Context gathering in Ubiquitous Environments: Enhanced Service Discovery

Suparna De and Klaus Moessner

Centre for Communication Systems Research
University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom
S.De@surrey.ac.uk, K.Moessner@surrey.ac.uk

Abstract. Delivering individualized services that conform to the user's current situation will form the focus of ubiquitous environments. A description of the networked environment at a semantic level will necessitate contextually oriented knowledge acquisition methods. This then engenders unique challenges for the crucial step of resource discovery. A number of service discovery protocols exist to perform this role. In this paper, we identify the requirements inherent for such an environment and investigate the suitability of the available protocols against these. A suitable candidate solution is proposed with an implementation with semantic extensions and reference points for further enhancements.

1 Introduction

A personalized user experience attuned to the user's needs and current situation that works seamlessly with the surrounding mobile networked environment forms the vision of 'ubiquitous delivery of services'. This aim puts the focus on the user, rather than the current status quo of interoperations between devices and applications that defines pervasive environments. The plethora of feature-rich devices and communication mechanisms available today hold out the possibility of offering a wide range of service functions to the user. Services today extend from the more traditional view of printing or fax services to software offering streaming music and video, text and instant messaging, language translation to mobile television. This brings up the question of looking for, configuring, communicating with and delivering services. Hence, service discovery forms one of the most important functions within the envisaged ubiquitous environment. It enables devices and services to properly discover, configure and communicate with each other [1]. The user should be relieved of the repetitive tasks of configuring and decision making while being presented with an automated and personalized service experience. The twin goals of automation and personalization require constant sensing of the user's context which may include the current environment and network characteristics, while at the same time being cognizant of the user's preferences when searching for and delivering services to the user. Another aspect deals with continuous service access as the ambient environment changes due to user mobility, without the need for user intervention or interruption of the service. Thus, ubiquitous environments call for more flexible service discovery queries and announcements that take into account the dynamic nature of the user and services.

Current context aware service discovery approaches include those that are based on and extend the Web services initiative which makes use of standards such as Web services Description Language (WSDL) [2] and XML [3]. WSDL is syntactic in nature, merely specifying the format of the service [4]. The related service discovery protocol, Universal Description, Discovery and Integration (UDDI) [5] makes use of service directories and data structures that are more relevant for business organizations advertising their services on the web; focusing on electronic commerce, which differs from the ubiquitous target environment. Moreover, UDDI cannot be readily adapted to ubiquitous applications as it doesn't allow for extensions beyond its domain of business oriented services [6]. Also, this approach places no limitations on the discovery scope, whereas ubiquitous applications have a strong focus on locality. Other approaches rely on service directories that necessitate a wired infrastructure, making them at odds with the highly mobile and transient ubiquitous domain.

Existing resource discovery protocols are well placed for performing context sensitive service discovery and provisioning since knowledge of the availability of resources in the networked environment can form the first step in context information gathering [7]. This paper looks at the existing industry-standard service discovery protocols and how these can be augmented to be suitable for a ubiquitous environment. In particular, the different approaches are weighed on the basis of their support for flexibility for handling future device types and support for reasoning about services. The remainder of this paper is organized as follows: section 2 identifies the requirements and challenges pertinent to service discovery in the target domain. Section 3 presents brief outlines of the different service discovery protocols and how they tackle and define the roles for resource discovery as well as how they compare in light of the identified requirements. Section 4 discusses a possible candidate approach for the problem domain, its merits and shortcomings as well as a design for extensions for the target domain. Conclusions are drawn in section 5.

2 Requirements

Existing protocols address the service discovery problem with their own definitions of roles (client/ server/ service repository) and terminology to achieve the required functionalities. The target domain is characterized by the dynamic nature of devices (or services) joining and leaving a service area. In such an ambient environment, service discovery protocols must meet certain criteria, in addition to their expected service provisioning role, in order to fulfill the goal of context aware service provisioning. This section sets out the metrics that are required for a resource discovery protocol to perform a context gathering service.

Initial communication method: though unicast is the most efficient initial communication method among the components involved in service discovery, it requires a priori knowledge of the network configuration, which makes it infeasible for our dynamic problem domain. Multicast offers an elegant solution with initial service advertisement and resource query messages using UDP (User Datagram

Protocol) multicast before components determine network addresses and switch to unicast to make efficient use of bandwidth. Link layer broadcast is another option that offers the same advantage as multicast, but limits the message scope to within a single hop of the underlying network [1].

Description capabilities: services are described by service types and attributes. Most protocols provide a template for defining service names and attributes. The extensibility provided to service developers is a key issue here, which can provide means for augmenting the expressiveness of the service and scope for its expandability. Scope for extending templates to include semantic descriptions and ontology concepts that relate attributes names with values that are predefined or within a permissible range, are desirable. Introduction of semantics into the descriptions and scope for possible mapping to ontologies can provide the means for services to differentiate themselves from similar offerings and allow reasoning through the use of an inference engine.

Query capabilities: typical service search procedures are structured by type and/or attribute values. Possibility of including or augmenting the search process with semantic information is a desirable characteristic as then the returned results could be tailored to the context of the (service-) requester. Another problem area concerns standardized service terminology used by the various approaches. For instance, what happens when a client looks for a "print" service when a service implements a "printing" service [1]? Searching by wildcards or prefix matches might offer a solution in this case. Browsing of all available services offers more search options.

Discovery scope: proper discovery scopes can minimize computation overheads and deliver services best matched to the user's needs. Many approaches equate scope with network topology such as a LAN (local area network) or wireless network range while administrative domain-based approaches require prior knowledge of the target domain. Scopes are more appropriately defined by taking into consideration context information such as user location and tasks as well as role (services accessible to a visitor would be markedly different from that of a host). This allows service discovery to be placed at a higher level of abstraction than the primitives defined by the protocol itself.

State notification: provision of event notification to clients provides them reliable information about the service state. As ubiquitous environments allow multiple service providers to coexist at a place, this allows clients to keep track of the available services at any given time along with the service information state. This capability also facilitates graceful, soft-state service management mechanisms and simpler system design [1]. Clients usually subscribe to a service, which then sends notifications of events or changes in service status.

Security: security and privacy features are required both for users and service providers. Specifically in dynamic environments, issues of authorization and access rights are more complicated than in static ones. How much personal information is given out during service discovery and with what degree of conscious approval from the user defines the privacy issue from a user standpoint. Security is essential for the service provider as well specifically when the underlying network may be insecure. Using existing security solutions may not meet security requirements unique to the current problem domain.

3 Service Discovery Approaches

3.1 Jini

Jini [8] is a distributed service-oriented architecture by Sun Microsystems that uses the Java software platform. It provides mechanisms for service registration and use through RPC (Remote Procedure Call) and RMI (Remote Method Invocation) to access Java proxy objects. A Jini Lookup Service (JLS) maintains dynamic information about the available services in a Jini federation (Jini service collection). To join a Jini federation, a Jini service first discovers one or many JLS from local or remote networks and then uploads its service proxy (i.e. a set of Java classes) to it. The service clients can use this proxy to contact the original service and invoke methods on the service.

Jini uses multicast both for clients and services to discover lookup services and for the lookup service to announce its presence in the network. Heavy use of multicast though, can affect protocol performance. Services are described using user-friendly names and attributes. With descriptions primarily being Java interfaces to services and geared towards a programming perspective, expressiveness is not factored in. Overall, Jini does not provide a structured language for building descriptions and it also lacks the definition of an organizational scheme for services [6]. Service discovery is supported through queries on service type and attribute value, with more than one attribute capable of being specified in the query. Browsing is also available. Search queries are processed through Java interface matching. This necessitates that the client should be aware of the exact name of the Java class implementing the service. Discovery scope is well-defined with search options tailored by network topology (LAN), administrative domain or location. Jini offers notification services through agents that aggregate, filter and send events to clients. This approach benefits both clients and services but requires additional resources. The security issue in Jini is native to Java and RMI due to the provision of downloadable code. This is addressed through verification and granting of permissions. Abstract interfaces are also defined for designers to use various security protocols. Deployment of Jini requires JVM (Java virtual machine) to be installed on participating networked devices, which is not possible on resource constrained devices such as mobile phones.

3.2 Universal Plug and Play (UPnP)

The UPnP specification [9] describes a set of protocols for enabling networked devices to initialize autonomously, discover and then share one another's services (Fig. 1) [10]. It offers functionalities for discovery, service information provision, service control by clients, and notification of changes in offered services by the servers to clients as well as graphical presentation of services and their access methods.

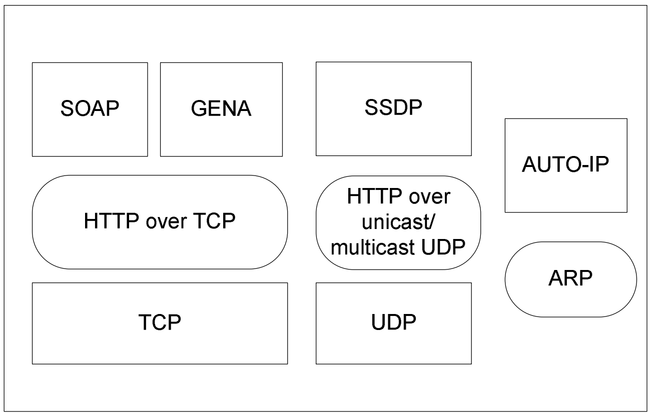


Fig. 1. UPnP protocol stack [10]

Multicast messages in UPnP are used by services to announce their presence as well as exit and by clients to search available services. Service descriptions are based on XML templates that include input/output variables and related state variables. The XML template allows more information about devices and offered services to be described through its flexible format and also provides support for cross-platform lightweight usability [11]. Though device and service types are standardized, vendor defined types are also supported to extend capabilities. Queries are matched against service types by using the XML matching mechanism. Responses are generated if the query is a prefix of or equal to supported device or service type, which rules out learning protocol specific terminology. However, this match takes place only at a syntactic level. Browsing is supported. The scope of the search though, is bound to the local LAN. Notification is explicitly addressed in the specification with clients subscribing to servers to receive status events. Security issues are well handled with many authorization methods and by factoring in interactions between users and environments.

3.3 Salutation

Salutation [12] is an open software platform that supports service registration, discovery, availability and session management. It specifies a generic transport interface, which can be used to bind onto an underlying transport protocol. It can operate both in a peer-to-peer as well as directory model. Salutation Managers (SLM) handle service registration information and discovery requests (Fig. 2). SLMs also coordinate amongst themselves through RPC for collecting discovery responses.

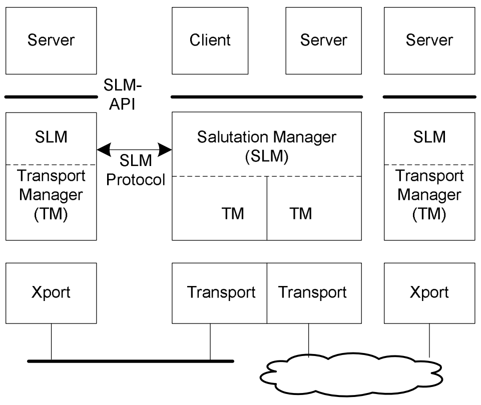


Fig. 2. Salutation manager network model [13]

Service registration and query is directed to the local SLM, which may be discovered through broadcast on the mapped transport interface. Descriptions are structured by predefined service types (or features) and attributes that provide further detail. Attributes relevant to each service type are defined in the service description, with individual services filling in values for these attributes. The description language used is ISO ASN.1 (Abstract Syntax Notation One). Queries are directed at type or attribute matching. However, scope of the search is restricted to network domains. Possibility of notification is tied in with application design as it is based on RPC to access and use the service. Authentication is the only security issue addressed with a user-id and password scheme. It is worthwhile to note that the dissolution of the Salutation consortium in June 2005 calls into question its future support and evolution.

3.4 Service Location Protocol (SLP)

SLP [14] is a protocol for IP based networks, which uses the UDP and TCP transport layers to offer service advertisement, discovery, search and (optionally) registration. SLP service agents advertise service handles to the directory agents to make services available to the user agents. This protocol also supports a simple service registration leasing mechanism.

Initial communication methods cover the range of uni-, multi- and broadcast. Service descriptions feature templates with standardized types and descriptive features. However, the description feature allows only a simple categorization of services based on ‘service URLs’ [15]. The service:URL specifies a service type, address and a set of text-based attribute value pairs. The template contents also guide the querying mechanism, with queries expressed by an LDAPv3 search filter. A good feature is that queries can specify scope that group services by location or administrative domains. Event notification is not supported. Authentication and integrity are handled through service information.

3.5 Bluetooth Service Discovery Protocol (SDP)

Bluetooth SDP [16] forms part of the Bluetooth communication protocol suite (Fig. 3) and is used for service discovery and search. At the server end, it offers service registration in a repository located in the local device.

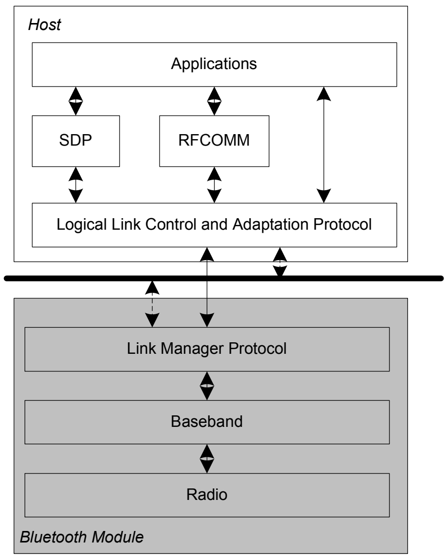


Fig. 3. Bluetooth protocol stack [17]

Device discovery is performed through broadcast on the Bluetooth network interface followed by service discovery on the discovered device (unicast). Services are described by globally unique identifiers for predefined service types. However, there has been an effort [18] to extend the SDP with semantic information expressed in RDF or DAML+OIL to provide more expressiveness to device information. Service searches target type and attribute value, though the scope of the search is limited to the local Bluetooth network. Notifications are not supported. A device can make itself undiscoverable unless the search is performed by a known trusted device. This security feature augments the Bluetooth link layer security.

The comparison of the different service discovery protocols profiled above is summarized in Table 1.

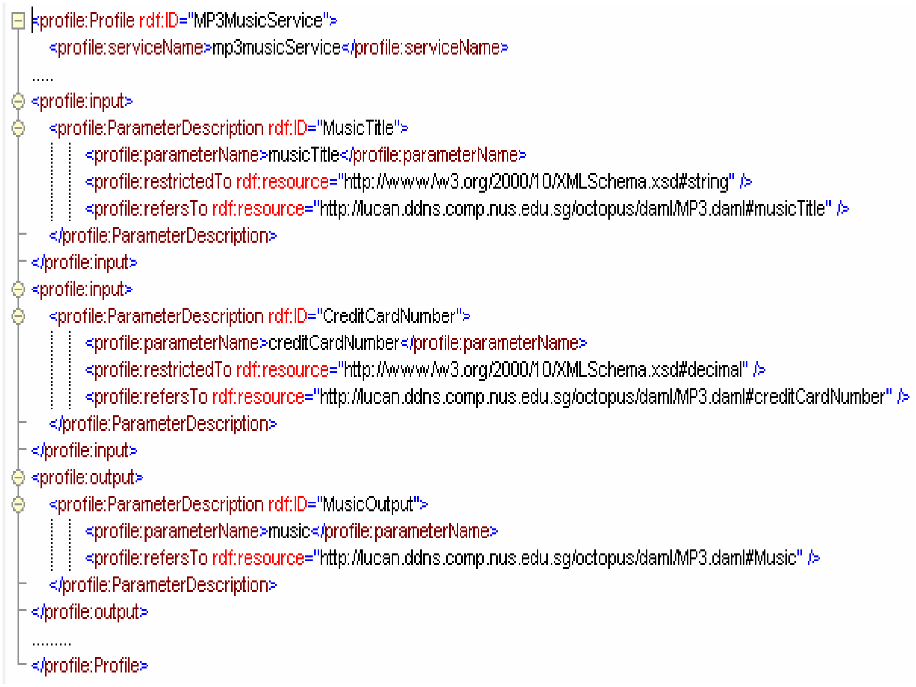
Table 1. Comparison of service discovery protocols

Feature	Jini	UPnP	Salutation	SLP	Bluetooth SDP
Initial communicatio-n method	Unicast and multicast	Unicast and multicast	Unicast and broadcast	Unicast, multicast and broadcast	Unicast and broadcast
Descriptio-n capability	Template (Java interfaces)	XML template, vendor defined types supported	Predefined XML-based template	Template, simple service categories	128-bit universal unique IDs for service types
Query capability	Service type and attribute value. Browsing support	Service type, prefix matches supported. Browsing support	Type and attribute matching. Browsing support	Type, attribute value. Browsing support	Type, attribute value. Browsing support
Discovery scope	LAN, administrative domain, location	LAN	LAN	Administr-ative domain, LAN, location	Bluetooth network
State notification	Through agents	Subscription mechanism	Dependent on application design	Not supported	Not supported
Security	Verification, permissions. Interfaces to existing security protocols	Device-user interactions handled, many authorization procedures	Authentica-tion (user-id, password scheme)	Authentica-tion	Trusted discovery, Bluetooth security

4 Enhancing Service Discovery

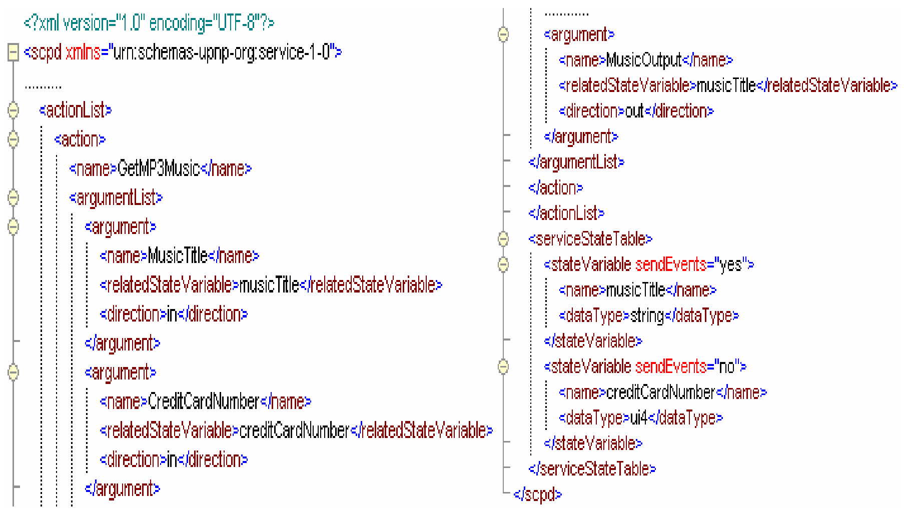
Based on the analysis performed in section 3 on a per-protocol basis, UPnP emerges as the most suitable candidate for the given problem domain. Although any of the proposed candidates can be deployed in various pervasive environment settings, no global standard has evolved as yet. UPnP proves to be the best choice for our stated metrics, specifically in terms of flexibility and support for service reasoning. From a networking standpoint, its strength lies in self-configuration with automatic assignment of IP addresses and DNS names to participating devices. This augurs well for a mobile ad-hoc network that typically characterizes the target domain. The description capabilities are well advanced. UPnP caters to independent descriptions of the physical device as well as software services hosted on the device. Device descriptions model a device as a logical container with one or more embedded devices, each capable of being discovered and used independently of the container device. This allows hierarchical description of the hardware components of a device such as a mobile phone with screen, keypad etc. This closely follows the CC/PP

(Composite Capability/Preference Profiles) concept of describing devices as composed of different (hardware/ software) components [19]. Versions associated with descriptions allow addition of features in a description document. It is worth noting that addition of properties to existing device descriptions and related notifications figure in the list of requirements identified for a ‘device description repository’ by the Device Independence Working Group within the W3C [20]. The XML-based service description template also lends itself to semantic extensions with the approach being similar to DAML-S. DAML-S is a web service ontology that represents services semantically in terms of human readable description, functionalities and functional attributes. The functional specification represents a service in terms of inputs, outputs, preconditions and effects [21]. Analogously, UPnP services have provision for input and output parameters that are related to state variables that represent the service’s current status. A default value for each state variable is specified which is within a defined permissible range. Fig. 4 [21] depicts a MP3 music service description in DAML-S and its counterpart as may be modelled in UPnP (Fig. 5).



```
<profile:Profile rdf:ID="MP3MusicService">
  <profile:serviceName>mp3musicService</profile:serviceName>
.....
<profile:input>
  <profile:ParameterDescription rdf:ID="MusicTitle">
      <profile:parameterName>musicTitle</profile:parameterName>
      <profile:restrictedTo rdf:resource="http://www.w3.org/2000/10/XMLSchema.xsd#string" />
      <profile:refersTo rdf:resource="http://ucan.ddns.comp.nus.edu.sg/octopus/daml/MP3.daml#musicTitle" />
  </profile:ParameterDescription>
</profile:input>
<profile:input>
  <profile:ParameterDescription rdf:ID="CreditCardNumber">
      <profile:parameterName>creditCardNumber</profile:parameterName>
      <profile:restrictedTo rdf:resource="http://www.w3.org/2000/10/XMLSchema.xsd#decimal" />
      <profile:refersTo rdf:resource="http://ucan.ddns.comp.nus.edu.sg/octopus/daml/MP3.daml#creditCardNumber" />
  </profile:ParameterDescription>
</profile:input>
<profile:output>
  <profile:ParameterDescription rdf:ID="MusicOutput">
      <profile:parameterName>music</profile:parameterName>
      <profile:refersTo rdf:resource="http://ucan.ddns.comp.nus.edu.sg/octopus/daml/MP3.daml#Music" />
  </profile:ParameterDescription>
</profile:output>
.........
</profile:Profile>
```

Fig. 4. Music service advertisement in DAML-S [21]



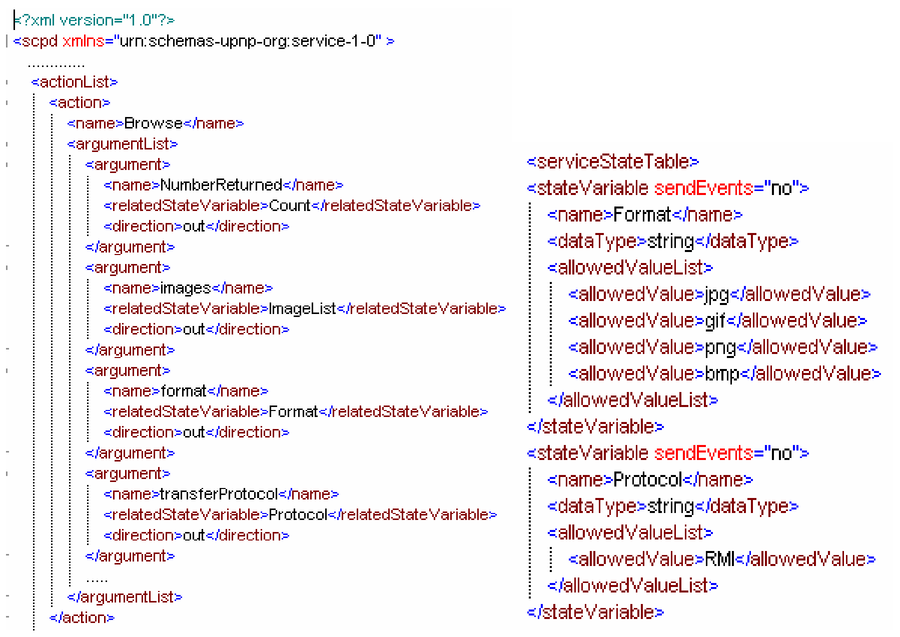
```
<?xml version="1.0" encoding="UTF-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
..........
  <actionList>
    <action>
      <name>GetMP3Music</name>
      <argumentList>
        <argument>
          <name>MusicTitle</name>
          <relatedStateVariable>musicTitle</relatedStateVariable>
          <direction>in</direction>
        </argument>
        <argument>
          <name>CreditCardNumber</name>
          <relatedStateVariable>creditCardNumber</relatedStateVariable>
          <direction>in</direction>
        </argument>
        ...........
        <argument>
          <name>MusicOutput</name>
          <relatedStateVariable>musicTitle</relatedStateVariable>
          <direction>out</direction>
        </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>musicTitle</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>creditCardNumber</name>
      <dataType>ui4</dataType>
    </stateVariable>
  </serviceStateTable>
</scpd>
```

Fig. 5. Music service description in UPnP

The following section shows an example implementation of how the UPnP service description can be extended with semantic concepts.

4.1 UPnP Service Description Extension

Our example implementation to extend the UPnP description template to deliver more meaningful service descriptions consists of a “HomeMedia” system with a *ContentStore* and a *MediaRendering* device. The content store is a repository of media content that interfaces to the content stored on the physical device (only still images in this case) and exposes this through the *ContentDirectory* service. Users of this service can browse through the stored list and retrieve the list of image names. The list of image names is accompanied by the format parameter tag that specifies that the stored images are jpg, gif, png or bmp files. The *ContentStore* also implements an out-of-band transfer protocol for the transfer of the actual media content (we have implemented this in Java RMI). The associated XML description is shown in figure 6.

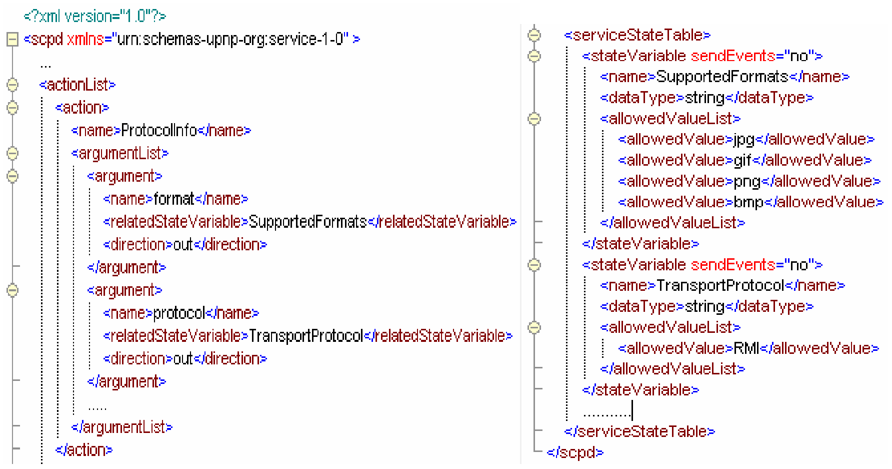


```xml
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0" >
  ............
  <actionList>
    <action>
      <name>Browse</name>
      <argumentList>
        <argument>
          <name>NumberReturned</name>
          <relatedStateVariable>Count</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>images</name>
          <relatedStateVariable>ImageList</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>format</name>
          <relatedStateVariable>Format</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>transferProtocol</name>
          <relatedStateVariable>Protocol</relatedStateVariable>
          <direction>out</direction>
        </argument>
        .....
      </argumentList>
    </action>
```

```xml
<serviceStateTable>
<stateVariable sendEvents="no">
  <name>Format</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>jpg</allowedValue>
    <allowedValue>gif</allowedValue>
    <allowedValue>png</allowedValue>
    <allowedValue>bmp</allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>Protocol</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>RMI</allowedValue>
  </allowedValueList>
</stateVariable>
```

Fig. 6. ContentStore service description

The *MediaRendering* device is an electronic screen that offers a Display service for optimal image rendering. Transparent to the user, the *ConnectionManager* service implements a transfer protocol to pull the actual image to be displayed from the content store within the "HomeMedia" system. This service has a *ProtocolInfo* property that defines what image formats are supported by the Display function as well as the transport protocol used to retrieve the image, as depicted in figure 7.



```xml
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0" >
  ...
  <actionList>
    <action>
      <name>ProtocolInfo</name>
      <argumentList>
        <argument>
          <name>format</name>
          <relatedStateVariable>SupportedFormats</relatedStateVariable>
          <direction>out</direction>
        </argument>
        <argument>
          <name>protocol</name>
          <relatedStateVariable>TransportProtocol</relatedStateVariable>
          <direction>out</direction>
        </argument>
        .....
      </argumentList>
    </action>
```

```xml
  <serviceStateTable>
    <stateVariable sendEvents="no">
      <name>SupportedFormats</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>jpg</allowedValue>
        <allowedValue>gif</allowedValue>
        <allowedValue>png</allowedValue>
        <allowedValue>bmp</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>TransportProtocol</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>RMI</allowedValue>
      </allowedValueList>
    </stateVariable>
    ..........
  </serviceStateTable>
</scpd>
```

Fig. 7. MediaRenderer service description

This service description, in conjunction with the information received while browsing through the list of image content from the *ContentStore*, can help the *MediaRendering* device make a decision if it is capable of retrieving and displaying the particular chosen image.

4.2 Discussion

Enhanced capabilities of service discovery coupled with context descriptions of the offered services can act as a mechanism of context gathering as well as a first step towards context reasoning in the target domain. As shown in our implementation, the extended service descriptions can be used to enhance the service offerings with contextual aspects and deploy a coarse filtering mechanism. This allows a service consumer to filter out interesting and compatible services and then load and evaluate only these services. Use of available tools also enable automatic mapping of these service descriptions to a defined ontology which can aid precise reasoning about the various service offerings.

Another important characteristic that makes UPnP suitable to the target mobile ad-hoc heterogeneous environment is that the service advertisement messages contain a duration field that specifies the length of time for which the advertisement is valid. This ensures that device availability information is kept up-to-date in the network and makes UPnP self-healing for the possibility when devices leaving the network do not announce this. The minimum duration specification also prevents large amounts of traffic being generated by repeated advertisements. UPnP devices also respond to explicit requests by clients.

UPnP's limitations exist in the area of discovery scope as discovery is usually limited to the local network and not guided by user roles or context information. This feature is thus amenable to further improvement and extension. UPnP Security standardizes the human-computer interaction procedure and incorporates the heterogeneity of networked devices through various authorisation procedures. It addresses many security issues related to ubiquitous environments [1].

5 Conclusion

As our implementation has shown, UPnP can act as a mechanism for facilitating context awareness in the target domain. Future research work will focus on extending the scope of discovery with contextual aspects and augmenting service descriptions with ontology concepts to form a semantically enriched set of services amenable to contextual reasoning.

Acknowledgement

The work reported in this paper has formed part of the Ubiquitous Services Core Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com. This research has been funded by the DTI-led Technology Programme and by the Industrial Companies who are Members of Mobile VCE. Fully detailed technical reports on this research are available to Industrial Members of Mobile VCE.

References

1. F.Zhu, M. W. Mutka, and L. M. Ni, “Service Discovery in Pervasive Computing Environments,” IEEE Pervasive Computing, vol. 4, pp. 81-90, 2005.
2. W3C, "Web Services Description Language: Core Language," W3C Candidate Recommendation v. 2.0, 27 March 2006.
3. W3C, "XML Schema," W3C Recommendation October 28 2004.
4. L. Simon, A. Bansal, A. Mallya, S. Kona, G. Gupta, and T. D. Hile, "Towards a Universal Service Description Language," presented at International Conference on Next Generation Web Services Practices, NWeSP 2005.
5. OASIS, (2002, July). "UDDI API Specification," v. 2.04 [Online]. Available: http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf.
6. M. S. Thompson and S. F. Midkiff, "Service Description for Pervasive Service Discovery," presented at IEEE International Conference on Distributed Computing Systems Workshops, 2005.
7. R. Robinson and J. Indulska, "A Context-Sensitive Service Discovery Protocol for Mobile Computing Environments," presented at International Conference on Mobile Business, 2005.
8. Sun Microsystems, (2003, June). "Jini Technology Core Platform Specification," v. 2.0 [Online]. Available: www.sun.com/software/jini/specs/core2_0.pdf.
9. UPnP Forum, (2003, December). "UPnP™ Device Architecture," [Online]. Available: http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf.
10. G. G. Richard, Service and device discovery protocols and programming, 1 ed: Mc Graw Hill 2004. ISBN 0071379592
11. L. Choonhwa, A. Helal, N. Desai, V. Verma, and B. Arslan, "Konark: A system and protocols for device independent, peer-to-peer discovery and delivery of mobile services," IEEE Transactions on Systems, Man and Cybernetics, Part A, , vol. 33, pp. 682-696, 2003.
12. Salutation Consortium, "Salutation Architecture Specification," 1999.
13. B. Miller, "Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer,” Bluetooth SIG, Bluetooth Whitepaper 1999.
14. E. Guttman et al, "Service Location Protocol," IETF RFC 2608, v. 2, June 1999.
15. D. Elenius and M. Ingmarsson, "Ontology-based Service Discovery in P2P Networks," presented at Second International Workshop on Peer-to-Peer Knowledge Management, San Diego, CA, 2005.
16. Bluetooth SIG, "Specification of the Bluetooth System," 2003.
17. O. Marques and N. Barman, "Wireless communications using Bluetooth," in Wireless Internet handbook: technologies, standards, and application: CRC Press, 2003, pp. 307-333. ISBN 0-8493-1502-6
18. S. Avancha, A. Joshi, and T. Finin, "Enhanced Service Discovery in Bluetooth," Computer, vol. 35, pp. 96-99, 2002.
19. W3C, (2002) "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies," [Online]. Available: http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115.
20. W3C, (2006, April 10). "Device Description Repository Requirements 1.0," in W3C Working Draft, Device Independence Working Group, [Online]. Available: http://www.w3.org/TR/2006/WD-DDR-requirements-20060410.

21. T. Gu, H. C. Qian, J. K. Yao, and H. K. Pung, "An Architecture for Flexible Service Discovery in OCTOPUS," presented at Proceedings of the 12th International Conference on Computer Communications and Networks, Dallas, Texas, 2003.
22. M. Ghader, K. Moessner, and R. Tafazolli, "Service Discovery and Provision Protocols for Wireless Networks" presented at 13th IST Mobile & Wireless Communications Summit, Lyon, France, June 27-30, 2004.
23. California Software Labs, (1999) "UPnP, Jini and Salutation - A look at some popular coordination frameworks for future networked devices," whitepaper, [Online]. Available: http://cswl.com/whitepapers/upnp-devices.html.
24. UPnP Forum, (2001). "Device and Service Versioning in SSDP," [Online]. Available: http://www.upnp.org/resources/documents.asp.