UPnP Control Point for Mobile Phones in Residential Networks

Andreas Häber[1], Frank Reichert[2], and Andreas Fasbender[3]

***Abstract*— Together, Ericsson and HiA are studying the role of WiFi-enabled mobile phones in residential networks. This article describes a first step to allow mobile phones to discover and control networked residential media servers and renderers following the standards developed by the UPnP Forum.**

***Index Terms*—Residential Networks, Home appliances, Mobile communication, Wireless LAN, UPnP, Networked AV, J2ME**

I. Introduction

A. Background

Personal media such as music, photos, and videos are now mostly handled in digital form. The latest home appliances follow Universal Plug & Play (UPnP) and Digital Living Network Alliance (DLNA) [1-4] recommendations to offer users a convenient and easy way to manage and play their content. UPnP/DLNA certified media servers [8,9] on home PCs or networked attached storage appliances allow users to enjoy their content at any place in their homes.

In 2010 we expect Wi-Fi [10] in mobile phones to be as common as Bluetooth and IrDA technologies are today. Mobile phones will easily link to network based services with high bandwidth at very low or no cost.

Both trends together will create a wide range of new opportunities for businesses to create new attractive services for residential users.

B. Motivation

We assume that users will appreciate the possibility to use their most personal device as a tool to control and consume networked AV service services from sources within and outside the home.

UPnP is the leading standard for self-configuring, residential AV services. We use it as a starting point to investigate the feasibility of XML Web Services technology on mobile phones and to create a solid test platform for future research, e.g., on user acceptance, bootstrapping of residential device ownership, security, trust and privacy, context reasoning, combinations of local and remote identity management, linking of residential communication protocols with upcoming, external networking standards such as the IP Multimedia Subsystem (IMS) [5] by 3GPP [12]

[1] A. Häber, Agder University College, Norway, andreas.haber@hia.no
[2] F. Reichert, Agder University College, NO, frank.reichert@hia.no
[3] A. Fasbender, Ericsson, Germany, andreas.fasbender@ericsson.com

C. Contributions

In this paper we evaluate how our UPnP solutions perform on current mobile phones with Wi-Fi support, and describe the following aspects:

- UPnP architecture (section II).
- Present the UPnP AV architecture (section III).
- Phone based control of media servers and renderers (section IV).
- Evaluation of performance (section V).
- Conclusions (section VI).
- Future work (section VII).

II. Universal Plug & Play

The UPnP [2] architecture is an initiative by the UPnP Forum for i.a. service discovery and control. It is divided into the six phases (0-5) described below. At a high level the architecture describes the interaction of UPnP devices and control points.

A UPnP device consists of one or more services and zero or more embedded devices. The UPnP services contain *state variables* (properties) and *actions* (methods). When a state variable changes it may send an *event* to let control points know about the change of state.

A. Phase 0: Addressing

UPnP uses Internet Protocol (IP)-addresses for addressing, and can obtain addresses either through Dynamic Host Configuration Protocol (DHCP), which is preferred, or using Auto-IP. Both protocols avoid any user interaction for obtaining an address.

B. Phase 1: Discovery

In this phase a control point becomes aware of the devices and services available. The Simple Service Discovery Protocol (SSDP) is used at this phase, which is an extension to the Hypertext Transfer Protocol (HTTP). This phase is two-part: either the control point can explicitly ask devices to respond to a search request, or it awaits status messages that all devices and services regularly send out to inform about their current status.

This phase uses multicasting when sending the necessary requests and service announcements.

C. Phase 2: Description

All devices and services are described in a document using the XML derived UPnP Template Language (UTL).

These description documents can be downloaded using HTTP and parsed by the control point to find out exactly how to control that device. Information such as manufacturer and a "friendly name" to display to users are included here. Most important for a control point is to check which services a device supports, and which actions a service can perform.

The last three phases are not dependent on each other, and can be invoked simultaneously.

D. *Phase 3: Control*

A control point controls devices using the actions of its services. Action invocation is done using SOAP protocol.

E. *Phase 4: Eventing*

A control point receives information about critical state changes and updated through notifications based on the UPnP General Event Notification Architecture (GENA).

The control point must first subscribe to events before it can receive events.

F. *Phase 5: Presentation*

Optionally a device may provide a user interface for viewing and/or controlling the status of the device. The user interface is described with the Hypertext Markup Language (HTML).

III. UPnP AV Architecture

The UPnP AV Architecture is based on the general UPnP Device Architecture as explained above. It defines two basic devices called media server and renderer, with related services.

In addition it includes a control point to control these devices. A key goal of the architecture is to let AV content flow between the server and the renderer directly.

A logical diagram of this architecture can be seen in Fig. 1, followed by a short description of the devices and services of the architecture.

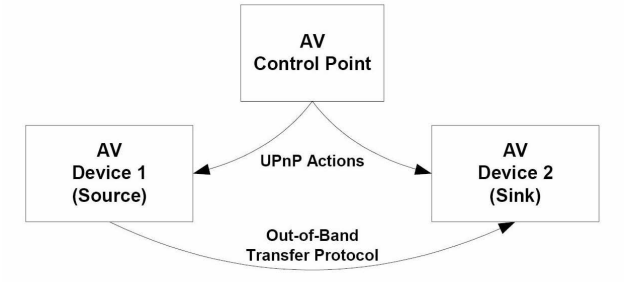

Fig. 1. Logical diagram of UPnP AV Architecture [1].

Notice that it is possible to combine a device type with a control point, e.g., TV may both render media and control a media server.

A. *Media Server*

The media server provides access to containers of media items. Potential media server devices are devices which produce and/or stores media, such as desktop PCs and digital cameras.

It consists of the following services:

1) *Content Directory:* Provides the core functionality of the media server, like browsing and searching for media.

2) *Connection Manager:* Used to manage the connections associated with a device. Through the Connection Manager Service a control point can get access to the device's AV Transport Service.

3) *AV Transport (Optional):* Allows a control point to control the media stream with actions such as play and stop.

B. *Media Renderer*

The media renderer renders media as delivered to it by a media server.

This device includes the following services:

1) *Rendering Control:* This service let the control point control properties of the renderer device such as brightness and contrast of a screen, audio volume, etc.

2) *Connection Manager:* See description above.

3) *AV Transport (Optional):* See description above.

C. *Control Point*

The control point discovers the devices and uses the services they expose, as explained in sections II.D and II.E. A control point can e.g. set the media renderer to play media found on a specific media server.

IV. Design of ONE Portable Player

A. *Application design*

The ONE project by Ericsson and HiA realizes a Portable Player in its initial phase. We create a Java 2 Platform, Micro Edition (J2ME) application, based on the Connected Limited Device Configuration (CLDC) 1.1 and the Mobile Information Device Profile (MIDP) 2.0 specifications. This is commonly referred to as a MIDlet. As a first step we implemented a UPnP AV control point, as described in section III above.

The application is composed of six components, as depicted in Fig. 2.

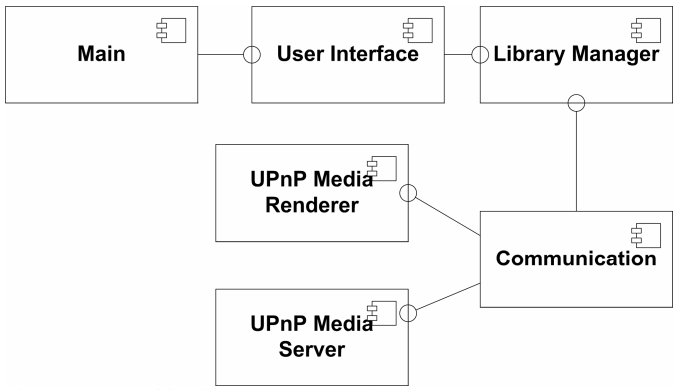

Fig. 2 ONE Portable Player Component Diagram.

Following is a brief description of each of these components.

1) *Main:* The Main component is the main entry point for the application, and responsible for starting the other components.

It also keeps control over settings for the application, especially settings concerned with the user interface. The

settings are event-driven, which means that listeners for such events will be notified whenever a setting is changed. This makes it easy for the other components to update accordingly to the new setting.

The Main block issues UPnP control point commands based on events received from the user interface part.

2) User Interface: The User Interface handles all interaction with the user. It is described in more details later in this paper.

3) Library Manager: The intention of the Library Manager is to create an abstraction to the User Interface, so the User Interface can handle local and remote media in the same manner. (Based on our evaluation, see section V, this design will be changed later).

4) *Communication:* Takes care of all low level communication tasks, e.g., attaching to access networks, sending & receiving IP multicast packets, and more.

5) *UPnP Media Server:* Manages a media server, and makes it easy for the rest of the application to interact with the server, such as browsing the content directory of the server.

6) *UPnP Media Renderer:* Manages a media renderer, and provides an abstraction for the other components to interact with the renderer.

B. Example of Component Interaction

Here we give a short description of how the components interact together and communicate with a remote entity.

All of the actions by the application are results of user interaction. One such action is to browse the media server for music to listen to, which is done whenever the user accesses a container of the *Browse media* menu.

As described above, the control point must first discover the other devices before it can start to control them. ONE Portable Player lets the user do this when necessary, and after one or more devices have been discovered the user must select which one to use. To control the devices UPnP requests are used. An example of a request for browsing the media server's content directory is shown in Fig. 3.

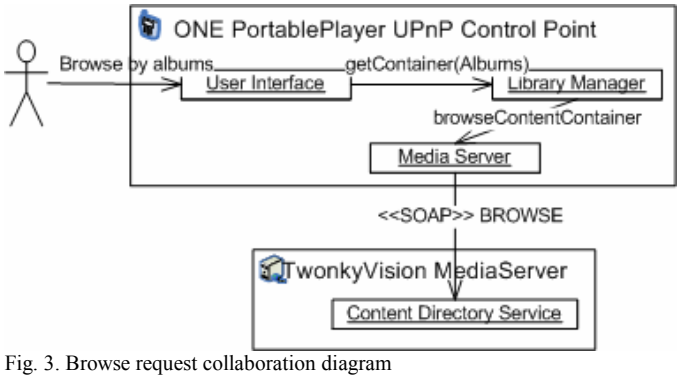

Fig. 3. Browse request collaboration diagram

C. Draft User Interface

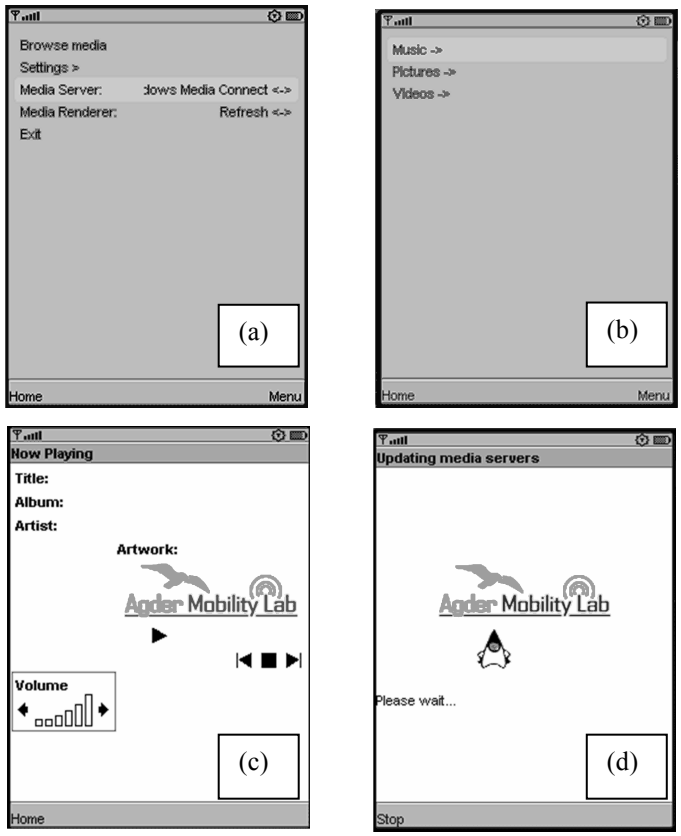

Fig. 4. Early Screenshots of the user interface in the prototyping phase.

The application's user interaction was designed to accommodate for a consistent way of interaction, e.g., local and remote media sources and renderers are handled in the same way. An illustration of our vision is in the figure below that we took from a storyboard to visualize certain concepts.

Users shall be able to select media from anywhere, even their phones. Examples are photos taken by a built-in camera that shall be shown on a TV screen. User shall be able to render media on any device, including their phones.

The design recognizes the fact that media servers have their own way of organizing media, and may provide media types unknown to phones but that media may be playable through other residential media renderers, e.g., using Apple Quicktime encoded media.

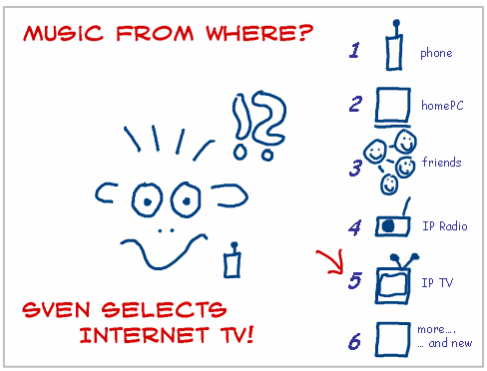

Fig. 5 Example of storyboarding work for source selection.

V. Evaluation

A. Performance

UPnP uses large XML documents and we were concerned that our test device would not be able to parse all the XML documents fast enough. If parsing is slow it will degrade the user experience seriously.

Our implementation runs on different types of cellular PDA phones: a Qtek 9100 with built-in WiFi, and a HP 6515 with a Socket SDIO WLAN card. We plan to use a Sony Ericsson P990 Symbian Smartphone for future performance tests. The media server is based on a Maxtor network attached storage running Twonkyvison Media Server 2.91 [8]. We run a IBM J9 Java Virtual Machine for J2ME that replace the JVM supplied with Windows Mobile 5.0 [13] (more details see subsection D below).

Our biggest concern was the media information from the media server. Each media element is described in XML, so a list of songs from a music album can become a quite large XML document to parse.

To mitigate this concern we planned to add a client side cache of the media information in the library manager component. This would however incur an increased amount of memory usage, which is not recommendable either.

But our initial testing showed that this will not be a problem. Downloading and parsing the media information of a music album is quite seamless for the user. In fact it is quick enough that we have decided to not cache the media information.

B. Picky UPnP Entities

During the preparation of our testing we tried different devices, with varying success. Some devices "choke" when a request does not completely comply with the request format the entity expects, even though the request message is valid and complies with the relevant standards. Some devices give little or no indication what was wrong with the request. This makes it a bit harder to ensure that the control point can control all relevant devices, and not only a strict subset of those.

C. Bluetooth

We initially wanted to use Bluetooth devices for testing, by connecting them to the network using the *Personal Area Network (PAN)* profile. Unfortunately there are not many mobile phones which implement support for that Bluetooth Profile. And more unfortunate is it that some sites use the term *Private Area Network (PAN)* to refer to the Bluetooth piconets established for synchronization of devices etc. This makes it very confusing to know if the device really can support IP over Bluetooth or not.

D. Java Virtual Machines

The mobile phone we use for testing currently, a Qtek 9100, ships with Tao intent JVM. Unfortunately the library included does not implement support for the *optional* MIDP 2.0 UDPDatagramConnection class, which is necessary for UDP communication required by UPnP, obviously. Instead we are now using IBM Websphere Everyplace Micro Environment, with the IBM J9 JVM., which does include support for UDP communication.

The differences between the development emulators and the real implementations made it difficult to know, without thorough testing, which devices a MIDlet can execute successfully.

E. Multicast Support

MIDP does not support multicast listening, which is required by UPnP. We could work around this challenge as most functions work by only sending out multicasts. We will later try to add support using native code for proper multicasting, for devices with multicasting support.

VI. Conclusion

We are very satisfied with the performance of the application, and will therefore continue to develop it with more features. Some implementation challenges will be hopefully resolved by upcoming J2ME features.

VII. Future work

The ONE Portable Player is a step towards more general research challenges combining networked AV and phone based control, e.g., to address security, remote access to homes, and SIP interoperation.

A. Stage 2 – Advanced Phone based Media Server & Renderer

1) Local Media Renderer: Mobile phones come with increasing support for playing audiovisual media, and this feature will take advantage of their latest capabilities and transform the device into a more flexible mobile media renderer. Both phone based and remote UPnP control points will be supported.

2) *Local Media Server:* This feature allows users to make personal media stored on their phones, such as music and pictures, available for internal and external media renderers, as well as to other mobile phones.

3) *Redesign user interface:* We will continuously improve the user interaction and overall concept based on feedback from user trials.

A diagram for how the application will work at the next stage is shown in Fig. 6.

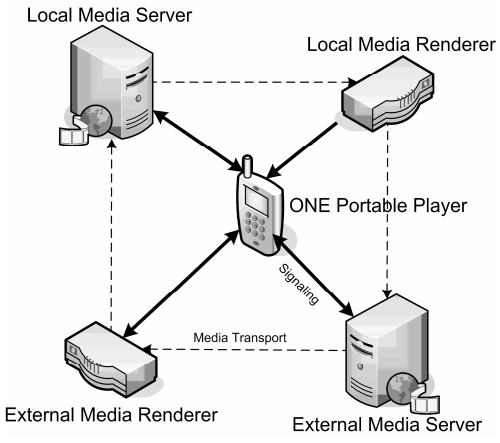

Fig. 6. Logical diagram for stage 2 of ONE Portable Player. *Local* means that the device is hosted at the mobile phone and external is a device external to the mobile phone.

B. Stage 3

Currently most communication is based on UPnP procedures. A more flexible approach must take SIP and its central importance for real-time end-2-end communications into account.

We intend to let users treat incoming SIP calls from the external network in principle like any AV media stream, e.g., to display a video conference on a big TV screen, instead of the small mobile phone screen. The study would investigate whether a phone could acts bridge and proxy between UPnP and SIP communications. This is depicted in Fig. 7.

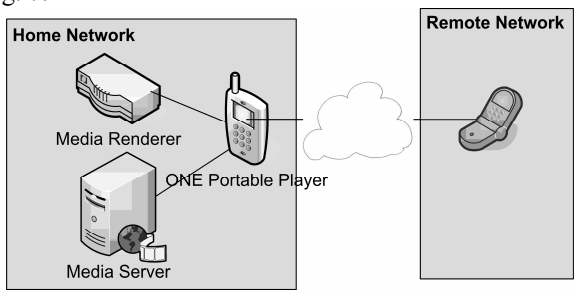

Fig. 7. Logical diagram for stage 3 of ONE Portable Player.

In addition to these features, we will start to investigate security for resource awareness in a residential context, and add the necessary security features to the application. UPnP and other standardization fora have started to investigate this topic.

REFERENCES

[1] UPnP Forum. Available: http://www.upnp.org/.

[2] Digital Living Network Alliance (DLNA). Available: http://www.dlna.org/.

[3] M. Jeronimo and J. Weast, *UPnP design by example : a software developer's guide to universal plug and play*. Hillsboro, OR: Intel Press, 2003.

[4] B. A. Miller, T. Nixon, C. Tai and M. D. Wood, "Home networking with Universal Plug and Play," *IEEE Commun. Mag.*, vol. 39, pp. 104-109, Dec. 2001.

[5] G. Camarillo and M. Garcia-Martin, *The 3G IP Multimedia Subsystem (IMS) : Merging the Internet and the Cellular Worlds*. Chichester: John Wiley and Sons, 2004.

[6] H. Schulzrinne, W. Xiaotao, S. Sidiroglou and S. Berger, "Ubiquitous computing in home networks," *IEEE Commun. Mag.*, vol. 41, pp. 128- 135, Nov. 2003.

[7] S. Patil, and J. Lai, "Who Gets to Know What When: Configuring Privacy Permissions in an Awareness Application," in *Proc. SIGCHI'05,* 2005, pp. 101-110.

[8] Twonkyvision Media Server, by Twonkyvision GmbH. Available: http://www.twonkyvision.de/

[9] Windows Media Connect, by Microsoft. Available: http://www.microsoft.com/windows/windowsmedia/devices/wmconnect/default.aspx .

[10] World Wide Web Consortium (W3C). Available: http://www.w3c.org/.

[11] Third Generation Partnership Project (3GPP). Available: http://www.3gpp.org/

[12] Java Community Process (JCP). Available: http://www.jcp.org/.

[13] J9 Java Virtual Machine by IBM, included in IBM Workplace Client Technology, Micro Edition. Available: http://www-306.ibm.com/software/wireless/wctme/