A Scalable Location Aware Service Platform for Mobile Applications Based on Java RMI

Olaf Droegehorn, Kirti Singh-Kurbel, Markus Franz, Roland Sorge, Rita Winkler, and Klaus David

IHP, Im Technologiepark 25, D-15236 Frankfurt (Oder), Germany
{droegehorn, singh, franz, sorge, rwinkler, david}@ihp-microelectronics.com

Abstract. In this paper, a generic service platform for supporting location aware applications is presented. The platform is based on Java to be independent of operating systems. We use RMI (Remote Method Invocation) for communication in the distributed system and Jini as an additional tool to offer and find services. The mobile client invokes methods on platform objects via RMI and offers its own remote methods to be used by the platform. To ensure the scalability of the system, central units (platform servers) are replicated in accordance with the deployment scenario. A new mechanism for the hand-over of objects, shared by clients and servers, between platform units is described. We introduce a new concept of objects and auras for efficient support of innovative location aware applications. The platform is also extended for thin clients without a JVM (Java Virtual Machine). A thin client communicates with the platform via a platform gateway using e.g. HTTP.

1. Introduction

For the future success of W-LAN and cellular systems, innovative applications and the underlying middleware platform are important building blocks. Like several other researchers, we believe that location and context based applications have a significant scope in this field of mobile computing [2]. The applications developed and investigated up to now by the research community were basically prototyped applications. On one side these applications demonstrate the potential of taking location and context awareness into account, however, on the other side, it can be seen that these applications are nontrivial to create for two reasons: First, these applications are very complex, they require special skills and knowledge of underlying communication infrastructure. Second, these applications were developed as prototypes and can be used only by few terminals but not with a mass of users.

Location and context aware applications need a common set of functions. To unify the application programming interface for all applications concerning this topic, a universal platform is needed. The platform must be generic in order to be independent of a specific communication network or positioning system. Furthermore, it is important to support a variety of applications, not restricted on a specific business model.

In the literature, several approaches for the design of a service platform have been reported. For example, in [8] a platform called MASE was designed to support mobile users. For this platform, applications like a city guide, mobile multimedia services, etc. were developed. These applications were prototypes and the issues of real world like scalability were not addressed in this project. Furthermore, features like tracking a mobile user and finding local available services were not supported by this platform.

An important aspect of a location aware service platform is keeping track of mobile objects in a scalable way [1]. The Globe project [6, 9, 10] concentrates on the development of an efficient search mechanism for mobile objects. The same issue is addressed by the Nexus project [3, 7].

In this paper, we introduce a platform architecture, which takes care of the basic communication infrastructure required for mobile wireless systems and basic functions required by a location aware application. It is designed to be flexible enough to replicate the platform server as often as necessary in order to fulfil the user requirements concerning the response time of the system. The platform itself is realized in Java RMI/Jini technology in order to be independent of a specific operating system. We introduce a concept of auras. An aura defines a space of interest for an object and can have any dimensions. Auras enable a large number of new services which are not possible by just stating the physical location of an object. Based on the aura concept, the user can register for events such as "object A has entered aura X". The support of thin clients which have limited computational capability is also considered.

The rest of the paper is organized as follows. Section 2 describes the overall architecture of the platform. In section 3, the concept of auras and objects is explained. The support of not Java enabled thin clients is introduced in section 4. Section 5 summarizes the paper and gives an outlook of future work.

2. Service Platform Architecture

Our service platform intends to integrate all functions that are required to support innovative location aware applications. The platform consists of several components responsible for special functions. The platform itself is based on Java RMI/Jini technology in order to be independent of the underlying operating system. Platform server units can be replicated as often as required to fulfil the user requirements concerning the system's response time.

We decided to use Java RMI and not CORBA as the communication mechanism, since CORBA seems to be less suitable for mobile terminals due to the complexity of a CORBA client. Moreover, we are developing the system from the scratch and no legacy systems have to be taken into account. We utilize RMI to transfer Java specific objects between platform servers to avoid the transformation of object types.

The service platform operates at two different locations, one part on the server side and the other on the mobile terminal which is, in our case, the client. The part of the platform, which resides on one or several platform servers implements the desired functionality for each component. The structure of a platform server is shown in Fig. 1.

The location management component deals with location awareness and provides all necessary functions concerning this topic. For example, the application can inquire the location of an object. This object can be a logical element defined by the application, e.g. a service available only at specific locations or just another mobile terminal. To provide these location based functions the platform needs to handle several things. On the first hand it must use multiple databases to store the locations of the desired objects. It is important that this subsystem is implemented very efficiently because this database can be very large for cellular systems.

To get informed about the object's location the platform uses a so called sighting proxy. This proxy gets notifications from an underlying sighting system (location system using e.g. cell ID, some form of triangulation or GPS). This system can be implemented by the network provider or by another service provider.

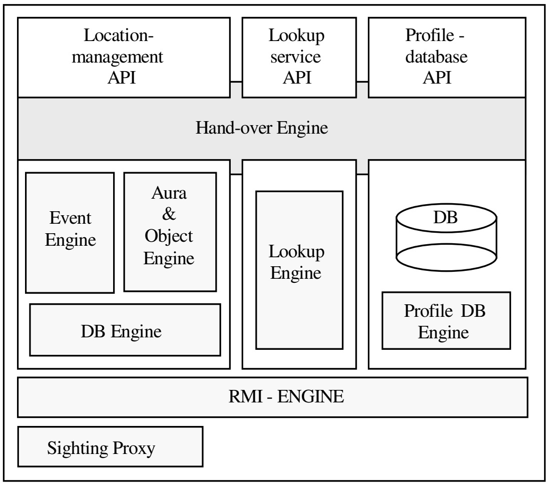

Fig. 1. Architecture of a platform server.

The DB engine has to filter all the sighting notifications in order to reduce the amount of information flow. The DB engine forwards not all sighting updates which it receives from the sighting proxy, but only the first sighting of an object at a particular server and sighting updates for objects on which an event is registered. For example, a user who wants to be notified if he is near a particular moving object, the positions of the objects must be observed. To do this in an efficient way the event engine registers with the DB engine to get all sighting updates for the objects involved in the event. The event engine monitors the positions of the objects and forwards only relevant notifications to other subsystems.

To perform user queries, the engine for auras and objects uses rendering algorithms. The goal is to find out which objects are located in which aura and to determine which auras intersect.

For using a lookup service, the platform supports queries on the Jini lookup system [4, 5]. Users can register their own services with this lookup system or can just use services offered by other devices. The lookup engine at the server side handles the queries and sends appropriate objects to the client.

The profile database is responsible for the support of personalization. Each mobile terminal can give a profile to the platform which contains some common properties of the terminal and the user. For example, security related information can be stored in this profile. Furthermore, the hardware and software capabilities of a specific terminal can be described inside this profile database. All processes which will be handled by the platform in an automatic manner need information about preferences of objects. These preferences are also stored in the profile component. The main engine for this profile database resides on the server and realizes an internal distributed database that is virtually available on each platform server.

To ensure the scalability of our system, the platform unit shown in Fig. 1 is replicated to distribute the workload among a number of servers. The communication between the platform servers is based on hierarchical structures.

The platform must be able to offer the right object references on the right platform server to the mobile client. Because of efficiency reasons not all objects will be available at all locations by definition. The platform has to move several objects from one location to another. This is done in the background by the platform and is not visible to the user or even the platform client. When a client communicates with the platform, the platform chooses the server for the communication with the particular client. For the communication, the server holds remote references of objects residing on the client and vice versa. All the data related to the client (auras, profile, registered events) as well as the remote references of the client's objects are stored at the related platform unit. When the client changes its location and the platform realizes that the client is now in the responsibility range of a new platform unit, a hand-over of the client's remote references and data from the old platform unit to the new one is required. The new platform unit requests the client's data from its neighbored platform units. The old platform unit which holds the data replies by transferring the client's remote references and all other client related data to the new platform unit. The server at the new platform unit registers itself with the client using the clients remote references. The new platform unit overwrites the RMI references of the old platform unit at the client with its own.

The part of the platform running on a client is very small because of the limited computational resources on the wireless terminal. It implements just a proxy for the platform functions which are available on the server side. The application can use well defined API's provided by the components of the platform.

3. Objects and Auras

In the literature, typical approaches concerning location awareness determine just the position of objects or users. This is done by conventional positioning systems or on the basis of the communication infrastructure. That leads in most cases to a model of the real world which deals with mobile terminals and base stations of the wireless system as physical objects and the area around each base station as the space where a mobile terminal can be seen. The main disadvantage of this approach is the dependence of the abstract model inside the service platform on the hardware infrastructure used. If an application designer wants to define objects or areas of interests independent of the used hardware it is mostly impossible to design this model into the service architecture.

For this reason, we propose a very flexible concept of objects and auras. An aura can have any dimensions and defines a space of interest for an object. Each object can have several auras, which are totally independent of each other. An object can be a physical entity like a printer or a mobile terminal. Additionally, we include logical objects in our platform. A service, for example, which is only available in a specific room is a logical object having the aura of this room. For instance, a mobile client can request all objects whose auras overlap with its own aura.

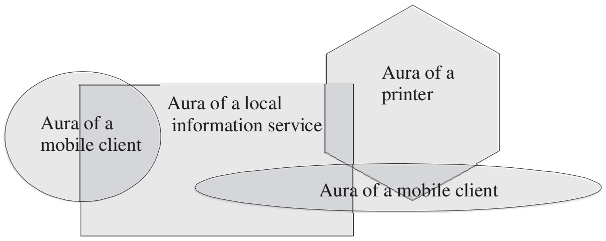

Fig. 2. Aura concept. For each object one or more auras can be defined.

Based on this approach it is the task of rendering algorithms to check if an object or an aura crosses another aura. Which properties these logical objects can have is not specified inside the service platform but in the definition of each application.

With this aura based location aware platform, we are in the position to create a whole range of innovative applications. For example, person A can get informed about the event that person B has approached him by a certain distance (B has entered an aura of A). Upon this, several useful actions can be triggered, like automatic finding of a free meeting room, ordering some coffee, etc.

4. Enhancement for Thin Clients

Since our platform is based on Java and the Java RMI communication mechanism, any client which wants to use a service offered by the platform must have a JVM running on it. Thin terminals like mobile phones or other devices which do not have a JVM cannot use such a service directly. To support such clients, we have extended the infrastructure mentioned in the previous section.

In this enhanced system, we use a mediator which converts messages between a platform server and the thin client. For the platform server side communication, this mediator acts as a regular Java enabled client similar to other mobile clients, which requests objects and services. To communicate with the thin terminal, the mediator uses a common access protocol such as HTTP or WAP. The data received from the platform server are translated into HTML or WML and are sent to the thin client. In this case, the thin client needs only to be able to access the web. It can request HTML pages that meet its service requirements. The mediator is responsible for the correct translation of the messages.

5. Summary and Outlook

In this paper we have introduced a service platform based on JavaRMI/Jini for supporting location and context aware applications for mobile users. Scalability is achieved by replication of server units and a hand-over mechanism for object's references and object related data. The platform can be used for indoor applications as well as for outdoor applications. It is independent of both the operating system as well as the positioning system that provides the location information. An aura concept is used to define regions of interests for the objects used. This enables innovative applications features such as triggering events based on the relative distance of two objects. The platform also provides a lookup service where terminals can offer their services to other people and also can use services offered by other users. Furthermore, the enhancement of the platform for not Java enabled thin clients has been pointed out.
By the end of this year the platform kernel will be implemented. Later, we will develop new location aware applications based on this platform to test its performance and efficiency. These applications will be tested considering a large number of clients. The performance regarding the hand-over of remote references and efficient database searching will be studied with help of field trials and previously performed simulations.

Reference

1. Black A., Artsy, Y.: Implementing Location Independent Invocation. IEEE Trans. On Par. Distr. Syst. Vol. 1 (1), (Jan. 1990) 107-119
2. Brown, P.J., Bovey, J.D., Chen, X.: Context-Aware Applications: From the Laboratory to the Marketplace, IEEE Personal Communications, (October 1997), 58-64
3. Hohl. F., Kubach, U., Leonhardi, A., Schwehm, M., Rothermel, K.: Nexus: An open global infrastructure for spatial-aware applications. Proceedings of the fifth International conference on Mobile Computing and Networking (MobiCom 99), ACM Press (1999)
4.http://developer.java.sun.com/developer/technicalArticles/ConsumerProducts/jinicomm/unity
5. http://jini.org/
6. http://www.cs.vu.nl/~steen/globe/
7 Leonhardi, A., Kubach, U.: An architecture for a distributed universal location service. Proceedings of the European Wireless '99 Conference, Munich, Germany, ITG Fachbericht, VDE Verlag, (1999) 351-355
8. Keller, B., Park, A.S., Meggers, J., Forsgern, G., Kovacs, E., Rosinus, M.: UMTS: A Middleware Architecture and Mobile API Approach. IEEE Personal Communications, (April 1998), 32-38
9. van Steen, M., Hauck, F.J., Homburg, P., Tanenbaum, A.S.: Locating Objects in Wide-Area Systems. IEEE Communication Magazine, (January 1998), 104-109
10. van Steen, M., Tanenbaum, A.S., Kuz, I., Sips, H.J.: A Scalable Middleware Solution for Advanced Wide-Area Web Services. In: Sips. Distributed Systems Engineering, Vol. 6 (1), (March 1999) 34-42