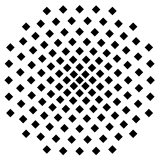

Universität Stuttgart
Fakultät Informatik

Analysis of Distribution Schemes for the Management of Location Information

Authors:
Dipl.-Inf. Uwe Kubach
Dipl.-Inf. Alexander Leonhardi
Prof. Dr. K. Rothermel
Dr. Markus Schwehm

Institut für Parallele und Verteilte
Höchstleistungsrechner (IPVR)
Fakultät Informatik
Universität Stuttgart
Breitwiesenstr. 20 - 22
D-70565 Stuttgart

Analysis of Distribution Schemes for the Management of Location Information

U. Kubach, A. Leonhardi,
K.Rothermel, M. Schwehm

Bericht 1999/01
Januar 1999

Abstract

New applications in the area of mobile computing make heavy use of knowledge about the application's run-time environment. Applications running on mobile devices in particular exploit knowledge about their current geographical position or query for the location of other interesting objects. To manage such queries some applications provide a location service specifically tailored for their needs. The efficient and application-independent handling of such queries calls for a global and universal location service. Considering a large number of users and queries to be handled, a distributed implementation of a location service is necessary. This paper analyses three schemes for the partitioning of location information and derives a performance model for these partitioning schemes. Finally, an example for the application of the analysis' results is presented for a universal location service within the Nexus system, an infrastructure for location aware mobile computing.

1 Introduction

Most research in the area of Mobile Computing has so far been concerned with getting existing networking applications to run efficiently on mobile devices and wireless communication networks and to hide the side-effects of mobility from the user. A new type of application in the context of mobile computing makes heavy use of the mobility itself by taking into account the position of the mobile devices for selecting location-dependent information or for triggering certain actions. Such applications that, besides being aware of the position of the device they are running on, are possibly also aware of the position of other objects and other aspects of the user's environment, are called location- or context-aware (SCHILIT 1995, BEADLE *et al.* 1997). Examples of location-aware applications are car navigation and fleet management systems as well as mobile city guides (PFEIFER *et al.* 1998) and tools for locating people in an intelligent office environment (HARTER and HOPPER 1994). The component that is responsible for the management of the storage and updating of location information is called a location service.

Specialized location services usually only support a certain granularity of location information and a limited set of queries tailored for a particular application. A single *universal location service*, as proposed by LEONHARDT (1998) and to some degree by MAAß (1997), which is independent of the application and positioning technology, could support the development of location-aware applications and use resources much more effectively. Such a service would have to support the following two basic types of queries which are the building blocks for more complex queries:

- A *location-of-object-query* (LOO-query) takes as parameter the identifier of a given mobile object and returns the current position of this object.
- The *objects-at-location-query* (OAL-query) takes as parameter the description of a given location and returns the identifiers and positions of all objects currently at this location.

Within the Nexus project (NEXUS 1998) we plan to design an infrastructure to support location-aware mobile computing and spatial oriented information access. One building block of such an infrastructure, which is intended for a global, possibly world-wide scope, is a universal location service. The location service will be used for the management of mobile objects, users and information. As this service obviously can not be realized by a single centralized component in an efficient way, this paper discusses different alternatives for distributing a location service. The application area, the characteristics of accesses and of the movements of tracked objects as well as special requirements, can have a big influence on the design of a location service and lead to very different architectures.

However, the influence of these parameters on the distribution of a location service is not fully understood yet. LEONHARDT (1998) describes some basic alternatives of distribution but does not indicate when to use which alternative. The research of Location Management (e.g. JANNINK *et al.* 1996) in this area is only applicable to cellular phone networks. In this paper we introduce a taxonomy, by which the distribution of existing location service architectures can be classified. Based on this taxonomy we present an analysis which calculates the load on the underlying network. In this analysis it is examined how the main alternatives perform for position updates and a given mix of location-of-object-queries and objects-at-location-queries. Thus important parameters are identified and their effect on the performance of a certain architecture is illustrated. The results of the analysis can be used to evaluate an existing architecture as well as to support certain design decisions.

The remainder of the paper is organized as follows: after a introduction to our model of the infrastructure of a location service and its sensor technology in Section 2, we describe possible alternatives for distributing a location service in Section 3. Section 4 examines and classifies related work according to our taxonomy. In Section 5 we present an analysis of the load caused by different alternatives for distribution and use the results for a general comparison of the alternatives. Section 6 shows how the results of the analysis can be used to derive the architecture of a location service from given requirements. Finally, Section 7 contains a short summary of this paper and presents our plans for future work.

2 System Model

In this section we introduce the system model underlying the subsequent considerations. It consists of four kinds of entities: client systems (CS), server systems (SS), tracked objects (TO), and locators (TS, PS) (see Figure 1).

Although we do not exclude stationary *client systems* from our considerations, our focus is on mobile client systems, e.g. Personal Digital Assistants or subnotebooks. Such systems typically have a low computational power and memory capacity. Furthermore, we assume that mobile client systems communicate with the location service over wireless networks, which usually have a low bandwidth. The location service itself is implemented by one or more *server systems*. If there is more than one server, these servers are connected over a wired network.

The location service determines the location of mobile objects through *locators.* Such a locator may be a sensor based tracking system (TS) like the Active Badge technology (WANT *et al.* 1991) or a positioning system (PS), e.g. GPS, DGPS or GLONASS (HOFMANN-WELLENHOF *et al.* 1997). Since in tracking systems a location is described

by an area identifier, e.g. a room number, and in positioning systems by coordinates, we have to handle both forms of location description. While tracking systems have a limited coverage area and track a limited number of objects, positioning systems might have global coverage and determine only the position of the object they are attached to. A *tracked object* is an object that is observed by the location service. Tracked objects are mobile, i.e. these objects change their location over time. As we consider a wide variety of tracked objects ranging from fast moving objects, e.g. planes or cars, to rather slow moving objects like people or books, we make no general assumption about how often an object changes its location. Note that mobile clients may be tracked objects as well. Due to connectivity problems or the limited coverage area of a tracking system the current location of a tracked object may be temporarily unknown to the location service.

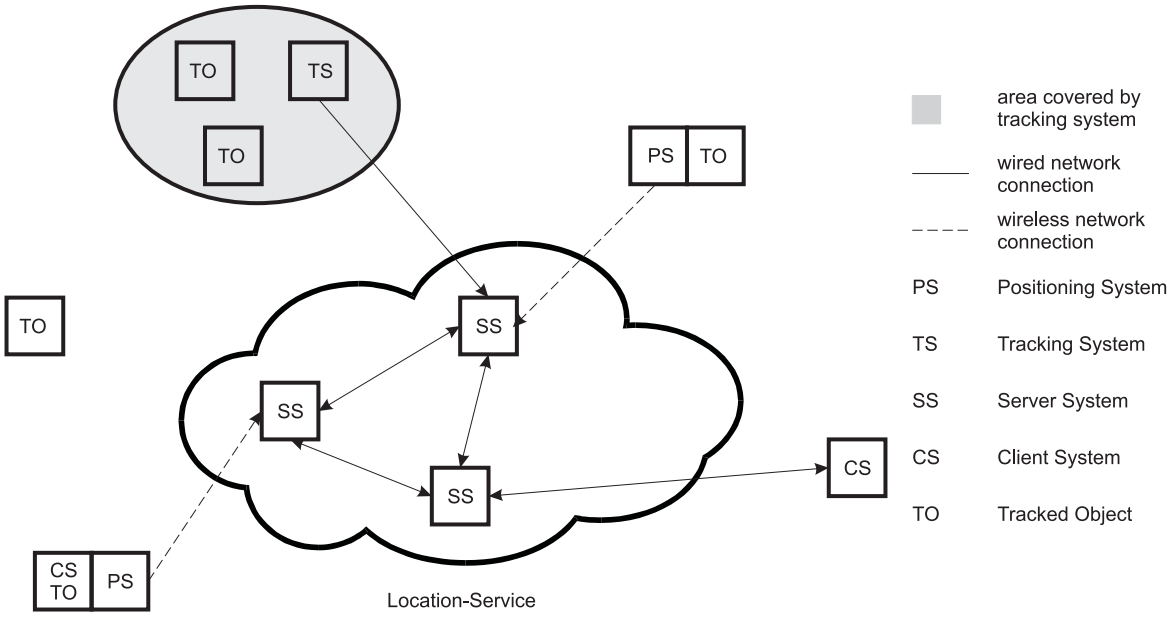

Figure 1: Components of the system model

3 Distribution Alternatives

In this section we present alternatives for the design of an universal location service. The alternatives differ in the distribution of the location information and possibly required registers. To be able to classify location services we develop a taxonomy for the different alternatives of distribution.

3.1 Distribution of Location Information

The location service must allow both LOO-queries and OAL-queries. Basically, we can think of a location service as a table mapping objects to locations and vice versa (see Figure 2).

Object	Location
O14	L10
O3	L31
O59	L42

Figure 2: Location table

For the distribution of this table over the server systems there are three possibilities:

- *centralized:* There is only one server which holds the entire location table.
- *distribution by object:* each server system is responsible for a set of tracked objects, e.g. all objects of a certain type. For each tracked object there is one responsible server system.
- *distribution by location:* each server system is responsible for a set of locations. Each location is covered by at least one server system.

If the location information is distributed, we need two mechanisms in order to find the suitable server systems for each query: one to determine a server system with location information about a certain object and another one to find out where information about a certain location is stored. The former is called *object register* the latter *location register.*

3.2 Object Register

The typical query which requires an object register is a LOO-query. The object register maps the object identifier in the LOO-query to the server system which is currently responsible for the object. In Figure 3 the processing of such a query is illustrated.

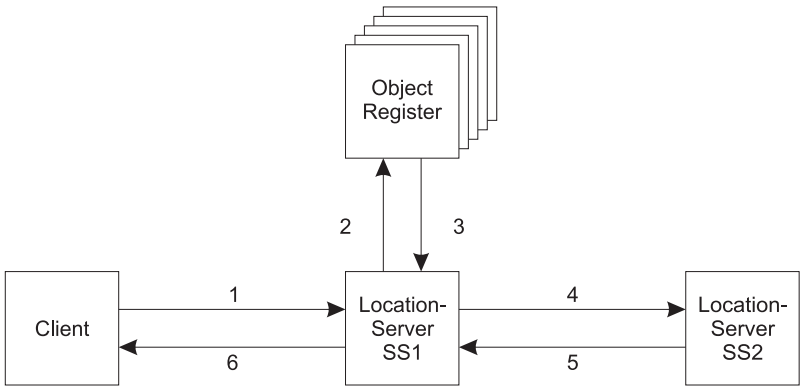

Figure 3: Query processing

It starts with the client sending a request to a location server SS_1 (1). If this server is not responsible for the request, it determines the responsible server by asking the object register (2). After receiving the answer from the object register (3) the server SS_1 forwards the request to the responsible server SS_2 (4). SS_2 sends the reply to SS_1 (5), which finally sends it to the client (6). This processing scheme can be optimized in many ways, e.g. the client itself might query the object register directly for the server system responsible for its query and send the query directly to the responsible server system. The use of the scheme here is only to illustrate the basic task of an object register. If the location information is distributed by object, the distribution does not change over time and is done according to a well-defined criterion, an explicit object register is not necessary. In this case the appropriate server system can be determined by the object's type, name or whatever the criterion for the distribution of the location information is. Then steps 2 and 3 can be omitted. If the distribution of the information changes, a mechanism is needed to find the actual server system for a particular object.

With a location information distribution by location, however, it is not possible to derive the server system responsible for an object from the object's attributes, except if there is an explicit attribute for it. So here an object register is needed to route the query to the appropriate server system[1]. Such an object register can be realized as a centralized or distributed database. The partitioning of this database can only be done by object, since the entries have to be found through the object's attributes.

For the realization of the object register the following alternatives can be identified:

• *none:* an object register is not realized, because the location information is distributed according to objects.

1. Even if each object has an explicit attribute, in which the address of the responsible server system is stored, the entirety of these attributes constitutes a distributed register.

- *centralized:* the object register is realized as a centralized database.
- *distributed:* the object register is realized as a partitioned database.

3.3 Location Register

To answer a OAL-query it is necessary to discover the server systems offering location information about any given range of locations.The processing of an OAL-query is quite similar to that of a LOO-query. The only difference is that a location register overtakes the role of the object register and that more than one server might be responsible for a query.

If the location information is distributed by object, it is impossible to derive the responsible server systems from the location range directly, because the objects located in a specific range might be distributed over an arbitrary set of server systems. Therefore, the only possibility to find the server systems without checking all server systems is an explicit location register.

If the information is distributed by location, it is possible to have a fixed mapping from a location range to a set of server systems, which are responsible for this range. With this approach a location register is not necessary. Such an approach works only if there is a predefined set of ranges, which does not change over time. To allow a flexible specification of the location ranges within OAL-queries, e.g. by arbitrary polygons, a location register is needed to map ranges to server systems.

Like the object register the location register can also be centralized or distributed. A distributed location register would be distributed by location, e.g. in hierarchical manner. So for the realization of the location register we have the same alternatives as with the object register realization: none, centralized and distributed.

4 Existing Location Services

In this section we classify existing location services using our taxonomy. We especially focus our considerations about these services on the distribution of the location information and the mechanisms to find responsible server systems.

Global System for Mobile Communication (GSM)

GSM is a digital cellular system for mobile voice and data communication. To route a call to a user’s cellular phone the system has to locate the called user first. For this purpose a location service is used.

The part of a GSM network that is responsible for locating mobile stations, i.e. cellular phones, and routing calls to them is called the Switching and Management Subsystem (SMSS). The SMSS is built up by one or more Mobile Switching Centers (MSC). Each MSC is responsible for a certain area of the network. The MSC's use two kinds of databases to locate mobile stations: the Home Location Register (HLR) and the Visitor Location Register (VLR). Usually there is one centralized HLR per network and one VLR per MSC. Each mobile station within the coverage area of a MSC is registered at the corresponding VLR and its actual location area is always stored in the VLR. The HLR holds all the permanent information about the mobile stations registered in its network, e.g. the subscriber identity, and for each of these stations a link to the VLR which the mobile station is actually visiting. To determine the location of a mobile station, its actual VLR is requested from the HLR. Subsequently its location area can be looked up in the reported VLR.

Therefore, according to our taxonomy, we can say that the location information is distributed by location over the VLRs and that the HLR serves as a centralized object register. OAL-queries are not explicitly supported in GSM and so no such thing as a location register can be found in the GSM architecture.

For the area covered by one MSC the according VLR operates as a centralized location service, since all mobile stations within this area report the location area they are actually visiting to the VLR. Reports are sent by a mobile station every time it enters a new location area.

For further details on the GSM architecture see e.g. EBERSPÄCHER and VÖGEL (1998). For future personal communication services (PCS), e.g. the Universal Mobile Telecommunication System (UMTS), the research area of location management plays an important role. It addresses the problem of distributing the location information in such a way that the costs for LOO-queries and updates are minimized.

Xerox PARC Ubiquitous Computing Environment

The location service for the Xerox PARC Ubiquitous Computing Environment (WEISER 1993) proposed by SPREITZER and THEIMER (1993) handles LOO-queries and OAL-queries differently.

Only the users of the system can be the subject of LOO-queries. These queries are handled by so-called user agents. For each user there is exactly one user agent which holds the location information about the user. A LOO-query about a user is sent to the user's agent, which, according to the user's privacy policy, answers the query or not. The location information about the users is therefore distributed by object over the user agents.

To answer OAL-queries, a so-called Location Query Service is used. It is organized by regions, with a centralized server, called the LocationBroker, running in each region. All the objects within a region are registered at the region's LocationBroker. To grant privacy to objects they are allowed to register in an anonymous fashion. According to our terminology the location information to answer OAL-queries is distributed by location. Since there are no registers offered in this location service, a client can only send requests to a user agent or LocationBroker it has an address for.

Location System for the Active Office

The Active Office described by HARTER and HOPPER (1994) uses a centralized approach. It is based on the Active Badge system developed by Olivetti and the University of Cambridge (WANT *et al.* 1991). The Active Badge System determines the location in a building visited by a user through an infrared sensor placed at this location. The sensor reads a signal emitted periodically by the user's Active Badge.

In the Active Office location system, all location information is stored in a central location server, which gets the location information from the Active Badge system. The system is designed for an Active Office application within one single enterprise. So a centralized solution can be used without running in scalability problems. Via a so-called Exchange Server, location information from remote systems can be requested. The Exchange Server implements remote access to another organization's location server.

Motion Databases

Another way of realizing the functionality of a location service is to extend a database system in a manner that allows to handle information about moving objects and their location. Such an extended database is sometimes called a Motion-Database (MOD). MODs are e.g. used for fleet management of transportation vehicles. The properties of MODs are examined by the DOMINO project whose objective is to integrate the functionality of a MOD into existing database management systems (WOLFSON *et al.* 1998). Research issues are the modelling of imprecision of position data and the reduction of position updates by using a dead-reckoning policy. In a MOD all queries are sent to the database system. Therefore a MOD constitutes a centralized approach and no registers are required to locate any server systems.

5 Analysis

In this section we discuss the cost for different ways of distributing a universal location service by comparing the overall load inflicted on the underlying network for a few selected approaches. The load is caused by the messages necessary for a single position update, LOO- or OAL-query and the overall number of updates and queries in a certain period of time. To estimate this load we make further assumptions based on the model described above.

5.1 Definitions

It is assumed that the area covered by the location service can be divided into *c* different cells. These cells are only used for organizational purposes and location information is provided with a much higher resolution. To each cell a location server can be associated. This could be done by physically placing the server at a central point within the cell, thus allowing local communication with the users in it. An example for such a cell would be a local site with an Active Badge installation in which users and server communicate over a (wireless) LAN. If the overall number of tracked objects or querying users increases and the maximum number of objects or users that can be handled by a server is fixed, the number of cells increases with the number of servers. The number of cells increases too, if the area covered by the location service grows because the maximum range of a cell is fixed (e.g. limited by the range of a wireless communication technology). The number of cells therefore is a good parameter for measuring the scalability of an approach.

To keep this analysis more manageable we do not distinguish between the characteristics of movements of users that issue queries and of movements of tracked objects. Instead, we consider a total of *n* mobile tracked objects, which are at the same time users that issue queries, moving within and between these cells. Being users, they regularly query the location of other tracked objects. It is further assumed that each of the objects has a home cell in which it spends more time compared to other cells, and that the objects are equally distributed over the cells. Thus, there is an average number of $\frac{n}{c}$ objects within any cell as well as an average of $\frac{n}{c}$ objects having one given cell as their home cell.

The ratio between the time an object spends in his home cell and the time it spends in any other cell is denoted by *h*. A high value of *h* indicates a strong locality of the objects's movements, a lower value can indicate a high mobility. Note, that if it spends the same amount of time in any cell *h* is $\frac{1}{(c-1)}$ rather than 1. Accordingly, *r* is the

ratio between queries concerning an object or a location in the current cell of the inquiring user and queries concerning all other cells.

To calculate the load we consider the overall number of updates and queries as well as the cost for a single update and query. The number of position updates generated by all tracked objects in a certain period of time is given by u (u can e.g. consist of an update-ratio multiplied with the number of objects). Accordingly, q_l denotes the number of LOO-queries and q_o the number of OAL-queries. In this paper we do not consider approaches that further reduce the number of position updates, e.g. by using a dead-reckoning policy as described by WOLFSON *et al.* (1998).

For the cost of a single update or query we assume that communication with a server at a different cell then the querying user is by a factor of m more expensive than communication with a server at the same cell. With C_u and C_q denoting the cost (e.g. the number of messages) for a local update or a query, the cost for a remote update and query can be obtained by multiplying C_u and C_q by m.

In the following subsections we define functions describing the overall network load for three different approaches of a location service. These functions are denoted l_x^y, where y is the type of distribution $y \in \{c, d_l, d_o\}$ and $x \in \{u, q_l, q_o\}$ denotes cause of the load. For a centralized approach $y = c$, for distribution by object $y = d_o$ and for distribution by location $y = d_l$. If the load is caused by updates $x = u$, if the load is caused by LOO-queries $x = q_l$ or $x = q_o$ if it is caused by OAL-queries.

When the location service is distributed either by object or by location, the appropriate servers for a query have to be found by a look-up in a register (see above). In this analysis we only consider centralized registers, which always have to be accessed by remote communication. The cost for a look-up is given by mC_l. If the object or location has been accessed a time short enough before and if the register information has been cached, the look-up may be unnecessary. Let a_l and a_o be the fraction of LOO- or OAL-queries for which a look-up is necessary. The registers only need to be updated if an object crosses a cell boundary. Set u_c to be the percentage of location updates which involve the changing of a cell.

Table 1 summarizes important variables used in the analysis.

var.	meaning	var.	meaning
c	number of cells	l	load caused by the total number of updates or queries per unit of time
n	total number of tracked objects (users)	a_l	percentage of LOO-queries which require a register look-up

Table 1: Variables used in the analysis.

var.	meaning	var.	meaning
h	ratio between the time an object spends in his home cell and any other cell	a_o	percentage of OAL-queries which require a register look-up
r	ratio between queries concerning the current cell of a user and any remote cell	u_c	ratio between normal updates and updates with changing of cell
u	number of position updates per unit of time	p_h	probability that an object is at its home cell
q_l	number of LOO-queries per unit of time	p_f	probability that an object is at a foreign cell
q_o	number of OAL-queries per unit of time	p_{f_c}	probability that an object is at a certain foreign cell
m	ratio of cost for a remote message compared to cost for a local message	p_h^1	probability that at least one object from a certain cell is at this cell (its home cell)
C_q	cost of single local query (mC_q for a remote query)	$p_{f_c}^1$	probability that at least one object from a cell is at a certain foreign cell
C_u	cost of single local update (mC_u for a remote update)	p_l	probability that a query concerns the current cell of the querying user
C_l	cost of single local look-up (mC_l for a remote look-up)	p_r	probability that a query concerns a remote cell

Table 1: Variables used in the analysis.

5.2 A Single Centralized Location Server

For many applications a centralized location server is sufficient. To keep our formulas simple, we assume that it is located outside of any cell and therefore communication with it is always remote. Thus, the total load caused by the updates in a certain period of time can be obtained by multiplying the total number of updates u with the cost for a single remote update mC_u. The same holds for the load of the queries. Since there is only one server, no registers are required.

$$l_u^c = umC_u \qquad l_{q_l}^c = q_l mC_q \qquad l_{q_o}^c = q_o mC_q$$

5.3 Distribution by Object

As a straightforward approach for distributing a location service by object we consider a location server for every cell which stores the position information of all tracked objects that have their home there. A location register is used to find all servers where the positions of objects currently visiting a certain cell are stored. Since each object is assigned to one location server (the server associated with its home cell), no object register is needed to find its position information.

An update can be performed by local communication, with a cost of C_u, if the object is in its home cell which is the case with a probability of p_h. If it is at any foreign cell (p_f), this has to be done by remote communication, with a cost of mC_u. Additional remote messages are needed to bring the centralized location register up to date, when the object crosses a cell boundary. The ratio between updates where an register update is necessary and updates where it is unnecessary is given by u_c. Therefore, the total load caused by the communication for updating the position can be calculated by:

$$p_h = \frac{h}{h+1} \qquad p_f = 1 - p_h = \frac{1}{h+1}$$

$$l_u^{d_o} = \underbrace{u(p_h C_u + p_f m C_u)}_{\substack{\text{location}\\ \text{server}}} + \underbrace{u_c u m C_u}_{\substack{\text{area}\\ \text{register}}} = uC_u\left(\frac{h+m}{h+1} + u_c m\right)$$

The cost of a LOO-query concerning a certain tracked object depends on whether it involves the current cell of the querying user (with a probability of p_l) or a remote one (with a probability of p_r). If in the first case the home and current cell of the queried object are identical to the queried cell (and current cell of the querying user) the cost for communicating with the server is local. In the second case the query can also be performed locally, if the queried object's home cell is identical to the current cell of the querying user and it is currently at the queried remote cell. The probability that the object is at this foreign cell is denoted by p_{f_c}. In any other case the communication has to be done remotely. A look-up is not necessary because the server for a certain object can be found through its home cell (which is e.g. given by its identifier):

$$p_l = \frac{r}{r+1} \qquad p_r = \frac{1}{r+1} \qquad p_{f_c} = \frac{p_f}{(c-1)} = \frac{1}{(c-1)(h+1)}$$

$$l_{q_l}^{d_o} = \underbrace{q_l((p_l p_h + p_r p_{f_c})C_q + (p_l p_f + p_r(1-p_{f_c}))mC_q)}_{\text{location server}} = q_l C_q\left(\frac{r(h+m)(c-1) + m(h+1)(c-1) - m + 1}{(r+1)(h+1)(c-1)}\right)$$

To carry out a query that returns the objects at a certain cell, all location servers have to be found, where the position of objects currently at this cell are stored. If the information is not cached due to a previous request for this cell, a look-up in the centralized location register is necessary (i.e. with a probability a_o). Then, each location server has to be contacted which stores the position information of at least one object in the queried cell. Let p_h^1 be the probability that for a given cell at least one of the objects is currently at home. Let $p_{f_c}^1$ be the probability that at least one object from a given home cell is at a certain foreign cell. If the query is local, the location server for the current cell of the user is contacted by local communication if at least one of the objects from this cell is at home. Communication is also local for querying a remote cell if at least one object with the current cell of the querying user as his home cell has moved to the requested foreign cell. In any other case communication is global.

$$p_h^1 = 1-(p_f)^{\frac{n}{c}} \qquad p_{f_c}^1 = 1-((c-2)p_{f_c}+p_h)^{\frac{n}{c}}$$

$$l_{q_o}^{d_o} = \underbrace{q_o((p_l p_h^1 + p_r p_{f_c}^1)C_q + (p_l(c-1)p_{f_c}^1 + p_r(p_h+(c-2)p_{f_c}^1))mC_q)}_{\text{location server}} + \underbrace{a_o q_o m C_l}_{\text{area register}}$$

5.4 Distribution by Location

In the case of distribution by location a location server is again associated with each cell, this time storing the positions of all objects currently within this cell. We assume that for OAL-queries the server responsible for objects at a certain location can be derived directly from the location specified in the query and therefore no location register is needed. An object register is used to find the server that currently stores the position information of a certain object. The derivation of the formulas is similar to the one above and is not presented in detail.

When an object changes its position within a cell boundary, only the local location server for this cell needs to be informed. If the object crosses a cell boundary the centralized object register also has to be updated, with a probability of u_c. Therefore, the cost for updating is:

$$l_u^{d_l} = \underbrace{uC_u}_{\text{location server}} + \underbrace{u_c u m C_u}_{\text{object register}} = uC_u(1+u_c m)$$

To obtain the current position of a certain object, the object register is queried to find the appropriate server. This is only necessary, if this information has not been cached from a previous access. The cost for the actual query only depends on whether the query concerns a local or a remote cell.

$$l_{q_l}^{d_l} = \underbrace{q_l(p_l C_q + p_r m C_q)}_{\text{location server}} + \underbrace{a_l q_l m C_l}_{\text{object register}} = q_l\left(C_q\left(\frac{r+m}{r+1}\right) + a_l C_l m\right)$$

The load for OAL-queries is equivalent, but without the need to contact a register, since there is always one fixed server responsible for the objects in a certain cell.

$$l_{q_o}^{d_l} = \underbrace{q_o(p_l C_q + p_r m C_q)}_{\text{location server}} = q_o C_q\left(\frac{r+m}{r+1}\right)$$

5.5 Comparison

In the following section we use the above formulas to compare the approaches for distributing a location service according to some important parameters. Due to the limited space we can only examine some basic aspects for each

of the approaches. As our key concern is the scalability of the location service, one of the aspects we consider is how the load behaves for an increasing number of cells (see above). We compare a centralized location service, a location service distributed by object with a centralized location register and a location service distributed by location with a centralized object register. Since the formulas depend on many parameters we have to make further assumptions to show the behaviour according to the parameters we are interested in.

As we are only interested in comparing the approaches and not in the absolute value of the load, we divide the functions in all but the last case by the load of the centralized approach which is thus set to 1. It is assumed that the costs for a single update, query and look-up are equal ($C_u = C_q = C_l$). The cost of a remote query is set to be $m = 10$ times more expensive than for a local query. This is a reasonable assumption, if we use hop count as a measurement for the cost of a message in the Internet and have an average number of at least 10 hops compared to 1 for a local message. The difference should be even larger when considering communication in a local and a global wireless network (1 MBit/s compared to 9,6 kbit/s or 10ms delay compared to >1s). If not specifically mentioned, the number of objects per cell is set to 100. The parameters for the locality of objects *h* and the locality of queries *r* are supposed to be $\frac{5}{c-1}$ and $\frac{10}{c-1}$, when they are not examined themselves. The value of *h* is set to a moderate level because we assume more mobile objects which do not have a high locality of their movements. Since location-aware queries often concern the vicinity of the querying user, as observed in HARTER and HOPPER (1994) e.g. in a navigation system, the value of *r* is higher. The parameters that indicate how frequently the registers have to be accessed are set as follows: $a_l = a_o = 0.2$ and $u_c = 0.1$. If we have a high accuracy and therefore frequent updates, u_c could be even smaller. All these values are intended to show the general behaviour of a location service and are not supposed to reflect a particular real world scenario.

First, we compare position updates. Figure 4 shows how the load for updates behaves when the number of cells is increased. With distribution by location the communication is always handled in the local cell compared to global communication for the centralized approach; the ratio between the two loads is therefore constant. As with more cells the probability decreases that an object is in its home cell, the expense for updates increases in a distribution by object approach. Note, that the load for the centralized approach is always set to 1, even if the absolute load increases with more cells and objects. Next, we examine the effect of the locality of objects, *h*. As Figure 5 shows, the cost of updates decreases (comparatively) for a distribution by object, if the locality *h* of the objects increases.

In Figure 6 the load behaviour of LOO-queries is compared for centralized, distribution by object and distribution by location approaches. In case of the standard parameters, a distribution by object is slightly better for a higher number of cells. The load caused by the distribution by location approach depends on the locality of queries r and improves if there is a high locality, as can be seen in Figure 7.

The difference between the approaches is more evident for OAL-queries. As the number of cells increases and with it the probability that more objects from different cells are at the queried location, the distribution by object approach generates a much higher load with the given parameters than a distribution by location (Figure 8). This is the case because we assume a high number of objects per cell. Figure 9 shows how the load for a distribution by object approach depends on the average number of objects at each cell and that it becomes better if this number is very low.

In Figure 10 the overall load caused by updates and queries of the three approaches is depicted for our standard parameters. Since for a high accuracy of the position information many updates are needed and OAL-queries are more complicated to use than LOO-queries, we have set u to be 10 times higher than q_l and q_l to be again 10 times higher than q_o.

To get an impression of how the absolute values of the loads behave, we have set the cost for a global update and query to 1, as well as the period of time, in which an object creates one update. The resulting curves for the absolute load are shown in Figure 11 for the three approaches.

6 Application Example of the Analysis' Results

In the previous sections we identified the different possibilities of distributing a location service and developed formulas to compare these possibilities. Now we show how the results of this comparison can be used to derive a location service architecture from some given requirements. We consider the requirements derived for a location service for the Nexus system.

In this system a universal location service is needed, i.e. a location service that has a global coverage and integrates different tracking and positioning technologies. This service has to answer both LOO- and OAL-queries. It should be possible to specify the location range to which a OAL-query refers by an arbitrary polygon. Furthermore the location service has to deal with a large number of tracked objects, clients and a large coverage area.

According to the requirements we obviously have to distribute the location information in order to handle the large number of tracked objects and clients. To find the best way to partition the location information, we consider the

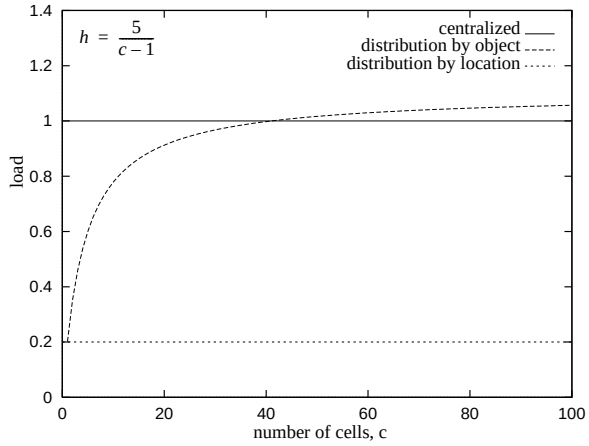

Figure 4: Update load versus number of cells.

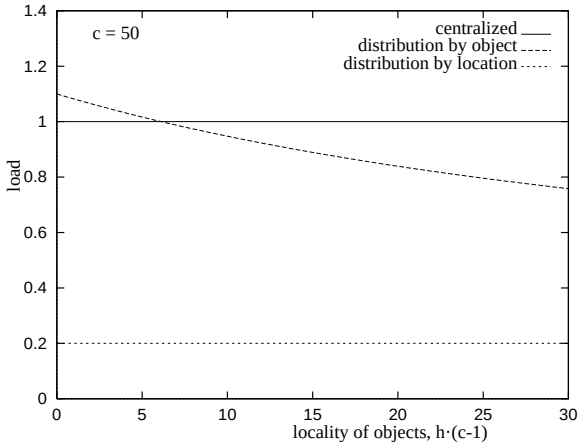

Figure 5: Update load versus locality of objects.

values for the parameters used in the formulas in Section 6, as they are expected in the Nexus environment. Since the Nexus location service must have a large coverage area, the number of cells will be high ($c \gg 100$). Also a high

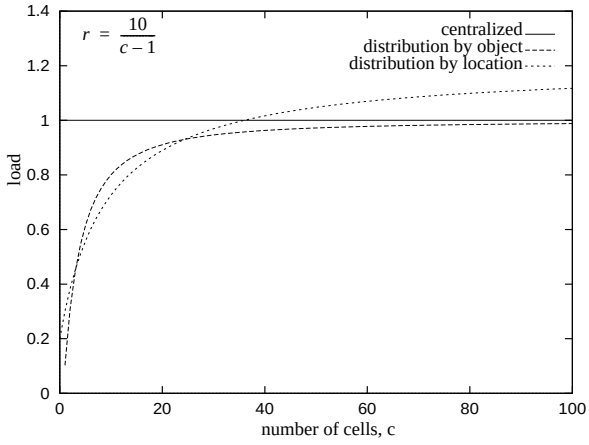

Figure 6: LOO-query load versus number of cells.

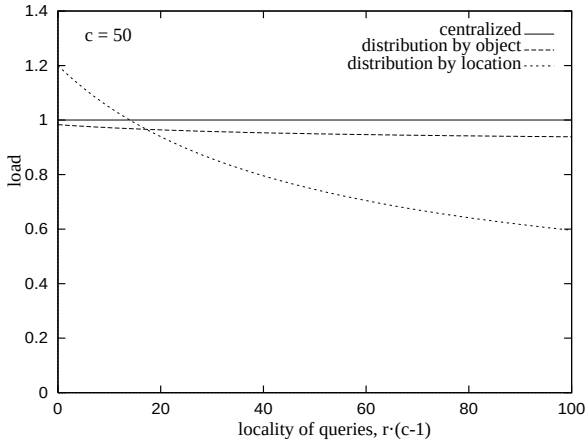

Figure 7: LOO-query load versus locality of queries.

value for the locality of queries parameter ($r > 0.75$) is expected, because the users are primarily interested in objects around them. In contrast, the locality of the users themselves will be moderate ($h < 0.5$). Now the results from Section 6 can help us to decide which kind of distribution we have to choose. Figure 4 and 5 show that the update load is always minimal for a distribution by location. From the LOO-query load formulas we can see that the gain of a distribution by object over an distribution by location is rather small with our expected high value of r. So as illustrated in Figures 8 and 9 this small gain is outperformed by the advantages of a distribution by location if the

OAL-queries are considered, especially with the expected high number of cells and objects per cell. Since the

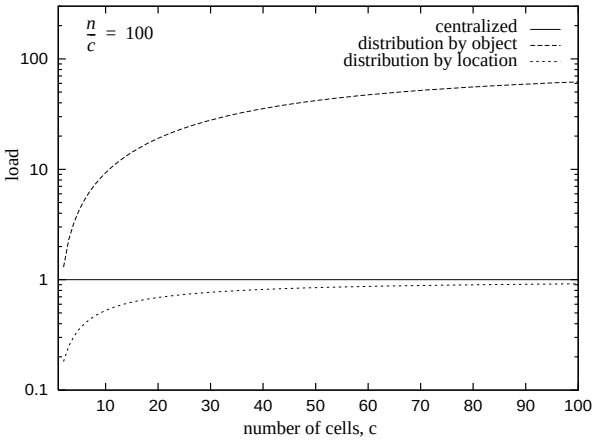

Figure 8: OAL-query load versus number of cells.

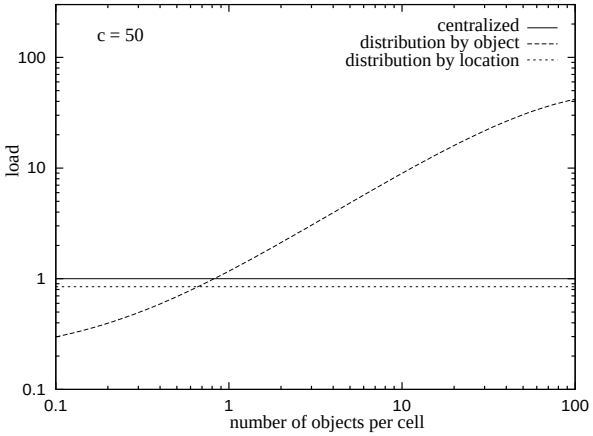

Figure 9: OAL-query load versus number of objects per cell.

number of LOO-queries is not expected to be more than 10 times higher than the number of OAL-queries we can use Figure 10 to see that the advantage of a distribution by location holds for the overall load.

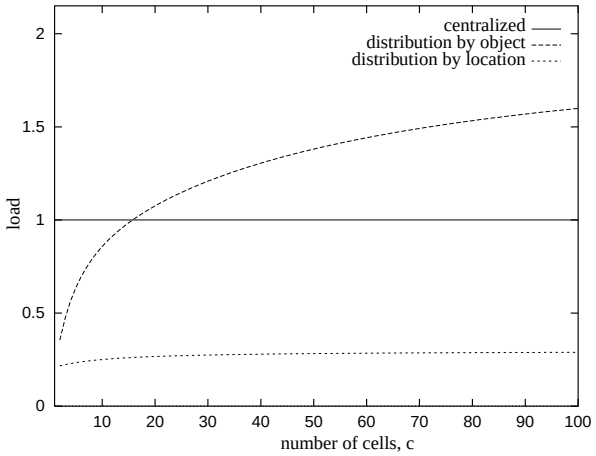

Figure 10: Comparison of overall load.

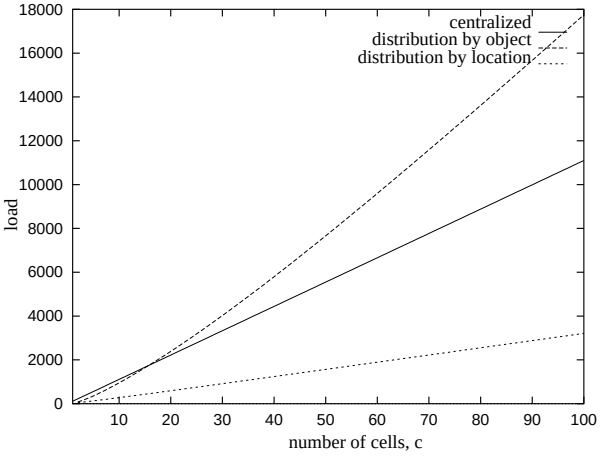

Figure 11: Absolute values for overall load.

So for the requirements of the Nexus system a distribution by location fits best. A fixed mapping of locations to server systems cannot be used, because a flexible specification of the ranges in the OAL-queries is required. Therefore, as explained in Section 3.3, a location register is needed. With the distribution of the location table by location we also need an object register in order to find location information about a certain object (see Section 3.2). So the Nexus location service should consist of a set of location servers, over which the location information is distributed by location, an object register and a location register.

7 Conclusion

In this paper we presented a system model for the distribution of a location service and an analysis, by which different alternatives given in the model can be evaluated. Our model distinguishes between a centralized location service and distribution by object or distribution by location. In a distributed approach an object register is used to find the server that stores the location information of a certain object. A location register returns the servers which store the information for all objects currently at a given location. In a (distributed) location service there is either none, a centralized or a distributed object and/or location register.

Next, we compared distribution by object to distribution by location by calculating the load caused by the updates and queries of the approaches. As the calculations depend on a number of assumptions, they are in this form not suitable to obtain absolute values for the load caused by a certain design. However, they will be useful when deciding which sort of distribution to use for a component according to its purpose. Based on them, we finally outlined an architecture for a global distributed location service based on the results of the evaluation.

Concerning our future work, we have already begun with a more detailed design of the location service and plan to implement a prototype soon. Outdoor location information will be provided by DGPS-sensors, indoor by an Active-Badge-like system of infrared sensors. This prototype will be part of the Nexus-Project (NEXUS 1998) which deals with spatial oriented information access. It will be used to gather data about the usage of a location service in a real world environment and to further improve and validate our calculations and system architecture.

References

BEADLE H., HARPER B., MAGUIRE JR., G. AND JUDGE, J. (1997), Location Aware Mobile Computing, in *Proceedings of the EEE/IEEE International Conference on Telecommunications*, pp. 1319-1324.

EBERSPÄCHER, J. AND VÖGEL, H.-J. (1998), *GSM Switching, Services, and Protocol*, B.G. Teubner, Stuttgart, Germany.

HARTER, A. AND HOPPER, A. (1994), A Distributed Location System for the Active Office, *IEEE Network*, 36(1), pp. 62-70.

HOFMANN-WELLENHOF B., LICHTENEGGER, H. AND COLLINS, J. (1997), Global Positioning System: Theory and Practice, Springer-Verlag TELOS, ISBN 3211828397.

JANNINK, J., LAM, D., SHIVAKUMAR, N., WIDOM, J. AND COX, D.C. (1996) Efficient and Flexible Location Management Techniques for Wireless Communication Systems, in *Proceedings of the Second ACM International Conference on Mobile Computing and Networking (MobiCom '96)*, pp. 38-49, ACM Press.

JOHNSON, D. (1995), Scalable Support for Transparent Mobile Host Internetworking, *Wireless Networks*, vol. 1, pp. 311-321.

LEONHARDT, U. (1998), *Supporting Location-Awareness in Open Distributed Systems*, PhD-thesis, Imperial College of Science, Technology and Medicine, University of London.

MAAß, H. (1997), Location-Aware Mobile Applications based on Directory Services, in *Proceedings of the Third International Conference on Mobile Computing and Networking (MobiCom '97*), Hungary, pp. 23 -33, ACM Press.

NEXUS (1998), Nexus Project Pages, University of Stuttgart, Stuttgart, Germany, `http://www.informatik/uni-stuttgart.de/ipvr/vs/projekte/nexus.html`.

PFEIFER, T., MAGEDANZ, T. AND HÜBENER, S. (1998), Mobile Guide - Location-Aware Applications from the Lab to the Market, in *Proceedings of the 5th International Workshop IDMS*, Lecture Notes in Computer Science 1486, Springer-Verlag, Germany.

SCHILIT, B. (1995), *A System Architecture for Context-Aware Mobile Computing*, PhD-thesis, Columbia University.

SPREITZER, M. AND THEIMER, M. (1993), Providing Location Information in a Ubiquitous Computing Environment, in *Proceedings of the Fourteenth ACM Symposium on Operating System Principles*, pp. 270-283, volume 27 of ACM SIGOPS.

WANT R., FALCAO, V. AND GIBBONS, J. (1991), The Active Badge Location System, *ACM Transactions on Information Systems*, 10(1), pp. 91-102.

WEISER, M. (1993), Some Computer Science Issues in Ubiquitous Computing, *Communications of the ACM*, 36(7), pp. 74-83.

WOLFSON, O., SISTLA, A. P., CHAMBERLAIN, S. AND YESHA, Y. (1998), Updating and Querying Databases that Track Mobile Units, to appear in *Distributed and Parallel Databases Journal*.