Context-aware Service Discovery in Mobile Heterogeneous Environments

N. Blefari-Melazzi†, E. Casalicchio*, S. Salsano†
* Dip. Informatica Sistemi e Produzione, Universita di Roma "Tor Vergata", Italy
† Dip. Ingegneria Elettronica, Universita di Roma "Tor Vergata", Italy
Email:emiliano.casalicchio@uniroma2.it

***Abstract*— To date, mobile services have failed to match the explosive growth of the Web. This, we argue, is because current mobile services are difficult to find, to use, to trust, to design and deploy. The IST SMS (Simple Mobile Services) project takes up the challenge of creating innovative tools addressing the specific needs of mobile users and making it easier for individuals and small businesses to become providers of Simple Mobile Services. In this paper, we focus on the first requirement of SMS: how to find a mobile service matching the user needs and appropriate for the user context. The contribution of this paper is twofold: first we formulate the context-aware service discovery problem in a heterogeneous mobile environment; then we propose a novel solution that provides mechanisms to discover services at every level of the service stack, from infrastructure services to end-user services. We propose a hybrid solution that combines the advantages of a directory-based solution and of a peer-to-peer fully distributed solution. The advantage of a fully distributed solution is the increase of efficiency in a local area. The advantage of a directory-based solution, exploiting an XML-based service discovery protocol and using a hierarchical organization of the directories, is that it allows obtaining a scalable and fault-tolerant system, which can be used both in a local and in a wide-area network.**

I. Introduction

Mobile services have not matched the success of the Web. There are many reasons: users cannot find the services they need, many services are difficult to use, users do not trust them, services are difficult to design and deploy (especially for "small" service providers, e.g. SMEs, local government departments, NGOs, individuals).

The goal of the IST SMS project [6] and [12] is to create innovative tools enabling a new class of services, addressing the specific needs of mobile users and enabling individuals and small businesses to become service providers. We call these services Simple Mobile Services (SMS). Unlike current universal services, each Simple Mobile Service will have a scope, it will target specific environments of interest to specific classes of mobile user performing specific activities. This means that SMS services will be easy to find. SMS services will be easy to use: automatic authentication and configuration; automatic content and interface adaptation. SMS services will be terminal and network independent, working with a broad range of mobile devices and network infrastructures. SMS services will be trust-worthy, providing end-to-end standard-based mechanisms for positive user identification, authentication, and data encryption (both on terminals and during transmission). Last, but not least, SMS services will be easy to design and deploy.

In this paper we face the problem of how to find a service matching the user needs and appropriate for the current user context. This is a well known problem in the literature, and it is referred to as context-aware service discovery (we may say that SMS services are a sub-class in the broader family of context-sensitive services). Current approaches to this problem may be classified according to the type of service, to the technology, or to the the applications that one may consider.

In [8] the authors propose a context-aware service discovery and selection mechanism. The proposed solution uses a centralized architecture, where a dedicated server manages the context processing. In [3] the authors propose a context-aware service discovery architecture and related protocols, addressing the discovery problem in many of its aspects. Their solution aims to discover a pre-defined set of services, such set has to be configured by the user, both in the local or in a foreign network. On the other hand, our goal is to discover available services matching context and user profile, without having any prior knowledge of the services themselves.

The novelties of our proposed solution are: (i) it is designed to find services that are indeed useful to the users, without being known in advance by the users themselves; (ii) it is designed from the beginning having in mind context-aware mobile services, rather then proposing a mobility support or an extension of an existing architecture devised for wired solutions. (iii) it can support multimodal services and aspect-oriented components. Furthermore, our solution combines the advantages of a directory-based architecture and of a peer-to-peer fully distributed solution. The advantage of a fully distributed solution is the increase of efficiency in a local area, which allows also a p2p exchange of services. The advantage of a directory-based solution, exploiting an XML-based service discovery protocol and using a hierarchical organization of the directories, is that it allows obtaining a scalable and fault-tolerant system, which can be used both in a local and in a wide-area network.

The paper is organized as in the following. In Section 2 we introduce the problem of context-aware service discovery in heterogeneous networks and we define the requirements of the SMS service discovery infrastructure. Section 3 deals with service description aspects. Section 4 faces the context modeling problem. In Section 5 we present our porposals for

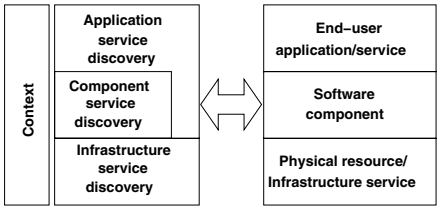

Fig. 1. Context-aware service discovery stack

context-aware mobile service discovery protocols and for the for the service infrastructure. In Section 6 we make some final remarks.

II. Context-aware service discovery

The service discovery is typically defined as the process to automatically discover devices (e.g. sensors, printers, data storages unit, etc), software components and/or distributed applications which are available in a network. If the discovery process is driven by context information or user specific requirements we deal with context-aware service discovery.

In our vision, the service discovery process is organized in three layers (see fig. 1): Infrastructure service discovery, Component service discovery, "Application-level" or "End-user" service discovery. The context is an intra-layer concept, as observed in [3]. Starting from the lower level, infrastructure service discovery is concerned with the discovery of physical resources or network services (e.g. authorization or location services, or physical devices like a printer). At this level, services are typically accessed only by knowing low level information (or raw information) such as the IP address and port number of the node providing the service. The way in which the terminal can find out service locations depends on the service discovery protocol (SDP) used. Directory-based SDPs (e.g. JINI, IETF-SLP or DNS-based service discovery [2]) periodically broadcasted the address of a lookup service (or a service proxy), so that it can be caught by the mobile terminal. Then, services are announced and discovered trough the lookup service. In peer-to-peer based SDPs, e.g. UPnP, the infrastructure services periodically broadcast their announcements.

Moving to the upper level, let us consider the end-user service discovery. A end-user service typically returns to the user some type of contents: information about its position, a route to a given destination, the set of attractions near a given location or in a given area, the set of restaurants matching the user profile, and so on. All these information are correlated with the user profile and context. Then, the problem to discover end-user services has things in common with the matching problem in publish/subscribe systems: a user first specify its interests and performance requirements in a user profile and then receives notifications of services that match his profile and that are appropriate for the actual context. A proposal of a matching algorithm for SMS is presented in [4].

In order to understand the functionality of the intermediate level, i.e., the component service discovery, we have to make an assumption on how the end-user services are realized. We assume that end-user services are provided by means of a set of applications running on the network and interacting with each other. Such applications could run both at the terminal side or at the server side and are referred to here as "components". Therefore end-user services can be realized by putting together software components. Note that the composition can be made either in a static or in a dynamic way. In the former case, the component service discovery is needed during the authoring phase of the end-user service, so that the application will know where the software components will be, at the execution time. In the latter case the binding of the components is dynamic: a component will need to discover, at run time, where are other components that allow satisfying the requirements of the end-user service.

Following the approach of the SMS project, in this paper we will mainly focus on component and end-user service discovery, while for the infrastructure service discovery we assume to rely on existing solutions.

A. Requirements for context-aware service discovery

There are some general requirements that the context-aware mobile service discovery architecture should satisfy, independently from the level. Scalability, fault tolerance, platform independence, and high expressiveness for service and context description are among these general requirements. The service discovery infrastructure should scale with the number of nodes, with the number of networks and with the number of services and users. The SDP should consume as less resources as possible, both in terms of bandwidth and in terms of CPU cycles and memory. The SDP should also be platform independent (both at terminals and at server side). The chosen services and context description model should have an high expressiveness, allowing a detailed description of the service semantic, functionality and requirements. The service description model should also be independent from the service technology and the service implementation.

Coming to the requirements that depend on the specific level of the service discovery stack, we only mention some requirements of the infrastructure service discovery. It should be automatic and transparent to the end-user and service developer. Possibly, a terminal (if enabled) should automatically detect available SMS services.

At the component level, first of all the infrastructure should preserve the compatibility with existing service discovery protocols, which today are typically dependent on the technology used to implement the service itself (e.g. JINI, Web services, IETF-SLP or DNS-based service discovery). Moreover, both static and dynamic component service discovery should be supported. If we use a static component service discovery, i.e. performed during the component service authoring phase, then it is possible to have a very efficient service selection process. However, in this case, it could be complex or impossible to implement features like context awareness and multimodality. If we use dynamic service discovery, then components are able to select other components to bind with at run time. Run

time binding allows realizing an ”abstract” service definition, to be executed in any SMS-enabled network that provides an implementation for the specified service. The dynamic service discovery offers a good support for service multimodality and aspect oriented components.

As regards multimodality, in this paper we define a service as multimodal if it can be implemented in different ways, and the implementation is selected as a function of the user profile or context. An example is a payment service. The payment could be done by using a credit card, a transaction authorization code stored in a RFID, a cell phone credit, etc. When more then one implementation of a service exist, this characteristic should be specified in the service description, and, depending on the user profile and user context, a particular component service implementation should be used. Another advantage of the dynamic service discovery is the possibility to use service a selection policy, with the goal to optimize some system performance metrics (e.g., cost, response time, reputation) [9].

Coming to the end-user service discovery, it should basically consist of a matching algorithm that associates users with services that best match their current context. In other words, given a user profile (describing user characteristics and preferences), a user context and a list of service profiles (defining the target for each service) the SMS machinery has to be able to rank service profiles in terms of their match to the user.

III. Context-aware service description model

We claim that a common service description model can be used for service discovery at the component level and at the end-user level. Note that at the component and end-user level, we should allow both users and processes to search for available services. In order to achieve this, the service description must be available both in human and machine understandable language. The usage of an XML based language will help achieving this goal. In particular, it is possible to add more intelligence to the system by representing the service description as RDF.

The component service description model should allow: an abstract description of the service semantic and API; an abstract description of the minimal performance requirement to run the service (service requestors could be able to support/execute only a limited set of services, could have limitation in content visualization or could have particular security/performance - QoS - requirements); an abstract description of multimodality features. Given the diversity of the information that should be represented, It could be useful to employ a set of XML documents (WSDL, Topic map, etc.) rather than a simple one.

The expressiveness of the end-user service description model should allow an abstract description of the service semantic, functionality and available level of service quality. The service description fields should be compatible with the user profile and context description so as to facilitate the matching of user preferences.

A context-aware service descriptor involves, at least, three distinct parts (see fig. 2). The service semantic and APIs are

```
<?xml version='1.0' encoding='UTF-8'?>
<service_descriptor>
     <functional_requirements>
          <semantic>
          </semantic>
     </functional_requirements >
     <nonfunctional_requirements>
          <security>
          </security>
          <qos>
          </qos>
     </nonfunctional_requirements >
     <pseudo_requirements>
          <middleware>
          </middleware>
          <configuration>
          </configuration>
          <multimodality>
          </multimodality>
     </pseudo_requirements >
</service_descriptor>
```

Fig. 2. An XML representation of the service descriptor.

described by means of the functional_requirement tag. The nonfunctional_requirements tagis, for example, for security and QoS service specifications. The security tag contains security-related information, such as security policies applying on the service or restrictions to user's access. The qos tag describes quality of service related aspects regarding the service requirements for different quality levels. All the service specific characteristics and the concepts that the service deals with are represented into the pseudo-requirements part. For instance, this part is concerned with: explicit multimodality capability, configuration requirements of the software component or of the service requestor's device, requirements for a specific communication protocol or execution platform.

IV. Context modeling

Context modeling is one of the main building blocks of the SMS project. Deliverable [4] describes the SMS project context modeling approach and includes in appendix an extensive state of the art analysis on the subject. The context modeling approach is derived from ContextUML [5], which is an UML-based model for the specification and model-driven development of Context-aware Web Services. The SMS context model information includes information about the user, the devices, the services, the networks. The schemas for modeling these information are largely derived from the work of the IST Simplicity project ([7] and [13])

The user profile contains personal (non technical) and human sensitive preferences (e.g. food preferences, favorite attraction, favorite shopping, friends' white and black list, etc).

The device-related information describes the capability and technical characteristicsof the terminal device (e.g. the processing capability, the storage and memory capacity, the display resolution, the connectivity capacity, the operating systems version, etc). Moreover, there is a section that describe user preferences related to the usage of the device. The service profile contains information about the service provided. For example, a restaurant would like to publish a reservation

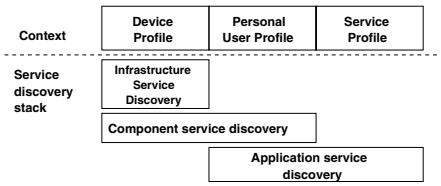

Fig. 3. Relationship among the Context model and service discovery layers

service and information such as average price, opening hours, smoking or non smoking area availability, etc.

Being the context orthogonal to the service discovery layers, theservice discovery process will use context information depending on the service level (see fig. 3): the device profile is used in the infrastructure service discovery; device and user profile are used in the component service discovery; the end-user service discovery will use the personal user profile and the service profile.

An open issue is how smart one has to make the context processing in order to enable end-user service discovery. In this regard, different techniques could be used to process the context and to match it with the user profile, for exampleartificial neural networks [8], or semantic indexing methods [10].

V. The service discovery protocols

Service discovery protocols could be directory-based or peer-to-peer based. The peer-to-peer approach is appropriate for small networks with a limited number of services and nodes, because of the huge traffic produced by the multicasting of service advertisements or discovery messages. The advantage of this solution is that it avoids the setup and maintenance of registries. The directory-based approach has the advantage to reduce the bandwidth consumption, it moves the intelligence into the network and the directories could be hierarchically connected, so as to cover a wide area network. The drawback is obviously the setup and maintenance of registries. Our proposal in an hybrid one, which combines the advantages these two approaches.

Our idea is that each node in the network could provide end-user and component services and it should be capable to advertise the offered services or to acknowledge service requests. Then, each node in the network, at a given instant, could became an SMS service provider (SMSsp). Two main solutions can be envisaged for the user to access SMS services. A first solution is that users launch their SMS application or SMS enabled browser and get back a list of services that satisfy its profile. A different approach is that users are automatically notified of existing SMS services, every time that a new one is available.

Whatever is the solution used to access the list of available SMS services, the user terminal will use the infrastructure service discovery protocol (iSDP) to discover SMSsp and then it will use the end-user service discovery protocol (euSDP) to ask to the SMS servers the services that match the user profile and context (see fig. 4). Obviously, this solution requires that the iSDP uses broadcasts and that every SMSsp is capable to

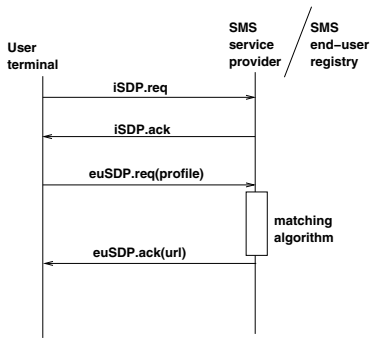

Fig. 4. Infrastructure service discovery protocols and end-user service discovery protocols messaging. In case of a p2p solution the `iSDP.req` is broadcasted in the network.

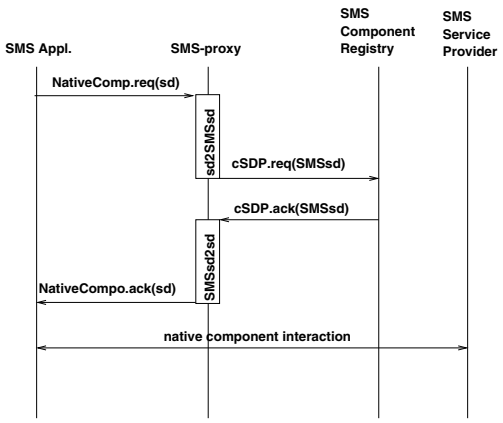

Fig. 5. Component service discovery protocol messaging.

match the user profile with the service profile. To avoid such overhead, a possibility is to introduce the end-user service registry, a specialized server, which stores service descriptors and executes the end-user service matching algorithm.

When the SMS application is executed (on the the user's terminal, on the SMS server node or on both of them), the business logic could require the invocation ofexternal components. The application logic will generate a native component request (e.g. uddi, RMI or JXTA), depending on the technology used in the implementation. On the SMS node that executes the application there is an SMS-proxy that receives the native component request and creates a cSDP request, which contains in its body the native request (see fig. 5). The cSDP request is then sent to the component service registry, which will search for the matching components and will reply with the component service descriptor. The SMS-proxy will extract the native service descriptor from the cSDP message and will deliver it to the application. The address of the more appropriate component service registry could be automatically configured by using the iSDP every time the mobile terminal enters an SMS network.

A. The service registries

The use of an end-user service registry allows a more complex service matching algorithm and reduces the traffic

generated by a broadcast of service discovery requests. The main drawback of using registries is their setup and maintenance.

At component level, the use of a registry will speed up the dynamic binding process. As a matter of fact, an SMS node needs to discover the address of the service registry only one time.

Because the service descriptor is an XML document (as the context descriptor), the registry could be easily implemented by an XML database (e.g. eXist [11]) and queried by an XML querying language (e.g. the W3C XMLQuery).

If the hybrid solution is used, each SMSsp does not need to implement a XML database to store the service descriptor, but it can simply implement the matching rules between the user/service profile and the service descriptor.

B. The hierarchical registry organization

To cover a wide area network and to realize an efficient and recursive service discovery, the component and end-user service registries could be organized into a hierarchy. This architecture satisfies the scalability requirement and it is also fault-tolerant. Each SMS network has its own local Service Registry, both for component services and for end-user services. To realize a fault-tolerant system, and also to increase the performance in crowded networks, one or more secondary local service registry could be introduced. "'near"' SMS networks have a common, i-th level, global registry. Analogously, "'near"', i-th level, global registries have a common (i-1)-th level global registry, and so on, until the 1th level (or root) global registry is reached. The "near" network could be identified by using different proximity metrics, for example the physical distance, the latency, the number of hops, etc.

With a hierarchical architecture, when a user's terminal searches for available end-user services, first it queries the local registry; then the query goes up in the hierarchy of registries, in a recursive way. It is reasonable that the probability to find a context/profile compliant service in a far network is very low, but this depends on the context and on the user profile.

In the same way, when an application searches the component services to bind, first it queries the local registry and then the request is propagated up in the hierarchy.

VI. Concluding remarks

In this paper, we specified how to perform service discovery in a particular environment, that defined in the framework of the SMS project. However, we also proposed possible solutions for service discovery protocols at different levels.

During our study, we soon realized the need for new service discovery protocols allowing context awareness at every level of the stack, from infrastructure service discovery to end-user service discovery. We gave a possible answer to this need by introducing and detailing three service discovery protocols: infrastructure SDP, component SDP and end-user SDP. Our proposed solution: i) faces the problem of end-user service discovery in local area by giving the possibility to use or not a central registry; ii) allows an easy way to advertise custom services in local area and the creation of small community exchanging services; iii) enables also wide area service discovery, allowing the use of centralized registry; at component level, we encourage the use of centralized directories; as a matter of fact, component registries could be hierarchically organized toobtain a scalable and fault tolerant system, usable both in local and in wide-area networks.

As for future work, we are currently engaged in implementing our solution, so as to test and evaluate the it in real environments. Also simulation studies will be performed to compare the performances of a centralized or fully distributed solutions.

Acknowledgment

This work has been performed in the framework of the European Union co-funded project SMS. The authors would like to acknowledge the contributions of their colleagues from the SMS consortium. The Community is not liable for any use that may be made of the information contained therein.

References

[1] A.Friday, N.Davies, N.Wallbank, E.Catterall, and S.Pink. *Supporting service discovery, querying and interaction in ubiquitous computing environments.* ACM Baltzer Wireless Networks (WINET) Special Issue on Pervasive Computing and Communications, 10(6):631-641, November 2004.

[2] S.Cheshire, M.Krochmal. *DNS-Based Service Discovery* Internet-Draft. http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt. August 2006.

[3] Choonhwa Lee, Sumi Helal. *A Multi-tier Ubiquitous Service Discovery Protocol for Mobile Clients*. 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems.

[4] S.Salsano (ed.) *D3.1: Initial system architecture specification*. IST-SMS Project deliverable, Dec. 2006, available at http://www.ist-sms.org/server/deliverables.php

[5] Q. Z. Sheng, and B. Benatallah, *ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services*, The 4th International Conference on Mobile Business (ICMB'05), IEEE Computer Society. July 11-13 2005, Sydney, Australia

[6] G. Bartolomeo, N. Blefari Melazzi, G. Cortese, A. Friday, G. Prezerakos, S. Salsano, R. Walker. *SMS: Simplifying Mobile Services - for Users and Service Providers*, International Conference on Internet and Web Applications and Services, ICIW'06, Feb. 2006, Guadeloupe, French Caribbean.

[7] G. Bartolomeo, N. Blefari Melazzi, F. Martire , S. Salsano, *Defining and Using Profiles to Personalize and Manage Reconfigurable Services*, 15th IST Mobile&Wireless Communications Summit 2006, June 4-8 2006, Myconos, Greece.

[8] Al-Masri, E. and Mahmoud, Q. H. *A context-aware mobile service discovery and selection mechanism using artificial neural networks*. In Proceedings of the 8th international Conference on Electronic Commerce Fredericton, New Brunswick, Canada, August 2006. ACM Press, New York, NY,

[9] Cardellini, V.; Casalicchio, E.; Grassi, V.; Mirandola, R., *A Framework for Optimal Service Selection in Broker-Based Architectures with Multiple QoS Classes*. Services Computing Workshops, 2006. SCW '06. IEEE , vol., no.pp.105-112, Sept. 2006

[10] Skouteli, C., Samaras, G., and Pitoura, E. *Concept-based discovery of mobile services*. In Proceedings of the 6th international Conference on Mobile Data Management. Ayia Napa, Cyprus, May 2005. MDM '05. ACM Press, New York, NY

[11] Akmal B. Chaudri, Awais Rashid, Roberto Zicari (Eds.): XML Data Management: Native XML and XML-Enabled Database Systems. ISBN: 0-201-84452-4. Published by Addison Wesley Professional, March, 2003.

[12] IST SMS Project www.ist-sms.org.

[13] IST Simplicity Project www.ist-simplicity.org.