Architecture for the Interoperability among Ubiquitous Components

Anabel Pineda[1], Rolando Menchaca[2] and Giovanni Guzmán[3]

[1] – Computing and Systems Department
Matamoros Institute of Technology, Matamoros, Mexico
apineda@itmatamoros.edu.mx
[2,3] - Centre for Computer Research, National Polytechnical Institute, Mexico City, Mexico
{rmen, jguzmanl}@cic.ipn.mx

Abstract. Implementing spontaneous interoperability among nomadic components is currently one of the most relevant open problems in ubiquitous computing. There are three main issues related with achieving spontaneous interoperability, namely, initialization, discovery and interaction among components. In this work we present a novel architecture which is based on ontologies and peer-to-peer algorithms that solve these three aspects in an integrated way. The architecture is composed of a set of services implemented using JXTA and an ontology-based inference engine. Using our infrastructure, clients (mobile or static) are able to perform semantic and geographic-aware queries to the ubiquitous environment and dynamically find instances of a desired service. We also present a couple of applications that were developed using our proposed architecture. These applications allow us to show the potential of our infrastructure to implement two classic ubiquitous computing scenarios, both using mobile devices and heterogeneous wireless networks.

1 Introduction

Ubiquitous Computing, also known as Ubicom, is a relatively new paradigm first defined by Mark Weiser that describes a class of computing systems whose main goal is to improve the quality of life of human users by means of a seamless integration of hardware and software components into people's everyday activities. The services provided by these systems are ubiquitous in the sense that they can be provided by almost any everyday object, moreover, ubiquitous hardware and software components tend to disappear into the environment and the user only perceives their advantages while the complexity is hidden as much as possible [1]. The enormous potential of this new paradigm has attracted the attention of an increasing community of researchers who are working to materialize the seminal ideas of Mark Weiser [1]. There are many research areas directly related with the ubiquitous computing, for instance, software engineering, human-computer interaction, semantic computing, computer networks, natural language and others. In this work we focus on defining new software architecture that foster the constructions of ubiquitous systems by providing a set of fundamental services that have been identified as useful and desirable for almost any ubiquitous system.

A. Gelbukh, S. Suárez, H. Calvo (Eds.)
Advances in Computer Science and Engineering
Research in Copmuting Science 29, 2007, pp. 211-224

Received 02/07/07
Accepted 21/10/07
Final version 25/10/07

To be more precise, a ubiquitous system is in essence a saturated environment with computing and communication components that are integrated with the everyday tasks of their human users [2]. The Ubicom community has identified two fundamental characteristics of a ubiquitous system: they are physically integrated into everyday objects and their components have to be capable of interacting with other ubiquitous components without need of manual re-configurations. Here, we go one step further of the second characteristic by designing and implementing a novel software architecture that solves the three main issues related with achieving spontaneous interoperability. To solve these three problems we developed a set of protocols that enable heterogeneous components to dynamically recognize other and interact among them. The computer networks that provide communication support to the ubiquitous systems have to be heterogeneous and allow the interoperability of wired, infrastructure-based wireless and ad hoc networks. Moreover, these networks have to be highly dynamic and support continuous arrival and departure of new components without need of explicit re-configurations [3]. To cope with these requirements we employ a peer-to-peer (P2P) software architecture. P2P systems are highly dynamic, composed of a set of independent software elements with equivalent functionality and they do not require centralized management. These characteristics endow to P2P systems with properties such as scalability, flexibility [4], fault tolerance and a fairly simple management [5] that can be quite useful when designing an infrastructure for the development of ubiquitous systems in general and in particular for the infrastructure proposed in this work.

Nowadays, there are several P2P systems available, for example Gnutella and Napster. However, most of these systems are designed for specific application, such as file sharing. Therefore, our software architecture uses the P2P infrastructure of JXTA in order to enable services among peers which may be hidden behind NAT, firewalls, to dynamically join and leave the P2P network, with the possibility of changing their locations. This is the general purpose for which the infrastructure JXTA was designed, to provide interoperability, platform independence and ubiquity. Inside the existent P2P infrastructures, JXTA is the one that fulfills most of the characteristics that our architecture needs.

Although JXTA provides the ideal characteristics to our architecture, it does not currently provide an adequate solution to the discovery service problem because is not enough to provide a basic advertisement-search mechanism. JXTA needs a flexible discovery service system to enable to locate all available services conforming to a particular functionality.

The rest of document is organized as follows: Section 2 presents a short introduction and the main characteristics of current discovery service protocols; in Section 3 we respond to the question why JXTA?; Section 4 describes the need to extend JXTA discovery service, Section 5 describes a novel architecture solution (ArCU) –integrating a set of services using JXTA with ontology-based inference engine– and the implementation of USE (Ubiquitous Services Environment). Finally Section 7 presents the conclusions of present work.

2 Discovery Service Protocols

There are recent solutions related with the integration of peer-to-peer (P2P) infrastructure and ontologies to discovery services [19][20]. These proposals are focused in offer precision in the discovery of services with wished functionalities. Currently, there are many protocols that provide a solution for discovery service, one of the most important requirements to reach spontaneous interoperability. Next, we present a short introduction and the main characteristics of them. Especially, we will focus in the discovery service because it is the most relevant for our proposal.

2.1 Universal Plug and Play (UPnP)

Universal Plug and Play is a technology developed in the UPnP Forum [6] for automatically configuring devices, discovering services and providing P2P data transfer over an IP network. UPnP technology is built upon IP, TCP, UDP, HTTP and XML, among others. The UPnP discovery protocol is based on the Simple Service Discovery Protocol (SSDP). The discovery process of UPnP based in SSDP is as follow: given an IP address, when a device is added to the network, the UPnP discovery protocol allows the device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. In both cases, the fundamental exchange is a discovery message containing a short essential description about the device or its services, i.e., its type, unique identifier, and an URL to obtain more detailed information. The principal limitation of SSDP is that it does not support the search for multiple types in the same request and attribute-based search [7].

2.2 Service Location Protocol (SLP)

Originally, the Service Location Protocol (SLP) [8] was proposed as an Internet Engineering Task Force (IETF) standard track protocol, to provides a framework to allow networking applications to discover the existence, location, and configuration of networked resources in networked resources, such as devices and services. SLP eliminates the need for a user to know the name of a network host that supports a service. Rather, the user supplies the service name and a set of attributes, which describes the resource.

The resources are modeled as clients that need to find servers attached to the enterprise network at a possibly distant location. For cases where there are many different clients and/or available resources, the protocol is adapted to make use of nearby Directory Agents that offer a centralized repository for advertised services.The basic operation in SLP is that a client attempts to discover the location for a resource. In small installations, each resource is configured to respond individually to each client. In larger installations, resource will register their services with one or more

directory agents and clients contact the directory agent to fulfill request for resource location information. This is intended to be similar to URL specifications and make user of URL technology. The principal limitations are the ability to reflect SLP's orientation toward enterprise service discovery and heavyweight directories [8].

2.3 Salutation

The Salutation architecture was developed by the Salutarion Consortium, to solve the problems of discovery service and utilization among a broad set of appliances and equipment in a wide-area or mobile environment [9]. The Salutation architecture is composed of two elements: Salutation Lookup Manager (SLM) and Transport Manager. The SLM functions as a service broker for services in the network. The SLM can classify the services based on their meaningful functionality, called Functional Units (FU). The services are discovery by SLM by means of a comparison of the requerid service types with the types stored in the SLM directory. The discovery service process can be performed across multiple Salutation Lookup Managers, where one SLM represent its client while communicating with another SML to discover services [10].

2.4 Bluetooth

Bluetooth is a radio standard and communications protocol designed for low power consumption, with short-range radio frecuency [11]. Bluetooth defines its own protocol stack, including a service-discovery protocol, SDP. This protocol is based on unique identification numbers (UUIDs), with several predefined services such as phones, printers, modems, and headsets. The Bluetooth specifications are developed and licensed by the Bluetooth Special Interest Group.

Bluetooth communication is P2P, so it does not assume a fixed network infrastructure. Thus, discoverability is based on actual physical proximity rather than closeness in the IP routing infrastructure. In that sense, in comparison to the rest of the services discoveries, it simplifies the discovery and setup services. A Bluetooth device advertise all its services, making them more accessible, without the need to worry about network addresses, permissions and all other considerations related with typical networks.

2.5 Jini

Jini is a distributed service-oriented architecture developed by Sun Microsystems [12]. Jini is a simple insfrastructure for providing services in a network, and for creating spontaneous interactions between theses services. Services can join or leave the network in a robust fashion, and clients can rely upon the availability of visible services, or at least upon clear failure conditions [13].The service advertisement takes the form of interface descriptions. This simple form of the advertisement mechanism can be easily employed to provide high-level abstraction both for software and

hardware entities in the network. Jini discovery insfrastructure provides a good base foundation for developing a system with components distributed in the network that need to discover each other. However, the Jini discovery process is tightly bound with the simple interface description advertisement. This leads in a loss of expressive power in the component description. For example, Jini discovery and lookup protocols are suficient for service clients to find a print service. However, they are not suficient for clients to find a print service through their geographical localization or particular functionality such as color laser printer service. This is a limitant because can create problems in a mobile environment. Furthermore, the simplicity of the Jini architecture also leads to the cross-domain service interoperablity problem.

2.6 JXTA

JXTA (short for "juxtapose") is a set of open protocolos that facilities Peer-to-Peer communication. This technology allows connecting a wide variety of devices that they can be anything with an electronic hearbeat [14]. JXTA is based upon a set of open XML protocols, this way allow exchange messages and collaborate independenlty of programming language, platform or network transport.

In JXTA the peers are organized in *peergroups* to represent types of services, location, etc. All network resources in JXTA such as peers, peergroups, pipes and services are represented by advertisements that are XML documents that announce the existence and some properties of these resources. Every Advertisement in JXTA has a string Name Field. For the search, JXTA Advertisement usually uses their name to indicate the type of service the peers provide. Advertisements provide a uniform way to publish and discover network resources and they have a lifetime to specify the lifetime of its associate resource.

The general purpose of JXTA is providing interoperability across verying P2P systems and communities, platform independence to support diverse languages, systems and networks, and ubiquity, in which every device has a digital heartbeat.

3 Why JXTA?

JXTA provides three aspects that the rest of the services previously mentioned individually they do not provide.

- The JXTA infrastructure adopt P2P systems characteristics: highly dynamics, set of independent software elements with equivalent functionality and decentralized management. These characteristics endow to JXTA with properties such as scalability, flexibility, fault tolerance and a fairly simple management.
- JXTA infrastructure provide interoperability, platform independence and ubiquity, and
- The opportunities for extending JXTA are manifold because its support arbitrary XML so it can integrate emerging standards (such as the Ontology Web Language [OWL] for description) as a relevant approach for refining searches.

These characteristics are necessary for the development of ubiquitous systems and for that reason we decide to use JXTA infrastructure.

4 Extending JXTA Discovery Service

In the introduction, we have mentioned that although JXTA provides the ideal characteristics to our architecture, it does not currently provide an adequate solution to the discovery service problem, because is not sufficient to provide a basic advertisement-search mechanism. JXTA needs a flexible discovery service system to enable to locate all available services conforming to a particular functionality. For that reason, we intend to integrate peer-to-peer algorithms and ontologies as alternative to approach to refined search.

Ontology in computer science is usually defined as an explicit specification of a conceptualization of a domain [15]. But, why is it important to use ontologies? We can use ontologies to describe a shared conceptualization of domain of services, devices and other concepts that could influence the discovery service process, such as different kinds of context [16], for example, in a particular case, geographical localization of a printer.

The following section will discuss discovery service enabling ontologies, as our architecture integrates such ontologies with JXTA, and which is the procedure to discovery service.

5 Architecture

In this section we present the design of ubiquitous computing architecture (ArCU) as well as a description of each of its components. We emphasize on the P2P nature of our architecture that is achieved by using JXTA [14] as our communication infrastructure. We also discuss how our ontology-based descriptions and inferences provide extra benefits to ArCU. As one of the most evident benefits of using JXTA is the fact that the JXTA's protocols implement a virtual overlay network that hides all the details about the particular instantiation of the physical communication network. In this way, our components or peers are able to interact with other components not regarding the type of communication network (wired, behind a firewall, infrastructure-based wireless) they are using [14]. As it is shown in Fig. 1, ArCU is composed by the following elements: One or more Clients (ArCU-MC) that can be mobile or static, one or more Ubiquitous Services (ArCU-US) and at least one Basic Inference Service (ArCU-BIS).

5.1.1. Mobile Clients (ArCU-MCs)

Using Clients or Mobile Clients, users are able to find and interact with the Ubiquitous Services provided by the environment. The ubiquitous services are discovered by means of queries that are issued to the Basic Inference Service (ArCU-

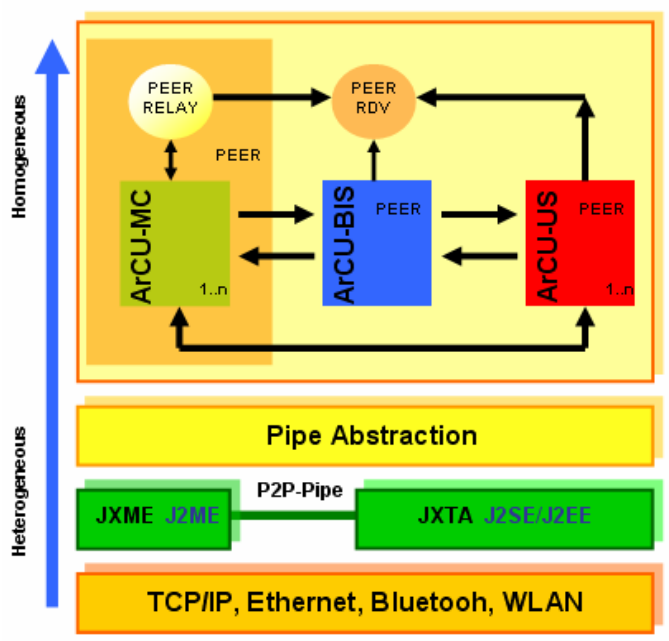

Fig. 1 ArCU Architecture.

BIS). Every Ubiquitous Service publishes a set of device-independent user interfaces (codified in XML) that can be analyzed and displayed by the ArCU-MCs. Every ArCU-MC displays the interfaces in the way that best fits its hardware capabilities. The flexibility provided by the device-independent definition is very important because a very wide range of devices may act as ArCU-MCs and some of them (i.e. cellular phones) may have restricted displaying capabilities.

On the other hand, small devices such as cellular phones or PDAs commonly have restrictions on their storage and computing capabilities and they may experience continuous disconnections due to mobility or lack of battery power. To cope with all these restrictions imposed by the nature of the hardware devices employed in ubiquitous systems we use the JXTA version for mobile devices, namely, JXME [14].

5.1.2. Ubiquitous Services (ArCU-US)

The Ubiquitous Services (ArCU-USs) are services viewed as functional components. That functionality is typically implemented as software components; some services specifically support, or are offered by devices (hardware components) where they characteristics and capabilities may play a role in the description and behavior of the services it offers. Examples of Ubiquitous Services: a projector, a printer, a public display, databases, software presentation and so on. The ArCU-USs employs JXTA announcements to publish their services in the environment. These announcements include semantic descriptions codified in OWL [17] about the services they offer.

5.1.3. Basic Inference Service (ArCU-BIS)

The Basic Inference Service (ArCU-BIS) is the most important component of ArCU. This component is subdivided in three elements: Communications Administration Service, Ontology Administrator and Search Engine.

The Communication Administration Service is in charge of the establishment and maintenance of the communication channels. The Communication Administration Service employs JXTA pipes as its communication abstraction. The JXTA pipes provide a virtual pipe that is able to communicate peers that do not have a direct physical link, or in other words, that reside in different types of networks or behind different firewalls. The endpoints of the pipes are dynamically linked (at run-time) to the endpoints of the peers and hence, peers are able to move from one domain to another in a transparent way. Moreover, using the pipe abstraction, our services can transparently recuperate from failures in any of the physical endpoints. This way, the Communication Administration Service is able to hide the fact that some devices may be mobile and change from one network to another.

The Ontology Administrator has the responsibility of creating and managing a dynamic ontology. That means? For example, a scenario is illustrated in the Fig. 2(a) with three available services. If 15 minutes later a service leave the environment such as show the Fig. 2(b).The Ontology Administrator has to refresh in no more of 5 minutes the ontology state as can be seen in the Fig. 2(c), all this in order to maintain the ontology as small as possible to help reduce the time and space complexity of the semantic searches. The Fig. 3 illustrates a section of a device ontology proposed by [18]. To this work, we adopt the use of this device ontology because helps to describe devices and their services in a rich and expressive way to facility a semantic discovery of services.

Finally, the Search Engine is in charge of performing the semantic and geographic-context-aware searches that are issued by the clients of the ubiquitous environment. As a result of these searches, Search Engine returns references to the services that meet the criteria specified by the clients. These references are further used by the clients to access the services.

5.1.4 ArCU Communication Infrastructure

ArCU builds on the basic advertisement-search JXTA mechanism. We purpose three important additons that peers providing services using ArCU must implement:

1. Pointer to XML document specify to graphic description of interfaces.
2. Pointer to OWL file that describe the service.
3. A XML Service Advertisement with need information to publish and invoke the documents previously mentioned. As it is shown in Fig. 4.

The OWL Web Ontology Language [17] is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilities the interpretability of content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

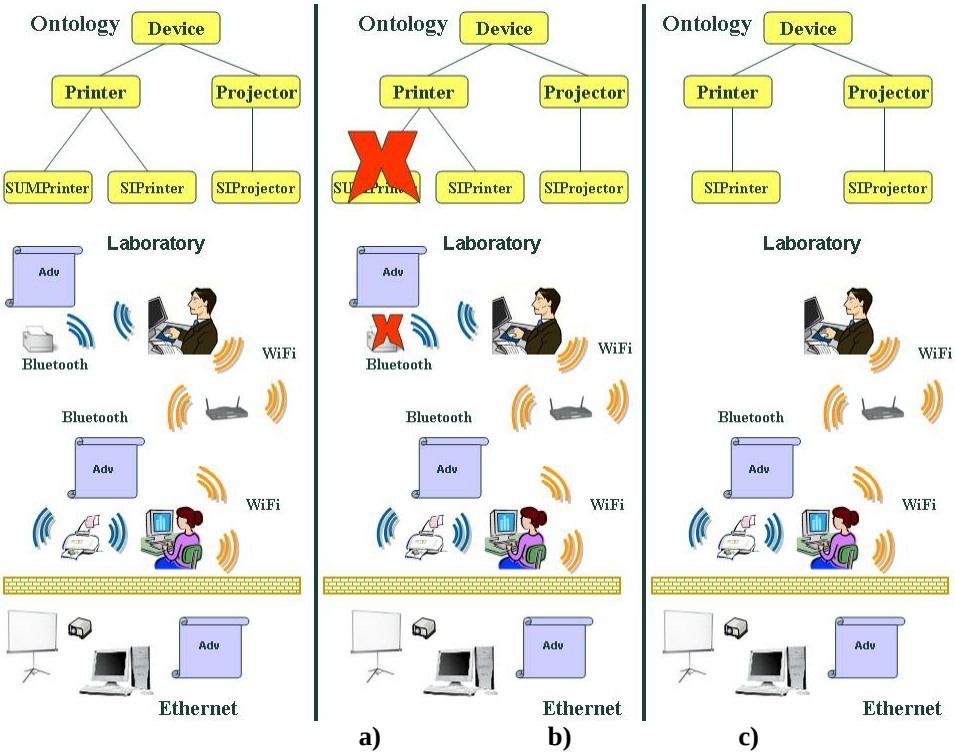

Fig. 2 Dynamic Ontology Scenario.

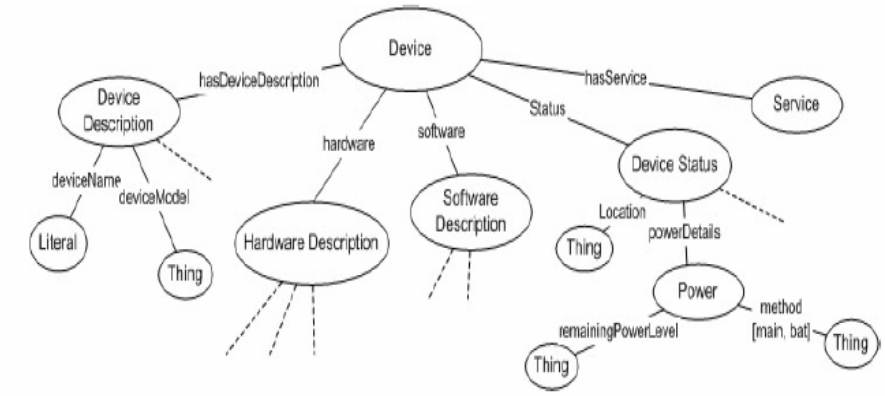

Fig. 3 Device Ontology.

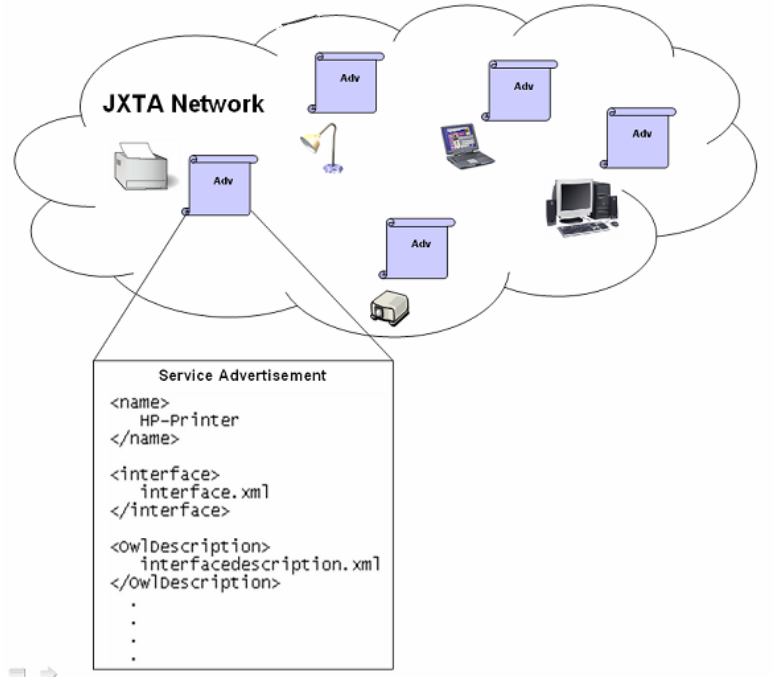

Fig. 4 A service advertisement in ArCU.

In the case of our architecture that ArCU-BIS peer wish to matches a set of searching criteria a service need retrieve OWL file that describe the service. This remote communication is kept as previously mentioned by usign JXTA pipes as mechanism for exchanging messages between services. These messages in JXTA are also based in XML format, standarized by W3C same as OWL [17].

JXTA and OWL uses XML, the integration of both was straightforward since it doesn't commit to some standard of communication.

5.1.5 ArCU Discovery Service Mechanism

To explain the ArCU Discovery Service Mechanism is necessary describes the interactions among ArCU components for to publish and discover services. As we previously mentioned, the communication among the components of ArCU is carried out using standard JXTA protocols. A typical communication sequence is shown in Fig. 5.

1. The ArCU-BIS publishes its announcement in the distributed hash table implemented by a special-type peer known as rendezvous peer.
2. The ArCU-US peer also publishes its announcement in the distributed hash table implemented by a rendezvous peer (it may or may not be the same peer as in the step 1).

3. In order to make invocations over the services implemented by the ArCU-BIS, an ArCU-US has to find the announcement of an ArCU-BIS peer. In JXTA the management of the advertisements is carried out by the rendezvous peers who implement the distributed hash table.
4. Upon receiving a query, the rendezvous peer looks for an index of the announcement in the distributed hash table. If it finds the index, the rendezvous peer relies the query to the peer that published the announcement (in this case the ArCU-BIS). When the ArCU-BIS receives the query, it replies with its advertisement that is further received by the ArCU-US.
5. With the advertisement, the ArCU-US acquires the capability of invoking the services provided by the ArCU-BIS. In this way, the ArCU-US is able to record it services in the database of the ArCU-BIS.
6. The ArCU-MCs are implemented by a special type of peer that is called edge peer. These peers need to be coupled with another special type of peer that is called relay peer. So, in order to access the services of the ubiquitous environment, every ArCU-MC has to establish communication with a relay peer.
7. The functionality of a pair edge-relay peer is equivalent to the one of a regular JXTA peer. When the ArCU-MC is connected to a relay peer, it sends a query to a rendezvous peer looking for the announcement of an ArCU-BIS peer.
8. The rendezvous peer looks in the distributed hash table for the index of the announcement. If it finds the index, the rendezvous peer relies the query to the peer that published the announcement (in this case the ArCU-BIS).
9. When the ArCU-BIS receives the query, it replies with its advertisement that is further received by the ArCU-MC.
10. The ArCU-MC can now issue requests to the ArCU-BIS looking for a service that matches a set of searching criteria.
11. When the ArCU-BIS finds the service it replies with the service's advertisement.
12. Now, the ArCU-MC is able to send a request to the ArCU-US asking for its device-independent graphic interface.
13. Finally, the user can employ this interface to interact with the ubiquitous service.
14. It is important to insist that the graphic interface is described in XML document that is analyzed by the client device and then displayed according to its hardware capabilities.

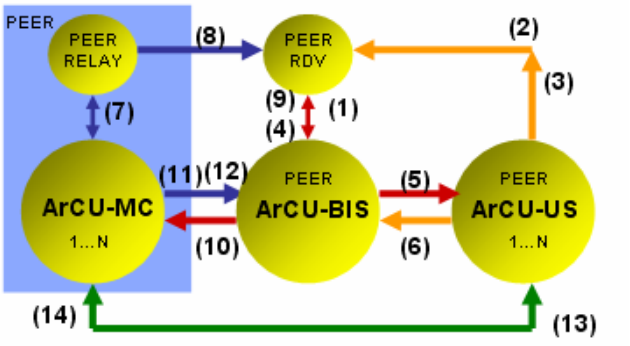

Fig. 5 Interaction among components in our ubiquitous computing platform.

5.1.6 Prototype

In order to materialize the architecture proposed we have been developed USE. USE is a Ubiquitous Services Environment. Into USE we implemented two classics ubiquitous computing scenarios, both using mobile devices and heterogeneous networks. In this section, only we illustrate one scenario. Fig. 6 show images of the graphics interfaces used in the print service scenario described below.

Fig. 6 Print service graphics interfaces in USE.

Scenario: Rolando is a researcher in the area of ubiquitous computing and he is currently attending a meeting in the main conference room of an important research center. Later, he will be presenting some of his latest results in the same room and he needs to make some printouts to distribute them among his colleagues. To do so, Rolando uses one of the ubiquitous services provided by the conference room. Despite the fact that this is the first time that Rolando visits this particular research center, he is able to use his PDA to look for a public printer that is also located in the main conference room. As a result, the PDA gets a list of printing services that meet the criteria specified by Rolando. Once Roland selects the desired service, the PDA gets the graphic interface of the selected service. Finally, Rolando uses the interface to select the file to be printed and to send the job to the printer.

We have experimentally confirmed the functionality with real devices. Particularly we use a Sony Clié PEG-UX50 Handheld with PALM OS 5.2 with WiFi and Bluetooth technologies. With this experiment, we were able to test the potential of our infrastructure.

6 Conclusions

In this paper we presented a software architecture that allows clients (fixed or mobile) to perform semantic and geographic-context-aware searches over the services and resources contained in a ubiquitous environment. Our architecture employs a distributed hash table that works in conjunction with a set of inferences processes that are based on ontologies. In our proposal, the ubiquitous environment acts a dynamic repository that contains the ontologies that describe the services available in the environment in a given point in time. To make or architecture scalable, the algorithms employed to implement the repository were designed with the objective of reducing as much as possible the time and space complexity involved in the semantic searches. The communication among components is carried out using the standard JXTA protocols which is based on the interchange of XML documents. In the same way, the user interfaces are specified using XML documents, and each client is free to display those interfaces in the way that best matches its hardware and software capabilities. These last two design choices allow us to implement heterogeneous systems that can be composed of a wide variety of (off the shelf) hardware and software platforms.

Acknowledgments

The authors of this paper wish to thank the CIC, SIP, CONACYT, IPN and ITM for their support. Additionally, the authors wish to thank the reviewers for their pertinent comments.

References

1. Weiser, M., “The Computer for the 21st Century”, Scientific American, Vol. 265, No. 3, September 2001, pp. 94-104.
2. Satyanarayanan, M., “A Catalyst for Mobile and Ubiquitous Computing”, IEEE Pervasive Computing, Vol. 1, No. 1, January - March 2002.
3. Menchaca, R. and Favela, J., “Arquitecturas P2P para el Desarrollo de Sistemas Móviles y Ubicuos”, Septiembre 2003.
4. Balakrishnan, H., Kaashoek, F., Karger, D., Morris, R. and Stoica, I., “Looking Up Data in P2P Systems”, Comunications of the ACM, Vol. 46, No. 2, February 2003, pp. 43-48.
5. Kubiatowicz, J. M., “Extracting Guarantees from Chaos”, Comunications of ACM, Vol. 46, No. 2, February 2003, pp. 33-38.
6. Universal Plug and Play. www.upnp.org.
7. Guttman, E., “Service Location Protocol: Automatic Discovery of IP Network Services”, IEEE Internet Computing, Vol. 3, No. 4, August 1999, pp. 71-80.
8. Edwards, W. K., “Discovery Systems in Ubiquitous Computing”, IEEE Pervasive Computing, Vol. 5. No. 2, April 2006, pp. 70-77.
9. Salutarion Arquitecture Specification, v. 2.1, Salutation Consortium, 1999.
10. Chakraborty, D., Perich, F., Avancha, S. and Joshi, A., “DReggie: Semantic Service Discovery for M-Commerce Applications”, in Workshop on Reliable and Secure Applications in Mobile Environment, Symposium on Reliable Distributed Systems, October 2001.
11. Specification of the Bluetooth System, v.1.1 core, Bluetooth Consortium, 2001; http://bluetooth.com.

12. Waldo, J., The Jini Specifications, edited by Ken Arnold. Addison-Wesley Professional, Second edition. December 15, 2000.
13. Arnold, K., Wollrath, A., O'Sullivan, A., Scheifler, R. and Waldo, J., The Jini Specifications. Addison-Wesley, Reading, MA, USA, 1999.
14. Proyect JXTA 2.0 Super-Peer Virtual Network. http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf .
15. Gruber T. "Toward Principles forthe Design of Ontologies Used for Knowledge Sharing". Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University, CA, 1993.
16. Schmidt, A., "Ubiquitous Computing—Computing in Context", Submitted for the degree of Doctor of Philosophy, Lancaster University, England, U.K. November 2002.
17. http://www.comp.lancs.ac.uk/~albrecht/pubs/.
18. OWL Web Ontology Language: Overview. http://www.w3.org/TR/owl-features/
19. Bandara, A., Payne, T., De Roure, D. and Clemo, G., "An Ontological Framework for Semantic Description of Devices", the 3rd International Semantic Web Conference, Japan, November 2004.
20. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S. and Miller, J.,"METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services", Journal of Information Technology and Management, under review.
21. Paolucci, M., Sycara, K., Nishimura, T. and Srinivasan, N., "Using DAML-S for P2P Discovery", in Proceedings of the International Conference on Web Services, 2003.