An Open Architecture for Developing Mobile Location-Based Applications over the Internet

Rui José, Adriano Moreira, Filipe Meneses
Information Systems Department
University of Minho
Azurém, 4800-058 Guimarães, Portugal
email {rui,adriano,meneses}@dsi.uminho.pt

Geoff Coulson
Distributed Multimedia Research Group
Department of Computing, Lancaster University
Bailrigg, Lancaster, LA1 4YR, UK
geoff@comp.lancs.ac.uk

Abstract

The mobile Internet is enabling a broad range of new applications that dynamically obtain information that is relevant to their current location. This type of application would greatly benefit from generic mechanisms for supporting the association between network resources and physical space, but existing systems are typically based on vertical approaches valid only for narrow application scenarios. This paper argues that a comprehensive solution to this issue should address the important challenges of heterogeneity and openness, and proposes an approach based on the concept of location-based service, i.e. a service whose usage is associated with physical space, as a generic abstraction to support the development of location-dependent systems. The paper describes a model for associating location scopes with services, an architecture to support the discovery of location-based services on the Internet, and a prototype infrastructure in which several services and applications have been developed for validating the architecture.

1. Introduction

With the expansion of mobile Internet services, mobile users now expect to have access to information in the places where it is more relevant and not just in front of a home PC. In particular, the increasing availability of information about the location of mobile users is prompting considerable interest for applications that dynamically select the information that is relevant for that location, raising the issue of how to support a systematic correlation between information sources and physical location. We believe that a comprehensive solution to this issue should consider two important challenges of the mobile Internet environment. The first challenge is to abstract over the technological heterogeneity that characterises that environment, avoiding assumptions about the use of specific networking or positioning technologies and offering high-level abstractions capable of hiding the complexity of the system from application developers. The other challenge is to create an open architecture to which new components may be added or removed, allowing the overall system to evolve and grow continuously without assuming a centralised control of its components.

We propose to address these issues with a service-based architecture that supports location-based discovery. The decomposition of applications into independent services is a classical solution for addressing the issues of heterogeneity and openness in distributed systems. Services provide the resources that applications need but are independent from them, and thus can be independently developed and used by multiple applications. To extend the advantages of service-based architectures to the development of location-dependent applications, we propose in this paper a model of location-based service as a generic mechanism for associating network resources with location.

The outline of the paper is as follows: The following section describes our model for the association between network services and physical location. Then, in section 3, we describe the architecture of the AROUND system, a discovery mechanism that allows services relevant to a specific location to be found over the Internet. This is followed, in section 4, by a case study of a prototype application that exploits the location-based services framework and provides the evaluation of the system. In section 5, we analyse some existing systems relating them to our own work, and, finally, in section 6, we present our concluding remarks.

2. A Model for Location-Based Services

In the context of our work, a location-based service is a process or system providing a facility to the network whose usage is directly associated with physical space.

* 6th IEEE Symposium on Computers and Communications, 3-5 July, 2001, Hammamet, Tunisia

This location-based usage is a fundamental characteristic of these services, and may result from the provision of some interaction with the physical environment, e.g. controlling the temperature of a room, from acting as electronic counterparts to real-world entities, e.g. a restaurant, or from the provision of information associated with a geographical area, e.g. maps or traffic information. The abstraction of locality introduced by location-based services, while not appropriate for every application domain, represents a simple and yet powerful concept that may be useful for developing location-based systems across a wide range of scenarios. This section describes how the association of services to location is supported by the AROUND architecture.

2.1. Proximity models for service discovery

A key issue when considering a service discovery mechanism that selects services relevant to some physical location is how to model the relationship between services and physical space, and in particular which geographical criteria to use for determining the "nearby" services that the client should discover. We have considered two distinct models of location-based service discovery, as follows.

The *distance-based* model uses a concept of proximity based on the physical location of servers. A client is able to specify a distance range and discover the servers located within that range from its own position. A limitation of this model is that the correlation between context and proximity tends to decrease as we enlarge our notion of proximity, i.e. more things that we do not care about will be seen as being in our "proximity". This raises concerns about the scalability of this model for supporting service discovery over large areas.

The *scope-based* model uses a concept of proximity based on services scopes. Each service is assumed to have an associated scope that specifies the physical range in which it should be available. A client is able to discover a service if it is located within that service's scope. The client can further refine discovery by limiting it to services with a scope that is larger or smaller than some physical scale. By centring proximity on service scopes, this model makes the location of servers irrelevant, both in terms of physical location and network location. Its most important characteristic is that the correlation between context and proximity is assured. The services discovered, no matter how distant, are guaranteed to be relevant to the client's location. This provides a very powerful and scalable model of proximate selection, in which the range of proximity can be expanded from very small scales to very large scales without any exponential increase in the number of discovered services.

We believe that it is relevant to support both these models because each of them is the best approach for different types of location-based service. Distance-based is the most adequate model for services with a strong association with a specific point in space, typically services acting as electronic counterparts to real-world entities, e.g. a restaurant or a bus-stop. Finding the physically nearest server is usually the most natural type of query for services of this nature. On the other hand, scope-based is the most adequate model for services with a usage that is geographically bounded but is not linked to any particular point in space, such as maps, weather forecasts or restaurant guides. Our architecture is centred on the scope-based model, and uses it as the primary mechanism for associating services with location. Services are always assumed to have some associated scope. This design choice is primarily motivated by the scalability of the scope-based model, and by the fact that servers do not need to be located within the area they serve. As a complement to the use of scopes, we also support distance-based selection, allowing services to have a location attribute and to be discovered based on that criterion. However, given the high level of abstraction of this architecture, we do not aim to support fine-grained real proximity in small physical ranges.

2.2. Location contexts

An important element of this architecture is the definition of a scope model that explicitly expresses service usage as an area in physical space. Our approach to the expression of service scopes is to support a shared set of symbolic locations, known as location contexts, which are explicitly referenced through global names, and can be used across multiple networks and domains as contexts for service location. Servers making service registrations and clients making service requests include a reference to the particular location context in which they wish to perform their respective operations. Thus servers, when registering location-based services, use these location contexts to indicate the scope of particular services; i.e. the set of location contexts with which a service is associated. Similarly, clients, when searching for location-based services, use one or more location contexts to indicate the required service selection area. Location contexts are abstract symbolic entities that refer to physical environments, e.g. "Building A" or "City Centre", and can be typed according to the nature of the places they represent, e.g. building, room, or town. The additional spatial semantics introduced by context types allows applications to deal with relative distances and geographical scale without requiring explicit user input.

2.3. Relationships between location contexts

In our architecture, location contexts can be linked through unidirectional relationships whose goal is to enhance the process of service discovery by transforming

location contexts from isolated service aggregations into components of a shared distributed service location space organised according to spatial criteria. Relationships between contexts establish a mechanism for the propagation of queries from a source context to a target context, resulting in a graph structure similar to that of federated trading architectures [8]. However, unlike links between federated traders, links between location contexts have a spatial semantics that determines the way queries are propagated in the graph and thus permit the search domain to be specified in spatial terms.

Our architecture employs two types of context relationships: containment and adjacency. The containment relationship reflects the spatial inclusion of the area of a contained context within the area of a container context, and defines a partial order over any arbitrary set of location contexts. Each location context can be contained by more than one context. Containment is employed as the central relationship of the architecture because of its key role in supporting the scope-based proximity model. A containment relationship implicitly makes the services registered at the container context available in the contained context. Fig. 1 exemplifies a set of location contexts linked by containment relationships and a set of services, A,B and C, being registered in their respective contexts. The lower half of each context indicates the services that can be discovered at that context.

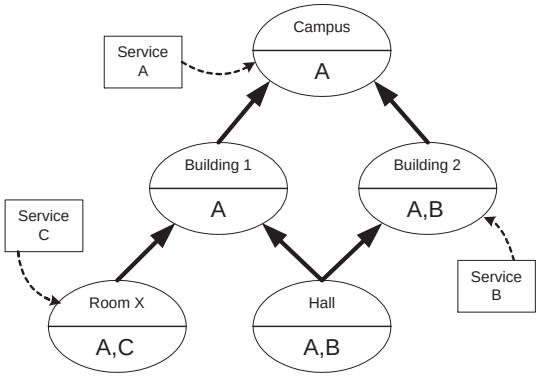

Figure 1. Containment relationships

Service A, registered at the "Campus" context, is available everywhere because all other contexts are directly or indirectly contained in the "Campus" context, whereas service C, registered at the "Room X" context, only becomes available at that context. Thus, a client searching for services in the "Campus" context can only discover service A whereas a client searching in "Room X" can discover services A and C.

In addition to the containment relationship, we also make use of the adjacency relationship, which expresses an immediate physical proximity between the areas of two location contexts. The main objective of the adjacency relationship is to support the model of distance-based proximity, thus allowing the physically nearest services to be discovered, even if out-of-scope. From an initial context, a search query can be passed to its adjacent contexts. Then, assuming that services have a location attribute, they can be selected based on their distance to the client and not just on the contexts in which they were registered. The adjacency relationship can also play an important role in supporting rapid context changes by allowing clients to adopt a "pre-fetching" attitude in which they perform service discovery in adjacent contexts before actually entering them.

3. The AROUND system

This section describes the architecture of the AROUND system, which we propose as a mechanism for supporting the discovery of location-based services over the Internet.

The functional structure of the AROUND architecture comprises the *AROUND service*, the *contextualisation* process, and the *name service*. The *AROUND service* is a distributed service location infrastructure organised by location contexts and managed by AROUND server entities. *Contextualisation* is the process of determining the location context that best corresponds to the current location of a mobile device. This process can be based on information obtained from multiple types of location sources and addresses the issue of associating the physical location of a device with a location context. The location context or contexts that result from the contextualisation mechanisms are called *base contexts*. The *name service* resolves global location context names into references to specific AROUND servers at which they can be accessed.

To make its behaviour depend on the changing set of services available at each given location environment the application interacts with the various components of the architecture in the following way: When the contextualisation process detects a change in the current location context, it notifies the application of that change, indicating the name of the newly entered context. This event triggers the mechanisms that will lead to a change in the set of services being used by the application. Firstly, the application contacts a name service to resolve the name of new location context into one or more references to AROUND servers that provide access to that particular location context. Having obtained such references, the application queries the respective AROUND server about the services available for the newly entered location context and selects the ones that are most appropriate for its particular information needs. The application can then interact with location-based services that are specific to its current physical location and obtain the information it needs to reflect the particular information space of the new environment.

3.1. AROUND service

Our service location architecture is implemented in the AROUND system primarily through the mechanism of AROUND servers, each of which manages a set of location contexts. The information associated with each location context includes not only its respective service registrations, but also local policies for query management, attributes describing the location context (e.g. its type), and any containment or adjacency relationships to other contexts. These links to other contexts, which we assume to be set up manually, can either be internal to the server, i.e. to other location contexts on the same server, or external, i.e. to location contexts on other AROUND servers. External links enable servers to share their respective service offer spaces and form a larger distributed context space. Service location queries can thus traverse multiple servers and possibly multiple administrative domains before being completely answered. In order to improve the performance and reliability of these multi-server queries, location contexts can be replicated on multiple AROUND servers. This is particularly important for location contexts representing large scopes, as their services may be utilised by a large number of devices.

The interface to AROUND servers is largely based on the interface of the CORBA Trading service [8], but new features have been added to embody the particular characteristics of our service location architecture. The basic parameters of the query operation are: i) the name of the location context at which the query starts, ii) the types of services to select, and iii) a set of constraints that the attributes of the selected services must satisfy. These basic parameters have been extended with spatial scoping policies and a preferences expression for spatial attributes. Spatial scoping policies determine the set of location contexts that the query will traverse. In particular, queries can be spatially scoped by specifying a bounding location context or context type, and by defining the spatial direction of the query, i.e. whether to follow adjacency or containment links. Preferences can be used to request a particular ordering of the set of matched services. The "*nearest <position>*" expression has been added to support the ordering of services with an attribute location by their distance to the specified position, thus enabling the model of distance-based proximity in service discovery.

Supporting the AROUND service on the hostile Internet environment implies basic security services like authentication and data integrity of messages between the various entities in the architecture. We have defined a security framework for these basic services, and are currently extending that framework to support additional security services such as service certification. These mechanisms are not discussed in this paper.

4. Evaluation

To evaluate the architecture and gain some insight into the issues raised by the use of location-based services, a prototype infrastructure has been developed in the context of the AROUND project [1].

4.1. Prototype Infrastructure

We have developed a JAVA implementation of the AROUND server using the Jini framework [9]. Jini lookup services are used as service registries, and service definitions, e.g. service types and attributes, are based on the Jini programming environment. This option for a Jini-based implementation has been essentially due to reasons of easy prototyping, and does not imply any dependency of the architecture on the Jini technology. The system has been developed using JDK 1.2.2 and the SUN reference implementation of Jini (version 1.0).

The prototype infrastructure established for the AROUND project is now operational and we have begun to explore the use of location-based services for the creation of various location-dependent systems. Two AROUND servers have been installed to manage a heterogeneous set of location contexts, ranging from rooms at the University campus to areas of various sizes in the town of Guimarães and its surroundings.

The applications created for this case study are all focused on assisting travellers and in particular in providing public transportation information. This common theme has facilitated the creation of a service infrastructure to be used by multiple applications. The following service types have been created: a *BusInfo* service that gives information about the bus services for a specific area, including the buses locations; a *BusStop* service that acts as an Internet counterpart to the information typically available at real world bus-stops; a *Map* service that provides maps in several formats for specific areas; a *Weather* service from which several formats of weather forecast can be obtained; and a *SpatialInfo* service that provides structured information about the current location of the device, e.g. postal code, address, or a simple description. Some of these services, e.g. *SpatialInfo*, are available at various levels of the hierarchy of location contexts with various degrees of specificity. Others are available at specific types of location context, e.g. the *BusStop* service can typically be found in contexts representing small town areas.

The main application developed so far is a generic travel assistant running on a mobile device equipped with GSM and IEEE 802.11 network connections, and in which contextualisation is supported by GPS and network beacons. The application is designed around a set of independent agents, each of which acts as a mini-application, and en-

capsulates the intelligence required to interact with the set of services needed for a specific thematic area, e.g. transportation, guidance or weather. When a new context is entered, agents request the services they need, interact with those services, and generate a location-dependent output in HTML that can be displayed in a browser-like area. As the user moves, he or she can select any of those agents and access the respective information for the current location.

4.2. Experiences with the AROUND system

The realisation of the AROUND prototype has demonstrated the practical application of the concept of location-based services in developing location-dependent applications, and has allowed us to gain a better understanding of the kind of applications that location-based services can enable and of the main issues involved in such development. This section analyses some of our experiences during the development of the prototype.

As previously stated, one of the objectives of this work was to create an open architecture in which independently developed components could be put together to form a working system. Unfortunately, the current lack of established standards in a number of areas has prevented us from using truly independent components. For example, the lack of standard service types, and consequently of standard services has forced us to develop our own service types, our own services, and our own client application to make use of those services. Although we have tried to keep the development of the components independent, the evaluation of the prototype as a really open architecture may have been compromised to some extent. However, we believe this limitation to be transitional. Multiple industrial groups are now working on standards for various forms of automatic exchange of electronic data [7, 11]. These will enable an open infrastructure of information exchange, and lead the Internet to better approximate a truly service-based infrastructure. When these yet missing pieces start to be in place, the advantages of an open approach will become more and more evident.

In the development of the application, and in particular in the definition of the rules used by agents to select services, it has become clear that service location can involve complex decision rules that may demand more than simple attribute-based matching. While this work has focused on the location-based aspects of service discovery, future location-based applications could benefit from more powerful service selection interfaces, capable of addressing issues such as cost, trust, as well as *fuzzy* forms of service selection.

The development of an interactive application based on the paradigm of location-based services has raised particular issues concerning the design of the user interface. Some of them are common to context-aware applications in general [2]; e.g. how to make the user interface reflect context changes or how to combine explicitly retrieved information with context-aware information, but others were prompted directly by the use of location-based services. One of such issues has been how to present the user with its options in terms of the service selection process without going into a complex parameter questionnaire. This especially involves the representation of the space of location contexts. An adequate representation of these elements should provide users with important information about the currently available alternatives in terms of the spatial scoping of queries and is essential in allowing them to take advantage of the multiple proximity models supported by the AROUND architecture. We felt the need for metaphors that could support this interaction in an appropriate way.

Another user interface issue concerns the representation of contextual factors associated with information. In our application, the user does not directly select his sources of information, and is even unaware of which sources are being used. Furthermore, an application may combine information obtained from multiple sources and present it as a single piece of information. This prevents the user from assessing contextual factors that are normally associated with information, such as its source, the way it is presented, or where it is referenced. Without these elements, the user may fail to evaluate the background and trustworthiness of the information available on a given environment. The application should therefore be able to either represent some of these contextual factors or to give some aggregated indication on the quality of the currently available information. For example, user-defined parameters could associate higher levels of trust with certified service providers.

5. Related Work

Our work has many aspects in common with work on location-dependent applications, as we also aim to support systems that react to changes in their location. The GUIDE [4] provides visitors to the city of Lancaster with a dynamic and context-sensitive tourist guide. The increasingly available location-based services for cellular networks allow customers to access information specific to their location via their handheld devices and WAP enabled mobile phones. Systems such as these provide a valuable insight into the development of location-dependent applications. However, the approaches they use for associating network resources with location are based on strong assumptions about key aspects of computational environment, e.g. the available network or positioning technologies. These assumptions simplify the problem domain, but restrict the applicability of these approaches. What mainly distinguishes our approach is the use of the generic mechanisms of ser-

vice discovery to support the association between physical location and computational resources.

Several service location frameworks are now available. The Service Location Protocol (SLP) [5] is proposed for service location in LANs under a single administrative domain. Jini [9] is a Java-centric technology that aims to support the association of groups of autonomous devices and software components into dynamic systems in which members can share access to services. The Universal Plug and Play (UPnP) [10] technology aims to simplify the transparent interconnection of appliances, PCs and services by leveraging Internet technology. These service selection frameworks differ from this work mainly in that discovery is typically restricted to the LAN environment and based on network proximity rather than physical proximity. The Service Discovery Service (SDS) [3] is a globally-distributed architecture for wide-area service location. SDS aims to support global service discovery but does not provide specific mechanisms for associating services with location or for geographically scoping queries. It aims to support the selection of a server anywhere on the Internet, with location being just a search criterion like any other, whereas we use location as the main criterion for service selection and optimise the architecture to support this form of service discovery. The architecture proposed by Hodes in [6] separates the roles of discovery, performed by a beaconing daemon, and service querying, supported by a specific server. The indirection level supported by beacons allows network proximity to be translated into physical proximity, but the scope model does not support any form of hierarchy or relationship between scopes. We propose a concept of location-based service based on abstract scopes that can scale to large areas and a discovery process that can scale to the wide-area.

6. Conclusions

We argue that existing approaches to the development of location-dependent applications are typically vertical systems tailored for narrow application scenarios. When considering in particular the issue of allowing an application to dynamically select information that is relevant to the current location of a mobile user, we have argued that a comprehensive solution should address the challenges of heterogeneity and openness. We therefore proposed our concept of location-based service as an approach to address these issues, and have described an architectural framework for enabling such a service model in the Internet environment. The application of the framework has been evaluated in a prototype system, highlighting some of issues involved in the use of location-based services.

Our overall results suggest that location-based services can effectively provide an adequate abstraction for the development of location-dependent systems. The resulting prototype exhibits significant potential for evolution, as new network types, new information sources or new locations can be easily incorporated. The main limitation in terms of our initial objectives has been the current lack of standards, without which the potential advantages of an open approach become compromised. We believe, however, that this limitation is merely transitional and that in the near future the adoption of open standards for information exchange will allow the benefits of openness to be fully realised.

Acknowledgements

This work was carried out as part of the AROUND project (PRAXIS/P/EEI/14267/1998) and supported by grant PRAXIS XXI/BD/13853/97.

References

[1] AROUND. Around project web site, 2000. http://www.dsi.uminho.pt/get/around.

[2] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. The role of connectivity in supporting context-sensitive applications. In H.-W. Gellersen, editor, *HUC99, International Symposium on Handheld and Ubiquitous Computing*, Karlsruhe, Germany, September 27-29 1999. Published by Springer-Verlag as Lecture Notes in Computer Science (Vol. 1707).

[3] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An architecture for a secure service discovery service. In *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking - Mobicom 99*, pages 24–25, Seattle, Washington, USA, August, 15-20 1999.

[4] N. Davies, K. Cheverst, K. Mitchell, and A. Friday. Caches in the air: Disseminating tourist information in the guide system. In *Second IEEE Workshop on Mobile Computer Systems and Applications*, New Orleans, Louisiana, 25 - 26 February 1999.

[5] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. RFC 2608, June 1999.

[6] T. D. Hodes, R. Katz, E. Servan-Schreiber, and L. Rowe. Composable ad hoc mobile services for universal interaction. In *3rd ACM/IEEE International Conference on Mobile Computing and Networking - MOBICOM97*, Budapest, Hungary, 26-30 September 1997. ACM.

[7] OGC. Open GIS consortium web site, 2001. http://www.opengis.org/.

[8] OMG. Corba services: Common object services specification. Technical report, December 1998.

[9] Sun Microsystems. Jini web site, 2001. http://www.sun.com/jini/.

[10] UPnP Forum. Universal plug and play web site, 2000. http://www.upnp.org/.

[11] W3C Consortium. The W3C consortium: Mobile access web page. http://www.w3.org/Mobile/.