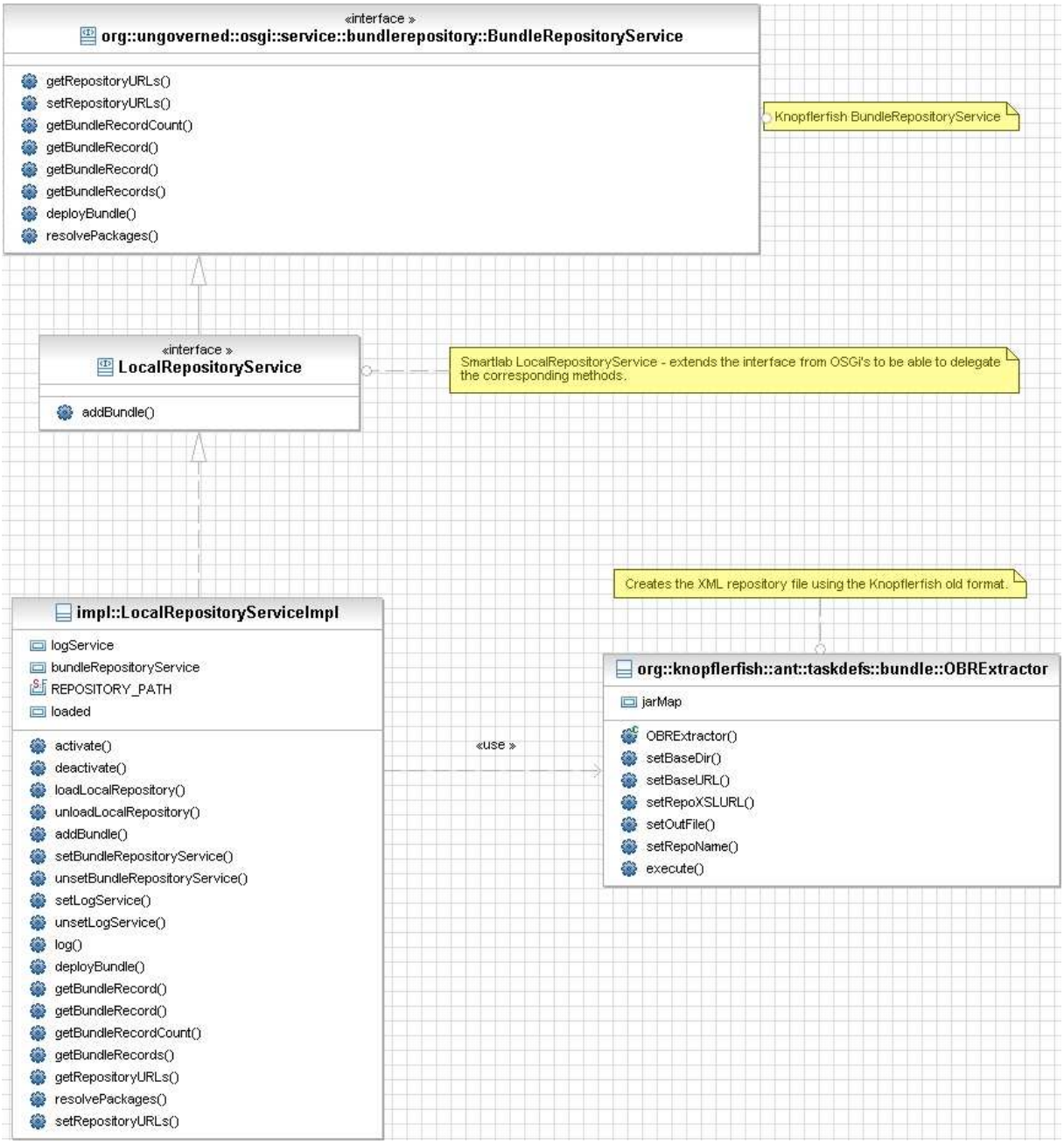

Figura 6 .- Diagrama UML del LocalRepositoryService

Este servicio es el encargado de la creación de un repositorio local donde cachear los diferentes bundles que se vayan descubriendo en el sistema. La interfaz del servicio se compone de un único método propio que permite añadir un nuevo bundle al repositorio local. Esta interfaz hereda de BundleRepositoryService (la interfaz del OBR), la cual incluye los métodos necesarios para añadir URLS de repositorios, buscar bundles e instalarlos resolviendo previamente las dependencias. En nuestro caso la implementación de los

métodos de esta interfaz se hará por delegación a la implementación real proporcionada en Knopflerfish.

Este repositorio local se encarga de recibir nuevos bundles, almacenarlos en disco duro y regenerar el fichero de descripción del repositorio. Para agilizar el proceso de generación dinámica de este fichero (repository.xml) se ha adaptado una tarea de ANT que realizaba dicha acción (incluída en la distribución de Knopflerfish) para poder utilizarla desde este servicio.

DEPENDENCIAS

Al extender la funcionalidad del BundleRepositoryService necesita de este servicio para la funcionalidad de gestión del repositorio y la instalación con control de dependencias.

6 ESCENARIO DE FUNCIONAMIENTO

A continuación se expone un posible escenario de funcionamiento de todo el proceso de descubrimiento y descarga de un bundle de un dispositivo que se encuentra en el entorno:

1. Un nuevo dispositivo aparece en el entorno. Dicho dispositivo cuenta con un módulo de descubrimiento, su bundle de control y una descripción en XML de los servicios o las funcionalidades que proporciona.

2. Cuando se activa el dispositivo, éste anuncia por multicast su nombre de bundle y versión (mecanismo para identificarlos unívocamente).

3. El servicio de descubrimiento captura este anuncio y crea un BundleMetadata con los datos del bundle candidato a ser descargado. Este "candidato" es enviado al InstallerService.

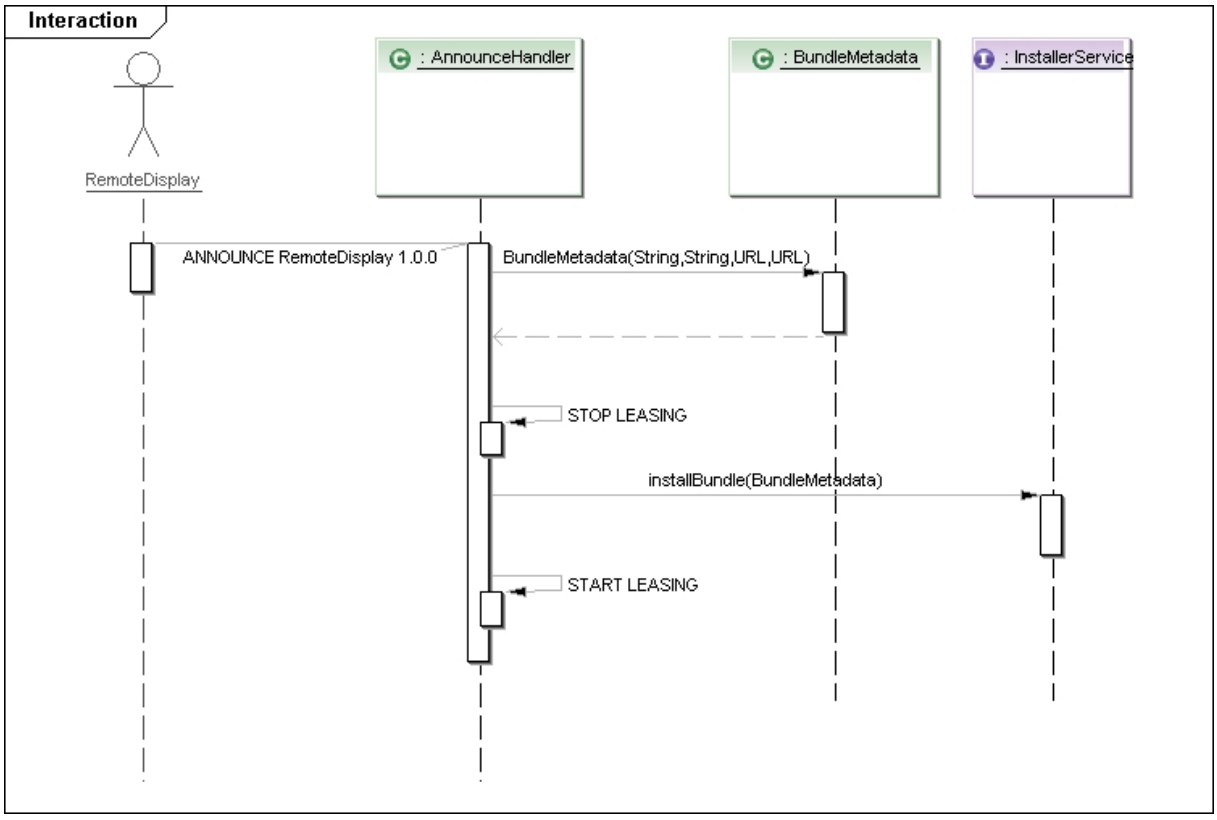

Figura 7 .- Descubrimiento de un Bundle Recibir mensaje ANNOUNCE (Pasos 1, 2 y 3)

4. El InstallerService debe encargarse de comprobar si hay que descargar el bundle o no y si hay que instalarlo. Para ello extrae de los metadatos el nombre y la versión, consultando en el LocalRepositoryService si ya existe un bundle con dicho nombre.

5. Si no existe, se procede a descargarlo del cliente. Para ello se obtiene directamente apartir de los metadatos la URL donde se aloja. En nuestro ejemplo, se ha implementado un manejador de la URL sml:// para que sea posible incorporar cualquier otro protocolo (http, ftp,…).

6. Una vez descargado el fichero (todavía en memoria), el InstallerService solicita al LocalServiceRepository que lo almacene en disco.

7. El LocalServiceRepository mantiene un OBR local con todos los bundles que se hayan descargado a partir del mecanismo de Smartlab. Cuando llega un nuevo jar, lo almacena en disco, y actualiza el repositorio en base al funcionamiento del OBR.

8. Actualizar el repositorio implica regenerar el fichero de descripción (repository.xml) y refrescar el servicio de OSGi BundleRepositoryService.

9. Una vez hecho esto se indica al InstallerService la ubicación del bundle recién almacenado, para que pueda instalarlo convenientemente.

10. El InstallerService decidirá si instalar el bundle solicitándolo al LocalRepositoryService, el cual delegará en el BundleRepositoryService la instalación real.

11. El BundleRepositoryService ya implementa las opciones de resolución de dependencias en base a las URLs que tenga registradas. También se encarga de instalar el bundle y activarlo convenientemente.

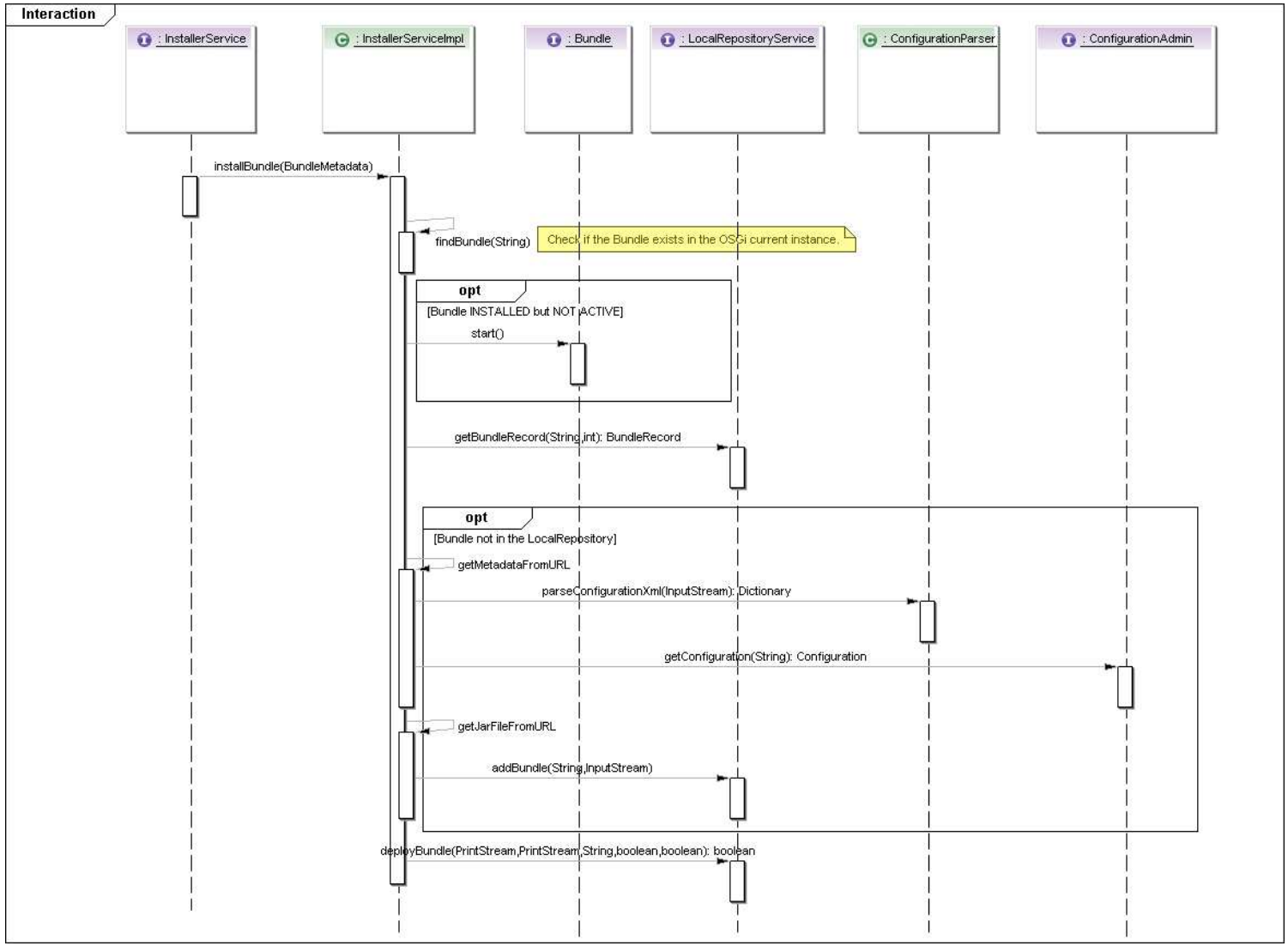

Figura 8 Descubrimiento de un Bundle (Pasos 4,5,6,8,9,10)

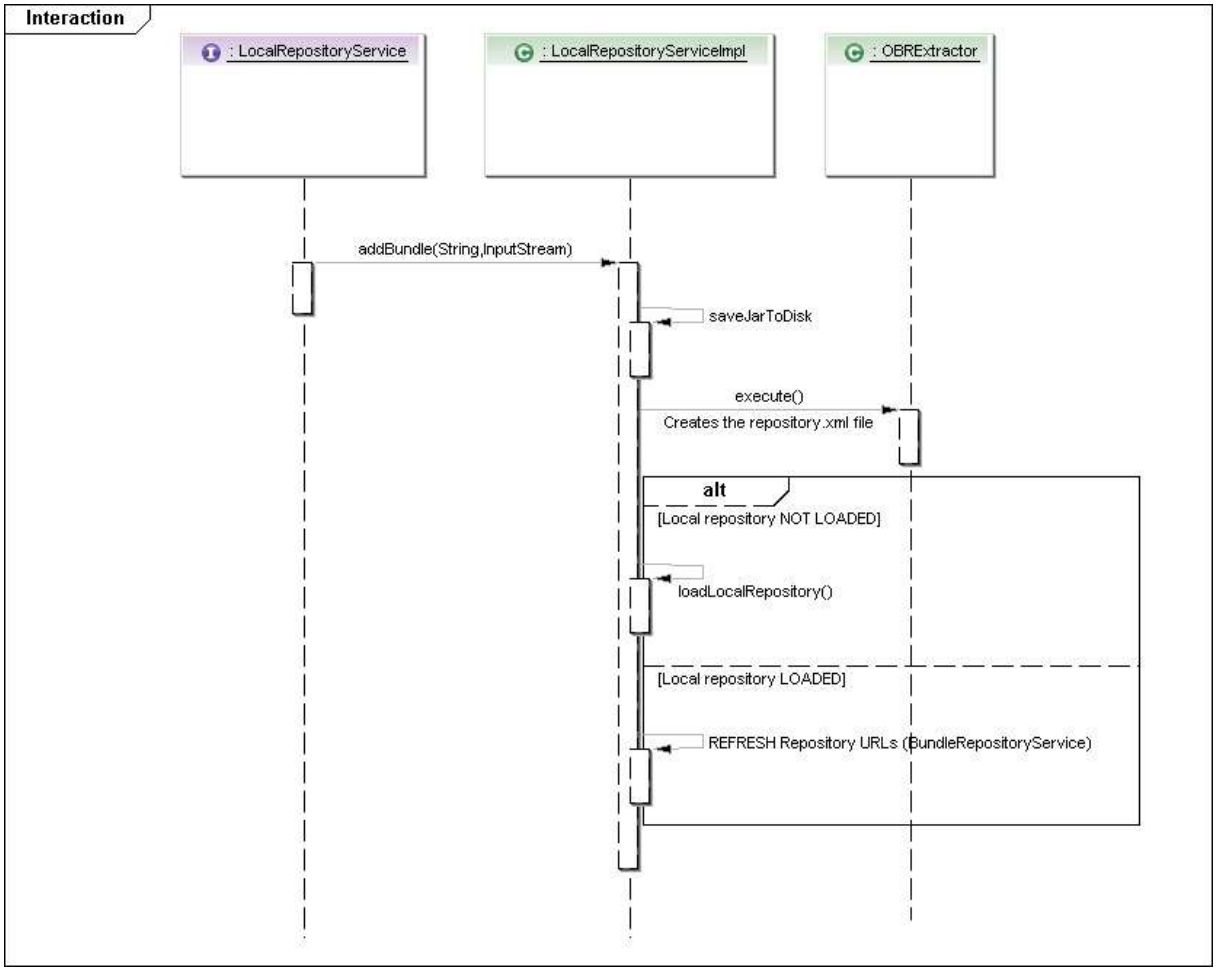

Figura 9.- Descubrimiento de un Bundle (Paso 7)

7 REFERENCIAS

[1] W. K. Edwards, "Discovery Systems in Ubiquitous Computing," *IEEE Pervasive Computing*, vol. 5, pp. 70-77, 2006.

[2] S. Y. Stephen and K. Fariaz, "An Energy-Efficient Object Discovery Protocol for Context-Sensitive Middleware for Ubiquitous Computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 14, pp. 1074-1085, 2003.