Suitability of existing service discovery protocols for mobile users in an ambient intelligence environment

Davy Preuveneers
Department of Computer Science
K.U.Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium
Email: davy.preuveneers@cs.kuleuven.ac.be
Telephone: (+32) (0)16 327853
Fax: (+32) (0)16 327996

Yolande Berbers
Department of Computer Science
K.U.Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium
Email: yolande.berbers@cs.kuleuven.ac.be
Telephone: (+32) (0)16 327636
Fax: (+32) (0)16 327996

Abstract—Service discovery enables users to access information, resources and services anytime, anywhere. This involves a client, a service provider and an infrastructure for service registration and lookup. Clients use service discovery protocols (SDPs) to issue a service request to a service registry or an individual service provider. In this paper we investigate the suitability of existing SDPs for mobile users in an ambient intelligence context. After reviewing the most well-known SDPs, common problems in the domain of user mobility and solutions that are currently available are discussed.

Keywords—Service discovery, ambient intelligence, mobile computing, ad-hoc networking.

Presentation—**Should this paper be accepted, it will be presented by Davy Preuveneers.**

I. INTRODUCTION

In their report *Scenarios for Ambient Intelligence in 2010* [9], the IST Advisory Group shows concrete visions of life in tomorrow's user-friendly information society where people are surrounded everywhere all the time by unobtrusive, interconnected intelligent objects. Ambient intelligence is a user-centered paradigm, based on the belief that the many networked devices will be moved into the background, while providing personalized services to the user and anticipating a user's desire at any place and time. To reduce the number of interactions with humans, services need to become more autonomous, for example by adapting to user preferences and situational context and interacting with other services in supporting a user who is performing a specific task.

Typical devices on which a mobile user nowadays wants to use these services are laptops, tablet PCs, PDAs, mobile phones, etc. It is clear that these devices have varying capabilities and different resource constraints, for example limited processing power, memory, network bandwidth and battery life time. With the emergence of wireless ad-hoc networks and ubiquitous and pervasive computing, it is believed that these devices will be everywhere, but hidden from the users or nicely integrated into their environment.

To support user mobility, several service discovery protocols (SDPs) have been proposed that enable device and service interaction. These protocols change the way services are configured, deployed and advertised on a network. Their aim is to enable mobile users to discover new services and to support seamless and scalable device and service interoperability. Service announcement and registration, service lookup, query and event mechanisms are the key features of the most common SDPs. Among the contenders are Jini [16], Universal Plug and Play (UPnP) [21], Salutation [19], Service Location Protocol (SLP) [6] and the Bluetooth SDP [4].

In this paper, we discuss the main characteristics of these protocols and focus on their applicability for a mobile user employing devices with limited capabilities. In section 2 we shortly describe existing SDPs. Section 3 presents common problems with these protocols with regard to user mobility and mechanisms to overcome them. Related work is described in section 4 and we end this paper with a conclusion in section 5.

II. SERVICE DISCOVERY PROTOCOLS

To make use of services in our environment, protocols need to be defined that describe user-service and service-service interactions. In this section we will describe the following protocols: Jini [16], Universal Plug and Play (UPnP) [21], Salutation [19], Service Location Protocol (SLP) [6] and the Bluetooth SDP [4]. They differ in support for several capabilities, like mobile code, heterogeneous network transport media, interoperability based on open standards and protocols, etc.

A typical scenario requiring service discovery would be engineer Jack who is leaving home for the office. While on the road, he is using his handheld PC to synchronize with his agenda that is being maintained on the office network. This

way he knows which meetings have been scheduled for today. Then he notices he forgot to take along an important technical report he was working on yesterday evening at home on his desktop PC. He retrieves the document with his handheld PC and sends it to the printer at the office. In the meantime, he receives a message from the city traffic control system, installed by the local authorities, that there is a traffic jam on his way to work, suggesting to take another road. Being a little late, he receives an urgent incoming call on his handheld PC from the project manager. Jack immediately switches on the Internet connected audio system in his car and continues the communication using VoIP. Some aspects of this scenario can already be implemented using the current state-of-the-art in network technology, others require more effort. Several service discovery protocols are discussed below.

A. Jini

General description: Jini [16] is a Java-based architecture, developed by Sun Microsystems, based on the idea of federating groups of users and resources into a single dynamic distributed system, turning the network into an easy administered tool. It provides an infrastructure and programming model for developing and federating services to other members of the federation.

Building blocks: Service communication is accomplished by a service protocol using *Remote Method Invocation* (RMI). A *lookup service* is used to locate services. When a device or application registers with a Jini network to advertise supplied services, it first uses the *discovery* protocol to locate an appropriate lookup service, and then joins it by using the *join* protocol. The lookup service then maps the interfaces and other attributes that describe the functionality of the service to the objects that implement it. Hence, a lookup service acts as a registry for all services available on the network, containing objects in the Java programming language.

Discovery: Lookup occurs when a user or service needs to locate and invoke a service described by its interface. The service object in the lookup service is then downloaded by the interested client and is used as local proxy to the service that stored the object in the lookup service. The client interacts directly with the service provider via the service object and not by passing through the lookup service.

Miscellaneous: The Jini architecture provides support for *transactions* and *events*. Transactions allow a series of operations within one or multiple services to be wrapped as one entity. A service protocol is provided to coordinate a *two phase commit*, so that all the changes occur atomically or none of them occur. Events are used to notify other entities in the network. Objects may register interest in events in another object and receive a notification when the event takes place. Code mobility is possible by making use of the provisions of the Java runtime environment.

B. Universal Plug and Play

General description: Universal Plug and Play (UPnP) [11], contributed by Microsoft and now further maintained by the

UPnP Forum, is a technology that extends Plug and Play (PnP) to allow devices, appliances and services to connect to networks, enabling automatic discovery and control of devices, supporting zero-configuration. Devices can dynamically join a network, obtain an IP address, advertise their capabilities and discover those of other devices. UPnP uses standard TCP/IP and Internet protocols, including IP, TCP, UDP, HTTP and XML, and does not specify the APIs applications will use. Hence, it is independent of any programming language or operating system.

Building blocks: The basic building blocks of a UPnP network are *devices*, *services* and *control points*. UPnP devices host the services and may contain nested devices. A control point is a controller for discovering and controlling other devices and may be incorporated in a device. When a device connects to the network, it tries to receive an IP address from a DHCP server. When no DHCP server is available, it selects a random address from a reserved range and sends out an ARP packet to see if the address has already been taken. This protocol is known as AutoIP and avoids any explicit network configuration.

Discovery: UPnP uses Simple Service Discovery Protocol (SSDP) [5] for service discovery. Devices use it to announce their presence to others and to discover other devices or services. This protocol is similar to the discovery, join and lookup protocols of Jini.

Service descriptions: Service descriptions are given in XML documents. They provide information about actions, parameters and variables that model the state of the service at runtime. When a control point sends an action request to a device's service, it sends a control message expressed in XML using SOAP to the control URL for the service.

Miscellaneous: When a service changes its state at runtime, it publishes updates by sending event messages which other control points may subscribe to. These messages contain the names and values of the state variables of the service and are also expressed in XML and formatted using Generic Event Notification Architecture (GENA), a protocol to send and receive notifications using HTTP over TCP/IP and multicast UDP.

C. Salutation

General description: The Salutation architecture [19] is an approach developed by an open industry consortium known as the Salutation Consortium.

Building blocks: It consists of *Salutation Managers* (SLMs) acting as service brokers that coordinate with one another. Services register their capabilities with a local or nearest available SLM and clients query an SLM when they need a service. The request for utilization of a service also passes through an SLM. SLMs do everything on behalf of their clients. A call-back mechanism notifies devices of events.

Discovery: SLMs discover other nearby SLMs and exchange service registration information, even if SLMs are located on different transport media. This is done by using *Transportation Managers*, transport-dependent modules. They provide reliable

communication channels to the SLMs regardless of the underlying network infrastructure. SLMs redirect discovery requests to other SLMs using Sun's Open Network Computing Remote Procedure Call (RPC).

Invoking services in a Salutation architecture is very flexible. In *native* mode Salutation provides support for vendor specific protocols that bypass SLMs. In *emulated* mode, SLMs manage the communication session by carrying the data from one party to another, while the data format is selected by client and server. In *salutation* mode, SLMs not only take care of the transmission but also define the structure of the message being transmitted. Salutation provides APIs to invoke these operations and to gather the results.

Service descriptions: A service description is broken down into *functional units*, representing some essential feature of the service. Each unit provides a set of records which can be queried during service discovery.

Miscellaneous: SLMs do not provide lease based access to services, like Jini and UPnP, but may be asked to periodically check the status of functional units to see if they are still available.

D. Service Location Protocol

General description: The Service Location Protocol Version 2 (SLPv2) [7] is a standard from Internet Engineering Task Force (IETF) and uses multicast and DHCP to initialize the framework. There is no need to configure single SLP *agents*. SLP scales from a single LAN to enterprise networks. It does not scale to the Internet, because DHCP and multicast require configuration and the Internet does not provide a centralized place for administration.

Building blocks: The main components of SLPv2 are *User Agent* (UA), *Service Agent* (SA) and *Directory Agent* (DA). SAs advertise location and attributes on behalf of services, while UAs discover these properties on behalf of client software. DAs act as information caches that store relevant information from SAs about available services on the network. To decrease network bandwidth usage, UAs then query DAs for service discovery, but the use of DAs is optional.

Discovery: There are two methods for DA discovery: *active* and *passive*. In active DA discovery, UAs and SAs multicast *Service Requests* for the DAs to the network. The available DAs will then respond to the request. In passive DA discovery, DAs multicast periodically advertisements for their services to the network. UAs and SAs can also learn the location the location of DAs using DHCP, by configuring DHCP to distribute the addresses of DAs to hosts that request them.

When at least one DA is present, it collects all the information advertised by SAs, and UAs sent unicast messages to the DA, otherwise UAs and SAs communicate directly with one another. UAs then repeatedly multicast requests to SAs. SAs in turn unicast their response to the UA if they match the search criteria.

Service descriptions: Services are advertised using a Service URL, which contains the service location, including IP address, port number and, if necessary, a path. Clients that obtain this URL have all the necessary information they need to connect to the service. The actual protocol used for communication between client and service is SLP independent. Service descriptions are given by *Service Templates*, defining a set of attributes, which UAs use to compose their requests. **Miscellaneous:** Service registration with a DA is lease based, and must be renewed to avoid expiration.

E. Bluetooth Service Discovery Protocol

General description: The Bluetooth SDP [4] is a discovery protocol only available in Bluetooth environments and therefore provides limited functionality compared to other protocols.

Discovery: Bluetooth SDP provides service discovery by searching for service class or service attributes. Service browsing is used when no knowledge is available a priori. It provides no service brokering, nor service advertisements or registrations, nor eventing.

III. COMMON PROBLEMS WITH SDPS

In the previous sections we have discussed the main characteristics of several service discovery protocols. We will now present some challenges and shortcomings to achieve the goals of ambient intelligence.

A. Semantic service discovery and user profiles

Service discovery is very important from a user's point of view. When a user enters a previously undiscovered network, he or she would like to find out which services are available. Service registration and lookup in UPnP, Salutation and SLP is based on the exact matching of name/value pairs. Jini lookup requires the matching of interfaces in the service object. However, searching for services in terms of service properties is not adequate enough, considering the possibility of incomplete matches or synonyms for service attributes. Only returning services that completely match would limit the positive responses to service requests. When Jack in the above scenario would request a videophoning application service for mobile communication with his boss, the VoIP audio system in his car might be adequate enough in the user's context.

One solution for the matching problem would be to use a standardized vocabulary to describe services. The United Nations Standard Products and Services Code (UNSPSC) [20] provides a multi-sector classification of products and services. Using this classification scheme to identify each service on all Jini, UPnP, Salutation and SLP devices would even improve interoperability.

Using service ontologies and user preferences on top of the aforementioned protocols is another solution that improves the quality of responses to service requests. Service ontologies describe relationships between different entities. By specifying rules and constraints on service attributes, for example by imposing a priority on each option based on user preferences, facilitates inexact matching. Two frameworks for describing ontologies are DAML+OIL [8] and OWL [14]. The OWL Services Coalition, formerly the DAML Services Coalition, have

recently described an ontology specifically for web services in these frameworks, namely DAML-S [17] and OWL-S [18]. Enhanced service discovery in Bluetooth using ontologies is discussed in [2].

B. Interoperability and network transparency

One thing that is not likely to change in the near future is the heterogeneity of available computing systems and devices. These systems will require self configuration. However Jini and Salutation do not address this issue. Before lookup services can be used, a Jini device needs to be assigned an IP address, subnet mask and optionally a gateway and DNS server. It would not be easy for Salutation, being a transport independent framework, to provide a generic solution. SLP and UPnP use DHCP for self configuration, however DHCP does not scale well to the Internet.

Another requirement is the interoperability between different software platforms. The wide variety of available service discovery protocols is the main reason why it is important to have bridges between these protocols to allow protocol interoperability.

Jini and UPnP interoperability is discussed in [1]. Virtual services are introduced as bridges between Jini and UPnP devices. The main problem to overcome is the fact that Jini relies heavily on mobile code for service discovery, while UPnP uses simple primitive types for communication. Therefore, Java interfaces and XML specifications need to be standardized.

The combination of Jini and Bluetooth is discussed in [24]. Bluetooth is used as the underlying network to get Jini objects to communicate. The main problem is that a Bluetooth device needs extra memory, processing power and storage for the Java classes.

A SLP-Jini bridge is discussed in [7]. When memory and processing resources for a Java Virtual Machine are too costly for an embedded device to advertise its services, SLP could be used instead. A *Java Driver Factory*, a class that can be used to produce Java objects, is used by the SLP-Jini bridge to instantiate a Java driver object based on the attributes of the SLP advertisement. The bridge then registers the service with a Jini Lookup server.

Salutation and Bluetooth interoperability is discussed in [12]. Because a Salutation Manager is independent of underlying protocols and operating environments, it can interface to numerous protocols, including Bluetooth. The main advantage is that the Salutation Architecture allows to advertise and register services, while Bluetooth does not. By mapping the Bluetooth service profile to the Salutation APIs and the Bluetooth SDP to the Salutation Manager, it is possible for an application to locate services in, for example Bluetooth and TCP/IP networks by communicating to the Salutation Manager.

Another problem is posed by the mobility of users and devices. Not only do network disconnections and different network connectivity cause problems, software can be mobile as well. A user might request to download a service to his PDA, for example a mail client. The Java Virtual Machine already provides support for code migration, but support for service migration between heterogeneous software platforms will be a must in the future. Also code mobility and state transfer of a remotely running application will be required, if the network connection is going to disappear due to user mobility. In most of the SDPs discussed above, services can only be run on the devices on which they are deployed.

C. Resource constraints

Embedded devices are known to have many resource constraints, like limited processing power, memory, network bandwidth and battery life time. Sometimes a trade-off needs to be made, for example between network bandwidth and processing power: will I run the application remotely or will I download it and run it on my personal device.

Today there already exist various software platforms [15] [10] to allow service implementations on embedded devices. However they are usually trimmed down versions with limited capabilities of large development platforms.

IV. RELATED WORK

In this section we discuss briefly UDDI and WSDL, as they are well known standards in the area of web service specifications and registries. Also mentioned is the Semantic Web, which will allow better semantic discovery and interaction. The aforementioned OWL is one of the specifications being incorporated into the Semantic Web.

A. UDDI

Universal Description, Discovery and Integration (UDDI) [13] is a standard for an online registry to enable the publishing and dynamic discovery of web services. Services are described according to an XML schema defined by the UDDI specification.

B. WSDL

The Web Services and Description Language (WSDL) [23] is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. These endpoints are network addresses associated with certain protocols and data format specifications. WSDL does not support semantic description of services and hence is therefore less suited for semantic discovery. It is often used in conjunction with SOAP.

C. Semantic Web

The Semantic Web [22] was defined by Tim Berners-Lee e.a. [3] to be an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation. To goal is to not only share data on the web for display purposes, but to allow this data to be used by computers for automation, integration and reuse of data across various applications, where computers have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning.

V. Conclusion

After an overview of the current state-of-the-art in service discovery protocols including Jini, UPnP, Salutation, SLP and Bluetooth, we discussed their applicability in an ambient intelligence context. The major problems are limited service discovery based on attribute matching, network transparency and platform interoperability. Semantic service discovery using service ontologies and user profiles with preferences could be used to enhance service lookup. Protocol bridges are existing approaches that only provide a partial solution to overcome the user mobility problem, as services need to be mobile as well and hence require heterogeneous platform interoperability for true code mobility. Future work will be focused on integrating these technologies, while keeping in mind the limitations of the current resource constraint devices.

References

[1] J. Allard, V. Chinta, S. Gundala, and G. G. Richard III. Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability. In *Proceedings of the 2003 International Symposium on Applications and the Internet (SAINT 2003)*.

[2] Sasikanth Avancha, Anupam Joshi, and Tim Finin. Enhanced service discovery in bluetooth. *IEEE Internet Computing*, 35(6):96–99, June 2002.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. Scientific American, May 2001.

[4] Bluetooth SIG. Bluetooth Core Specification version 1.2, volume 3 - Part B, November 2003.

[5] Yaron Y. Goland, Ting Cai, Paul Leachand, Ye Gu, and Shivaun Albright. Simple Service Discovery Protocol/1.0, IETF Draft draft-cai-ssdp-v1-03.txt, October 1999.

[6] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2, RFC 2608, June 1999.

[7] Erik Guttman. Service location protocol: Automatic discovery of IP network services. *IEEE Internet Computing*, 3(4):71–80, 1999.

[8] Ian Horrocks, Frank van Harmelen, and Peter Patel-Schneider. DAML+OIL, Darpa Agent Markup Language and Ontology Interference Layer, March 2001.

[9] ISTAG. Scenarios for Ambient Intelligence in 2010. http://www.cordis.lu/ist/istag-reports.htm, Februari 2001.

[10] Microsoft Corporation. Microsoft .NET Compact Framework.

[11] Microsoft Corporation. Understanding Universal Plug and Play: A White Paper. http://upnp.org/resources/whitepapers.asp, June 2000.

[12] Brent Miller. Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer, Bluetooth Consortium 1.C.118/1.0, July 1999.

[13] OASIS. The Universal Description, Discovery and Integration (UDDI), September 2000.

[14] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. Owl web ontology language guide, August 2003.

[15] Sun Microsystems Inc. JavaTM 2, Micro Edition (J2METM) Wireless Toolkit 2.1.

[16] Sun Microsystems Inc. Jini Architectural Overview, Technical White Paper, May 2003.

[17] The DAML Services Coalition. DAML-S: Semantic Markup for Web Services. http://www.daml.org/services/daml-s/0.9/daml-s.html, May 2003.

[18] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.html, November 2003.

[19] The Salutation Consortium. Salutation Architecture Specification (Part-1), Version 2.1, 1999.

[20] United Nations Development Programme and Dun & Bradstreet Corporation. The United Nations Standard Products and Services Code. http://www.unspsc.org/, 1998.

[21] UPnP Forum. Universal Plug and Play Device Architecture Reference Specification, Version 1.0.1, May 2003.

[22] W3C. Semantic Web. http://www.w3.org/2001/sw/.

[23] W3C. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001.

[24] Markus Wallmyr and Ola Markström. Bluetooth and Jini. The future of Networking, Department of Computer Systems, Institution of Information Technology, Uppsala University, 1999.