

Architecture of a Large-scale Location Service

Alexander Leonhardi and Kurt Rothermel

University of Stuttgart

Institute of Parallel and Distributed High-Performance Systems

Breitwiesenstr. 20-22, 70565 Stuttgart

e-mail: {leonhardi, rothermel}@informatik.uni-stuttgart.de

Abstract

Location-aware services are a promising way of exploiting the special possibilities created by ubiquitous mobile devices and wireless communication. Advanced location-aware applications will require highly accurate information about the geographic location of mobile objects and functionality that goes beyond simply querying the user's position, for example determining all mobile objects inside a certain geographic area. In this paper, we propose a generic large-scale location service, which has been designed with the goal of managing the highly dynamic location information for a large number of mobile objects, thus providing a common infrastructure that can be employed by location-aware applications. We propose a hierarchical distributed architecture, which can efficiently process these queries in a scalable way. To be able to deal with the frequent updates and queries resulting from highly dynamic location information, we propose a data storage component, which makes use of a main memory database.

1. Introduction

Location-aware services provide the basis for a wide range of promising application areas (see, for example, [1]). In order to support a wide range of location-aware applications, a generic location service (LS) is needed that maintains the geographic locations of tracked objects, like persons or vehicles (see also [3]). While for some applications it might be sufficient to retrieve the current position of a given object (*position query*), others require more sophisticated types of queries. A *range query* determines all mobile objects that are inside a certain geographic area. A *nearest neighbor query* the mobile object nearest to a certain location. For example, in a city guide application an information service for public transportation might want to announce the delay of a bus to all users waiting at the next station. In consequence, a user may want to find the nearest available taxi.

Position sensing devices differ in the accuracy with which they record location information. While GPS is accurate to within 10 m, an indoor positioning system (e.g.,

Active Bat) might have a finer resolution. Consequently, the location information maintained by a LS may differ in accuracy, if it is provided by various sensor systems. When designing an API and query processing algorithms for the LS this fact must be considered to allow clients and tracked objects to specify the requested accuracy. Moreover, different positioning systems can deliver the same information but in different forms. For example, an Active Badge system delivers the position by means of cell identities, while GPS is based on a geographic coordinate system. A LS implementation should hide this heterogeneity as far as possible from applications. Protecting the privacy of the tracked object's recorded location information will be crucial for the acceptance of such a service. Therefore, authentication and authorization mechanisms must be integrated into a LS (for a discussion of this aspect see [3]).

We believe that the location-awareness of applications will not be limited to small-scale indoor or outdoor situations but will become a global issue, which has to be supported by a LS by integrating various indoor and outdoor positioning systems. We further believe that many applications will become location-aware as soon as the required infrastructure gets available. Consequently, we expect that a global LS must be able to handle hundred thousands of tracked objects and clients concurrently.

In this paper, we sketch the architecture and mechanisms of the LS. For further details the interested reader is referred to [2].

2. Architecture

To ensure the scalability required for a large-scale deployment of the service, the location servers that belong to the LS are organized in a hierarchical manner comparable to the architectures proposed for the location management in Personal Communications Services or the GLOBE location service for mobile software objects.

The LS is configured to cover a certain geographic area, its so-called *service area*. Objects that are located within its service area can be registered with the LS to be tracked. The service area covered by a LS is structured in a hierarchical fashion: A service area can be subdivided into sub service areas, which again can be subdivided, and so on

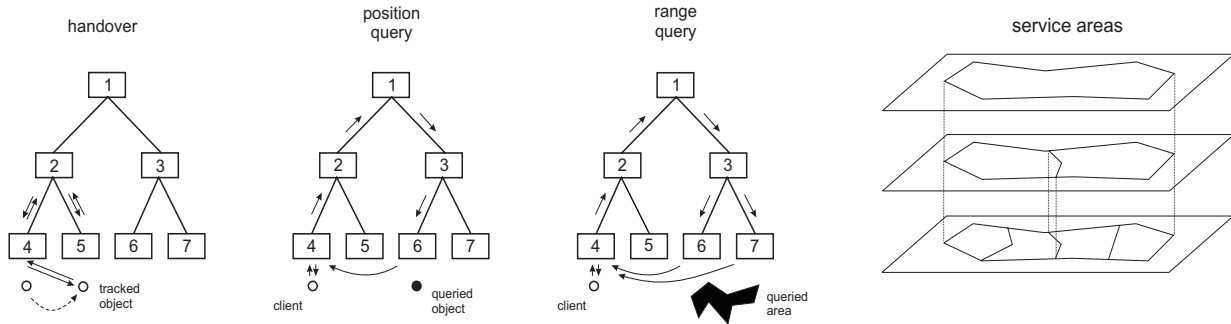


Figure 1: Basic algorithms of the LS.

(see Fig. 1).

Associated with each service area is a *location server*, which is responsible for tracking all the objects visiting its service area. Consequently, the service area hierarchy resembles the location server hierarchy. *Leaf servers* are associated with leaf service areas. Since leaf service areas are not supposed to overlap, at any point in time for each tracked object there exists exactly one leaf server that is responsible for keeping track of the object's position. We will call this leaf server the object's *agent*. Of course, whenever a tracked object moves from one service area to another a hand-over has to be performed. A *non-leaf server* is responsible for a service area that is the union of the service areas associated with its child servers. A non-leaf server stores for all tracked objects that are currently within its service area a so-called *forwarding reference*, which identifies the child server that is responsible for the mobile object. Consequently, only leaf location servers store the location information for a mobile object, whereas non-leaf servers store forwarding references that specify a path from the root server to the object's agent, which manages its location information.

Location updates for a tracked object are always sent to the object's agent, which then updates the object's location information. When a tracked object moves to a new service area, the tracking responsibility is handed over and the forwarding path has to be adapted accordingly. Hand-over processing must then make sure that the tracked object learns about its new agent.

A client of the LS sends a query request to a nearby leaf location server, which becomes the *entry server* for this query. To perform a position query, the query is forwarded upwards until a server is found that holds an appropriate forwarding reference and by then following it downwards. For a range query, the hierarchical relationship of the service areas is exploited by propagating the query upwards until a server is found, whose service area includes the specified area entirely. From there the request is propagated downwards to all children with

service areas overlapping with the specified area. These leaf servers determine the objects that are in the corresponding sub area and send the result back to the entry server, which is responsible for constructing the answer that is then returned to the client. Nearest neighbor queries are performed in a similar fashion.

3. Data Storage

To achieve a high accuracy of the location information, the LS needs to be able to cope with frequent update messages. Our experiments show that existing spatial database systems are optimized for more complicated queries and large data sets and are not suited for the large number of simple updates required for the LS. We therefore propose a combined data storage component, where the location information is held in a fast volatile main memory database, while registration information is stored in a traditional database. In case of a failure, the location information is recovered on the fly from the incoming updates.

Our experiments with a prototype implementation of the LS show that our distributed architecture is able to process more than thousand update message per second and almost as many queries on a single location server (for details see [2]).

Reference

- [1] K. Cheverst, N. Davies, K. Mitchell and A. Friday: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project, in *Proc. of the 6th Int. Conf. on Mobile Computing and Networking (MobiCom 2000)*, pp. 20-31, 2000.
- [2] A. Leonhardi and K. Rothermel: *Architecture of a Large-scale Location Service*, Technical Report TR-2001-01, Faculty of Computer Science, University of Stuttgart, Germany, 2001.
- [3] U. Leonhardt: *Supporting Location-Awareness in Open Distributed Systems*, PhD-thesis, Imperial College of Science, Technology and Medicine, University of London, 1998.