Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Unidade Acadêmica de Engenharia Elétrica

Laboratório de Sistemas Embarcados e Computação Pervasiva

Trabalho de Conclusão de Curso

Título:

Reconhecimento de Perfis de Usuários em Redes UPnP

Aluna:

Fernanda Baracuy da Cunha Campos

Orientadores:

Angelo Perkusich

Leandro Melo de Sales

Campina Grande, Outubro de 2008

Biblioteca Setorial do CDSA. Fevereiro de 2021.

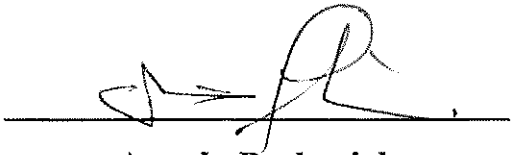
Sumé - PB

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Engenharia Elétrica
Laboratório de Sistemas Embarcados e Computação Pervasiva

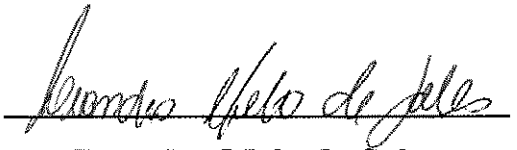
Título:

Reconhecimento de Perfis de Usuários em Redes UPnP

Fernanda Baracuy da Cunha Campos
Aluna



Angelo Perkusich
Orientador



Leandro Melo de Sales
Orientador

Glossário

DHCP: *Dynamic Host Configuration Protocol*

GENA: *General Event Notification Architecture*

HTTP: *Hypertext Transfer Protocol*

IP: *Internet Protocol*

SOAP: *Simple Object Access Protocol*

SSDP: *Simple Service Discovery Protocol*

TCP: *Transmission Control Protocol*

UDA: *UPnP Device Architecture*

UDP: *User Datagram Protocol*

URI: *Uniform Resource Identifiers*

URL: *Uniform Resource Locator*

XML: *Extensible Markup Language*

Sumário

LISTA DE FIGURAS

LISTA DE CÓDIGOS

1. Introdução

Nos últimos anos a disponibilização de diferentes aplicações no contexto de sistemas embarcados tem aumentado consideravelmente. Um dos motivos desse crescimento é o massivo incentivo da indústria através de investimentos para o desenvolvimento de dispositivos capazes de oferecer serviços de rede. Nesse sentido, diversas iniciativas da indústria de software têm resultado em modelos para dar suporte a aplicações.

O padrão UPnP (*Universal Plug-and-Play*) [3] é um conjunto de protocolos de rede de computadores utilizado para detectar dispositivos e disponibilizar seus respectivos serviços. É baseado no modelo de *Plug-and-Play* para periféricos, que permite conexão direta entre os diversos periféricos e um computador através de um componente de software chamado de driver (*device drivers*). A idéia é permitir que um dispositivo se conecte dinamicamente em uma rede, obtenha um endereço IP e anuncie suas características e serviços, ao passo que possibilite que este dispositivo aprenda sobre as características e serviços dos demais dispositivos conectados à rede, sem a necessidade de conhecer detalhes tanto de como se conectar à rede quanto de como aprender sobre os demais dispositivos. Portanto, o UPnP herda a idéia do modelo PnP, porém com as seguintes modificações: descoberta automática de dispositivos, ao invés de periféricos, e acesso aos respectivos serviços desses dispositivos através de uma interface genérica e padronizada, ao invés de fazer uso dos *device drivers*, os quais são substituídos por protocolos de rede.

Desse modo o padrão UPnP se torna uma solução viável na disponibilização de serviços pervasivos, tais como configuração automática e descoberta padronizada de dispositivos, oferecendo mecanismos para o compartilhamento e acesso a recursos entre diferentes tipos de dispositivos e em diferentes tipos de redes de computadores.

No contexto da implementação que será detalhada nesse trabalho, utilizou-se o BRisa UPnP framework [1] que é um arcabouço que permite que usuários descubram dispositivos multimídia, compartilhem, pesquisem e reproduzam conteúdos multimídia através de uma rede local ou através da Internet. Ele foi desenvolvido usando as especificações UPnP, as quais utilizam padrões bem estabelecidos na Internet como o HTTP, o UDP e o SOAP [7].

1. Objetivo

Tendo em vista o propósito UPnP, equipamentos com a essa capacidade podem dinamicamente entrar em uma rede transmitir seus recursos e tomar conhecimento da presença e dos recursos de outros dispositivos, sem que a rede tenha informações específicas de quem está se conectando a ela.

Apesar do padrão UPnP oferecer soluções para descoberta e disponibilização de serviços, ele não possibilita o gerenciamento de informações de contexto, dificultando a personalização de serviços e informações .

Como escopo, UPnP é suficientemente grande para abranger muitos cenários existentes, incluindo automação doméstica, impressão e imagem, entretenimento de áudio/vídeo, utensílios de cozinha, redes de automóveis e redes adjacentes em locais públicos.

O objetivo do presente projeto é descrever a implementação de uma extensão do protocolo UPnP para que seja possível o reconhecimento do perfil do usuário na rede e assim dar suporte a serviços personalizados.

Isso conduz ao desenvolvimento de um servidor UPnP para armazenamento de dados do usuário e a modificação do processo de descoberta UPnP, porém, mantendo a compatibilidade com a especificação atual desse padrão.

3. Tecnologias Relacionadas

Nessa seção são apresentados os principais conceitos e ferramentas utilizados no desenvolvimento deste trabalho.

3.1. UPnP

A tecnologia UPnP define uma arquitetura para conectividade pervasiva P2P entre aparelhos inteligentes, dispositivos móveis e PCs de diversos fabricantes. Ela é projetada para proporcionar fácil usabilidade, flexibilidade, conectividade baseada em padrões de rede *ad-hoc* em ambientes domiciliares, pequenos negócios ou espaços públicos. A tecnologia UPnP fornece uma arquitetura de rede aberta utilizando TCP/IP e as tecnologia web.

A UDA (UPnP Device Architecture) [2] é mais do que uma simples extensão do modelo de periféricos *plug and play*. Ela é projetada para suportar a configuração zero, redes "invisíveis" e descoberta automática de uma amplitude de categorias de dispositivos. Isso significa que um dispositivo pode entrar dinamicamente na rede, obter um endereço IP, anunciar as suas capacidades e conhecer sobre a presença e capacidade de outros dispositivos. Finalmente ele pode deixar a rede automaticamente sem deixar nenhum estado indesejado para trás.

Esta tecnologia possui uma arquitetura de rede distribuída e aberta, a qual tem como base os protocolos padrões da Internet como o TCP, UDP, IP, HTTP e SOAP, permitindo o controle e a transferência de dados entre dispositivos conectados a rede. Os contratos são declarativos, expressos em XML e comunicados via HTTP.

A arquitetura do dispositivo UPnP define os protocolos para comunicação entre controladores, ou *Control Points* e dispositivos. Para os mecanismos de descoberta, descrição, controle, evento e apresentação, a arquitetura do dispositivo UPnP utiliza a pilha de protocolos ilustrada na Figura 1.

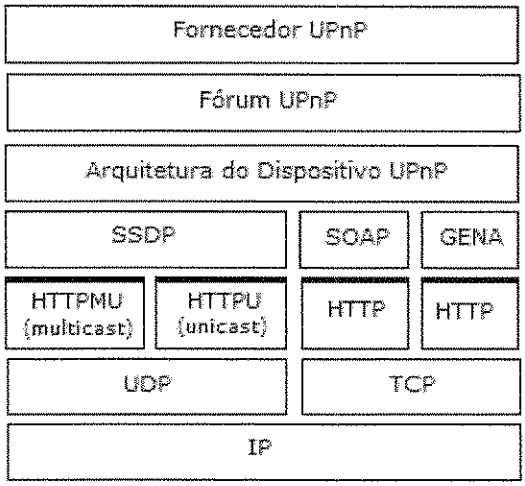


Figura 1 – Pilha de protocolos utilizada pela arquitetura do dispositivo UPnP.

No nível mais alto da pilha estão contidas informações UPnP específicas dos fornecedores sobre seus dispositivos. Deslocando-se para baixo na pilha, o conteúdo do fornecedor é complementado por informações definidas pelo Fórum UPnP [12]. Estas informações acima são armazenadas em protocolos específicos UPnP como o *Simple Service Discovery Protocol* (SSDP) e o *General Event Notification Architecture* (GENA). As mensagens acima são *unicast* ou *multicast* enviadas via HTTP executando sobre UDP, ou o padrão HTTP executando sobre TCP. Por último, todas as mensagens são enviadas sobre IP.

Duas classificações gerais de dispositivos são definidas pela arquitetura UPnP: dispositivos controlados (ou simplesmente "dispositivos") e *Control Points*. Um dispositivo controlado funciona como um servidor, respondendo aos pedidos de *Control Points*. Ambos *Control Points* e dispositivos controlados podem ser implementados em diferentes plataformas incluindo computadores pessoais e sistemas embarcados. Múltiplos dispositivos, *Control Points* ou ambos podem operar numa rede simultaneamente.

3.1.1. Endereçamento

A fundação para a rede UPnP é o endereçamento IP. Cada dispositivo deve ter um *Dynamic Host Configuration Protocol* (DHCP) e procurar por um servidor DHCP quando o dispositivo é conectado a rede. Se um servidor DHCP está disponível, ou seja, a rede é gerenciada, o dispositivo deve usar o endereço IP atribuído a ele. Caso contrário, o dispositivo deve usar o Auto IP para obter um endereço, que define como um dispositivo

inteligentemente escolhe um endereço IP de um conjunto de endereços reservados e está habilitado para se juntar facilmente a redes gerenciadas e não gerenciadas.

Um dispositivo que suporta Auto IP e está configurado para a atribuição dinâmica de endereçamento, começa por pedir um endereço IP via DHCP enviando uma mensagem DHCP DISCOVER. O cliente DHCP fica escutando o DHCP OFFERS, se ele receber o pedido o dispositivo continua o processo de atribuição dinâmica de endereço, caso contrário, poderá usar a auto-configuração IP.

Se for utilizado o Auto IP, o endereço deve então ser testado para determinar se o endereço está em uso. Os endereços são distribuídos aleatoriamente para evitar colisões.

Depois que um dispositivo tem um endereço IP válido para a rede ele pode ser localizado e referenciado na rede através desse endereço.

3.1.2. Descoberta

Quando um dispositivo é adicionado à rede, o protocolo de descoberta UPnP permite que o dispositivos anunciem seus serviços para *Control Points* na rede. De forma similar, quando um *Control Point* é adicionado à rede, o protocolo de descoberta UPnP permite que o *Control Point,* através de mensagens multicast de descoberta, procure por dispositivos de interesse na rede. A diferença fundamental em ambos os casos é a mensagem de descoberta que contem informações essenciais específicas sobre o dispositivo ou um dos seus serviços.

Todos os dispositivos devem ouvir e responder se algum dos seus dispositivos embutidos ou serviços correspondem aos critérios de pesquisa na mensagem de descoberta.

Se o dispositivo enviou uma mensagem anunciando seus serviços, ou se respondeu a um *Control Point*. Em qualquer caso, se o ponto de controle está interessado em um dispositivo e quer saber mais sobre ele, utiliza informações contidas na mensagem de descoberta para enviar uma mensagem de descrição.

Quando um dispositivo é adicionado à rede ele publica seus serviços para os *Control Point*. Ele faz isso através da mensagem de descoberta multicast para o endereço padrão e porta (239.255.255.250: 1900). *Control Point* escutam esta porta para detectar quando novas capacidades estão disponíveis na rede. As mensagens devem incluir duração até a propaganda expirar; se o dispositivo continuar disponível, a propaganda deverá ser re-enviada com nova duração. Se o dispositivo torna-se indisponível, o dispositivo deve

explicitamente cancelar suas propagandas, mas se o dispositivo não é capaz de fazer isso, a propaganda irá expirar por conta própria.

A mensagem de descoberta contém informações tais como URL para a descrição do dispositivo, seu tipo e identificador. O anúncio segue o formato descrito no Código 1:

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.250: 1900
3 CACHE-CONTROL: <Max - age>
4 LOCATION: URL para descrição do dispositivo
5 NT: "dispositivo alvo"
6 NTS: ssdp: alive
7 SERVER: OS/Version
8 USN: UUID do anúncio
```

Código 1 - Anúncio de descoberta UPnP

Como a Figura 2, toda mensagem de descoberta contém o método NOTIFY, para enviar notificações e eventos, com os seguinte campos de cabeçalho: *Host*, o qual associa o endereço multicast reservado para o protocolo SSDP; *Cache-Control*, refere-se ao tempo em que o anúncio é válido, após este período os *Control Point* devem assumir que o dispositivo ou serviço não está mais disponível; *Location*, o qual é atribuído uma URL para a descrição do dispositivo principal; *NT* responsável pelo tipo de notificação; *NTS* que descreve o subtipo de anúncio (podendo ser do tipo ssdp:alive ou ssdp:bye-bye); *Server* que contém o nome e a versão do Sistema Operacional e o nome e versão do produto. Todos esses itens são especificados pelo fabricante do dispositivo.

Quando o *Control Point* é adicionado a rede e busca por dispositivos, a requisição de busca segue o formato especificado no Código 2.

```
1 M_SEARCH * HTTP/1.1
2 HOST: 239.255.250: 1900
3 MAN: ssdp: discover
4 MX: tempo de esperar resposta da requisição
5 ST: "dispositivo alvo"
```

Código 2 - Requisição de descoberta UPnP

Para ser encontrado o dispositivo deve retornar uma resposta UDP para o endereço IP e porta que enviou o pedido multicast. Este dispositivo só responderá a requisição se uma dessas três possibilidades estiver no campo ST da requisição de busca do Código 2:

- "ssdp:all": Onde o *Control Point* requisita todos os dispositivos disponíveis na rede;
- "ssdp:rootdevice": Onde procura apenas por dispositivos principais;
- "uuid": Onde procura por um dispositivo UPnP específico. Nesse caso ele é seguido pelo identificador do dispositivo.

A resposta da requisição segue o formato especificado no Código 3 e será bastante útil no processo de descrição UPnP descrito na próxima subseção. Uma outra opção de resposta é utilizando o método NOTIFY , como mostrado no Código 1.

```
1  HTTP/1.1 200 OK
2  CACHE-CONTROL: <Max - age> segundos até o tempo de procura expirar
3  DATE: data do momento em que a requisição foi gerada
4  EXT:
5  LOCATION: URL da descrição do dispositivo
6  SERVER: OS/Version
7  ST: alvo de procura
8  USN: UUID buscado
```

Código 3 - Resposta da requisição de busca UPnP

3.1.3. Descrição

No momento que um *Control Point* descobre um dispositivo, ele ainda sabe muito pouco sobre o dispositivo. Para o *Control Point* aprender mais sobre o dispositivo e suas capacidades, ou interagir com o mesmo, ele deve recuperar a URL (*Uniform Resource Locator*) adquirida a partir do campo *Location* encapsulada na mensagem de anúncio (ver Código 1) ou resposta de descoberta (Ver Código 3). Dispositivos podem conter outros dispositivos, como também podem ter unidades funcionais ou serviços, portanto a Descrição UPnP para um dispositivo é subdividida em duas partes: *descrição do dispositivo* e *descrição do serviço*, ambas expressas em XML [11]. No primeiro caso são fornecidas as informações específicas do fabricante do dispositivo tais como: nome e número do modelo, número serial, nome do fabricante, URLs das páginas de Internet para

as especificações dos vendedores, etc. Para cada serviço presente no dispositivo são encapsuladas informações como: tipo de serviço, nome, URL para a descrição do serviço, URL para controle e URL para evento.

3.1.4. Controle

Após a etapa de descrição, o *Control Point* pode enviar ações para um serviço do dispositivo. Para isto, ele invoca remotamente esses serviços enviando mensagens de controle para a URL do serviço disponibilizada na etapa de descrição. Mensagens de controle são expressas em XML usando o *Simple Object Access Protocol* (SOAP) [7]. Como função das chamadas, em resposta à mensagem de controle, o serviço retorna alguns valores específicos da ação. Os efeitos da ação, se houver algum, são modelados pelas mudanças nas variáveis que descrevem o estado do serviço em tempo de execução.

3.1.5. Evento

Uma descrição UPnP de um serviço contém uma lista de ações para os quais o serviço responde e uma lista de variáveis que modelam o estado do serviço em tempo de execução. O serviço publica atualizações quando ocorrem mudanças nas variáveis, e um *Control Point* pode se inscrever para receber estas informações. Por exemplo, uma chamada remota ao serviço "*Play*" do dispositivo de reprodução de Áudio/Vídeo modificaria o estado da mídia de "*Pause*" para "*Reproduzindo*". Através de mensagens de *evento* os *Control Points* são informados sobre essas mudanças e assim atualizar variáveis correlacionadas a esse novo estado.

3.1.6. Apresentação

Se um dispositivo possui uma URL para apresentação, então um *Control Point* pode recuperar uma página usando esta URL, carregar a página em um *browser*, e dependendo dos serviços disponibilizados pela página, permitir um usuário controlar um dispositivo e/ou visualizar o status do mesmo. Este mecanismo de apresentação substitui a necessidade de chamadas SOAP, ou seja, o *Control Point* acessa o dispositivo apenas via URL.

3.2. A Linguagem de Marcação XML

O XML (Extensible Markup Language) [11] é uma linguagem de marcação, como o HTML, com os seguintes objetivos básicos:

- Prover o transporte de documentos através da Web de forma independente de sistemas operacionais ou formatos de arquivos,
- Suportar uma grande gama de aplicações, permitindo a definição de elementos pelo usuário (ou aplicação) para estruturar o documento,
- Facilitar a análise de documentos XML por programas,
- Ter uma especificação formal para a marcação de documentos.

Um arquivo XML deve conter pelo menos um elemento, mas pode conter mais de um elemento aninhado dentro dele. Os elementos podem ser vazio, texto, outros elementos ou texto com outros elementos e devem estar fechados (tag inicial < > e tag final < >). Além disso, os elementos podem possuir atributos, que servem para dar informações adicionais sobre o próprio elemento e devem vir sempre entre aspas (simples ou duplas).

XML não possui um conjunto pre-definido de tags ou elementos. Por isto pode ser definido livremente de acordo com o domínio dos dados e da aplicação, o que traz uma maior autonomia ao usuário na criação e manutenção de arquivos XML.

No Código 4, a seguir, temos um exemplo de um arquivo XML.

```
1  <?xml version = "1.0"  encoding = "ISO-8859-1"?>
2     <pedidos>
3        <pedido id = 1  cliente = "Fernanda Baracuy">
4           <itemDescricao> 10E5T </itemDescricao>
5           <quantidade> 10 </quantidade>
6        </pedido>
7        <pedido id = 2  cliente = "Daniel Campos">
8           <itemDescricao> 85H8T </itemDescricao>
9           <quantidade> 28</quantidade>
10       </pedido>
11    </pedidos>
```

Código 4 - Exemplo de uma arquivo XML

Por oferecer uma maior simplicidade e flexibilidade na troca de dados na Internet, o uso dos documentos XML tem aumentado consideravelmente. Um exemplo disso são tecnologias como UPnP, que fazem uso de documentos XML para representação e troca de dados.

Para manter a compatibilidade com a especificação atual e a facilidade de uso, a implementação da extensão do protocolo UPnP, mostrada neste trabalho, continuará fazendo uso de documentos XML devido a sua eficácia na troca de dados entre diferentes sistemas.

3.3. Computação Pervasiva

A computação pervasiva [6] é a interação transparente homem-máquina, permitindo que dispositivos e serviços façam, cada vez mais, parte do cotidiano das pessoas. Transparência ao usuário significa que a interação com o sistema de computação pervasiva é feita sem que o usuário tenha ciência disso.

A computação pervasiva altera ou processa o ambiente em função dos contextos, alteração destes ou da necessidade de recursos. Por exemplo, imagine um carro que liga o limpador de pára-brisa quando começa a chover e usa o seu sistema de som como viva voz para o celular do dono. A partir desse conceito, podemos imaginar portas se destrancando para o dono da casa. Equipamentos fazendo auto-testes e solicitando consertos automaticamente.

Desse modo, a computação pervasiva é definida como a computação que está em qualquer lugar, a qualquer momento e possui mobilidade. Para que isso seja possível, o sistema deve obter informações do ambiente e do usuário. Isto é feito com o uso de sensores e de informações de perfil de usuário.

3.4. Brisa

O BRisa [1] é um *framework* que permite que usuários descubram dispositivos multimídia, compartilhem, pesquisem e reproduzam conteúdos multimídia através de uma rede local ou da Internet. Assim é possível que *Control Points* naveguem por arquivos multimídia armazenados em um servidor e os reproduzam em dispositivos disponíveis na rede.

Como mostrado na Figura 2, o BRisa é composto por três entidades: o *Media Server* UPnP, o *Media Renderer* UPnP e o *Control Point* UPnP.

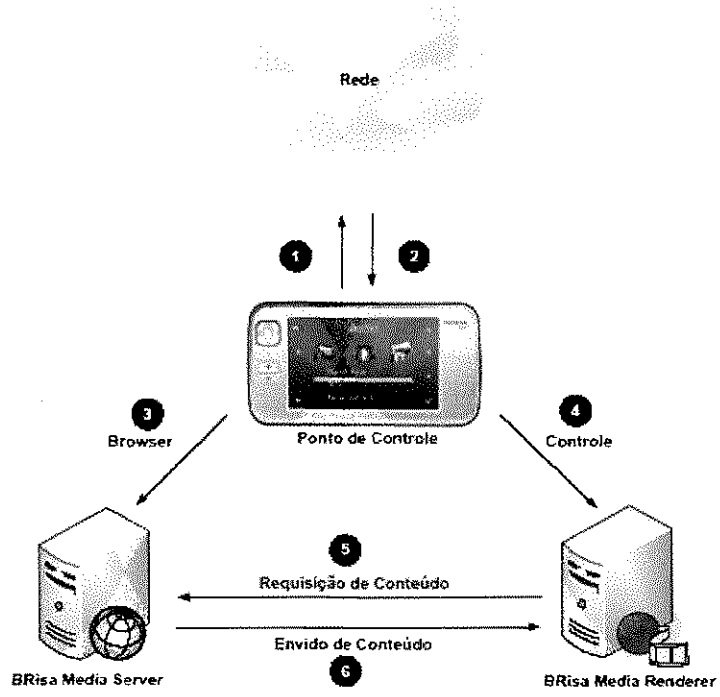


Figura 2 – Comunicação entre os dispositivos UPnP e o ponto de controle.

O Media Server é responsável pelo compartilhamento de arquivos multimídia como áudio, vídeo e imagem, além de itens multimídia acessados remotamente. Estes itens multimídia podem estar armazenados no sistema de arquivos local ou serem acessados de forma remota na Internet. Por sua vez, o Media Renderer é responsável pela reprodução dos arquivos multimídia armazenados e compartilhados pelos *Media Servers* UPnP.

Quando os servidores de mídia e os dispositivos de reprodução são descobertos pelo Control Point através de mensagens *multicast* ele passa a conhecer todos os serviços oferecidos por eles e assim ele pode navegar por mídias disponíveis no servidor de mídia e escolher um dos dispositivos de reprodução para reproduzir o conteúdo.

Como o BRisa foi desenvolvido usando as especificações UPnP, as quais utilizam padrões bem estabelecidos na Internet como o HTTP, o UDP e o SOAP, ele não possibilita o gerenciamento de informações de contexto, dificultando a personalização de serviços e informações .

4. Implementação

Com o objetivo de habilitar perfis de usuários para o padrão UPnP é apresentado a seguir a descrição da implementação de uma extensão UPnP-UP (acrônimo de Universal Plug and Play – User Profile). Para que fosse possível a realização de testes, toda a implementação foi feita utilizando o Brisa UPnP framework.

Foi necessário o desenvolvimento de um servidor UPnP, denominado de Servidor de Perfis UPnP-UP, para armazenamento de dados do usuário, tais como *nome completo*, *login* e *senha* e dados específicos referentes a especificação UPnP, como por exemplo *Áudio* e *Vídeo*. Para a implementação do UPnP-UP também foi necessário uma modificação do processo de descoberta UPnP, porém, mantendo a compatibilidade com a especificação atual desse padrão.

4.1 Servidor de Perfis UPnP-UP

Como todos os dispositivos UPnP disponibilizam um arquivo XML descrevendo as características e serviços providos pelo dispositivo, foi criado dentro dos *devices* do BRisa um novo dispositivo denominado *userProfile_sever* onde é criado o XML desse novo dispositivo descrevendo informações básicas do dispositivo e os serviços disponibilizados por ele.

O Código 5, a seguir, apresenta a descrição do *userProfile_sever* utilizando a mesma estrutura dos dispositivos UPnP.

```
1  <root xmlns = "urn : shemas-upnp-up-org: up-1-0">
2     <specVersion>
3        <major>1</major>
4        <minor>0</minor>
5     </specVersion>
6     <URLBase> base url value </ URLBase>
7      <device>
8        <deviceType>
9           urn : shemas-upnp-org : device : UPServer : 1
10       </deviceType>
11       <friendlyName> User Profile Server </friendlyName>
12       <modelName> Model Name  </modelName>
13       <modelNumber> 0.6 <modelNumber>
14       <modelURL> The Model URL </modelURL>
15       <serialNumber> 00000006 </serialNumber>
16       <serviceList>
17          <service>
18             <serviceType>
19                urn : shemas-upnp-org : service : UPAuthentication: 1
20             </serviceType>
21             <serviceId>
22                urn : upnp-org : serviceID: 11
23             </serviceId>
24             <SCPDURL>/UPServices/up-authentication.wsdl</SCPDURL>
25             <controlURL>/UPServices/up-authentication/control</controlURL>
26             <eventURL>/UPServices/up-authentication/event</eventURL>
27          </service>
28          <service>
29             <serviceType>
30                urn : shemas-upnp-org : service : UPProfile: 1
31             </serviceType>
32             <serviceId>
33                urn : upnp-org : serviceID: 22
34             </serviceId>
35             <SCPDURL>/UPServices/up-profiles.wsdl</SCPDURL>
36             <controlURL>/UPServices/up-profiles/control</controlURL>
37             <eventURL>/UPServices/up-profiles/event</eventURL>
38          </service>
39       </serviceList>
40       <presentationURL> URL for presentation </presentationURL>
41    </device>
42 </root>
```

Código 5 – Descrição do servidor de perfis para o padrão UPnP

Como se pode observar no Código 5 o arquivo XML disponibilizado pelo servidor UPnP-UP, além de informações básicas relacionadas ao dispositivo, define também os serviços oferecidos por ele.

Foi implementado no BRisa os serviços *UPAuthentication* e *UPProfile* descritos detalhadamente a seguir:

- *UPAuthentication*

O serviço *UPAuthentication* foi implementado com o objetivo de permitir a autenticação dos dispositivos através dos serviços descritos a seguir:

-auth: foi implementado para que o dispositivo se autentique no servidor UPnP-UP. Recebe como parâmetro o login e a senha do usuário e retorna ao dispositivo um identificador UUID de comunicação relacionado ao usuário autenticado;

-renewSession: esse serviço não foi implementado, mas a proposta é que após estabelecida a autenticação o servidor UPnP-UP considere válido o UUID, retornado pelo serviço *auth*, por um tempo pré-determinado. Após esse tempo seria necessário a chamada desse método para renovar a autenticação, permitindo assim que o servidor tenha conhecimento sobre a permanência do usuário na rede;

-logout: esse método também não foi implementado, porém a proposta de sua invocação é permite finalizar a seção do usuário no servidor UPnP-UP.

- *UPProfile*

A implementação do serviço UPProfile possibilitou a disponibilização de informações relacionadas ao perfil do usuário através dos métodos descritos a seguir:

-getUserInfo: foi implementado para disponibilizar informações relacionadas aos dados pessoais do usuário, tais como *nome* e *email*. Recebe como parâmetro o UUID gerado na autenticação e retorna um arquivo contendo os dados pessoais do usuário. O arquivo retornado é no formato XML, pois as informações devem ser estruturadas, "intercambiáveis", uniformes, extensíveis e padronizadas. A descrição é mostrada no Código 6 a seguir:

```
1  <root xmlns = "urn : shemas-upnp-up-org: up-1-0">
2      <specVersion>
3          <major>1</major>
4          <minor>0</minor>
5      </specVersion>
6      <userInfo>
7          <userId> user ID at UPnP-UP Server </userId>
8          <fullName> user full name </fullName>
9          <userName> username@upnp-up.org </userName>
10         <email> user email </email>
11         {...}
12     </userInfo>
13 </root>
```

Código 6 – Dados pessoais do usuário UPnP-UP

-getProfile: esse método foi implementado para disponibilizar o perfil do usuário. Recebe como parâmetro o UUID gerado da autenticação e o número da categoria desejada (ex: 1, para a especificação A/V) e retorna um arquivo no formato XML contendo as preferências do usuário para aquela categoria. No Código 7 pode-se observar uma estrutura básica para essa especificação.

```
1  <root xmlns = "urn : shemas-upnp-up-org: up-1-0">
2      <specVersion>
3          <major>1</major>
4          <minor>0</minor>
5      </specVersion>
6      <userProfile id = " UserId">
7          <upnpCategoryList>
8              <category id = " CategoryId"  name =  "CategoryName">
9                  {...}
10             </category>
11         </upnpCategoryList>
12     </userProfile>
13 </root>
```

Código 7 - Perfil do usuário UPnP-UP

4.2 Armazenamento do perfil do usuário

A solução para a inserção de dados no servidor UPnP-UP foi a utilização do SQLLite [9], uma biblioteca que implementa um banco de dados relacional, embutido e sem configurações.

Como todo o trabalho foi desenvolvido na linguagem de programação Python [10], foi utilizado como suporte para acesso ao banco de dados o SQLObject, uma biblioteca de *mapeamento objeto-relacional*, que permite converter linhas de tabelas em objetos Python e manipulá-las para controlar o banco de dados.

4.3 Nova descoberta UPnP

Tendo especificado o novo dispositivo, servidor de perfis UPnP-UP, foi feita uma pequena modificação no processo de descoberta de dispositivos UPnP para que fosse possível informar sua compatibilidade com a extensão UPnP-UP. Para isso foi adicionado no cabeçalho das mensagens de anúncio (ver Código 1) e resposta de requisição de descoberta (ver Código 3) um novo atributo denominado UP, que através de valores *Yes* ou *No* informa se o dispositivo oferece ou não suporte a extensão UPnP-UP. A seguir, Códigos 8 e 9, estão descritas as novas mensagens de anúncio e resposta de requisição.

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.250: 1900
3 CACHE-CONTROL: <Max - age>
4 LOCATION: URL para descrição do dispositivo
5 NT: "dispositivo alvo"
6 NTS: ssdp: alive
7 SERVER: OS/Version
8 USN: UUID do anúncio
9 UP: Yes
```

Código 8 – Novo anúncio de descoberta UPnP

```
1  HTTP/1.1 200 OK
2  CACHE-CONTROL: <Max - age> segundos até o tempo de procura expirar
3  DATE: data do momento em que a requisição foi gerada
4  EXT:
5  LOCATION: URL da descrição do dispositivo
6  SERVER: OS/Version
7  ST: alvo de procura
8  USN: UUID buscado
9  UP: Yes
```

Código 9 - Nova resposta da requisição de busca UPnP

Após o novo processo de descoberta, o *Control Point* através da chamada ao método *auth()* se autentica no servidor de perfis UPnP-UP, o qual retorna o identificador UUID de autenticação.

Em seguida o *Control Point* está apto a informar seu UUID aos dispositivos que dão suporte a extensão UPnP-UP. Estes dispositivos podem se conectar ao servidor de perfis e através da chamada ao método *getProfile()* receber um arquivo no formato XML contendo as preferências do usuário, como mostra o Código 10 descrito a seguir.

```
1  <root xmlns = "urn : shemas-upnp-up-org: up-1-0">
2      <specVersion>
3          <major>1</major>
4          <minor>0</minor>
5      </specVersion>
6      <userProfile id="235">
7          <upnpCategoryList>
8              <category id=1 name= "AudioVideoImage">
9                  <genreList>
10                     <genre id = 10 title= "MPB" >
11                         <artistList>
12                             <artist> Caetano Veloso </artist>
13                             <artist> Ana Carolina </artist>
14                         </artistList>
15                     </genre>
16                 </genreList>
17             </category>
18         </upnpCategoryList>
19     </userProfile>
20 </root>
```

Código 10 - Exemplo de um arquivo XML com preferências do usuário UPnP-UP

Como descrito até então, o novo processo de busca pode ser resumido na Figura 3. Esse processo de descoberta e autenticação apresenta uma forma padronizada de comunicação para obter perfis de usuários permitindo personalização de serviços, o que torna viável a concretização da pervasidade no uso de informações de contexto em redes UPnP.

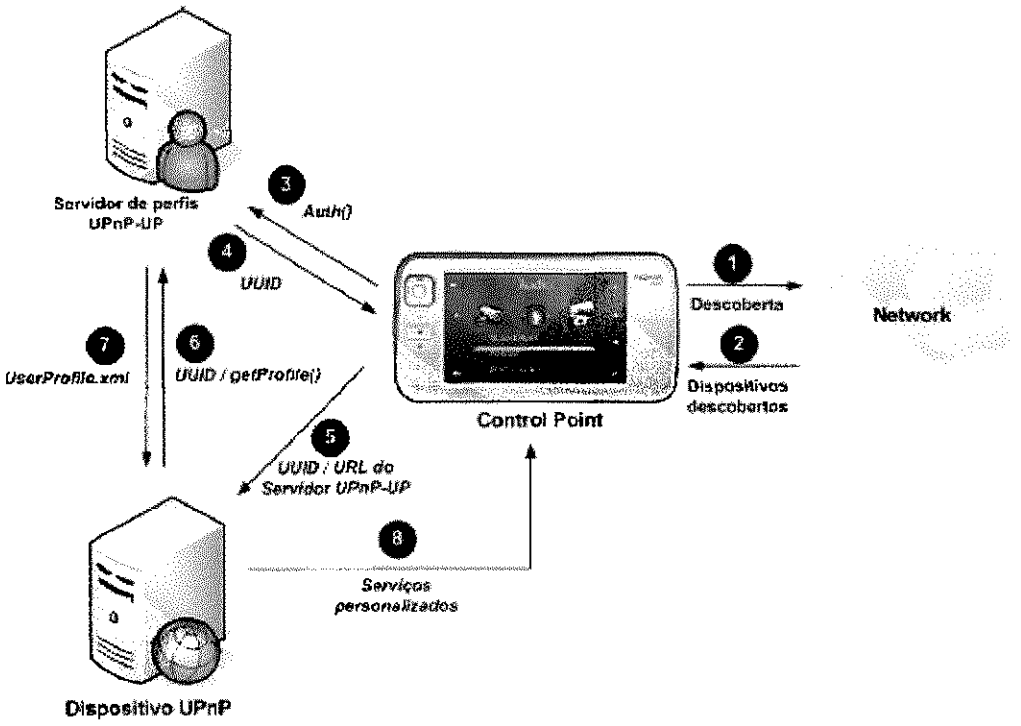


Figura 3 - Processo de autenticação UPnP – UP

5. Conclusão

Neste trabalho foi realizada a implementação de uma extensão UPnP-UP para dar suporte a serviços personalizados em redes UPnP. Foi desenvolvido um novo dispositivo UPnP que oferece serviços de autenticação e disponibilização de perfis de usuários conectados a rede.

Objetivando manter a compatibilidade com o padrão UPnP foi utilizado a tecnologia XML para promover flexibilidade na disponibilização dos perfis dos usuários em redes UPnP. Devido a esta decisão, é possível que diversas aplicações UPnP façam uso da extensão UPnP-UP para personalização de serviços. Como estudo de caso, utilizou-se o BRisa, um arcabouço que permite a criação de dispositivos e serviços em redes UPnP.

Como sugestões para trabalhos tem-se a implementação de uma interface Web para que os usuários cadastrem seus dados e preferências pessoais. Sugere-se também a implementação de mais dois serviços na extensão UPnP-UP. O primeiro é o *renewSession*, que permitirá validar o UUID após um determinado período do acesso ao usuário na rede. O segundo serviço é o *logout*, que possibilitará ao usuário finalizar a sessão junto ao servidor de perfis.

6. Bibliografia

[1] Brisa UPnP A/V Framework for Maemo, http://garage.maemo.org/projects/brisa. Último acesso em Outrubro/2008.

[2] Nokia, Intel & Microsoft (2006), 'Upnp device architecture'. http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0-20060720.pdf. Último acesso em Outubro/2008.

[3] Nokia, Intel & Microsoft (2002), 'Upnp av architecture'. http://www.upnp.org/specs/arch/UPnP-av-AVArchitecture-v1.0-20020622.pdf. Último acesso em Outubro/2008.

[4] SALES, Leandro de Melo, NASCIMENTO, José Luis, C. JUNIOR, Francisco, ALMEIDA, Hyggo Oliveira de, PERKUSICH, A., CHENCAREK, Renato. Uma Implementação de Código Aberto do Padrão UPnP de Áudio/Vídeo/Imagem para Dispositivos com Recursos Limitados In Anais do IX Workshop de Software Livre, IX Workshop de Software Livre (WSL 2008) Porto Alegre, RS 2008. P. 217-222.

[5] Thiago Bruno de Melo Sales. Habilitando perfis de usuários para dar suporte a serviços personalizados em redes UPnP (2008). Trabalho de Conclusão de Curso (Curso de Ciência da Computação) – Universidade Federal de Alagoas.

[6] D. Saha, A. Mukherjee, Pervasive Computing: A Paradigm for the 21st Century, IEEE Computer, IEEE Computer Society Press, pp. 25-31, March 2003.

[7] P. Louridas. SOAP and Web Services, IEEE Software, vol. 23, no. 6, pp. 62-67, December, 2006.

[8] SALES, Leandro de Melo, ALMEIDA, Hyggo Oliveira de, PERKUSICH, Angelo, SALES, ThiagoBruno de Melo, GORGÔNIO, Kyller. Towards the UPnP-UP: Enabling User Profile to Support Customized Services in UPnP Networks, no UBICOMM 2008.

[9] http://www.sqlite.org/about.html. Último acesso em Outubro/2008

[10] Python Programming Language. http://python.org. Último acesso em Setembro/2008.

[11] XML- Extensible Markup Language. http://www.gta.ufrj.br/grad/00_1/miguel/. Último acesso em Outubro/2008.

[12] Nokia, Intel & Microsoft (2008), 'UPnP Forum'. http://www.upnp.org/. Último acesso em Junho/2008.