J Braz Comput Soc (2010) 16: 261–277
DOI 10.1007/s13173-010-0022-2

ORIGINAL PAPER

A UPnP extension for enabling user authentication and authorization in pervasive systems

Thiago Sales · Leandro Sales · Hyggo Almeida · Angelo Perkusich

Received: 6 April 2010 / Accepted: 14 September 2010 / Published online: 7 October 2010

Abstract The Universal Plug and Play (UPnP) specification defines a set of protocols for promoting pervasive network connectivity of computers and intelligent devices or appliances. Nowadays, the UPnP technology is becoming popular due to its robustness to connect devices and the large number of developed applications. One of the major drawbacks of UPnP is the lack of user authentication and authorization mechanisms. Thus, control points, those devices acting as clients on behalf of a user, and UPnP devices cannot communicate based on user information. This paper introduces an extension of the UPnP specification called UPnP-UP, which allows user authentication and authorization mechanisms for UPnP devices and applications. These mechanisms provide the basis to develop customized and secure UPnP pervasive services, maintaining backward compatibility with previous versions of UPnP.

Keywords Pervasive computing · Universal Plug and Play · Authentication and authorization

T. Sales (✉) · H. Almeida · A. Perkusich
Federal University of Campina Grande, Aprigio Veloso Street, Campina Grande, Brazil
e-mail: thiagobruno@embedded.ufcg.edu.br

H. Almeida
e-mail: hyggo@dsc.ufcg.edu.br

A. Perkusich
e-mail: perkusic@dee.ufcg.edu.br

L. Sales
Federal University of Alagoas, Lourival Melo Mota Avenue, Maceió, Brazil
e-mail: leandro@embedded.ufcg.edu.br

1 Introduction

In recent years, the wide dissemination of mobile devices with wireless technologies has allowed the conception of pervasive computing solutions [2]. Technologies like *Bluetooth*, *Wi-Fi,* and 3G allow people to migrate their tasks from desktop-based platforms to mobile ones, accessing services and information anytime, anywhere. The concept of pervasive computing is becoming practical, with computing systems more and more integrated in people's daily lives, seamlessly interacting with devices and users in the environment [1].

Due to its robustness and still simple networking architecture, the Universal Plug and Play (UPnP) technology [3] is a promising solution to provide pervasive services for a new generation of electronic devices. A UPnP network is a collection of interconnected computers, network appliances, and consumer electronic devices that use standard protocols to discover, advertise, and access network services. The goal is to provide an easy-to-use framework for creating tools to support the communication of network-based devices. UPnP supports communication between devices such as *cell phones* and *internet tablets*, and conventional peripherals, such as printers and wireless household electronic gadgets for controlling appliances, lights, doors, and curtains.

The main motivation for conceiving the UPnP specification was the lack of protocols for service discovery in pervasive networks, mainly considering interoperability, decentralization, and language-independence issues. Some existing implementations of Service Discovery Protocols (SDPs) do not address these issues, such as Jini [4], Salutation [5], and SLP [6]. UPnP is based on the standard eXtensible Markup Language (XML), with control protocols defined using SOAP [7]. SOAP is an Internet-based protocol that promotes interoperability in distributed systems on top of

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) stacks. In this way, besides supporting communication between consumer electronics devices, UPnP enables Internet-based services.

As a widely available and easy-to-implement stack of protocols, UPnP has been used in many open-source and proprietary projects available on the Internet. One worth-citing project is BRisa,[1] a UPnP framework implemented in Python, C++, and Java, developed by the Embedded System and Pervasive Computing Laboratory at UFCG, Brazil.

Despite the strong popularity and growth of manufactured UPnP-compatible devices, UPnP does not provide a standard for user authentication and authorization mechanisms, limiting the development of customized and secure applications. Examples of these applications include: recommendation systems [8]; customized adjustment of environment appliances, such as fuzzy light level or air-conditioner temperature; and rule-based access control for UPnP devices, such as printers; among others. For these applications, it is mandatory to identify the user and his preferences. By supporting these requirements, UPnP solutions would provide customized and secure UPnP services.

This paper introduces UPnP-UP (Universal Plug and Play—User Profile), an extension of the UPnP technology that enables user authentication and authorization for UPnP devices and applications, providing the basis to develop customized and secure UPnP services. Basically, the UPnP-UP extension provides a seamless connectivity model for managing user profiles to customize UPnP services and allowing access control to UPnP appliances based on user profile. As a consequence, future UPnP solutions can provide different behaviors by collecting user profiles and acting proactively. Moreover, UPnP-UP has been designed to modify the current UPnP specification as little as possible, still providing backward compatibility with the UPnP core stack.

After presenting some features of UPnP in Sect. 2, some related works are discussed in Sect. 3. Sections 4 and 5 detail the specification of UPnP-UP and two case studies, respectively. Section 6 discusses the results of this work. Section 7 describes the current and future work, while Sect. 8 concludes by summarizing the major topics covered in this research.

2 Technology overview

This section presents an overview of the main technologies and concepts involved in the development of the UPnP-UP solution.

[1] http://brisa.garage.maemo.org/.

2.1 Universal Plug and Play (UPnP)

The UPnP protocol is defined as a set of different steps [3]. First, after attaching an IP address, a control point searches for available UPnP devices during the discovery mechanism (Step 1). After finding the available devices, the control point can use the description process (Step 2) to learn about the devices' capabilities. In order to execute services from devices, the control point uses the control phase (Step 3) through the SOAP protocol. In the event process (Step 4), the control point keeps listening to state changes of hooked up devices, updating the graphical user interface accordingly in the presentation process (Step 5).

To discover UPnP devices, a control point sends a multicast SSDP-based request message to a standard multicast address and port. In addition, a control point may unicast a discovery message to a specific IP address on port 1900 (Listing 1). When a new UPnP device is added to the network, it sends multicast messages to a standard address and port (Listing 2) for each embedded device and service.

UPnP A/V [8] is the UPnP specification for audio and video, which is illustrated in Fig. 1. The control point browses multimedia items from a media server (Step 3) and these items can be rendered in the Media Renderer (Steps 5 and 6). This specification is focused on the UPnP technology dedicated to distributing and executing digital content (music, videos, and images) through the network. Some solid solutions that implement the UPnP A/V specification worth mentioning are BRisa [9] and *Google Media Server*, announced by Google for sharing local multimedia items among users.

Despite offering zero configuration and a flexible way of connectivity, the UPnP does not provide user authentication and authorization mechanisms. These requirements would allow customized UPnP applications by collecting user preferences and information from the environment. For instance, consider an application for recommending multimedia contents based on user preferences, such as music genres *rock* and *blues*.

Moreover, since the basic idea for UPnP is to support an open networking architecture, UPnP services do not cope with the user properties when accessing them. For example, there is no way to *grant* or *deny* access to a UPnP service based on user attributes and information from the environment. A simple scenario is a UPnP Printer device that provides the *CreateJob* service for printing. In the current UPnP specification, anyone with access to the control point can request the *CreateJob* service as many times as desired, without user authentication or authorization.

Figure 2 illustrates a basic scenario where a UPnP Internet Gateway Device (IGD) can be used for a DNS Injection attack [10]. The UPnP IGDs [11] provide (1) information of

```
1 M-SEARCH * HTTP/1.1
2 HOST: 239.255.255.250:1900
3 MAN: "ssdp:discover"
4 MX: seconds to delay response
5 ST: search target
6 USER-AGENT: OS/version UPnP/1.1 product/version
```

Listing 1 UPnP discovery message based on SSDP

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.255.255.250:1900
3 CACHE-CONTROL: 500
4 LOCATION: http://10.20.30.40/device_description.xml
5 NT: ssdp:all
```

Listing 2 Notify device and services through the SSDP NOTIFY message

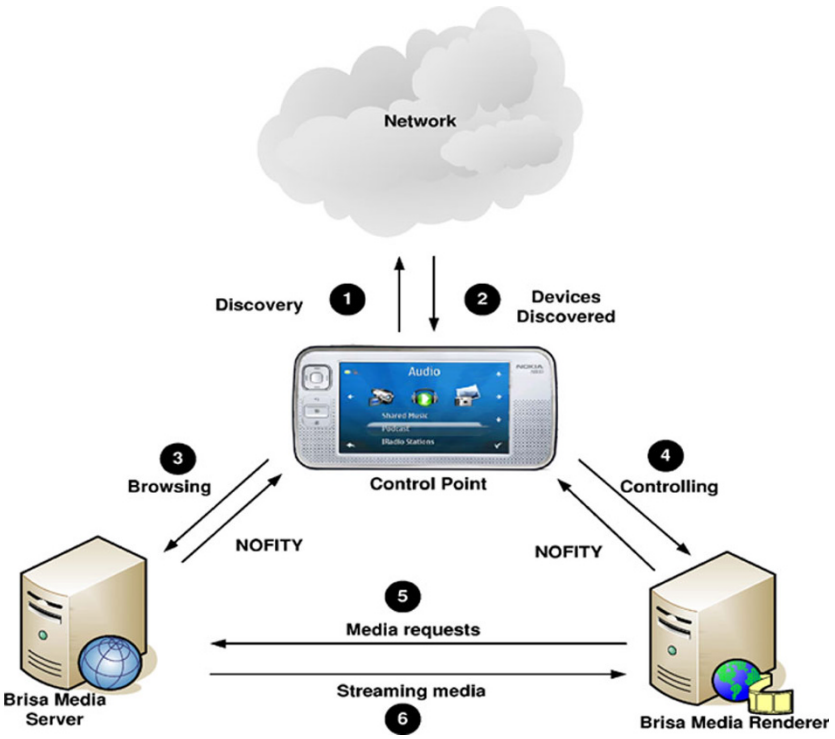

Fig. 1 Basic communication scheme for UPnP A/V devices

status and events on connections, (2) control of initialization and tear drop of connections, and (3) management of host configuration services (DHCP, Dynamic DNS). A remote user, from an external network, requests a domain name translation for www.bank.com (Step 1), receiving back the IP address from the DNS server available at the private network (Step 2). However, a UPnP IGD specification allows a user on the local LAN to request a port mapping without user authentication or authorization. This is a security issue. An attacker can request a port mapping to a UPnP IGD service, called *addPortMapping*, to add a port forward that redirects all the DNS traffic to a malicious DNS server. For example, a user requests a port mapping for port 53 (Step 4) to redirect to the IP *192.168.1.109*. As a result, all external subsequent requests to port 53 (Step 5) will be forwarded

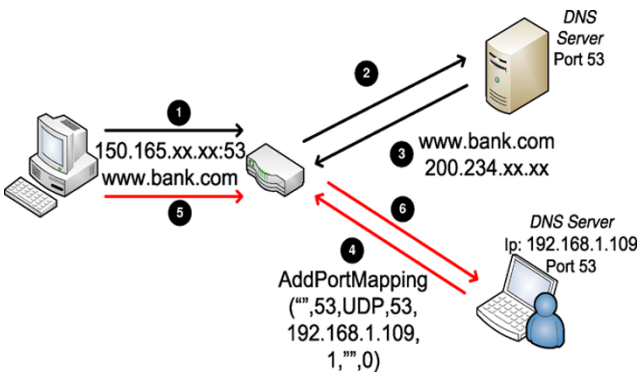

Fig. 2 UPnP IGD attack

to the malicious server (Step 6), running a fake DNS server, which resolves the given URL to a fake website.

Nowadays many network administrators disable the UPnP support of the UPnP IGD devices. Nevertheless, by providing a port mapping with the *addPortMapping* service, a UPnP device behind IGD can enable *Nat Traversal*, allowing external applications to establish a connection with it.

The aforementioned scenarios can also be extended to other UPnP application domains. For example, intelligent appliances for home automation (IAHA) aims at connecting devices around residential environments, providing a way to control lights, curtains, air-conditioner, TVs, doors, etc. Due to the absence of an access control mechanism for protecting resources in the UPnP network and the lack of user information to recognize each of them in the environment, these devices are unable to protect or provide customized behavior, allowing anyone to freely access them.

Mechanisms for authentication and authorization must be addressed to guarantee medium or high level security on the pervasive applications, due to the heterogeneity of devices, services, and users in pervasive environments. Recent advances in pervasive computing have brought new solutions which use UPnP as the technology for discovering devices and services. Nevertheless, such solutions use non-standard mechanisms for device and user authentication and authorization processes. Since UPnP defines several standards (called *UPnP profiles*) for devices and services, it would be important to provide a specification and an architecture that leverage UPnP-based technologies for authentication and authorization to UPnP services, allowing better device-to-device interoperability in a scalable networked environment.

2.2 Authentication and authorization in pervasive environments

In computer security, access control includes, among other features, the authentication and the authorization mechanisms. Identification and authentication are the processes of checking something (or someone) as authentic. In short, authentication is the basic building block of security. In the electronic world, the authentication of an entity can be processed by using shared secrets (including passwords), public key cryptography schemes, biometrics, and so forth. User identification and authentication in pervasive environments are also important due to the range of devices and services to which users have access. Currently, many solutions use IP addressing identification and authentication, which are not enough in pervasive computing. This model works well for traditional deskbound personal computers, where the user usually works with a static IP address in the same machine. On the other hand, in a pervasive computing environment, users have different devices and connect to different networks.

An authorization process can be seen as the mechanism to allow or deny an access to a set of available resources of a computational entity (called *objects*). If a *subject* (a user, a device, etc.) tries to get access to these resources, they will be allowed if they have some degrees of permissions. Access control models can be divided into two classes: those based on the *capability* and those based on an *access control list* (ACL). The former is a non-falsifiable reference (or capability) that a subject gains from an object to access it, which is analogous to having a password for a physical safe. The ACL-based access control allows a subject to gain access to an object if its identity is on a list associated with the object. The access control in pervasive computing should also take into account the context of the objects in the environment, such as current time and user activities. These requirements bring new challenges into the computer security domain due to the high heterogeneity of such objects.

Computers are increasingly entering our environments, ubiquitously embedded into devices and appliances available in people's everyday lives. In addition, the diversity of people in a pervasive environment requires novel security solutions in order to protect available resources (e.g., files, devices, services) in the network. To protect the environment from illegal accesses and a variety of threats, researchers have proposed many frameworks and architectures that can be used in pervasive applications [12–15]. However, these solutions provide non-standard technologies, which bring challenges to achieve interoperability with other solutions. In order to safely deploy UPnP services and appliances based on user profiles, it is required to build an architecture that uses UPnP-based technologies to be easily integrated with UPnP networks.

3 Related work

Within the context of this research, some important works have been proposed. Microsoft's patent [16] for UPnP authentication and authorization proposes a secure handshake service based on digital signatures to provide authentication for devices. Devices allow control points to access a given service if the control point features match the requirements of the service, including device model, supported media formats, etc. User information is not defined or available during the handshake process.

Another patent offers a dedicated solution for user authentication and authorization in UPnP networks [17]. A UPnP device must provide a hierarchy of authentication folders configured in a control directory server. A user PIN (Personal Identification Number) is used for authentication and for providing data access control according to the authentication level. However, only data access control is not

enough in pervasive environments, where a proliferation of services are available all the time. Services access control also plays an important role in pervasive systems. A considerable drawback for those solutions is that the patents are protected by intellectual property laws, requiring fees in order to use them in third-party applications.

In the UPnP Forum solution, the devices enforce their own access control through the UPnP Device Security [18] and Security Console [19] specifications. Device Security provides services for authentication, authorization, replay prevention, and privacy of SOAP actions. In order to establish and maintain the access control policies, a special control point called Security Console manages all security-aware devices, those that implement Security Device specification, available in the entire network. In spite of being a standardized UPnP specification, no user-related information is required during the authentication and authorization sessions to provide access control.

The authentication and authorization mechanisms discussed in previous works are not enough in pervasive environments. Pervasive applications require information about the users and the environment such as their current activities, personal information, time, weather conditions, etc., in order to provide "anytime and anywhere customized services". As new pervasive applications use UPnP as the main technology for devices and services discovery [20, 21], it would be important to provide an architecture that leverages UPnP-based technologies to provide user authentication and authorization mechanisms in UPnP networks. Nowadays, these applications use proprietary solutions to cope with user information, bringing new challenges to achieve device-to-device interoperability when dealing with user profiles.

4 An extension for the UPnP technology

This section presents UPnP-UP, an extension for providing user authentication and authorization for UPnP appliances. The user authentication and authorization mechanisms provide the basis to develop customized UPnP services. UPnP-UP defines the User Profile Server and a set of UPnP services and events in order to build customized and secure UPnP services.

4.1 The UPnP user profile server

The User Profile Server is introduced as a new UPnP device profile called UPServer. It is similar to the UPnP Media Server [8]; however, it is responsible for storing user profile information, such as full name, login, and password. Moreover, specific information regarding any UPnP specification, such as preferences for UPnP Home Automation can also be stored. The UPServer also keeps authorization policies regarding the available services in the local network, such as controlling access to UPnP Light devices [22]. Once the UPServer is defined, Fig. 1 must be modified to insert the new UPnP device to the basic UPnP A/V solution. The result of this change is illustrated in Fig. 3.

According to Fig. 3, after discovering available UPnP devices (Steps 1 and 2), a control point invokes the *auth* web service method (Step 3). Later, the control point sends the username and password to the UPServer. The control point receives back a user authentication token ID (UUID generated by the UPServer). This identification will be used to identify the user after a successful authentication (Step 4). In this way the UPnP-UP enabled application will be capable of getting user profile and providing access control.

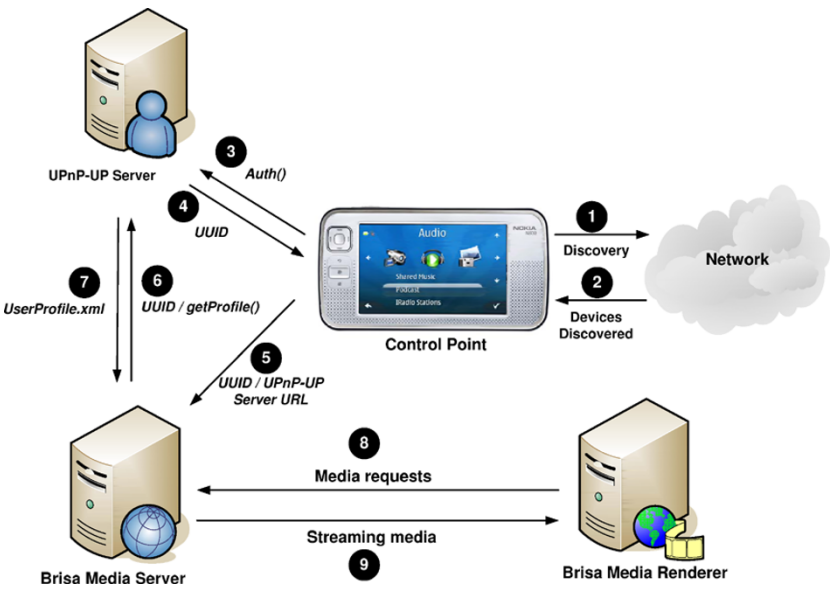

Fig. 3 UPnP-UP communication scheme

```
1<root xmlns="urn:schemas-upnp-org:device-1-0">
2 <!-- basic UPnP description fields were omitted -->
3  <device>
4   <deviceType>
5    urn:schemas-upnp-org:device:UPServer:1
6   </deviceType>
7   <!-- some device information were omitted -->
8   <serviceList>
9    <service>
10    <serviceType>
11     urn:schemas-upnp-org:service:UPAuthentication:1
12    </serviceType>
13    <serviceId>
14     urn:upnp-org:serviceId:11
15    </serviceId>
16    <SCPDURL>/UPServices/up-authentication.wsdl</SCPDURL>
17    <controlURL>/UPServices/up-authentication/control
18                                                </controlURL>
19    <eventURL>/UPServices/up-authentication/event
20                              </eventURL>
21   </service>
22   <service>
23    <serviceType>
24     urn:schemas-upnp-org:service:UPAuthorization:1
25    </serviceType>
26    <serviceId>
27     urn:upnp-org:serviceId:22
28    </serviceId>
29    <SCPDURL>/UPServices/up-authorization.wsdl</SCPDURL>
30    <controlURL>/UPServices/up-authorization/control
31                                                </controlURL>
32    <eventURL>/UPServices/up-authorization/event
33                              </eventURL>
34   </service>
35   <service>
36    <serviceType>
37     urn:schemas-upnp-org:service:UPProfile:1
38    </serviceType>
39    <serviceId>
40     urn:upnp-org:serviceId:33
41    </serviceId>
42    <SCPDURL>/UPServices/up-profiles.wsdl</SCPDURL>
43    <controlURL>/UPServices/up-profiles/control
44                              </controlURL>
45    <eventURL>/UPServices/up-profiles/event</eventURL>
46   </service>
47   </serviceList>
48   <presentationURL>URL for presentation</presentationURL>
49  </device>
50</root>
```

Listing 3 UPServer XML description

All UPnP devices expose an XML file that describes them and the services they provide. Listing 3 shows the UPServer device description. From this information, the control points can get access to the service description and invoke the respective service (see the values of the tags <SCDPURL> and <controlURL>). Besides the basic information about the device, such as the device name and manufacturer (these tags were omitted), Lines 11, 24, and 37 of Listing 3 define the authentication-based services (*UPAuthentication*), the authorization-based services (*UPAuthorization*), and services based on user profiles (*UPProfile*), respectively.

Besides the *auth* web service, UPServer also provides the *getProfile*, *renewSession,* and *logout* services. The former allows UPnP devices to retrieve the user profile, while the second one is invoked by control points to renew the session with the UPServer every time this session expires. The UUIDs sent by the UPServer to the control points are valid for 300 seconds. In case of a timeout, the control point must invoke the *renewSession* method to renew its communication UUIDs. Additionally, this web service is also useful when the control points suddenly disconnect from the network without invoking the *logout* service. It also adds a security level in case a user without authentication acquires a valid UUID. All UPServer services are described according to the UPnP services template, available at http://upnp.org/resources/documents/Service-Template-1.01_000.doc.

Concerning the authorization policies, the UPServer provides the *getACL* and *Authorization* services, which allows UPnP devices to retrieve the access control list and to request for an access control decision for a specific

user over a given service, respectively. ACLs are described using XACML (eXtensible Access Control Markup Language) [23]. By exposing the user profile and the access control policies through web services, the UPServer can store this information on any back-end the developer wishes to use. For the prototype developed and described in Sect. 5, an LDAP back-end was used to store the user profiles.

Regarding the UPnP events for UPServer, the UPnP-UP extension provides three events:

- The *onChangeProfile* event 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 read;
- The *onUserEnter* event is triggered when a new user is authenticated in the UPServer, allowing UPnP services and the control points to be aware of it;
- The *onUserExit* event indicates the exit (logout) of the current user from the UPnP network.

4.2 UPnP-UP design

The UPnP-UP extension was designed to provide backward compatibility with old versions of the UPnP technology. That is, the UPnP-UP extension must satisfy the requirements for letting UPnP devices and UPnP-UP enabled devices communicate with each other.

Our proposal is based on different steps of customizing the UPnP stack. The first step is to modify the UPnP NOTIFY message sent from the UPnP devices to the network. If the UPnP device is UPnP-UP compatible, this device must send a modified version of the NOTIFY message to the target control point. The new version of this message must be one as illustrated in Listing 4, Line 6. 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 device is not compatible with the UPnP-UP extension, allowing backward compatibility with the UPnP core stack. If a UPnP device is not UPnP-UP compatible, control points that are aware of the UPnP-UP extension must work properly and must not consider the authentication process. Older UPnP devices must be changed (new firmware, for instance) to provide customized services, setting the UP field to the NOTIFY messages and providing new services implementations. If a control point is not UPnP-UP compatible but the UPnP device is aware of the UPnP-UP extension, then the UPnP device will just allow access to those services that are "free" (no access control) and do not need authentication, blocking other service requests that need user identification.

Once the control point has been notified with the UP field for UPnP-UP extension, it is able to invoke the remote *auth* service to the UPServer. In Fig. 3 the control point invokes this service and receives back a random UUID (Steps 3 and 4), indicating that the authentication occurred successfully. Next, the control point accesses a Media Server service, say the *browse* service, which returns a list of available multimedia items. When the control point invokes the *browse* web service, it sends its UUID to the UPnP Media Server. The UPnP Media Server requests the user profile (Steps 6 and 7) to provide some kind of customized service, like recommending multimedia content based on the user profile (Steps 8 and 9). The idea behind the UPnP-UP extension can be used with other UPnP devices, such as to adjust autonomously a UPnP fuzzy light level or to adapt a desired environment temperature of the air conditioner according to the user preferences.

4.2.1 The new model of UPnP devices

After the user authentication process is finished, UPnP devices should interact between each other according to the UPnP-UP extension. To achieve this process, the control point must send the user identifier in every message transmitted and the other UPnP end-points should use the user identifier to adapt their responses according to the user profile. Also according to Fig. 3, the mechanism may cache the user profile the first time the media server retrieves it. Hence, for the second time and so forth the media server retrieves the user profile from the cache, avoiding unnecessary data transfer in the network. In addition to caching the user profile, the control point should register to the *onChangeProfile* event to be notified if the user changes its profile.

The control point must use a SOAP Security extension [24] to transport user UUID in the SOAP header. It is also highly recommended to use a digital signature and/or data encryption provided by W3C XML Encryption Recommendation to transport the sensitive user information,

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.255.250:1900
3 CACHE-CONTROL: 500
4 LOCATION: http://10.20.30.40/device_description.xml
5 NT: ssdp:all
6 UP: Yes
```

Listing 4 New UPnP NOTIFY message

```
1<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy">
2  <Subject>
3    <SubjectMatch ...>
4      <AttributeValue ...>organization.org</AttributeValue>
5    </SubjectMatch>
6  </Subject>
7  <Resources>
8    <Resource>
9      <ResourceMatch MatchId="...">
10       <AttributeValue DataType="...">Printer
11         </AttributeValue>
12     </ResourceMatch>
13   </Resource>
14 </Resources>
15 <Rule RuleId="PrintRule" Effect="Permit">
16   <Target>
17     <Action>
18       <ActionMatch ...>
19         <AttributeValue ...>CreateJob</AttributeValue>
20       </ActionMatch>
21     </Action>
22   </Target>
23   <Condition ...>
24     <AttributeValue ...>studentOrProfessor
25       </AttributeValue>
26   </Condition>
27 </Rule>
28 <Rule RuleId="DenyOtherActions" Effect="Deny"/>
29</policy>
```

Listing 5 Access policy for controlling UPnP printer CreateJob service

such as passwords and UUIDs. 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" [25].

Considering our principle of providing backward compatibility with non UPnP-UP enabled devices, adding a new type of UPnP device does not impact the UPnP current specification. The unique point of the UPnP specification that must be changed is the original content of the UPnP NOTIFY message by the addition of a new field called UP. This field is used by the control points and UPnP devices to determine whether a certain UPnP device supports the UPnP-UP extension.

4.2.2 *Access control for UPnP services*

The UPnP-UP authorization mechanism is based on XACML. The XACML specification is a standard to support authorization mechanisms based on XML (eXtensible Markup Language). The XACML provides a model for describing policies to a target resource. It is also a protocol for requests and responses during an authorization process. XACML is maintained by the OASIS[2] consortium.

The security access policies and their respective rules can be stored by the local UPnP network manager, recovering available UPnP services from other devices and applying desired rules. For a better understanding of the UPnP-UP authorization process, Listing 5 describes the security policy based on XACML for the following rule: *only graduate students and professors at the domain organization.org are allowed to print using the UPnP printer available on the network.*

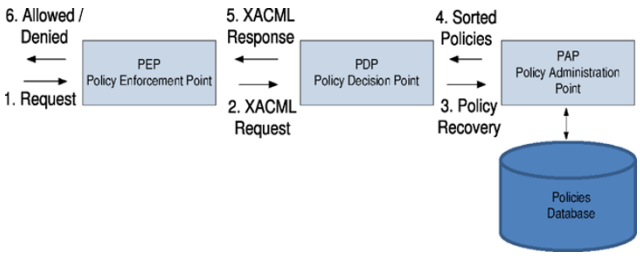

Fig. 4 Interactions between UPnP-UP authorization modules

The domain in which the security policy is being applied is specified between Lines 2 and 6. The policy is applied for all users in the domain *organization.org*. An example of the relationship between a policy and a resource (services, files, devices, etc.) is depicted between Lines 7 and 13 for a UPnP Printer device. A rule for that policy is defined between Lines 15 and 27: the waited action *CreateJob* at Line 19, and the condition to allow the access to the resource, *studentOrProfessor* at line 24. Note that one or more rules may be defined (e.g., *DenyOtherActions* rule, Line 28).

As illustrated in Fig. 4, the UPnP-UP authorization subsystem has three modules. When a new request is received, the Policy Enforcement Point (PEP) module creates an XACML request (Step 1), described in Listing 6, which includes user personal information (Lines between 2 and 6),

[2]Organization for the Advancement of Structured Information Standards.

```
1 <Request xmlns="urn:oasis:names:tc:xacml:1.0:context" ...>
2   <Subject>
3     <Attribute ...>
4       <AttributeValue>user@organization.org</AttributeValue>
5     </Attribute>
6   </Subject>
7   <Resource>
8     <Attribute ...>
9       <AttributeValue>Printer</AttributeValue>
10    </Atribute>
11  </Resource>
12  <Action>
13    <Attribute ...>
14      <AttributeValue>CreateJob</AttributeValue>
15    </Attribute>
16  </Action>
17 </Request>
```

Listing 6 XACML request for accessing CreateJob service

```
1 <Response>
2   <Result ResourceID="UPnPrinter">
3     <Decision>Permit</Decision>
4     <Status>
5       <StatusCode Value="urn:oasis:names:tc:xacml:1.0:
6                 status:ok"/>
7     </Status>
8   </Result>
9 </Response>
```

Listing 7 Example of XACML response for Media Renderer Play service

the target resource (Lines between 7 and 11), and the action that the user wishes to access (Lines between 12 and 16).

The XACML request is forwarded to the Policy Decision Point (PDP) module (Step 2, Fig. 4). PDP retrieves the authorization policies from the Policy Administration Point (PAP) module (Steps 3 and 4) and decides to allow or deny the access to the specific resource. Then, the PDP module returns back an XACML response to the PEP module (Step 5), as described in Listing 7. The resource identifier and the authorization process result are defined in Lines 2 and 3, respectively. The PEP module returns the final result of the authorization process, allowing or not allowing the access to the target resource.

4.2.3 User profile for UPnP-UP enabled devices

The user profile uses an XML format and it is divided into two types of profile descriptions: (1) the *UPnP-UP Personal User Profile*, which aims at keeping user-related information, such as *id*, username, email address, etc.; and (2) the *UPnP-UP User Profile by Specification* where for each UPnP specification, such as UPnP A/V, UPnP Light, and Printer, users have their associated preferences divided into containers. The goal is to split the user profile into different containers to facilitate the process to retrieve user information, optimizing the user profile access and the transmission of useful profile information to the UPnP end-points. In this case, the UPnP devices access the preference container they need to retrieve. For instance, the user profile of Listing 8 shows some personal information and user preferences for multimedia applications.

The user profile described in Listing 8 shows some personal information between Lines 4 and 9 (up:personal_data) and external user profiles information between Lines 11 and 15 (up:external_profiles). In addition, the user preferences information is described between Lines 17 and 36 (up:preferences_data), which is divided into different containers. For instance, container *id* equal to 0 defines the user preferences for multimedia contents. As previously discussed, the UPnP technology defines the UPnP A/V specification for UPnP multimedia applications. Multimedia contents in UPnP A/V are described as DIDL (Digital Item Declaration Language) [26] digital information metadata. Therefore, multimedia contents in user profile preferences also have DIDL metadata attached to container *id* 0, as shown between Lines 20 and 29. This user gave 4 as the weight for the first *<item>*. The weight of each item means how much a user likes that item, and it can be used to provide customized multimedia services.

It is important to note that a given user can play different roles depending on the environment that he or she has been assigned. For example, if a user is at her home, she has all the access privileges for controlling electronic conventional peripherals, such as the home front door, printers, and TVs. On the other hand, when she moves to other places, such as at work or at a given university, she plays a different role and

```
1 <?xml version="1.0" encoding="UTF-8">
2 <up:profile up-id="TBMS1302" xmlns:up="urn:schemas-upnp-org:up-1-0">
3
4   <up:personal_data id="personal_profile">
5     <up:name>Thiago Bruno Melo de Sales</up:name>
6     <up:username>thiagobrunoms</up:username>
7     <up:email>thiago.sales@ee.ufcg.edu.br</up:email>
8     <!-- Other perfonal information omitted -->
9   </up:personal_data>
10
11  <up:external_profiles>
12        <up:profile name="Embedded/UFCG" at=150.165.63.2/>
13        <up:profile name="Home" at=200.199.88.88/>
14        <up:profile name="IC/UFAL" at=200.17.114.38/>
15  <up:/external_profiles>
16
17  <up:preferences_data>
18    <up:container_list>
19       <up:container id=0 name=AV>
20         <DIDL-Lite xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
21            xmlns:dc="http://purl.org/dc/elements/1.1/">
22       <item weight=4>
23             <dc:title>GenRosso - StreetLight - Brazilian Show</dc:title>
24         <dc:description>A musical based on the true story of
25 Charlies Moats.</dc:description>
26         <dc:format>video</dc:format>
27       </item>
28       <!-- Other multimedia items omitted -->
29     </DIDL-Lite>
30       </up:container>
31       <up:container id=1 name=HVAC>
32         <!-- UPnP HVAC preferences go here, such as environment
33       temperature -->
34       <up:container>
35    </up:container_list>
36  </up:preferences_data>
37 </up:profile>
```

Listing 8 A user profile example that describes personal and preferences information

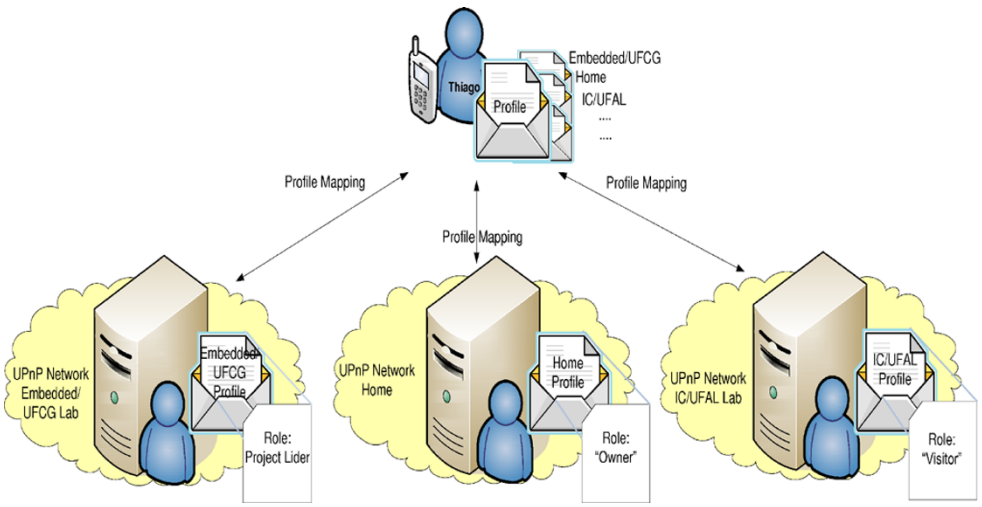

Fig. 5 UPnP-UP user profile architecture

must have other access privileges. Due to this requirement, the user profile for UPnP-UP extension has the concept of "external profiles", which associates the user profile to other remote user profiles (described in Lines between 11 and 15) and depicted in Fig. 5. Each local UPServer stores this profile and retrieves it when the user authenticates herself, according to the user UPnP-UP identification (*up-id* attribute). Finally, the proposed user profile was defined based on the available set of UPnP specifications, which allows one to decompose and to retrieve an appropriate user information as quickly and efficiently as possible.

4.2.4 *Considerations for providing service customization with UPnP-UP*

By enabling user profiles in the UPnP architecture, new types of systems can be designed, which should make use

of the UPnP technology to search and discover devices in a network. For instance, suppose a UPnP system is capable of sharing multimedia items such as music, movies, and images. According to the user profile obtained by the UPnP-UP, the system can offer songs based on the genre given by a user as the genre of his preference. In this way, it is possible to develop a more sophisticated system that can rate all the songs listened to by the user through the media renderer and then infer which genre he prefers, such as rock, jazz, etc.

Taking into account the availability of user authentication and authorization mechanisms in the UPnP network, many other new requirements can be developed in UPnP applications, such as customized services on top of UPnP, since the proposed extension allows the user identification in this type of network. For instance, consider home automation applications. Briefly, a UPnP-based application that adapts the fuzzy light level, according to the user preference, can be deployed in a living room, acting upon the user when he enters the environment. Furthermore, the mechanism of authorization 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.

5 Case studies

In the case studies, the BRisa UPnP Media Server is used to run the experiments. The BRisa Media Server [9] is a UPnP end-point device developed in the Embedded Systems and Pervasive Computing Laboratory, at Federal University of Campina Grande, which implements the UPnP A/V specification. It has been written in Python using a plug-in based architecture. In order to compute the recommendation of multimedia items, the BRisa Media Server was modified to support UPnP-UP following the steps described in Sect. 4.2. It has a dynamic multi-thread plug-in loading feature focusing on resource limited devices, mainly for the Nokia Maemo Platform. Additionally, it has a database scheme to avoid loading many objects into the main memory. BRisa can be freely downloaded from http://brisa.garage.maemo.org/, since it is distributed as an open source software under the MIT license. In what follows, two case studies are presented, highlighting the main features of UPnP-UP.

5.1 Case study 1—multimedia content recommendation

Recommendation systems are capable of recommending items based on user profiles and by using information filtering techniques. In face of this type of scenario, the user profile is compared to a given multimedia content (content-based filtering [27])—which can be basically some textual information, such as *genre* and *author*—and compared to the interests of other users, collecting user preferences (collaborative-based filtering [28]).

Neighborhood-based methods [29] are the most prevalent algorithms based in the collaborative approach. They create a subset of appropriate users (neighbors) based on their similarities to the current user. The cosine or Pearson correlation [30] can be used to achieve this goal. In (1), $W_{a,u}$ is the weight of the current user a to a given user u, $r_{a,i}$ is the rating of the user a to the (multimedia) item i, and $\bar{r}_a$ is the average of the ratings of the active user a.

$$W_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a)^2} * \sqrt{\sum_{i=1}^{m}(r_{u,i} - \bar{r}_u)^2}}$$

(Pearson correlation). (1)

After finding the similarities for all users, a weighted aggregate of neighbors ratings is used to generate predictions for the current user (2). This predicted value would be the evolution that the user would give to an item, if he consumes it.

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u) * W_{a,u}}{\sum_{u=1}^{n} |W_{a,u}|}$$

(a weighted aggregate of neighbors ratings). (2)

Another approach related to content filtering is called Content-Based Filtering. It is based on the information retrieval (IR) field [31]. A user profile is composed of a set of keywords and associated weights that indicate the strength of the word in the filtering process. The user profile is matched against a collection of documents, and the most similar are recommended. There are many ways to compute the similarity between two strings. The most common approach is the *TF–IDF* (Term Frequency–Inverse Document Frequency) algorithm [32], which is based on (3). The term frequency (TF) in the given document is the number of occurrences of a given term in that document, and the Inverse Document Frequency is a measure of the general importance of the term in a set of documents (D is the total number of documents in the collection and N_p is the number of documents in which the term occurs). The more rare the term is, the greater will be its IDF.

$$\text{idf}(p) = \lg \frac{D}{N_p}$$

(Inverse Document Frequency). (3)

The user profile and each document are represented by a vector containing the most relevant term. The cosine correlation computes the similarity between these two vectors. This correlation ranges between 0 and 1.

When a new user wishes to have multimedia files recommendations, he needs to set up his multimedia profile. As

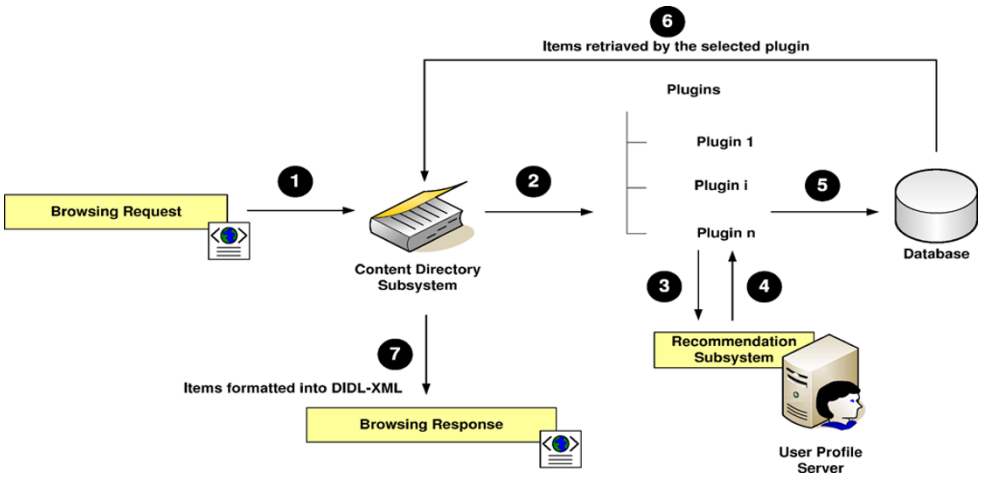

Fig. 6 The browsing process using the Recommendation Subsystem

a result, this process consists of building his user profile for music, videos, or audios of his preference. In this case study, the user profile is collected from a private website, available to users internally, at the Embedded Systems and Pervasive Computing Laboratory, at the Federal University of Campina Grande. Future work will allow a user to associate his profile through the BRisa Control Point.

The first step in the BRisa Recommendation Subsystem is the collaborative filtering approach. For each multimedia consumed item, a listened-to song, a watched video clip, etc., the user specifies a weight to the consumed item considering his preferences. The possible values can be specified ranging from 0 to 10, which indicates the importance of the multimedia items to him. The collaborative filtering approach generates prediction values for those multimedia items that the user did not consume in the past.

Secondly, the content filtering approach is processed by selecting those items for which the calculated prediction value is higher than or equal to a previously given threshold, say 5. The idea is to find similar items with the TF-IDF algorithm. To achieve this last approach, the multimedia file metadata is obtained through a DIDL description.

Another important feature of the BRisa Media Server implementation is the plug-in based architecture, which is shown in Fig. 6. There was developed an extensible and flexible architecture which enables third-party developers to implement plug-ins. These plug-ins can share multimedia data from a particular source.

In the first step, a BRisa plug-in developer creates a plug-in file and puts it into the plug-in directory. Then, the developer implements a Python class assigning a plug-in name and implementing the methods *load*, *unload*, *execute,* and *browse*. In our case study, we developed a set of plug-ins that support browsing of multimedia contents.

Basically, three plug-ins have been developed: one to gather multimedia content from the local file system, a second to retrieve YouTube videos from a personal user account, and a third for retrieval of pictures from a personal FlickR user account. Each plug-in has its own data source. For instance, the file system plug-in data source is the disk, and for the YouTube plug-in its data source is the list of available videos from the *youtube.com* website. Regardless of the plug-in selected by the BRisa Media Server, as illustrated in Fig. 6, the browser method invokes the Recommendation Subsystem at the UPServer. It recommends multimedia items to the user based on the user preferences. The whole process can be summarized as follows:

1. the BRisa Media Server receives a browsing request from the control point, also provided in the BRisa Application package. Listing 9 presents an example of the UPnP *browse* action. Lines 6 and 7 specify the UPServer UDN and the user UUID, respectively;
2. the BRisa Media Server selects the proper plug-in to handle the request;
3. a plug-in *i* is selected to invoke the recommendation subsystem in order to determine which multimedia content will be retrieved from its specific repository;
4. the Recommendation Subsystem at the User Profile Server executes the recommendation mechanism as described before and sends back to plug-in *i* the results of the recommendation process;
5. the plug-in *i* accesses its respective data source and collects the filtered items;
6. the selected items are sent back to the BRisa Media Server;
7. the set of items is formatted into a Browsing Response considering the UPnP A/V specification and it is returned to the user control point.

5.2 Case study 2—exploring UPnP-UP authorization

A prototype for providing user authorization on the *CreateJob* service of a UPnP printer has also been developed. As a proof of concept, we have used CUPS, which is a standard-based, open source printing system developed by Apple for

```
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
    <s:Header>
        <Session>
            <udn>c78dc82a-11c4-4d7c-bc68-9d7404442340</udn>
            <uuid>497277f0-8f57-4d6e-96c9-c6ef17cbe34e</uuid>
        </Session>
    </s:Header>
    <s:Body>
      <ns0:Browse xmlns:ns0=
"urn:schemas-brisaupnp-com:service:Multimedia:1">*</ns0:Browse>
    </s:Body>
</s:Envelope>
```

Listing 9 Browse Action Request. User UUID available in header tag allows UPnP end-points to identify who is requesting for multimedia items

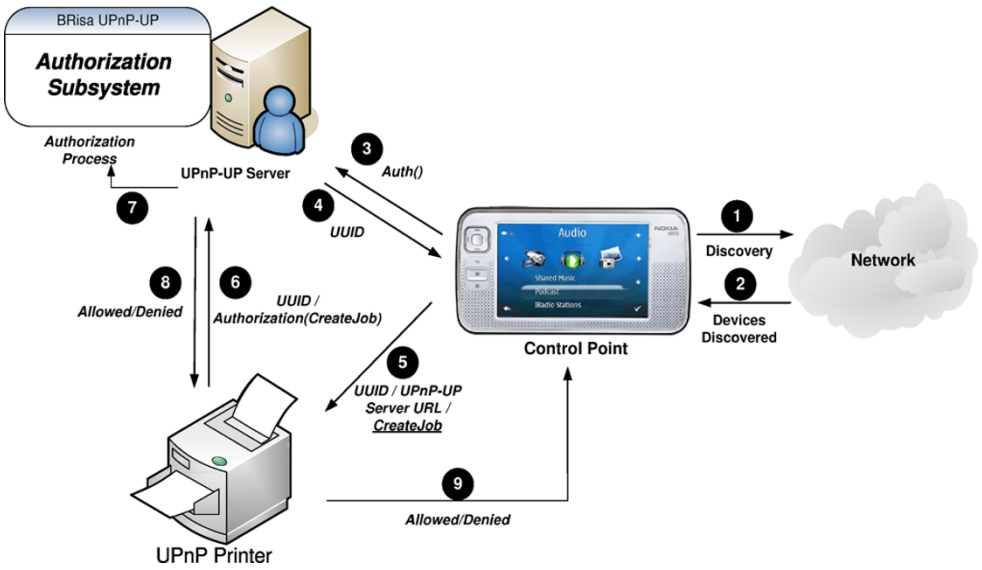

Fig. 7 UPnP-UP authorization process for UPnP printer device access control

MacOS and other UNIX-like operating systems. We used the CUPS API to implement a UPnP module and driver. The CUPS API provides the convenience functions needed to support applications, filters, printer drivers, and back-ends that need to interface with the CUPS scheduler. Our UPnP module and driver for CUPS are responsible for some tasks such as:

1. advertise the printer as a UPnP device and handle *CreateJob* service;
2. find and connect to the UPServer, as well as retrieve user profiles and ACLs for a specific user and device;
3. handle *CreateJob* requests sent by the UPnP control points and check whether the user is allowed to create a printing job.

Figure 7 illustrates the basic interaction between the UPnP Control Point and the two UPnP devices used in our case study. The XACML messages used in this case were discussed in the previous section (Listings 3, 4, and 5). Basically, after authenticating in the UPServer, the control point tries to access the *createJob* service (Step 5). Our UPnP-UP-compatible printer device invokes the *authorization* service of the UPServer to verify the current user access permissions and, finally, receives a response from the UP-Server. This response allows or denies the access to the *CreateJob* service (Steps 8 and 9).

6 Experiments and results

Since the UPnP-UP extension requires additional UPnP requests for user authentication, authorization, and user profile sharing, it is important to discuss the amount of extra data sent by a UPnP device (or a set of them) to the network in order to achieve the proposed solution. Consider as the *UPnP-UP startup* phase the process of establishing a user authentication and invoking any UPnP action of any UPnP device for the first time. Thus, let d be the total number of UPnP devices, and n be the total number of UPnP control points at a given time t. So, the maximum number of UPnP requests during startup at a given time t is $(2 * d + 1) * n$ requests. For instance, if there are 5 UPnP devices ($d = 5$), one UPServer, and one control point ($n = 1$) at a given instant t, the UPnP-UP startup requires $(2 * 5 + 1) * 1 = 11$ UPnP

requests. Figure 8 depicts the number of UPnP actions for $n = 1$ in both the UPnP and UPnP-UP startup processes.

After the startup process, UPnP-UP behaves in the same way as the current UPnP specification, although it adds new data to the UPnP requests in order to identify the current user. This extra data size is about 16 bytes for each UPnP action, which corresponds to the UUID size attached to the SOAP header, as described in Listing 6. Moreover, to evaluate our proposed UPnP-UP extension, a series of experiments has been performed. The first set of experiments aims to observe the impact of the user authentication and authorization response time in UPnP networks. The experiments were performed by considering the number of UPnP control points and devices in a local area network with traffic isolated to only the UPnP peers defined as follows: three desktop PCs (Intel Core 2 processor, E7400 2.8 GHz, and 1 GB of memory) running UPnP device instances; and three Nokia N810 mobile devices running UPnP control point instances.

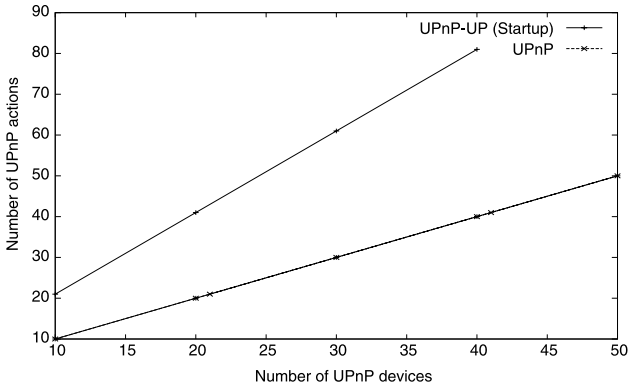

Fig. 8 The number of UPnP-UP requests during startup

The results are described in Tables 1, 2, and 3. Basically, we considered the scenario depicted in Fig. 7, assuming 10, 20, and 30 UPnP printer devices and 1, 3, 5, and 10 UPnP control points. The tables describe the average times for control points to authenticate a user in a UPServer and request the *CreateJob* service available by the 10, 20, or 30 UPnP printer devices for the first time. In addition, the same number of control points and UPnP printer devices was considered when using the current UPnP specification, allowing observation of the impact on the time for providing user authentication and authorization in UPnP networks. The average times are in seconds and, based on a *z-distribution*, Tables 1, 2, and 3 also present the 95% confidence intervals for each set of collected samples.

As expected, due to the processing time and the network congestion, the average time increases when new control points and UPnP printer devices are available in the network. The growth of processing time is more evident in UPnP-UP architecture due to the XML processing of the user profiles and access control policies. Moreover, control points must

Table 1 Average time for UPnP-UP startup process with 10 UPnP printer devices

Control points	10 UPnP printer devices	
	UPnP	UPnP-UP
1	$\mu = 0.108356$ s 0.103003 s $\leq t$ (s) $\leq$ 0.113710 s	$\mu = 0.778184$ s 0.582597 s $\leq t$ (s) $\leq$ 0.973770 s
3	$\mu = 0.158032$ s 0.143945 s $\leq t$ (s) $\leq$ 0.172119 s	$\mu = 1.138857$ s 1.066515 s $\leq t$ (s) $\leq$ 1.211200 s
5	$\mu = 0.130992$ s 0.089611 s $\leq t$ (s) $\leq$ 0.172372 s	$\mu = 1.190738$ s 1.053702$s \leq t$ (s) $\leq$ 1.327775 s
10	$\mu = 0.180954$ s 0.159699 s $\leq t$ (s) $\leq$ 0.202209 s	$\mu = 2.156792$ s 1.959179 s $\leq t$ (s) $\leq$ 2.354406 s

Table 2 Average time for UPnP-UP startup process with 20 UPnP printer devices

Control points	20 UPnP printer devices	
	UPnP	UPnP-UP
1	$\mu = 0.219176$ s 0.211158 s $\leq t$ (s) $\leq$ 0.227194 s	$\mu = 1.384315$ s 1.185720 s $\leq t$ (s) $\leq$ 1.582909 s
3	$\mu = 0.274146$ s 0.260461 s $\leq t$ (s) $\leq$ 0.287831 s	$\mu = 2.220593$ s 1.993325 s $\leq t$ (s) $\leq$ 2.447861 s
5	$\mu = 0.297919$ s 0.256718 s $\leq t$ (s) $\leq$ 0.339120 s	$\mu = 3.095876$ s 2.986757 s $\leq t$ (s) $\leq$ 4.332378 s
10	$\mu = 0.560082$ s 0.527621 s $\leq t$ (s) $\leq$ 0.592544 s	$\mu = 5.186616$ s 4.969243 s $\leq t$ (s) $\leq$ 5.403990 s

Table 3 Average time for UPnP-UP startup process with 30 UPnP printer devices

Control points	30 UPnP printer devices	
	UPnP	UPnP-UP
1	$\mu = 0.530408$ s 0.452629 s $\leq t$ (s) $\leq$ 0.608187 s	$\mu = 1.999712$ s 1.952061 s $\leq t$ (s) $\leq$ 2.047364 s
3	$\mu = 0.559336$ s 0.452067 s $\leq t$ (s) $\leq$ 0.666604 s	$\mu = 2.839887$ s 2.753026 s $\leq t$ (s) $\leq$ 2.926748 s
5	$\mu = 0.574434$ s 0.558496 s $\leq t$ (s) $\leq$ 0.590372 s	$\mu = 4.119828$ s 3.907278 s $\leq t$ (s) $\leq$ 4.332378 s
10	$\mu = 0.847150$ s 0.806331 s $\leq t$ (s) $\leq$ 0.887970 s	$\mu = 6.519950$ s 6.304011 s $\leq t$ (s) $\leq$ 6.735888 s

check if the target UPnP device is UPnP-UP compatible by checking the *UP* field. From the user perspective, it is important to note that the average times using UPnP-UP are still very acceptable even when the average time hits 5 seconds above the current UPnP specification (10 control points and 30 UPnP printer devices).

7 Discussion and future work

Recently, UPnP has attained worldwide recognition by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) due to the increasing number of new UPnP-based applications [33]. Both institutions have acknowledged the UPnP architecture position as the leading technology for devices and services discovery, and control of networked devices.

Pervasive applications require mechanisms to acquire and model user activities and profiles. In addition, information from the environment is also interesting for providing context-aware applications. Before UPnP-UP, the UPnP technology had no user authentication and authorization specification. Thus, UPnP-based applications could not provide user-based access control and customized services that leverage UPnP-based technologies. The UPnP support for security deals with device information, which is not enough to acquire user information.

In order to provide user authentication and authorization in UPnP networks, the UPnP-UP extension requires the addition of a new UPnP device specification called UPServer. The UPServer is responsible for managing the authentication and the authorization process, and also providing the services to store, update, delete, and remove access control policies. For instance, the DNS Injection attack can be solved by creating a set of access control policies that establish the range of allowed ports to be mapped (above 1024) by users. However, these policies are maintained in the server, requiring IGD devices to access them every time a user tries to request a port mapping. As a result, it is also required to distribute these policies along the available devices. For instance, UPServer would provide a UPnP-based service for collecting policies based on the device type or service.

From a network infrastructure point of view, the UPServer is a single point of failure. However, a distributed authentication and authorization mechanism is more complex and hard to manage and maintain due to the input and output of devices. To overcome such drawbacks, it is also possible to provide more than one UPServer in the network and provide load balance mechanisms using IPv6 anycast, for instance. Also, devices that will support UPnP-UP can be considered as "managed" or "unmanaged" devices. That is, if no UPServer device is found in the local network, it still can allow the access to a set of selected "free" services, if possible.

The proposed solution still has some problems. First, in spite of the possibility of changing the user UUID at a given time interval, an attacker can obtain other user UUIDs and renew the authentication session. We are currently investigating a solution to provide mechanisms to build a set of user token IDs (UUIDs). In this solution, each user has a set of identifiers that are generated based on a Lamport hash chain model [34]. Each identifier can be used only once, for each UPnP device to which the user has access. This mechanism avoids some kinds of attacks, such as *UUID spoofing*. In addition, hash functions have low computational costs, which can be executed by resource-constrained devices, such as cellphones. Another important weakness of the current solution is related to replay (or playback) attacks [35, 36]; a network attacker is still allowed to maliciously or fraudulently resubmit a valid request. Within the context of the UPnP-UP network architecture, an access to a UPnP printer device (which takes valid UUID) can be resent by another user, or attacker. As a result, the attacker can gain access to the device on behalf of another user.

As current and future work, the use of OWL (Ontology Web Language) [37] has been investigated as a standard approach to describe user profiles, providing a better process of context interpretation and inference. Within this context, the

authors have also investigated its impact regarding performance, since UPnP devices can be resource-constrained and OWL requires high hardware performance. Another point of improvement concerns the authentication process, which has been evolved to provide different levels of security, data integrity, confidentiality, and prevention to replay attacks. Moreover, the authorization service is still being evaluated and more experiments must be performed. It is still responsible for 63% of the total time consumption presented in the experimental results.

8 Conclusion

This paper presented an extension of the UPnP technology to enable user authentication and authorization mechanisms for the UPnP connectivity architecture. The extension allows the development of customized UPnP applications and ensures access control for available resources for UPnP enabled networks. Besides extending UPnP to support user authentication, a new UPnP device profile, called *User Profile Server,* has been introduced. As a proof of concept, we presented two scenarios that show how the UPnP-UP extension can be applied in order to achieve customized and secure UPnP services. Moreover, some experimental results were presented in order to validate the UPnP-UP extension and show that, even with the addition of new UPnP devices (UP-Server) and UPnP actions, the average time that a control point takes to request for a UPnP job is still acceptable.

Nowadays, the available UPnP services that follow the UPnP specification cannot make use of user information. Because of this, UPnP devices are unable to dynamically adapt the services based on the user information. As a result, each service developer implements his own non-standard proprietary solution, making interoperability among available UPnP services more difficult for providing customized services. On the other hand, UPnP-UP is a research contribution for user authentication and authorization mechanisms in the UPnP networks, keeping the specification compatible with the current UPnP implementations and enabling a new set of applications to be deployed over a UPnP network. UPnP-UP extension has become possible due to the flexible and extensible UPnP design offered by the UPnP Forum, allowing the addition of new devices and services through a definition of artifacts such as XML file descriptions and SOAP Web Services.

References

1. Loureiro E, Ferreira G, Almeida H, Perkusich A (2007) Pervasive computing: what is it anyway? In: Lytras M, Naeve A (eds) Ubiquitous and pervasive knowledge and learning management: semantics, social networking and new media to their full potential, pp 1–34
2. Weiser M (1999) The computer for the 21st century. SIGMOBILE Mob Comput Commun Rev 3(3):3–11. doi:10.1145/329124.329126
3. Presser A, Farrel L (2008) UPnP device architecture. http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf. Last access on May, 2008
4. Kumaran I, Kumaran SI (2001) Jini technology: an overview. Prentice-Hall PTR, Upper Saddle River
5. Consortium S (1999) Salutation architecture specification. ftp://ftp.salutation.org/salutesa20e1a21.ps
6. Guttman E, Perkins C, Veizades J, Day M (1999) Service location protocol, version 2. RFC. http://tools.ietf.org/html/rfc2608
7. W3C (2007) Simple object access protocol. http://www.w3.org/TR/soap/
8. Langille G et al. (2008) Mediaserver:3 device template version 1.01. http://upnp.org/specs/av/UPnP-av-MediaServer-v3-Device.pdf. Last access on May, 2009
9. Guedes A, Santos D, do Nascimento J, Sales L, Perkusich A, Almeida H (2008) Set your multimedia application free with BRisa framework: an open source UPnP implementation for resource limited devices. In: 5th IEEE consumer communications and networking conference, 2008. CCNC 2008, pp 1257–1258 (10–12 January 2008). doi:10.1109/ccnc08.2007.297
10. Lin JC, Chen JM, Liu CH (2008) An automatic mechanism for adjusting validation function. AINAW, pp 602–607. 10.1109/WAINA.2008.89
11. Prakash Iyer UW (2001) Internetgatewaydevice:1 device template version 1.01. http://upnp.org/standardizeddcps/documents/UPnP_IGD_1.0.zip. Last access on May, 2009
12. Hengartner U, Steenkiste P (2004) Protecting access to people location information. In: Lecture notes in computer science, vol 2802. Springer, Berlin, pp 222–231
13. Robinson P, Beigl M (2004) Trust context spaces: an infrastructure for pervasive security in context-aware environments. In: Lecture notes in computer science, vol 2802. Springer, Berlin, pp 119–129
14. Kvarnstrom H, Hedbom H, Jonsson E (2004) Protecting security policies in ubiquitous environments using one-way functions. In: Lecture notes in computer science, vol 2802. Springer, Berlin, pp 71–85
15. Creese S, Goldsmith M, Roscoe B, Zakiuddin I (2004) Authentication for pervasive computing. In: Lecture notes in computer science, vol 2802. Springer, Berlin, pp 439–488
16. Klemets A, Da Costa B (2008) UPnP authentication and authorization patent. http://www.freepatentsonline.com/y2008/0092211.html
17. Karl J (2010) UPnP CDS USER PROFILE. http://www.patents.com/UPnP-CDS-USER-PROFILE-20100125907.html
18. Ellison C (2003) DeviceSecurity: 1 Service Template. http://www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf. Last access on December, 2008
19. Ellison C (2003) SecurityConsole: 1 service template. http://www.upnp.org/standardizeddcps/documents/SecurityConsole_1.0cc.pdf. Last access on December, 2008
20. Nakajima T (2003) Pervasive servers: a framework for creating a society of appliances. Pers Ubiquitous Comput 7(3–4):182–188. doi:10.1007/s00779-003-0222-2
21. Chen W, Kuo SY, Chao HC (2009) Service integration with UPnP agent for an ubiquitous home environment. Inf Syst Front 11(5):483–490. doi:10.1007/s10796-008-9122-3
22. Sahm C, Langels HJ (2003) Dimmable light device template. http://www.upnp.org/standardizeddcps/documents/DimmableLight1.0cc.pdf. Last access on May, 2008
23. Kim K, Ko H, Choi W, Lee E, Kim U (2008) A collaborative access control based on XACML in pervasive environments. In: International conference on convergence and hybrid information technology, 2008. ICHIT'08, pp 7–13

24. Rahaman MA, Schaad A, Rits M (2006) Towards secure SOAP message exchange in a SOA. In: SWS'06: proceedings of the 3rd ACM workshop on secure web services. ACM, New York, pp 77–84. doi:10.1145/1180367.1180382
25. Snyder RM (2007) Security programming using python: man-in-the-middle attacks. In: InfoSecCD'07: proceedings of the 4th annual conference on information security curriculum development. ACM, New York, pp 1–6. doi:10.1145/1409908.1409911
26. Hashemipour S, Ali M (2004) MPEG-21 & DIDL: dawn of a new multimedia EVA. In: IEEE international symposium on consumer electronics, 2004, pp 91–95
27. Balabanovic M, Shoham Y (1997) FAB: content-based, collaborative recommendation. Commun ACM 40:66–72
28. Im I, Hars A (2007) Does a one-size recommendation system fit all? The effectiveness of collaborative filtering based recommendation systems across different domains and search modes. ACM Trans Inf Syst TOIS 26(1):4. doi:10.1145/1292591.1292595
29. Deshpande M, Karypis G (2004) Item-based top-*n* recommendation algorithms. ACM Trans Inf Syst 22(1):143–177. doi:10.1145/963770.963776
30. Benesty J, Chen J, Huang Y (2008) On the importance of the Pearson correlation coefficient in noise reduction. IEEE Trans Audio Speech Lang Process 16(4):757–765. 10.1109/TASL.2008.919072
31. Minker J (1977) Information storage and retrieval: a survey and functional description. SIGIR Forum 12(2):12–108. doi:10.1145/1095515.1095516
32. Yantao Z, Jianbo T, Jiaqin W (2007) An improved TFIDF feature selection algorithm based on information entropy. In: Chinese control conference, 2007. CCC 2007, pp 312–315. doi:10.1109/CHICC.2006.4346845
33. Sherwin L (2009) UPnP specifications named international standard for device interoperability for IP-based network devices. innovation validated by record-breaking number of UPnP implementations in 2008. http://www.upnp.org/news/documents/UPnPForum_02052009.pdf. Last access on September, 2009
34. Lamport L (1981) Password authentication with insecure communication. Commun ACM 24(11):770–772. doi:10.1145/358790.358797
35. Malladi S, Alves-Foss J, Heckendorn RB (2002) On preventing replay attacks on security protocols. In: Proc international conference on security and management. CSREA Press, pp 77–83
36. Syverson P (1994) A taxonomy of replay attacks. In: Proceedings of the 7th IEEE computer security foundations workshop. Society Press, New York, pp 187–191
37. Yan Y, Zhang J, Yan M (2006) Ontology modeling for contract: using OWL to express semantic relations. In: 10th IEEE international enterprise distributed object computing conference, 2006. EDOC'06, pp 409–412. doi:10.1109/EDOC.2006.37