From Location to Position Management: User Tracking for Location-based Services

Axel Küpper and Georg Treu

Mobile and Distributed Systems Group, Institute for Informatics
University of Munich, Germany
[axel.kuepper|georg.treu]@ifi.lmu.de

Abstract: This paper compares traditional location management, which focuses on tracking mobile subscribers in the topology of a cellular network, with position management, which we define to be a set of functions for tracking mobile targets in terms of geographic positions as needed for LBSs. As the air interface is the most limited resource in a mobile network, location management has been optimized with regard to signaling overhead caused by updating location data (and paging) between the terminal and the network. Position management is exposed to the same limitations, but, on the other hand, imposes much stronger requirements with regard to accuracy of position data as well as flexibility of tracking. This paper presents an architecture for tailoring the tracking process according to the special LBS requirements on the one hand and limiting the resulting update traffic at the air interface on the other. The architecture is shared between mobile terminals and an application server, for which we present first details of our implementation.

1 Introduction

Location-based Services (LBSs) combine the provisioning and processing of mobile services with the geographic position of their users. The crucial point when realizing LBSs is obviously the positioning of targets, where a target may be the LBS user himself or any other entity of the real-world connected to the service, e.g. members of a community service or vehicles for supporting fleet management (or for collecting tolls, if it works). Initially, operators of cellular networks saw LBSs merely as a facility for generating new revenues by reusing available location data collected for location management. The term ”location” is here related to the network topology in terms of sub-networks, location areas (LAs), and radio cells.

However, it soon became apparent that this type of location is difficult to map onto geographic positions and is very inaccurate compared to accuracy requirements of LBSs. Therefore, vendors and operators are currently discussing the pros and cons of introducing advanced positioning methods in cellular networks like *Differential GPS* (D-GPS) or *Enhanced Observed Time Difference* (E-OTD) (see [Ku05] for an overview), which are are terminal-based, i.e., the *mobile terminal* (MT) performs self-positioning. This is in

analogy to location management, where MTs sense broadcast emissions to determine their current LA.

In general, the drawback of terminal-based positioning is that the MT knows where it is, but the *LBS application server* (LBAS) in the fixed network does not. In cellular location management, this problem is solved in that the MT reports its LA to the network by using the *location update* (LU) procedure. Basically, the same principle can be applied for transferring a geographic position obtained by an MT to an LBAS. However, the general problem is that each update burdens the air interface, which is the most valuable resource in a mobile network. In the past, considerable efforts have been made to reduce the associated signaling overhead, and the developed solutions work fine for locating users when network-originated calls or data are to be delivered. However, LBSs have much stronger demands on updating strategies, especially with regard to accuracy and availability of position data, and, even worse, these demands are very different for the various types of LBS applications.

In this paper, we introduce a concept for position management, which differs from the traditional location management in that it deals with all aspects of obtaining and managing geographic position data instead of topological network locations. We identify positioning, updating, storage of position data, and privacy aspects as the key functions of position management, but in this paper only deal with updating aspects. We present an approach for dynamically configuring basic updating strategies, which can be composed to more sophisticated, high-level strategies. In this way, it is possible to tailor the updating process with regard to the particular requirements of an LBS application on the one hand, and saving air interface resources on the other. Thus, position management is a key concept for enabling sophisticated LBSs, which require a permanent tracking of targets and which go far beyond the simple and boring restaurant finder applications available today.

The following section presents an overview of different update strategies and discusses their pros and cons for location and position management. Section 3 motivates our approach of position management by introducing different application scenarios. The architecture is highlighted in Section 4, followed by a short description of implementation details in Section 5. Section 6 concludes the paper with an outlook on further work.

2 Overview of Updating Strategies

A broad range of update strategies have been proposed for both location management, see for example [Mu03], and for user tracking based on geographic positions [LeRo02]. For the purposes of LBSs in cellular networks, these strategies are applied between a *Gateway Mobile Location Center* (GMLC), which is a central node in the fixed network domain, and an external LBAS, see [3GPP]. The update process between these nodes is controlled by the *Mobile Location Protocol* (MLP) [LIF]. However, the process for exchanging position data between MT and GMLC is applicable for 3GPP-based networks only and is not suited to be tailored according to the special requirements of a particular LBS.

Our approach, on the other hand, focuses on bypassing the GMLC in that updating is organized directly between MT and LBAS. Thus, our approach is also independent of a particular (type of) network and serves also very well, for example, if the MT is connected via WLAN. For our approach, we have adopted the following strategies, which can be configured by one or several parameters and applied in conjunction:

- **Query strategy.** In a querying strategy, the LBAS requests the position from the MT on demand.
- **Piggybacking.** Position data is included in a service request that is passed from the MT to an LBAS.
- **Immediate strategy.** An immediate update is triggered each time the position changes with regard to the last reported position.
- **Periodic strategy.** A periodic update is triggered if a pre-defined time interval has elapsed since the last update.
- **Distance-based strategy.** For this strategy, the MT always determines the distance between the current and the last reported position. If this distance exceeds a pre-defined threshold, it performs an update.
- **Zone-based strategy.** An update is initialized if the target enters or leaves a pre-defined zone, where a zone can be fixed as a single point, a circle or ellipse, a line, or a polygon.

The traditional location management uses a combination of the immediate and the periodic strategy, which is implemented in each MT. An immediate LU occurs when the MT enters a new LA in order to inform the network about where to page a subscriber when network-originated calls or data are to be delivered. The average frequency of immediate LUs depends on the number of radio cells per LA, but also on the geometry of LAs with regard to infrastructures like cities or highways as well as to the call and mobility behavior of subscribers. LAs must be arranged in a way that overhead caused by LUs on the one hand and paging on the other are in balance. A periodic LU is implemented as a fallback solution in case of database failures. The time-interval between LUs can be configured by the operator and reported to the MTs on a broadcast channel. The distance and zone-based strategies are not applicable to location management, as they can only be used for geographic positions.

The approach followed by location management is obviously not suited for position management. For example, it is certainly not practicable to follow an immediate strategy if the MT obtains its position with an in-built GPS receiver. Each time the target slightly moves, an update would be triggered, resulting in a nearly continuous data stream passed over the air interface when the target is on the move. Conversely, a periodic approach would waste valuable resources if the target does not move at all. Furthermore, it is also not useful, for example, to solely implement a distance-based strategy in the MT, because some LBS applications may not be interested in the distance covered by the target, but only on whether or not it has entered or left a certain zone, for example a building. Other LBS applications,

on the other hand, want to continuously track the target, but with different resolutions, for example 1 km for fleet management and 100 m for location-based gaming.

Due to the diversity of requirements imposed by different LBSs, it is useful to implement all of the strategies sketched before in the MT and make them dynamically configurable by the LBS application server. Before explaining our approach of dynamic position management in detail, a further motivation is given in the next section by introducing different application scenarios we focus on.

3 LBS Application Scenarios

It is very hard to give an overall classification of LBSs from which technical decision can be derived. A well-known distinction is made between *reactive* and *proactive LBSs*, see for example [FiMe02] and [Ku03]. Reactive services are characterized by the fact that they are explicitly invoked by the user, while proactive services are automatically triggered if a predefined event occurs, for example if a target enters a certain zone and the LBS user wants to be notified about that. Furthermore, we want to distinguish between *user self-centric* and *other-centric LBSs*. In the former category, user and target are the same individual, i.e., the user's own position is processed by the application, while in the latter both entities are different. The combination of both classification criteria leads to four different classes of LBSs we want to consider in the following.

In today's networks, almost all LBSs are reactive and user self-centric. The user initializes, for example, a WAP session in order to request a list of nearby *points of interests* (PoIs) like restaurants or automated tellers. If the terminal is able to perform positioning, it is obvious to use the piggybacking strategy, i.e., to send position data together with the service request to the application server. Another class focusses on other-centric reactive LBSs. The most prominent example is the buddy finder, where users' can request the position of their buddies (assuming that each buddy agrees on that). If the user invokes this service, the LBAS applies the querying strategy for collecting the positions of all targets. Querying is often applied in conjunction with caching, where the last received position is processed instead of requesting for an update.

Proactive LBSs are scarcely available today, one reason for that being certainly the missing of an efficient position management (and actually the approach presented here was primarily motivated by the wish to realize proactive LBSs, see also [Ku03]). An example of a self-centric proactive LBS is a tourist guide that automatically alerts the user if he is in close vicinity to landmarks. For efficient updating, the landmarks might be represented as zones, for example polygons with coordinates as edges, and passed to the MT. Each time the user enters a zone, the MT initializes an update to the LBAS, which then transmits information on the respective landmark to the user. A typical example of an other-centric proactive LBS is a community service, which automatically notifies a member as soon as another member approaches, for example when the distance between them falls below a pre-defined threshold. Consequently, a distance-based updating strategy might be applied here.

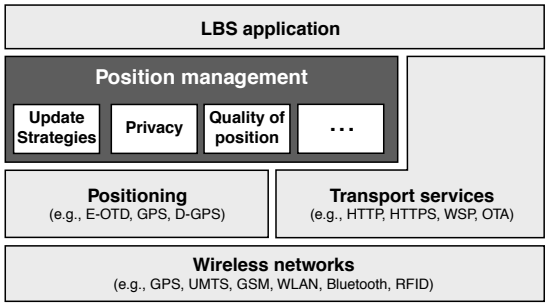

Figure 1: Position Management Overview

Finally, consider a scenario where a professor wants to track his students, which is an other-centric proactive LBS. Since these services are very sensitive with regard to privacy, the students agree on tracking at the university campus only. In this case, a zone-based strategy is used to detect when a student enters the campus. If so, the LBAS re-configures the MT for periodic or distance-based updating, which is executed as long as the student stays on the campus. This is an example of a high-level strategy where different basic strategies are combined and dynamically re-configured.

4 Architecture

From an architectural point of view, the scenarios presented before pose a number of requirements on position management. First, the dynamic (re-)configuration of strategies on an MT as well as direct position requests to an MT must be supported. Second, besides a general mechanism for privacy and integrity of position data, a tracked target must be able to selectively decide about who may access his position data and when. Third, the system must support multiple position technologies in a transparent way. Different environments require the use of different position technologies (e.g., GPS for outdoor and RFIDs for indoor positioning), while these details should be hidden from an LBAS. We also consider the negotiability of quality of position data in terms of accuracy, age, etc. . Figure 1 shows a sketch of the envisioned architecture. In this paper, we have the focus on the different update strategies and their dynamic composition. The other identified aspects remain subject to future work.

Figure 2 outlines the components involved in the strategy management as well as their interactions. An LBAS uses a *position gateway* to dynamically configure an MT with one or a combination of the update strategies discussed above. The gateway is either realized as a software library, or it is installed on an independent host and works as a proxy. On the MT, there is a component called *position monitor*. It interfaces with the tracking system of the MT and communicates with the gateway. In the simple query strategy (1) as well as in the Piggybacking approach (2), the monitor only needs simple forwarding capabilities

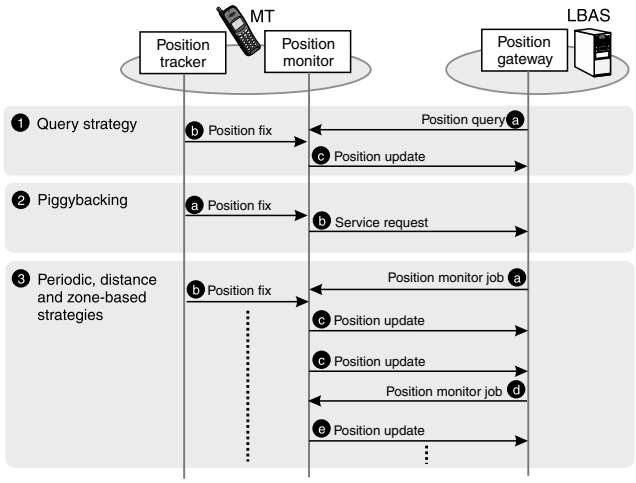

Figure 2: Sequence Diagram

(1c, 2b). For periodic, distance- and zone-based strategies more intelligence is necessary. Therefore, we introduce the concept of a *monitor job* which represents an update strategy. A monitor job can be dynamically placed on (3a) and removed from an MT. The removal is either done explicitly by the gateway or implicitly when the time scope of the job is exceeded. Furthermore, in order to save signaling overhead jobs already registered with an MT can be dynamically updated (3d), activated and deactivated.

A monitor job carries all information necessary for deciding whether or not a position fix results in an update to the gateway. Therefore, all jobs contain start and end time. In addition, periodic jobs specify the update periodicity and distance-based jobs carry the critical distance that must be covered by the MT with respect to the last update. Finally, zone-based monitor jobs include the representation of a geographical shape against which position fixes can be checked for containment so that updates are generated on entering or leaving a zone. The position monitor continuously evaluates the set of registered monitor jobs (3a, d) against position fixes delivered by the tracking system (3b) and then decides whether an update (3c) is to be sent to the gateway.

5 Implementation

The basic configuration of our implementation consists of an Ipaq PDA connected to a GPS receiver via Bluetooth. Connectivity is provided via GPRS or WLAN, and the position tracker and monitor are realized with J2ME. We started implementing the most fundamental components of the system, which are the dynamic placement and removal of monitor jobs, a secured transport service as well as a module for MT-based positioning.

For representing monitor jobs as well as position updates, we incorporate the *Mobile Location Protocol* (MLP) [LIF]. Initially, it has been designed for the exchange of position data between an LBAS and the fixed GMLC. But its fundamental protocol elements can be applied to terminal-based position management as well. MLP specifies services equivalent to monitor jobs by XML DTDs. They provide means to uniquely reference MTs, specify time constraints on the positioning process and select the delivered quality of position as well as the geographic reference system. Although currently MLP only supports immediate and periodic position updating, it is possible to use its built-in extension mechanism to represent all of the update strategies. For instance, with the *Shape Element Definition DTD* in MLP, it is possible to define geographical shapes like points, ellipses, or polygons. To represent a monitor job of the zone-based strategy, we simply include the start and stop time of the job as well as a shape definition, which are standard elements of MLP. Furthermore, we introduce a trigger criterion to determine if an update must be sent on entering or leaving a zone, or both. The respective XML document is shown below.

```
<zone_based_job>
  <start_time utc_off="-0100">20050330154500</start_time>
  <stop_time utc_off="-0100">20050330184500</stop_time>
  <trigger_criterion><entering/></trigger_criterion>
  <trigger_criterion><leaving/></trigger_criterion>
  <shape>
        <CircularArea srsName="www.epsg.org#4326">
                <coord>
                        <X>48 08 57.200N</X>
                        <Y>11 35 47.600E</Y>
                </coord>
                <radius>15</radius>
        </CircularArea>
  </shape>
</zone_based_job>
```

In contrast to MLP where HTTPS is used as transport protocol in both directions, in the terminal-based approach, it is hard to install a respective server on a resource-restricted MT. Therefore, we rely on one secured socket connection (SSL) that is kept open in idle times and that can be used for data push in both directions. For this purpose we use the J2ME MIDP 2.0 API that features client side SSL.
The access to position data on an MT has been standardized by the *Java Location API* [JSR179]. With the API, a positioning method can be chosen from the available technologies on an MT by abstract criteria like accuracy, response time or power consumption. In our implementation, the position monitor implements the interface of a so called *LocationListener* and registers with the API. The position fixes passed out by the API are continuously evaluated against the current set of monitor jobs and position updates are sent to the gateway when necessary. Although the Location API is well suited for technologies like GPS that deliver spatial coordinates, we are aware of the fact that other positioning technologies like e.g. RFIDs are more likely to support symbolic position descriptions only and thus cannot be used with the Location API that fixes WGS-84 as a reference system. With this restriction in mind, the Location API is integrated as one possible source of position data for the position monitor.

6 Conclusion

Our approach of dynamically composable update strategies for MTs is promising. We consider it as highly flexible yet economical w.r.t. the air interface and therefore suited for the specific demands of future LBS.

The composal of higher-level update strategies from basic strategies is an open and interesting issue, especially for services that interrelate the positions of various tracked entities, like community services. For instance, a *buddy tracking* [Am04] service notifies registered users when their distance falls below a specified threshold. A higher-level update strategy for this purpose could be realized by dynamically placing distance-based monitor jobs on the users' MTs in a way that minimizes the number of updates.

Finally, the incorporation of functions such as position privacy management, accounting, etc. into the proposed architecture will be an important issue of future work.

References

[3GPP] 3GPP TS 23.271 *Functional Stage 2 Description of LCS*.

[Am04] Amir, A., A. Efrat, J. Myllymaki, L. Palaniappan, K. Wampler. Buddy Tracking — Efficient Proximity Detection among Mobile Friends. *Proceedings of IEEE Infocom 2004*, Hongkong, March 2004.

[FiMe02] Fischmeister, S., G. Menkhaus. The Dilemma of Cell-based Proactive Location-Aware Services. *Technical Report TR-C042*, Software Research Lab, University of Constance, 2002

[JSR179] JSR 179 Expert Group. *Location API for Java 2 Micro Edition*. Version 1.0

[Ku03] Küpper, A., F. Fuchs, M. Schiffers, T. Buchholz. Supporting Proactive Location-Aware Services in Cellular Networks. *Proceedings of the 8th IFIP-TC6 International Conference on Personal Wireless Communications (PWC 2003)*, Vencie Italy, September 2003, Springer-Verlag LNCS, 349–363.

[Ku05] Küpper, A. *LBS — Fundamentals and Operation*. To be published. John Wiley & Sons, 2005.

[LeRo02] Leonhardi, A., K. Rothermel. Protocols for Updating Highly Accurate Location Information. A. Behcet (Ed.). *Geographic Location in the Internet*. Kluwer Academic Publishers, 111–141.

[Mu03] Mukherjee, A., S. Bandyopadhyay, D. Saha. *Location Management and Routing in Mobile Wireless Networks*. Artech House Publishers, 2003.

[LIF] Location Interoperability Forum. *Mobile Location Protocol Specification*. Version 3.0.0