ISSN 0103-9741

Monografias em Ciência da Computação
n° 22/07

Discovering Services with Restricted Location Scope in Ubiquitous Environments

José Viterbo Filho
Markus Endler
Vagner José do Sacramento Rodrigues

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 22/07 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Agosto, 2007

Discovering Services with Restricted Location Scope in Ubiquitous Environments

José Viterbo Filho, Markus Endler and Vagner José do Sacramento Rodrigues[1]

[1] Instituto de Informática - Universidade Federal de Goiás

viterbo@inf.puc-rio.br, endler@inf.puc-rio.br, vagner@inf.ufg.br

Abstract. In ubiquitous computing systems, the mobility of users and their devices results in recurring disconnections and reconnections with different networks, and the corresponding dynamic change of the network and domain-specific resources and services accessible from the user's device. On the other hand, some services are available to be used only by users that are located in a well defined region. In this highly dynamic and heterogeneous scenario, applications must be capable of discovering the appropriate instances of the required services in each visited network or region. In order to support such spontaneous interaction, we propose a discovery service based on the notion of a (geographic) location scope. This discovery service is one of the core services of the MoCA architecture, a middleware that supports the development and deployment of location-aware ubiquitous applications.

Keywords: Context-aware Computing, Ubiquitous Systems, Service Discovery

Resumo. Em sistemas ubíquos, a mobilidade dos usuários e seus dispositivos resulta na desconexão e reconexão recorrente a diferentes redes, e a correspondente alteração dinâmica dos recursos específicos de cada domínio e dos serviços acessíveis a partir do dispositivo do usário. Por outro lado, alguns serviços estão disponíveis para uso somente dos usuários que estão situados em regiões bem definidas. Nesse cenário altamente dinâmico e heterogêneo, as aplicações devem ser capazes de descobrir as instâncias apropriadas dos serviços necessários em cada rede ou região visitados. A fim de dar apoio a essa interação espontânea, propomos um serviço de descoberta baseado na noção de escopo de localização (geográfica). Esse serviço de descoberta é um dos serviços principais da arquitetura MoCA, um sistema de middleware que provê apoio ao desenvolvimento de aplicações ubíquas cientes de localização.

Palavras-chave: Computação Sensível a Contexto, Sistemas Ubíquos, Descoberta de Serviços

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

1 Introduction

Ubiquitous computing environments involve the interaction of diverse computational devices aiming to provide functionalities for the users in a transparent, continuous and seamless way [1]. For this purpose, ubiquitous computing environments encompass different kinds of sensors and mobile devices (e.g. PDAs, notebooks, smartphones), interconnected through a combination of several wireless network technologies [2]. A software infrastructure to support the execution of ubiquitous applications has to be flexible and reactive to deal with the unpredictable, heterogeneous and dynamic nature of the computing environments encountered due to the user's mobility [3].

Some ubiquitous applications have as primary purpose to execute a given task using the resources or services that are available in the user's vicinity or in the currently visited network domain [4], as in a mobile grid application, for instance. Moreover, some services are available to be provided only within a well defined region. For example, services provided by active spaces systems such as smart houses, offices or classrooms, are ment to be used only by the users that are inside a given room. Such services have their location scope restricted to that limited region and should not be advertised for clients outside their scope area. On the other hand, some services will be appropriate only to those clients that have interest in a specific coverage area, like, for instance, a service that transmits images of a camera or data from sensors located in a given room. This service needs not to be advertised for that clients that are not interested in the coverage area of the service. For achieving these features, the underlying infrastructure for ubiquitous computing needs to rely not only on a location service capable of dynamically locating servers and clients but also on an appropriate service discovery mechanism capable of matching the scope of the services with the area of interest of the clients.

MoCA architecture [5] has been designed to support the development and deployment of context-aware applications involving mobile devices, and its core services are also adequate for the development and deployment of ubiquitous applications. MoCA's *Location Inference Service* (LIS) allows to infere the location (and follow the movements) of 802.11-enabled mobile devices within an organization. Hence, MoCA based applications can be aware of the current location of such devices and be be notified whenever they change location. And in order to enable applications to discover (and interact with) services that are available in a given location, we designed, implemented and incorporated into MoCA a new discovery service named *Ubiquitous Discovery Service* (UDS) with the aforementioned location-based search and matching capabilities.

In the next section, we describe a typical scenario of ubiquitous computing where there are services with such restricted location scope. In Section 3, we discuss the definition of location scope for ubiquitous applications. In Section 4, we give an overview of the MoCA architecture. In Section 5 we present the proposed discovery service and in Section 6 we describe a case study. Section 7 dicusses some related work and, finally, Section 8 draws some conclusions.

2 Scenarios

In this section we present a ubiquitous computing scenario where services should be available only for application clients running on devices located in a given region. This scenario highlights the advantages of using a discovery service designed specifically for ubiquitous envirenmonts and mobile access. Of course, most of the required discovery functionality could also be implemented by the application, but this would increase the application's complexity, make it more dependent on the network and service configurations, and put an extra burden to the application programmer.

For the proposed scenario, which consists in an *active learning environment*, let's assume that an institution has some classrooms and seminar rooms equipped with several multimedia and portable devices, as well as several applications that enable a rich and interactive learning experience among instructors and students. As one of the applications, consider a *slide presentation service*, where a server executes on a (static) host, and the clients run on a tabletPC used by the instructor, allowing him to conduct a slide show in the classroom. As another application, consider a *slide sharing application*, where a server that executes on the tabletPC used by the teacher makes possible for him to share slides with the students that run a client application on their smartphones, but only with those located in that same classroom, while interaction by electronic means may be blocked for devices outside the classroom. In this scenario, a discovery service would be used to connect the respective clients and servers entering the room.

Considering that these classrooms spread over larger areas, like different floors and buildings, some basic services, such as a *printing service*, may be provided for a whole set of rooms grouped as sub-regions of a larger region. In such a scenario, a user who wants to print some document must have her application connected to the printing service available for the region where she is currently located. Again, a discovery service with location-awareness would be required to discover the corresponding printer server which has the desired properties (e.g. capabilities, exact physical location, configurations, etc.), but in this case the knowledge of a hierarchical structure defining compound regions and their nested sub-regions is also necessary.

3 Defining the Service Scope

The scenarios presented in Section 2 exemplify why in many cases the physical location of the user needs to be considered when searching for the most appropriate service for a user. Generally, in ubiquitous computing scenarios, the set of interacting applications and services changes dynamically and very frequently, due to the users' mobility and also because of the intermittent connectivity with wireless networks. Under these circumstances, many services — like the slide show server described in our scenario, for instance — are useful only for the users that are in the very same region that is served (or covered) by the service, while others, — like the printing service — comprise a whole set of regions. In addition to the region-specific services, of course, there may be several other services that are available to applications independently of the user's location or current network access point.

Based on these considerations, we propose two types of *scope of coverage* of a service. Services that are available for any application independently of where is located the device on which it is running are said to have *global area scope*, while services that are available only for application clients running on devices located in a given region are said to have *local area scope*. In order to be able to discover services available at a specific network domain and region, one requires not only a location service for identifying where a mobile device is located at any moment, but also the capability of registering services with their coverage scope, and of searching them according to this scope and the service requester's physical location. In other words, a ubiquitous application client running on a mobile device must be able to query a discovery service to get the addresses of the services of interest that are available specifically at a given region. Moreover, mobile devices may also be service providers. In this case, due to their mobility and to possible disconnections, the scope of these services will have to be updated very often. Hence, the discovery service must but also be prepared to cope with the dynamic nature of the service providers i.e. the fact that some services may suddenly become available or unavailable at a given location without prior announcement.

4 The MoCA architecture

The MoCA architecture provides services that support the collection, distribution and processing of context information probed directly from mobile devices or inferred through some other services. In addition, MoCA offers a set of APIs that support the implementation of context-aware applications using the MoCA context-provisioning services. In Figure 1 we see the main components of MoCA. A service called Monitor must run on each mobile device to collect raw context data, such as percentage of CPU usage, memory available, battery level, wireless connectivity, etc. This information is periodically sent to the Context Information Service (CIS), which is responsible for collecting, storing and processing the data received, making it available to the interested applications.

Besides context management and synchronous querying, CIS also supports context monitoring, allowing clients to register their interest in specific context states (involving one or several context variables) modeled as logical expressions, and to be asynchronously notified whenever the corresponding context-expression is satisfied (or ceases to be satisfied) [5]. For instance, with the expression {`"FreeMemory < 10KB" OR "APChange = True"`}, an application may subscribe to be notified when the free memory of the device is less than 10KB or when occurs a change of 802.11 network access point.

The Location Inference Service (LIS) is a special MoCA service which is responsible for inferring the location of a mobile device from the information about RF signal patterns collected from 802.11 access points [6]. The service keeps track of the location of any registered device of the system. The location information is provided in terms of symbolic regions, which correspond to well-defined physical areas in a given organization (i.e. rooms, halls, buildings) that may be of interest to the location-aware applications. The location information of each device can be queried by interested applications, which only need to know the device's MAC address.

MoCA allows not only the creation of logical regions of arbitrary size and shape but also their aggregation into a hierarchical structure defining regions and nested sub-regions. This functionality is implemented in an auxiliary service named Symbolic Region Manager (SRM), which provides an interface to create, manage and request information about hierarchies of symbolic regions. However, all composite symbolic regions must be build from a base set of atomic regions, which are determined by the LIS administrator.

In a way similar with CIS, LIS may also notify subscribed clients when a given device changes location or whenever any device enters or leaves a given symbolic region, and SRM may notify subscribers about changes in a hierarchy. This asynchronous communication

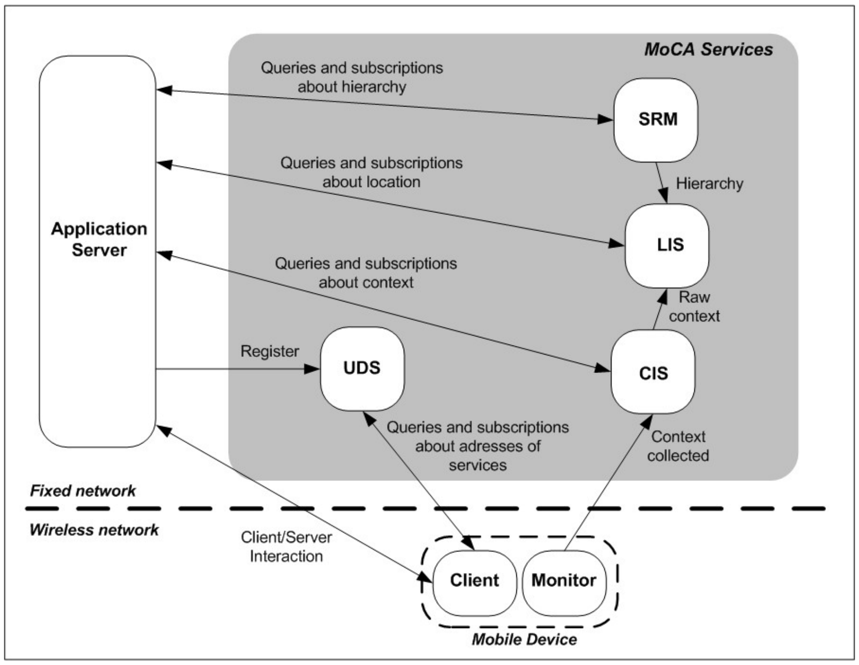

MoCA Services
Queries and subscriptions about hierarchy
SRM
Hierarchy
Queries and subscriptions about location
LIS
Application Server
Queries and subscriptions about context
Raw context
Register
UDS
CIS
Queries and subscriptions about adresses of services
Context collected
Fixed network
Wireless network
Client/Server Interaction
Client
Monitor
Mobile Device

Figure 1: Architecture of MoCA

functionality reduces the cost to build client applications, since the programers are relieved from managing the context information delivery.

5 MOCA's Discovery Service

We designed, implemented and incorporated into MoCA a new discovery service named *Ubiquitous Discovery Service* (UDS) for supporting the transparent and spontaneous discovery of services in ubiquitous environments populated with mobile service providers and requesters. In traditional discovery systems, the domain on which a client may discover a service is defined by the servers that may be reached by a multicast service advertisement — in systems based on network and IP address — or to those servers inside the transmission range of a client — in systems based on RF transmission. However, it may be difficult for the user to discover and select the most appropriate service among the large number of services available in a ubiquitous environment applying these traditional means to manage distributed services [7]. To be able to continuously and spontaneously interact with environments that change all the time, ubiquitous applications have to rely on a discovery service capable of dynamically registering and finding instances of services that match these application's needs considering also their location.

UDS handles the scopes of availability of services as presented in Section 3, that is, the specification of the area wherein a requester must be located (or connected) in order to be able to have access to services provided by a given provider. In practice, a UDS *service record* (i.e. service description) contains the scope of a service, in addition to other data and attributes commonly used for describing services in other discovery protocols, such as name, access information (IP address and port), description of the service (attribute-valor pairs). Hence, any provider that registers a service with UDS should inform its coverage scope, otherwise it will be assumed to have global area scope.

As with most discovery services, client applications (i.e. service requesters) may query UDS to search for a given service, and for this it must specify the search criterion, i.e. the properties and attributes of the service they are looking for. As part of this criterion, the client has to inform the region where it is currently located, i.e. the location of the device on which it is running, and which can be obtained from MoCA's LIS. In the search process, UDS will select from all the services that match the criterion provided by the client, only those service providers that are available for the requester's current region. Hence, the set of service providers that UDS will inform to the client will include: those with services that have global area scope, providers of services with local area scope matching exactly the location of the client, and providers of services whose area scope have a sub-area matching the location of the client. To determine the latter, UDS retrieves the location hierarchy of regions from SRM and subscribes for notifications about any hierarchy change.

Operation	**Description**
Register	The service provider sends a message to UDS containing the service name, description and scope, and informing if it is mobile or stationary (default). If the scope is not informed, it is assumed to be global. The UDS replys with a confirmation message containing a unique registration code.
Refresh	Periodically, the service provider sends a pulse to UDS containing its unique registration code to revalidate a service record that it has previously registered.
Remove	The service provider sends a message to UDS containing its unique registration code to remove a service record that it has previously registered.
Query	A client sends a message to UDS containing its location and the description of the service it is looking for and gets a list of services that matches the description and whose scope comprises the region of the client. If the client doesn't inform its location, it will only get addresses of global services.
Subcribe	A client sends a message to UDS containing its location and the description of the service it wants to subscribe for being notified whenever a service that matches the description and scope registers at UDS
Unsubcribe	A client sends a message to UDS removing a subscription previously posted

Table 1: The main operations implemented by UDS.

UDS also supports providers of services with local area scope, executing on mobile devices. In this case, it may be possible that a service requester (client) queries UDS searching for a given service which is unavailable at the moment, but which may eventually become available as soon as the mobile device where the corresponding provider is executing enters the same region of the client. In order to handle this kind of situation, UDS offers another functionality which we believe to be extremely useful for ubiquitous scenarios like this. To prevent a client application from having to repeatedly poll the discovery service to check if a desired service has become available, UDS provides also an asynchronous notification service, which works as follows: a client subcribes for the detection of a service with a given selection criterion and scope, and when a provider of a service matching this criterion and scope registers with UDS, the client is notified, receiving all information needed to connect to that service provider.

Similarly, services offered by providers on mobile devices may become suddenly unavailable at a region for a number of reasons, such as the device's migration to another place, shortage of energy, drop of the wireless connection, or simply, because the device's user turned it off. Hence, a discovery service should also handle this kind of situation, which is the case for UDS. A UDS service record contains also a field that defines if that service is mobile, i.e. the provider is executing on a mobile device, or stationary, i.e. provider running on a static machine. This information is given by the provider when it registers its service with the UDS, with the following consequence: while stationary services will stay registered until they are explicitly unregistered by the provider, mobile services have to periodically refresh their advertisement at the UDS, in order to keep their UDS service record active. If they fail to do so (within a predefined time interval), UDS will assume that the service is no longer available and remove the corresponding service record.

According to all of the aforementioned capabilities, UDS offers an APIs (shown in Table 1) for the implementation of clients (requesters) and service providers that need to discover each other in an ubiquitous environment, through queries or asynchronous notifications.

5.1 Interaction with UDS

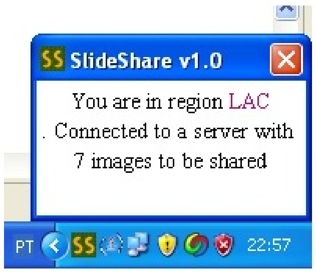
The interaction between UDS and the applications devised for the *active learning environment* scenario (presented in Section 2) illustrates well how the discovery service works. Considering the *slide presentation service* (SPS), when initiated, the SPS server would register with UDS informing its location scope as a given classroom (e.g. "room 202"). When a instructor enters that room and starts the SPS client on his device, it would first query the LIS server to get its present location (which happens to be the same "room 202"). Then it would query the UDS server to get a list of all appropriate servers available in "room 202" and would receive the address of that only SPS server.

For the *slide sharing application* (SSA), we could imagine that some students have arrived in the classroom before the teacher. When they start the SSA client on their devices, each application would first query the LIS server to get the location (again the "room 202"). Then each SSA client would query the UDS server to get a list of all appropriate servers available in that region and get an empty list as reply. In that case, each client would subscribe with UDS to be informed if a proper serve becomes available for that region. When the teacher comes to the room and starts the SSA server, it would query the LIS server to get the location ("room 202") and then register with UDS informing its location scope as "room 202". In that case, all SSA clients would receive notification messages from UDS informing the address of the newly arrived server. If some students arrived late, the SSA clients running on their devices would easily get the address of the server in the same way described in the first paragraph.

As to the *printing service* (PS), similarly to the first application described, the PS server would register with UDS informing the location scope of the service, but this time the region of scope would be defined as a larger one that contains some sub-regions (e.g. "2nd floor" that contains rooms 201 to 207). As the UDS server gets the information about hierarchies of symbolic regions from the SRM (which is responsible for managing it), if a client application located in "room 205" and other in "room 207", for instance, queried UDS to get the address of a PS server, both would receive the address of that server registered for "2nd floor".

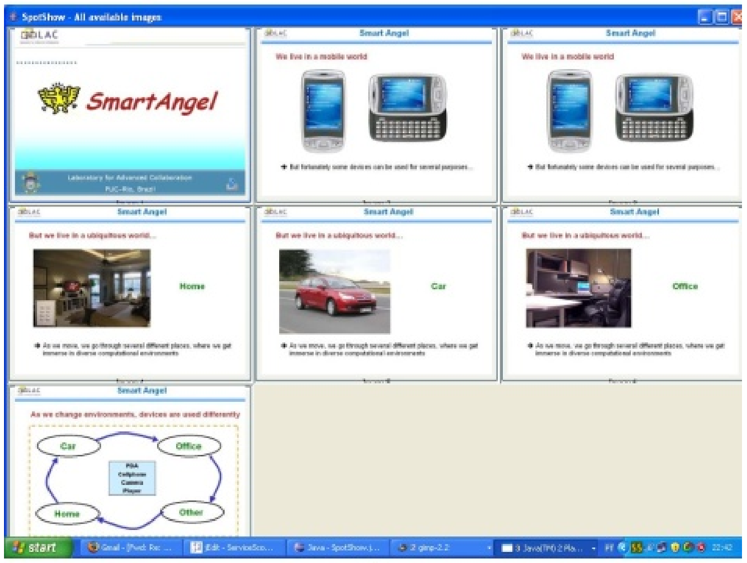
6 Case Study

We designed a prototype application called *SlideShare* aiming to support interaction among instructors and students in a classroom. Such an application is useful in the scenario B we described in Section 2. The *SlideShare* server, running on the teacher's notebook, is able to send a set of images to the students attending a class and carrying a mobile device where the client application in being executed. For that sake, the scope of the service is defined to be restricted to the region where the teacher (and his device) is located, i.e. a given classroom. Figure 2 (a) shows the pop up window the *SlideShare* client opens when it connects to a server. Figure 2 (b) shows the *SlideShare* window displaying all received images.


SlideShare v1.0

You are in region LAC
. Connected to a server with
7 images to be shared

(a)


SpotShow - All available images
SmartAngel
Smart Angel
We live in a mobile world
But we live in a ubiquitous world...
Home
Car
Office
As we change environments, devices are used differently
Other
start

(b)

Figure 2: The windows shown to the user at the SlideShare client

In fact, with the support of UDS this prototype application could be enhanced to become a P2P collaborative application, allowing the full interaction of teacher and students without the need of a central server. In such a way, an application running on the mobile device of a student or teacher that entered in the same room, would register with UDS having its address notified for all other people/applications in the same room. Then it would subscribe UDS, to be notified whenever a new person/application would come into the room. This MoCA based P2P application could operate in any place requiring only the MoCA core services running.

7 Related Works

The need to support not only “administratively scoped” but also “location scoped” discovery services has already been pointed out elsewhere [8] [9]. In nearly all discovery services already proposed, however, the scope of services are tightly related to the notion of administrative network domain [10], not considering the fact that the scope of a service might be restricted by the physical region wherein the given service is applicable. In other words, in most systems, the scope of a service is defined by the set of clients that can be reached within a given network.

Generally, in these systems the scope of a service is defined by the clients that are reachable by a multicast-based service advertisement (e.g. protocols that employ network-based address discovery mechanisms like SSDP [11], Jini [12] [13], SLP [14], Salutation [15]), or by the transmission range of low-level wireless broadcast mechanisms (e.g. systems based on RF transmissions like Bluetooth Service Discovery Protocol [16]). A different example is the Splendor protocol [17], a recent proposal of discovery service that takes into account the location of the service provider and the client, as well as the dynamic characteristics of systems where users move frequently. Clients and servers using Splendor can notice when they enter a new environment, and when this happens, they will look for a new service directory responsible for this new environment. Since in each region the directories are found using multicast announcements, also here the notion of service scope is very tied to the administrative scope. In discovery services like the ones just mentioned, the domains are logical but have no exact correspondance with the physical areas/places. However, as mentioned earlier, for ubiquitous applications it is often necessary to locate physically — not logically — nearby resources. For instance, if someone needs to transfer an image from a PDA to a computer with a larger screen, it is mandatory to select one with such a display that is close to the user, instead of one that is close to the client or server software in the network topology [12] [9].

Great ubiquitous computing projects chose to apply particular solutions for implementing service discovery. *Interactive Workspaces* [18] and *Aura* [19] have similar schemes in which each environment is responsible for managing the services available locally and the clients that want to use these services. *Oxygen* [20] adopts a totally distributed solution that pays more attention to the mobility of devices and takes into account the location of the clients to select the services. While in both *Interactive Workspaces* and *Aura* a client only can get information about services that are within the same network domain, the latter also also supports queries that select the closest services, based in the number of hops. None of the three, however, give support for locating services based on their physical location, nor to define the scope of availability for the services as does the system we implemented. While in those traditional systems the scope of a service is determined by a network-specific metric which bears no relation with the physical world, in UDS, on the other hand, service scope is defined in terms of physical locations that are not associated with network boundaries. Moreover, the matching criterion to select a service with a given location scope also takes into account the hierarchy of regions, thus enabling a more flexible and appropriate discovery service for ubiquitous computing.

8 Conclusion

A discovery service for ubiquitous systems must consider the mobility of the users and devices, which results in recurring disconnections and reconnections with different networks, and the corresponding dynamic change of the network and domain-specific resources and services accessible from the user's device. In such a dynamic scenario, some applications are interested in using the resources or services that are available in the user's vicinity or in the currently visited network domain, while on the other hand some services are available to be provided only within a well defined region. Considering these aspects, we implemented the *Ubiquitous Discovery Service* (UDS) for the MoCA architecture. In addition to several other common attributes of a service, UDS also uses the service provider's *scope of availability*, defined in terms of physical regions, as a selection criterion. Hence, UDS allows application clients (requesters) to search for a service matching the scope of availability of the server and the client's location. Furthermore, UDS offers a notification service that makes possible for client applications to register their interest in services with given characteristics and scope, and to be asynchronously notified when a desired service becomes available at the client's physical location.

To evaluate UDS we have implemented the application called *SlideShare*, that shows the behavior of a client/server application where the scope of the service is limited to a region (a classroom), and hence clients have to discover the correct application server depending on the location of each other. With this prototype application we tested the main functionalities of UDS service. A noticeable advantage of this way of implementing the discovery service is the outcoming simplicity to configure client applications, that previously need only to know the UDS address. Even the addresses of the basic MoCA services such as CIS and LIS may be obtained from UDS. As such, this new discovery service achieves its purpose in liberating the user from the tedious and repetitive activity of configuring the applications [21]. Moreover, the implementation of services that are available only for restricted areas (scope), allows the distribution — and hence simplifies — the management of such services.

As future work, we plan to evolve UDS so as to implement also the management of administrative network domain scopes — defining aggregations of devices and services based on administrative requirements —, but first we will have to adapt the MoCA architecture so as to provide some support for the description of administrative network domains. Furthermore, we are aware that other ubiquitous applications should be implemented to fully explore the new functionalities offered by UDS, in particular, to show the benefits of asynchronous notifications for service provider's availability.

References

[1] WEISER, M.. **The computer for the twenty-first century**. Scientific American, 265(3):94–104, September 1991.

[2] KINDBERG, T.; FOX, A.. **System software for ubiquitous computing**. Pervasive Computing Magazine, 2002.

[3] MURPHY, A.; PICCO, G. ; ROMAN, G.-C.. **Lime: a middleware for physical and logical mobility**. In: PROCEEDINGS OF THE 21ST INTERNATIONAL CONFERENCE IN DISTRIBUTED COMPUTING SYSTEMS, 2001.

[4] CHAKRABORTY, D.; JOSHI, A. ; AMD T. FININ, Y. Y.. **Toward distributed service discovery in pervasive computing environments**. IEEE Transactions on Mobile Computing, 2006.

[5] SACRAMENTO, V.; ENDLER, M.; RUBINSZTEJN, H. K.; LIMA, L. S.; GONÇALVES, K.; NASCIMENTO, F. N. ; BUENO, G. A.. **MoCA: A middleware for developing collaborative applications for mobile users**. IEEE Distributed Systems Online, 5(10), 2004.

[6] NASCIMENTO, F. N.; SACRAMENTO, V.; BAPTISTA, G.; RUBINSZTEJN, H. K. ; ENDLER, M.. **Development and evaluation of a positining service based in ieee 802.11 (in Portuguese)**. In: PROCEEDINGS OF THE XXIV BRAZILIAN SYMPOSIUM ON COMPUTER NETWORKS (SBRC 2006), 2006.

[7] THOMPSON, M. S.; MIDKIFF, S. F.. **Service description for pervasive service discovery**. In: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS WORKSHOPS (ICDCSW'05), 2005.

[8] FRIDAY, A.; DAVIES, N. ; CATTERALL, E.. **Supporting service discovery querying and interaction in ubiquitous computing environments**. In: PROCEEDING OF THE 43RD ACM SOUTHEAST CONFERENCE, 2001.

[9] MCGRATH, R.. **Discovery and its discontents: Discovery protocols for ubiquitous computing**. Technical Report UIUCDCS-R-99-2132, Department of Computer Science University of Illinois, Urbana-Champaign, May 2002.

[10] EDWARDS, W. K.. **Discovery systems in ubiquitous computing**. Pervasive Computing Magazine, 5(2):70–77, April-June 2006.

[11] JERONIMO, M.; WEAST, J.. **UPnP Design by Example**. Intel Press, 2003.

[12] FRIDAY, A.; DAVIES, N.; WALLBANK, N.; CATTERALL, E. ; PINK, S.. **Supporting service discovery, querying and interaction in ubiquitous computing environments**. Wireless Networks, 10(6):631–641, 2004.

[13] HARIHAR, K.; KURKOVSKY, S.. **Using jini to enable pervasive computing environments**. In: PROCEEDING OF THE 43RD ACM SOUTHEAST CONFERENCE, March 2005.

[14] GUTTMAN, E.. **Service location protocol: Automatic discovery of ip network services**. IEEE Internet Computing, 3(4):71–80, 1999.

[15] SALUTATION CONSORTIUM. **Salutation Architecture Specification v. 2.1**, 1999.

[16] BLUETOOTH CONSORTIUM. **Specification of the Bluetooth System v. 1.1 core**, 2001.

[17] ZHU, F.; MUTKA, M. ; NI, L.. **Splendor: A secure, private and location-aware service discovery protocol supporting mobile services**. In: PROCEEDING OF THE 1ST IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS (PERCOM'03), 2003.

[18] JOHANSON, B.; FOX, A. ; WINOGRAD, T.. **The interactive workspaces project: Experiences with ubiquitous computing rooms**. Pervasive Computing Magazine, 2002.

[19] SOUSA, J. P.; GARLAN, D.. **Aura: An architectural framework for user mobility in ubiquitous computing environments**. In: PROCEEDINGS OF 3RD IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE, 2002.

[20] RUDOLPH, L.. **Project oxygen: Pervasive, human-centric computing - an initial experience**. In: PROCEEDINGS OF 13TH INTERNATIONAL CONFERENCE IN ADVANCED INFORMATION SYSTEMS ENGINEERING (CAISE 2001), 2001.

[21] ZHU, F.; MUTKA, M. ; NI, L.. **Classification of service discovery protocols in pervasive computing environments**. Technical Report MSU-CSE 02-2, Michigan State University, East Lansing, 2002, 2002.