Towards UPnP-UP: Developing Context-Aware UPnP Applications (position paper)

Thiago Sales, Leandro Sales, Hyggo Almeida, Angelo Perkusich

[1]Embedded System and Pervasive Computing Laboratory
Electrical Engineering and Informatics Center
Federal University of Campina Grande (UFCG)
Campina Grande, PB – Brazil

{thiagobruno,leandro,hyggo,perkusich}@embedded.ufcg.edu.br

Abstract. *The Universal Plug-and-Play (UPnP) standard aims at connecting consumer electronics, intelligent appliances and mobile devices from many different vendors. Albeit providing a sophisticated solution for device and services discovery, UPnP control points and devices cannot communicate with each other by providing user and context sensitive information, preventing UPnP application developers from building smarter solutions for Pervasive Computing and allowing unauthorized users to grant access for UPnP resources. For this reason, this paper proposes an extension of the UPnP standard named UPnP-UP, which allows user authentication and authorization mechanisms for the UPnP standard, maintaining backwards compatibility with previous versions of UPnP.*

1. Introduction

In recent years, there has been an extraordinary advance of wireless connection technologies for mobile devices and a worldwide availability of connectivity infrastructure, such as *Wifi* and the *Global Positioning System* (GPS), allowing people to migrate their tasks from desktop-based platform to the mobile ones. Therefore, the concepts of mobile and pervasive computing are becoming more and more practical, where computing would be integrated in our daily lives, allowing us to seamless interact with the environment [Weiser 1991].

The Universal Plug-and-Play (UPnP) ([1]) standard is a good candidate to provide pervasive services for a new generation of electronic devices. A UPnP network is a collection of interconnected computers, network appliances, and wireless devices that use standard protocols to discover, advertise, and access network services. As a matter of fact, UPnP supports connections for devices such as *cell phones* or *internet tablet* PCs, as well as benefits for connecting to conventional peripherals such as printers or wireless household electronic gadgets for controlling appliances, lights, doors, and curtains.

As a widely-available and easy-to-implement protocol, UPnP has been used in many open-source and proprietary projects throughout the Internet, such as Canola ([2]) and BRisa ([3]). Despite of the strong popularity and growth of manufactured UPnP-

[1]http://upnp.org/
[2]http://openbossa.indt.org.br/canola/
[3]http://brisa.garage.maemo.org

compatible devices, *UPnP does not provide* a standard for user authentication and authorization mechanisms, limiting the development of context-aware solutions. It may include music recommendation systems while using a UPnP media renderer ([4]), the adjustment of a fuzzy light level or an air-conditioner temperature according to the user's preferences, an access-control for allowing or denying an access to a UPnP Printer device based on authorization policies [Dridi et al. 2004]. By supporting these features, the UPnP infrastructure would provide autonomic [Lin et al. 2005] and context-aware UPnP services.

In this context, this paper presents UPnP-UP (Universal Plug and Play - User Profile), an extension of the UPnP standard that enables user authentication and authorization for UPnP devices and applications, providing a flexible way for developing context-aware UPnP solutions. In a nutshell, UPnP-UP would provide a sophisticated mechanism for managing user profiles and access control rules among available resources in a UPnP network.

This paper is organized as follows: after presenting some features of the UPnP standard in Section 2, we detail the specification of the UPnP-UP in Section 3, describing the main components to achieve context-aware UPnP applications. Then, we present some prototype scenarios in Section 4 and, finally, in Section 5, our conclusion and future works.

2. The Universal Plug and Play Standard

The UPnP protocol first uses the control point ([5]) to enter in the discovery step (1) by searching for UPnP-enabled devices. The discovery process enables the description step (2), where the control point learns about the devices capabilities. In order to consume services from devices, the control point uses the control step (3) through SOAP protocol. In the event step (4), the control point keeps listening to the changes of state of the hooked up devices, updating the graphical user interface accordingly, which is defined in the presentation step (5).

Despite offering zero configuration and a flexible way of connectivity, the UPnP standard does neither provide user authentication nor authorization mechanisms. These requirements would allow context-aware UPnP applications by collecting users' attributes and preferences with information from the environment. For instance, you can imagine a scenario capable of recommending multimedia contents based on user's preferences, such as the two well-known music genres: *rock* and *blues*. In addition, since its basic idea is to support an open networking architecture, theres no way to *granting* or *denying* an access to a UPnP service based on users' attributes and information from the environment. A simple scenario is a UPnP Printer device that expose the *CreateJob* service for printing. At the current UPnP specification, anyone with access to the control point can request the *CreateJob* service as many as times as desired, without user authentication or authorization.

Finally, due to the absence of an access control mechanism for protecting resources in the UPnP network, as well as the lack of users' contextual information to

[4] `http://upnp.org/specs/av/`
[5] `http://www.upnp.org/resources/`

recognize each of them in the environment, these devices are unable to protect or provide context-aware behavior, allowing anyone to freely access them.

3. An Extension for the UPnP Standard

In this section, we present UPnP-UP (Universal Plug and Play - User Profile), an extension for providing user authentication and authorization for UPnP appliances. The goal of the UPnP-UP extension is to change the UPnP standard to provide context-aware UPnP applications.

A UPnP device should send a modified version of the NOTIFY message, which will apprize control points that it supports service customization through the UPnP-UP extension. The new version of this message should be one as illustrated in Listing 1, line 5. The proposed NOTIFY message contains a new field named UP (User Profile), which indicates whether the remote UPnP end-point is UPnP-UP compatible or not. The absence of this field means the UPnP-UP device is not compatible, allowing backward compatibility with the UPnP core stack.

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.255.255.250:1900
3 LOCATION: http://10.20.30.40/device_description.xml
4 NT: ssdp:all
5 UP: Yes
```

Listing 1: New UPnP NOTIFY message

In order to manage the user profiles, we introduce a new UPnP device specification called *User Profile Server* (UPServer). It is responsible for storing user profiles information, such as full name, email address, login, password and specific information regarding any UPnP specification, such as preferences for UPnP Home Automation (air-conditioner temperature, level of brightness etc). The UPServer is also responsible for keeping authorization policies regarding the available services in the network, such as controlling access to a UPnP Light or a UPnP Printer device.

The UPnP services ([6]) of UPServer are described in Listing 2, lines 11, 22 and 33 - *UPAuthentication*, *UPAuthorization* and *UPProfile*, respectively. From these information, the control points can get access to the service description and invoke a service (see the values of the tags <SCDPURL> and <controlURL>).

```
1 <root xmlns="urn:schemas-upnp-org:device-1-0">
2   <device>
3     <deviceType>
4       urn:schemas-upnp-org:device:UPServer:1
5     </deviceType>
6     <serviceList>
7       <service>
8         <serviceType>
9           urn:schemas-upnp-org:service:UPAuthentication:1
10        </serviceType>
11        <serviceId>
12          urn:upnp-org:serviceId:11
13        </serviceId>
14        <SCPDURL>/UPServices/up-authentication.wsdl</SCPDURL>
15        <controlURL>/UPServices/up-authentication/control</controlURL>
16        <eventURL>/UPServices/up-authentication/event</eventURL>
17      </service>
18      <service>
19        <serviceType>
20          urn:schemas-upnp-org:service:UPAuthorization:1
21        </serviceType>
22        <serviceId>
23          urn:upnp-org:serviceId:22
24        </serviceId>
25        <SCPDURL>/UPServices/up-authorization.wsdl</SCPDURL>
```

[6] All UPnP devices expose an XML file that describes their main features and the services that they provide (see Listing 2)

```
26        <controlURL>/UPServices/up-authorization/control</controlURL>
27        <eventURL>/UPServices/up-authorization/event</eventURL>
28      </service>
29      <service>
30        <serviceType>
31          urn:schemas-upnp-org:service:UPProfile:1
32        </serviceType>
33        <serviceId>
34          urn:upnp-org:serviceId:33
35        </serviceId>
36        <SCPDURL>/UPServices/up-profiles.wsdl</SCPDURL>
37        <controlURL>/UPServices/up-profiles/control</controlURL>
38        <eventURL>/UPServices/up-profiles/event</eventURL>
39      </service>
40    </serviceList>
41  </device>
42</root>
```

Listing 2: UPServer device description.

All UPServer services will be described according to the UPnP services template ([7]). Besides the *auth* SOAP method, the authentication-based services also provide *logout* and *renew session* services for managing user authentication. The former is a simple user *sing out* and the last one is described in deeper details in Section 4. In addition, the authorization-based services manage the authorization policies and rules for the UPnP resources access control according to the users' attributes and information from the environment. Finally, the user profile-based services expose users' information to UPnP devices, such as preferences for the UPnP fuzzy light intensity, air-conditioner temperature or, as depicted in Listing 3, multimedia (audio, video and image) contents. By exposing the user profile through web services, the UPServer can store the users' preferences and the access control policies into any back-end the developer prefers.

```
1<root xmlns="urn:schemas-upnp-up-org:up-1-0">
2  <userProfile id="7569">
3   <category id="1" name="AudioVideoImage">
4     <genreList>
5       <genre id="17" title="MPB">
6         <artistList>
7           <artist weight="83">Toquinho</artist>
8             <itemList>
9               <item contentType="AUDIO" weight="82">Aquarela</item>
10              <item contentType="AUDIO" weight="93">O Caderno</item>
11              <item contentType="VIDEO" weight="75">A Grande Viagem</item>
12            </itemList>
13          </artist>
14        </artistList>
15      </genre>
16      <genre id="8" title="Rock"/>
17    </genreList>
18  </category>
19  </userProfile>
20</root>
```

Listing 3: User preferences for Audio and Video UPnP specification.

Regarding to the UPnP events (step 4 of the UPnP standard) for UPServer, we can describe three main events: The event *onChangeProfile* occurs every time a user profile is changed. This event may be useful to the UPnP devices that want to cache user profile information to avoid retrieving the user profile from the UPServer every time it is necessary to be processed. The event *onUserEnter* is fired when a new user authenticates in the UPServer, allowing UPnP services and control points (other users) to be aware of it. And finally, the event *onUserExit* indicates the departure (logout) of the current user from the UPnP network.

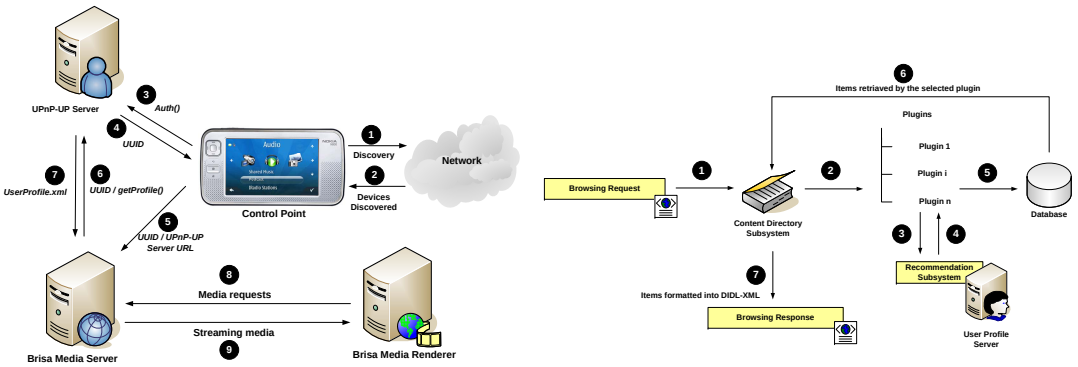
4. Prototype Development

Taking into account the availability of user profiles and an access control list (ACL) in a UPnP network, context-aware pervasive solutions on top of UPnP can be developed.

[7] http://upnp.org/resources/documents/Service-Template-1.01_000.doc

Briefly, a UPnP based application that adapts the level of brightness according to the user preference can be deployed in a living room, acting upon the user when s/he gets into the environment. Furthermore, the mechanism of ACLs also furnishes the entire application with a security high level service, like determining whether the user has permission to control the light or to change the air-conditioner temperature.

Once the UPServer is defined, the UPnP A/V specification can be used to show how UPnP-UP can help the development of context-aware UPnP applications. This configuration is illustrated in Figure 1(a). After discovering available UPnP devices (steps 1 and 2), a control point invokes the *auth* SOAP method (step 3). At this point, the control point sends the username and password to the UPServer and receives back a user authentication token ID (UUID). This token will identify the user after a successful authentication (step 4), allowing devices to collect user profiles (steps 5, 6 and 7) and customize their services.



(a) UPnP-UP communication schema.

(b) The browsing process using the Recommendation Subsystem.

Figure 1. Basic scenarios.

For instance, suppose a UPnP system capable of sharing multimedia items such as musics, movies and images. According to the user's profile, the system could infer and decide which songs would meet the current user's preferences. As a proof of concept, we have been developed the BRisa Framework, a UPnP infrastructure that implements the UPnP A/V specification. After authenticating in the UPServer, a control point requests the *browse* SOAP method to select multimedia items from the *BRisa Media Server*, as depicted in Figures 1(a) and 1(b), steps 5 and 1, respectively. This device, in its turn, selects the proper plug-in according to which plug-in is responsible to handle the request instead of presenting all available media to the control point. In this way, it invokes the recommendation subsystem at the UPServer, sending the current user UUID identification. Based on the current user's attributes and preferences, the *BRisa Media Server* returns a set of multimedia items to the control point, which is formatted into a *Browsing Response* as defined in the UPnP A/V specification.

The UUID is valid during 300 seconds and, in case of a timeout, the control point should invoke the *renew session* action to renew its communication UUID. This service is also useful in case a control point suddenly disconnects from the network without explicitly invoking the *logout* service. Additionally, it adds a security level in the case of a user UUID is obtained by an another non-authenticated user. It's important to point out

that the UUID should be encrypted. By adopting this strategy, the UPnP-UP extension expects to provide a mechanism to avoid network attacks, like data modification during transportation, UUID spoofing and Man-in-the-Middle scenarios.

5. Conclusion and Future Works

As discussed in this paper, all the available UPnP services based on the UPnP specification cannot make use of context sensitive information about the users and the environments. Therefore, UPnP devices are unable to customize services and thus not allowing UPnP networks to be more dynamic. As a result, each service developer implements his own non-standard proprietary solution, making interoperability among available UPnP services more difficult to this goal. On the other hand, since our research is a work in progress, the UPnP-UP extension appears to be a contribution for *user* authentication and authorization mechanisms to the UPnP standard, keeping the specification compatible with the current UPnP implementations and enabling a new set of applications to be deployed over a UPnP network.

The UPnP-UP extension requires minor changes to enable both user authentication and authorization for UPnP network. This satisfied our principle of changing the UPnP standard as little as possible and keeping interoperability among UPnP and UPnP-UP applications. Finally, the UPnP-UP has been possible thanks to the flexible and extensible standard offered by the UPnP Forum, which allows the addition of new devices and services through a definition of artifacts such as XML file descriptions and SOAP Web Services.

As future works, user profiles will be described through the OWL (Web Ontology Language) [Yan et al. 2006] standard and the security policies will be defined as a set of SWRL (Semantic Web Rule Language) [Lee et al. 2007] rules. As a consequence, the solution can provide a more sophisticated process of inference and context interpretation due to the high dynamism and heterogeneity of users, services and devices available in pervasive environments. Additionally, we are investigating where the interpretation engine should be in order to maximize the availability of context-aware UPnP applications - in case when the UPServer is unavailable - and to minimize the effort of current solution to support the UPnP-UP extension.

References

Dridi, F., Muschall, B., and Pernul, G. (2004). Administration of an rbac system. pages 6 pp.–.

Lee, K.-C., Kim, J.-H., Lee, J.-H., and Lee, K.-M. (2007). Implementation of ontology based context-awareness framework for ubiquitous environment. *Multimedia and Ubiquitous Engineering, 2007. MUE '07. International Conference on*, pages 278–282.

Lin, P., MacArthur, A., and Leaney, J. (2005). Defining autonomic computing: a software engineering perspective. pages 88–97.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):66–75.

Yan, Y., Zhang, J., and Yan, M. (2006). Ontology modeling for contract: Using owl to express semantic relations. *Enterprise Distributed Object Computing Conference, 2006. EDOC '06. 10th IEEE International*, pages 409–412.