On the Design of a SIP-based Binding Middleware for Next Generation Home Network Services

Mourad Alia[1], Andre Bottaro[1], Fatoumata Camara[1], and Briac Hardouin[2]

[1] Orange Labs., 28 chemin du vieux chêne, 38243, Meylan, France
e-mail: (alia.mourad, andre.bottaro, fatoumata2.camara, briac.hardouin)@orange-ftgroup.com
[2] Orange Labs., 4 rue du Clos Courtel , 35512 Cesson, Sevigne

Abstract. This paper proposes a two-layer component-based middleware framework that copes with the complexity of managing and constructing efficient and useful SIP-based home services. In the first layer, the device integration framework overcomes the heterogeneity of media home devices by providing protocol-independent components that reify the underlying devices. At the second layer, the binding framework allows constructing open mobile media bindings between SIP and non SIP communication protocol endpoints including media home devices. The openness of our framework is motivated by the need of constructing highly flexible home services such as context aware adaptation, session mobility, media session enrichment and QoS. Our framework is implemented as part of a context-aware adaptive middleware on top of the OSGi platform and an illustrative use case is shown.

1 Introduction

The proliferation of smart communication devices coupled with the improvement of networking infrastructures and ever increasing broadband penetration has made home networks a coveted platform for telecom and service providers to supply and to deliver their services. One direct consequence is the growth of voice-over-IP (VoIP) and multimedia home entertainment service market that has changed profoundly the common telecommunication landscape.

Within this growth, the adoption and the acceptance of the SIP (Session Initiation Protocol) [1], notably by the 3rd Generation Partnership Project (3GPP, www.3gpp.org), has particularly played a substantial role in reconciling mobile and fixed telecom technologies. In the traditional circuit-switched system, telephones were required to have essentially the same set of capabilities. The mechanics of reaching them were based on their being at the end of a particular fixed section of copper wire. In the new pervasive systems, devices become more autonomous, discoverable and controllable and a user can attach such devices anywhere on the Internet at any time and be immediately reachable. That device can have a wide range of capabilities including many different codecs, support for bidirectional video, and possibly even file sharing. In this context, the SIP protocol is used to reach the communication participants (rendezvous), to negotiate and re-negotiate session properties (media types, etc.) before establishing effectively a communication channel between them.

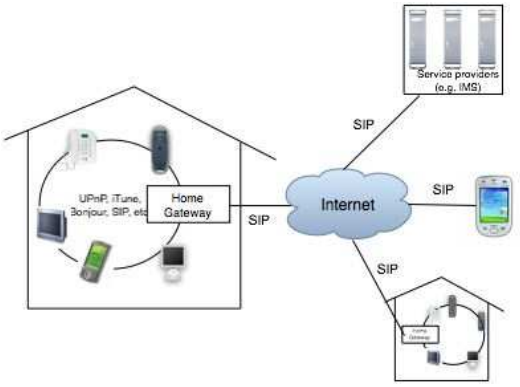

Fig. 1. Home Service Platform and the SIP protocol

As shown by figure 1, the SIP protocol is used for interpersonal communication enriched with services (video sharing, presence, gaming, etc.) that involve an outdoor mobile user with a SIP mobile handheld device or yet another home network peer. Further, SIP is used to access to service providers services – such as typically IP IMS (Internet Multimedia Subsystem) [2]. Within this context, the objective of our work is the design of next generation home network middleware platform for the development and the deployment of flexible SIP media home services. The flexibility is stressed by the need of supporting dynamic concerns management such those related to context aware adaptation, media session mobility and QoS that obviously require dynamic reconfiguration mechanism.

The researches on the design of home services platform have mainly focused on the problem of indoor devices heterogeneity, dynamicity and composition. Only few of them address the problem of mixing SIP interpersonal communication with multimedia home devices [3–6]. Although the majority of these works follow the SOA (Service Oriented Architectures) paradigm on top of the OSGi platform, the proposed solutions lack of openness and flexibility: these solutions are hardly reusable to develop flexible and useful SIP-based services. From the Telco side, much effort is done in the specification of the 3G SIP Core IMS framework [2]. However, this framework targets mainly the core network side shared between the different operators without real emphasis on home platform.

Our open middleware solution is a two layered architetcure. At the low layer, the middleware provides a device integration framework that overcomes the heterogeneity of media home devices (e.g., UPnP/DLNA (`www.upnp.org`), Apple Bonjour/iTunes (`www.apple.com`), and IGRS (`www.igrs.org`), SIP) by providing protocol-independent components that reify the underlying devices. On top of this framework, the binding framework, which is the main contribution of this paper, offers necessary abstractions for the management of SIP-based media sessions that involve SIP and non SIP endpoints. Each session is designed as an explicit *open binding* component between heterogeneous home media devices which embodies media type negotiation and adaptation

and offers reflective control interfaces in order to introspect and to reconfigure the managed session. The approach is assessed through an implementation of the framework on top of the OSGi platform as well as its integration into a generic context-aware adaptive system. The implementation is further accompanied with an illustrative example and few evaluation numbers.

The remainder of the paper is organised as follows. Section 2 gives a brief overview on the SIP protocol principles. Then, section 3 discusses the motivations, the requirements and the overall architectural approach. Section 4 presents the device integration framework. Section 5 describes the SIP-based binding framework architecture and its main interfaces. The OSGi implementation of our framework is presented by section 6. Finally, section 7 compares our approach to the existing approaches before section 8 concludes.

2 SIP Protocol Overview

SIP [1], or Session Initiation Protocol, is an application network layer signaling protocol used to establish a relationship between endpoints so that ongoing bearer paths between them can be established, modified and torn down. SIP infrastructure is built on *proxy* and *registrar* servers that form an overlay networks on regular IP networks. SIP messages transit through proxy servers which are in charge of routing SIP messages to the current SIP callee location. Proxies use registrar SIP servers to resolve SIP names and to get the current mobile user location. SIP uses URIs for user addressing, in the form of e-mail addresses:user@domain. Signaling messages (`invite, ok, ack, bye, option`) are exchanged asynchronously through SIP *User Agents* (UAs).

Figure 2 illustrates a typical interaction diagram to establish a media session between Greg and Fatou users. Once the callee is localised, the UAs exchange signaling messages to negotiate the media session properties to be established between the involved endpoints. The localisation is carried out transparently by the underlying SIP infrastructure (proxies + registrars) while the session negotiation depends on the endpoints properties. The objective of the negociation is to establish a contract between the involved endpoints using SDP [7]. This contract includes primarily parameters related to the media types (video, audio, etc.), transport protocol (RTP/UDP/IP, H.320, etc.) and the format of the media (mpeg, H.261, avi, etc.), IP addresses and the ports where to receive the different medias. Furthermore, it could also contains information related to the QoS. Part (a) of figure 2 shows a typical SDP description scheme.

The established contract could at any moment be renegotiated by sending a re-INVITE method on the current session with the new SDP parameters. SIP offers also a REFER method [8] so as to implement call transfer reconfiguration. This method allows one SIP endpoint to move the session – with all the multimedia flows – to another SIP endpoint much like call transfer in phone calls.

In addition to the above signaling methods, many RFCs propose methods for particular communication purposes. One can cite the MESSAGE method is used to implement instant messaging (IM) [9] and PUBLISH and NOTIFY methods to the presence [10]. This makes SIP agnostic in the sense that it can support any type of communication session whether it is voice, video, IM and presence or a combination of them.

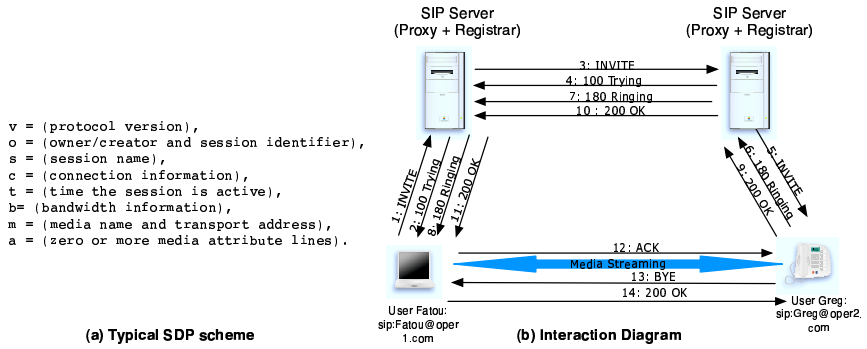

Fig. 2. SIP Architecture and Principles

One interesting property of SIP is that it separates control – i.e signaling – from the multimedia flows. While this architectural principle provides a good basis to construct reconfigurable multimedia bindings, the following section shows that it is not enough to build context- and QoS-aware SIP home services.

3 Challenges, requirements and overall approach

A home network is a pervasive system composed of users and heterogeneous networked devices. In what follows, we emphasise the need for an open and flexible middleware through a set of architectural requirements.

- **Device heterogeneity and media adaptation:** The home network is filled with multimedia devices. However, making devices interoperate between each other remains a challenge. Devices offer services through various protocols. The envisioned use case mixes home multimedia and interpersonal communication application domains. In each of these domains, many signaling (control) protocols are well-spread. In the multimedia domain, UPnP/DLNA, Apple Bonjour/iTunes and IGRS technologies are taking most of the market shares. In the interpersonal communication domain, SIP is becoming the main standard technology while many proprietary protocols exist. In order to make interpersonal communication protocols interoperate with home multimedia ones, home gateways not only need to bridge signaling protocols but also to negotiate multimedia formats of the sessions and adapt the media types and protocols if needed. This is typically the case when streaming media between UPnP/DLNA devices that use exclusively HTTP and SIP endpoints which are usually RTP-based.
- **Media Session enrichment:** The first motivation of this work is to enrich interpersonal communication between users of a home or distinct homes with the use of available home multimedia devices. For instance, a user talking on the phone in the lounge with a friend would be able to see his friend on the TV screen. The scene is being captured by a movie camera in order for the friend to be also able to watch

the caller speaking. This use case becomes feasible today thanks to the connection of indoor devices to the home network and to the SIP functionalities for outdoor connections.

– **Media session mobility management:** Session mobility allows a user to maintain a media session even while changing terminals. For example, a user may want to continue a session begun on a mobile device on the desktop PC when entering her office room. He may also want to move parts of a session, e.g., if he has specialized devices for audio and video, such as a video projector, video wall or speakerphone. Another typical example is the so called *follow me* (see section 6.3) scenario where part of the multimedia session is transferred from one screen to another following the user position at home. Session mobility using SIP can be programmed either using a re-INVITE request that changes the session properties including the new IP addresses and ports to be considered by the sender using the third-party call control (3PCC) [11] or the REFER mechanism in case of call transfer reconfiguration [12]. The main concern of our middleware regarding this issue is to offer high level control routines that allow moving part of a given session between SIP and non SIP endpoints.
– **QoS, context-awareness and adaptivity:** As the user is the main actor in our application domain, the QoS management related to multimedia streaming is one of the important aspect that our middleware should not ignore. QoS management is a balance between the user preferences, the provided services, the existing resources (e.g. computational resources) and the environmental contexts (e.g. lightness). The QoS objectives may be achieved by changing the media encoder type or by updating the media scheme quality at the source side or by buffering the media in order to support time shifting streaming, etc [13, 14]. Thus, our middleware should allow a QoS - aware deployment of SIP media services and context-aware dynamic adaptation. A prerequisite for supporting that is to reconfigure dynamically media sessions which may involve changing the negotiated codec or yet updating the video frame quality, etc.
– **Hiding the SIP complexity:** The SIP protocol is a low level signaling protocol that needs expert knowledge. Current developments of SIP services are usually based on the usage of the SIP stack (e.g. JAIN: http://java.sun.com/products/jain) component which basically implements the signaling behaviour of the SIP UA. From the middleware viewpoint, this is somehow equivalent to saying providing a TCP/IP stack to the developer to construct their own RPC or RMI. Therefore, there is a need for more abstractions to integrate seamlessly the SIP protocol with media streaming and home devices so as to provide a higher level framework for the development of SIP media home services. This will enforce the developer to avoid merging between signaling (INVITE, ACK, etc.) and application – or business – code.

The previous challenges and requirements stress the need for an open and flexible middleware solution to constructing adaptable SIP media home services. Figure 3, depicts our overall middleware approach. It is a two layered framework composed of the device integration framework and the SIP binding framework. The device integration framework acts as as a low level framework for the discovery and the access to home

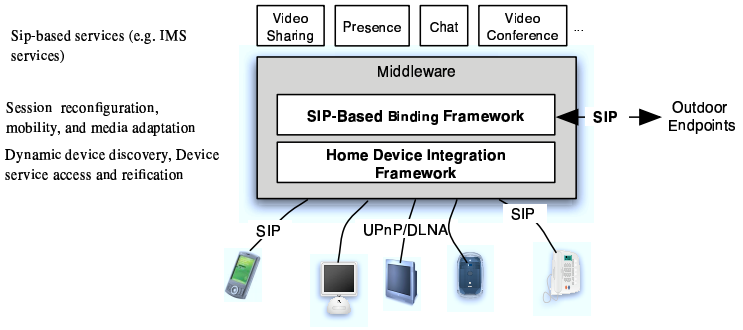

Fig. 3. Overall Approach

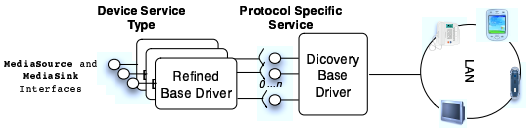

Fig. 4. Home Device Integration Approach

devices overcoming the device protocol heterogeneity. The SIP binding framework allows the creation and the management of reconfigurable SIP-based bindings that makes interoperable SIP and non SIP endpoints. The following sections present in detail each of these frameworks.

4 Device Integration Framework

The aim of the Device Integration Framework is the discovery and the control of available media devices on the home network and especially media devices. It copes with:

- *Distribution*: The framework allows to access to the distributed underlying device services transparently using classical proxies generation techniques.
- *Protocol Heterogeneity*: The framework makes it possible to hide the underlying device protocol using mediation techniques that adapt protocol-specific objects into application-specific interfaces. The separation between media interfaces and protocol-specific implementations render the applications independent of the protocols and able to bind available media instances at runtime.
- *Dynamicity*: The framework is aware of the dynamic availability of devices.

This framework is based on our earlier approach [15] with further emphasis on media-based devices. As depicted by the figure 4, it is a two floors integration approach represented by the *Base Driver* and the *Refined Driver* components.

4.1 Device discovery and proxy generation

At the first floor, every *Base Driver* discovers and manages one set of protocols and reifies the available devices into a protocol-specific object without distributed and parsing details. They use the specific features of the associated set of protocols in order to maintain the set of available devices up to date: active and passive discovery, metadata description retrieval (see [16] for the description of plug-n-play protocols on local networks). Base Drivers register, modify and unregister a representative device object proxy in a registry according to its availability on the network.

4.2 Media device interfaces specialisation

At the second floor, *Refined Drivers* wrap the first floor device services into objects implementing common media service semantics related to the device role. In our case, one distinguish between `MediaSource` and `MediaSink` interfaces (i.e., Media Device Type in figure 4). Typically, UPnP Media Servers, UPnP Media Renderers and iTunes servers are represented by protocol-specific proxies by base drivers. Refined drivers will then wrap these proxies into objects implementing protocol-independent interfaces that are media sources and sinks. Refined Drivers react to the registration - resp. deregistration – of the Base Driver proxies in the registry and register - resp. deregister - adequate adapters. These dynamic chained reactions make `MediaSource` and `MediaSink` instances be available on the platform whenever matching devices are available on the network.The `MediaSource` interface offers methods for browsing a media content repository, negotiating streaming parameters and controlling the streaming of these items. The streams can be offered through various streaming protocols, mainly RTP, HTTP, and various media formats, e.g., mpeg2, divx. The `MediaSink` interface exposes methods for negotiating streaming parameters and controlling rendering features. Rendering features are related to the control of video rendering parameters, e.g., brightness, colours, and the control of audio rendering ones e.g., the sound levels, balance.

Compared to usual home devices (UPnP, iTunes, etc.), SIP home devices are managed differently. The first difference is that SIP devices are not dynamically discoverable. The second one is that they could play both sink and source roles at the same time. Therefore, a `SIPDevice` interface is introduced to reify SIP device services (see section 6.1 for more details).

5 SIP-based binding framework

Constructing flexible home SIP-based services comes down by constructing and managing binding chains between SIP endpoints and non SIP endpoint home devices which constitute the main objective of the binding framework. The SIP endpoint primarily refers to a remote SIP outdoor peer that could be a mobile user handheld or another home network. It could also merely represent an indoor SIP home device.

As depicted by the part (a) of figure 5, a SIP-based session is composed of – or put together – a set of participants, SIP user agents, a set of media and a set of home

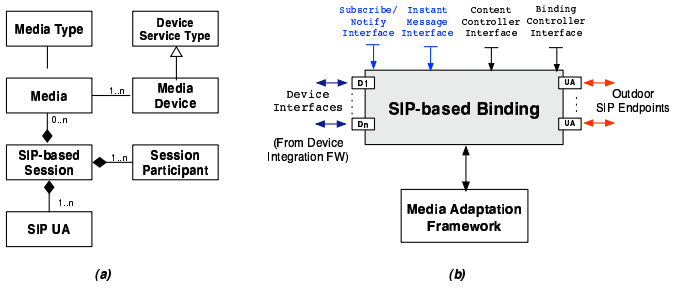

Fig. 5. SIP-based binding model

devices. A session participant refers to the SIP user involved in the session. Depending on the SIP services, one could have one or more users involved in a given session. One can imagine simple case, that is one user – when the user is streaming a media inside the home between a SIP and non SIP device or more elaborated scenarios with many users such as typically a video conference service. The SIP User Agent is used to communicate with the different SIP endpoints. Each session is composed of a set of typed media streams that are defined through the media content type and the protocol delivery type. Each of these media is streamed to or streamed from a given device represented by a device service.

5.1 A SIP-based open binding model

The binding framework is the notion of binding as it is defined in the ODP specification [17] with further extensions. As represented by the part (b) of figure 5, a Home SIP-based Binding is modeled as a software component that plays the role of a third party *signaling orchestrator* to negotiate, establish and reconfigure sessions between SIP endpoints and media home devices. The interaction with SIP endpoints is done through UA interfaces whereas the interaction with the involved indoor devices is carried out through their device interfaces that are generated and exported by the device integration framework (See section 4). The Media Adaptation Framework is used by the Home SIP-based Binding component to adapt the media streams between the sinks and the sources if necessary. The Binding component is open in the sense that it provides interfaces that allows controlling and updating a given session.

Media negotiation and session establishment: Before the effective establishment of the session, media negotiation is the first task that the Home-based Binding component has to perform between the sinks and the sources. The negotiation may concern different aspects of the including protocol types, ports, media types and possibly extra-functional aspects such as QoS. As in the SIP protocols, we adopt SDP (Session Description Protocol) to describe the exchanged medias between the various endpoints. Basically, the Sip-based Binding component requests the supported protocols from the involved indoor device and gets the supported protocols from the SIP endpoints through the INVITE

method. If the negotiation fails, then it uses the Media Adaptation Framework to create an appropriate media transformation pipe that adapts the media between the sources and the sinks. More details on this component are given in section 5.2. For example, in the case of SIP and a DLNA/UPnP endpoints, the associated Binding component has to create a media adaptation pipe that transforms the media delivery protocol from RTP to HTTP and possibly the media codec type if the SIP and the DLNA/UPnP media schemes are not compatible.

Dynamic reconfiguration: Once the session is established and the streaming started, the Binding component allows to reconfiguring the session using the controller interfaces. One distinguish between the following three reconfiguration types namely Dynamic contract update, Mobility, and Media enrichment.

Dynamic contract update refers to reconfiguration that update the session properties between the involved endpoints that basically aims at modifying the QoS. A typical example is to change the encoding scheme with a less or more consuming network bandwidth or yet to change the source (e.g. media server) streaming properties (luminescence, etc.) for the media quality adaptation purpose. Note that this reconfiguration does not have an influence on the binding structure. From the SIP endpoint side, the Binding component uses the re-INVITE method to modify the streaming properties whereas it uses the appropriate method call on the device interface in case of indoor home device.

Mobility refers to session mobility which is handled by the the Binding component through the `Binding Controller` interface (See figure 6). This interface offers methods that allows to bind/unbind to a given device dynamically. To bind to a new device, the whole session establishment process including media negotiation is re-executed using the re-invite SIP method to update the session properties. For example, to switch between two media UPnP renderers (sinks), the bind method sends a re-INVITE method to the remote SIP endpoint with the new supported encoders, the IP address of the new device, etc. The bind method could also be used to send the same media stream to different devices in which case the Media Adaptation Framework is used to broadcast the media to the involved devices.

Regarding *Media enrichment* reconfiguration, new media streams could be added or removed from the current session. This could be initiated either by the remote SIP endpoint on the current SIP session ID of the Binding component or by the local endpoint – that is also the Binding component. Locally, this comes down to bind or unbind to or from the given device using the `Binding Controller` interface and remotely, this is achieved by sending new SIP re-INVITE messages. Streams that are added or removed are described using the SDP protocol.

Synchronous and asynchronous communication: In the controller interfaces, a SIP-based component binding could be enriched with synchronous and asynchronous message communication between the involved SIP endpoints. The Binding component offers the `Instant Message` interface to send/receive messages using the MESSAGE SIP method to/from the remote endpoints [9]. It also provides the `Subscribe/Notify`

Binding Controller Interface	Bind and unbind media instances to a given device. Add or remove a given media within the current binding. Transform a given media (Mux, demux), etc. Update the session properties (e.g. choose a new codec, etc.) Transfer/delegate the binding call to another SIP participant. Add another SIP participant into the binding.
Content Controller Interface	Browse the session content including: devices, participants and the medias instances.

Fig. 6. Controller interfaces methods description

interface that basically allows to subscribe/notify to events with the remote SIP endpoints using the SUBSCRIBE (and NOTIFY) SIP event methods [10]. Recall that these operations could be used in parallel with media streaming between the SIP endpoints.

The SIP event binding feature could be used to keep track of the listening status of a set of SIP accounts. This allows for avoiding sending SIP INVITE messages to check whether the receiver is listening or not. In a wider context, this could be used to be notified when an alarm goes off in the home, a certain temperature is reached, or someone rings the doorbell in the context of home automation.

The binding component could be seen as a pattern that is used to construct controllable and flexible SIP-based services. The different controller interfaces, the messaging and the event interfaces are not mandatory. Indeed one can have binding component configurations that are used to construct very simple services. For example a messaging service is constructed using a binding component only with the `Instant Message` interface without media streaming. The same principle is applied to construct a Presence service with the `Subscribe/Notify` interface. It is also possible to have a binding component for media streaming without the controller interfaces if one does not need to reconfigure this service. Obviously, it is also possible to construct more elaborated services by combining the different interfaces. The controller interfaces are used to reconfigure and to adapt the service. The realisation section 6 gives an overview on how the SIP-based component is used and exploited to construct generic context-aware adaptive home services.

The binding component is responsible for the set-up and the reconfiguration of SIP media sessions. Hence, this component is not resource consuming and could therefore be deployed in a handheld device to remotely control SIP media sessions at home as well as in a home gateway. However, this is not the case of the Multimedia Adaptation Framework which is CPU intensive and therefore should be deployed in one or many machines to provide a better QoS.

5.2 Multimedia Adaptation Framework

The main responsibility of the Multimedia Adaptation Framework component is to provide a multimedia adaptation proxy between sources and sinks related to the session managed by a given Binding component. This adaptation could be related to both media content type (e.g., *audio/x-wav, audio/basic*, *audio/mpeg*, *audio/midi*, *video/mpeg*)

as well as the media delivery protocol (e.g., *local file, disk I/O, HTTP, RTP live media streaming*. In more than these adaptation types, this component could provide enhanced media processing operations such as multimedia mixing and multiplexing.

This component is not the added value of our work. We reuse the research results in media transcoding adaptation [18–20] where one can find many open source frameworks such as NMM (www.networkmultimedia.org) and GStreamer (www.gstreamer.freedesktop.org).

As in all these works a media adaptation is represented a media flow graph where one can distinguish between source and sink nodes that are based on protocol delivery, multiplexer, demultiplexer and processor nodes (encoders, decoders, filters), etc. Architecturally, this is usually represented as a component composition where each component represents a given node type. In the realisation section, the diagram represented by figure 8 gives an example on a media adaptation pipe SIP/RTP - DLNA/HTTP adaptation (see operations 5 – 10).

Basically, this component offers basic functions to create, update and resume graph flows used by the Binding component. The updates may involve changing one given graph node implementation with another, tuning some node component parameters, or yet updating the deployment graph in case of many machines are exploited. The ultimate objective of such reconfigurations is to improve the overall user utility and QoS thanks to the user preferences [13, 20].

6 An OSGi realisation and evaluation

Based on the previous architectural models, a version of our middleware is implemented on top of the Apache Felix (felix.apache.org), an open source OSGi compliant implementation. The OSGi dynamicity, composability and deployment properties provides key architectural ingredients that facilitate the implementation of our framework.

Figure 7 represents the overall architecture of our system as a set of interconnected and pluggable components through provided and required service interfaces according to the OSGi component model. As to evaluate the flexibility of our architecture, the middleware is integrated into a context-aware adaptation loop control through the *Context Manager* and the *Adaptation Manager* components explained in the following sections.

6.1 Core middleware components

The **Device Integration Framework** implements and supports various devices including UPnP, Apple Bonjour and SIP compliant media devices. In the figure 7, the **Device Discovery** is the entry point to search for and to discover the different media devices using the OSGi service registry. Customised Base Driver bundles namely the **UPnP Base Driver**, **Bonjour Base Driver**, and **SIP Base Driver** are dedicated to reify respectively UPnP, Bonjour, and SIP media devices through typed objects that are registered/unregistered in the OSGi service registry. `UPnPDevice` objects are constructed and registered in the service registry according to the network availability of the corresponding device thanks to UPnP/SSDP active and passive discovery protocol and

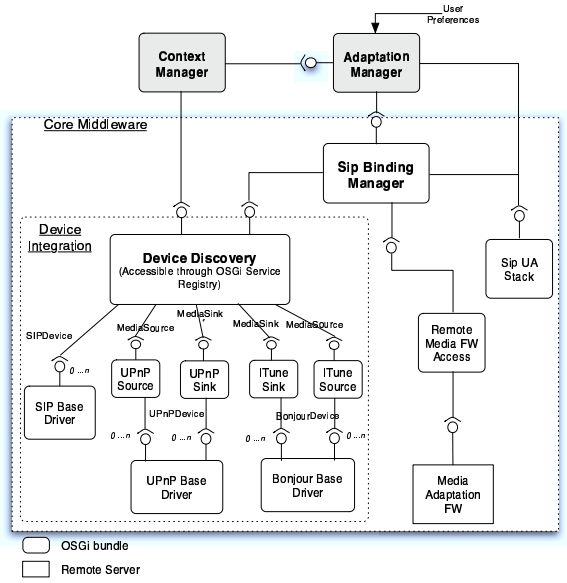

Fig. 7. Overall Architecture

thanks to the OSGi UPnP Device Service specification [21]. Methods invocation on the `UPnPDevice` objects are translated into UPnP SOAP message communications accordingly. Similarly, `BonjourDevice` objects are registered in the service registry with the attribute-value pairs matching device properties.

According to the pattern described in section 4, **Media Refined drivers** are dynamically reacting to the availability of home media devices. Regarding UPnP, `UPnPDevice` services registered with the Media Server type – `urn:schemas-upnp-org:device:MediaServer:x` – and the Media Renderer type – `urn:schemas-upnp- org : device : MediaRenderer:x` – and register `MediaSource` and `MediaSink` services wrapping the matching `UPnPDevice` objects. Since iTunes relies on a protocol set dedicated to the multimedia application domain, it is directly treated as a media refined driver that dynamically tracks `BonjourDevice` services of the iTunes type – `daap.tcp.local` – and register `MediaSource` services wrapping `BonjourDevice` services.

We have also implemented a **SIP base driver** bundle which represents home SIP devices through the `SIPDevice` interface. Our SIP base driver is associated to a SIP registrar to which the home SIP devices register. Since SIP does not have a service discovery protocol defined on local networks, this approach allows the dynamic platform notification of the presence of SIP devices at home.

The **SIP Binding Manager** is responsible for managing SIP-based Bindings components according to the pattern described in section 5.1 including creation, reconfiguration and releasing.

The **SIP UA Stack** encapsulates the SIP JAIN stack[1] which provides usual SIP UA services to send and receive SIP messages with remote SIP endpoints. Typically, this component is charged to notify the **Adaptation Manager** about incoming calls and depending on the user preferences instructs the SIP Binding Manager to create an appropriate Binding Component with the selected home devices.

Finally, for the **Multimedia Adaptation FW** we reused and customised an existing framework as a remote server that is used to create media graph flows that transcode and adapt the media streams. Developed in C, we developed the **Remote Media FW Access** bundle which is used to control and to use the transcoder functionalities from the OSGi platform. In more than usual remote method invocation, this component provides a SIP UA service which is used to instruct the remote server to create an adaptation graph using the SIP signaling [22] (see figure 8).

6.2 Context aware adaptation components

The logic of the adaptation of services is embedded within the **Context Manager** and the **Adaptation Manager** components. As in usual context-aware models, the Context Manager is responsible for collecting information that influences the system (i.e our SIP-based services) through appropriate context sensors, aggregates semantically this information and notifies the Adaptation Manager component of relevant context changes. Depending on the adaptation objectives, context information could be related to computing resources including home devices or yet environment state. The Context Manager exploits naturally the OSGi service registry dynamic behavior to subscribe to given system changes. The latter are not only the appearance and the disappearance of the media devices but also the device property changes that are directly performed by the context manager on the registered service references [23]. The Adaptation Manager component is responsible for reasoning on context changes and taking decisions whenever an adaptation is required. This reasoning embodies adaptation policies that could be of action based type (if Event then do ...) or more elaborated approach such as utility-based [13]. The derived decisions consist mainly of invoking methods on the controller interfaces of the SIP-based Binding component that used to manage the substrate SIP-based media services exhibited through the SIP Binding Manager. The targeted adaptation type may vary from simple usefulness adaptation types such as the follow me scenario (see next section) to more elaborated models that target QoS concerns [13].

6.3 Experimenting a context-aware adaptive service (Follow me)

The aim of this section is to show the flexibility of our framework by experimenting a SIP-based audio/video communicator service that supports the so called follow-me behaviour. The overall principle is to switch between the different available media devices – either sources of sinks – depending on the user location that is the current home room during a SIP-based multimedia communication. The user named Fatou is moving between the living-room, which is equipped with a SIP softphone, and the bedroom, which is equipped with a DLNA/UPnP TV media renderer and a UPnP camera server. From

[1] https://jain-sip.dev.java.net/

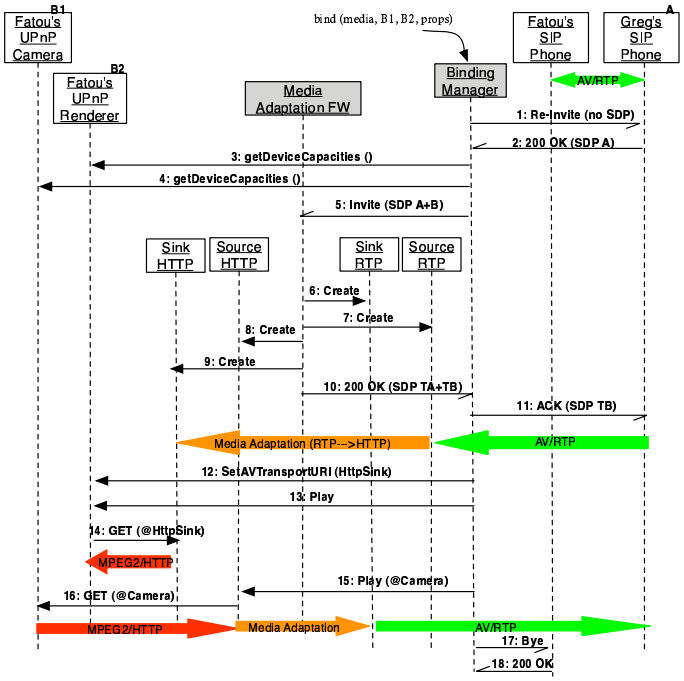

Fig. 8. SIP-UPnP binding reconfiguration diagram

the outdoor, the second user named Greg initiated a SIP conversation with Fatou on her SIP softphone while she is in her living room. During the communication, the users exchange symmetrically their online users' images taken from a camera device and the audio conversation. Our middleware makes it possible to continue the communication when Fatou moves to her bed room hands free. Greg's image is then automatically displayed on the bedroom UPnP TV and Fatou's image, which was initially captured by her softphone, is now captured through the bedroom UPnP camera. Literally, the context manager notifies the Adaptation Manager about the new location of Fatou which in turns executes the bind/unbind methods on the corresponding Binding component. Figure 8, shows the bind method interaction diagram related to this scenario and which involves video session mobility adaptation from SIP phones to UPnP devices with RTP to HTTP media adaptation.

In the figure 9, the window on the left shows the entry point of the application from Fatou's view. In the current implementation the context information related to user location change are simulated by just a clic on the right room from the list when the user move from one room too another. The window on the right side represents a Wengo SIP softphone, which is used to simulate the SIP Device. WengoPhone (www.openwengo.org) is a free software SIP compliant VoIP client developed by the OpenWengo community under the GNU General Public License (GPL).

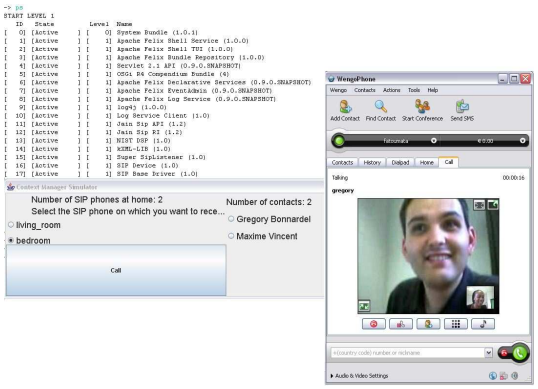

Fig. 9. Experience snapshot

Quantitatively, the reconfiguration time average SIP-UPnP related to this experiment varies from 3 s to many seconds depending on the used UPnP media renderer (DMR). Thus we can sometimes perceive a certain latency at the DMR. This latency is partly due to the fact that the DMR starts playing only when the buffer is full. We have also measured the reconfiguration time in case of SIP to SIP mobility without media adaptation from one SIP soft phone to another. The average time is about 0.325 seconds which is obviously strictly less than SIP-UPnP reconfiguration (SOAP messages and Media adaptation overheads).

7 Related work

During the last years, architecting home network services has been the target of many researches generated from different communities varying from middleware to telecommunication ones. In the following, we distinguish between works on binding middleware frameworks and works that integrate SIP and UPnP into the OSGi platform.

7.1 Open binding middlewares

Constructing open media frameworks has emerged during the ten last years as one of the important core component of reflective middlewares such as [24–26]. The design philosophy of these works is to represent a binding as an explicit object that could be manipulated by separating the control and management concerns from the functional aspect of a given application. This is particularly the case of multimedia middleware where a multimedia binding type is constructed as a component composition [27–29] that could also be seen as a kind of nested bindings from protocol to the media session.

While our design approach inherits from these works principles, one think that it provides a step further by exploiting the recent advances in device technologies, coupling media devices with the binding framework with the clear separation between the

signaling and the streaming thanks to the SIP protocol. Furthermore, besides OSGi, our work could also be seen as a way of componentising the SIP protocol using a component model close to Fractal [25] and OpenCom [24]. This provides a good basis to integrate the SIP binding type into such open component-based programming models thus offering all the ingredients to construct adaptive home services.

7.2 Indoor and Outdoor protocol integration: SIP, UPnP into a modular platform

Although the use of SIP to address various devices in local networks from the wide area network is interesting, SIP relatively failed in gaining market shares in the home network. Service Discovery Protocols, e.g., UPnP SSDP, DPWS ws-discovery, Apple Bonjour, are better at dynamically discovering and identify devices in local networks. Particularly, this has driven many researches on how to combine SIP for interpersonal communication with UPnP as a de-facto AV home protocol [30, 3]. [30] is especially relevant since it splits the VoIP device into a SIP phone, a UPnP Media Server and a UPnP Media Renderer. However, the link is tightly made with only one device control protocol, the paper does not make the reader perceive the code modularity of a framework written in the C language and it finally fails at controlling real legacy UPnP devices that would have provide HTTP streams that are incompatible with the SIP defacto RTP choice.

In some approaches such as [5, 4, 6], SIP-UPnP bridging is carried out by just inserting UPnP control messages into SIP protocol as text messages using the SIP MESSAGE method. Consequently, the SIP protocol is miss used and the sense that It does not exploit its main strengths related to the mobility and session reconfiguration. Our approach keeps the protocols as they are and bridge them opportunistically by inserting intelligence in bridging platforms that host innovative applications, that is taking the best of every detected protocol set to satisfy the user activity.

Bushmitch, Brown et al. [4] targeted also the remote access use cases with a specific bridge between SIP and UPnP protocol sets. They moreover showed the interest for a design above the OSGi platform in [5]. Although we think that the OSGi platform is the right technology to bring loosely-coupled and well-structured applications to the home, the API shown by their work lacks the essentials of the OSGi Service Oriented Programming. SIP protocol details are not transparent to the developer and event SIP entities are not programmed as OSGi services (see the static SIPDevice factory in [5]). The dynamicity of SIP services ought to be mirrored by OSGi service dynamicity.

Even though these works follow an architectural approach to integrating SIP, UPnP into OSGi, the proposed solutions do not provide a general solution to handle the interoperability between two different device protocol media communications. Consequently, the problem of session dynamicity and reconfiguration has not been considered and the proposed systems do not provide any support for the management of aspects such as session mobility and media enrichment with respect to our requirements.

8 Conclusion

This paper presents the architecture of an open component-based middleware that allows constructing and managing efficient and useful SIP home services. The overall approach consists of revisiting the SIP protocol from a software engineering viewpoint and proposes a binding framework that hides the underlying protocol complexity and at the same time exhibits and exploits many interesting functionalities. This binding framework is coupled with a low level device integration framework thus bridging home device protocols with the SIP protocol to enable a flexible and efficient interpersonal communication. The proposed architectural models are mapped down and implemented on top of the OSGi platform accompanied with a real word example that illustrates the usability of our approach thus facilitating the development and deployment of useful and adaptive next generation home services.

The openness of the middleware architecture proposed here is a prerequisite for dealing with QoS management. Indeed, the fact that our SIP-based services are componentised with support to dynamic configuration offers a way to reason on the underlying architectural artifacts. We think in particular that the general QoS approaches that are based on component compositions selection could be applied in our context [13]. This issue is being investigated at the second stage of the middleware development.

Furthermore, our framework could obviously be part of a much wider middleware component framework that take into account other concerns such as security and content integration as it is partly shown in section 6. This will be done in our ongoing IMS proxy implementation on the home gateway.

References

1. IETF RFC 3261: SIP: Session Initiation Protocol (2002)
2. 3GPP Technical Specification Group Services and System Aspects: IP Multimedia Subsystem (IMS). Technical report (2006)
3. Kumar, Rahman, B.: Mobility support for universal plug and play (UPnP) devices using session initiation protocol (SIP). Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE **2** (8-10 Jan. 2006) 788–792
4. A., B., Kolberg, Bushmitch, M., Lomako, D., Ma, G.: A SIP-based OSGi device communication service for mobile personal area networks. Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE **1** (8-10 Jan. 2006) 502–508
5. Bushmitch, Lin, W., Bieszczad, Kaplan, A., Papageorgiou, A., Pakstas, A.: A SIP-based device communication service for OSGi framework. Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE (5-8 Jan. 2004) 453–458
6. Moyer, S., Marples, D., Ghosh, A.: Service Portability of Networked Appliances. IEEE Communications Magazine (February 2002)
7. IETF RFC 2327: SDP : Session Description Protocol (1998)
8. IETF RFC 3515: The Session Initiation Protocol (SIP) Refer Method (2003)
9. IETF RFC 3428: Session Initiation Protocol (SIP) Extension for Instant Messaging (2002)
10. IETF RFC 3265: Session Initiation Protocol (SIP)-Specific Event Notification (2002)
11. IETF RFC 3725: Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP) (2004)

12. Schulzrinne, H., Wedlund, E.: Application-layer mobility using SIP. SIGMOBILE Mob. Comput. Commun. Rev. **4**(3) (2000) 47–57
13. Alia, M., Eide, V.S.W., Paspallis, N., Eliassen, F., Hallsteinsen, S.O., Papadopoulos, G.A.: A Utility-Based Adaptivity Model for Mobile Applications. In: AINA Workshops (2). (2007)
14. Alia, M., Horn, G., Eliassen, F., Khan, M.U., Fricke, R., Reichle, R.: A component-based planning framework for adaptive systems. In: 8th International Symposium on Distributed Objects and Applications (DOA), Springer Verlag (2006)
15. Bottaro, A., Gerodolle, A.: Home SOA - Facing Protocol Heterogeneity in Pervasive Applications. In: 5th IEEE International Conference on Pervasive Services (ICPS'08), Sorrento, Italy, July 2008. (2008)
16. Helal, S.: Standards for Service Discovery and Delivery. IEEE Pervasive Computing **1**(3) (2002) 95–100
17. ISO: ITU Open Distributed Processing – Reference Model, International Standard ISO/IEC 10746-1– 4 ITU–T Recommendation X.901– X.904 (1996)
18. Lohse, M., Repplinger, M., Slusallek, P.: Dynamic Media Routing in Multi-User Home Entertainment Systems. In: Proceedings of The Eleventh International Conference on Distributed Multimedia Systems (DMS), Knowledge Systems Institute (2005) 271–276
19. Layaïda, O., Ben Atallah, S., Hagimont, D.: A framework for dynamically configurable and reconfigurable network-based media adaptations. In Journal Of Internet Technology, Special Issue on Real Time Adaptive Media Delivery over the Internet **5**(4) (2004)
20. Eide, V.S.W., Granmo, O.C., Eliassen, F., Michaelsen, J.A.: Real-time video content analysis: QoS-aware application composition and parallel processing. TOMCCAP **2**(2) (2006) 149–172
21. OSGi Alliance: Osgi service platform core specification release 4, http://www.osgi.org (October 2005)
22. IETF RFC 4117: Transcoding Services Invocation in the Session Initiation Protocol (SIP) Using Third Party Call Control (3pcc)) (June 2005)
23. Bottaro, A., Hall, R.S.: Dynamic Contextual Service Ranking. In: 6th International Symposium on Software Composition (SC 2007), Braga, Portugal. (2007)
24. Coulson, G., Blair, G., Grace, P., Joolia, A., Lee, K., Ueyama, J.: A component model for building systems software. In: Proceedings of IASTED Software Engineering and Applications (SEA'04), Cambridge MA, USA. (2004)
25. Bruneton, E., Coupaye, T., Leclercq, M., Quema, V., Stefani, J.B.: The FRACTAL component model and its support in Java: Experiences with Auto-adaptive and Reconfigurable Systems. Softw. Pract. Exper. **36**(1112) (2006) 1257–1284
26. Grace, P., Blair, G.S., Samuel, S.: ReMMoC, A Reflective Middleware to Support Mobile Client Interoperability. In: Proceedings of International Symposium on Distributed Objects and Applications. (2003)
27. Lai, B., Hanrahan, H.: Design of a TINA based Stream Management/Binding Framework. In: SATCAM. (2000)
28. Rafaelsen, H.O., Eliassen, F.: Trading and negotiating stream bindings. In: Middleware '00: IFIP/ACM International Conference on Distributed systems platforms, Secaucus, NJ, USA, Springer-Verlag New York, Inc. (2000) 289–307
29. Fitzpatrick, T., Gallop, J.J., Blair, G.S., Cooper, C., Coulson, G., Duce, D.A., Johnson, I.J.: Design and Application of TOAST: An Adaptive Distributed Multimedia Middleware Platform. In: Proceedings of the International Workshop on Interactive Distributed Multimedia Systems (IDMS). (2001)
30. Vilei, A., Convertino, G., Crudo, F.: A new UPnP architecture for distributed video voice over IP. In: MUM '06: Proceedings of the 5th international conference on Mobile and ubiquitous multimedia, New York, NY, USA, ACM (2006) 2