Integrative approach of Web Services and Universal Plug and Play within an AV scenario

Brigitte Oesterdieckhoff, Chris Loeser, Isabell Jahnich
Paderborn University
Paderborn, Germany
{brigitte | loeser | bella}@c-lab.de

Rainer Glaschick
Siemens Business Services
Paderborn, Germany
rainer.glaschick@c-lab.de

***Abstract*— The increase of networking capabilities even of small devices leads to a rising demand for distributed applications, for which seamless interoperability between devices and applications is essential. As a basis for the required and emerging middleware, Service Oriented Architectures (SOA) have become the dominant design paradigm. In the field of embedded, mostly home used devices, Universal Plug and Play (UPnP) can be regarded as an early SOA standard, that has gained momentum and is supported by more and more products. A more advanced suite of SOA standards is known as *Web Services* (WS), that were originally developed from the needs for interoperability in commercial communications between enterprises. In the SIRENA funded project, Web Services have been selected as basis for device oriented middleware in several fields like industrial, telecommunications, automotive and home devices. In the latter, the rising availability of UPnP demands an integrative approach for advanced solutions in a distributed audio-video environment with a large number of devices and distributed applications. The paper highlights some UPnP issues and WS characteristics, proposes practical rules for proper interface design of device related Service Oriented Architectures, and finally describes the integrative solution for an advanced media replication service based on Web Services and UPnP devices.**

Keywords: Service Oriented Architecture, SOA, Service Oriented Computing, SOC, Web Services, Universal Plug and Play, UPnP Device Architecture.

I. Introduction

The proliferation of independent computer devices together with the advances in networking support even for small embedded devices, require more and more universal and simple cooperation, to a large extent without user intervention, between devices in a networked environment. Thus several middleware-technologies emerge that support interacting services of networked device.

Service Oriented Computing (SOC) uses a Service Oriented Architecture (SOA) to provide a structure where a relatively simple, clear, and host system independent interfaces for inter-application communication are provided. The now generally accepted SOA standard are Web Services (WS) [13] hosted by the Word Wide Web Consortium (W3C), using extensively other internet and WWW standards, like URIs and URLs for address syntax, XML to describe data, SOAP for message encapsulation, and UDDI for service advertisement and discovery. The Web Services Description Language (WSDL) is a standardized (XML based) method to describe the interfaces in a system independent way. WSDL is targeted towards software generation tools to generate and check interface code for the services.

An early Service Oriented Architecture is Universal Plug and Play (UPnP), defined by the UPnP Forum [1]. It defines communication formats and protocols for zero-configuration networks of small devices like media appliances or computer peripherals. SOAP messages coded in XML and transmitted via HTTP are used, as are in Web Services. The UPnP specification describes device addressing, service advertisement and discovery, service and device description, device control, eventing and presentation. With eventing, a device signals state changes to devices registered for the event. The presentation interface provides an HTML based user interface as an additional access path independent of the SOAP based interfaces.

As an application example in the SIRENA [14] project, a distributed media system for hotels, enterprises or home use was selected. Media streaming and replication use embedded devices in a nearly real time environment, providing a good test case for the application of a service architecture to devices. In order to reuse existing UPnP devices, an integrative approach was selected that uses Web Service protocols for new services and interfaces, in order to overcome the limitations of UPnP and to provide additional features.

This contribution is organized as follows: Section II refers to related Work. Section III describes the UPnP architecture and the Web Services paradigm. In Section IV, the design of service interfaces for UPnP and Web Services is discussed. Section V illustrates the integrative approach for a distributed Audio-Video (AV) scenario. Finally, section VI provides conclusions and outlook.

II. Related Work

A lot of papers deal with standards such as UPnP and Web Services, and their extensions and modifications to cope with

new problems.

UPnP considered as a middleware that has borrowed technologies such as SOAP from Web Services for remote procedure calls is discussed in [10]. The paper is based on the statement that SOAP is inappropriate for Web Systems, but primarily deals with this criticism in the context of UPnP, and proposes a solution without SOAP.

In the area of device discovery, a protocol similar to HTTP is known as SLP [11]. JINI [12] provides device discovery fully integrated into the JAVA world, using interesting techniques that cannot be easily used outside this realm.

A novel Web Service system for achieving ad-hoc mobile application integration is proposed in [9]. This will require communication to be enabled dynamically, need to work across a large number of platform combinations and will have a large potential for semantic mismatch between communicating software. In this context Web Services are regarded as well-suited to addressing these problems. He also deals with UPnP and its limitations in the context of service composition. By using syntactic matching to discover a service, UPnP is not able to discover services that may be needed to be used in a composition.

An extension of WS in the area of device networking environments is *Devices Profile for Web Services* [2].

The gSOAP [7] toolkit supports developing SOAP/XML web service applications for C and C++, mostly starting from a description in WSDL. In [8] an extension of gSOAP for developing embedded Web Services is described. This open-source development environment includes a fully automated RPC compiler.

III. UPnP and Web Services

A. UPnP

UPnP [1] is an existing suite of definitions and standards to use networked devices in a plug-and-play manner, i.e. without configuration and setup. UPnP has established as a technology to enable seamless proximity networking in addition to control and data transfer among networked devices. It is in fact an early implementation of a Service Oriented Architecture.

The UPnP architecture starts with a document called *UPnP Device Architecture* [3], defining structural aspects like message format, eventing, discovery, etc., and the device description language to be used to specify UPnP devices. The current official version is still V1.0, but a update is currently in its final phase.

The UPnP architecture uses existing standards such as TCP/IP, HTTP, and XML, and each UPnP-enabled device implements a set of standardized protocols to provide for discovery, control and data transfer between connected UPnP devices. Beside the devices more basic abstractions of UPnP are services and control points. Services are units of functionality implemented by a device, while control points are able to work with the functionalities provided by the devices. Fig. 1 shows a control point invoking an action of a UPnP device.

B. Web Services

One of the most famous realizations of SOA is Web Services (WS). Primary WS support data exchange via the Internet.

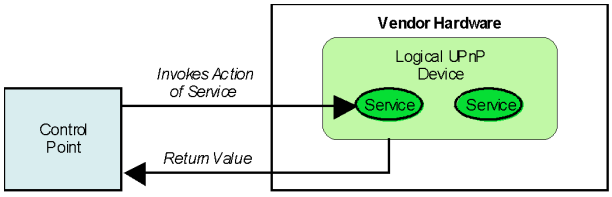

Fig. 1. Control Point Invoking an Action.

The architecture is organized into layers that build upon one another (see Fig. 2). From the bottom up these layers are network, transport, packaging, description and discovery. The latter provides mechanisms to discover services, for example the UDDI (Universal Description, Discovery, and Integration) project. The description of a Web Service provides information about the service in the Web Service Description Language (WSDL). For application data to be moved around the network, it must be packaged in a format that all parties can understand. Therefore SOAP (Simple Object Access Protocol) is used in the packaging layer. The transport layer includes the various technologies that enable the direct communication with the aid of protocols such as TCP, HTTP, SMTP, XCM and Jabber. The network layer provides the basic communication, addressing, and routing capabilities.

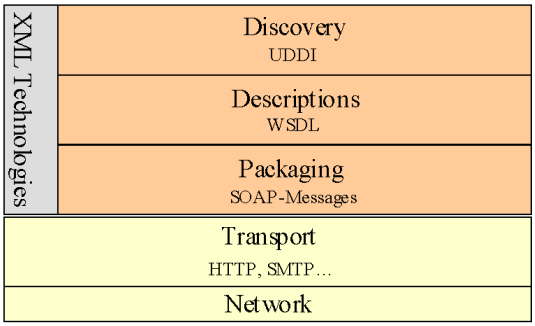

Fig. 2. WS level architecture.

C. Devices Profile for Web Services

The Device Profiles for Web Services (DPWS) standard, an extension of WS, is fully compatible with the WS standard. DPWS adds WS-Discovery and WS-Eventing to the core protocols of WS to support plug-and-play and the eventing mechanism. WS-Discovery defines a multicast protocol to search for and locate services that are seen as devices in the DPWS context. Services are provided within the network once the device is discovered. Multicast-based discovery is limited to subnetworks. For discovery to be scalable in enterprise-wide scenarios, discovery proxies are introduces. The WS-Eventing defines an event handling protocol that allows one Web Service to register interest with another WS. In [6] DPWS is discussed for using in industrial automation and DPWS is mentioned as an intention to foreshadow the next major upgrade of UPnP.

IV. Service Oriented Design Observations

Using the *UPnP Device Architecture* [3], a number of *standardized device protocols* are provided, describing a specific

set of similar devices. Currently, there have been published *UPnP Device Profiles* for AV appliances and for Internet firewalls (included in DSL access combos). Also available, as originally intended, are profiles for shared networked devices like printers, scanners etc. The architecture for AV is defined for distributing digital audio and video and uses multiple logical devices with a control point that coordinates their actions and data exchange. Fig. 3 shows a corresponding situation. It is characterized by two device types: the *Media Server*, to store media content, and the *Media Renderer*, that renders (displays) content transmitted outside the UPnP protocol.

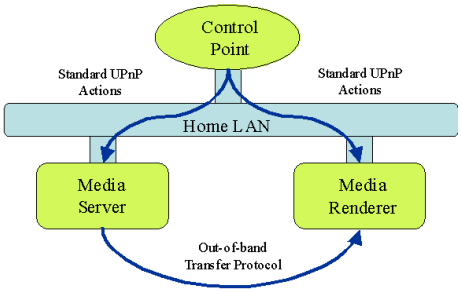

Fig. 3. UPnP AV Architecture.

In the case of AV appliances, the standard control point has a user interface, from which the selection and playing of AV media is controlled. If there are more than one control point operated by different persons, any operational and resource conflicts can be settled out of band. As UPnP is targeted to zero-configuration, it is restricted to the broadcast range of a subnet.

In the context of the SIRENA-Project, the AV environment needs overlapping resources, and consequently access to UPnP devices in different subnetworks (see Fig. 4). Available UPnP-Stacks are restricted to one network which will be controlled by a control point. Control points and UPnP devices can overlap but there is no possibility of central coordination of the overall system.

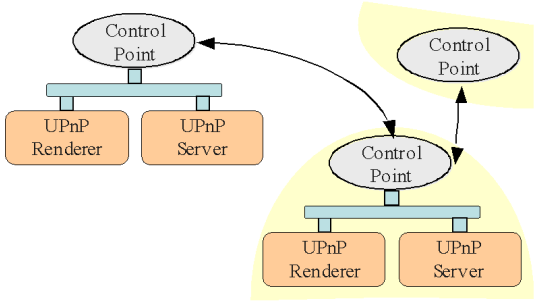

Fig. 4. Sirena AV scenario.

We observed some peculiar problems with existing device profiles, making UPnP as a general service oriented protocol in this area less attractive. Moreover, even is broadly supported by tools, object oriented designs should not be simply exposed as-is as service oriented interfaces.

A. Consistency and Transactions

Fig. 3 is an example for the common situation where a single control point is the only master that controls a number of passive devices. Here, no concurrency related problems arise. However, two or more control points can control the same device concurrently, which requires proper consideration at design time in order to avoid solutions that sometimes fail for obscure reasons. The traditional solution using connection oriented protocols and only one connection per device is not available. Device locking is also no appropriate for distributed systems.

A solution can be found in the UPnP media list device, that can be queried by a control point to obtain a list of media available for streaming. The device provides an interface for recursive traversal through a tree that represents the media directory. If the tree changes during inspection, the control point will create garbage, fail or crash. UPnP uses a solution on application level delegating the check to the control point: Each call returns a change count or data version number to be checked by the caller to be equal to that of the previous call.

However, leaving the consistency check to the caller is not advisable. No use case is seen where the caller could safely ignore a changed data version. State information must be kept by the control point and the check operation has to be programmed for each control point again. It will in several cases be postponed to the final version and forgotten. Better would be to have the check in the device, and stop delivering data if not matched.

This safeguard should be extended to calls changing the state of the device, and have a parameter that contains the change count or state number as returned by previous inquiry calls. Only if the device is still unchanged, the modifying call is accepted.

An extension would be to define transaction brackets, so that a sequence of setting calls can be transmitted to the device, and only activated after a commit call as whole set or not at all.

Many of these problems occur if the device actions are thoughtlessly exported methods of an object oriented design with its typical large number of simple GET/SET methods (and the object not designed to be shared between threads). If, on the other hand, the service oriented interface uses a small number of action calls with possibly complex semantics and more parameters, many of these consistency and transaction problems can be totally avoided.

Eventing (see below) is not really a solution. While it may be used to signal a state change caused by a second control point, this only reduces the time window for concurrency errors.

B. Eventing

With eventing, a client may register a callback in case that predefined events occur.

In UPnP, there is only a single event indicating the (potential) change of any of the state variables. The event message received by the control point contains names of state variables and their current, possibly new, values. An initial event

message is sent containing all evented state variables. This is fully consistent with the state model of UPnP: The control point has a copy of all state variables, and the event message delivers their values to this array. Upon receiving an event, the control point will re-examine the values and update e.g. the user interface with the changed state.

Without transaction support, the use of eventing requires careful programming if the event triggers other actions. Imagine that the client, upon receiving an event, starts an evaluation, e.g. a workflow step, and sends the results directly or indirectly to the device from which the event came. A new event arises, and the danger of a possible endless chain of events is given. If a control point calls several actions in a row that change state variables, after each call the device may signal the change as an event, even if this is only in intermediary step. Something similar to a broadcast storm may result, i.e. a momentary very busy situation where a lot of control points act on transient data. In an industrial control device, this could easily result what is called 'ringing', i.e. short term oszillations of the output values, which has the potential of propagating the oscillations to other devices and is hard to discover and compensate.

It has been proposed in UPnP to introduce a grace period for events, waiting a while after an event is sent until the next is to be sent. This is regarded an inferior substitute for real transaction brackets and regarded unreliable, even if it normally would reduces the amount of oszillations caused by improper design.

The Rendering Control device profile does not use the above UPnP state model, but has a single state counter variable, that is updated each time the state changes. Like UNIX signals, the necessary data is then collected by a chain of inquiry calls, for which consistency cannot be guaranteed.

C. Discovery

Discovery dynamically signals the presence of devices to others. In UPnP under the zero-configuration paradigm, broadcasting is used, restricting the community to a physical subnet.

The broadcast message contains the network address and a class identifier only, because broadcast messages should not be fragmented. Consequently, each control point will in turn send an inquiry to the new device to get its device description. In a home AV network, the number of control points is small, so the new device is not overloaded. In a factory plant, however, with hundreds of control points and devices, each new device will immediately after its initial broadcast receive a huge number of queries for its device description, overloading the device and the network. The next UPnP version proposes that a control point waits a random time until issuing the request. This will slow down the process significantly.

Nearly all other discovery protocols require or allow a proxy server for discovery. In Web Services, if a discovery proxy exists, only the proxy queries the new device, and all clients (i.e. control points) query the proxy server in turn or, preferably, register for that event. The new device can start its normal operation without spending its power for administrative issues; and slowing down of the proxy server does at least not disturb the new embedded device. Furthermore, subnets are easily spawned if the discovery proxies exchange their device lists.

V. Integrative AV Architectures

Within the SIRENA project we have implemented an experimental platform for large AV environments integrating UPnP devices and extending their capabilities with Web Services (WS) concepts. Fig. 5 illustrates the distributed video streaming scenario. There we consider an autonomous system and presume several subnetworks which are interconnected with RSVP-enabled routers. Within our testbed we maintain in each subnetwork the following devices:

- set-top boxes and/or standard PCs
- UPnP Media Renderer (e.g. [4])
- UPnP Media Server (e.g. [5])

As described in the previous chapters the discovery limitations of UPnP do not allow inter-subnet communication. Furthermore it is also not possible to extend UPnP services or to add own AV related services. Moreover it is desirable to include existing UPnP devices when designing further services. Therefor we introduce a Media Control Application (MCA) which is running on a PC or set-top box acting as a supernode for UPnP Server and UPnP Renderer.

Before we focus on the individual components of the system we present the set of services. The main topics are

1) the replica creation and distribution due to long- and short-term popularity values,
2) a global directory covering all media references in all subnetworks,
3) the network wide resource management (bandwidth).

These tasks are realized by aggregating of the following services:

- **Media Control Services** are provided by media controllers running on a non-UPnP device. They are aggregated into different use groups, e.g. User Interface, GMIDB, other media controller, resource inquiry etc. A Media Control instance may be considered as a WS-enabled management instance for UPnP devices.
- **Global Media Information Database (GMIDB) Service** is a peer-to-peer service managing content references for the whole network. Each subnetwork maintains one GMIDB instance which provides this information service to all nodes in the appropriate subnetwork.
- **Media Access Statistics Services** are responsible to collect the raw data when and how long a media object has been accessed. Similar to the GMIDB this is a distributed service, where one instance in a subnetwork is responsible for a sub-set of content objects.
- **Media Replication Services** are those provided by the replication process. This service is responsible to replicate and distribute media according to information provided by the Media Access Statistics Service.
- **Resource Allocation and Reservation Services (RARS)** help to assure an acceptable level of service, e.g. avoiding overload of resource limited parts.

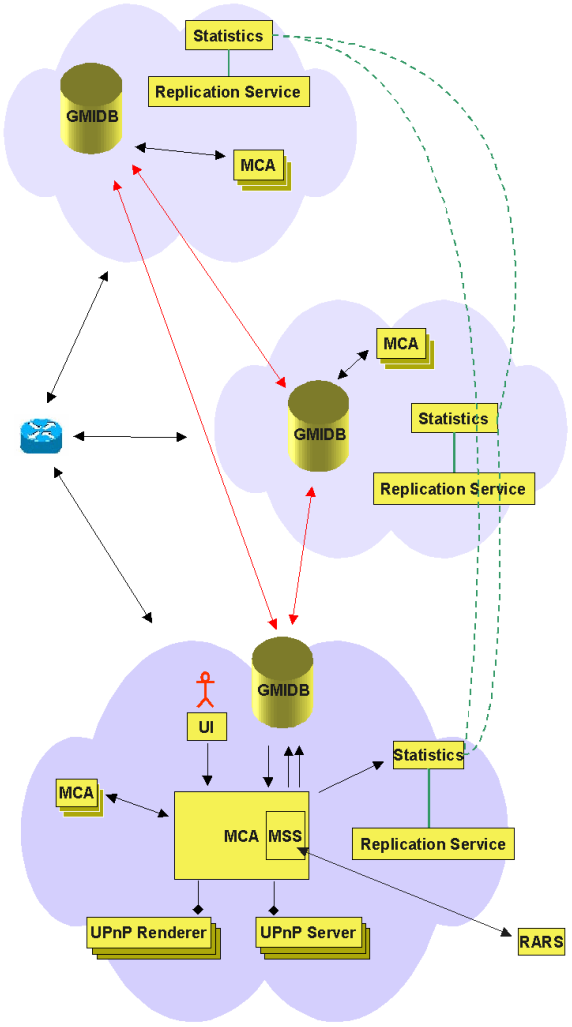

Fig. 5. AV UPnP-WS Integrative Architecture (of several subnetworks).

A. Media Control Service

To realize the mentioned services and to overcome the limitations of UPnP we introduce the Media Control Application (MCA) a management instance extending UPnP devices by additional services. Thus, the MCA is (also) a UPnP control point, as seen from the UPnP devices. Each MCA offers Media Server and/or Media Renderer capabilities in two ways. Within the first scenario an MCA is a set-top box acting as a Media Server and Media Renderer. However if there are UPnP devices the MCA controls the content provided by the interconnected UPnP Media Server and ensures that the Renderer is known as a streaming sink outside the own subnetwork.

The MCA is the main component to control the use of media. Its services are used by the GMIDB, by other MCA and by the User Interface (UI). The UI itself is a client application for a service provided by the MCA, which only delivers and receives the data, while the UI presents this data, often in a graphical way. Because the UI is not integrated into the MCA, but conceptually divided in the service interface and the UI application itself, it can reside anywhere in the network; either on the same node as the MCA, or e.g. on a PDA.

Each MCA maintains a sub-service, namely the Media Selection Service (MSS). When the MCA requests the global list of media via the web service interface from the GMIDB, the media in the list are all distinct; replica are automatically reduced to one entry by the GMIDB. This list contains the ID of the media (MID) and other information for the user, in particular the title string, date of publication, playing time, artists etc. It is given to the UI, where the user selects one of the media. With the MID, the UI calls a service in the MCA to start the play of this media on a certain media renderer, that is either implicitly or user selected in the UI.

Now, the source media selection process takes place (i.e. determining the source peer). This is indicated as a Media Selection Service (MSS). This service will be hosted by the MCA node.

Consequently, the GMIDB has to provide a second service, providing a list of all replicas for a media with a given MID. This list does now contain technical information only, i.e. the location of the media including the network address of the MCA that reported the presence of that media, etc.

More information on the media will then be obtained via a service of that MCA. For weighting all possible media sources the MSS takes into account both, (a) application specific parameters as (the sum of) popularity values of possible source peers, free streaming slots of source peers and (b) network/system specific parameters as bandwidth, distance, etc. These information are maintained by the RARS.

B. Global Media Information Database (GMIDB) Service

As currently missing in the UPnP AV architecture the GMIDB Service has to be added. This could be realized by a centralized lookup server. However we preferred a distributed peer-to-peer approach due to a higher fault tolerance.

One GMIDB peer is present in each subnet that participates in media sharing. Each peer queries all visible MCA to supply a list of media that could be used. This is the homed media list for the respective GMIDB.

Furthermore each GMIDB holds media content information of all other known GMIDB peers. Effectively, each GMIDB caches the media lists of the other GMIDB peers. Eventing is used to avoid polling the GMIDB each other; a change in the core list of locally determined media is signalled to all other GMIDB peers, that in response will update their cache. In contrast to UPnP, a replication service initiates media replication. Replicated media are pairwise identical, but located on different hosts or in different subnets. In order to identify replicas, each media in the GMIDB has a Media IDentivier (MID) which is equal for all replicas, irrespective of the location where they are currently stored or the media server that could supply it. A hash value (of the first MBytes) of the media would be best, as it allows to recreate the MID even if the user did a manual replication. Automatic replication has to ensure that the MID is the same for replica of the same media.

Of course all GMIDB instances have to get aware of one another. The address of the other GMIDB instances, as they are in other subnetworks, are controlled via the SIRENA

management tool and interface. But also the Media Controllers have to know their assigned lookup service. This discovery is done by a simple broadcast.

C. Media Access Statistics Service

In each subnetwork there is a Media Access Statistics Service located. Each time a user - either from the set-top box or a UPnP Media Renderer - requests a movie the Media Access Statistics Service stores the point of time when the movie started, when it has been finished and if it has been watched completely. First of all the Media Access Statistics Service maintains the access data of the content within the own subnetwork. These information have to be merged with the data of the other subnetworks. Thus each Media Access Statistics Service in a subnetwork is responsible for a subset of movie object statistics. This is realized by distributed hashing. On basis the raw data for one movie taking the whole network into account it is possible to determine (and even predict) the popularity of content. The Media Replication Service makes use of these information.

D. Media Replication Service

Especially in large networks it makes absolutely sense to distribute replica of highly popular content. Therefore the Media Replication Service requests the access information from the Media Access Statistics Service. This is (usually) done once per hour.

Its general operation is as follows: the GMIDB is queried and the list of media fetched. For each MID, the list of replica is obtained and the respective Media Access Statistics Service inquired for statistical data, the popularity index is determined, entered into the GMIDB and compared to the number and distribution of replica.

To start a replication, the Media Replication Service addresses the MCA that hosts the media to be replicated, which in turn contacts the target media controller and establishes the replication process, so that the replication controller can be fairly general and needs no detailed technical information about the media. The target media controller is obliged to signal the GMIDB the change as an event.

E. Resource Allocation and Reservation Service

The major task of the MSS is a resource allocation and reservation process. For this purpose, Resource Allocation and Reservation Services (RARS) are used. In particular, the MSS probes for given combination of media source and media renderer if the resources for this combination are all available. This includes not only application specific, but also network related parameters. For the network, a RARS is provided in the router, if the router has QoS functionality. If exclusively a network switch is used, the network RARS could be a small embedded device attached to the network that simply accumulates all registered bandwidth requests up to a given upper limit. The RARS is responsible to maintain and deliver the following information regarding a peer:

- bandwidth/stream counter: up & down capacity available. Therefor the RARS has to be aware of the current streams into and from the peer and their bandwidth consumption. MCA eventing is responsible to inform the RARS about the stream.
- distance: we presume the use of a link-state routing protocol (as OSPF). Thus RARS maintains a map of the network topology and is able to determine the hop-count from one peer to another.
- up-time: peers which have a long uptime have a higher probability to remain up. By sending ping/pong packets in a predefined period (several minutes) the RARS is aware of peers being alive.
- CPU load: the average CPU load may be piggybacked on the pong messages.

VI. Conclusion and Outlook

From the SIRENA [14] funded project, an integrative approach for a Service Oriented Architecture for devices has be described, using Web Services and Universal Plug and Play (UPnP) for a distributed media replication environment. UPnP issues and Web Services were presented with proposed practical rules for proper interface design of device related Service Oriented Architectures.

In the future, the aspects of automated configuration support, the process of setting up a systems, advanced diagnosis and fault detection tools and further peer-to-peer techniques can be studied with the described setup.

Acknowledgment

Parts of the work described herein was funded by the German Federal Ministry of Education and Research (BMBF) within the ITEA-SIRENA project [14].

References

[1] UPnP Forum: www.upnp.org/
[2] S. Chan, C. Kaler, T. Kuehnel, A. Regnier, R. Bryan Roe, D. Sather, J. Schlimmer, H. Sekine, D. Walter, J. Weast, D. Whitehead, D. Wright. *Devices Profile for Web Services*, August 2004, Microsoft Developers Network Library: http://msdn.microsoft.com/library/en-us/dnglobspec/html/devprof.asp
[3] *UPnP Device Architecture 1.0*, Version 1.0.1, December 2003, www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf
[4] Philips Streamium UPnP Renderer, www.streamium.com
[5] Tonkyvision UPnP Server, www.twonkyvision.com
[6] F. Jammes, H. Smit: *Service-Oriented Paradigms in Industrial Automation*, International Conference on parallel and distributed computing and networks (PDCN 2005), Innsbruck, Austria, Feb. 2005
[7] gSoap Developers Webpage at Sourceforge, http://gsoap2.sourceforge.net
[8] R. van Engelen: *Code Generation Techniques for Developing Web Services for Embedded Devices*, 9th ACM Symposium on Applied Computing SAC, Nicosia, Cyprus, 2004, pages 854-861.
[9] Steele, R., *A Web Services-based System for Ad-hoc Mobile Application Integration*, IEEE Int. Conf. on Information Technology: Coding and Computing '03, 2003
[10] J. Newmarch, *A RESTful Approach: Clean UPnP without SOAP*, IEEE Consumer Communications and Networking Conference, 2005
[11] Service Location Protocol, RFC 2608. See also http://www.openslp.org/
[12] Ken Arnold (Ed.), The JINI Specification. Addison-Wesley 2000
[13] World Wide Web (W3C) Consortium: Web Services, http://www.w3.org/2002/ws/
[14] SIRENA, Service Infrastructure for Real-time Embedded Networked Applications, Eureka Initiative ITEA (Information Technology for European Advancement) Project (01ISC09E), the German part is funded by the *Bundesministerium für Bildung und Forschung* (BMBF), http://www.sirena-itea.org/