UPnP in Integrated Home- and Building Networks

Rolf Kistler, Stefan Knauth and Alexander Klapproth
Lucerne University of Applied Sciences and Arts, CEESAR
Technikumstrasse 21, 6048 Horw, Switzerland
{rolf.kistler, stefan.knauth, alexander.klapproth}@hslu.ch

Abstract— **Classic building automation systems have shown that networks of seamlessly integrated devices add true user value to commercial as well as residential building environments. They may positively influence factors such as energy efficiency, flexibility, security and comfort. But these benefits are not convincing enough, if the handling of such networks is complex and acquisition and engineering costs are high. Ideally, the integration of a new device happens without any effort spent on commissioning. Once integrated, devices and services must be easy to use for different users. The UPnP (Universal Plug and Play) protocol suite is a good choice to address these topics. It offers plug and play capabilities and comes up with a standardised interface to control devices. It's powerful, but it cannot fulfil all the requirements needed, especially in commercial buildings. This paper explores lacks and drawbacks of UPnP, discusses possible additions and introduces CARUSO, an architecture implementing the proposed mechanisms.**

I. Introduction

Networks of connected devices become increasingly popular. Today, a modern building environment may well host independent networks for automation, IT, telecommunication, security & safety, multimedia and even domestic appliances. There is the vision of having one integrated network in which all these different networks and their devices work seamlessly together. While such a network opens the field for new distributed applications, it must keep management overhead and operational costs low. One of the most impressive examples of what a standard network protocol can achieve is TCP/IP. And although many of today's technologies adoptable for buildings have their strong reasons to exist, trends indicate that TCP/IP will finally play a major role in a truly integrated solution.

UPnP, which bases on TCP/IP, is a widely accepted, powerful and yet simple approach to connect and control internet gateways and multimedia devices in homes. The UPnP forum is supported by over 800 members across the industry. The Digital Living Network Association (DLNA) [1] has chosen UPnP as their means to easily connect devices from leading IT, consumer electronics and mobile device manufacturers. Although UPnP initially targeted devices in private homes, studies and prototypes [2] [3] [4] indicate that its core mechanisms can also be adopted for commercial buildings. The UPnP device architecture [5] covers IP addressing, device and service discovery, description, control, eventing and presentation. On top of it, the UPnP forum has developed specifications for standardised device classes, the "Device Control Protocols" (DCPs). A DCP defines a common interface for a class of devices allowing to easily handle UPnP DCP compliant devices from any vendor. DCPs have been defined for media servers & renderers, printers, internet gateways, HVAC, lighting etc.

UPnP solves a lot of problems. However, looking at the requirements of a solution for both home and commercial building domains, some of them still persist. During the course of this paper, additions to UPnP are listed, discussed and for each of them a proposal is made. The implementation is part of the CARUSO project [6], which seeks to unify the supervision and control of networked devices in buildings.

II. UPnP Additions

During the development of UPnP devices, one gets aware of its benefits and drawbacks. Some have been identified by forum members [7] and new specifications were added. We found that the drawbacks can be overcome with additions and extensions that don't break the UPnP standard. The CARUSO project currently investigates the following additions:

- **Easy network access**. UPnP assumes that all devices and Control Points are part of an IP network. "Control Point" (CP) is the UPnP term for entities used to control UPnP devices. But getting into the network is often a problem.
- **Large network sizes**. For discovery, UPnP sends chatty IP multicast messages. IP multicast does not scale very well on big networks.
- **Low bandwidth actions**. UPnP actions are transported via HTTP/SOAP telegrams. Turning on a light may result in hundreds of bytes of bidirectional network traffic.
- **Users, access rights and security**. The UPnP device architecture has no concept of a user. It's not in the scope of UPnP to define and manage users. There is no session concept and all devices and services are exposed without restriction. As for authentication and encryption, UPnP relies on the underlying IP network.
- **Sophisticated dynamic user interfaces**. Besides of a hyperlink to an optional presentation page, UPnP does not describe the user interface of a device.
- **Smart services across devices and networks**. UPnP services are device-centric. But to take full advantage of an integrated network, services will have to act across several devices. What's more, UPnP is not designed for the Internet. But it might be interesting to have services contributed from external providers via the WWW.

III. Architecture and Implementation

A. Network topology and infrastructure

The following considerations assume that all controllable devices can be seen through UPnP interfaces. Integration of

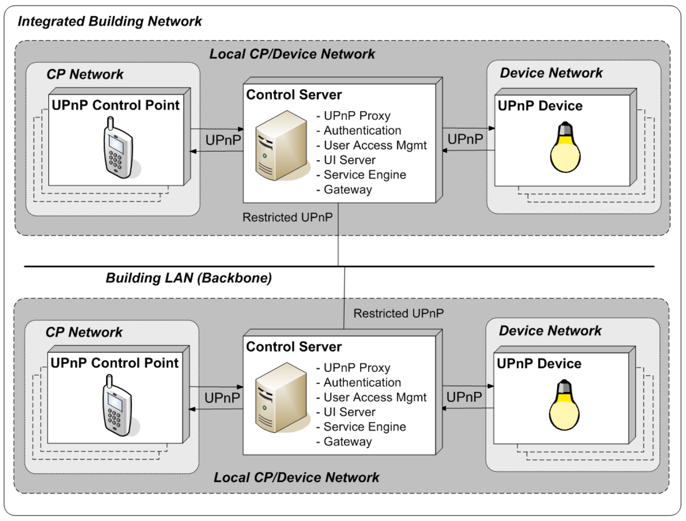


Fig. 1. The network structure with servers separating CPs and devices.

non IP devices through smart gateways and UPnP proxies have been investigated in other projects [2].

The network topology greatly affects the characteristics of the overall system. CARUSO separates the Control Points (CPs) from the devices. Although UPnP is a peer-to-peer technology and ideally, no additional infrastructure should be assumed, a server divides between the control network and the device network (Fig. 1). There are no modifications made to neither CPs nor devices. So if there is no server, UPnP runs as usual, but without the mentioned additions.

All CP/Device communication flows through a hardware server, although not necessarily through the same. To keep the multicast UPnP network scalable, there can be more than one of these servers, segmenting the whole network. The servers themselves are connected via an Ethernet backbone and present themselves as UPnP devices. That makes it possible to discover them. Traffic in the backbone is cut down as the ordinary UPnP multicast traffic does not leave the local CP/Device networks. To control a device that is not present in the local CP/Device network, the CP sends a UPnP search message to its local server, which forwards it to the other servers. These will now answer in the common UPnP fashion with the devices connected to their local networks matching the search pattern. UPnP allows for specific searches. To prevent huge network bursts after a search, the search area is restricted (e.g. search for ”all devices” will not be allowed). Further, there is a server parameter, specifying the number of answers forwarded after a search message. What it all boils down to is, that the feature of UPnP to detect new and leaving devices ad hoc is restricted to local networks. But it’s possible to find and control all devices in the network through an explicit search for them. The usage of the standard UPnP search mechanisms prevents us from defining new protocols between the server components. For situations in which bandwidth is limited, a UPnP compression algorithm can reduce traffic between two aware network elements by a factor of about 5 [8].

Whether there is one of those servers for the whole building, one on each floor or one in every room, depends on the specific requirements. Typically, such a server consists of a small embedded PC running the OSGi framework [9]. The proposed additions are modularly implemented as OSGi bundles installed on each server. The complementary UPnP and OSGi technologies are ideally suited for such environments. The OSGi Alliance considered this when it included the ”UPnP Device Service Specification” into the OSGi Service Platform.

B. Security and User Access

A network which interconnects devices of a building raises the security question. There is the legitimate fear of a user bringing down a whole building from the lobby with his mobile phone. While newer technologies such as ZigBee integrated a strong security concept in their initial design, the main source of UPnPs critics was its lack of security ceremonies. A flaw in Microsoft’s UPnP implementation encouraged this even more. The UPnP forum delivered the missing DCPs in addition in form of ”Device Security” and ”Security Console”. Device security secures a UPnP device while the console provides access to secured devices and manages access rights. Unfortunately, these DCPs came much later, are optional and not very easy to understand and implement. Most of the UPnP devices today ignore them. Well, in home environments, secure light switches may not be an important feature. And even in commercial buildings, passwords are often transmitted as plain text and access mechanisms can be overcome with common field bus tools. Still, our solution needs a concept of users and a way to protect the building from unwanted access.

In CARUSO, the user approaches the system with a WLAN enabled mobile device. The first hurdle is to connect this CP to the network. There is a risk that configuration issues, such as acquiring and entering network keys, will distract many from using their client. If an unsecured network is not an option, there is the way of applying emerging NFC and RFID technologies to increase usability. If a new user enters a building, she just holds the client near an NFC access point (like a badge) and immediately enters the local CP network (or is denied). A similar procedure connects devices to the device network. In general, the premise is that the device network is much less accessible than the CP network. Once the client is part of the CP network, it needs to authenticate to the server. This is where UPnP device security enters the scene. As all the traffic flows through the server and the device network is (more or less) save, the authentication needs to be done only once, based on the server’s security service. The server may act as security console to the device network, to configure devices implementing UPnPs device security.

The user description and access right management is a compromise between the highest possible security solution and the one with lowest configuration efforts. For CARUSO, no individual users are defined but four typical roles with their own access schemes based on access levels. Access control information is stored on the server. The smallest access entity is an UPnP action. Discoveries and searches are currently not secured. The server holds the predefined access levels for all actions of all known UPnP DCPs. A global policy exists for all other (non-DCP) devices (allow all, deny all). As an example, this makes it possible to restrict access for a standard user to devices he could also control using conventional controls (e.g. light switches). If the client is aware of its location, access could be granted on the condition of physical presence in a room. While in those scenarios, no configuration is needed, introducing new user levels, new device classes or differentiating between instances of individual users or devices, result in human intervention. Decisions on the server are simple: If an authenticated SOAP action is received from the CP network, the access level of the user and the action are compared. If the access level of the user is equal or higher than the one of the action, the action is forwarded to the device.

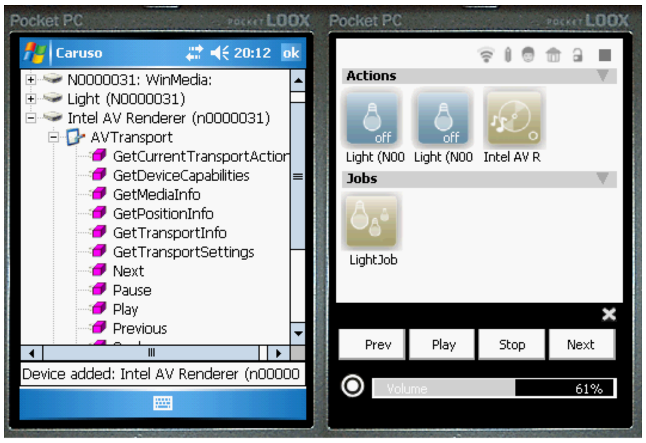
C. User Interfaces

The user interface (UI) provides the means to control the integrated devices and services. This makes it one of the most important factors concerning the user acceptance. In our case, it must be able to adapt to the context of use. Different users should see different, task based views of the system. The UI shall be properly rendered on mobile devices running Windows Mobile 5. Intuitive, responsive and graphically appealing are further adjectives describing a good UI.

It's a common design practice to separate the functional aspects of a system from its representation to the user. UPnP provides functional device and service descriptions. But its designers also thought about presentation issues. That's why the XML device description holds an optional ”presentationURL” tag with a link to a UI description. Later on, the possibilities were expanded by defining DCPs for remote UI servers and clients. These allow UPnP aware remote UI clients to detect possible (proprietary) sources of UIs on UI servers and download them for rendering. [10] discusses these concepts.

Taking ”plain” UPnP (no UI DCPs), there are several alternatives for the CP to render the UI. The first fundamental decision is whether the UI information is delivered over the network or resides in an application on the CP. For the sake of context awareness, maintainability and homogeneity, we decided to deliver it over the network.

In a straight forward approach, the CP could generate a UI out of the pure XML description delivered by UPnP. The device description contains useful metadata and the service descriptions list all ”actions” a user can execute on the device, including the ”arguments” and their data types. For a UPnP Media Renderer, the CP sees that there is a service ”RenderingControl” with an action ”SetVolume” that takes an Integer from 0..100 as ”DesiredVolume”. It could build up a tree view listing the detected devices with all their services and actions (Fig. 2). Then it could provide an entry dialog for each action which simply lists all arguments top down. As an example, for ”SetVolume” there would be a text entry field or a slider for 0..100. It's up to the user to interpret these terms shown on the UI coming directly from the UPnP description. Such a UI may be helpful for an engineer reading parameters out of a device he knows well. But it's not meant for standard users. In our Media Renderer example, all the user might want to do is to play/stop a movie and set the volume.



Device added: Intel AV Renderer (n00000

```
<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <ScrollViewer>
    <StackPanel>
      <DockPanel>
        <Button Click="AVTransport:Previous:InstanceID:0">Prev</Button>
        <Button Click="AVTransport:Play:InstanceID:0:Speed:1">Play</Button>
        <Button Click="AVTransport:Stop:InstanceID:0">Stop</Button>
        <Button Click="AVTransport:Next:InstanceID:0">Next</Button>
      </DockPanel>
      <Slider SelectionChanged="RenderingControl:SetVolume:InstanceID:0:
                                Channel:Master:DesiredVolume:#Value"
               Name="Volume" Minimum="0" Maximum="100" Value="50" />
    </StackPanel>
  </ScrollViewer>
</Page>
```

Fig. 2. Left: User interface generated out of pure UPnP descriptions. Right: User interface of a simple media renderer generated out of the XAML file depicted below.

The two UPnP services achieving this hold 46 actions and a lot of cryptic arguments. The CP has no information on which actions to present to which user and how to chose and arrange elements for e.g. the play/stop actions.

An alternative is to generate web pages delivered over the presentation link. Unfortunately, investigations have shown that today's mobile browsers are not ready to deliver the performance and the feature set needed for the UI we have in mind. Future generations of devices and technologies such as AJAX, Flash Lite, MS Silverlight etc. may change that. In CARUSO, the CP gets a file written not in HTML but in a User Interface Description Language (UIDL). Its a declarative description of the UI. Numerous such XML languages have been created already (a recherche came up with around 25). The XAML language, which is an integral part of the Microsoft WPF (Windows Presentation Foundation), has been evaluated. As the standard UPnP device is not likely to send this file, a UI bundle running on a CARUSO server generates the XAML description. The server takes the UPnP description coming from the device, injects a link to the XAML file and forwards it to the CP. The event handlers in the XAML file point to UPnP actions executed when a user interacts with the according UI elements (Fig. 2). The file does not contain absolute location coordinates and sizes of elements etc. The CP makes these decisions when it parses the XAML file and renders the UI (depending on display resolution, hardware performance etc.). [11] further discusses dynamic UI generation.

D. Services

Users approach the system with certain tasks in their mind. These may be as simple as ”turn on the light” or ”set the room temperature to 20 degrees centigrade". But they can also become more complex, like "watch a DVD" or "prepare room 302 for a presentation meeting". The system shall place services at the user's disposal to achieve her goals. UPnP defines services and how to discover and utilize them. However, UPnP - as a peer-to-peer technology - is inherently device centric. Its services always relate to one specific device. But the services the user has in mind may involve an intelligent orchestration of several devices. In general, the user is neither interested in which devices are involved nor what sequence of actions to execute on them. She just wants to get the job done.

Such "high-level" services can also be published through UPnP. To make this happen, we introduce a new UPnP device, the "Services Server". The CPs in the system can detect services servers connected to the UPnP network and query for their services. Each UPnP service of such a server device provides a high-level service to the user. The high-level services themselves run as OSGi bundles on the server node. This node also has knowledge of the other UPnP devices attached to it (via device network) and has the ability to control them. One could think of dynamically created services such as "turn on all the lights" or services carefully hand crafted for specific tasks. In the end it's up to the implementer of the service bundle, how intelligent the service interacts with the network and the user. There are promising research results on smart services [12] [13].

UPnP has been designed for local networks. Sending multicast discovery messages around the Internet is not feasible. It is possible to connect two local UPnP networks over the Web using VPN techniques. An idea has been proposed to connect mobile Internet devices via the SIP protocol invented for IP phones [14]. And a web based UI front-end can be used to remotely manage and control the local network if that is a wish. But for stakeholders of a building as well as for service providers, it might be an interesting option to integrate external services over the Internet. One could think of services reaching from simple weather forecasts to intelligently control blinds, up to health care or ambient assisted living (AAL) scenarios for elderly people still living at home.

Web services are the natural choice to implement these services. Regrettably, UPnP descriptions are not compliant with the XML dialect that describes web services (WSDL). The reason is that the web services standard did not exist, when UPnP was designed. There are two ways of opening the network for web services. They both lead over the services server. First, the server could act as a web services directory (UDDI) for the local network. That would require the CPs to handle the web services standard and bring a completely new mechanism into the system. As there is already a service concept based on UPnP, we decided to build an UPnP-to-Web Services gateway into the services server. So for each web service, an (automatically generated) wrapper bundle is running on the server translating UPnP actions to web services remote procedure calls (both SOAP based). The web services are internally published over the UPnP services list of the services server. At this point, it's worthy to mention that there is an initiative going on to merge the two standards. The Device Profile for Web Services (DPWS), sometimes called UPnP V2.0, seeks to close the gap. It intends to include missing device support and plug and play capabilities for web services, which on their terms add some of the missing features to UPnP. Time will show if DPWS is a better solution and reaches the wide acceptance of UPnP [15].

IV. Conclusion

The paper exposed additions to UPnP in an integrated home- or building network. Key requirements of such a network include ease of use, minimal engineering efforts and broad, standards based applicability. Having this in mind, we proposed a network structure and possible solutions to connect network participants, manage users and restrict their access, provide sophisticated dynamic user interfaces and integrate smart high-level services. These topics are studied and implemented in the course of the ongoing CARUSO project, which seeks to unify and simplify the supervision and control of devices in buildings. We think that the proposed additions fully enable UPnP to play a major part in such networks without losing its undeniable strengths.

References

[1] DLNA - Digital Living Network Alliance, "DLNA Overview and Vision Whitepaper 2006", *[ONLINE] http://www.dlna.org*

[2] W. Kastner and H. Scheichelbauer, "UPnP Connectivity for Home and Building Automation", *Parallel and Distributed Computing and Networks, 2004*

[3] S. Knauth, R. Kistler, D. Käslin and A. Klapproth, "SARBAU - towards highly self-configuring IP-fieldbus based Building Automation Networks", *IEEE International Conference on Advanced Information Networking and Applications 2008*

[4] A. Klapproth, D. Käslin and T. Peter, (Apr. 2005), *Lowcost Wireless Webserveer, ZigBee Entwicklerforum 2005 Munich, [ONLINE] http://www.ceesar.ch/cms/upload/pdf/Paper%20Wireless%20Webserver%20HTA%20Luzern.pdf*

[5] UPnP Forum, "UPnP Device Architecture", Version 1.0, June 2000, *[ONLINE] http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0.pdf*

[6] R. Kistler, S. Knauth, D. Käslin and A. Klapproth, " CARUSO - Towards a context-sensitive architecture for unified supervision and control", *IEEE International Conference on Emerging Technologies & Factory Automation, 2007. ETFA, p. 1445-1448*

[7] F. Reynolds, "The Ubiquitous Web, UPnP and Smart Homes", *[ONLINE] http://www.w3.org/2006/02/reynolds-paper.pdf*

[8] S. Knauth, R. Kistler, D. Käslin and A. Klapproth, "UPnP Compression Implementation for Building Automation Devices", *5th IEEE International Conference Industrial Informatics, INDIN 2007, p. 75-79*

[9] OSGi Alliance, "The Dynamic Module System for Java", *[ONLINE] http://www.osgi.org/*

[10] G. Zimmermann, G. Vanderheiden, C. Rich, "Universal Control Hub & Task-Based User Interfaces", 2006, *[ONLINE] http://myurc.org/publications/2006-Univ-Ctrl-Hub.php*

[11] J. Nichols and B. A. Myers, "Controlling Home and Office Appliances with Smartphones", *IEEE Pervasive Computing, special issue on Smart-Phones, Vol. 5, No. 3, July-Sept, 2006, p. 60-67, 2006*

[12] M. Valle, F. Ramparany, L. Vercouter, "Flexible composition of smart device services.", *The 2005 International Conference on Pervasive Systems and Computing(PSC-05), 2005*

[13] IST-AMIGO, European Information Society Project on Ambient Intelligence for the Networked Home Environment, *[ONLINE]http://www.hitech-projects.com/euprojects/amigo/*

[14] B. Kumar and M. Rahman, "Mobility Support for Universal Plug and Play (UPnP) Devices Using Session Initiation Protocol (SIP)", *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE, Vol. 2, p. 788-792*

[15] H. Bohn, A. Bobek, F. Golatowski, " SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains", *IEEE International Conference on Networking Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006, p. 43-48*