Journal of Virtual Reality and Broadcasting, Volume 4(2007), no. 16
JVRB EuroITV 2006 Special Issue, no. 6

Exploiting OSGi capabilities from MHP applications*

Rebeca P. Díaz Redondo, Ana Fernández Vilas, Manuel Ramos Cabrer, and José J. Pazos Arias

Dept. of Telematics Engineering,
University of Vigo,
36200 Vigo, Spain
email: {rebeca, avilas, mramos, jose}@det.uvigo.es

Abstract

In this paper we introduce a cooperative environment between the Interactive Digital TV (IDTV) and home networking with the aim of allowing the interaction between interactive TV applications and the controllers of the in-home appliances in a natural way. More specifically, our proposal consists of merging MHP (Multimedia Home Platform), one of the main standard frameworks for IDTV, with OSGi (Open Service Gateway Initiative), the most widely used open platform to set up Residential Gateways. To overcome the radically different nature of these specifications the function-oriented MHP middleware and the service-oriented OSGi framework , we define a new kind of application, coined as XbundLET. Although this software bridge is suitable to enable the interaction between MHP and OSGi applications in both directions, we concretely focus on exposing our implementation experience in only one direction: from MHP to the OSGi world.

*Partly supported by the R+D project TSI 2004-03677 (Spanish Ministry of Education and Science) and by the EUREKA ITEA Project PASSEPARTOUT.

First presented at the 4th European Interactive TV Conference EuroITV 2006, extended and revised for JVRB

Keywords: MHP, OSGi, gateways

1 Introduction

Nowadays, we are living important technological changes that mainly affect our life at home and the communication with the outside world. Residential Gateways (RGs) play an essential role in this field because they constitute a bridge between the networked home and the outside world turning out to be indispensable in the smart home. However, and although nowadays there is not a wide consensus about their configurations, the functionality of new RGs is wanted to be extended to support digital, multimedia and entertainment features. On this matter, it is believed [HBJK04] the current Set-Top Boxes (STBs) have a strong potential to evolve into popular residential gateways. In fact, the great evolution of Interactive Digital TV (IDTV) has entailed the diversification of TV receivers. Consequently, new STBs are not only a decoder for digital television broadcast, but also an entry point to the Information Society and a suitable platform to support the execution of interactive applications.

The proposal we introduce tries to go one step further and allow the interaction between TV applications and the controllers of the in-home appliances. With the aim of assuring the interoperability, we have adopted widespread standard solutions for each one: the MHP (Multimedia Home Platform) specification [MHP03] for the STBs; and the OSGi (Open Service Gateway Initiative) Service Platform specification [All03] for home networking. However, the convergence is not as simple as it could seem. First, the current MHP specification implies a dead end at home, because of the absence of connection to the in-home network. Besides, each platform has a radically different na-

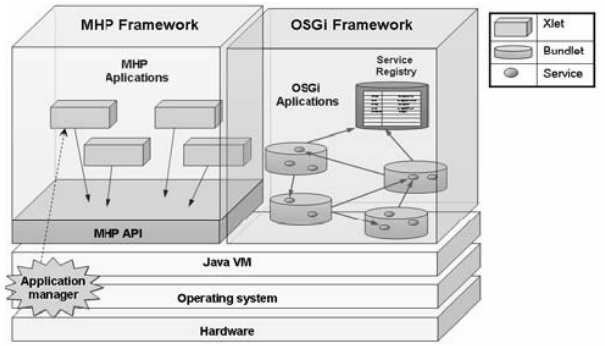

Figure 1: MHP vs. OSGi

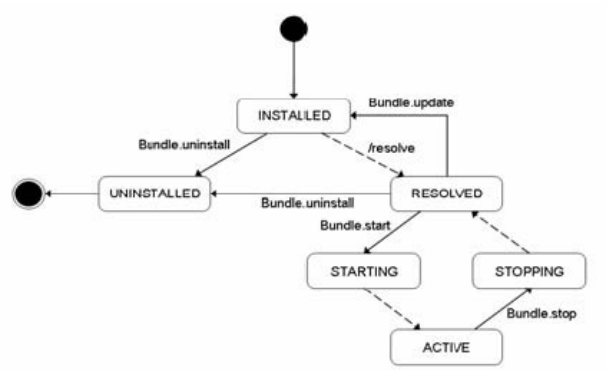

Figure 2: Lifecycle management in OSGi. Dashed lines represent automatic or pseudo-transitions

ture: the function-oriented MHP middleware and the service-oriented OSGi framework. With the aim of overcoming these problems, we have defined a new kind of applications, called XbundLET, that allows the interaction between MHP and OSGi applications in a natural way.

Our proposal tries to provide the MHP-OSGi communication in both directions: (i) enabling an MHP application to use OSGi services; and (ii) that any OSGi device is able to take advantage of the MHP functionality. However, in this paper we specially pay attention to one of them: from MHP to OSGi, describing our implementation experience in detail by using a use scenario.

In order to introduce our proposal, the next section describes both the OSGi and MHP specifications and in Section 3 we expose a scenario at home where one MHP application needs to interact with the in-home devices. In order to support this communication, we introduce in Section 4 the new software component (XbundLET), focusing on the interaction from MHP to OSGi. In sections 5, 6 and 7 our implementation experience is detailed and, finally, related work, conclusions and future work are presented in sections 8 and 9.

2 MHP vs. OSGi

OSGi and MHP share a similar high-level goal: an open standard for a horizontal market. However, the scenarios where both platforms are located IDTV and in-home appliances are very different, so the kind of architectural solution is different too (Figure 1).

OSGi is essentially a service-oriented architecture for residential gateways. In this framework, any running component (known as *bundle*) exports the services it provides and it also can discover other services by means of a third role, the *OSGi Service Registry*. Being more precise, a bundle is a Java ARchive (JAR) representing the minimal component in OSGi that can be installed, uninstalled or updated. On another hand, the minimal unit of functionality is really what OSGi calls a service.

The best way to describe the bundle dynamics is by inspecting its lifecycle diagram, consisting of the six states in Figure 2 In brief, once a bundle is `INSTALLED`, it can be started, but it must be `RESOLVED` (the framework resolves all code dependences) before getting control. After that, in its real lifetime, the bundle can be started (`ACTIVE`) and stopped (`RESOLVED`) by using the intermediate states `STARTING` the bundle is being started and `STOPPING` the bundle is being stopped which highlight the fact that both processes are not instantaneous. The moments when a bundle becomes `NSTALLED` and `UNINSTALLED` are the end points in its lifecycle, which implies the periodical change between `RESOLVED` and `ACTIVE` states.

The MHP specification is a multiple layer model which mainly defines the boundary layer MHP API between the system software and the MHP applications (Figure 1). MHP applications (commonly referred as *Xlets*) are usually sets of Java class files written by using a well-defined set of libraries in the MHP API[1]. The other defining characteristic of Xlets is that they are totally controlled by the Application Manager residing on the STB which is responsible for monitoring,

[1]Other kind of MHP applications follows the DVB-HTML model, but because of its complexity and computational costs, DVB-J applications (Xlets) are by far the most popular.

starting and/or stopping them.

In fact, the lifecycle of an Xlet is as follows. When the initial Java class of an Xlet is loaded and instantiated (either from the transport stream or locally from the STB), it enters the LOADED state (see Figure 3). Once the Xlet is LOADED, it can be terminated at any time (changing into the DESTROYED state). Most frequently, after LOADED, the Application Manager signals the Xlet to initialize itself and the Xlet enters the PAUSED state, where it is ready to become ACTIVE. This state indicates the application is executing and can be changed to PAUSED to be resumed again.

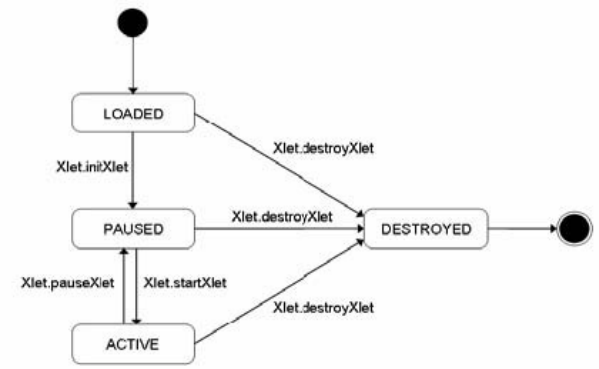

Figure 3: Lifecycle management in MHP.

To sum up, both OSGi and MHP specifications define an implemented functionality as a Java application with a restricted lifecycle externally manageable. For the former, the objective is making an extremely dynamic environment manageable; for the latter, lifecycle management makes a resource-limited environment more reliable and predictable. The communication schemas are also different. An OSGi application (bundle) can provide services to other bundles and/or invoke other bundle's services by using the OSGi Service Registry. On the other hand, an MHP application (Xlet) can communicate with other Xlets via the IXC (*Inter-Xlet Communication*), a special form of RMI (*Remote Method Invocation*) defined in [MHP03].

3 Why MHP to OSGi communication is necessary

As we have previously mentioned, one of the advantages of merging MHP and OSGi platforms is the possibility that IDTV applications and home devices can interact. In fact, the following scenario shows how an MHP application needs to communicate to the OSGi framework to use several services it provides:

Scenario: In John's Set-Top Box, a recommender of TV contents is in charge of recording the TV shows in which John may be interested. Since he is keen on cooking, a documentary about Ferrán Adriá is recorded. When John decides to see it, he is notified that the recipe the cook is explaining was also broadcast and recorded as an Xlet with different options. As he prefers having a hardcopy, he selects the option "print the recipe". Besides, there is also the option of sending the recipe to the fridge, which checks if all the ingredients are available; if anythin is missing it is added to the shopping list.

As Figure 4 illustrates, the infrastructure at home consists of the in-home network, the Internet access and the STB-Residential Gateway, where a home controller is placed. The Xlet (the recipe) has been broadcast and stored in the PVR (*Personal Video Recorder*) by the recommender (1). As soon as John gets into home (2) the atmosphere is adapted to his tastes (3) and the TV cooking program previously recorded is offered to him (4-5). The recipe is transferred from the STB to the fridge and /or to the printer (6). Finally, at the end of the day the fridge sends the shopping list to the supermarket using the Internet connection (7). The interaction among the domestic appliances is provided by the OSGi framework in the usual way. However, since the Xlet carrying the recipe needs to interact with the fridge and the printer, the interaction between MHP and OSGi is needed.

The interaction among the domestic appliances is provided by the OSGi framework in the usual way. However, since the Xlet carrying the recipe needs to interact with the fridge and the printer, the interaction between MHP and OSGi is needed.

4 XbundLET: a conceptual bridge

Our proposal to integrate MHP and OSGi platforms into a smart solution for home networking consists of introducing a conceptual entity of communication, which we have named XbundLET (Figure 5). An XbundLET is a hybrid application that integrates the defining characteristic of both Xlets and bundles: (i) as any Xlet, its lifecycle is controlled by the Application Manager and it also can communicate with other Xlets via the IXC ; (ii) as a bundle, it can provide services to other bundles and/or invoke other bundle's services.

As having both an Xlet interface and a bundle inter-

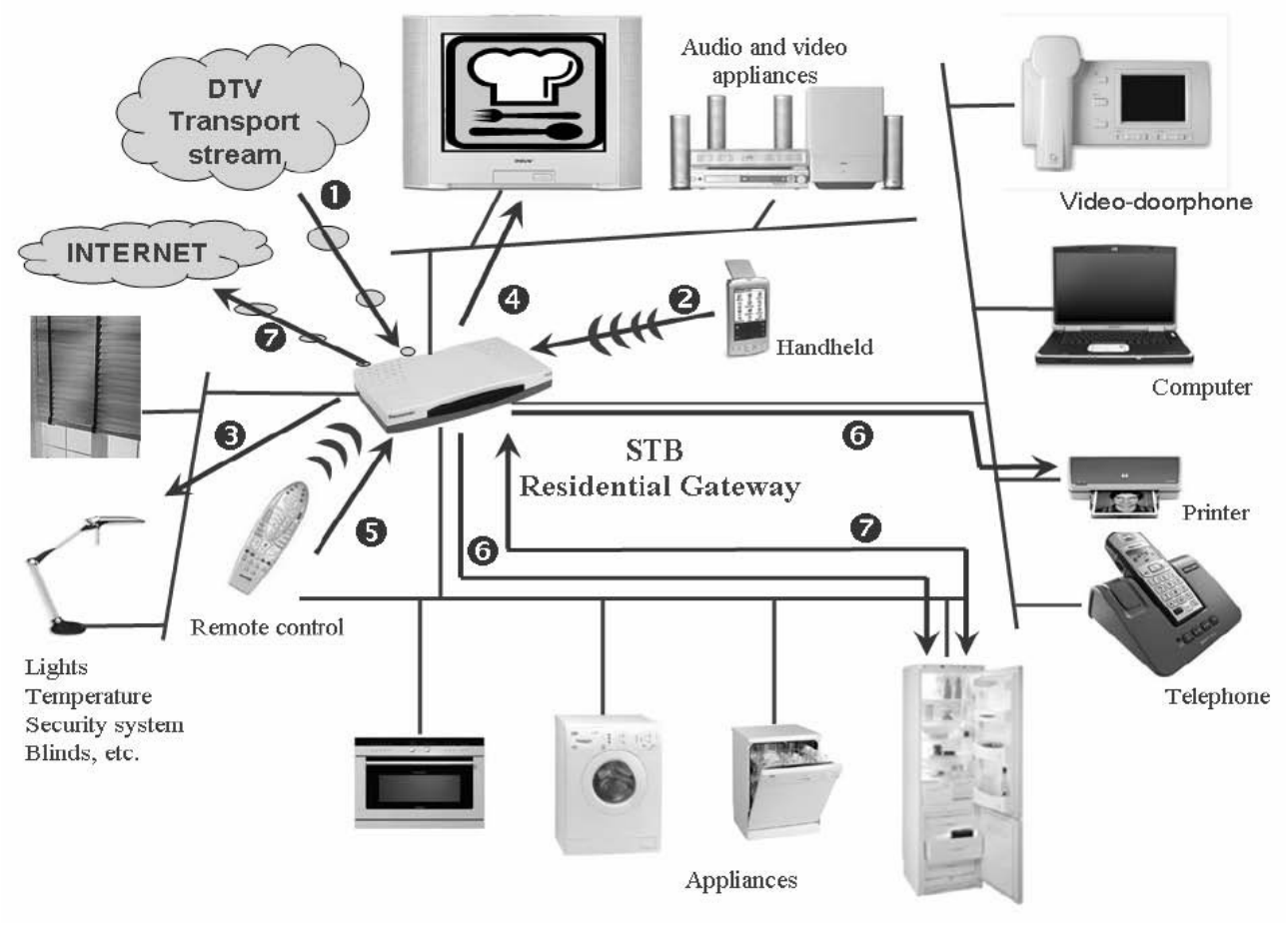

Figure 4: An example scenario: from the MHP to the OSGi framework

face, the dynamic behaviour of the XbundLET entity is as shown in Figure 5. In this virtual lifecycle, we have defined four main states:

- `INITIATED`. In this state, the XbundLET is not yet ready to run as either an Xlet or a bundle, though it is being prepared for that. It has to wait for resolving all code dependences (the pseudo-transition `resolve`) in the OSGi framework and for being signalled by the MHP Application Manager (`Xlet.initXlet`).
- `INACTIVE`. In this state, the XbundLET is ready to run as both an Xlet and a bundle. As soon as an activation order is received it enters the `active` state. Whatever the activation order received (`Xlet.startXlet` or `bundle.start`), the other one is triggered internally.
- `ACTIVE`. In this state, the XbundLET is active, i.e., the application is executing and so it exists both in the OSGi framework and in the MHP platform. Then, the application lives in a cycle of `inactive` and `active` states. Similarly to the activation order, whatever deactivation order received (`Xlet.pauseXlet` or `bundle.stop`), the other one is triggered internally in order to pause the XbundLET in both execution environments. The `inactive-active` cycle continues until the XbundLET is finished and it enters the `finished` state.
- `FINISHED`. In this state, the XbundLET is being destroyed. It has received the finish order, `Xlet.destroyXlet` or `bundle.uninstall`, so it frees its resources in the MHP platform and notifies bundles in the OSGi framework that the XbundLET is being uninstalled. As Fig. 2 shows, a bundle can only be uninstalled if it is not active. So, when the finish order is received via `Xlet.destroyXlet`, a set of automatic transitions and intermediate

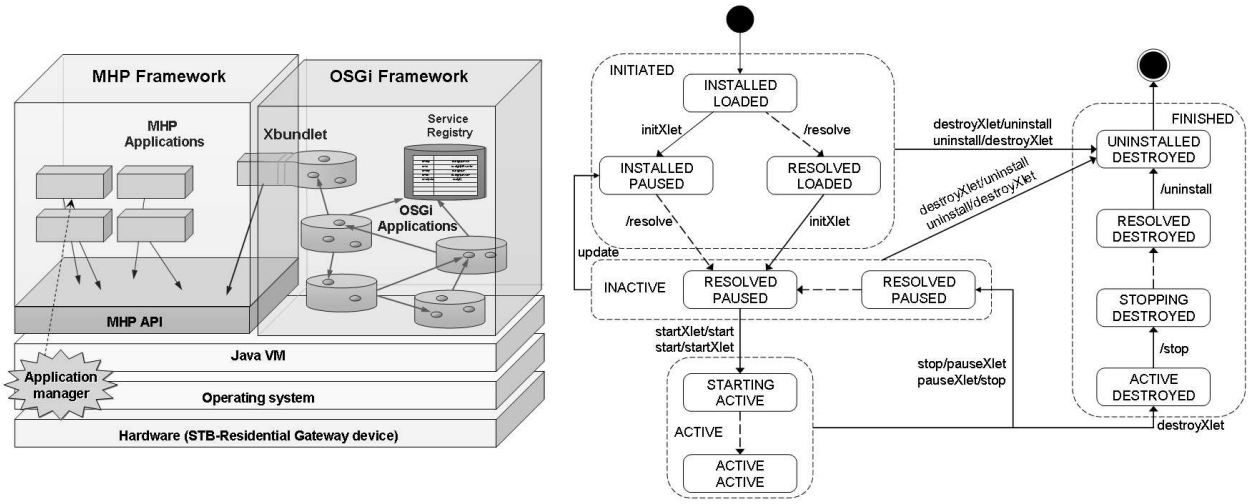

Figure 5: The XbundLET entity and its virtual lifecycle

states are included in the virtual lifecycle.

This virtual lifecycle enable the interaction between MHP and OSGi in both directions without losing the adequate interface for each one. Therefore, any change in the OSGi lifecycle corresponds with an appropriate reaction in the MHP lifecycle and vice versa, maintaining the coherence among both frames. Besides, we specify the behaviour of the XbundLET and how it must communicate with the MHP and OSGi actors (the Application Manager, the OSGi Service Registry, etc.) in order to really provide the needed mechanisms for a real communication between both platforms.

4.1 From MHP to OSGi

Although XbundLETs play their main role whenever inter-platform communication is needed, in this paper we only focus on communications initiated from the MHP side, i.e, one Xlet needs to use one or more services provided by the in-home OSGi appliances. The process is schematised in Fig. 6, where the communication before the Xlet arrives is represented using dashed lines:

1. The bundles controlling the devices at home firstly register the services they offer in the Service Registry of the OSGi platform.

In the following two communications, a new entity called Master XbundLET plays the essential role of linking both OSGi and MHP worlds. Mainly, the Master XbundLET is occupied in paying attention to any new service coming from the OSGi world to inform the MHP environment about it:

2. The Service Registry notifies the presence of a new service, via Service Listener, to the Master XbundLET, assuring it is always up-to-date with all novelties coming from the OSGi platform.

3. Whenever the Master XbundLET is informed about any change, it simply notifies the news to the IXC Registry. Therefore, any Xlet of the MHP framework can look up the IXC Registry to find specific and updated OSGi services.

By the previous exchanges of information, the OSGi environment is able to advertise its services in the MHP world. But, how can an Xlet use these features? In the same figure (Fig. 6), the communications involved in this process are represented by continuous lines:

1. Firstly, the Xlet can find in the IXC Registry the services it is looking for.

2. After that, it asks to the Master XbundLET to be put in touch with the appropriate XbundLET.

3. The Master XbundLET creates and launches one MHP2OSGi XbundLET to provide access from the MHP side to the OSGi service. Doing so, the Master XbundLET avoids (i) the overload of being in charge of every single data interchange and (ii) the overload of having one XbundLET

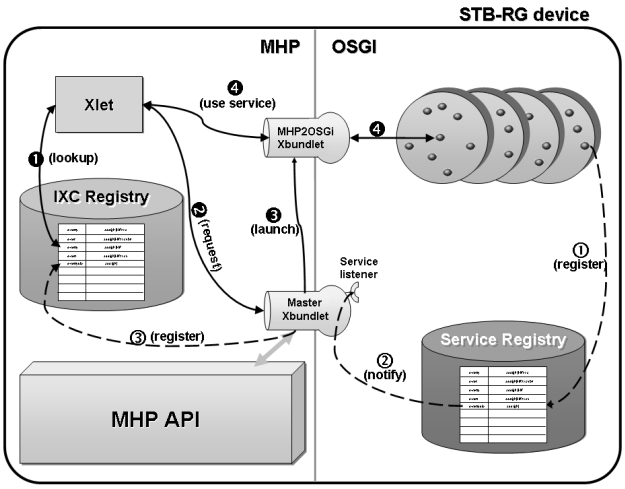

Figure 6: Accessing OSGi servicer from Xlets.

for each even when most of them will probably never be requested by any Xlet.

4. Once the MHP2OSGi XbundLET has been launched, it will remain available not only for the requesting Xlet, but also for potential future Xlets[2]. This MHP2OSGi XbundLET will be also responsible for managing all requests from the MHP side to all services the bundle provides through the same service interface, i.e., there exists at most one MHP2OSGi XbundLET for each corresponding service interface in the bundle.

5 Implementation Overview

For the implementation of the proposal we have started from open software implementations of both the OSGi framework and the MHP platform. On the MHP side, after studying several open source possibilities (*XleTView* [Xle04], *openmhp* [Ope03] and *mhp4free* [Tec04]) we decided to use *XleTView* because of its suitable current status of implementation. However, *XleTView* does not include a complete implementation of IXC communication yet, so we have done several works in this line. On the OSGi side, after studying different possibilities, we have opted to use the OSCAR (*Open Source Container ARchitecture*) implementation [OSC05], because others, like the *Knopflerfish OSGi* [OSG04] and *JEFFREE* (Java Embedded Framework FREE) [JEF03], offer less functionality.

Both environments run on the same physical device (PC platform) and, on boot, the MHP Application Manager is responsible for initialisation of the STB-Residential Gateway. To do this, our Application Manager launches the OSCAR OSGi framework by executing `oscar.jar`. As it is normative in the OSGi specification, all bundles (XbundLET too) run in a single Java Virtual Machine (JVM), the one on which the OSCAR execution environment runs. On the contrary, accordingly with the MHP specification, Xlets may run on this JVM or not. This does not entail a problem, because the communication between Xlets (and, by extension, the Xlet and XbundLET communication) is always assured by the IXC.

The list of bundles to automatically install and start when OSCAR is started is specified in `oscar.auto.start`; the bundle Master XbundLET belongs to this list. Of course, its first task is creating a Service Listener, ready to detect any change in the OSGi services to inform properly about them to the MHP side. On the OSGi side, when bundles controlling the printer and fridge are started, they register the services they provide in the OSGi Service Registry, like `Printing` and `Supplying` (from the printer and fridge, respectively). Consequently, the Service Listener of the Master XbundLET detects this situation and informs the IXC Registry about that.

According to the tasks mentioned in the above paragraphs, we have modified the source code of the Application Manager (offered by the *XleTView* implementation [Xle04]), and we have also added inter-Xlet communication. Of course, adding new functionalities to the Application Manager is allowed by the MHP specification, so, in any case, we have respected the standard.

Regarding the XbundLETs design and implementation, remark that since every XbundLET implements Xlet and bundle interfaces, it should accept requests addressed to both of them, and guarantee the integrity of both lifecycles. Finally, because we have defined a dual interface, an XbundLET can not only be installed from the OSGi framework, but also an XbundLET arriving from the IDTV stream can be launched by the Application Manager. So, any IDTV provider can play also the role of a OSGi service provider only by following the schema proposed in this paper.

[2] If the Application Manager needs to free occupied resources, it may decide to destroy the XbundLET; in this situation, the Master XbundLET is in charge of launching it again on demand.

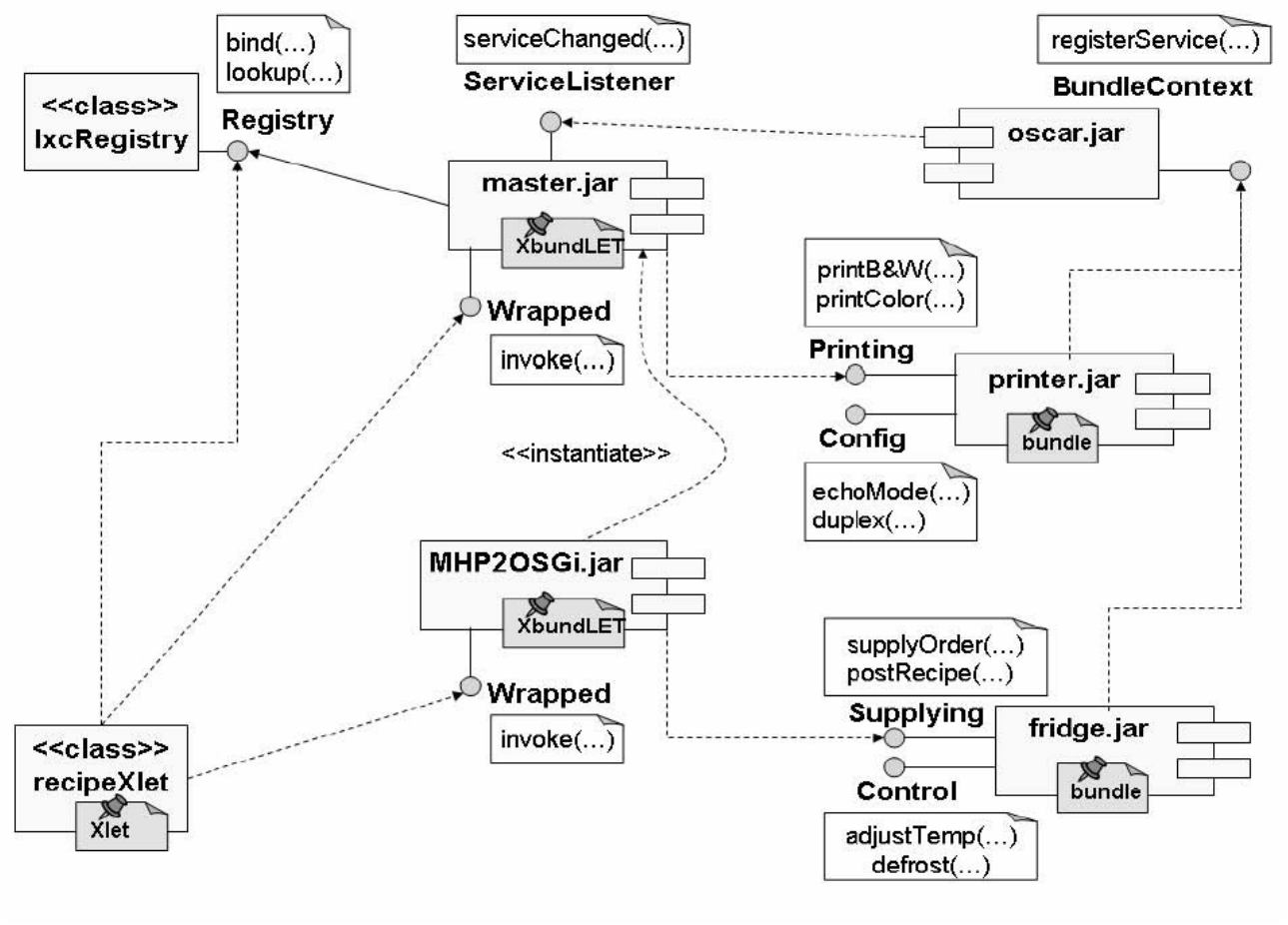

Figure 7: Main classes and components in the scenario

6 The components in action

The main classes and components which are involved in invoking OSGi services from the MHP platform are shown in the class diagram of Fig. 7 which is focused on the usage dependencies among elements.

On the OSGi framework, apart from the framework itself (component `oscar.jar`), the main components in the scenario are the bundles of the printer (`printer.jar`) and the fridge (`fridge.jar`); both of them are supposed to be installed and started (for instance, automatically on Oscar boot) when the scenario begins. The printer and the fridge use the interface `BundleContext` for registering the service interfaces they match, `Printing/Config` and `Supplying/Control` respectively. To be precise, in our scenario, the Xlet with the recipe add-ons only needs to use the following methods in these interfaces:

```
printer.jar
public interface Printing
{
void printB&W(File myfile);
...
}

fridge.jar
public interface Supplying
{
void postRecipe(File myrecipe);
...
}
```

On the MHP platform, the main elements are the Xlet with the recipe add-ons, client class `recipeXlet`, and a broker element for communication, the class `IxcRegistry`. This class enables inter-Xlet communication by providing the `Registry` interface (package `javax.microedition.xlet.ixc`) for binding (`bind, rebind, unbind`) and looking up (`lookup, list`) remote object references bound with string names in the registry. The recipe's Xlet will use the above interface for locating a printer or a fridge to which to send the recipe (for printing or updating supplying, respectively).

The components in the middle of the diagram, `master.jar` and `MHP2OSGi.jar`, are the ones which set up the bridging structure, and so, they are what we have called XbundLETs. The XbundLET `master.jar` implements the `ServiceListener` interface in order to track the availability of new service objects in the framework and, consequently, to bind or unbind remote objects in the `IxcRegistry` when OSGi services are registered or unregistered. Apart from the dependencies which result of using interfaces, the figure shows that the component `master.jar` depends on the component `MHP2OSGi.jar`. This dependency is stereotyped as `instantiate` to indicate that operations on the source element create instances of the target element. That is, the bundle `master.jar` create and install XbundLETs `MHP2OSGi.jar` on the framework.

According to the conceptual model in Sec. 4, the components which really act as a bridge between MHP and OSGi are instances of `MHP2OSGi.jar`. Every MHP2OSGi XbundLET component is linked to a service interface *SI* which has been registered in the OSGi framework by a bundle *B*. To be precise, given a *SI* implemented by *B*, it will be an `MHP2OSGi.jar` instance which will offer a Wrapped interface to the pair (*B,SI*). The main method in that Wrapped interface is

```
Object invoke(methodref meth,object[] args)
```

which calls a method meth in the OSGi service object, *SO*, corresponding to the pair (*B,SI*). That is, the *SO* the bundle *B* has registered to provide the *SI*. The method `meth` is managed dynamically by means of a method reference and all the input and return parameters are of type Java `object`, so they can be cast to any desired type. The implementation of this `Wrapped` interface is dynamically constructed at installation-time, when the MHP2OSGi XbundLET is created and initialised by the Master XbundLET. The wrapped code acts as an interface between the caller Xlet and the code which, in the bundle *B*, implements *SI*. To do this, the `Wrapped` interface also offers public methods to configure itself, that is methods which allow the Master XbundLET to customize the bridge; linking it to a concrete bundle (for instance `printer.jar`) and to a concrete service interface, for instance `Printing`.

7 Details of the interaction with the printer

In the scenario proposed in Sec. 3, the recipe (received as an Xlet) broadcast with the cooking show is printed and/or sent to the fridge for updating the supplying, so the Xlet needs to use the `Printing` service offered by the bundle of the printer and the `Supplying` service offered by the bundle of the fridge. In the next paragraphs, we explain in detail the collaboration of the objects which interact to perform the scenario. Since both communications are based on the same philosophy, we only focus on the former (printing).

The UML Collaboration Diagram of Fig. 8 shows all the interactions. Inside the bundle of the printer (`printer.jar`), the class `PrintingImpl` provides the implementation of the service interface `Printing`. The classes `masterXbundLET` and `MHP2OSGiXbundLET` are the ones which implement the service interface `Wrapped` inside the bundles `master.jar` and `MHP2OSGi.jar`, respectively. Master XbundLET implements, moreover, a service listener to track all service events in the framework.

An instance of the class `masterXbundLET` (`mxb` in the diagram) is the main object in that collaboration and its first task is creating a Service Listener (**1** and **2**). When the new service `Printing` is registered (**3**), the Master XbundLET detects this new service (**4**) and informs the IXC Registry (in the MHP side) about it (**5**).

The interaction between the MHP and OSGi worlds starts when the Xlet `rcp`, coming from the data carousel and initiated by the Application Manager (**6**), tries to print the recipe. Therefore, it looks for this service in the IXC Registry (**7**). Assuming no Xlet has required this service before, the reference stored in the IXC Registry corresponds to the Master XbundLET, so the Xlet sends the printing order to it. This is a consequence of our decision of creating the MHP2OSGi XbundLETs on demand, which avoids the overload implied by having XbundLETs to attend potential request from the MHP side that might never occur.

Once the MasterXbundLET receives the printing request (**8**) it reacts as follows:

- It creates an instance `pxb` of the `MHP2OSGiXbundLET` class which, like Master XbundLET, implements the XbundLET interface. An MHP2OSGi XbundLET is in

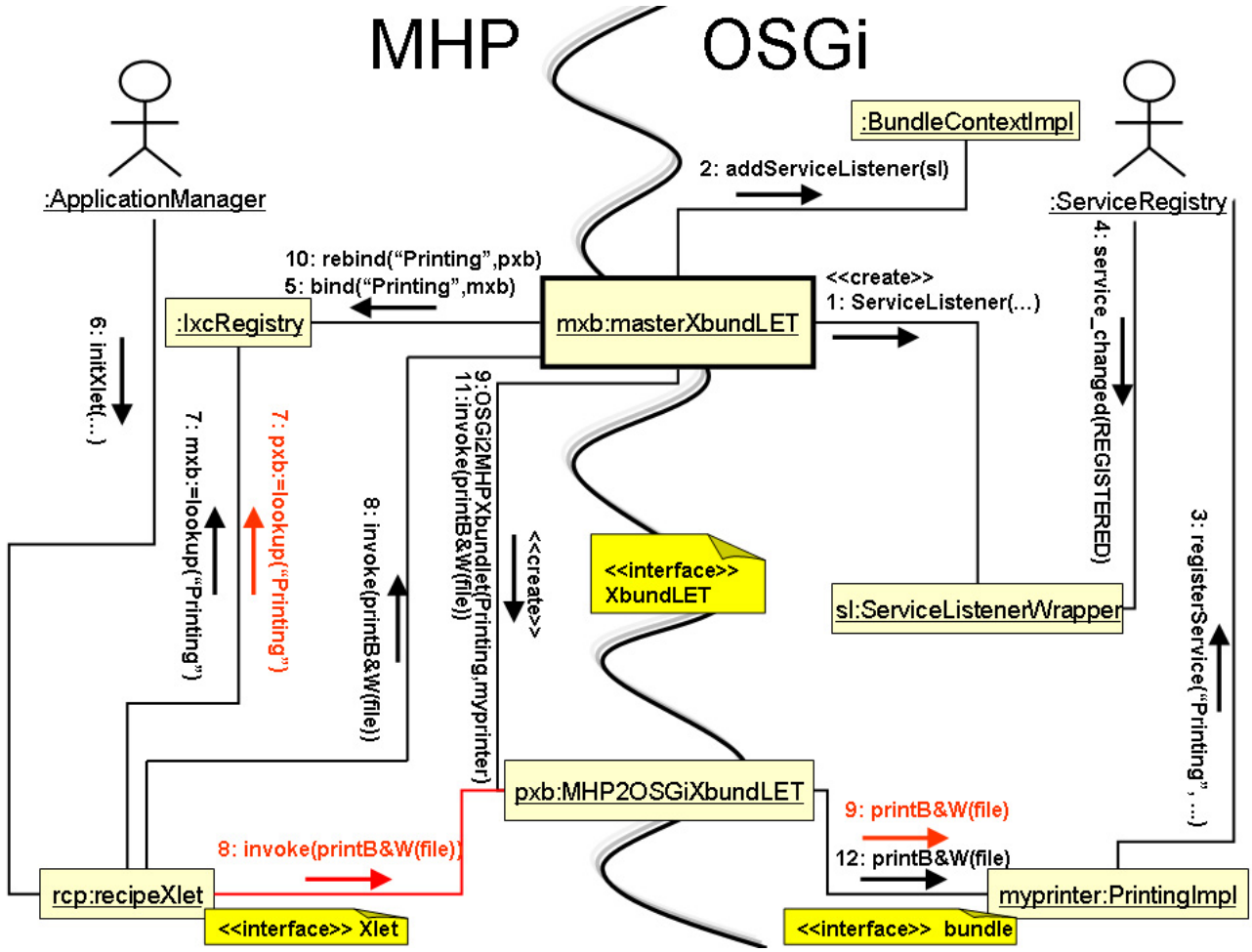

Figure 8: UML Collaboration Diagram for the interactions in invoking OSGi services from the MHP platform.. The invoking Xlet is `recipeXlet` and the invoked service is `printB&W` provided by the class `printingImpl` inside the bundle `printer.jar`. Grey arrows are used when the pxb object is already launched and ready

charge of sending service commands from the MHP to the OSGi side (**9**). These requests are wrapped under the order `invoke`. Apart from creating an instance to manage the service `Printing`, the Master XbundLET also have to customize this instance: constructing the wrapper according to the service interface `Printing` and linking it to the service object `myprinter`.

- It modifies the information stored in the IXC Registry (**10**), such that for subsequent requests, the reference returned to the corresponding Xlet belongs to the `pxb` object.
- It sends the `PrintB&W` command to the recently built `pxb` object in this case (**11**)

Finally, the MHP3OSGi XbundLET transforms the wrapped request `invoke(printB&W(file))` into the direct order `printB&W(file)`, which is sent to the service object `myprinter` of bundle `printer.jar` (**12**)

If the `pxb` object has already been created, the interactions would be slightly different. The reference given by the IXC Registry to the Xlet would be the `pxb` object itself. Therefore, the Xlet would directly send the tunnelled request to `pxb` (**8** in grey) and the order would be untunnelled by the MHP2OSGi XbundLET, to be correctly understood by the corresponding bundle (**9** in grey)

The final appearance of the application running in the *XleTView* emulator is shown in Fig. 9. The screen capture was taken just when the user selects the option of printing the recipe. As we can see, other options

Figure 9: Screenshot for the recipe Xlet running in the *XleTView* emulator.

like asking for help, sending the recipe to the fridge, or simply exit from the Xlet frame are also offered to the user.

8 Related Work

Although adopting networking facilities at home is a relatively new field, the technological novelties that have been appeared during the last years have totally changed the common perception of home networking. The appearance of digital information appliances like Internet refrigerators and microwave ovens can be considered the first effort of this technological progress. After that, attempts to develop home controllers to connect and manage all these appliances have raised naturally. However, these attempts are in a beginning state, mainly because there is not a wide consensus about the configurations and functions of this kind of controlling systems yet.

In fact, according to its main interests, each company proposes its own home server with differentiate characteristics. We can perceive these differences observing three sectors: communication companies, game companies and consumer electronics ones. First, communication companies are mainly keen on the network functions of the home server. For instance, Ericson have developed a home server called E-Box to provide home informatization and Internet services at home. Second, game companies focus on developing a home entertainment server based on their game consoles. Sony considers PlayStation, its world famous game console, as a next generation home entertainment server by adding network functions. Finally, consumer electronics companies, such as NEC, Hitachi, and LG also propose their own architecture of the home server adding PC functions to the consumer electronics products such as refrigerators. Within this sector, STB vendors also try to add new facilities to their products to support the control of the networked home.

On this matter, trying to converge to an unique solution is obviously the hot spot, and the current STBs have a strong potential to evolve into popular residential gateways [HBJK04]. Although many other factors are also crucial in this regard, from a technological perspective, the STB merges enough characteristics to be considered a suitable candidate. In fact, not only STBs offer a possibility of joining Internet services at home and entertainment, but also a suitable platform to support all the activities involved in a controller for home networking.

The first step through this convergence is assuring the interoperability between IDTV and the smart house. On this matter, and in accordance with EU Commission Recommendations, the European Committee for Electrotechnical Standardization (CENELEC) initiates broading the possible contribution base for technology consensus or standardization in the field of intelligent homes, by launching the CENELEC Smart House Workshop Initiative. In the Final Report [Cen03], as conclusions and recommendations, it is highlighted the potential of Digital TV/DVB standards for the Smart House. Therefore, the DVB consortium will have to play a major role as regards further Smart House Standardization, taking it into account for future MHP releases. In fact, the promotion of MHP as residential gateway is already reflected in the agenda of the DVB consortium with the creation of the MHP-HN (*MHP Home Networking*) group.

Meanwhile, other proposals have risen, like [TKK04], whose authors propose an architecture supporting the communication between both worlds. In their approach the STB and the OSGi gateway are placed in dedicated devices and the interaction of IDTV Xlets and OSGi bundles is implemented by sending special commands via an IP network. However, the solution is far from being a general one which supports legacy software. On the contrary, it mainly proposes a solution for delivering OSGi bundles via the broadcast carousel, and for controlling in-home services and appliances via the IDTV platform. The solution is not good enough to maintain the dynamic spirit of OSGi.

Besides, and since new possibilities at digital home are continously rising, researchers focus their attention on finding appropriate solutions to maintain the interoperability among the different home devices. The AVPACK project [iNR04] adopts and enhances the MHP framework to formulate a fully converged Audio/Video delivery platform specifically for residential entertainment devices, also called Video Gateways. In [MHWN03] a new architecture called *personal home server* is proposed to customize how to use the home appliances according to users' preferences. Continuing with this line, the authors of [KKKC04] propose an intelligent agent model, the UT-AGENT, for smart home environments, where the environment is able to learn about the user's preferences to assist them properly. In [KL04], the authors propose an implementation for the management agent of residential gateways. This role has been introduced in the OSGi specification, although its interfaces have not been detailed yet. Finally, some global solutions based on the OSGi platform have been published. For instance, the proposal of [BYK+03] introduces a global architecture to integrate a home multimedia server, a home control server and a home information server to achieve the so desired interoperability among all devices at home.

9 Conclusions and Future Work

In this paper we have explored the possible connections between two widely accepted standards, MHP and OSGi, in the context of Residential Gateways. If both OSGi and MHP specifications are promoted as the technological layout of residential gateways, their differences and, what is more important, their compatibility should be investigated.

In this respect, the work in this paper proposes a smart solution to this desirable integration. A conceptual entity, the XbundLET, acts as a bridge from the layered MHP to the service-oriented OSGi. An XbundLET acts like an Xlet in the context of the MHP system software (controlled by the Application Manager), and, at the same time, acts like a bundle in the context of the OSGi framework (monitored by OSGi Service Registry). Therefore, the desired cooperation is achieved without interfering any of both philosophies.

In a strong relation with the work introduced in this paper, we are currently working on the communication from OSGi to MHP. The same conceptual bridge, XbundLET, supports for bundles to use the functionality in the MHP platform: for instance, bundles showing messages in the TV screen or bundles querying an EPG (Electronic Programming Guide). Remaining the same the software solution which enables this other round of communication, our current objective is defining a set of MHP functionalities which would be appealing to construct services deployed through the OSGi framework. In fact, we are developing a proposal for "*MHP Service Specification* in the scope of the actual OSGi specification. The goal is to describe a bridging service between OSGi and MHP applications like, for instance, the one defined for UPnP ("*Universal Plug and Play (UPnP) Device Service Specification*)

Besides, in an environment were both platforms coexist, it would be very interesting to improve the way in which both kinds of applications search for the services they need. For the moment, this search is based on syntactic rules, though adding also semantic reasoning would surely allow more effective results. The objective is to build a *home networking* ontology in such a way that requests for services can be totally independent of the particular services names. What is more, within this desirable context, it will be also possible to tackle the personalization of the in-home services. Since our research group has gained experience in the field of TV personalization [BFPAGS+04], we are working now in applying this experience to the ambient intelligence area.

Citation

Rebeca P. Díaz Redondo, Ana Fernández Vilas, Manuel Ramos Cabrer and José J. Pazos Arias *Exploiting OSGi capabilities from MHP applications*, Journal of Virtual Reality and Broadcasting, 4(2007), no. 16, July 2007, urn:nbn:de:0009-6-11100, ISSN 1860-2037.

References

[All03] OSGi Alliance, *OSGi Alliance*, `www.osgi.org`, 2003, OSGi Service Platform, Release 3, last visited July 24th, 2007.

[BFPAGS+04] Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, Belén

Barragáns-Martinéz, Martin López-Nores, Jorge García-Duque, Ana Fernández-Vilas, and Rebeca P. Díaz-Redondo, *AVATAR: An Advanced Multi-agent Recommender System of Personalized TV Contents by Semantic Reasoning*, 5th International Conference on Web Information Systems Engineering WISE 2004 (Heidelberg), Lecture Notes in Computer Science, vol. 3306, Springer, 2004, ISBN 978-3-540-23894-2, pp. 415–421.

[BYK+03] Changseok Bae, Jinho Yoo, Kyuchang Kang, Yoonsik Choe, and Jeunwoo Lee, *Home Server for Home Digital Service Environments*, IEEE Transaction on Consumer Electronics **49** (2003), no. 4, 1129–1135, ISSN 0098-3063.

[Cen03] Cenelec, *European Smart House Standardisation in the eEUROPE context*, Tech. report, CENELEC, July 2003.

[HBJK04] F. T. H. Den Hartog, M. Balm, C. M. De Jong, and J. J. B. Kwaaitaal, *Convergence of residential gateway technology*, IEEE Communications Magazine **42** (2004), no. 5, 138–143, ISSN 0163-6804.

[iNR04] inAccess Networks R&D, *IST-34447-AVPACK project: Adoption and enhancement of open frameworks for converged Audio/Video provision to the home user*, 2004.

[JEF03] JEFFREE, *Java Embedded Framework FREE*, `jeffree.objectweb.org`, 2003, last visited July 24th, 2007.

[KKKC04] Neeraj Kushwaha, Minkoo Kim, Dong Yoon Kim, and We-Duke Cho, *An Intelligent Agent for Ubiquitous Computing Environments: Smart Home UT-AGENT*, Proceedings of the 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WSTFEUS), May 2004, ISBN 0-7695-2123-1, pp. 157–159.

[KL04] Kyu-Chang Kang and Lee Jeon-Woo Lee, *Implementation of Management Agents for an OSGi-based Residential Gateway*, Proceedings of the 6th International Conference on Advanced Communication Technology, vol. 2, 2004, ISBN 89-5519-119-7, pp. 1103–1107.

[MHP03] MHP, *Digital Video Broadcasting, Multimedia Home Platform (MHP) Specification 1.1.1.*, `www.mhp.org`, 2003, last visited July 24th, 2007.

[MHWN03] Kyoko Matsuura, Tadahiro Hara, Akashi Watanabe, and Tatsuo Nakajima, *A New Architecture for Home Computing*, Proceedings of the IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WSTFEUS), May 2003, ISBN 0-7695-1937-7, pp. 71–74.

[Ope03] OpenMHP, *OpenMHP 1.0.3b*, `www.openmhp.org`, March 2003.

[OSC05] OSCAR, *OSCAR: an OSGi framework implementation*, `oscar.objectweb.org`, 2005, last visited July 24th, 2007.

[OSG04] Knoflerfish OSGi, *Knoflerfish OSGi*, `www.knopflerfish.org`, 2004, last visited July 24th, 2007.

[Tec04] Galaxis Technology, *Mhp4free*, 2004.

[TKK04] Dmitry Tkachenko, Nickolay Kornet, and A. Kaplan, *Convergence of iDTV and home network platforms*, First IEEE Consumer Communications and Networking Conference CCNC2004, 2004, ISBN 0-7803-8145-9, pp. 624–626.

[Xle04] XleTView, *XleTView 0.3.6*, `xletview.sourceforge.net`, 2004, last visited July 24th, 2007.