Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2008, Article ID 835438, 12 pages
doi:10.1155/2008/835438

Research Article

Context-Aware UPnP-AV Services for Adaptive Home Multimedia Systems

Roland Tusch, Michael Jakab, Julius Köpke, Armin Krätschmer, Michael Kropfberger, Sigrid Kuchler, Michael Ofner, Hermann Hellwagner, and Laszlo Böszörmenyi

M3-Systems Research Laboratory, Institute of Information Technology, University of Klagenfurt, 9020 Klagenfurt, Austria

Correspondence should be addressed to Roland Tusch, roland.tusch@m3-systems.com

Received 25 June 2008; Accepted 15 July 2008

Recommended by Harald Kosch

One possibility to provide mobile multimedia in domestic multimedia systems is the use of Universal Plug and Play Audio Visual (UPnP-AV) devices. In a standard UPnP-AV scenario, multimedia content provided by a Media Server device is streamed to Media Renderer devices by the initiation of a Control Point. However, there is no provisioning of context-aware multimedia content customization. This paper presents an enhancement of standard UPnP-AV services for home multimedia environments regarding context awareness. It comes up with context profile definitions, shows how this context information can be queried from the Media Renderers, and illustrates how a Control Point can use this information to tailor a media stream from the Media Server to one or more Media Renderers. Moreover, since a standard Control Point implementation only queries one Media Server at a time, there is no global view on the content of all Media Servers in the UPnP-AV network. This paper also presents an approach of multimedia content integration on the Media Server side that provides fast search for content on the network. Finally, a number of performance measurements show the overhead costs of our enhancements to UPnP-AV in order to achieve the benefits.

1. INTRODUCTION

During the last 25 years, a number of digital audio and video broadcasting standards and systems for large-scale broadcast scenarios have been developed. Some of these standards are *Digital Audio Broadcasting* (DAB) and its successor DAB+ for digital audio transmissions [1], South Korea's DAB-based *Digital Multimedia Broadcasting* (DMB) for broadcasting multimedia data to mobile devices [2], and the complete suite of *Digital Video Broadcasting* (DVB) standards [3, 4]. Similar to DMB, DVB also specified its own transmission system for handheld terminals entitled *DVB-H* [5]. Moreover, the DVB suite also includes a set of Java-based open middle-ware specifications for interactive digital television, called the DVB *Multimedia Home Platform* (DVB-MHP) [6]. DVB-MHP is designed to work across all DVB transmission technologies and requires an additional return channel for each interactive TV application.

However, such broadcast systems are usually not applicable to small-scale environments like single-site home entertainment systems for the following two reasons. First, multimedia broadcasting to mobile devices in a domestic multimedia environment is not economical, since the information coverage area usually is simply too small. Second, in a home multimedia environment, maybe some but not all users are usually interested in the same content at the same time. These reasons result more in the need for multimedia unicasting and multicasting than for multimedia broadcasting in domestic multimedia systems. Therefore, for home multimedia environments, the widely accepted *Universal Plug and Play Audio Visual* (UPnP-AV) [7, 8] standard may be of interest, which is an extension of the original *Universal Plug and Play* (UPnP) [9] standard.

While *UPnP* enables automatic discovery of common devices and services in a local area network, *UPnP-AV* deals with multimedia devices and especially multimedia content. UPnP-AV specifies device and service descriptions for *Media Servers* and *Media Renderers*, which represent multimedia sources and multimedia sinks, respectively. In between these two device classes, a *control point* acts as a dispatcher of multimedia content. Metadata about the available multimedia content is provided to the Control

Point via the Media Server's *Content Directory Service* (CDS). The Control Point queries the *CDS* for the desired content and initiates the playback of the appropriate streams on a Media Renderer, which in turn is responsible for the correct decoding and rendering of the streams.

However, today's *UPnP-AV* implementations have two major drawbacks which make their use difficult in a heterogenous home multimedia environment with several Media Servers and many different Media Renderers. First, standard Control Point implementations only query one Media Server at a time. If there is a larger number of Media Servers in the local area network, there is no global view on the content of all Media Servers in the network. This makes a search for specific content very difficult. Typically, the search is performed by browsing the content directories of all Media Servers. Second, the multimedia content must be consumed by the Media Renderers as provided by the Media Servers. There is no provisioning for customization of the media content to the capabilities of a Media Renderer device. A typical workaround to this problem in most *UPnP-AV* implementations is that if a Media Renderer is not able to render a specific format, the rendering of the stream can not be initiated at the Control Point.

This paper addresses these two drawbacks of today's *UPnP-AV* implementations. In Section 2, the notion of *Usage Context* for customizing multimedia content to different profiles like user and device profiles is introduced. Section 3 presents our *Integrating Media Server* which integrates multimedia metadata from all available Media Servers in the local network. Our extensible *Context-aware Media Renderer* is presented in Section 4. In Section 5, the internal behavior of our *Context-aware Control Point* is described by an example of a control and data flow. Section 6 comes up with performance evaluations of the *Integrating Media Server* and *Context-aware Media Renderer* implementations. Finally, Section 7 concludes the contribution of this paper to context-aware provisioning of mobile multimedia in domestic multimedia systems.

2. THE NOTION OF USAGE CONTEXT

Customization of multimedia content in multicasting or broadcasting systems is not an easy task, since multicasting implies that the delivered data is to be consumed by a number of consumers simultaneously, whereas personalization is rather a powerful content adaptation method for unicast content delivery scenarios. The usual goal of personalization is to deliver a customized version of multimedia content for exactly one consumer. However, there are already approaches of multimedia personalization in large-scale environments like 3DTV and terrestrial DMB (T-DMB), which are mainly based on multiview video and multichannel audio broadcasting techniques [10, 11]. Currently, these approaches are limited to adapting the content according to specific user profiles (i.e., the language, age, or interests of the consumers).

However, in mobile multimedia systems, customization is not limited to the user profile only. besides, the user profile, there are several other profiles which may also require an adaptation of the multimedia content. Especially in environments with a number of different mobile devices, additional constraints to the content delivery are imposed, for example, by the *terminal* and *location profiles* [12]. The terminal capabilities and the current location of the consumer device play an important role in distributed mobile multimedia systems.

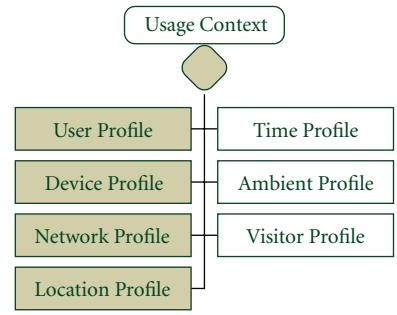


Figure 1: General usage context model for our context-aware multimedia services.

To cope with all relevant profiles for multimedia content tailoring, we introduced the notion of *Usage Context*. The following two subsections provide an overview of our usage context profiles, and of the three possibilities to add context awareness to *UPnP-AV*.

2.1. Usage context profiles

Basically, a universal set of context profiles that is valid for all application areas does not exist, since context is always an issue of the interaction between a user and an application [13]. For example, in a small-scale *UPnP-AV*-based home multimedia environment, it may not be relevant which content the user consumed at what time. However, in a large-scale multimedia tour guide environment as described in [14] this information is definitively of interest to the system provider, since the content consumption history affects possible content demand in the future.

In general, there are three aspects of context which are valid for all application areas: (i) where you are, (ii) who you are with, and (iii) what resources are nearby [15]. For multimedia applications, these aspects have already been addressed in the *MPEG-21 Digital Item Adaptation* standard by the means of *Usage Environment Descriptions* [16–18]. For ubiquitous mobile devices like mobile phones, the *User Agent Profile* (UAProf) [19] has been developed as a de facto standard for describing the resource aspect (i.e., the Device Profile). Moreover, UAProf also describes the preferences aspect, since it is based on the *Composite Capability/Preference Profiles* (CC/PP) [20] vocabulary extension of the *Resource Description Framework* (RDF) [21].

However, neither the MPEG-21 DIA's Usage Environment nor the CC/PP-based UAProf profiles contain sufficient information regarding the context profiles needed by our context-aware application domains, including the context-aware UPnP-AV services and the context-aware large-scale tour guide application [14]. Thus, we derived an own

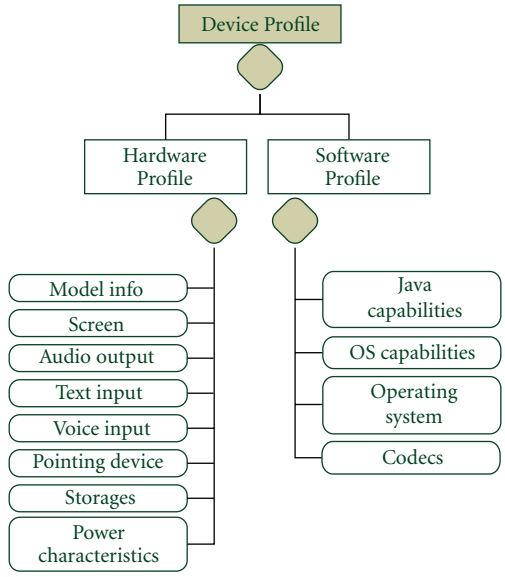


Figure 2: The Device Profile as a composition of Hardware and Software profiles.

context model for our context-aware multimedia services. An overview of this *XML Schema*-based model is given in Figure 1.

Basically, this context model includes the four profiles *User*, *Device*, *Network*, and *Location Profile*. These profiles are required for all of our application domains and represent the four profiles needed by the context-aware UPnP-AV services. The *User Profile* collects data about the characteristics of the user like preferred language, age, interests, and presentation preferences (like audio only and video only for handicapped persons, or mixed). The *Device Profile* delivers information about the hardware and software properties of the consumer device (see Figure 2). For context-aware UPnP-AV services, the *Device Profile* is one of the most important profiles for content adaptation.

The *Network Profile* gathers data about available networks including average bit rate, reliability, latency, and transmission costs. And the *Location Profile* provides information about the current location of the consumer device, according to the used localization technique [22]. For context-aware UPnP-AV services in domestic multimedia environments, localization techniques using Bluetooth and/or Radio Frequency ID (RFID) technology are suitable.

In addition to these four basic profiles, three further profiles (*Time*, *Ambient*, and *Visitor Profiles*) are required for our large-scale multimedia tour guide application. They are not used by the context-aware UPnP-AV services, but for the sake of application-domain comparison they are mentioned here. The *Time Profile* tracks the current system time of the content consumption. Current environmental conditions including weather, temperature, and air conditions are collected by the *Ambient Profile*. And finally, the *Visitor Profile* tracks the content consumption history of a tourist, as well as the data about her/his vacation, such as location, date of arrival, date of departure, and number of persons.

2.2. Adding context awareness to UPnP-AV

As mentioned in Section 1, one major drawback of the UPnP-AV specification is that the multimedia content must be consumed by the Media Renderers as provided by the Media Servers. The only adaptation step existing Control Point implementations typically perform is to avoid the rendering of a Media Server's media stream on a Media Renderer if the renderer is not able to deal with the coding format of the stream or if the renderer does not support any provided transport protocol of the Media Server.

To overcome this drawback, two improvements can be added to UPnP-AV services. First, the Media Renderer can be extended with means for querying the current usage context (see previous Section 2.1) on the renderer device. Second, the Control Point may incorporate a content transcoding application, which adapts a Media Server's media stream to a given usage context, before the content is delivered to the Media Renderer.

Basically, there are three possibilities to enhance a standard UPnP-AV Media Renderer with context provisioning [23]. First, a selected service of the Media Renderer can be extended by an additional action which returns the current context information. An *enhanced* Control Point may then call this action to acquire the desired information. Second, a selected service of the Media Renderer can be extended with a set of state variables describing the context. A Control Point may query these variables to acquire the context via UPnP-AV's eventing mechanism. And third, the Media Renderer can be extended with a new service which encapsulates the aforementioned variables, provides actions to query them and offers eventing for the notification of value changes to these variables.

All three possibilities have their advantages and disadvantages. Our decision to extend the Media Renderer's *Connection Manager* service by an additional *GetContextInfo* action is described in Section 4. The Control Point's usage of the context data for customizing a media stream is presented in Section 5.

3. THE INTEGRATING MEDIA SERVER

To overcome the first drawback of standard UPnP-AV mentioned in Section 1, an *Integrating UPnP-AV Media Server* was implemented [24]. Figure 3 illustrates the steps undertaken to integrate multimedia metadata from *n* UPnP-AV Media Servers in the network.

When a Media Server starts up, it first fills its own *Content Directory* with metadata obtained from locally available multimedia sources (i.e., files from the file system or live sources). The gathered metadata is then provided to a Control Point via the Media Server's *Content Directory Service* (CDS). In the next step, the *Integrating Media Server* browses the CDS of each detected Media Server, and integrates its metadata into its own CDS. In order to be able to connect to other Media Servers, the Integrating Media Server implements its own Control Point, which listens to arrivals and departures of Media Servers in the UPnP-AV network, and performs metadata integration or segregation steps, respectively.

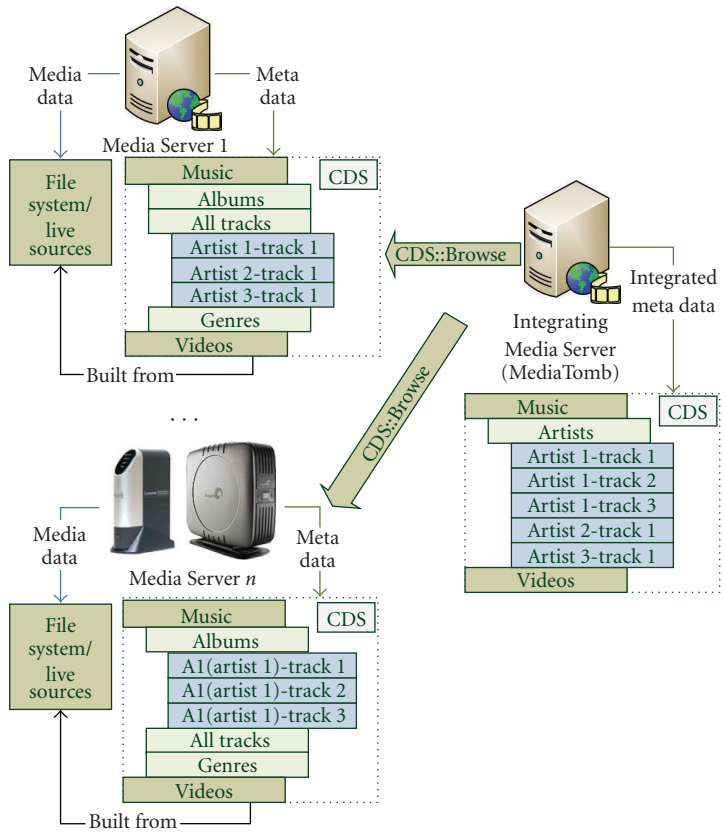


Figure 3: Metadata integration from UPnP-AV Media Servers.

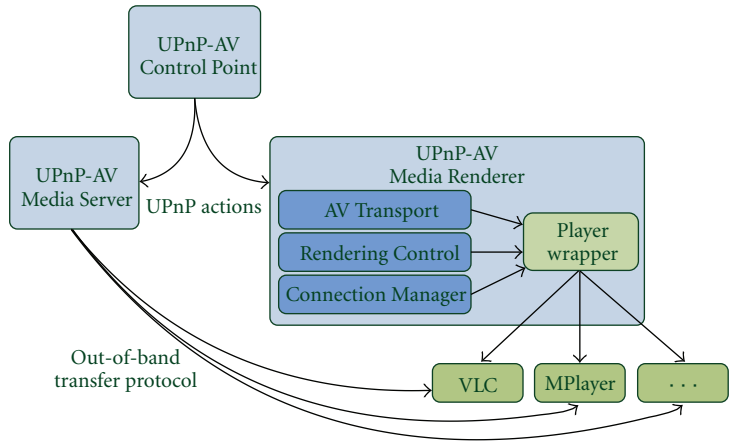


Figure 4: Conceptual view on the generic UPnP-AV Media Renderer architecture.

Integrating metadata views from a remote Media Server does not simply mean an exact import of them to the local CDS. While the latter approach is also known as *metadata mirroring* [25], *metadata integration* enhances metadata mirroring by reorganizing the mirrored metadata into one unified view [24]. For example, in Figure 3 the metadata view *All Tracks* on Media Server 1 and the metadata view *Albums* on Media Server n are integrated to a metadata view *Artists* on the Integrating Media Server.

The multimedia data itself remains on the origin Media Servers and is referenced as a resource in *DIDL-Lite*-based media item descriptions. DIDL-Lite [26] is a subset of MPEG-21's *Digital Item Declaration Language* (DIDL) [27] used in UPnP-AV. It is also based on XML as DIDL, but its schema restricts the possible metadata fields to the UPnP and *Dublin Core* [28] namespaces. Algorithm 1 shows an example DIDL-Lite response of Media Server 1 to a *Browse* action call of the Integrating Media Server's Control Point.

```
<DIDL-Lite>
  <container id="100" parentID="10"
      childCount="3" restricted="1">
    <dc:title>All Tracks</dc:title>
    <upnp:class>
      object.container.musicContainer
    </upnp:class>
  </container>
  <item id="101" parentID="100" restricted="1">
    <dc:title>Dancing Queen</dc:title>
    <upnp:artist>Abba</upnp:artist>
    <upnp:album>Arrival</upnp:album>
    <upnp:genre>Pop</upnp:genre>
    <res size="3482846" duration="0:03:52.110"
      protocolInfo="http-get:*:audio/mpeg:*" >
      http://192.168.1.5:9001/disk/101.mp3</res>
    <upnp:class>
      object.item.audioItem.musicTrack
    </upnp:class>
  </item>
</DIDL-Lite>
```

Algorithm 1: An example media item description.

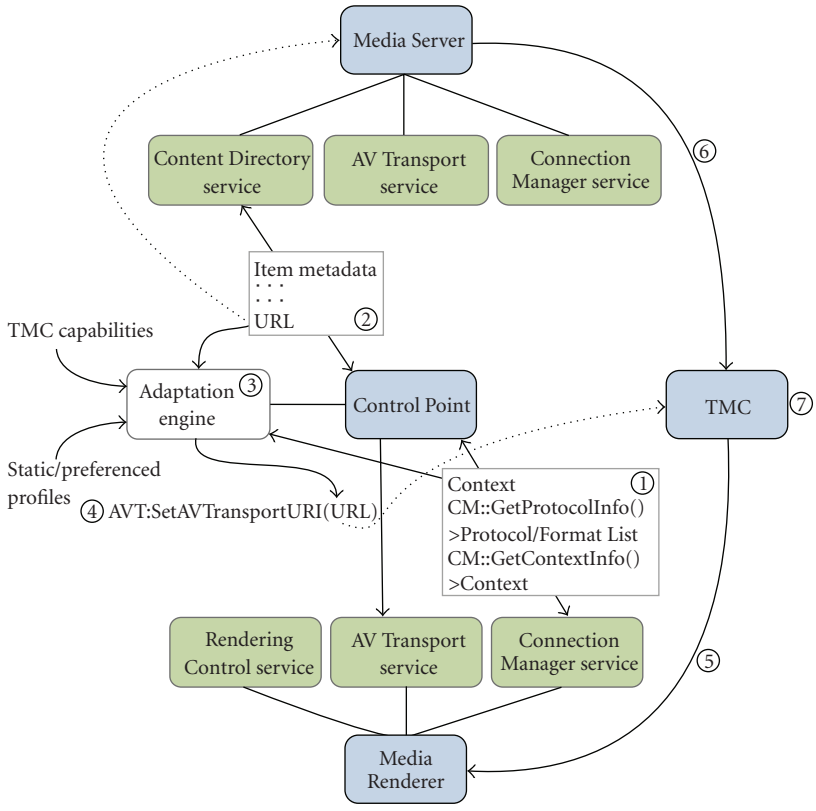


Figure 5: Action call sequence of the Context-aware UPnP-AV Control Point.

The *res* element provides information about the media item's encoding properties, as well as the URL to use for requesting the media item.

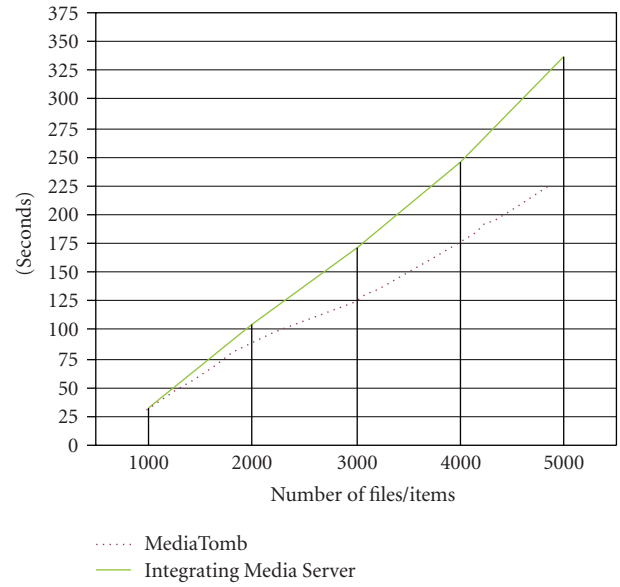
For the implementation of the Integrating Media Server, the open source UPnP-AV Media Server *MediaTomb* [29] and the open source UPnP-AV stack *libupnp* [30] were used.

The presented approach of metadata integration brings two major advancements for Control Points compared to standard UPnP-AV. First, the Integrating Media Server can easily implement the optional *Search* action to enable a Control Point to query the integrated view of metadata for certain multimedia content, even though some Media Servers do not implement this action themselves. And second, the integration step allows to reorganize the metadata in customizable views to customize the multimedia content provisioning according to the profiles of the Usage Context.

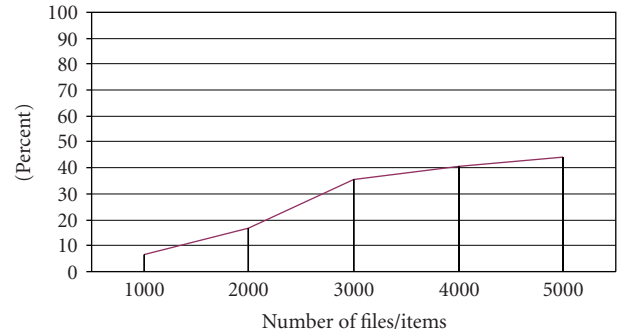
4. THE CONTEXT-AWARE MEDIA RENDERER

The architecture of our context-aware UPnP-AV Media Renderer is kept generic in order to be able to use any existing UPnP or non-UPnP enabled media player as media renderer [31]. Figure 4 provides a conceptual view on this architecture. Existing media players like the *MPlayer* [32] or the *VLC* [33] can be used by our Media Renderer to playback media streams from a UPnP-AV Media Server. While the VLC itself is already UPnP-AV enabled, the standard MPlayer currently does not have support for UPnP-AV. Neither of them is context-aware by default. Our generic Media Renderer architecture allows to integrate the binary version of any supported media player and hence turns it into UPnP-AV enabled. This is achieved by a *Player Wrapper* inside the Media Renderer, which delegates selected action requests to the Media Renderer's *AV Transport*, *Rendering Control*, and *Connection Manager* services to a concrete player instance. In our prototype implementation of the Media Renderer, the MPlayer has been chosen as media player, and an *MPlayer Wrapper* delegates all requests to a running MPlayer instance. The wrapper is also responsible for returning all results of the MPlayer instance to the Media Renderer's calling services.

The ability to publish its context is added to the Media Renderer by extending its *Connection Manager* service with an additional *GetContextInfo* action. This approach has the advantages that (i) it is the responsibility of the Control Point to obtain the context from the Media Renderer, and (ii) the Control Point is also able to control the additional traffic overhead generated by context data. If context awareness was instead realized by eventing, the additional generated network load would have depended on the change frequency of the evented context properties. Considering very dynamic context properties such as *Storages* of the *Device Profile*, UPnP-AV's eventing mechanism may generate a large number of events during a Media Renderer session, since the available memory in the local storage subsystem may often change during a session. Although UPnP-AV provides the concept of *deferred eventing* by the use of thresholds, the eventing behavior is not desirable in many cases, since the transmission of each event generates considerable additional load on the network. On the other hand, the *pull-based* approach to receiving context information from the Media Renderer has the drawback that the Control Point has to periodically query the *GetContextInfo* action. But comparing the advantages and the disadvantages of both approaches, using an additional action is the right decision. Section 6.2.2 provides some performance figures about the costs of calling this action on two different renderer devices.



(a) Build times from file system versus integration times



(b) Integration overhead

Figure 6: Overhead of integrating media items.

5. THE CONTEXT-AWARE CONTROL POINT

Besides the Control Point implementation for metadata integration in the Integrating Media Server presented in Section 3, we also developed a full implementation of a *Context-aware Control Point* [23]. This Control Point operates as dispatcher of media streams from a Media Server to available Context-aware Media Renderers. Figure 5 illustrates an action call sequence for initiating the rendering of a media stream on a Context-aware Media Renderer.

First, the Control Point queries the Usage Context from the Connection Manager service as soon as it connects to a new Media Renderer. In step two, it queries the metadata

Table 1: Test stream variations.

Variation	Resolution	Video bit rate	Audio bit rate	Total bit rate
V_1	320 × 240	200 kbps	64 kbps	264 kbps
V_2	352 × 208	464 kbps	64 kbps	528 kbps
V_3	352 × 208	500 kbps	64 kbps	564 kbps
V_4	800 × 480	200 kbps	64 kbps	264 kbps

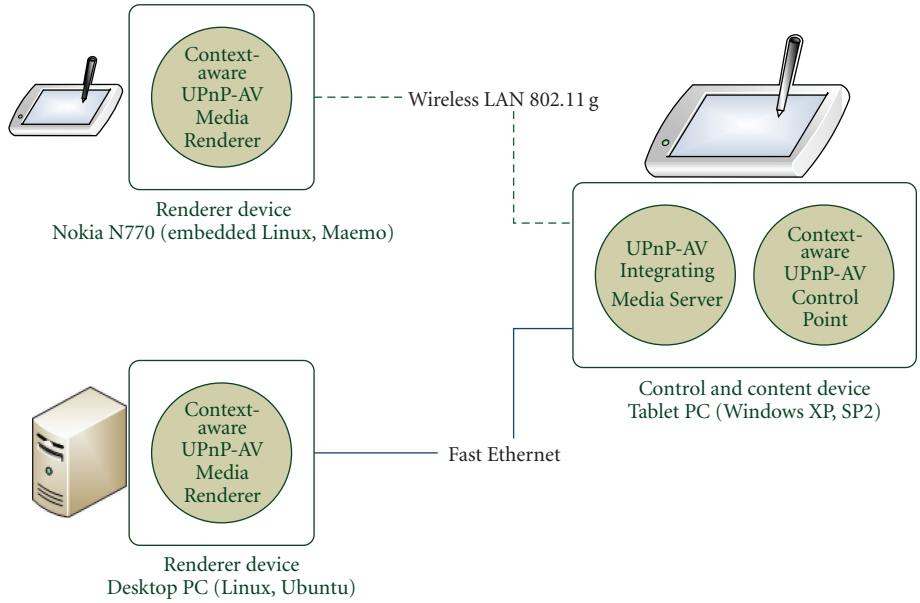


Figure 7: UPnP-AV Media Renderer performance test setup.

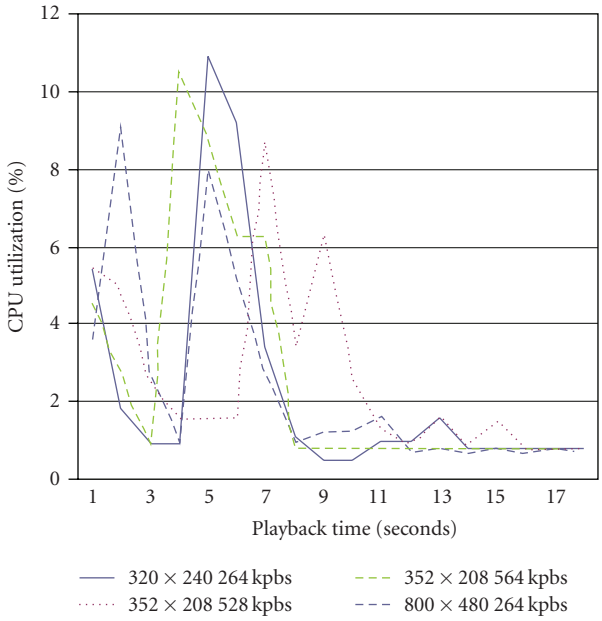
item for the selected media item from the Integrating Media Server. At this point, it has to be mentioned that our Context-aware Control Point provides a Web-based user interface for browsing and searching content. In the example of Figure 5, the Media Renderer selects a media item via this interface. Thus, this example illustrates a pull-based unicast scenario. However, a push-based multicast scenario can also be realized. In the third step, an internal *Adaptation Engine* of the Control Point is used to initiate an adaptation of the media item according to the given Usage Context. This is achieved by incorporating our transcoding service *Transcoding Media Cache* (TMC), which is able to transcode and compose multimedia streams from various input to various output formats [34]. The transcoding of the media item does not take place at this step, it is only initiated by the Adaptation Engine by rewriting the URL in the item's metadata to point to the TMC, and subsequently calling the action *SetAVTransportURI* of the Media Renderers *AV Transport* service with this rewritten URL as parameter (step four). In step five, the Context-aware Media Renderer requests the media item from the TMC, which in turn fetches the original media item from the corresponding Media Server (step six), and transcodes the media item according to the transcoding parameters set by the Adaptation Engine (step seven). Finally, the adapted stream is sent to the Media Renderer by the use of an out-of-band (i.e., non-UPnP) transport protocol. Steps six and seven can be omitted if the TMC already has a cached version of the requested stream for the given Usage Context.

6. PERFORMANCE EVALUATION

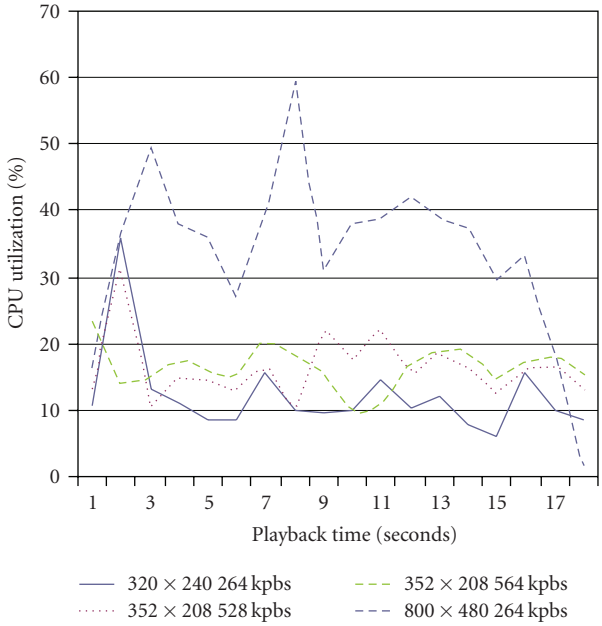
The following two subsections evaluate the performance overhead costs of our approach to extend the UPnP-AV Media Server *MediaTomb* with media item integration capabilities and to implement a UPnP-AV Media Renderer by the use of the non-UPnP-AV-enabled third party media player *MPlayer*, respectively.

6.1. Integrating Media Server

Figures 6(a) and 6(b) illustrate the overhead of the integration of media items in the UPnP-AV Integrating Media Server, compared to the construction of the content directory from a file system as performed by the *MediaTomb* Media Server. The comparison is based on five data sets containing 1000, 2000, 3000, 4000, and 5000 media items from a remote MediaTomb Media Server and the same number of files from a local file system, respectively [24]. As test items, common media items and files from music albums were used. The integration times were measured in an insulated Fast Ethernet LAN.

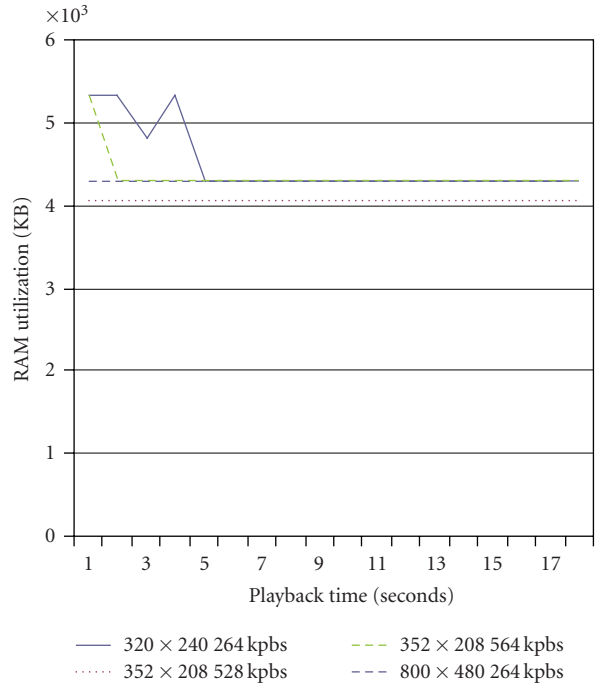


(a) Renderer

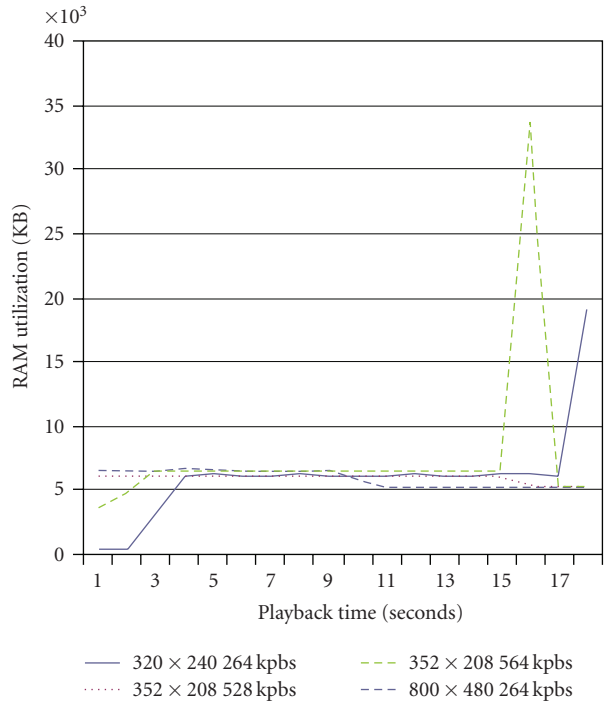


(b) MPlayer

Figure 8: CPU utilization on the Nokia N770.



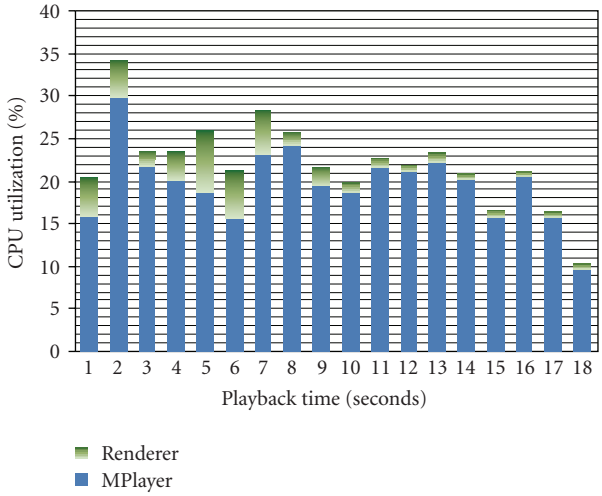
(a) Renderer



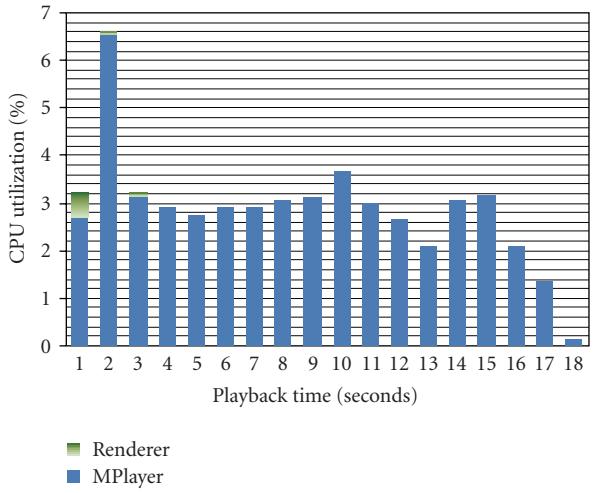
(b) MPlayer

Figure 9: RAM utilization on the Nokia N770.

The integration overhead steadily increases with the number of media items to retrieve from the remote MediaTomb Media Server. While the integration overhead for 1000 items is rather low with about 7%, it increases to about 44% for 5000 items. Although the increase is not strictly linear, it shows a linear gradient of 9.2% for 1000 additional media items on average. This overhead increase is mainly due to the protocol overhead imposed by UPnP-AV, accompanied by the continuous UPnP-AV *Browse* actions, which have to be called on the remote MediaTomb Media Server's *CDS* to query for all available remote media items. This integration overhead is acceptable for an adaptive domestic multimedia system, where the number of media items is seldom higher than 10000 media items (causing an integration overhead of at least 92%) and the frequency of integration activities is rather low.

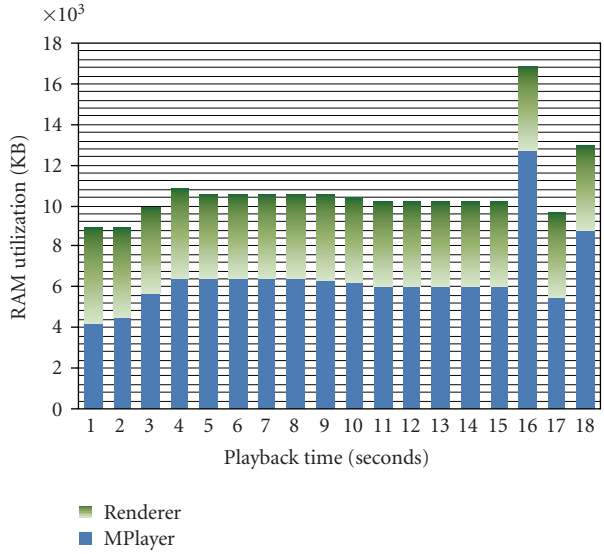


(a) Nokia 770

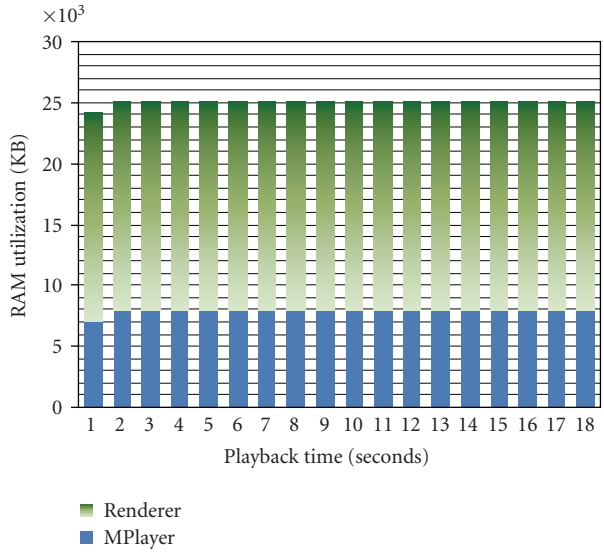


(b) Desktop PC

Figure 10: CPU utilization overhead.



(a) Nokia 770



(b) Desktop PC

Figure 11: RAM utilization overhead.

6.2. Context-aware Media Renderer

6.2.1. CPU and RAM utilization overhead

The CPU and RAM utilization overhead of the context-aware UPnP AV Media Renderer wrapping the MPlayer as player instance was evaluated using the test setup depicted in Figure 7.

On the right side, a Tablet PC running Windows XP (SP2) is used as control and content device. This device runs one instance of the Integrating UPnP-AV Media Server and one instance of the Context-aware UPnP-AV Control Point. In this test case, the media server does not integrate any content from other media servers. Instead, its *CDS* simply offers one system stream (i.e., a composed stream of one video and one audio elementary stream) in four variations regarding bit rate and resolution. Table 1 illustrates the properties of these four stream variations. Note that even though V_4 has the highest resolution, it has the lowest bit rate (actually equal to V_1). Video elementary streams are encoded as MPEG-4 SimpleProfile@Level3, audio streams are commonly encoded as MPEG-1@Layer3. All stream variations have a duration of 18 seconds in playback time. The context awareness of the Control Point is not used in this test case either, since the video variations are already prepared and available to the *CDS* of the Media Server.

On the left side of Figure 7, two renderer devices are used for evaluating the performance of the Context-aware UPnP-AV Media Renderer. The first renderer device is a Nokia N770 Internet Tablet running the Internet Tablet OS 2006 (Maemo 2.2 [35]). It ships with a 252 MHz *Texas Instruments* CPU, 64 MB RAM, 128 MB Flash memory, and a widescreen display with a maximum video resolution of 800 × 480 pixels. The second renderer device is a common Desktop PC running Ubuntu Linux 6.1, with an

Intel Pentium 4 2.53 GHz processor and 128 MB RDRAM installed, providing a maximum video resolution of 1920 × 1200 pixels.

Before discussing the CPU and RAM utilization overhead of the Context-aware Media Renderer, the CPU and RAM utilization of both the Media Renderer and the MPlayer on the Nokia N770 device are illustrated in Figures 8(a)-8(b) and 9(a)-9(b), respectively. Figures 8(a) and 9(a) confirm our expectation that the different stream variations do not have considerable impacts on the CPU and RAM utilization of the Media Renderer, since the Media Renderer is only the *UPnP-AV wrapper* of the MPlayer and hence does not directly operate on the media streams. In contrary, Figures 8(b) and 9(b) clearly show the impacts of the different stream variations on the CPU and RAM utilization of the MPlayer, respectively. While the stream variation with the highest video resolution (V_4) generates the highest load on the CPU, the stream with the highest bit rate (V_3) shows the highest RAM usage peak. Whereas the latter result is not surprising, the former is a bit unexpected since the CPU load of stream variation V_4 is about two times higher than those of the other variations, although it has the smallest bit rate. This is due to the higher computational requirements for larger video resolutions especially during the double buffering and bit-blitting operations.

The CPU and RAM utilization overhead of the Context-aware Media Renderer on both renderer devices is illustrated in Figures 10(a)-10(b) and 11(a)-11(b), respectively. The overhead is calculated on averaged values of CPU and RAM utilizations of all stream variations. Figure 10(a) shows a mean CPU utilization overhead of the Media Renderer of 12.5% on the Nokia N770 device, compared to the average CPU load generated by the MPlayer. On the Desktop PC, this overhead accounts for only 1.3%, as shown in Figure 10(b). Interestingly, the RAM utilization overhead shows a diametrical result. While the RAM utilization overhead of the Media Renderer on the Nokia N770 results in a mean value of 66.8% (see Figure 11(a)), the overhead on the Desktop PC accounts for 220%, as depicted in Figure 11(b). The latter result is due to the used *libupnp* library, which uses more dynamically linked libraries on the Maemo platform.

6.2.2. Response times to context queries

Figure 12 illustrates the response times (in millisecond) of calls to the *GetContextInfo* action in a run of 20 subsequent measurements on both, the Nokia N770 and the Desktop PC renderer devices. It is obvious that the execution of this action is much more expensive than other UPnP-AV actions like *SetAVTransportURI* or *Play*, which take about 10 milliseconds on average. This is due to the dynamic collection of context information each time this action is called. On the Desktop PC the initial call to this action results in a response time of about 1.8 seconds. This is the time the renderer needs on this device when all context information is queried by either directly invoking system calls or by executing shell scripts. However, since some context information is static and does not change during the whole life cycle of the Renderer (like the manufacturer and the device model of the Device Profile), this static context information is only queried once and cached for later action calls. The caching of static context information results in a significant response time reduction of about 80% to about 350 milliseconds.

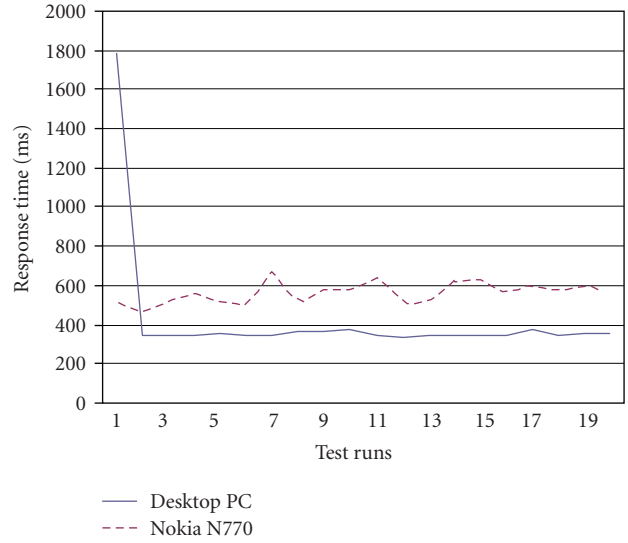


Figure 12: Response times to *GetContextInfo* action calls.

In comparison to the Desktop PC, the Renderer on the Nokia N770 device does not show this kind of *slow start*. Although equipped with a much less powerful CPU and I/O subsystem, the Renderer on the Nokia N770 starts faster than that on the Desktop PC. This is due to the statical assignment of some context properties like screen resolution, text and voice input capabilities in the program code on the Nokia N770. This, of course, results in a much faster collection of the required context data.

7. CONCLUSIONS

This paper presented a novel approach of realizing context-aware multimedia services for domestic multimedia systems by using the *Universal Plug and Play Audio Visual* (UPnP-AV) technology. Since UPnP-AV is designed for local area networks, it is also suitable for multimedia multicasting and broadcasting scenarios. However, standard UPnP-AV does not provide means for tailoring multimedia content to different context properties like the user, device, or network profile. To overcome this drawback, an extension to the Media Renderer has been realized which enables the Control Point to periodically query context information from the Media Renderers. This *Usage Context* information in turn is used by the Control Point to customize media streams from the Media Server by the use of a Transcoding Media Cache (TMC). This approach allows to optimize the multimedia content to the needs of the user with respect to the constraints of her/his usage environment.

The second enhancement of this contribution to standard UPnP-AV is the development of an Integrating Media Server which integrates media items from all other Media

Servers available on the local area network. This integration step provides a global view on all available multimedia content in the UPnP-AV network and allows the Control Point to perform fast queries on the available content. Finally, performance evaluations regarding the overhead of the integration step in the Integrating Media Server, as well as CPU and RAM utilization overheads of the Media Renderer implementation have shown that the overhead costs for achieving the benefits are rather low.

ACKNOWLEDGMENT

This work was supported by the Austrian Science Fund (FWF) under project L92-N13 (CAMUS: Context-Aware Multimedia Services).

REFERENCES

[1] European Telecommunications Standards Institute, "Digital Audio Broadcasting (DAB); Guide to DAB standards; Guidelines and Bibliography," ETSI TR 101 495, January 2005.

[2] European Telecommunications Standards Institute, "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to Mobile, Portable and Fixed Receivers," ETSI EN 300 401, April 2000.

[3] European Telecommunications Standards Institute, "Digital Video Broadcasting (DVB); A Guideline for the Use of DVB Specifications and Standards," ETSI TR 101 200, September 1997.

[4] European Telecommunications Standards Institute, "Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television," ETSI EN 300 744, June 2004.

[5] European Telecommunications Standards Institute, "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)," ETSI EN 302 304, June 2004.

[6] European Telecommunications Standards Institute, "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1," ETSI TS 102 812 V1.2.2, August 2006.

[7] UPnP Implementers Corporation, UPnP AV Architecture:0.83. White Paper, June 2002, http://www.upnp.org/standardizeddcps/documents/UPnPAvArchtiecture0.83.pdf.

[8] Intel R&D, "Overview of UPnP AV Architecture: A Digital Media Distribution Technology for the Home," White Paper, July 2003, http://cache-www.intel.com/cd/00/00/21/87/218764_218764.pdf.

[9] UPnP Implementers Corporation, UPnP Device Architecture 1.0. White Paper, 2006, http://www.upnp-ic.org/resources/UPnP_device_architecture_docs/UPnP-DeviceArchitecture-v1_0-20060720.pdf.

[10] K.-J. Oh, M. Kim, J. S. Yoon, et al., "Multi-view video and multi-channel audio broadcasting system," in *Proceedings of the 3DTV Conference (3DTV-CON '07)*, pp. 1–4, Kos Island, Greece, May 2007.

[11] T. Lee, Y. J. Lee, J. H. Yoo, and D. Jang, "Personalized audio broadcasting system through the terrestrial-DMB system," in *Proceedings of the International Conference on Consumer Electronics (ICCE '07)*, pp. 1–2, Las Vegas, Nev, USA, January 2007.

[12] S. Panagiotakis and A. Alonistioti, "Context-aware composition of mobile services," *IT Professional*, vol. 8, no. 4, pp. 38–43, 2006.

[13] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[14] J. Köpke, R. Tusch, H. Hellwagner, and L. Böszörmenyi, "Context-aware hoarding of multimedia content in a large-scale tour guide scenario: a case study on scaling issues of a multimedia tour guide," in *Proceedings of the International Conference on Signal Processing and Multimedia Applications (SIGMAP '08)*, Porto, Portugal, July 2008.

[15] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 85–90, Santa Cruz, Calif, USA, December 1994.

[16] International Organization for Standardisation, "Information technology—multimedia framework (MPEG-21)—part 7: digital item adaptation," Tech. Rep. ISO/IEC 21000-7, ISO, 2004.

[17] A. Vetro and C. Timmerer, "Digital item adaptation: overview of standardization and research activities," *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 418–426, 2005.

[18] A. Vetro, C. Timmerer, and S. Devillers, "Digital item adapation—tools for universal multimedia access," in *The MPEG-21 Book*, chapter 7, pp. 243–280, John Wiley & Sons, New York, NY, USA, 2006.

[19] OpenMobile Alliance, User Agent Profile - Approved Version 2.0, OMA-TS-UAProf-V2_0-20060206-A, February 2006.

[20] World Wide Web Consortium, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation," 2004, http://www.w3.org/TR/CCPP-struct-vocab/.

[21] World Wide Web Consortium, "Resource Description Framework (RDF): Concepts and Abstract Syntax," W3C Recommendation, 2004, http://www.w3.org/TR/rdf-concepts/.

[22] M. Santner, R. Tusch, M. Kropfberger, L. Böszörmenyi, and H. Hellwagner, "Ein Ortserkennungssystem für mobile Touristenführer," in *Proceedings of the DACH Mobility*, pp. 84–98, Ottobrunn, Germany, October 2006.

[23] M. Ofner, *Design and implementation of a context-aware UPnP-AV control point*, M.S. thesis, Institute of Information Technology, University of Klagenfurt, Klagenfurt, Austria, 2008.

[24] M. Jakab, M. Kropfberger, M. Ofner, R. Tusch, H. Hellwagner, and L. Böszörmenyi, "Metadata integration and media transcoding in universal-plug-and-play (UPnP) enabled networks," in *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP '07)*, pp. 363–372, Naples, Italy, February 2007.

[25] Intel R&D, "Designing a UPnP-AV MediaServer," White Paper, July 2003, http://cache-www.intel.com/cd/00/00/21/87/218762_218762.pdf.

[26] UPnP Implementers Corporation, "ContentDirectory:1 Service Template Version 1.01," White Paper, June 2002, http://www.upnp.org/standardizeddcps/documents/ContentDirectory1.0.pdf.

[27] International Organization for Standardisation, "MPEG-21 part 2: digital item declaration language (DIDL)," Technology Report ISO/IEC 21000-2, ISO, 2003.

[28] Dublin Core Metadata Initiative, "Dublin Core Metadata Element Set, Version 1.1," DCMI Recommendation, June 2008, http://dublincore.org/schemas/xmls/qdc/2008/02/11/dc.xsd.

[29] The MediaTomb Project, MediaTomb. http://mediatomb.cc/.

[30] "PUPnP SourceForge Community. Portable SDK for UPnP Devices (libupnp 1.6.6," SourceForge.net Project, June 2008,

http://pupnp.sourceforge.net/.

[31] S. Kuchler, *An extendable UPnP-AV media renderer*, M.S. thesis, Institute of Information Technology, University of Klagenfurt, Klagenfurt, Austria, August 2007.

[32] The MPlayer Project, MPlayer. http://www.mplayerhq.hu/.

[33] The VideoLAN Project, VLC media player. http://www.videolan.org/vlc/.

[34] M. Kropfberger, R. Tusch, M. Jakab, et al., "A multimedia-based guidance system for various consumer devices," in *Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST '07)*, pp. 83–90, Barcelona, Spain, March 2007.

[35] Maemo Community, Maemo. White Paper, June 2008, http://maemo.org/intro/white_paper/.