# UbiGate: A Gateway to Transform Discovery Information into Presence Information

Bissyandé Tegawendé, Laurent Réveillère, Yérom-David Bromberg

# UbiGate: A Gateway to Transform Discovery Information into Presence Information

Tegawendé Bissyandé
ENSEIRB
Department of Telecommunications
1 av. du docteur Sweitzer
BP99 - 33402 Talence Cedex, France
bissyand@enseirb.fr

Laurent Réveillère David Bromberg
University of Bordeaux 1
351 cours de la Libération
F-33405 Talence Cedex, France
{reveillere, bromberg}@labri.fr

## ABSTRACT

Pervasive computing involves various entities which need to coordinate tasks and share resources through different service discovery protocols. However, the multiplicity and the incompatibility of those protocols have made interconnectivity problematic. Moreover, most service discovery protocols require a strong participation of users to genuinely play their part. Consequently, service discovery in a pervasive environment has become a challenge that researchers as well as practicionners have tried to overcome through various approaches. Nevertheless, existing solutions mostly consist of designing new protocols which usually address specific application needs while participating in the increase of heterogeneity.

To address these problems, we present a new paradigm for service discovery involving the use of a gateway, called UbiGate, and relying on SIP, a widespread signaling protocol. Centered around the notion of presence, UbiGate enables real time availability of service information while hiding the heterogeneity of underlying protocols. We have developped a prototype of UbiGate supporting service discovery protocols such as the protocol used in Bluetooth service discovery mechanism and a protocol enabling the detection mechanism of RFID. Preliminary results show that UbiGate enables new service discovery protocols, either IP or non-IP based, to be seamlessly supported with no significant overhead in discovery latency.

## Keywords

UbiGate, SIP, Presence, Pervasive, SDP, Service Discovery

## 1. INTRODUCTION

*Ubiquitous Computing*, otherwise known as pervasive computing, refers to a computing paradigm where the user is constantly in interaction with computing devices [11]. As envisioned by Weiser, pervasive computing *takes into account the natural human environment and allows the computers themselves to vanish into the background* [27]. In such environments, computers surrounding the user share communication resources and computing capacities packaged as services. The pervasive computing scheme therefore aims at enabling the use of those services in the most seamless and effortless possible way. Users collect information that describes attributes of available services using Service Discovery Protocols (SDPs). Currently, pervasive computing makes use of a wide range of SDPs [2, 8, 9, 12, 15, 16, 24, 25, 26]. Therefore, a user will be isolated if his device lacks the capacity to discover resources available in its vicinity because of protocol incompatibility.

Existing SDPs are either based on the *Pull model* or the *Push model*. In the *Pull model*, service requesters regularly send requests to discover services available in their environment. In the *Push model*, service providers automatically announce the services they provide by sending advertisements. However, neither model correctly address the constraints of pervasive computing. Indeed, the volatility of devices requires service requesters to reiterate their discovery requests to get up-to-date information when using the *Pull Model*. In the *Push Model*, advertisements sent by service providers are performed cyclicly. Hence, users must wait for the next advertisement to get accurate information. The dynamicity of pervasive environments suggests that users should be notified when the state of a service changes in order to provide realtime snapshots of the context in which they operate. Such a realtime service discovery can be compared to the notification of contact status in Instant Messaging (IM) clients such as Windows Live™ Messenger or Google Talk™. In those IM clients, the user performs a realtime monitoring on the presence status of each member of its *buddy list*. This *ability to access realtime information about a person's status, communications capabilities, and preferences* as suggested by Rosenberg in [20] is relevant to virtually every means of communication.

So as to convey presence information dynamically, various instant messaging protocols are available, such as XMPP (also known as Jabber) [22], IRC [17] and SIP/SIMPLE [13, 19, 21]. In particular, the Session Initiation Protocol, otherwise known as SIP, is widely adopted for its simplicity, portability and extensibility[10, 20].

Taking into consideration both the aforementioned challenges of Service Discovery in pervasive computing and the


*SIPE'09* July 13–17, 2009,London, UK

SIP widespread adoption, we introduce a new service discovery paradigm based on SIP.

## This paper

This paper presents a new approach for service discovery based on the SIP protocol. It consists of exposing discovery information about services as presence information for users. To do so, we introduce the UbiGate gateway which is in charge of transforming discovering information into presence information. UbiGate handles all discovery tasks in the environment and delivers pertinent information to each user.

The contributions of this paper are as follows:

- We have proposed a new approach for service discovery involving the use of a gateway and relying on the widely used SIP protocol. Following our approach, we present UbiGate: a gateway to transform discovery information into presence information;
- We have developped dedicated modules for UbiGate to support RFID and Bluetooth technologies. Our preliminary experiments show that UbiGate does not introduce any significant performance penalty in service discovery while enabling support of new services to be easily added.

The rest of this paper is organized as follows. Section 2 introduces our approach and the architecture of the UbiGate gateway. Section 3 assesses our prototype implementation of UbiGate, demonstrating its benefits in a pervasive environment. In Section 4, we present other approaches that were proposed as attempts to unify the mechanisms of service discovery. Finally, Section 5 concludes and presents future work.

# 2. OUR APPROACH

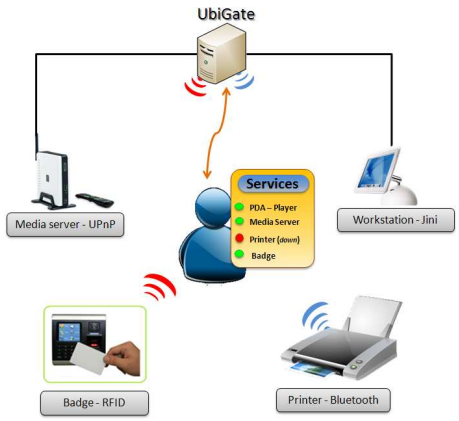


**Figure 1: Interaction among pervasive computers, users and UbiGate**

The most important concern in service discovery within pervasive computing is the interconnectivity among communication entities. Yet, the variety of protocols, because of their incompatibility, has made it difficult for one device to keep track of the availability of existing services. The underlying challenges are to enable service discovery using a unique protocol, and to improve reactivity, considering the volatility of pervasive environments. Our approach consists of building a gateway in charge of transforming service discovery messages into presence information for end-users. This gateway, called *UbiGate*, exchanges information with users through the SIP protocol, while managing various native discovery protocols, as illustrated in Figure 1.

UbiGate aims at unifying service discovery protocols so as to make them transparent for users. Besides, it is pertinent to suppose that the appearance of new protocols will facilitate the advertisement of more services that may interest users. Therefore, the infrastructure has been designed to easily welcome new protocols. In UbiGate, protocols are plugged as modules packaging their capabilities.

Practically, UbiGate enables service discovery through SIP compliant devices. Using the SIP protocol, users can address discovery requests to the gateway which will manage all tasks for information gathering. SIP is thus the pillar of our approach. So as to better understand how it sustains the infratructure, an overview of the protocol is presented.

## 2.1 SIP background

SIP is originally a signaling protocol for Voice over IP (VoIP) and third generation mobile phones. It is standardized by the IETF and adopted by the ITU.[1] This protocol enables creating, modifying and terminating a communication between parties. Communications include audio/video communications, games, and instant messaging.

The SIP protocol is based on a client-server model. A SIP message can be sent or received by a client. A sent message is said to be *outgoing*; otherwise, it is said to be *incoming*.

SIP is a text-based protocol similar to other well-known protocols such as HTTP[2] and RTSP.[3] A SIP message begins with a line indicating whether the message is a request (including a protocol method name) or a response (including a return code). A sequence of required and optional headers follows. Finally, a SIP message includes a body containing other information relevant to the message.

Logically, SIP is composed of three main entities: a *registrar server* to allow users to record their current location, a *proxy server* in charge of dispatching SIP messages, and a *user agent* required on each communication device to perform all SIP-related actions.

To support mobility, a user is assigned a SIP URI (Uniform Resource Identifier), which is a symbolic address, analogous to an e-mail address. When a SIP proxy receives a message for a local URI, it asks the local registrar server to translate the URI into contact information for a specific user agent. A user must thus inform the registrar server of the user agent at which he would like to receive messages.

Following the success of SIP, the IETF has produced many specifications related to presence and instant messaging with SIP. This set of specifications is known as SIMPLE[4] and covers topics ranging from protocols for subscription and publication to presence document formats.

## 2.2 UbiGate architecture

The UbiGate gateway aims at meeting the challenges of service discovery in pervasive computing by leveraging the

[1]ITU: International Telecommunications Union.
[2]HTTP: HyperText Transfer Protocol.
[3]RTSP: Real Time Streaming Protocol.
[4]SIMPLE: SIP for Instant Messaging and Presence Leveraging Extensions.

SIP protocol. The architecture of UbiGate is based upon the three major components illustrated in Figure 2.

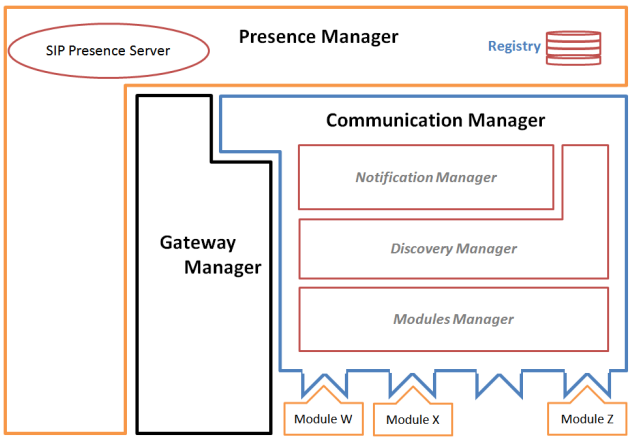


**Figure 2: UbiGate architecture**

The main issue that is addressed by the UbiGate gateway is the reliability of the communication channel between UbiGate and users. Indeed, service requesters rely on UbiGate to perform discovery tasks and regularly convey back to them appropriate information. So as to ensure a reliable and efficient access to information, service attributes are stored in a centralized registry managed by the gateway. Indeed, because of the dynamicity of pervasive environments, distributed storage systems are hard to deploy and maintain efficiently. Considering the capabilities of SIP, UbiGate deploys an event framework in charge of managing a SIP presence server. This *Presence Manager* handles the storage of services as entries, and matches requesters' subscriptions with those entries.

In order to discover services, UbiGate uses a service broker. This broker handles the native discovery protocols supported by the gateway. The *Communication Manager* (i.e. the service broker), is in charge of activating service discovery sessions and recovering the attributes of services that are available in the environment. This component cooperates with the previous one for service information storage.

The *Gateway Manager* enforces the coordination between the components. It therefore enables them to conjointly perform discovery and notification tasks. This component also provides a control interface for the administration of UbiGate, allowing users to easily add new features.

Furthermore, since our approach aims at providing an infrastructure that is as much extensible as possible, UbiGate's suported discovery protocols are integrated through modules. For a given discovery protocol, the module packaging its capacities can be plugged into the gateway and removed as well without impacting the behaviour of existing protocols. This design feature allows UbiGate to be easily integrated in the environment so as to limit the impact of changes for end-users. Typically, the gateway renders seamless the upgrades on discovery protocols as well as disruptions in environment settings. Indeed, service providers are not required to switch to or add new protocols in their service advertisement kernel. In point of fact, it is up to the gateway to adapt itself to the environment by integrating in its core new modules for supporting available discovery protocols.

## 2.3 Service Discovery using UbiGate

Service discovery through UbiGate can be split in two distinct activities: the discovery process activity which collects service information, and the notification activity during which information is delivered to service requesters.

UbiGate uses a *Pull model* for its discovery processes. Practically, all the native service discovery protocols managed by the gateway are regularly called upon to scan the environment so as to provide up-to-date information on available services. However, as far as service requesters are concerned, the discovery system is perceived as running a *Push model* that works in real time. Indeed, an advertisement is performed each time the context changes. This scheme allows UbiGate to unify service discovery mechanisms while improving the *Push model*. It is this notion of realtime notification that has encouraged the choice of SIP.

A user wishing to benefit from the capacities of UbiGate is required to register itself as a service requestor by subscribing to the presence of a service. The subscription sent to the gateway is performed by sending a SIP SUBSCRIBE request, including an URI[5] that represents the targeted service. For the purpose of guaranteeing a uniform interaction with the gateway, we have introduced an ad-hoc format for encoding relevant subscription information into URIs. This format indicates the type of service, or/and the provider's name, and/or the preferences on the protocol through which the service is advertised.

```
sip:UbiAny.Printer.UbiAny@192.168.200.18
```

**Figure 3: Example of subscription URI**

The service request format thus provided allows requesters to target a specific type of service on a given device while filtering the advertisement protocol. The latter option can be useful if the requester is sending requests on behalf of other entities that have strict constraints.

As illustrated in the example of Figure 3, keywords such as *Printer* may be used so that communicating entities may understand the significance of the terms utilized.

To discover the full range of available services in its vicinity, the client can send beforehand a SUBSCRIBE [19] request using the joker *UbiAny*. The gateway will then inform the client of all service types available through a list of keywords. Using them, the user can then properly transmit its subscription to the gateway.

```
<note> Bluetooth.OBEXObjectPush.0C:19:73:1F:1C </note>
```

**Figure 4: Compact description of a service**

Service attributes requested by a user are described in a standard PIDF[6] XML document attached to SIP NOTIFY [19] messages. To enable standard, unmodified, SIP user agents such as Windows Live Messenger to correctly process such a PIDF document, we use the optional *note* tag recommended by the RFC [23] for additional information, as illustrated in Figure 4. Each of these service descriptions contains only the essential information that characterizes a service: the service type, the address of the entity offering

[5]URI: Uniform Resource Identifier.
[6]PIDF: Presence Information Data Format.

the service, and the native protocol used by the provider to announce the service.

Thus, the compact description proposed in Figure 4 presents a *OBEX Object Push* service advertised by the Bluetooth Service Discovery Protocol supported by a Bluetooth-enabled device which MAC address is '0C:19:73:1F:1C'. This type of description is provided when requesters specify in their requests the service type and the protocol. The presence server then supplies the provider's address as requested information.

However, it is noteworthy to mention that the flexibility of SIP can be exploited so as to deliver better descriptions of services. Indeed, besides the essential attributes listed previously, a user may be interested in knowing the name of the service provider, its state along with the possible functionalities it yields. To deliver such an extended description, we have designed a new Service Discovery Information Format that is an extension of PIDF. In this new format, instead of using the PIDF *note* tag, UbiGate fills a *service* tag with all attributes values. The new format thus allows a more enhanced description of services as illustrated in the example of Figure 5.

```
<service id="service-101">
   <type> Printer </type>
   <name> Zeus </name>
   <brand> Xerox </brand>
   <protocol> RFID </protocol>
   <address> A0:23:1C:34:17 </address>
   <state> Ready </state>
   <functions>
      <function> Printer </function>
      <function> Copier </function>
      <function> Scanner </function>
   </functions>
</service>
```

**Figure 5: Extract of the extended description of a service**

The *protocol* tag in the Service Discovery Information Format contains a value that refers to the protocol used by the service provider to advertise the service. Thus, the value 'RFID' has been placed in the example of Figure 5 because an RFID tag has been attached to the printer *Zeus*, and it is the detection of this tag that has provided all the information. This extended description format has been designed to include all informations provided by Bluetooth Service Description Protocol as it was the experimental protocol used in UbiGate. Yet, the extensibility of the PIDF format guarantees future adaptations of the format to include any additional attribute proposed by any service discovery protocol. Moreover, the format proposed is flexible since the parser does not require all services to be completely detailed. Besides, *note* tags can be used when the description proposes more information that need to be conveyed to the requester.

***Advanced Features.*** With UbiGate, a user can subscribe to any available services, without any concern for diversity of entities that provide those services, nor for the protocols they use for advertisement. However, usually, users require advanced discovery features based on service providers or more often on protocols used for service advertisement. Typically, if we consider a user that needs to discover, with his PDA, the services advertised by his desktop computer, it is important to allow him to subscribe to the computer rather than to all service types that UbiGate can find in the entire environment. Likewise, a user wishing to track all RFID tags that penetrate his environment should receive information related only to the protocol used for RFID. Thus, UbiGate provides an optimisation of discovery requests. This optimisation consists of enabling a selection for the options mentionned above by including fields in subscription URIs.

# 3. ASSESSMENT

To assess our approach, we have developed a prototype implementation of UbiGate. This prototype includes about 10,500 lines of Java code, and relies on the SIP presence server[7] provided by NIST (National Institute for Standards and Technology).

For the purpose of presenting the contributions of UbiGate, we have considered two case studies involving the Bluetooth technology and RFID which is increasingly adopted for tracking purpose.

First, we consider a building manager who needs to be informed in real time when items of great value cross specific gates of his building. These items have been associated with RFID tags. As an important constraint, the manager should be able to receive information on his desktop or laptop computer and any SIP compliant device. To enable this scenario, we have integrated in UbiGate support for RFID by developing a module, as described in Section 2.2. This module consists of about 200 lines of Java code. For our experiments, we used the ASPX RFID Kit (Icode 2) RW-310 with C-100 Converter.

In our second case study, we consider that a user with a Bluetooth-enabled device will likely scan the environment several times to discover available Bluetooth services. Since, inquiries in Bluetooth respect the *Pull model*, service discovery is no longer seamless nor effortless because it involves perpetual manipulation from users and/or permanent inquiries from devices. Using UbiGate, the user only has to subscribe to the presence of Bluetooth entities and their services. To do so, we have developed a Bluetooth module consisting of about 1,100 lines of Java code and based on the *AvetanaBluetooth*[8] stack.

As described in Section 2.3, creating well-formatted requests requires that users know their syntax, which can be a complex task. To overcome these limitations, we have developed a SIP User Agent that is compliant with UbiGate's new format for service description. It provides a powerful graphical interface simplifying the subscription steps. We have tested the presence server of UbiGate using SIP compliant standard clients such as *SJPhone*[9] and *SIP Communicator*[10].

Within the same perspective, we have implemented an interface for UbiGate's administrators. They can use the interface to start pulling with a specific service discovery protocol even if there is no subscription yet. Thus, when requesters address their requests to UbiGate, the latter has already information to deliver. The interface also allows shutting down the gateway or disabling unused protocols.

Table 1 lists the sizes of the implementations for UbiGate's different components. It also includes the sizes of the two

[7]NIST SIP: http://www-x.antd.nist.gov/proj/iptel/
[8]Avetana web page: http://www.avetana-gmbh.de
[9]SJPhone: http://www.sjlabs.com
[10]SIP Communicator: http://sip-communicator.org

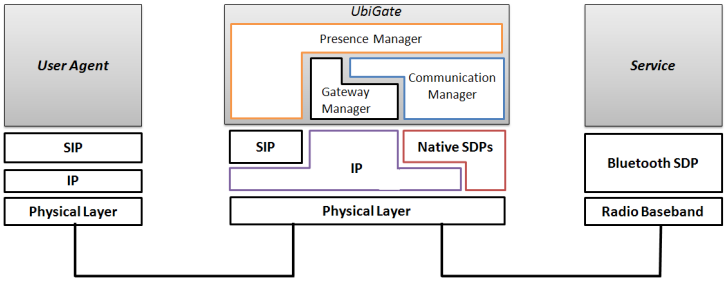


**Figure 6: UbiGate service discovery stack (*Example with Bluetooth SDP*)**

modules required for Bluetooth and RFID support.

| | *Component/Module* | *Package size* |
|---|---|---|
| **UbiGate** | Gateway manager | 4.8 Kb |
| | Communication manager | 25.1 Kb |
| | SIP Presence Server | 4.1 Mb |
| | ***Total size of UbiGate*** | ***4.13 Mb*** |
| **Modules** | Bluetooth | 22.2 Kb |
| | RFID | 6.1 Kb |

**Table 1: Package size of UbiGate components and modules**

We now present preliminary results of a performance evaluation of our UbiGate implementation. For our experiments, the gateway has been deployed on an Intel Pentium 4 computer with CPU frequency of 3.06 GHz and 1 MB of memory. We have also setup a native application using the Avetana stack directly. Due to specificities of Bluetooth technology, only first inquiry sessions are relevant. Indeed, the service discovery process is then complete and the devices do not use any information from previous inquiries to accelerate information exchanges. The comparison in Table 2 shows that there is no significant overhead using UbiGate compared to a direct discovery process using the Avetana Bluetooth stack. Indeed, over the 31 seconds required for bluetooth service discovery completion, UbiGate takes less than 1 second to reformat informations data and convey them through the network for display on the client side.

| | **Native** | **UbiGate** |
|---|---|---|
| Delay $1^{st}$ inquiry | $\approx$ 31 seconds | $\approx$ 31.7 seconds |

**Table 2: Comparison of Bluetooth inquiry delays**

We have also deployed Ubigate on an OSGI $^{TM}$ platform in order to facilitate its deployment, migration and the management of dependancies. For this, we have encapsulated UbiGate's components as OSGi bundles. The OSGi bundles for the modules enable dynamic upgrades of UbiGate, while presenting roughly the same size of code as the non-OSGi version. Our OSGI based implementation of UbiGate has been deployed and run successfully on the knopflerfish[11] OSGi platform. The OSGi bundles as well as the complete Java code of the UbiGate project are available at the project web page[12].

Finally, it is noteworthy to indicate that UbiGate infrastructure allows new service discovery protocols (such as Jini, UPnP, ...) to be easily added. Further, as outlined in Figure 6, Ubigate enables IP and non-IP applications to discover each other transparently.

[11]Knopflerfish: www.knopflerfish.org
[12]UbiGate: http://uuu.enseirb.fr/~bissyand/UbiGate/

## 4. RELATED WORK

Over the years many research groups and industries have focused on designing solutions to cope with the heterogeneity of service discovery protocols. Typically, providing SDP interoperability consists of enabling applications to switch their current SDPs on the fly according to their networked environment. This is made possible through the use of an intermediate representation of SDPs paradigms (i.e an intermediary protocol) in order to abstract incompatibilities among SDPs to exclusively consider their similarities [1, 3, 5, 7, 14, 18]. So far, two approaches have emerged depending on how applications are either bound or unbound to this intermediary protocol [4]. In an explicit approach, applications need to be explicitly designed to use a specific discovery API that translates, if required, the intermediary protocol to the SDP currently used according to the networked environment [18]. In a transparent approach applications are unaware of the translation process [3]. While the latter offers seamless interoperability to legacy applications, the former enables the extending existing SDPs with advanced features. Our approach combines the strengths of the current approaches to provide a new SDP paradigm based on the concept of presence in order to provide a realtime discovery as required by pervasive environments. Following the example of the Amigo service architecture [6], UbiGate allows the integration of heterogeneous technologies by establishing interoperability of SDPs through different mechanisms involving the deployment of an intermediate node.

## 5. CONCLUSION

The multiplicity of discovery protocols, the capacities of pervasive computers, and the expectations of users, have contributed to the rise of new challenges for service discovery. To meet these challenges, we have proposed a new approach for service discovery involving the use of a gateway and relying on a widespread protocol, SIP.

Following our approach, we have designed the UbiGate gateway in charge of transforming discovery information into presence information. UbiGate can then process subscriptions from SIP compliant devices and manage discovery tasks using adequate native service discovery protocols. Centered around the notion of *presence*, UbiGate enables a realtime availability of service information while hiding the heterogeneity of underlying protocols.

The suitability of the approach for pervasive computing has been illustrated through two case studies involving the use of the Bluetooth service discovery protocol and a protocol for RFID. We have then developed dedicated modules for UbiGate, thus demonstrating the ease for adding support of new protocols.

After tackling the issues in service discovery in this paper, future work will involve service delivery using the SIP protocol. Taking advantage of the peer-to-peer model of SIP and the portability aspect of UbiGate, service delivery can be dealt with efficiently.

## Acknowledgements

The authors would like to extend their gratitude to ENSEIRB graduate students Nabila Ayadi, Jean Collas and Aurélie Goubault de Brugière who contributed major parts of the implementation of UbiGate's prototype. We further thank Dr. Julia Lawall and the anonymous reviewers for their useful comments.

## 6. REFERENCES

[1] C. Becker, G. Schiele, H. Gubbels, and K. Rothermel. Base: A microbroker-based middleware for pervasive computing. In *Proc. of PERCOM'03*, pages 443–451. IEEE Computer Society, 2003.

[2] Bluetooth Consortium. Specification of the Bluetooth System Core version 1.0b: Part E, Service Discovery Protocol (SDP). Technical report, November 1999.

[3] Y.-D. Bromberg and V. Issarny. INDISS: Interoperable discovery system for networked services. In *Proceedings of the 6th International Middleware Conference*, pages 164–183, Grenoble, France, Nov. 2005.

[4] Y.-D. Bromberg, V. Issarny, and P.-G. Raverdy. Interoperability of service discovery protocols: Transparent versus explicit approaches. In *Proceedings of the 15th IST Mobile and Wireless Communications Summit*, Myconos, Greece, 2006.

[5] P. Costa, G. Coulson, C. Mascolo, L. Mottola, G. P. Picco, and S. Zachariadis. A Reconfigurable Component-based Middleware for networked Embedded Systems. *Journal of Wireless Information Networks*, 14(2):149–162, June 2007.

[6] N. Georgantas, S. B. Mokhtar, Y.-D. Bromberg, V. Issarny, J. Kalaoja, J. Kantarovitch, A. Gerodolle, and R. Mevissen. The amigo service architecture for the open networked home environment. In *WICSA '05: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, pages 295–296, Washington, DC, USA, 2005. IEEE Computer Society.

[7] P. Grace, G. S. Blair, and S. Samuel. A reflective framework for discovery and interaction in heterogeneous mobile environments. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):2–14, 2005.

[8] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. RFC 2608, Internet Engineering Task Force, June 1999.

[9] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - a service discovery and delivery protocol for ad-hoc networks, 2003.

[10] W. Jiang, J. Lennox, H. Schulzrinne, and K. Singh. Towards junking the PBX: Deploying IP telephony, 2001.

[11] T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE Pervasive Computing*, 1(1):70–81, January 2002.

[12] J.-C. Liang, J.-C. Chen, and T. Zhang. Mobile service discovery protocol (msdp) for mobile ad-hoc networks. In *Autonomous Decentralized Systems, 2007. ISADS '07. Eighth International Symposium on*, pages 352–362, 2007.

[13] M. Lonnfors, J. Costa-Requena, E. Leppanen, and H. Khartabil. Session initiation protocol (SIP) extension for partial notification of presence information. RFC 5263, Internet Engineering Task Force, Sept. 2008.

[14] E. Loureiro, F. Bublitz, N. Barbosa, A. Perkusich, H. O. de Almeida, and G. Ferreira. A flexible middleware for service provision over heterogeneous pervasive networks. In *Proc. of WoWMoM'06*, pages 609–614. IEEE Computer Society, 2006.

[15] R. S. Marin-Perianu, J. Scholten, P. J. M. Havinga, and P. H. Hartel. Cluster-based service discovery for heterogeneous wireless sensor networks. *Int. J. Parallel Emerg. Distrib. Syst.*, 23(4):325–346, 2008.

[16] A. N. Mian, R. Baldoni, and R. Beraldi. A survey of service discovery protocols in multihop mobile ad hoc networks. *IEEE Pervasive Computing*, 8(1):66–74, 2009.

[17] J. Oikarinen and D. Reed. Internet Relay Chat Protocol. RFC 1459, Internet Engineering Task Force, May 1993.

[18] P.-G. Raverdy, V. Issarny, R. Chibout, and A. de La Chapelle. A multi-protocol approach to service discovery and access in pervasive environments. In *The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services*, pages 1–9, San Jose, CA, USA, July 2006.

[19] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265, Internet Engineering Task Force, June 2002.

[20] J. Rosenberg. Presence: The best thing that ever happened to voice. *Comput. Teleph.*, 8(11):67–68, 2000.

[21] J. Rosenberg. A presence event package for the session initiation protocol (SIP). RFC 3856, Internet Engineering Task Force, Aug. 2004.

[22] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920, Internet Engineering Task Force, Oct. 2004.

[23] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson. Presence information data format (PIDF). RFC 3863, Internet Engineering Task Force, Aug. 2004.

[24] Sun Microsystems. Jini Architecture Specification version 2.0. Technical report, June 2003.

[25] The Salutation Consortium. Salutation Architecture Specification version 2.0c. Technical report, June 1999.

[26] UPnP Forum. UPnP Device Architecture version 1.0. Technical report, June 2000.

[27] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.