The Device Service Bus: A Solution for Embedded Device Integration through Web Services

Gustavo Medeiros Araújo
Federal University of Santa Catarina
Informatics and Statistics Department
Florianópolis, SC – Brazil – 88040-900
gustavo.medeiros@inf.ufsc.br

Frank Siqueira
Federal University of Santa Catarina
Informatics and Statistics Department
Florianópolis, SC – Brazil – 88040-900
frank@inf.ufsc.br

ABSTRACT
This paper presents a middleware infrastructure for integration of heterogeneous embedded devices in ubiquitous computing environments. The proposed infrastructure employs the Devices Profile for Web Services (DPWS) as the underlying integration technology, allowing devices that adopt different networking standards to interact with each other, through the use of interconnection devices and software components responsible for building a communication path among them. A prototype implementation of this middleware infrastructure is also described, and results of performance measurements obtained with this prototype are presented.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**]: Software Architectures. D.2.12 [**Software Engineering**]: Interoperability.

General Terms
Design, Standardization.

Keywords
Web Services, Interoperability, Embedded Systems, Ubiquitous Computing.

1. INTRODUCTION
The computing paradigm is rapidly changing from a computer-centric model to a highly distributed and mobile scenario, in which several devices are involved in the execution of computational tasks. These devices range from high-end devices, such as mobile phones and PDAs, to more limited equipment, such as wireless sensors and RFID tags. The communication technologies employed by these devices are different from servers and desktop computers, which avail from a stable networking infrastructure. In this scenario, devices build ad-hoc networks using wireless network protocols, and have to identify dynamically other devices and the services available on the network. This scenario characterizes what has been called a ubiquitous (or pervasive) computing environment [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.
Copyright 2009 ACM 978-1-60558-166-8/09/03…$5.00.

Recent developments in wireless and mobile networking led to the development of different networking standards, which are targeted at different applications and classes of devices. The integration of different classes of devices, which employ different networking technologies, is still an open research area.

In this paper we introduce a middleware infrastructure which aims to provide means for the integration of heterogeneous devices in ubiquitous computing environments. The proposed solution is based on the Web Services technology, which has been employed successfully as the integration media for business systems and for building distributed applications. In order to support the execution of web services on devices with limited computing and communication capabilities, the proposed architecture adopts the Devices Profile for Web Services (DPWS) [2], a standard tailored for the construction of web services in such environment.

The remainder of this paper is organized as follows. Section 2 presents the concepts and technologies adopted in the development of the proposed solution. The proposed middleware infrastructure for device integration is described in section 3. Section 4 presents the prototype implementation of the architecture, and performance measurements obtained with this prototype are shown in section 5. Section 6 compares the proposed architecture with similar projects found in the literature. Finally, section 7 summarizes the contribution of this paper and describes perspectives for further development in this field.

2. CONCEPTS AND TECHNOLOGIES
This section first analyzes the use of the Service Oriented Architecture (SOA) in the context of embedded systems. In the sequence, we describe the Devices Profile for Web Services (DPWS) – a SOA-based protocol stack that allows the interaction among heterogeneous embedded devices.

2.1 SOA and Embedded Systems
The Service Oriented Architecture (SOA) may be defined as a paradigm that allows the construction of loosely-coupled software components, which provide services to clients and may be dynamically located and invoked using a well-known communication protocol.

Web Services are the most popular technology for implementing service-oriented software. The Web Services technology adopts a set of standards for data representation and communication, so that service providers and consumers may interact. These standards - basically XML, HTTP and SOAP - are already available in most platforms, allowing the deployment of web services on virtually any computing device [3].

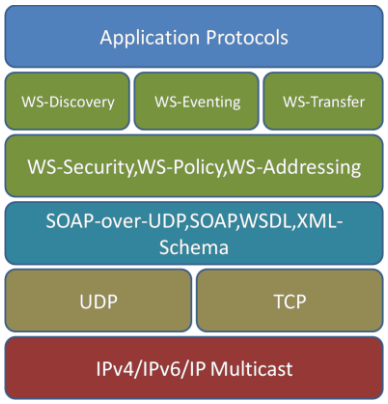

Figure 1 – DPWS Protocol Stack

SOA and Web Services have been successfully adopted in business environments, where the different information systems employed by a company for automating business activities are being turned into services which may easily interact with each other. This integration among systems allows companies to redirect their focus to business processes, instead of spending resources on system operation and maintenance. Despite being the most common scenario in which SOA is being used nowadays, this is not the only scenario in which SOA is capable of providing seamless integration among software [4].

During the last few years has been observed a sharp increase in the production of embedded devices and the ubiquity of these devices in our daily life. Such devices have also gained increasing processing power and novel features, which include the capacity of interacting with peers using communication technologies such as wireless networks, and also the capacity of running high-level network protocols such as the ones required by SOA. The availability of interconnection mechanisms allows interoperation among devices such as mobile phones and PDAs, industrial machinery, health care equipment and home appliances.

Despite the rapid growth in the manufacturing of embedded devices, the lack of widely adopted standards in this field is making difficult and sometimes even impossible the exchange of information among such devices. Some technologies, such as UPnP [5] and Jini [6], have been proposed aiming to standardize the communication and service location mechanisms, allowing the interaction among devices. Jini provides a robust solution for allowing device interoperability, but it is limited to the Java platform. UPnP adopts well-known, implementation-independent Internet standards such as HTTP, SOAP and XML for communication among devices, but this standard is also constrained by the boundaries of the communication network.

The next section describes DPWS, a protocol specification proposed by a group of companies, which intends to allow interoperability among devices and is not constrained by network boundaries or by characteristics of the computing platform.

2.2 Devices Profile for Web Services (DPWS)

The Devices Profile for Web Services (DPWS) is a protocol stack that provides high-level communication mechanisms for device interoperability. Aligned with the standards adopted by web services, DPWS becomes a link, in the embedded systems scenario, to the world of SOA-based applications, offering to

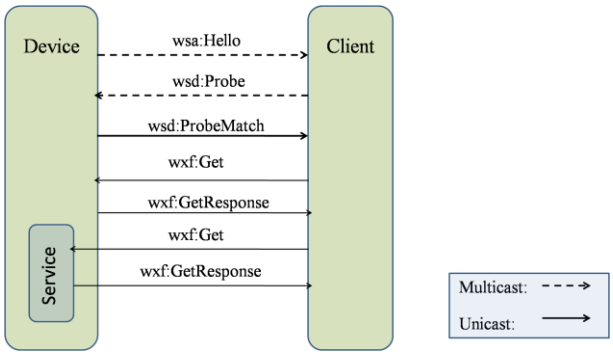

Figure 2 – DPWS Messages

embedded applications the same level of interoperability already available in business applications. The protocol stack presented by Figure 1 shows how DPWS leverages from Internet protocols such as TCP, UDP (both multicast and unicast) and HTTP. For message exchange, both SOAP-over-UDP and SOAP-over- HTTP are used.

DPWS allows the construction of two kinds of services:

- Hosting Services: service that hosts other services and receives different kinds of messages. In general, it represents the device in which services are hosted.
- Hosted Service: service hosted by a hosting service, with a lifetime that is limited by the lifetime of the hosting service. Hosted services are visible on the network, and are addressed independently from the hosting service.

As shown by Figure 1, the standard protocols adopted by the Web Services technology are also leveraged by DPWS. These are:

- *WS-Addressing*: groups all the network addressing information in message header fields, such as 'To:' and 'Reply to:'. This allows the SOAP protocol to be independent from the underlying transport protocol.
- *WS-MetadataExchange*: provides dynamic access to metadata describing hosted and hosting services.
- *WS-Discovery*: provides plug-and-play mechanisms for device location. The device discovery is performed using a multicast protocol for locating services available on the network. To extend the discovery to other networks, the standard defines a Discovery Proxy, which also reduces the network traffic generated by multicast communication.
- *WS-Policy*: provides a way to add information to a WSDL description in the form of policies supported by services.
- *WS-Eventing*: defines operations for event notification, allowing services to publish/receive asynchronous messages.
- *WS-Security*: provides mechanisms for securing the interaction among services, guaranteeing security properties such as authenticity, confidentiality and integrity.

The metadata of hosting and hosted services is divided into four sections:

- *ThisModel*: describes characteristics that are common to a group of devices that belong to the same class, such as manufacturer name and URL, model name and code.
- *ThisDevice*: describes characteristics that may be different from one device to another, such as *SerialNumber*, *FriendlyName* and *FirmwareVersion*.

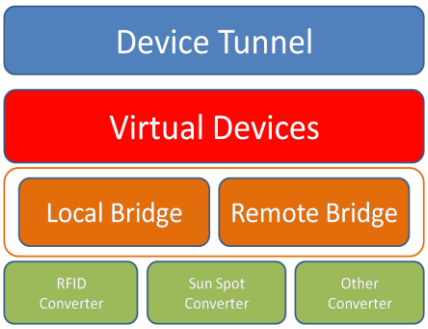

Figure 3 – The Device Service Bus

- *Relationship*: describes the relationship among services, giving details about a hosting service and its hosted services.
- *WSDL*: describes all operations, faults and data structures supported by a particular hosted service.

The messages exchanged during the discovery process by devices, its services and clients are shown by Figure 2. A *Hello* message is an advertisement multicast by a device that has just joined the network. By multicasting a *Probe* message, a client tries to locate hosting services (i.e. devices) with particular characteristics, which respond with *ProbeMatch* messages containing metadata in which the hosted services are listed. The available services may be described issuing *Get* messages, which result in *GetResponse* messages being returned to the client, containing the description of the queried service.

3. THE DEVICE SERVICE BUS

The Device Service Bus (DSB) is a software infrastructure based on a lightweight, portable platform which may run on machines ranging from a workstation to a small device. Following the SOA principle, DSB provides a way to integrate devices by acting as a broker between service providers and consumers. The main goal of this infrastructure is to create a tunnel that provides a way to expose devices which employ specific communication technologies, such as RFID [7] and Bluetooth [8] as web services. As shown in Figure 3, the *Device Tunnel*, which implements the DPWS stack [2], exposes devices and services found by *Bridges* and *Converters*. The Bridge handles all Converters available and is responsible for connecting Virtual Devices to the core of the architecture. The Converter, in the bottom of the stack, handles technology-specific issues for all devices which employ that particular technology. Therefore, DSB allows non-DPWS devices to interact with other devices, including DPWS-capable devices, through the provided communication infrastructure.

3.1 DSB Components

The architecture is composed by the following components, as shown by Figure 3:

- *Device Tunnel*: is a Hosting Service which implements the DPWS specification. The Device Tunnel can discover other Device Tunnels and other DPWS-capable devices.
- *Virtual Device*: provides a DPWS interface for a non-DPWS device. Each Virtual Device can host one or more Virtual Services, which represent services implemented by non-DPWS devices. Moreover, a Virtual Service may have one or more Virtual Action and Virtual Events, which represent actions and events implemented by these services.

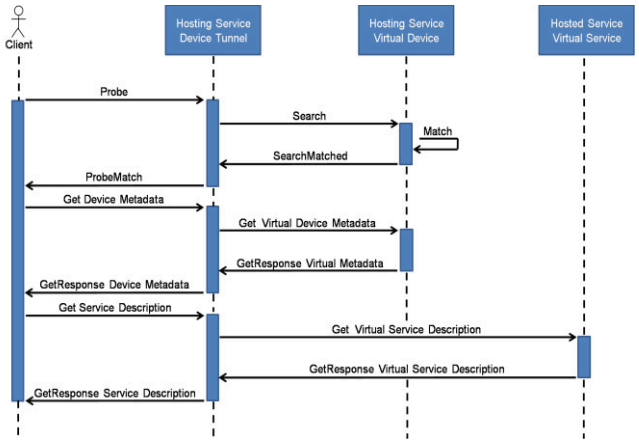

Figure 4 – Device and Service Discovery

- *Bridge*: maintains the Virtual Device cache. When a Virtual Device is added or removed by the Bridge, Hello and Bye messages are sent, according to the DPWS specification [2]. The Bridge allows the interaction among DPWS devices and Virtual Devices. A Bridge can be either local (i.e. may run on the actual device) or remote. Remote Bridges are employed when the device is not capable of running the whole DSB stack. The Remote Bridge is similar to the Discovery Proxy defined by the WS-Discovery specification [9].
- *Converter*: knows details about the device implementation, its interface and services. The Converter can manage more than one device that employs the same networking technology. The Converter is responsible for extracting device metadata and service descriptions and must know how to invoke the services provided by managed devices.

The actual devices plugged to the DSB Stack have their own lifecycle, and may become off-line without any advertisement. To deal with this situation, the Converter employs a keep-alive mechanism, which tracks all managed devices. If a communication timeout happens, the Converter notifies the Bridge, which removes the device from the Virtual Device Cache. The number of devices that can be plugged into the stack and the timeout employed by the keep-alive mechanism may be configured, and the keep-alive mechanism can be switched off in constrained environments in order to save bandwidth and energy consumption.

3.2 DSB Dynamics

As shown by Figure 4, when clients send DPWS messages with the intent of searching for a device, asking for device metadata or for a service description, the Device Tunnel deals with the incoming requests. The exact description of the Virtual Device and its Virtual Service(s) is taken from the Virtual Device Cache.

Figure 5 illustrates the service invocation process. If a service is invoked by a client, the Device Tunnel forwards the invocation to the Bridge. Then, the Bridge redirects the incoming request to the corresponding Converter. Similarly, the Converter forwards the request to the corresponding device, which performs the service. Furthermore, the Converter can wait for the service to be concluded in a request-response interaction. After the device has executed the service, the response is transferred to the client through the DSB stack.

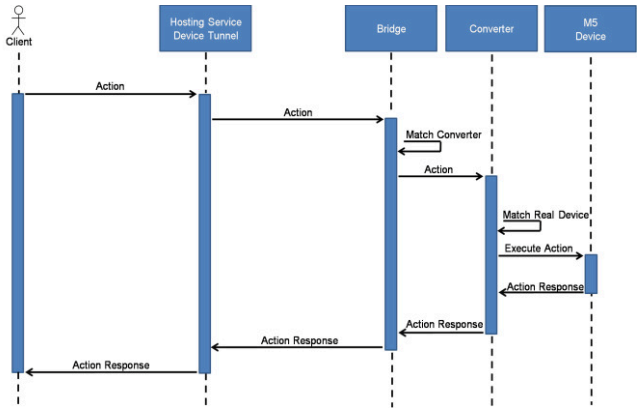

Figure 5 – Service Invocation

4. IMPLEMENTATION

A prototype implementation of the DSB has been built on top of the WS4D framework [7], which was employed for developing the Device Tunnel. The WS4D framework was extended to deal with virtual devices and services. The remaining components of the stack were implemented in Java ME CDC 1.1.2.

Each virtual device has its own metadata, with data corresponding to the description of the actual device – i.e. serial number, name and model number, manufacturer name, friendly name and so on – in order to comply with the DPWS specification. Virtual devices encapsulate descriptions of virtual services, establishing a one-to-many association among them. The Device Tunnel exposes as web services all virtual devices and their virtual services, associating a particular endpoint with each of them. The endpoint address is used by the Device Tunnel for demultiplexing requests and to tell the bridge which virtual device and service have been invoked. The bridge keeps in its cache a list of converters, and each converter provides an interconnection point for devices which use a particular communication technology.

In this prototype implementation, illustrated by Figure 6, a converter has been implemented for RFID equipment. The prototype was tested with the Sun Java Toolkit 1.0 for CDC and with the Mercury M5 and M5e RFID readers. A converter for Bluetooth devices [8] is also being developed. Devices with support for Java ME CDC, such as some smartphones, PDAs and SunSpot sensors [11] are considered native DPWS devices and do not require a converter to interact with other devices.

5. PERFORMANCE TESTS

Tests have been performed with the intent of evaluating if the Device Service Bus (DSB) was capable of handling a large amount Virtual Devices and of dealing with simultaneous requests to Virtual Services. These tests were executed on a testbed with the following configuration:

- DPWS Stack with WS4D extended.
- Sun Java Toolkit 1.0 for CDC as client.
- Client running on a Pentium 4 with 2.4 GHz processor with 512 MB of memory, with Windows XP operating system.
- DSB (Local Bridge) implemented using Java SE 1.5, executed on a Intel Core 2 Duo 2.0 GHz processor with 2GB of memory, running the Windows Vista operating system.

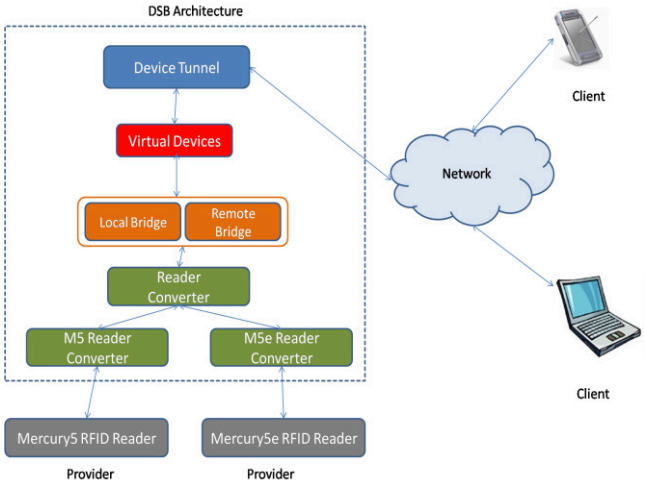

Figure 6 – Prototype Implementation

- Wireless router 802.11g with 2.4GHz.
- Two types of passive tags (UMP RAFLATAC shortdipole model and ALN-9540)
- Mercury M5 and M5e RFID readers.

The experiment was performed with two actual devices plugged to the DSB – the Mercury M5 and M5e RFID readers, shown in Figure 6 – and up to 98 simulated Virtual Devices plugged to the DSB. Each simulated Virtual Device had one Virtual Service and each Virtual Service had one Virtual Action with one input parameter and one output parameter. Each RFID reader had one service and one action, which was responsible for reading RFID tags and returning the contents of the tags read.

All requests were made remotely to the Virtual Device that represented the RFID reader. Therefore, the DSB had to locate the Virtual Device in its cache to send the correct probe match, device metadata, and service description. The client and the Bridge were running on different machines, with the M5e RFID Reader connected locally to the Bridge and the M5 Reader connected directly to the network. One hundred requests were performed in each round.

The first performance tests measured the response time to locate a device. The discovery process was considered the time to send a probe request, to process the probe match response, to get the Virtual Device metadata and the Virtual Service description. The average time to locate a Virtual Device and Virtual Service through the DSB stack is 38,74 ms for the M5 reader, and 32,86 ms for the M5e. It is important to notice that the discovery process requires three exchanges of messages through the network, as shown by Figure 2. One request is a SOAP-over-UDP multicast message to probe devices and the other two requests are sent through SOAP-over-HTTP to get device metadata and service description.

The second test consisted in measuring the invocation time, varying the amount of data returned by the RFID reader. The invoke process was considered just the time to invoke the action and to receive the response, which ranged from 10 to 100 KBytes. Figure 7 presents the results of these tests. The results show that the response time grows at a lower rate than the return size, showing that the invocation mechanism works well even for services that return data at a high transmission rate.

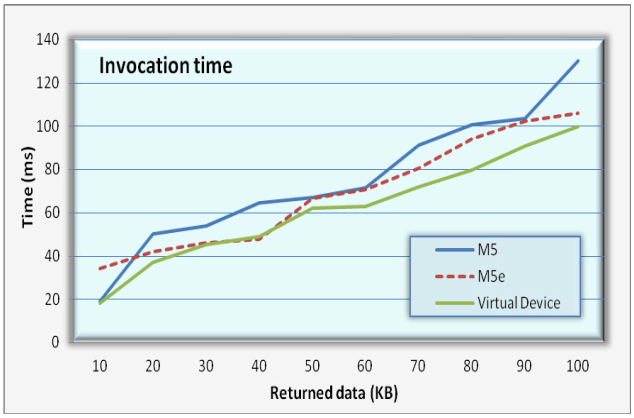

Figure 7 – Average Response Time

6. RELATED WORK

Several research efforts have been made with the aim of improving interoperability among devices, such as [12], which proposes interconnecting Bluetooth and RFID devices, using the first as the enabling technology for RFID reader mobility. On the other hand, projects such as [13] and [14] employ the same approach adopted by DSB, allowing devices provided with different networking technologies to be interconnected. However, the solutions presented by [13] and [14] require a robust infrastructure to manage devices and services available on the network. The solution proposed in this paper requires a more lightweight and portable core, which may be deployed on devices with limited resources such as a PDA or a Smartphone. In order to achieve this, the DSB stack can be configured according to the resources available in the hosting device. Besides, since DSB may be available in multiple devices, several access points may be created in order to exchange information through the DSB stack.

According to [2], the fact that DPWS employs a discovery proxy allows the area coverage to be extended, and also transfers the processing load generated by the discovery process to a different machine, allowing more limited devices to save resource consumption. In the DSB stack, the Bridge may run remotely, such as a discovery proxy, in a way that allows devices to benefit from less resource consumption.

The core of the DSB project is the DPWS implementation, which allows the project to profit from the benefits of a loosely coupled service-oriented architecture, built on top of largely available protocols such as SOAP, HTTP and UDP (both multicast and unicast). The plug and play capability, which is not a characteristic present in regular web services, allows the dynamic discovery of services and simplifies the effort necessary for exposing services provided by devices. This feature is employed by DSB to expose services and devices which do not support DPWS natively.

7. FINAL REMARKS

This paper introduced the Device Service Bus, a middleware infrastructure for integration of heterogeneous devices, which is based on the Devices Profile for Web Services (DPWS). DSB integrates devices that employ different communication technologies, such as RFID and Bluetooth, allowing devices and services provided by them to be exposed as web services.

The proposed solution for device integration is much lighter than other solutions proposed in the literature, being able to run on devices such as mobile phones and PDAs. Nonetheless, more limited devices such as wireless sensors and RFID tags may also be integrated through interconnecting devices, which host bridges and connectors tailored for a particular networking technology.

Currently we are working on the support for Bluetooth devices and SunSpot sensors. In the near future we intend to keep working on connectors to incorporate even more networking technologies to the Device Service Bus, in order to allow seamless integration of different classes of embedded devices in a truly ubiquitous and heterogeneous environment.

8. REFERENCES

[1] Weiser, M. Hot Topics: Ubiquitous Computing. IEEE Computer, 26(10):71–72, October 1993.

[2] Microsoft Corporation. Devices Profile for Web Services (DPWS). February 2006. Available at http://schemas.xmlsoap.org/ws/2006/02/devprof/.

[3] W3C. Web Services Architecture. February 2004. Available at http://www.w3.org/TR/ws-arch/.

[4] Machado, Guilherme Bertoni et al. Integration of Embedded Devices Through Web Services: Requirements, Challenges and Early Results. Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06). Calgary, Italy, June 2006.

[5] UPnP Forum. UPnP Device Architecture v1.0. July 2006. Available at http://www.upnp.org/resources/documents.asp.

[6] Sun Microsystems. JINI Specifications Archive v2.1, 2005. Available at http://java.sun.com/products/jini/2_1index.html.

[7] Want. R. An Introduction to RFID Technology. IEEE Pervasive Computing, Vol 5(1), January 2006.

[8] Chatschik, B. An Overview of the Bluetooth Wireless Technology. IEEE Communications Magazine, Vol. 39(12), December 2001.

[9] Microsoft Corporation. The Web Services Dynamic Discovery (WS-Discovery). April 2005. Available at http://specs.xmlsoap.org/ws/2005/04/discovery/.

[10] Zeeb, Elmar et al. Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services. 21st International Conference on Advanced Information Networking and Applications Workshops, 2007.

[11] Sun Microsystems. Getting Started with Sun SPOT, 2008. Available at http://www.sunspotworld.com/docs/.

[12] Siegemund, F., Flörkemeier, C. Interaction in Pervasive Computing Settings using Bluetooth-Enabled Active Tags and Passive RFID Technology together with Mobile Phones. Proc. IEEE PerCom 2003, March 2003.

[13] Yim, Hyung-Jun et al. Design of DPWS Adaptor for Interoperability between Web Services and DPWS in Web Services on Universal Networks. Proceedings of the IEEE International Conference Convergence Information Technology (ICCIT), November 2007.

[14] Raverdy, Pierre-Guillaume et al.: Efficient Context-aware Service Discovery in Multi-Protocol Pervasive Environments. Proceedings of the IEEE 7th Intl. Conference on Mobile Data Management (MDM'06), May 2006.