Architektur eines verteilten skalierbaren Lokationsdienstes

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Alexander Leonhardi

aus Ulm/Donau

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Prof. Dr.-Ing. habil. Bernhard Mitschang

Tag der mündlichen Prüfung: 2. Juni 2003

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart

2003

Danksagung

Mein besonderer Dank gilt meinem Doktorvater, Prof. Dr. Kurt Rothermel, dessen Betreuung mich viel über das wissenschaftliche Arbeiten lehrte und durch dessen Anregungen sich mir viele Aspekte der Thematik erst erschlossen haben. Prof. Dr. Bernhard Mitschang danke ich herzlichst für die bereitwillige Übernahme des Zweitgutachtens.

Großen Dank schulde ich auch allen Mitarbeitern der Abteilung Verteilte Systeme für das sehr gute und offene Arbeitsklima und für viele interessante Diskussionen. Besonderer Dank gebührt hier Christian Maihöfer und Martin Bauer. Beim Korrekturlesen verschiedener meiner Arbeiten habe ich von ihnen viele wertvolle Anregungen erhalten.

Des Weiteren danke ich allen Mitarbeitern der Universität Stuttgart, die an der von der Deutschen Forschungsgemeinschaft (DFG) geförderten Forschergruppe Nexus beteiligt waren, in deren Rahmen meine Dissertation entstanden ist. Besonders anregend und spannend war für mich die gute Zusammenarbeit in einem solchen interdisziplinären Umfeld.

Zum Gelingen dieser Arbeit haben zudem in großem Maße auch viele Studierende im Rahmen von Diplom- oder Studienarbeiten sowie als wissenschaftliche Hilfskräfte beigetragen. Diesen gebührt ebenfalls mein Dank, da ohne sie die Dissertation in dieser Form nicht möglich gewesen wäre.

Esslingen, den 22. Juni 2003

Alexander Leonhardi

Kurzfassung

Anwendungen mit Ortsbezug bieten Informationen oder Dienste an, die abhängig sind von der aktuellen geographischen Position ihrer Benutzer oder anderer mobiler Gegenstände, wie Fahrzeuge oder Bürogegenstände. Durch die Nutzung dieser Kontextinformation können dem Benutzer situationsgerecht relevante Informationen und Dienste angeboten werden, anstatt ihn mit Informationen zu überfrachten. Da dies besonders wichtig für die wachsende Anzahl kompakter mobiler Endgeräte wie Mobiltelefone und PDAs ist, werden Anwendungen mit Ortsbezug als wichtiges Argument für die Datendienste der kommenden Mobilfunkgeneration gesehen. Fortgeschrittene ortsbezogene Anwendungen verarbeiten hochgenaue geographische Positionsinformationen, wie sie durch moderne Positionierungssensoren bereitgestellt werden, und bieten eine Funktionalität, die über das einfache Betrachten der aktuellen Position eines Benutzers hinausgehen, z. B. indem sie alle mobilen Objekte bestimmen, die ein bestimmtes geographisches Gebiet betreten haben. Eine Betrachtung existierender Technologien wird zeigen, dass diese nicht für die dafür notwendige zentrale Verwaltung der Positionsinformationen mobiler Objekte geeignet sind.

In dieser Arbeit wird ein generischer skalierbarer Lokationsdienst vorgestellt, der speziell mit dem Ziel entwickelt wurde, die hochdynamischen Positionsinformationen für eine große Anzahl mobiler Objekte zu verwalten. Der Lokationsdienst kann als Infrastrukturkomponente von ortsbezogenen Anwendungen genutzt werden und sein Dienstmodell ist speziell auf die besonderen Eigenschaften von Positionsinformationen, wie deren Dynamik und eine unterschiedlich hohe Genauigkeit, ausgelegt. Die Skalierbarkeit des Dienstes wird durch eine hierarchische, verteilte Architektur auch für eine sehr große Anzahl von mobilen Objekten und Klienten sichergestellt. Um den Lokationsdienst darüber hinaus in die Lage zu versetzen, mit den häufigen Aktualisierungen zurechtzukommen, die sich aus der hochdynamischen Natur der Positionsinformationen ergeben, wird eine spezielle Datenhaltungskomponente vorgeschlagen, die auf einer sehr effizienten Hauptspeicherdatenbank basiert.

Eine wichtige Voraussetzung für den Lokationsdienst sind effiziente Protokolle zur Übertragung von Positionsinformationen, deren Untersuchung ein zweiter Schwerpunkt dieser Arbeit ist. Zu diesem Zweck wird eine umfassende Klassifizierung dieser Protokolle erarbeitet und die sich ergebenden Protokollklassen werden mittels Analyse und Simulation hinsichtlich ihrer Ef-

fektivität und Effizienz für typische Einsatzbedingungen verglichen. Innerhalb der Klasse der sogenannten Koppelnavigationsprotokolle, die sich dabei als besonders Erfolg versprechend herausgestellt hat, wird ein spezielles kartenbasiertes Koppelnavigationsprotokoll entwickelt, mit dem Ziel, den erforderlichen Nachrichtenaufwand weiter zu reduzieren.

Simulationsergebnisse und Experimente an einer prototypischen Implementierung des Lokationsdienstes zeigen die Machbarkeit und die Leistungsfähigkeit der beschriebenen Mechanismen. So lässt sich mit dem betrachteten kartenbasierten Koppelnavigationsprotokoll bis zu 91% des Nachrichtenaufwands gegenüber bisher eingesetzten Protokollklassen einsparen. Für den Lokationsdienst selbst zeigen die Ergebnisse, dass dieser Anfragen und Aktualisierungen effizient und skalierbar bearbeiten kann. Bereits auf einfacher Hardwarebasis ist unser Prototyp in der Lage, auf jedem einzelnen Lokations-Server über 500 Aktualisierungen pro Sekunde zu verarbeiten.

Architecture of a Large-scale Location Service

English Abstract

Location-aware applications offer information and services depending on the current geographic location of their users or other mobile objects like vehicles or office devices. Utilizing this important context information these applications can provide information and services pertaining to the current situation of their users instead of overwhelming them with information. This is especially important for the growing number of mobile devices like mobile phones or PDAs, which can only offer a limited user interface due to their size. Location-aware applications are therefore seen as an important argument for the data services of the coming generation of mobile communication systems.

Advanced location-aware applications require highly accurate location information as can be obtained from modern positioning sensors. They also provide a functionality that goes beyond just considering the position of a single user, for example by determining all mobile objects that have entered a certain geographic area. The latter functionality, for example, requires a central management of the mobile objects' location information. An evaluation of related work will show that existing technologies are not suitable for the management of the highly accurate location information, which is required for advanced location-aware applications.

In this thesis we present a generic location service (LS), which has been developed especially with the goal of managing the highly dynamic location information for a large number of mobile objects in a scalable way. It is intended to be used as an infrastructure component for location-aware applications. Besides a position query, which returns the current location of a certain mobile object, the main queries supported by the proposed LS are range and nearest neighbor queries. A range query returns all mobile objects inside a queried geographic area; a nearest neighbor query the object nearest to a certain geographic position. The service model of these queries has been designed to take into account the special properties of location information, that is, their dynamic nature and different accuracies. The scalability of the LS, even for a large number of mobile objects and clients, is achieved through a hierarchical, distributed architecture. To ad-

ditionally enable the LS to perform position updates in an efficient way, which is necessary because of the highly dynamic nature of the location information, we propose a special data storage component based on an efficient main memory database.

An important requirement for the LS is the ability to efficiently transmit location information with a given accuracy. The examination of suitable update protocols is therefore a second main topic of this thesis. To this end an extensive classification of protocols for updating location information is presented. The resulting protocol classes are compared by means of analysis and simulation regarding their efficiency and effectiveness for typical application environments. Especially the class of so-called dead-reckoning protocols proves to be very efficient. As part of this class, a special map-based dead-reckoning protocol is developed, with the goal of further reducing the number of required update messages.

Simulation results and experiments with a prototypical implementation of the LS show the feasibility and the efficiency of the proposed mechanisms. With the new map-based dead-reckoning protocol it is possible to reduce the necessary messages compared to currently used protocol classes by up to 91%. For the LS itself, the results show that queries and position updates can be handled in an efficient and scalable way. Even on a simple hardware platform, our prototype is able to process more than 500 position updates per second on each server.

The following sections contain an English summary of this thesis. In the next section related work is discussed and how it relates to the work done in the thesis. The following section sketches the architecture of the LS and that of its data storage component, while the subsequent section deals with protocols for updating position information. In the thesis we additionally discuss how to handle security and privacy aspects of storing accurate location information, which are not covered by this abstract because of space limitations. The interested reader is referred to LEONHARDI & ROTHERMEL (2002) for more details about the architecture of the LS and to LEONHARDI & ROTHERMEL (2001) and LEONHARDI, NICU & ROTHERMEL (2002) for details about protocols for updating location information.

Related Work

Much research effort has been invested in developing highly accurate positioning sensors. While outdoors the GPS system offers a universal and reliable solution, a similar system for indoors is still the focus of research efforts. Proposed indoor positioning systems include the highly accurate ultrasonic Active Bat system, the RADAR system, which utilizes the field strengths of a wireless LAN, and the Cricket system, which combines low resolution radio and high resolution ultrasonic positioning. For an overview of positioning systems see HIGHTOWER &

BORRIELLO (2001). However, all of these systems provide location information only for a local environment and use different location models and interfaces. Our proposed LS is designed to take the input from such positioning systems and to make it globally available in a uniform manner.

A simple way of realizing the data storage for a LS would be to use a spatial extension to an existing database product, as described in DAVIS (1998), or a Geographical Information System (GIS). These are, however, optimized for large amounts of data and complex queries and are not suitable for frequent updates. Current issues in database research are therefore moving objects databases (MODs) (see WOLFSON ET AL. (1998)) or spatio-temporal databases, which are especially designed for storing information about moving objects including their location. Research interests focus on aspects of data storage, for example indexing and query processing. The discussed systems do not consider the distribution of the location information on more than one node, which is necessary to create a large-scale service like the proposed LS.

Much research has been done on efficient, scalable location management for Personal Communications Systems. Here a distributed location management component is used for finding the current communication cell of a mobile phone when a call for this phone comes in. To improve scalability over current 2-layered systems, as used for example in GSM, hierarchical concepts and user profile replication have been proposed for future Personal Communication Systems (see for example WANG (1993)). Although many concepts of location management, like the hierarchical architecture and forwarding pointers, can be used for the LS, location management is primarily concerned with finding a certain mobile phone and does not consider range or nearest neighbor queries. Also, location information is only captured with the granularity of a location area, which consists of several of the network's communication cells and can be rather large (up to 40 km).

As an aspect of the location management of Personal Communication Systems, protocols for transmitting location information have also been examined extensively. In BAR-NOY, KESSLER & SIDI (1995), for example, a distance-based, a movement-based and a time-based reporting protocol are compared. In this case, the location information is a discrete identifier, describing a certain location area of the communication network. Update policies for location management have been optimized for these special requirements. Dead-reckoning strategies for updating more accurate location information have first been introduced in WOLFSON ET AL. (1999) for moving objects databases. With these promising protocols the database estimates the current location of a mobile object on a predefined route based on its last reported speed and the object sends an update of its location when it differs from this estimation by more than a certain distance threshold. Different dead-reckoning strategies are proposed and compared. However, up-

date protocols for highly accurate continuous location information have not so far been examined in a general manner.

A directory service, like the Intentional Naming System (ADIJIE-WINOTO ET AL. (1999)) for example, can be used to store and retrieve information about provided services – often physical devices like a printer or computer. One possible attribute of the service description is the geographic location of such a device. Service discovery systems (e.g., de-centralized systems like JINI (2001)) are intended for mobile clients and allow them to discover near-by service providers. Both directory services and service discovery systems allow the retrieval of the position of a device or service, but do not support general range or nearest neighbor queries nor varying accuracies of the location information. They are also often not intended for a large-scale use and assume more static data with fewer updates. The proposed LS is not assumed to compete with directory or service discovery services. Instead, the LS may be used to track the position of a mobile device, which has been found through a directory service, or a directory service may supply a description of a mobile object, which has been found through the LS.

A service for locating mobile objects (mainly software objects) worldwide is being developed as part of the Globe (see VAN STEEN ET AL. (1998)) project. To achieve the scalability for the desired very large number of mobile objects, the service uses a hierarchical search tree. However, the Globe location service returns the contact address of a queried object rather than a geographic position and therefore also does not support range or nearest neighbor queries.

The primarily targeted clients for the proposed LS are location-aware applications (such as CHEVERST ET AL. (2000)), which require a component that manages the location information for the involved mobile objects. Early systems were often limited to certain sensor systems or applications. In HARTER ET AL. (1999) a more general location service for a building-wide deployment and intended for very accurate location information is presented, which provides an efficient event mechanism based on the notion of geometric containment. Leonhardt (LEONHARDT (1998)) has proposed a large-scale distributed location service that is independent of application and sensor systems. In his PhD-thesis Leonhardt discusses fundamental issues of a location service, such as a location model for the integration of different types of sensor data and policies for access control, but does not propose or evaluate a specific architecture.

Architecture of the Location Service

In this section, we describe the service model and the architecture for the proposed large-scale LS. As part of a future location-aware information system for a larger city, such a LS may have

hundred thousands of users. Its scalability and the efficient execution of location updates and queries are therefore of great importance.

Service Model

In our service model for the LS we distinguish between the *location service* itself, which is typically implemented as a distributed system consisting of a number of location servers, *tracked objects*, which are mobile objects whose location information is managed by the LS, and *clients* that query their location information through the LS. The LS is responsible for managing the location information of mobile objects inside a certain *service area*. It stores the location information for a certain tracked object in a so-called *sighting record.*

To become a tracked object, a mobile device must register with the LS by issuing the *register* operation, which also specifies the accuracy the location information should be stored with. To update the sighting records stored in the LS's database, a tracked object or stationary tracking system regularly sends *update* requests to the LS (according to a given update protocol described in the following section), which guarantees the specified accuracy.

The LS basically offers three types of queries, namely position queries, range queries and nearest neighbor queries, to retrieve the position information of tracked objects. A position query, *posQuery*, is used to request the current position of a given tracked object. In a fleet management system, for example, this type of query could be used to get the current position of a certain truck, which has been scheduled for an inspection at short notice. A range query, *rangeQuery*, determines all tracked objects inside a certain geographic area. Such an area can be defined as an arbitrary connected polygon given by the geographic coordinates of its corners. An application in the area of fleet management would be to find all trucks that are in a given part of a city. Finally, the nearest neighbor query, *neighborQuery*, returns the object with the minimal distance to a given geographic position. It could be used to find the nearest (free) truck for a load of goods.

Location Service Architecture

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. While the shape of a service area may be any kind of polygon, the following two requirements must be fulfilled: (1) A non-leaf service area consists of its child service areas, that is, it is the union of its child areas, and (2) sibling service areas do not overlap.

Associated with each service area is a location server, which is responsible for tracking all 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 may not overlap, for each tracked object there exists exactly one leaf server that is responsible for keeping track of the object's position at any point in time. We will call this leaf server the object's *agent*. Of course, whenever a tracked object moves from one service area to another, the tracking responsibility has to be handed over to another leaf server, which then becomes the object's new agent.
- A ***non-leaf server*** is responsible for a service area that is the union of the service areas associated with its child servers. Obviously, the size of service areas associated with the servers increases in a leaf-to-root direction. A non-leaf server records all tracked objects that are currently within its service area and for each of these objects stores a so-called *forwarding reference*. An object's forwarding reference identifies the child server that is responsible for the child service area this object is currently located in.

Consequently, only leaf servers store sighting records, whereas non-leaf servers store a reference to the child server whose service area contains the position of the corresponding mobile object. Obviously, the collection of forwarding references stored for any given tracked object in the server hierarchy specifies a path from the root server to the object's agent, from where the object's sighting record can be retrieved.

Location updates for a tracked object are always sent to the object's agent, which then updates the object's sighting record. 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. Handover processing must then make sure that the tracked object learns about its new agent. Due to the hierarchical organization of service areas, handovers between higher level service areas will occur more rarely than for lower level areas.

A client *c* that is interested in the location information stored by the LS sends a query request to a nearby leaf server, which becomes the *entry server* for this query. Because of an assumed high locality of queries, the entry server will often be able to answer the query itself or has to go only a short distance to find the appropriate leaf server(s). If *c* is also a tracked object of the LS, a suitable entry server is already given by its current agent.

To execute a position query, the forwarding path is used to find the agent of the corresponding tracked object. A position query is forwarded upwards until a server containing a forwarding reference for the object is found. From there the request is forwarded along the object's forward-

ing path down to the object's agent that sends the object's location information back to the entry server. Finally, the entry server returns the location information to *c*.

For a range and a nearest neighbor query the inclusion relationship for service areas is utilized instead of the path of forwarding references. A request is propagated upwards from the entry server until a location 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, until all leaf servers are reached that are responsible for parts of the requested area. These leaf servers determine the objects that are in the corresponding sub area and send the result to the entry server, which is responsible for constructing the answer that is then returned to the client. A nearest neighbor query is handled in a similar way except that the leaf server has to be found that includes the given position (in some cases also neighboring servers).

Data Storage

For the storage of the data on a location server of the LS we distinguish between the storage of the more static registration data in the *visitorDB* and the storage of the highly dynamic location information in the *sightingDB*. The *visitorDB* is kept on stable storage, which is updated only when an object is registered, deregistered or a handover occurs. In other words, the objects' registration information and forwarding paths are supposed to survive system failures. Stable registration information also allows a location server to ask a visitor for a position update to restore its location information after a system restart following a failure.

We have decided to store the *sightingDB* in volatile main memory for two reasons. First, we expect position updates to occur very frequently, and hence, updates, in particular, should be performed most efficiently. Second, the overhead for making location data recoverable from stable storage is not worth the effort since the recorded positions would be most probably out-dated anyway after recovery. Instead, the position updates sent (periodically) by the tracked objects can be used to restore the data in main memory.

A spatial index over the location information in the sighting records is used to efficiently retrieve the results for range or nearest neighbor queries. We have performed experiments with a number of spatial index structures and found a Quadtree to be well suited for our purpose. Position queries are efficiently processed by using a hash index structure defined over the object identifiers. The index structures are also stored in volatile main memory. Our measurement results on a prototypical implementation show that even the spatial index can be built up very quickly, for example after a system failure.

Experiments

To evaluate the proposed mechanisms we have created a prototype implementation of the LS. With this prototype implementation we have performed various experiments to determine the throughput and response time of the data storage component and of local and distributed query processing. Our experiments with a single server show that the main-memory data storage component is able to perform more than 2000 operations per second on a PC server even for range queries with a large queried area and will therefore not be a bottle-neck of an implementation of the LS. For the overall performance we tested the performance of location updates and queries with different distributed configurations of the LS with a small hierarchy of 5 location servers. The achieved update rate would be sufficient to support location information for more than 55,000 mobile objects with an average speed of 3 km/h and an accuracy of 25 m. Similarly, local and remote position, range and nearest neighbor queries can be performed very efficiently.

Updating Location Information

In order to guarantee a certain accuracy of the location information stored in the LS, the information has to be transmitted (updated or requested) with an appropriate frequency from the tracked object or an external positioning system to the LS. Most often the location information is transmitted from a mobile device to a location server where a wireless channel has to be used. In this case, bandwidth is low and expensive. It is therefore important to use an update protocol that works efficiently, that is, requires as few messages as possible, as well as effectively, that is, achieves the desired accuracy of the location information on the server.

In this thesis we discuss the characteristics of different classes of update protocols and their subclasses as well as their strengths and weaknesses according to different areas of application. The two main classes of update protocols that can be identified are *reporting protocols*, where the location information is pushed by the mobile device, and *querying protocols*, where it is pulled by the location server.

With a *time-based reporting* protocol the location information is transmitted periodically whenever a certain time interval has elapsed. With a *distance-based reporting* protocol this is done when the mobile object has moved more than a predefined distance away from the last transmitted position. The *dead-reckoning* protocols discussed in the next section also belong to the class of reporting protocols.

Besides a *periodic querying* protocol, whose properties are similar to those of the time-based reporting protocol, the class of querying protocols includes a *cached querying* protocol. This

protocol stores the last transmitted position and returns it to a subsequent request without contacting the mobile object, when it is still considered to be accurate enough.

By means of an analysis we discuss the efficiency and effectiveness of these protocols, comparing the number of transmitted messages to the resulting minimum and average accuracy of the location information at the server for different mobility characteristics of mobile objects. Finally, we describe the results of various simulations, which we have performed based on real-world GPS traces and which confirm the outcome of our analysis. Our results show that only the distance-based reporting protocol and the cached querying protocol can guarantee a certain accuracy of the location information, as required. Of these the former is better suited for frequently queried location information and mobile objects with changing mobility characteristics, the latter for more infrequently requested location information and more steadily moving mobile objects.

Additionally, we introduce the class of *combined protocols*, which have both the advantages of reporting and querying protocols. Protocols of this class are able to adapt dynamically to different types of environments, by finding an optimal ratio between the number of position updates and the number of requests.

Dead-reckoning Protocols

A very promising approach to reduce the number of update messages when transmitting location information are so-called dead-reckoning protocols. With this approach both the mobile object and the location server that manages its location information assume the mobile object to keep on moving on a certain agreed future route. Initially, the mobile object reports its position and speed to the location server. When the location information is queried, the location server calculates the expected position on the assumed route based on the elapsed time and the last reported position and speed. The mobile object, on the other hand, monitors its actual position and only if it differs from the expected position by more than a certain desired accuracy it updates its current position and speed. Thus, a maximum deviation between the position returned by the server and the mobile object's actual position can be assured. Different dead-reckoning protocols mainly differ in the future route that is assumed for the mobile object.

In this thesis we first present a general overview and an appraisal of possible variants of dead-reckoning protocols. Based on this, we propose a new map-based dead-reckoning protocol, which uses a road map to predict the object's future movements by matching its position to a road on a map. Based on the map the protocol determines at each intersection the road the mo-

bile object is most likely to follow, and assumes that it keeps on moving on this road with its current speed.

To evaluate the efficiency of dead-reckoning protocols, we have performed extensive simulations based on real-world GPS traces with different movement characteristics of mobile objects. Our simulations show that, in case of a car on a freeway, the map-based protocol can reduce the number of necessary update messages by more than half as compared to a more simple dead-reckoning protocol, which already is a great improvement against a non dead-reckoning approach. It is also more stable for non-uniform movements (e.g., in case of a walking person), as it is able to predict the object's future course more accurately.

Conclusion

In this thesis we present the service model and the architecture for a generic distributed location service, which is suitable for large-scale deployment and for storing highly accurate location information. The service model proposed for the LS has been created especially for the highly dynamic location information of mobile objects. Its hierarchical architecture allows for an efficient processing of distributed range and nearest neighbor queries as well as of position queries and location updates. Finally, the proposed data storage component for the LS, which combines a main memory and a stable storage part, is designed to allow an efficient processing of local updates and queries. A performance evaluation of a prototype implementation of the LS shows the feasibility of our concepts. Each location server in a distributed configuration of this prototype is able to handle more than 500 location updates and distributed queries per second.

Additionally, we discuss different protocols for transmitting highly accurate location information and compare them according to their effectiveness and efficiency for characteristic types of mobile objects. With the help of the presented analysis a suitable protocol can be found for a given scenario. As an outcome of this evaluation, we also develop a new map-based dead-reckoning protocol and examine its performance compared to currently known protocols. Our simulations show that the number of update messages using this protocol can be reduced by up to 91%.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Mobilkommunikation ist in den letzten Jahren zu einem aus dem öffentlichen Leben nicht mehr wegzudenkenden Faktor geworden. So gab es laut REGTP (2001) im September 2001 alleine in Deutschland bereits über 55 Millionen Mobilfunkteilnehmer. Ab 2003 soll in Deutschland die dritte Mobilfunkgeneration (siehe z. B. MURATORE (2001)) die heute vorhandene zweite Generation (GSM) mit paketvermittelnder statt verbindungsorientierter Datenkommunikation und deutlich höheren Datenraten ablösen. UMTS wird so Datenraten von bis zu 2 MBit/s gegenüber nur 9,6 KBit/s in GSM bieten. Für diese dritte Mobilfunkgeneration werden Datendienste als ein wichtiges Gebiet für weiteres Wachstum gesehen. Besonders für die sogenannten Anwendungen mit Ortsbezug (engl. *location-aware applications*, im Mobilfunkbereich auch engl. *location-based services*, kurz LBS, genannt) wird, beispielsweise in LUTGE (2001), zukünftig ein hohes Potenzial vorhergesagt.

Einfache ortsbezogene Dienste entstehen zur Zeit bei vielen deutschen Mobilfunkbetreibern und ermöglichen beispielsweise die Suche nach einem nahegelegenen Restaurant oder das Abrufen von einfachen Weginformationen auf Basis der Funkzelle, in der sich der Benutzer aufhält (siehe z. B. VIAG INTERKOM (2001)). Eine weitere Klasse von Anwendungen, in denen die Position des Benutzers von zentraler Bedeutung ist und die sich zunehmender Beliebtheit erfreut, sind Navigationssysteme für Fahrzeuge. In naher Zukunft werden, zusätzlich zu einer flächendeckend mit hohen Datenraten verfügbaren Mobilkommunikation und dem außerhalb von Gebäuden vorherrschenden Satellitennavigationssystem GPS (HOFMANN-WELLENDORF, LICHTENEGGER & COLLINS (1997)), auch innerhalb von Gebäuden hochgenaue Positionierungssensoren zur Verfügung stehen (vgl. Abschnitt 2.1). Vor dem Hintergrund dieser Möglichkeiten bieten ortsbezogene Dienste die Grundlage für eine Reihe von fortgeschrittenen, äußerst viel

versprechenden Anwendungen aus den Bereichen Navigationssysteme, situierte Informationsräume und „*Sentient Computing*". Bei Navigationssystemen, die vor allem in Fahrzeugen bereits weit verbreitet sind, wird der Benutzer unter Berücksichtigung seiner Umgebung, worunter in Zukunft beispielsweise auch das aktuelle Verkehrsaufkommen fallen wird, zu einem meist stationären Ziel geführt (siehe z. B. ZHAO (1997)). Situierte Informationsräume, wie sie in FITZMAURICE (1993) beschrieben sind, ordnen Informationen und Dienste einem bestimmten geographischen Ort oder einem Objekt der realen Welt zu und ermöglichen es, mit diesen zu „interagieren". Die Vision von „Sentient Computing", wie sie z. B. in ADDLESEE ET AL. (2001) beschrieben ist, sieht vor, dass die Benutzungsoberfläche von Programmen in die reale Umgebung ihrer Benutzer verlagert wird. Entsprechende Programme können so Aktionen abhängig von der Interaktion eines Benutzers mit realen Objekten seiner Umwelt auslösen, wie beispielsweise seine Arbeitsumgebung auf denjenigen Rechner zu übertragen, vor den er sich gerade setzt (HARTER ET AL. (1999)).

Eine zentrale Voraussetzung für Anwendungen mit Ortsbezug ist dabei immer die Kenntnis über die geographischen Positionen der daran beteiligten mobilen Personen oder Objekte (z. B. Fahrzeuge). Der in dieser Arbeit betrachtete generische Lokationsdienst soll diese Positionsinformationen zentral für ein breites Spektrum an ortsbezogenen Anwendungen erfassen und bereitstellen. Während es für manche Anwendungen ausreichend sein kann, die aktuelle Position eines bestimmten Objekts zu erfragen, erfordern anderer Anwendungen weitergehende Funktionalität wie das Abfragen aller Objekte, die sich in einem bestimmten geographischen Gebiet aufhalten (Gebietsanfrage) oder das Ermitteln des sich zu einem bestimmten Ort am nächsten befindenden Objekts (Nachbarschaftsanfrage). Zum Beispiel könnte in einen elektronischen Stadtführer mit Ortsbezug (wie er beispielsweise in CHEVERST ET AL. (2000) beschreiben wird) ein Informationssystem für den öffentlichen Nahverkehr integriert sein, das die Verspätung eines Busses allen Benutzern meldet, die an der nächsten Bushaltestelle warten. Ein Benutzer könnte daraufhin diesen Stadtführer einsetzen, um nach dem nächsten verfügbaren Taxi zu suchen.

Wir sind der Überzeugung, dass der Ortsbezug von Anwendungen nicht wie bisher auf einzelne Einsatzgebiete innerhalb oder außerhalb von Gebäuden beschränkt bleibt, sondern dass er ein globales Ausmaß erreichen wird. Um dem Rechnung zu tragen, muss ein entsprechender Lokationsdienst auch ein breites Spektrum an Positionierungssensoren mit unterschiedlichen Einsatzgebieten unterstützen. Weiterhin nehmen wir an, dass ein global verfügbarer Lokationsdienst mit hunderttausenden von mobilen Objekten und Klienten gleichzeitig zurechtkommen muss. Die Skalierbarkeit eines solchen Dienstes wird damit zu einer zentralen Fragestellung

und macht einen verteilte Architektur des Lokationsdienstes erforderlich, welche die zu erbringende Funktionalität und die auftretende Last auf mehrere Lokations-Server aufteilt.

Während für einfache Anwendungen eine vergleichsweise ungenaue Positionsinformation ausreicht, wie z. B. für die oben erwähnten LBS-Anwendungen in heutigen Mobilfunknetzen, die eine Genauigkeit von im besten Fall wenigen hundert Metern erreichen, benötigen fortgeschrittenere Anwendungen die Informationen mit einer deutlich höheren Genauigkeit. Ein mobiles „*Augmented Reality*"-System (FEINER ET AL. (1997)) benötigt z. B. sehr genaue Positionsinformationen (im Bereich von unter einem Meter), um mit Hilfe einer halbdurchlässigen Datenbrille elektronische Informationen in die Sicht eines Benutzers auf die reale Welt einzublenden. Gleiches gilt auch für das Nexus-Projekt (HOHL ET AL. (1999)), in dessen Rahmen diese Arbeit entstanden ist und bei dem es u. a. möglich sein soll, mit einem entsprechenden Peripheriegerät, das durch einen integrierten digitalen Kompass in der Lage ist, seine räumliche Ausrichtung zu bestimmen, auf ein Objekt der realen Welt zu zeigen (z. B. ein Gebäude) und so Informationen über dieses abzufragen. Die Genauigkeit einer Positionsinformation ist damit ein wichtiges Kriterium für ihre „Verwendbarkeit" und kann daher als Maß für die Dienstgüte, die der Lokationsdienst erbringen soll, eingesetzt werden. Aus demselben Grund nimmt auch die Sicherheitsrelevanz von Positionsinformationen mit steigender Genauigkeit zu.

Verschiedene Sensorsysteme zur Positionsbestimmung unterscheiden sich in der Genauigkeit, mit der sie die Position eines mobilen Objekts ermitteln können. Während GPS (HOFMANN-WELLENDORF, LICHTENEGGER & COLLINS (1997)) eine Genauigkeit von bis zu 10 m aufweist, können Sensorsysteme innerhalb von Gebäuden eine höhere Auflösung haben. Beispielsweise besitzt das Active-Bat-System (WARD, JONES & HOPPER (1997)) eine Genauigkeit von unter 10 cm. Folglich werden die Positionsinformationen, die der Lokationsdienst verwalten soll, unterschiedliche Genauigkeiten aufweisen, wenn sie von unterschiedlichen Sensorsystemen stammen. Diese Tatsache muss beim Entwurf der Anwendungsschnittstelle (engl. *application programming interface*, kurz API) und der Mechanismen zur Anfragebearbeitung berücksichtigt werden, um es den Anwendungen zu ermöglichen, eine für die Positionsinformationen geforderte Genauigkeit anzugeben. Verschiedene Sensorsysteme liefern darüber hinaus die Positionsinformation in unterschiedlichen Formaten. Während ein Active-Badge-System (WANT ET AL. (1992)) die Position in Form eines symbolischen Bezeichners, wie beispielsweise einer Raumnummer, zurückgibt, liefert ein GPS-Empfänger eine geographische Koordinate. Der vorgeschlagene Lokationsdienst soll diese Heterogenität so weit wie möglich vor den Anwendungen verbergen.

Da es sich bei dem gegenwärtigen Aufenthaltsort einer Person um eine sensitive Information handelt, wird es schließlich für die Akzeptanz eines Lokationsdienstes von entscheidender Be-

deutung sein, Datenschutz und Datensicherheit für die von ihm verwalteten Positionsinformationen zu garantieren. Aus diesem Grund muss ein Lokationsdienst, der von unterschiedlichen Anwendungen genutzt werden soll, entsprechende Mechanismen für die Authentifizierung und Zugriffskontrolle bereitstellen (für eine ausführliche Diskussion dieser Problematik siehe auch LEONHARDT (1998)). In diesem Zusammenhang sind wir der Überzeugung, dass es wichtig ist, den Benutzern die Möglichkeit zu geben, die Genauigkeit festzulegen, mit der ihre Positionsinformation im Lokationsdienst gespeichert wird (z. B. „Ich bin im Bürogebäude" im Gegensatz zu „Ich bin in der Kaffeeecke"). Unabhängig von dem verwendeten Sensorsystem wird daraufhin ihre Positionsinformation nur noch mit dieser Genauigkeit an den Lokationsdienst übergeben. Der Benutzer soll in der Lage sein, diese Einschränkungen jederzeit entsprechend der jeweiligen Situation zu ändern. Die Genauigkeit der im Lokationsdienst gespeicherten Positionsinformationen hängt damit nicht nur von den technischen Möglichkeiten, d. h. von der Genauigkeit der verfügbaren Sensorsysteme, sondern auch von den Sicherheitsbedürfnissen seiner Benutzer und dem Vertrauen, das sie in den Dienst bzw. seinen Betreiber haben, ab.

1.2 Problemstellung und Anforderungen

Ziel dieser Arbeit ist, einen Lokationsdienst zu definieren, der speziell für die Verwaltung der hochgenauen und daher hochdynamischen Positionsinformationen einer Vielzahl von mobilen Objekten geeignet ist. Der Lokationsdienst soll als Infrastruktur für fortgeschrittene Anwendungen mit Ortsbezug dienen und muss damit neben einfachen Anfragen nach der Position eines mobilen Objekts auch Gebiets- und Nachbarschaftsanfragen beherrschen. Weiterhin muss auf die speziellen Eigenschaften mobiler Endgeräte Rücksicht genommen werden, die zu einem großen Teil als mobile Objekte und Klienten des Lokationsdienstes fungieren werden. Klienten des Lokationsdienstes können dabei sowohl einzelne Benutzeranwendungen, als auch ortsbezogene Informationsdienste sein, die verschiedene Benutzer mit Informationen versorgen.

Ausgehend von dem beschriebenen Einsatzzweck und den geforderten Zielen ergeben sich für den Lokationsdienst die folgenden speziellen Anforderungen:

- Genauigkeit der Positionsinformationen: Der Lokationsdienst muss in der Lage sein, Positionsinformationen mit unterschiedlicher und z. T. sehr hoher Genauigkeit zu speichern (im Bereich von wenigen Metern oder darunter). Konkret muss sich die unterschiedliche Genauigkeit der Positionsinformationen in der Anwendungsschnittstelle (API) widerspiegeln. Für die Behandlung von Positionsinformationen mit hoher Genauigkeit muss er darüber hinaus mit einer häufigen Aktualisierung dieser Informationen zurechtkommen, da ein

unmittelbarer Zusammenhang zwischen der Aktualisierungsrate und der Genauigkeit der Positionsinformationen besteht (siehe auch Kapitel 6).

- Skalierbarkeit und Effizienz des Dienstes: Um wie gefordert die Positionen einer Vielzahl von mobilen Objekten (z. B. hunderttausende von Fahrzeugen in einer größeren Stadt) mit hoher Genauigkeit zu speichern und dementsprechend auch mit vielen Klienten zurechtzukommen, muss ein wichtiges Ziel beim Entwurf des Lokationsdienstes die Skalierbarkeit von dessen Architektur und Mechanismen sein. Die erwartete große Anzahl mobiler Objekte und Klienten macht wiederum eine effiziente Bearbeitung von Anfragen und Aktualisierungsnachrichten erforderlich.
- Datenschutz und Datensicherheit: Der Lokationsdienst muss Datenschutz und Datensicherheit für die von ihm verwalteten Positionsinformationen garantieren. Dazu sind erstens entsprechende Mechanismen anzubieten, mit deren Hilfe ein Benutzer steuern kann, welche Anwendungen oder Personen mit welcher Genauigkeit Zugriff auf seine Positionsinformationen bekommen sollen. Zweitens soll der Benutzer steuern können, mit welcher Genauigkeit der Lokationsdienst selbst seine Positionsinformationen erhält, was sich auch in dessen API widerspiegeln muss.
- Fehlertoleranz: Da der Lokationsdienst in der hier anvisierten Form eine wichtige Ressource für Anwendungen mit Ortsbezug darstellt, muss er in der Lage sein, fehlertolerant gegenüber dem Auftreten von Fehlerfällen zu reagieren. Es sind daher Mechanismen bereitzustellen, die dafür sorgen, dass dessen Funktionalität und die gespeicherten Positionsinformationen das Auftreten von Fehlerfällen überdauern.

Die Betrachtung existierender Ansätze zur Speicherung von Positionsinformationen in Kapitel 2 wird zeigen, dass bislang keiner dieser Ansätze die hier gestellten Anforderungen in befriedigender Weise erfüllen kann.

1.3 Struktur der Arbeit

In dieser Arbeit werden Mechanismen und Konzepte für einen verteilten Lokationsdienst vorgestellt, die speziell für die Verwaltung von hochdynamischen Positionsinformationen und eine große Zahl von mobilen Objekten benötigt werden. Der Aufbau dieser Ausarbeitung soll nun kurz dargestellt werden.

Kapitel 2 gibt zuerst einen Überblick über verwandte Arbeiten und grenzt sie von den im weiteren beschriebenen Arbeiten, mit dem Ziel der Entwicklung eines allgemeinen Lokationsdienstes, ab. Für ein besseres Verständnis wird darauf folgend der Hintergrund dieser Arbeit, die im

Rahmen des Nexus-Projekts (NEXUS (2002)) an der Universität Stuttgart entstanden ist, in Kapitel 3 kurz dargestellt.

In Kapitel 4 wird ein generisches Modell für einen Lokationsdienst (bzw. eine generische API) erarbeitet, das auf den oben beschriebenen Einsatzzweck und die gestellten Anforderungen ausgelegt ist. Dieses Modell beschreibt die Semantik der Registrierung mobiler Objekte, der von Positionsaktualisierungen sowie von Positions-, Gebiets- und Nachbarschaftsanfragen unter Berücksichtigung einer durch die Genauigkeit der Positionsinformationen gegebenen Dienstgüte.

Kapitel 5 stellt eine hierarchische Architektur vor, anhand der ein skalierbarer effizienter Lokationsdienst entsprechend dem generischen Dienstmodell aus Kapitel 4 implementiert werden kann. Die dafür notwendigen Algorithmen für Registrierung und Positionsaktualisierungen sowie für die Bearbeitung von Positions-, Gebiets- und Nachbarschaftsanfragen werden ausführlich dargestellt und diskutiert. Beim Entwurf der Architektur und Algorithmen wird von einer deutlichen Lokalität der Anfragen ausgegangen, d. h. ein Benutzer des Lokationsdienstes interessiert sich häufig für mobile Objekte und Gebiete in seiner näheren Umgebung und weniger häufig für solche, die weiter entfernt sind.

Für die Datenhaltungskomponente des Lokationsdienstes wird ebenfalls in Kapitel 5 ein Realisierungsansatz vorgestellt, bei dem eine Hauptspeicherdatenbank für die Speicherung der dynamischen Positionsinformationen verwendet wird, wodurch Anfragen und insbesondere Positionsaktualisierungen sehr effizient bearbeitet werden können. Mechanismen zur Wiederherstellung der Positionsinformationen stellen sicher, dass diese auch nach Ausfall einer Komponente schließlich wieder verfügbar werden.

Aktualisierungsprotokolle, die für die Übertragung hochdynamischer Positionsinformationen geeignet sind, werden unter diesem Gesichtspunkt umfassend in Kapitel 6 diskutiert und ihre Eigenschaften mittels Analyse und Simulation bestimmt und verglichen. Darauf basierend wird ein kombiniertes Protokoll vorgeschlagen, das die Eigenschaften unterschiedlicher Protokollklassen verbindet und an verschiedenartige Einsatzumgebungen angepasst werden kann. Die viel versprechende Klasse der Koppelnavigationsprotokolle wird genauer betrachtet und ein neuartiges kartenbasiertes Koppelnavigationsprotokoll vorgestellt. Während ein einfaches Koppelnavigationsprotokoll gegenüber einem vergleichbaren Nicht-Koppelnavigationsverfahren bereits bis zu 83% der Aktualisierungsnachrichten einspart, reduziert das kartenbasierte Verfahren diese um weitere bis zu 60%.

In Kapitel 7 werden die Messergebnisse beschrieben, die wir durch Experimente mit einer prototypischen Implementierung des Lokationsdienstes erhalten haben. Sie zeigen die Machbarkeit und Leistungsfähigkeit der Datenhaltungskomponente sowie der verteilten Architektur des Lo-

kationsdienstes. So können bereits auf einem einfachen PC fast 500 Positionsaktualisierungen oder Positionsanfragen pro Sekunde abgearbeitet werden.

Schließlich werden in Kapitel 8 die Sicherheitsaspekte, die im Zusammenhang mit dem Lokationsdienst entstehen, betrachtet und die Schnittstelle und Mechanismen für eine einfache Zugriffskontrolle vorgestellt. Eine ausführliche Betrachtung der Sicherheitsproblematik der Positionsinformationen mobiler Benutzer wird von unseren Projektpartnern am Institut für Kommunikationsnetze und Rechnersysteme (IKR) der Universität Stuttgart durchgeführt (siehe auch HAUSER & KABATNIK (2001)) und ist nicht Fokus dieser Arbeit. Kapitel 9 beschließt die Arbeit mit einer Zusammenfassung und einem Ausblick auf mögliche weiterführende Arbeiten.

Kapitel 2

Stand der Wissenschaft

Der in dieser Arbeit behandelte Lokationsdienst steht in Bezug zu einer Reihe von Forschungsgebieten aus den Bereichen Mobilkommunikation und Datenbanken. Dieses Kapitel gibt einen Überblick über den Stand der Wissenschaft in den betroffenen Forschungsgebieten und ordnet den in dieser Arbeit beschriebenen Lokationsdienst in den jeweiligen Kontext ein (für einen Überblick über verwandte Arbeiten im Bereich Mobilkommunikationssysteme siehe auch PITOURA & SAMARAS (2001)).

Die für den Lokationsdienst notwendigen Teilfunktionalitäten lassen sich in die folgenden vier Aufgabenbereiche einteilen: Erstens die Bestimmung der Positionsinformation durch einen geeigneten Positionierungssensor (was hier nicht Gegenstand der Forschung sein soll). Zweitens die effiziente Übertragung dieser Information von dem Positionierungssensor zu einem entsprechenden Lokations-Server und von dort weiter zu einem Klienten, der die Information anfragt. Drittens die Speicherung der Positionsinformation sowie die Bearbeitung entsprechender Anfragen auf einem einzelnen Lokations-Servern und viertens Algorithmen für das Auffinden der angefragten Positionsinformation in einem verteilten System von Lokations-Servern. Abbildung 2-1 setzt die in diesem Kapitel besprochenen verwandten Forschungsgebiete in Beziehung zu den jeweiligen Aufgabenbereichen.

2.1 Sensoren zur Positionsbestimmung

Verschiedenste Sensorsystemen sind in den letzten Jahren mit dem Ziel einer möglichst genauen und zuverlässigen Bestimmung der aktuellen Position eines mobilen Objekts entwickelt worden (für einen Überblick über verfügbare Positionierungssensoren siehe z. B. HIGHTOWER &

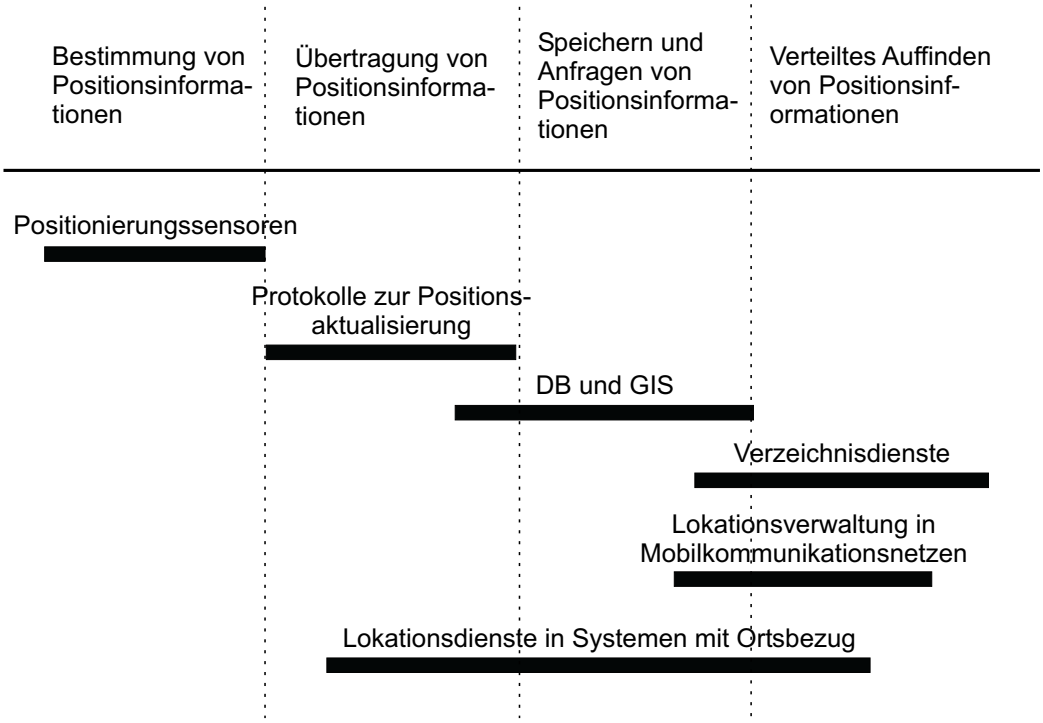


Abbildung 2-1. Übersicht über verwandte Arbeiten.

BORRIELLO (2001)). In diesem Abschnitt soll ein Überblick über die dazu eingesetzten Technologien gegeben und jeweils deren wichtigste Vertreter vorgestellt werden, ohne dass ein Anspruch auf Vollständigkeit erhoben wird.

Bei Systemen zur Positionsbestimmung kann nach LEONHARDT (1998) unterschieden werden zwischen *lokalen* Sensorsystemen (engl. *positioning systems*), die auf dem mobilen Objekt selbst die eigene Position bestimmen, und *externen* Systemen (engl. *tracking systems*), bei denen eine externe Sensorinfrastruktur benötigt wird. Ein weiteres wichtiges Unterscheidungsmerkmal ist das Format der zurückgelieferten Positionsinformationen. Dies kann einerseits ein *symbolischer* Bezeichner für die Position sein, wie eine Raumnummer, andererseits auch ein *geometrischer* Bezeichner entsprechend einem lokalen oder globalen Koordinatensystem, wie WGS84 (NIMA (1997)).

Ziel dieser Arbeit ist nicht, Forschung auf dem Bereich der Positionierungssensoren zu betreiben, sondern sie setzt auf die in diesem Bereich erzielten Ergebnisse auf. Ohne eine zusätzliche Infrastruktur ist die Ausgabe eines Positionierungssensors allerdings nur lokal auf dem mobilen Objekt selbst (bei einem internen System) oder für eine lokale Installation (bei einem externen System) verfügbar. Der Lokationsdienst soll eine Infrastruktur bereitstellen, welche die Positi-

onsinformationen der Sensorsysteme global verfügbar macht und integriert dazu die von beliebigen Sensorsystemen gelieferten Positionsinformationen.

2.1.1 Positionsbestimmung außerhalb von Gebäuden

Außerhalb von Gebäuden hat sich für die Positionsbestimmung das Satellitenpositionierungssystem GPS (engl. *global positioning system*, siehe HOFMANN-WELLENDORF, LICHTENEGGER & COLLINS (1997)) weitgehend durchgesetzt. Es handelt sich dabei um ein lokales System, das geometrische Positionsinformationen in einem durch das globale Koordinatensystem WGS84 definierten Format zurückliefert. GPS ist weltweit verfügbar und erreicht, nach der Abschaltung der künstlichen Unschärfe (engl. *selective availability*) durch den Betreiber – das Verteidigungsministerium der Vereinigten Staaten von Amerika – im Mai 2000, eine Genauigkeit von 10 bis 20 m. Durch den Einsatz von differentiellem GPS (DGPS) kann diese Genauigkeit auf bis zu 1 m, mit größerem Aufwand sogar auf bis zu 10 cm erhöht werden. Innerhalb von Gebäuden funktioniert GPS jedoch auf Grund des verwendeten Frequenzbandes nicht.

Für Mobiltelefone wurden, um den Aufwand durch einen zusätzlichen GPS-Empfänger zu sparen, verschiedene Systeme entwickelt, bei denen deren Position aus der Kommunikationsinfrastruktur u. a. durch Messung der Signallaufzeiten zu nahegelegenen Basisstationen (*enhanced observed time difference*, kurz E-OTD) bestimmt werden kann. Das Cursor-System (siehe CAMBRIDGE POSITIONING SYSTEMS (2001)) benötigt zu diesem Zweck entsprechende Modifikationen an den Geräten sowie der Infrastruktur des Mobilkommunikationssystems und erreicht eine Genauigkeit zwischen 15 und 150 Metern.

Im Rahmen einer Diplomarbeit hat Christian Seybold (SEYBOLD (2001)) untersucht, wie eine Positionsbestimmung in GSM durchgeführt werden kann, bei der keine Modifikation an aktueller Netzinfrastruktur oder Mobiltelefonen vorgenommen werden muss. Die Ergebnisse zeigen, dass unter diesen Bedingungen im Wesentlichen nur die Bestimmung eines Sektors der aktuell verwendeten Funkzelle mit einer Genauigkeit von 500 bis 1500 m im Stadtbereich möglich ist.

Die Positionsbestimmung über GSM funktioniert prinzipiell sowohl innerhalb als auch außerhalb von Gebäuden, allerdings erreicht sie nicht die für Anwendungen in Gebäuden erforderliche Genauigkeit.

2.1.2 Positionsbestimmung innerhalb von Gebäuden

Die genaue Positionsbestimmung von mobilen Objekten innerhalb von Gebäuden ist hingegen immer noch Gegenstand der Forschung, wobei eine Vielzahl von unterschiedlichen Technologien untersucht wurde. Erste Ansätze haben zu diesem Zweck Infrarotsender verwendet, welche periodisch ein eindeutiges Signal aussenden, sowie in den Räumen platzierte Empfänger, die diese Signale empfangen und an eine zentrale Verwaltung weiterleiten. An der Universität Cambridge wurde so das bekannte Active-Badge-System entwickelt (WANT ET AL. (1992)), bei Xerox das ParcTab-System (SCHILIT, ADAMS & WANT (1994)). Diese externen Systeme liefern als Ergebnis den Aufenthaltsort eines mit einem entsprechenden Sender ausgestatteten mobilen Objekts als (symbolische) Raumnummer.

Genau entgegengesetzt funktionieren Systeme, die auf sogenannten Infrarot-Baken basieren. Dort sendet ein einfacher Infrarotsender periodisch die Kennung für seinen Standort aus, oft ebenfalls eine Raumnummer. Diese Kennung wird entweder, wie bei dem vom MIT Media Lab entwickelten Locust-System (LOCUST (2001)), über ein spezielles Peripheriegerät eingelesen, mit dem das mobile Endgerät des Benutzers ausgestattet ist, oder direkt durch die in vielen Geräten schon eingebaute IRDA-Schnittstelle, wie bei dem Eyeled-System (BUTZ ET AL. (2001)). Die mit Infrarot-Baken arbeitenden Systeme haben den Vorteil, dass im Gegensatz zu den auf der Active-Badge-Idee basierenden Systemen, die Position des Benutzers nicht außerhalb des eigenen Endgeräts bekannt wird und daher meist weniger Sicherheitsbedenken gegenüber diesen Systemen bestehen.

Ein aktuellerer Ansatz ist das in WARD, JONES & HOPPER (1997) beschriebene Active-Bat-System (ein Nachfolger der Active-Badges). Dieses basiert auf Ultraschallempfängern, die im Abstand von etwa einem Meter in die Decke eines Raumes eingelassen sind, und kann die Position eines kleinen Ultraschallsenders bis auf 3 cm genau bestimmen. Für das deutlich weniger aufwändige RADAR-System, das die Position aufgrund der Signalstärke der Basisstationen eines Funk-LANs bestimmt, berichten BAHL & PADMANABHAN (2000) von einer durchschnittlichen Genauigkeit von etwa 3 m. Das verwandte Cricket-System (siehe z. B. PRIYANTHA, CHAKRABORTY & BALAKRISHNAN (2000)) verwendet sowohl Funk als auch Ultraschall, wodurch ein mobiles Objekt durch Betrachtung der Laufzeit der Ultraschallsignale einzelne Teilbereiche eines Raumes bis auf wenige Zentimeter genau unterscheiden kann.

2.2 Protokolle zur Positionsaktualisierung

Um die Positionsinformationen von einem Positionierungssensor zum Lokationsdienst zu übertragen, wird ein geeignetes Protokoll benötigt, das den Austausch dieser Informationen steuert. Da sich die Positionsinformationen ständig ändern können und die Übertragung mit einer bestimmten Genauigkeit stattfinden sowie möglichst wenig Nachrichten benötigen soll, wird ein Protokoll benötigt, das auf die speziellen Eigenschaften von Positionsinformationen zugeschnitten ist. Herkömmliche Protokolle sind – wie sich zeigen wird – nicht für die Übertragung von hochdynamischen und sehr genauen Positionsinformationen geeignet.

2.2.1 Allgemeine Aktualisierungsprotokolle

Das Verhalten von allgemeinen Aktualisierungsprotokollen für Kopien mit schwacher Konsistenz (engl. *weak consistency*), die nicht auf Positionsinformationen mobiler Objekte spezialisiert sind, wurde ausführlich im Forschungsgebiet der Verteilten Systeme untersucht. So wird in ALONSO, BARBARA & GARCIA-MOLINA (1990) der Begriff der Quasi-Kopie eingeführt, bei der die Konsistenz bestimmten durch Prädikate formulierten Einschränkungen unterliegt. In KOVACS (1999) werden entsprechende Aktualisierungsprotokolle z. B. für Daten betrachtet, die für die Verwaltung von Diensten in verteilten Systemen benötigt werden.

Die in Kapitel 6 angestellten Betrachtungen beruhen darüber hinausgehend sehr stark auf den speziellen Eigenschaften der Positionsinformationen mobiler Objekte. So ist, aufgrund der sich potenziell ständig bewegenden Objekte und der Ungenauigkeit der Sensorinformationen, von vorneherein nie eine vollständige Aktualität und Genauigkeit der Informationen zu erreichen. Weiterhin ergeben sich Besonderheiten aufgrund der durch die maximale Geschwindigkeit beschränkten Änderungen und die oft bestimmten Mustern folgenden Bewegungen der mobilen Objekte, die z. B. beim Zwischenspeichern von Positionsinformationen (siehe Abschnitt 6.2.1) und bei den Koppelnavigationsprotokollen (siehe Abschnitt 6.3) ausgenutzt werden.

2.2.2 Lokationsverwaltung in Mobilkommunikationssystemen

Bei der Lokationsverwaltung für Mobiltelefone in Mobilkommunikationssystemen (wie z. B. GSM, MOULY & PAUTET (1992)) ist die Aktualisierung der Positionsinformationen ein wichtiger Aspekt. Wenn ein Anruf an ein Mobiltelefon weitergeleitet werden soll, wird dieses auf einem speziellen Kanal in allen Funkzellen ausgerufen (engl. *paging*), in denen sich das Mobiltelefon aufhalten könnte. Das angesprochene Mobiltelefon meldet sich daraufhin und nimmt

den Anruf entgegen. Um die Zahl der in Frage kommenden Zellen zu reduzieren, übermittelt das Mobiltelefon regelmäßig – entsprechend einer festgelegten Strategie – die Funkzelle, in der es sich gerade aufhält, an die Mobilkommunikationsinfrastruktur. Diese Positionsinformation wird daraufhin in dem zum Mobiltelefon gehörenden Heimatregister (engl. *home location register*, kurz HLR) und in dem dieser Funkzelle zugeordneten Besucherregister (engl. *visitor location register*, kurz VLR) gespeichert.

Im Zusammenhang der Lokationsverwaltung für Mobilkommunikationssysteme (vgl. auch Abschnitt 2.5) wurden unterschiedliche Strategien zur Positionsaktualisierung betrachtet, mit dem Ziel, die Gesamtkosten für Positionsaktualisierungs- und Ausrufnachrichten zu minimieren. BAR-NOY, KESSLER & SIDI (1995) vergleichen ein entfernungsbasiertes (engl. *distance-based*), ein bewegungsbasiertes (engl. *movement-based*) und ein zeitbasiertes (engl. *time-based*) Protokoll zur Positionsaktualisierung, wobei sich ersteres als am besten geeignet für die gegebene Aufgabe erweist. Bei dem entfernungsbasierten Protokoll wird eine Aktualisierungsnachricht gesendet, sobald eine bestimmte Anzahl von Funkzellen zwischen der aktuellen und der zuletzt gesendeten Zelle liegt. Auf ähnliche Weise funktioniert das bewegungsbasierte Protokoll, bei dem dazu eine bestimmte Anzahl von Zellgrenzen überschritten werden muss, die aber durchaus immer zu denselben Zellen gehören können. Das zeitbasiertes Protokoll schließlich sendet eine Aktualisierungsnachricht jeweils nach einem vorgegebenen Zeitintervall.

Protokolle zur Positionsaktualisierung in Mobilkommunikationssystemen nutzen oft die speziellen Eigenschaften aus, die sich aus der Zellstruktur dieser Systeme ergeben. So wird in BHATTACHARYA & DAS (1999) ein "*LeZi-update*" genannter Ansatz vorgestellt, der die zu übertragenden Bezeichner der Funkzellen als Alphabet verwendet und häufig auftretende Kombinationen von nacheinander besuchten Funkzellen, analog zu dem bekannten Lempel-Ziv-Komprimierungsverfahren, in einzelnen Aktualisierungsnachrichten zusammenfasst. Dies geschieht jeweils pro Benutzer, so dass das System häufig verwendete Wege lernen und auch vorhersagen kann. In dem Artikel wird auch gezeigt, dass dieses Verfahren eine annähernd optimale Lösung in Bezug auf die übertragene Datenmenge darstellt.

In aktuellen Mobilkommunikationssystemen wie GSM wird eine Gruppe von angrenzenden Funkzellen zu einem sogenannten Aufenthaltsbereich (engl. *location area*) zusammengefasst. Positionsaktualisierungen werden jedesmal gesendet, wenn ein Mobiltelefon von einem Aufenthaltsbereich in einen anderen wechselt. Verschiedene Forschungsarbeiten (u. a. ABUTALEB & LI (1997)) beschäftigen sich mit der Ermittlung einer optimalen Form und Größe für diese Aufenthaltsbereiche, abhängig von den zu erwartenden Bewegungs- und Anrufmustern. Für aktuelle Mobilkommunikationssysteme existieren auch bereits Erweiterungen für die dort aufkommenden Anwendungen mit Ortsbezug, die Positionsinformationen mit einer höheren Ge-

nauigkeit bestimmen und speichern können. Zur Übertragung der Positionsinformationen werden allerdings bisher nur einfache anfragende oder zeitbasierte Protokolle verwendet (siehe LIF (2001)).

Im Gegensatz zu der in Kapitel 6 betrachteten Problemstellung ist bei Mobilkommunikationssystemen ohne eine vorherige Übertragung der Positionsinformationen keine direkte Kontaktaufnahme von der Infrastruktur aus mit den mobilen Endgeräten möglich, da deren aktuelle Funkzelle ja erst durch diese Mechanismen bestimmt werden soll. Damit können auch keine Protokolle (siehe unten) verwendet werden bei denen die Positionsaktualisierung von der Kommunikationsinfrastruktur angestoßen wird. Obwohl die Ansätze, die von den mobilen Endgeräten ausgehen, oft ähnlich sind, wird in unserer Aufgabenstellung kein Ausrufen (*paging*) des mobilen Endgeräts benötigt, was dazu führt, dass die Ergebnisse der Untersuchungen an Mobilkommunikationssystemen nicht direkt übertragen werden können. Schließlich basiert die Positionsaktualisierung bei Mobilkommunikationssystemen auf den symbolischen Bezeichnern der Aufenthaltsbereiche, die oft die Größe einer Stadt (bis 35 km) haben. Der LS verwendet dagegen geometrische Koordinaten, die eine sehr hohe Genauigkeit im Bereich von wenigen Metern oder darunter aufweisen können.

2.2.3 Koppelnavigationsprotokolle

In WOLFSON ET AL. (1999) wird erstmals ein als Koppelnavigationsstrategie (engl. *dead-reckoning)* bezeichnetes Verfahren beschrieben. Dieses wird im Rahmen einer Datenbank für mobile Objekte (siehe Abschnitt 2.3) zur Reduzierung der für die Aktualisierung von Positionsinformationen benötigten Nachrichten eingesetzt. Dabei wird angenommen, dass der Weg, auf dem sich das betrachtete mobile Objekt bewegt, im Voraus bekannt ist. Die Datenbank nimmt solange an, dass sich das Objekt auf diesem Weg mit der zuletzt gemeldeten Geschwindigkeit weiterbewegt, bis sie eine Aktualisierungsnachricht mit dessen aktueller Position und Geschwindigkeit erhält. Das mobile Objekt versendet eine solche Aktualisierungsnachricht, wenn sich durch eine Geschwindigkeitsänderung die Entfernung zwischen der realen Position und der von der Datenbank angenommenen einen bestimmten Entfernungsschwellwert überschreitet. Zur Bestimmung eines optimalen Schwellwerts wird eine Kostenfunktion definiert, die sowohl Kosten für den Entfernungsschwellwert und für die Abweichung zwischen realer und angenommener Position als auch Kosten für die versendeten Aktualisierungsnachrichten beinhaltet.

Verschiedene Varianten von Koppelnavigationsstrategien werden mit dem Ziel, diese Kostenfunktion zu minimieren, miteinander verglichen. Dabei zeigt sich, dass es sich unter diesen Bedingungen im Vergleich zu einfacheren Verfahren (engl. *speed dead-reckoning policy)* lohnt,

den Entfernungsschwellwert adaptiv anzupassen (engl. *adaptive dead-reckoning policy*). Um mit Unterbrechungen der Netzverbindung umgehen zu können, wird bei einer weiteren Variante der Schwellwert kontinuierlich verringert, falls keine weitere Aktualisierungsnachricht versendet bzw. empfangen wird (engl. *disconnection detection dead-reckoning policy*). Nicht Teil der Arbeiten von WOLFSON ET AL. (1999) ist, wie das mobile Objekt seine (zukünftige) Wegstrecke bestimmt.

Das in Abschnitt 6.3.2 vorgestellte kartenbasierte Koppelnavigationsprotokoll (engl. *map-based dead-reckoning*), das ausführlich in NICU (2001) beschrieben ist, bildet die Bewegung eines Objekts hingegen auf ein gegebenes Wegenetz ab und setzt daher im Gegensatz zu den oben beschriebenen Ansätzen nicht voraus, dass der Weg, auf dem sich das mobile Objekt bewegt, oder sein Zielort im Voraus bekannt ist. Des Weiteren wird ein allgemeinerer Überblick über die möglichen Klassen von Koppelnavigationsprotokollen gegeben und das Verhalten eines einfachen und eines kartenbasierten Protokolls ausführlich für unterschiedliche Bewegungscharakteristika mobiler Objekte mittels Simulationen untersucht und mit denen von Nicht-Koppelnavigationsprotokollen verglichen.

Ursprünglich stammt der Begriff Koppelnavigation aus der Schifffahrt, wo er für das Verfahren steht, die aktuelle Position anhand von einer bekannten Ausgangsposition und einem ständigen Aufaddieren der durch aktuellen Kurs, Geschwindigkeit, Abdrift etc. gegebenen Abweichungen zu bestimmen. Entsprechende Verfahren werden heutzutage auch in der Robotik (z. B. KLEEMANN (1992)), bei Luftaufnahmen oder bei der Fahrzeugnavigation eingesetzt, wo ein primärer Positionssensor (zumeist GPS) durch sekundäre Sensoren, wie Geschwindigkeitsmesser oder Inertialsensoren, die keine Position sondern nur Abweichungen von dieser liefern, unterstützt wird (siehe SKALOUD & SCHWARZ (2000)). Diese sekundären Sensorinformationen werden verwendet, um die Genauigkeit der Positionierung zu erhöhen oder wenn der Primärsensor kurzzeitig nicht verfügbar ist (z. B. weil der Benutzer ein Gebäude betreten hat). Allerdings addieren sich bei diesen Verfahren die Messfehler der Sekundärsensoren auf, so dass nach einer gewissen Zeit wieder auf den Primärsensor zugegriffen werden muss.

All diesen Verfahren ist jedoch gemeinsam, dass sie ständig auf die aktuellen Werte der Sekundärsensoren zugreifen müssen. Sie sind im Rahmen der hier betrachteten Problemstellung also nur für die Erhöhung der Genauigkeit bzw. Verfügbarkeit der Positionsinformationen geeignet und nicht für die Reduzierung der zur Übertragung benötigten Aktualisierungsnachrichten.

2.3 Räumliche Datenbanken und Geographische Informationssysteme

Ein einfacher Ansatz für die Realisierung der Datenhaltungskomponente eines Lokationsdienstes wäre, die Positionsinformationen in einem als Produkt verfügbaren relationalen Datenbanksystem zu speichern. Mit einer einfachen Datenbank lassen sich allerdings nur Positionsanfragen effizient durchführen. Um räumliche Anfragen wie Gebiets- und Nachbarschaftsanfragen effizient bearbeiten zu können, wird eine spezielle Erweiterung benötigt (wie der *Spatial Extender* (DAVIS (1998)) für IBM's DB2 oder die *Spatial Cartridge* (ORACLE (1997)) für Oracle Datenbanken), die eine spezielle mehrdimensionale Indexstruktur realisiert (für einen Überblick über mehrdimensionale Indexstrukturen siehe GAEDE & GÜNTHER (1998)). Die als Produkte verfügbaren Erweiterungen für räumliche Daten besitzen allerdings nicht die für den Lokationsdienst benötigte Leistungsfähigkeit und sind zum Teil noch nicht voll ausgereift bzw. nicht für alle Plattformen implementiert (in Kapitel 7 sind entsprechende Messergebnisse gezeigt).

Geographische Informationssysteme (GIS) kombinieren eine Datenbank mit Karteninformationen und sind daher speziell für die Verarbeitung von räumlichen Daten ausgelegt (siehe WORBOYS (1995) oder OPEN GIS (2001)). Wie bei den entsprechenden Erweiterungen für Datenbanken auch, sind kommerzielle Produkte (wie z. B. ArcGIS, ESRI (2001)) für die Verarbeitung von großen Datenmengen und komplexere Anfragen ausgelegt und bieten für die einfachen Anfragen des Lokationsdienstes nicht den erwünschten Durchsatz. Darüber hinaus sind beide Lösungen im Wesentlichen auf einzelne Installationen beschränkt und ermöglichen keine Verteilung der gespeicherten Daten.

Aktueller Gegenstand der Forschung im Bereich räumlicher Datenbanken sind Mechanismen, die speziell auf die Verwaltung der Daten mobiler Objekte ausgerichtet sind (engl. *moving objects database*, kurz MOD) und neben beschreibenden Attributen auch das Speichern einer aktuellen Position ermöglichen. Entsprechende Datenbanken werden z. B. für das Flottenmanagement in Transportunternehmen eingesetzt, wo sie den aktuellen Aufenthaltsort der Fahrzeuge verwalten. Untersucht werden auf diesem Gebiet die Definition eines geeigneten Datenmodells, die Erweiterung von Anfragesprachen um räumliche und zeitliche Aspekte sowie Protokolle für das Aktualisieren der Informationen (ein Überblick über die Thematik zu MODs wird in WOLFSON ET AL. (1998) gegeben). Wie bereits erwähnt (siehe Abschnitt 2.2.3) betrachtet WOLFSON ET AL. (1999) im Zusammenhang mit MODs auch erstmals verschiedene Möglichkeiten, um die Anzahl der für die Aktualisierung von Positionsinformationen benötigten Nachrichten durch Koppelnavigationsmechanismen zu reduzieren.

Eine Erweiterung davon sind sogenannte Raum-Zeit-bezogene Datenbanken (engl. *spatio-temporal databases*), die zusätzlich auch die Änderungen von räumlichen Informationen, wie z. B. der Positionsinformationen mobiler Objekte, über die Zeit betrachten. Sie erlauben somit Anfragen hinsichtlich der „Geschichte" der Bewegungen eines mobilen Objekts (siehe z. B. FORLIZZI ET AL. (2000)). Betrachtet werden hier neben Datenmodellen und Anfragesprachen vor allem geeignete Indexstrukturen, (z. B. in TAYEB, ULUSOY & WOLFSON (1998) oder SALTENIS ET AL. (2000)).

Bei der aktuellen Forschung zu MODs und Raum-Zeit-bezogenen Datenbanken stehen Fragen der Datenhaltung im Vordergrund. Aspekte der Verteilung oder Föderation, die für die Realisierung eines Lokationsdienstes mit vielen mobilen Objekten und häufigen Aktualisierungen notwendig sind, wurden bisher kaum betrachtet (vgl. auch PITOURA & SAMARAS (2001)). Des Weiteren wird meist davon ausgegangen, dass es sich um einen nicht-öffentlichen Dienst handelt und daher keine speziellen Fragen des Datenschutzes und der Datensicherheit (z. B. die Berücksichtigung einer vom Benutzer vorgegebenen maximalen Genauigkeit) auftreten.

2.4 Verzeichnisdienste

Verzeichnisdienste (engl. *directory services*) werden dazu verwendet, um Informationen über vorhandene Diensterbringer – in vielen Fällen sind dies gleichzeitig auch Peripheriegeräte, wie ein Drucker oder ein Videoprojektor – zu verwalten und z. B. einen gewünschten Dienst anhand von beschreibenden Attributwerten aufzufinden. Bei mobilen Geräten ist dabei oft die Position des Anfragenden und die des Diensterbringers ein wichtiges Suchkriterium. Ein häufig verwendetes Beispiel hierfür ist die Suche nach dem nächstgelegenen Drucker in einem fremden Bürogebäude. Positionsinformationen können daher ein Teil der Dienstbeschreibung sein und bestehen meist aus Gebäudeadresse und Raumnummer.

Ein einfacher Ansatz für die Zuordnung von Positionsinformationen zu Rechnern ist die in DAVIS ET AL. (1996) beschriebene Erweiterung des *Domain Name Systems* (DNS, siehe MOCKAPETRIS & DUNLAP (1988)) um einen zusätzlichen Eintrag (LOC), der die geographische Position des betreffenden Rechners enthält. Für einen Rechner oder Drucker, dessen DNS-Name bekannt ist, kann somit über das DNS auch dessen geographische Position abgefragt werden.

Für mobile Geräte, die entweder als Dienstnehmer oder Diensterbringer agieren, ist es wichtig Informationen über die in ihrer (sich ständig ändernden) Umgebung vorhandenen Ressourcen zu erhalten. Es ist daher eine Vielzahl von Systemen entstanden, die für das Auffinden von

Diensten (engl. *service discovery)* speziell in einer Umgebung mit mobilen Dienstnehmern und Diensterbringern geeignet sind.

Systeme für das lokale Auffinden von Diensten, wie das auf die Programmiersprache Java (siehe SUN (2001)) aufbauende *Jini* oder das von Microsoft entwickelte *Universal Plug and Play* (kurz UPnP), verwenden für das Ankündigen und Auffinden von Diensten eine Multicast-Nachricht, deren Lebensdauer (engl. *time to live*, kurz TTL) auf wenige Teilstrecken begrenzt ist. Damit werden nur Dienste in der näheren Umgebung des Anfragenden gefunden. Bei Jini müssen die für das Auffinden der Dienste verwendeten Dienstbeschreibungen immer von einem oder mehreren lokalen *Lookup-Services* verwaltet werden, bei UPnP können Dienstnehmer und Diensterbringer auch direkt kommunizieren. Für eine detaillierte Beschreibung der jeweiligen Mechanismen siehe JINI (2001) oder UPNP (2001).

Mit dem durch die IETF in GUTTMAN ET AL. (1999) standardisierten *Service Location Protocol* (SLP) werden Dienste, die durch frei zu vergebende Attribut-Wert-Paare beschrieben sind, ebenfalls mit Hilfe einer Multicast-Anfrage gesucht. Ein Dienst meldet sich dabei auf alle Anfragen, die er über eine wohlbekannte Multicast-Adresse empfängt und deren Suchprofil seiner Dienstbeschreibung entspricht. Aus Gründen der Skalierbarkeit wurden sogenannte Verzeichnisagenten (vergleichbar mit den Lookup-Services von Jini) eingeführt, bei denen sich die Dienste anmelden und die von den Klienten abgefragt werden können. Ein geeigneter Verzeichnisagent wird ebenfalls über eine lokale Multicast-Anfrage bestimmt. Um den Suchbereich einzuschränken, kann die Einsatzumgebung des SLP weiterhin in einzelne Gültigkeitsbereiche (engl. *scope*) unterteilt werden. Ein Dienst muss sich dazu bei allen Verzeichnisagenten anmelden, die denselben Gültigkeitsbereich haben. Die Zuordnung eines Dienstes zu einer geographischen Position ist, wie bei den folgenden Diensten auch, nicht explizit vorgesehen, kann aber durch die Verwendung von entsprechenden Attributen erreicht werden[1]. Die Prädikate, mit denen ein gesuchter Dienst spezifiziert wird, lassen für die Attribute nur einfache Vergleiche zu und erlauben daher keine allgemeinen Gebiets- und Nachbarschaftsanfragen. SLP wurde für den Einsatz in einem größeren Unternehmensnetzwerk konzipiert und setzt eine einheitliche Administration für die Multicast-Infrastruktur und die Gültigkeitsbereiche voraus. Für einen weitverteilten Einsatz ist SLP in der jetzigen Form laut GUTTMAN ET AL. (1999) daher nur bedingt geeignet.

Weitere Ansätze basieren auf einer verteilten Server-Infrastruktur, welche die für das Auffinden von Diensten benötigten Dienstbeschreibungen verwaltet. Oft wird für das Aussenden von

1. Eine implizite Zuordnung ist allerdings durch die Lokalität der Mulitcast-Anfragen und -Ankündigungen gegeben.

Dienstankündigungen und Dienstanfragen ebenfalls ein lokal begrenzter Multicast verwendet. ADIJIE-WINOTO ET AL. (1999) schlagen das *Intentional Naming System* (INS) vor, bei dem die Dienstbeschreibungen gleichzeitig als Namensraum verwendet werden. Ein Name besteht in diesem Fall aus einer hierarchisch geschachtelten Kombination von Attribut-Wert-Paaren, die den entsprechenden Dienst beschreiben. Ein Dienst oder eine Gruppe von Diensten wird dann durch eine geeignete Beschreibung – gegebenfalls mit eingefügten Platzhaltern (engl. *wildcards*) – adressiert. Basierend auf diesen Namen realisiert das System eine Weiterleitungsfunktionalität für Dienstanfragen auf Anwendungsebene. Dabei wird entweder ein beliebiger zu der Anfrage passender Dienst adressiert (*intentional anycast*) oder alle entsprechenden Dienste (*intentional multicast*). Die einzelnen Knoten eines INS tauschen regelmäßig Informationen über die ihnen bekannten Namen mit benachbarten Knoten aus. Da bei dem aktuellen Ansatz jedem Knoten sämtliche Namen bekannt sein müssen, ist er vorerst nur bis zu größeren Firmennetzwerken skalierbar.

Um auch für eine große Anzahl von Diensten skalierbar zu sein, verwendet der in CZERWINSKI ET AL. (1999) beschriebene sichere Service Discovery Service (SDS) eine hierarchische Struktur, in der die Dienstanfragen weitergeleitet werden. Server auf höheren Ebenen speichern dazu eine über eine spezielle Hash-Funktion erstellte Zusammenfassung der Dienstinformationen auf ihnen untergeordneten Servern. Diese ist so gewählt, dass bei einer Weiterleitungsentscheidung zwar falsche Teilpfade verfolgt, aber keine richtigen Pfade ausgelassen werden. Ein wesentliches Ziel bei diesem Ansatz war eine geheime und authentifizierte Übertragung der Informationen, was sich in einem erhöhten Aufwand für die Verarbeitung und Übertragung von Anfragen niederschlägt.

Als Teil des Globe-Projekts wird ein hochskalierbarer, weltweiter Dienst speziell für die Verwaltung des Aufenthaltsorts von mobilen Objekten (im wesentlichen Softwareobjekten) entwickelt (siehe VAN STEEN ET AL. (1998)). Um die für die angestrebte sehr hohe Anzahl von mobilen Objekten benötigte Skalierbarkeit zu erreichen, verwendet der Dienst ebenfalls einen hierarchischen Suchbaum. Die Kontaktadresse für ein Objekt kann dabei, je nach Grad der Mobilität, auf verschiedenen Ebenen gespeichert sein und wird über Suchverweise in den höheren Ebenen gefunden. In BAGGIO, BALLINITIJN & VAN STEEN (2000) wird zur Optimierung der Suche ein speziell dafür ausgelegter Mechanismus zum Zwischenspeichern von Anfrageergebnissen auf höheren Ebenen vorgestellt.

Zusammenfassend unterstützen Verzeichnisdienste zumeist nicht die für den Lokationsdienst benötigten Gebiets- und Nachbarschaftsanfragen, zumindest aber nicht für beliebige Gebiete und Positionen, und sind nicht auf die Verwaltung von Positionsinformationen mit unterschiedlicher Genauigkeit ausgelegt. Im Gegensatz zu dem hier betrachteten Lokationsdienst liefern

Verzeichnisdienste auch nicht die aktuelle Position des gesuchten Objekts, sondern die Kontaktadresse des betreffenden Geräts oder eines zuständigen Rechners zurück. Verzeichnisdienste sind weiterhin für Informationen ausgelegt, die sich weniger oft ändern, als die Positionsinformationen, die der in dieser Arbeit betrachtete Lokationsdienst verwalten soll (siehe z. B. COULOURIS, DOLLIMORE & KINDBERG (2001)). Viele sind für den Einsatz in einem Firmennetzwerk und nicht als weltweiter Dienst konzipiert.

Verzeichnisdienste und der Lokationsdienst verfolgen damit unterschiedliche Ziele und können und sollen sich in einer Anwendungsumgebung gegenseitig ergänzen. So kann der Lokationsdienst verwendet werden, um die genaue Position eines mobilen Objekts zu ermitteln, das über einen Verzeichnisdienst anhand einer allgemeinen Beschreibung bestimmt wurde. Andererseits können über einen Verzeichnisdienst auch nähere Informationen zu einem über den Lokationsdienst bestimmtes Objekt, das vielleicht gerade einen bestimmten Raum betreten hat, abgefragt werden.

2.5 Lokationsverwaltung in Mobilkommunikationssystemen

Systemen zur Lokationsverwaltung (engl. *location management*) in Mobilkommunikationssystemen sind in den letzten Jahren intensiv untersucht worden. Ziel solcher Systeme ist dabei die Entwicklung einer verteilten Komponente, die den aktuellen Aufenthaltsort einer sehr großen Anzahl von Mobiltelefonen auf effiziente und skalierbare Weise speichern und zurückliefern kann. Obwohl sich einige der entstandenen Konzepte auf den Lokationsdienst übertragen lassen, ist die Lokationsverwaltung in Mobilkommunikationssystemen speziell auf das Auffinden eines Mobiltelefons ausgerichtet – Gebiets- oder Nachbarschaftsanfragen werden nicht betrachtet. Darüber hinaus werden die Ortsinformationen nur mit einer der Granularität der Funkzellen des zugrundeliegenden Kommunikationsnetzes (von ca. 500 m bis über 10 km Durchmesser) entsprechenden Genauigkeit erfasst.

In existierenden Systemen für die Mobilkommunikation, wie zum Beispiel dem weit verbreiteten GSM-System (engl. *global system for mobile communications*), wird die Ortsinformation zu einem bestimmten Mobiltelefon einmal in dessen HLR und zum anderen in dem für den aktuellen Aufenthaltsort zuständigen VLR gespeichert (siehe MOULY & PAUTET (1992)). Die Ortsinformation beschreibt dabei einen Aufenthaltsbereich (engl. *location area*), der mehrere Funkzellen umfasst und in GSM eine Ausdehnung von bis zu 35 km haben kann. Bei einem eingehenden Anruf wird der aktuelle Aufenthaltsbereich für das angerufene Mobiltelefon anhand

von dessen Nummer aus dem Heimatregister ermittelt und das Telefon in allen Zellen, die der Aufenthaltsbereich enthält, ausgerufen (engl. *paging*). Die Ortsinformation im Heimatregister wird, wie bereits gesagt, aktualisiert, wenn das Mobiltelefon von einem Aufenthaltsbereich in einen anderen wechselt. Verschiedene Varianten für die dabei verwendeten Aktualisierungsmechanismen wurden in Abschnitt 2.2.2 vorgestellt.

Um für zukünftige Mobilkommunikationssysteme, die einerseits deutlich kleinere Funkzellen und andererseits eine wesentlich höhere Anzahl von Nutzern haben werden, eine effiziente und skalierbare Lokationsverwaltung zu ermöglichen, wurden hierarchische Architekturen untersucht (z. B. in WANG (1993), HO & AKYILDIZ (1997) oder ZIEGERT (2000)). Wie bei dem in dieser Arbeit vorgestellten Lokationsdienst speichern dabei die Blattknoten die Profile der Benutzer in ihrem jeweiligen disjunkten Zuständigkeitsgebiet, während Knoten auf einer höheren Ebene nur einen Verweis auf den zuständigen Knoten der nächst tieferen Ebene halten. In diesem Zusammenhang werden zu Optimierungszwecken oft auch Mechanismen für das Zwischenspeichern oder Replizieren von Benutzerprofilen diskutiert (siehe z. B. LAM ET AL. (1998)).

Für Anwendungen mit Ortsbezug (LBS), die eine über die Aufenthaltsbereiche hinausgehende Genauigkeit der Positionsinformationen benötigen, wird in aktuellen Mobilkommunikationssystemen oft eine zusätzliche Komponente bereitgestellt, die für die Bestimmung und Verwaltung von genaueren Positionsinformationen zuständig ist. Ein sogenanntes *Gateway Mobile Location Center* (GMLC, z. B. NOKIA (2001)), eine Zusatzkomponente zur GSM-Infrastruktur, bestimmt den aktuellen Aufenthaltsort eines Mobiltelefons entweder, in einer einfacheren Form, über dessen aktuelle Funkzelle oder durch Berücksichtigung mehrerer Funkzellen mittels eines genaueren Verfahrens wie E-OTD (siehe Abschnitt 2.1.1). Bei einem GMLC handelt es sich allerdings um eine zentralisierte Komponente, die im Wesentlichen nur Positionsanfragen unterstützt. Das in LIF (2001) beschriebene für den Zugriff auf ein GMLC konzipierte *Mobile Location Protocol* (kurz MLP) ist dementsprechend auch auf Mobiltelefone zugeschnitten. Es definiert Formate für Anfrage- und Antwortpakete, die erlauben, die Position eines bestimmten mobilen Endgeräts bei einem GMLC abzufragen, wobei auch eine geforderte Genauigkeit angegeben werden kann. Gebiets- und Nachbarschaftsanfragen werden allerdings noch nicht unterstützt.

2.6 Lokationsdienste für Anwendungen mit Ortsbezug

Informationen über die genaue Position von mobilen Objekten werden vor allem in sogenannten Anwendungen mit Ortsbezug (engl. *location-aware applications*) benötigt, die Informationen

und Dienste abhängig von der aktuellen Position ihrer mobilen Benutzer anbieten. Sie enthalten daher auch meist direkt eine Komponente, die für die Verwaltung der entsprechenden Positionsinformationen zuständig ist. Allerdings sind diese zumeist auf die Bedürfnisse der jeweiligen Anwendung und oft auch auf bestimmte Positionsierungssensoren zugeschnitten. Der in dieser Arbeit behandelte Lokationsdienst soll diese speziellen Lösungen ersetzen und von beliebigen Anwendungen genutzt werden können.

Die ersten Anwendungen mit Ortsbezug waren eng mit den entsprechenden Sensorsystemen wie Active-Badges oder ParcTabs verbunden (siehe Abschnitt 2.1.2). Für den Einsatz der Active-Badge-Technologie in Büroanwendungen ist so ein einfacher Lokationsdienst entstanden (siehe HARTER & HOPPER (1994)). In RIZZO, LININGTON & UTTING (1994) wird darauf aufbauend ein Lokationsdienst vorgestellt, der nicht auf die Active-Badge-Technologie beschränkt ist, sondern unterschiedliche Sensorsysteme integrieren kann. Allerdings werden die Positionsinformationen für jede Installation des Systems zentral verwaltet; es existieren nur rudimentäre Mechanismen für den Austausch von Positionsinformationen zwischen einzelnen Installationen. In dem in SPREITZER & THEIMER (1993) und SCHILIT & THEIMER (1994) beschriebenen Lokationsdienst für das ParcTab-System wird vorgeschlagen, die Position eines Benutzers aus Datenschutzgründen jeweils durch einen speziellen Benutzeragenten zu kapseln, der Zugriffe auf diese überwacht. Dies erschwert allerdings deutlich eine Durchführung von Gebiets- und Nachbarschaftsanfragen.

Im Rahmen des Sentient Computing Projekts an den AT&T Laboratories in Cambridge (siehe ADDLESEE ET AL. (2001)), wurde ein effizienter Lokationsdienst für eine lokale Installation eines Positionierungssystems[2] entwickelt, der einen Ereignismechanismus basierend auf dem Konzept von Enthaltenseinsbeziehungen von Aufenthaltsbereichen bietet (siehe z. B. HARTER ET AL. (1999) oder NAGUIB & COULOURIS (2001)). Dieser Ansatz für einen Lokationsdienst, der intern auf einem Quadtree basiert (vgl. Abschnitt 5.3), erreicht laut HARTER ET AL. (1999) auf einem 7-Prozessor SUN-Server lokal einen Durchsatz von 1700 Aktualisierungen pro Sekunde. Verteilungsaspekte wurden allerdings nicht betrachtet.

Ulf Leonhardt behandelt in seiner Dissertation (LEONHARDT (1998)) die Thematik eines universellen globalen Lokationsdienstes, der unabhängig von einer bestimmten Anwendung oder einem bestimmten Sensorsystem ist. Er untersucht dazu die fundamentalen Eigenschaften eines solchen Dienstes und klassifiziert diese in einem abstrakten Dienstmodell. Zusätzlich schlägt er ein Lokationsmodell vor, das verschiedene Typen von Positionsinformationen integriert, und beschreibt die Anforderungen sowie Richtlinien für die Zugriffskontrolle auf Positionsinforma-

2. In diesem Fall ein Active-Bat-System (siehe Abschnitt 2.1).

tionen. Die Arbeit betrachtet zwar Architekturaspekte eines solchen Lokationsdienstes, enthält aber keinen Vorschlag oder Bewertungen einer bestimmten Architektur.

MAASS (1997) beschreibt ein Datenmodell und die Zugriffsprotokolle, die notwendig sind, um einen universellen Lokationsdienst auf Basis des X.500 Verzeichnisdienstes (ITU/ISO (1997)) zu betreiben. Der erweiterte Dienst unterstützt Positions-, Gebiets- und Nachbarschaftsanfragen, sowie Benachrichtigungen, allerdings nur für vordefinierte Gebiete. Verzeichnisdienste sind, wie weiter unten beschrieben, nicht für Informationen ausgelegt, die sich häufig ändern; vor allem die Mechanismen zum Zwischenspeichern und zur Replikation von Informationen, durch die deren Skalierbarkeit erst erreicht werden kann, gehen von mäßig häufigen Änderungen aus.

2.7 Lokationsdienste für Spezialanwendungen

Für die Verwaltung von Positionsinformationen in speziellen Anwendungsbereichen sind verschiedene Speziallösungen entstanden. Zumeist gibt es in diesem Fall besondere Anforderungen, die dazu führen, dass diese Lösungen nicht für einen generellen Einsatz geeignet sind. Meist ist aus demselben Grund allerdings auch der hier beschriebene Lokationsdienst nicht für diese Spezialanwendungen geeignet.

Ein solch spezieller Lokationsdienst ist der in LI ET AL. (2000) beschriebene *Grid's location service* (GLS) zur Unterstützung der Nachrichtenvermittlung (engl. *routing*) in Ad-Hoc-Netzwerken. Für die Nachrichtenvermittlung in solchen Netzen wurden Protokolle vorgeschlagen, welche die geographischen Positionen der mobilen Knoten verwenden, um z. B. bei der Wegefindung (engl. *route discovery*) das sonst übliche Fluten des Netzwerks (siehe u. a. PERKINS, ROYER & DAS (1999)) auf den zuletzt bekannten Aufenthaltsort des Empfängers zu beschränken. Laut KO & VAIDYA (1998) ergibt sich dadurch im Vergleich zu herkömmlichen Ansätzen ein deutlich niedrigerer Aufwand für die Wegefindung.

GLS wurde entwickelt, um die für diese Protokolle benötigten Positionsinformationen für die mobilen Knoten eines Ad-Hoc-Netzes zu speichern und unterstützt nur einfache Positionsanfragen. Die Positionsinformationen für ein bestimmten mobilen Knoten können dabei prinzipiell auf einem beliebigen anderen Knoten des Netzes gespeichert werden, wobei ein spezieller Hash-Algorithmus dazu verwendet wird, um die jeweils zuständigen Knoten zu bestimmen. Eine Positionsanfrage wird durchgeführt, indem diese an bestimmte Knoten in der Umgebung des Senders geschickt wird, die ebenfalls durch diese Hash-Funktion bestimmt werden. Wie bei Ad-Hoc-Netzen üblich, ist das Ziel keine garantierte oder besonders effiziente Ausführung der

Anfragen, sondern eine möglichst hohe Erfolgsrate beim Auffinden der Positionsinformationen.

2.8 Zusammenfassung

Der vorangehende Überblick über verwandte Arbeiten lässt sich folgendermaßen zusammenfassen: In den letzten Jahren hat es große Fortschritte auf dem Gebiet der Positionierungssensoren gegeben und es sind verschiedene Systemen entstanden, die für unterschiedlichste Einsatzbereiche eine genaue und zuverlässige Bestimmung der Positionen mobiler Objekte ermöglichen. Diese lösen allerdings nur das Problem der lokalen Erfassung der Positionsinformationen und erlauben nicht einen globalen, einheitlichen Zugriff darauf. Ein solcher Zugriff auf die Positionsinformationen wird durch einen Lokationsdienst erbracht, der auf diese Positionierungssensoren aufsetzt und der in dieser Arbeit betrachtet werden soll.

Zu diesem Zweck müssen die Positionsinformationen zuerst effizient über eine meist drahtlose Verbindung zum Lokationsdienst übertragen werden. Zwar wurden Protokolle für die Übertragung von Positionsinformationen unter den speziellen Randbedingungen von Mobilkommunikationssystemen und für Spezialanwendungen untersucht, eine allgemeine Betrachtung von Protokollen, die für die Übertragung von Positionsinformationen mit einer hohen und/oder flexiblen Genauigkeit geeignet sind, existiert allerdings noch nicht.

Der Lokationsdienst selbst muss effizient und skalierbar sein, um auch mit einer hohen Zahl mobiler Objekte zurechtzukommen und die von modernen Positionierungssensoren gelieferte hohe Genauigkeit der Positionsinformationen zu unterstützen. Die Betrachtung verwandter Arbeiten hat gezeigt, dass zwar Systeme wie räumliche Datenbanken und erste Ansätze für Lokationsdienste in ortsbezogenen Anwendungen existieren, die z. T. die benötigte Funktionalität erbringen, aber nicht die gewünschte hohe Effizienz und Skalierbarkeit aufweisen können. Ansätze aus dem Bereich der Lokationsverwaltung für Mobilkommunikationsnetze und Verzeichnisdienste haben dagegen die geforderte Skalierbarkeit, unterstützen jedoch keine allgemeinen Operationen auf räumlichen Daten, wie Gebiets- und Nachbarschaftsanfragen, und kommen zumeist nicht mit Positionsinformationen unterschiedlicher Genauigkeit zurecht.

Ziel dieser Arbeit ist es daher, unter Berücksichtigung der Ergebnisse der im Vorhergehenden beschriebenen verwandten Arbeiten, die für einen universellen skalierbaren Lokationsdienst benötigten Mechanismen und Konzepte zu erarbeiten. Zu diesem Zweck betrachten wir im nächsten Kapitel zuerst das Dienstmodell für einen solchen Lokationsdienst, das die von diesem zu fordernde Funktionalität festlegt.

Kapitel 3

Hintergrund der Arbeit: Das Nexus-Projekt

Die hier beschriebenen Arbeiten sind im Rahmen der von der DFG geförderten Forschergruppe Nexus an der Universität Stuttgart (NEXUS (2002)) entstanden, die zum Ziel hat, eine Plattform für Anwendungen mit Ortsbezug zu schaffen. Der Lokationsdienst wird innerhalb dieser Plattform für die Verwaltung der Positionsinformationen mobiler Objekte eingesetzt. Obwohl die Konzepte des Lokationsdienstes unabhängig von dessen Verwendung innerhalb der Nexus-Plattform sind, sollen die Zusammenhänge für ein besseres Verständnis des Hintergrunds der Arbeit hier kurz aufgezeigt werden.

Die Nexus-Plattform soll verschiedenartigen Anwendungen mit Ortsbezug ein detailliertes, um digitale Informationen erweitertes Umgebungsmodell (engl. *Augmented World Model*) zur Verfügung stellen, über das sie Informationen zum aktuellen Zustand der Umgebung abfragen können, in der sich ihre Benutzer bewegen (siehe z. B. HOHL ET AL. (1999) oder COSCHURBA, KUBACH & LEONHARDI (2000)). Das Umgebungsmodell umfasst stationäre (z. B. Gebäude oder Bäume) ebenso wie mobile Objekte (z. B. Personen oder Fahrzeuge) und neben realen Objekten auch virtuelle Objekte, wie z. B. virtuelle Litfaßsäulen oder virtuelle *Post-Its*, die geographische Orte oder reale Objekte mit Informationen oder Informationsdiensten verbinden (siehe LEONHARDI ET AL. (1999) oder BAUMANN ET AL. (2001)). Durch die Einbeziehung von Sensorinformationen soll das von der Nexus-Plattform verwaltete Umgebungsmodell konsistent mit dem von ihm beschriebenen Teil der realen Welt gehalten werden. Ein wichtiger Aspekt ist dabei, dass mobile Objekte mit ihrer aktuellen Position im Umgebungsmodell beschrieben sind.

Innerhalb der Nexus-Plattform wird das Umgebungsmodell durch eine verteilte Server-Infrastruktur verwaltet. Anwendungen mit Ortsbezug können die dort gespeicherten Informationen über eine speziell für diesen Zweck entwickelte einheitliche Anfragesprache (engl. *Augmented World Querying Language*, kurz AWQL) abrufen. Für die Ergebnisse der Anfragen und für den Austausch von Daten steht entsprechend eine einheitliche Modellbeschreibungssprache (engl. *Augmented World Modelling Language*, kurz AWML) zur Verfügung. Zukünftig soll es Anwendungen mit Ortsbezug auch möglich sein, sich über einen *Publish&Subscribe*-Mechanismus für bestimmte Ereignisse zu registrieren, worauf sie von der Plattform bei dessen Eintreten benachrichtigt werden. Z. B. könnte es für einen Benutzer von Interesse sein, eine Benachrichtigung zu erhalten sobald er in die Nähe eines bestimmten Geschäfts kommt.

Da sich die Handhabung der dynamischen und kompakten Positionsinformationen wesentlich von der für die statischen und umfangreichen räumlichen Daten stationärer Objekte unterscheidet, wurde beschlossen, diese in einem ersten Schritt von zwei getrennten Komponenten verwalten zu lassen. Während die statischen Daten auf sogenannten Umgebungsmodell-Daten-Servern (engl. *Spatial Model Server*, kurz SpaSe) gehalten werden, ist für die Verwaltung der dynamischen Positionsinformationen als zentrale Stelle der in dieser Ausarbeitung betrachtete Lokationsdienst zuständig. In darauf folgenden Projektphasen ist jedoch eine Zusammenführung dieser Komponenten geplant (siehe auch Abschnitt 9.3).

Um die für einen angestrebten großflächigen oder gar globalen Einsatz der Nexus-Plattform notwendige Skalierbarkeit und Handhabbarkeit zu erreichen, muss es sowohl beim Lokationsdienst als auch bei den Umgebungsmodell-Daten-Servern möglich sein, die anfallenden Last, die durch die zu bearbeitenden Anfragen aber auch den Administrationsaufwand entsteht, auf mehrere Rechner zu verteilen. Bei den räumlichen Modellen ist ein Server jeweils für ein bestimmtes Dienstgebiet zuständig (ähnlich wie ein Blatt-Server des Lokationsdienstes) und speichert Informationen über bestimmte Klassen[1] der sich darin befindenden Objekte mit einer gegebenen Granularität. Die Nexus-Plattform ermöglicht somit zum einen eine „vertikale" Aufteilung, bei der jeweils ein Rechner, ein sogenannter Nexus-Knoten, für ein bestimmtes nicht überlappendes geographisches Gebiet zuständig ist. Zum anderen verwalten bei einer „horizontalen" Aufteilung bestimmte Nexus-Knoten das Umgebungsmodell mit einer groben Auflösung (z. B. auf der Granularität eines Stadtplans), während andere Rechner Ausschnitte davon mit einer höheren Auflösung speichern (beispielsweise ein detailliertes Modell der Innenräume eines

1. Beispiele für solche Klassen sind Informationen über Gasthäuser oder Touristenattraktionen, für die es günstig sein kann, sie auf verschiedenen Nexus-Knoten zu halten.

Gebäudes). Die Zuordnung zwischen Dienstgebieten und Nexus-Knoten wird in einem speziellen Verzeichnis (engl. *Area Service Register*, kurz ASR) verwaltet.

Wegen der hohen Anforderungen an die effiziente Verwaltung von Positionsinformationen sind für den Lokationsdienst allerdings spezielle Verteilungs- und Verwaltungsmechanismen notwendig (siehe auch unsere Betrachtung verwandter Arbeiten in Kapitel 2). Für die Realisierung eines skalierbaren Lokationsdienstes ist so eine hierarchische Architektur erforderlich, die in der Lage ist, für die hochdynamischen Positionsinformationen Aktualisierungen und Anfragen effizient zu bearbeiten. Diese Thematik wird in Kapitel 5 ausführlich behandelt.

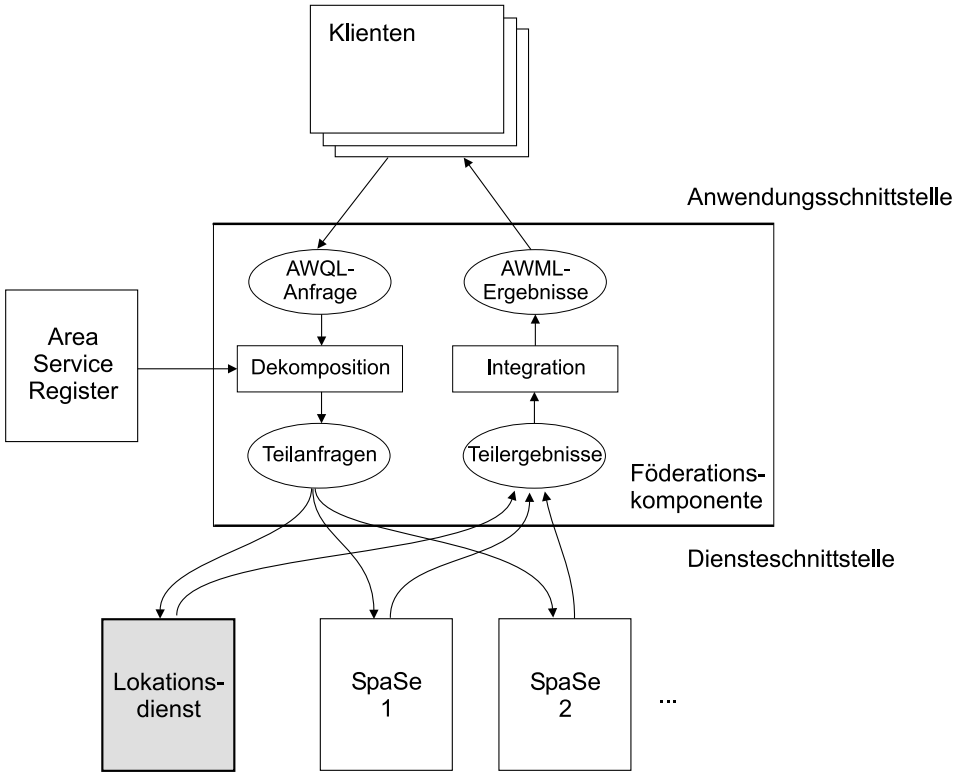


Abbildung 3-1. Die Föderationskomponente der Nexus-Plattform.

Für die verteilte Bearbeitung einer AWQL-Anfrage an die Nexus-Plattform ist die Föderationskomponente zuständig, die auf jedem Nexus-Knoten existiert. Ihre Funktionsweise ist in Abbildung 3-1 gezeigt. Die Föderationskomponente bestimmt unter Zuhilfenahme des ASR und anhand des von der Anfrage betroffenen Gebiets, der geforderten Granularität der Informationen sowie der gesuchten Klassen von Objekten, welche Umgebungsmodell-Daten-Server von einer Anfrage betroffen sind und leitet entsprechende Teilanfragen an diese weiter (siehe NICKLAS ET AL. (2001) und NICKLAS & MITSCHANG (2001)). Anschließend sammelt sie die

Teilergebnisse ein und integriert sie zu einem Gesamtergebnis. Zusätzlich bestimmt die Föderationskomponente aus den Teilelementen der AWQL-Anfrage, ob die Positionsinformationen mobiler Objekte von ihr betroffen sein könnten. Ist dies der Fall, z. B. weil alle Objekte in einem bestimmten Raum gesucht werden, wird zusätzlich eine entsprechende Teilanfrage an den Lokationsdienst weitergegeben (in unserem Beispiel eine geeignete Gebietsanfrage). Die vom Lokationsdienst zurückgelieferten Ergebnisse werden ebenfalls in das Gesamtergebnis der Anfrage integriert (siehe auch GROSSMANN ET AL. (2001)). Die Bearbeitung der Teilanfragen innerhalb des Lokationsdienstes ist ein zentrales Thema dieser Arbeit und wird in Kapitel 5 ausführlich beschrieben.

Kapitel 4

Dienstmodell des Lokationsdienstes

In diesem Kapitel soll aus den Anforderungen von ortsbezogenen Anwendungen ein Dienstmodell für den Lokationsdienst erarbeitet werden, das die von diesem anzubietenden Operationen und deren Semantik beschreibt.

In unserem Dienstmodell unterscheiden wir zwischen dem *Lokationsdienst* (engl. *location service*, kurz LS), der typischerweise verteilt durch eine große Zahl von Lokations-Servern realisiert ist, den *mobilen Objekten*, deren Positionsinformationen vom LS verwaltet werden (engl. *tracked objects*), und den *Klienten*, die über den LS auf Informationen über die Positionen der mobilen Objekte zugreifen (siehe Abbildung 4-1). Der LS ist dafür zuständig, die Positionsinformationen für die bei ihm registrierten mobilen Objekte innerhalb eines bestimmten geographischen Gebiets, seines *Dienstgebiets* (engl. *service area*), zu verwalten. Die Positionsinformationen der mobilen Objekte werden dabei durch lokale oder externe Positionierungssensoren (siehe Abschnitt 2.1) bestimmt und ständig in der Datenbank des LS aktualisiert.

Klienten des LS benutzen diesen, um Anfragen bezüglich der Positionsinformationen der mobilen Objekte zu stellen, z. B. nach der aktuellen geographischen Position eines bestimmten Objekts oder nach allen Objekten innerhalb eines gewissen geographischen Gebiets. Das mobile Endgerät eines Benutzers (beispielsweise ein PDA oder ein Mobiltelefon) wird dabei oft beide Rollen einnehmen, die eines mobilen Objekts, dessen Position der LS verwaltet, und die eines Klienten des LS. Ein mobiles Gerät kommuniziert mit dem LS über ein drahtloses Kommunikationsmedium, wie GSM (siehe MOULY & PAUTET (1992)) bzw. GPRS oder ein Funk-LAN (definiert durch den IEEE Standard 802.11, siehe IEEE COMPUTER SOCIETY (1997)).

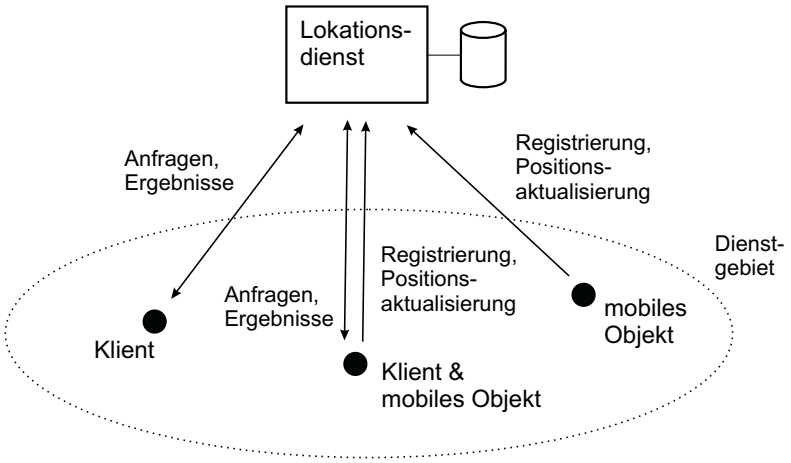


Abbildung 4-1. Grundlegende Komponenten und Interaktionen des Lokationsdienstes.

4.1 Positionsdaten

Aus Sicht eines Klienten speichert der LS in seiner Datenbank für jedes von ihm verwaltete mobile Objekt *o* einen sogenannten *Lokationsbezeichner.* Dieser setzt sich zusammen aus der geographischen Position von *o* sowie einer Angabe der Genauigkeit dieser Information. Die Positionsinformation ist dabei als Koordinate eines globalen geographischen Koordinatensystem angegeben. Intern verwendet der LS ein eindeutiges Koordinatensystem (in unserem Fall das bei GPS eingesetzte, weit verbreitete WGS84 Koordinatensystem (NIMA (1997)), um die für einen verteilten Dienst notwendige Einheitlichkeit zu gewährleisten. Allerdings kann er an seinen Schnittstellen eine Konvertierung in andere Koordinatensysteme, wie das in Deutschland häufig für Kartenmaterial verwendete Gauss-Krüger-System oder ein lokales Koordinatensystem, das z. B. auf einen Bezugspunkt in einem Gebäude ausgerichtet ist, anbieten.

Die im LS gespeicherten Koordinaten sind dreidimensional, d. h. sie beinhalten eine Höheninformation. Die Semantik der Anfragen und die Mechanismen des LS sind allerdings im Folgenden für den zweidimensionalen Fall beschrieben, wobei die Höheninformation der Koordinaten ignoriert wird. Dies dient einerseits einer einfacheren Darstellung und ist andererseits für die meisten Fälle bereits ausreichend, da bei den bisherigen Einsatzszenarien für ortsbezogene Anwendungen, wie einem Stadtführer, selten übereinander liegende mobile Objekte berücksichtigt werden müssen. Eine Erweiterung auf den dreidimensionalen Fall ist ohne große Änderungen der im Weiteren beschriebenen Mechanismen möglich.

Aufgrund der technisch bedingten Ungenauigkeit der Sensorsysteme und der Verzögerung bei der Übertragung der Positionsinformationen, kann der LS diese Informationen nur bis zu einer bestimmten Genauigkeit speichern. Die Genauigkeit hängt dabei von verschiedenen Faktoren ab, hauptsächlich von dem zur Aktualisierung der Informationen eingesetzten Protokoll und der Aktualisierungshäufigkeit sowie der Genauigkeit des Sensorsystems, das die Informationen liefert (Protokolle zur Aktualisierung von Positionsinformationen werden ausführlich in Kapitel 6 besprochen).

Ein Lokationsbezeichner (engl. *location descriptor*) für ein vom LS verwaltetes mobiles Objekt *o* ($o \in O$, der Menge der vom LS verwalteten mobilen Objekte) wird mit *ld(o)* bezeichnet und besteht aus zwei Elementen:

ld(o).pos: Die für das Objekt *o* gespeicherte Position, beschrieben durch eine geographische Koordinate. Die Menge aller möglichen Koordinaten wird mit *Pos* bezeichnet.

ld(o).acc: Die Genauigkeit der Positionsinformation, definiert als maximale Entfernung (in Metern) zwischen *ld(o).pos* und der tatsächlichen Position von *o*.

Wenn *rp(o)* die tatsächliche aktuelle Position von Objekt *o* bezeichnet, gilt daher die folgende Beziehung:

$$DISTANCE(ld(o).pos - rp(o)) \leq ld(o).acc \qquad (4\text{-}1)$$

Wie in Abbildung 4-2 gezeigt, wird damit garantiert, dass sich *o* innerhalb des kreisförmigen Gebiets aufhält, das sich mit Mittelpunkt *ld(o).pos* und Radius *ld(o).acc* aus *ld(o)* ergibt. Dieses Gebiet bezeichnen wir als *Aufenthaltsbereich* von *o* (engl. *location area*). Es ist wichtig zu beachten, dass damit ein kleinerer Wert für *ld(o).acc* eine größere Genauigkeit der Positionsinformationen bedeutet. Die Genauigkeit, mit welcher der LS die Positionsinformation für *o* verwaltet, kann bei der Registrierung ausgehandelt werden (siehe Abschnitt 4.2).

Wenn ein mobiles Objekt beim LS registriert wird oder es seine Position aktualisiert, wird ein sogenannter *Positionierungsdatensatz* (engl. *sighting record*) an den LS gesendet. Ein Positionierungssdatensatz *s* hat die folgenden Eigenschaften:

s.oID: Der Bezeichner für das mobile Objekt, der innerhalb des entsprechenden Namensraumes des LS, *OId*, eindeutig ist.

s.t: Ein Zeitstempel, der den Zeitpunkt der Positionsbestimmung angibt.[1]

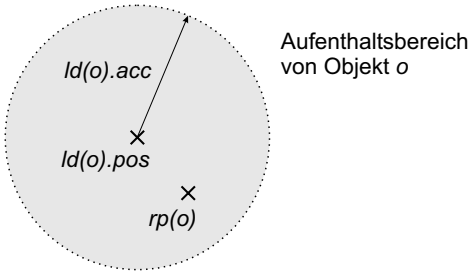


Abbildung 4-2. Der durch den Lokationsbezeichner ld(o) *beschriebene Aufenthaltsbereich.*

s.pos: Die Position des mobilen Objekts zum Zeitpunkt *s.t* beschrieben als geographische Koordinate.

$s.acc_{sens}$: Die Genauigkeit des Positionierungssensors, mit dem die Positionsbestimmung durchgeführt wurde, wiederum als die maximale Entfernung zwischen der angegebenen Position *s.pos* und der wirklichen Position des Objekts zum Zeitpunkt *s.t*.

Der LS kann *s.t* und $s.acc_{sens}$ auch verwenden, um zusammen mit der maximalen Geschwindigkeit des mobilen Objekts $\hat{v}$ die Genauigkeit der Positionsinformation zu einem späteren Zeitpunkt $t > s.t$ zu berechnen, d. h. die maximale Abweichung *maxdev()* zu bestimmen. Dies wird z. B. bei den in Kapitel 6 beschriebenen Protokollen zur Übertragung von Positionsinformationen verwendet. Die maximale Abweichung ergibt sich folgendermaßen aus der maximalen Entfernung, um die sich das Objekt in dem gegebenen Zeitraum bewegt haben kann:

$$maxdev(s, t) = s.acc_{sens} + \hat{v}\,(t - s.t) \qquad (4\text{-}2)$$

Beim Entwurf des Positionierungsdatensatzes haben wir ein geometrisches Modell für die Beschreibung der Positionsinformationen gewählt, bei dem die Position durch eine Koordinate in einem geographischen Koordinatensystem dargestellt wird. Die Alternative dazu wäre ein symbolisches Modell gewesen, bei dem die Positionsinformationen durch einen diskreten Bezeichner, z. B. eine Raumnummer oder die Zelle eines Mobilfunksystems, gegeben ist. Solche Bezeichner stehen jeweils für ein bestimmtes geographisches Gebiet und können hierarchisch or-

1. Für die Erzeugung dieses Zeitstempels setzen wir voraus, dass auf den mobilen Endgeräten und Lokations-Servern synchronisierte Uhren verfügbar sind. Zur Synchronisierung der Uhren kann z. B. die hochgenaue Zeitangabe verwendet werden, die ein GPS-Empfänger zurückliefert.

ganisiert sein. Der Grund für die Auswahl des geometrischen Modells war dessen bessere Skalierbarkeit, die auch die Darstellung von sehr genauen Positionsinformationen erlaubt, die vereinfachte Zusammenarbeit mit GIS-Systemen bzw. den Umgebungsmodell-Daten-Servern der Nexus-Plattform, die ebenfalls auf geographischen Koordinaten basieren. Ein weiterer Grund war die Möglichkeit verschiedene Berechnungen, wie die Ermittlung der Entfernung zwischen zwei Koordinaten oder der Bewegungsrichtung, auf einfache Art und Weise durchführen zu können, was z. B. den Umgang mit Positionsinformationen unterschiedlicher Genauigkeit erleichtert. Für die Umrechnung von symbolischen auf geometrische Koordinaten und umgekehrt kann auf einen externen Dienst, z. B. ein GIS-System oder einen Verzeichnisdienst, zurückgegriffen werden. Im Rahmen der Nexus-Plattform wird diese Funktionalität durch die Umgebungsmodell-Daten-Server, auf denen die räumlichen Modelle gespeichert sind, erbracht. Für eine ausführlichere Betrachtung der Vor- und Nachteile von geometrischen und symbolischen Modelle siehe LEONHARDT (1998).

Der hier verwendete Aufbau des Positionierungsdatensatzes ist wegen der Wahl des geometrischen Modells naturgemäß besser geeignet für den Umgang mit Positionierungssystemen, die als Ergebnis eine geographische Koordinate liefern, wie z. B. GPS oder das Active-Bat-System (WARD, JONES & HOPPER (1997)). Bei Positionierungssystemen, wie dem Active-Badge-System (WANT ET AL. (1992)), die stattdessen einen symbolischen Bezeichner für den Aufenthaltsbereich liefern (in diesem Fall die Raumnummer), wird als Position $s.pos$ das Zentrum (d. h. der Schwerpunkt) des entsprechenden Gebiets verwendet. Die maximale Ungenauigkeit $s.acc_{sens}$ ist dann die größte Entfernung eines Punktes aus diesem Gebiet von dessen Zentrum. Weil dabei allerdings ein Teil der Semantik der Positionsinformationen verloren geht, wäre eine alternative, besser geeignete Behandlung von symbolischen Koordinaten wünschenswert, die allerdings über den Rahmen dieser Arbeit hinausgehen würde (für einen Ausblick siehe Abschnitt 9.3).

4.2 Registrierung und Positionsaktualisierung

Mobile Objekte werden von einer registrierenden Instanz (engl. *registering instance*) beim LS angemeldet, z. B. wenn der Benutzer eine ortsbezogene Anwendung auf seinem mobilen Endgerät startet und die Anmeldung beim LS aktiviert. Bei der registrierenden Instanz kann es sich wie im obigen Beispiel um das mobile Endgerät eines Benutzers handeln, bei anderen Objekten, deren Position z. B. von einem externen Sensorsystem erfasst wird, aber auch um einen für sie zuständigen Zentralrechner. Im Folgenden soll aus Gründen einer einfacheren Darstellung angenommen werden, dass es sich bei der registrierenden Instanz um das mobile Endgerät selbst handelt.

Aus Sicherheitsgründen dürfen Anmelden und Abmelden sowie das Einstellen der Genauigkeit mit welcher der LS die Positionsinformationen verwaltet nur von der registrierenden Instanz durchgeführt werden, die sich unter der Kontrolle des entsprechenden Benutzers befindet. Weitere Mechanismen, die Datenschutz und -sicherheit gewährleisten, werden in Kapitel 8 besprochen.

Registrierung (engl. *register*): Mit Aufrufen der Funktion *register* meldet sich ein mobiles Objekt beim LS zur Verwaltung seiner Positionsinformation an. In der grundlegenden Variante hat die Registrierungsfunktion die folgende Signatur:

register(s, oInfo, acl, desAcc, timeout) $\rightarrow$ *offeredAcc*

Im Parameter *s* übergibt das mobile Objekt dazu seine aktuelle Position, der Parameter *desAcc* legt die gewünschte Genauigkeit der Positionsinformation fest. Diese ist auch gleichzeitig die maximale Genauigkeit mit welcher der Benutzer dem LS mit der Verwaltung der Positionsinformationen vertraut. Der Parameter *oInfo* enthält weitere Informationen über das registrierte Objekt, darunter eine Liste mit den verfügbaren Sensorsystemen und deren Eigenschaften. Mit dem Parameter *acl* wird eine Zugriffskontrollliste übergeben, die angibt, welche Personen(gruppen) auf diese Information zugreifen dürfen (siehe Abschnitt 8.3). Der Parameter *timeout* gibt einen Zeitschwellwert an, nach dem der LS das mobile Objekt automatisch deregistriert, wenn er innerhalb dieses Zeitraums keine Positionsaktualisierung empfangen hat (siehe Abschnitt 5.3.3). Der Rückgabewert *offeredAcc* gibt schließlich die auf dem LS verfügbare Genauigkeit (engl. *offered accuracy*) der Positionsinformation an und entspricht *desAcc*, wenn die Anmeldung erfolgreich war und eines der aktuell verfügbaren Sensorsysteme diese Genauigkeit unterstützt. Andernfalls wird eine entsprechende Fehlermeldung zurückgegeben.

Bislang existiert allerdings kein Positionierungssensor, der global verfügbar ist. Verschiedene (auch sich überschneidende) Bereiche werden zumeist, wie in Abbildung 4-3 gezeigt, von unterschiedlichen Sensorsystemen abgedeckt. So kann es passieren, dass sich die verfügbare Genauigkeit *offeredAcc* verschlechtert, wenn das mobile Objekt ein Gebiet betritt, in dem ein ungenaueres Sensorsystem verwendet werden muss oder überhaupt keines verfügbar ist.

Nach der Registrierung kann der LS auch jederzeit von der registrierenden Instanz angewiesen werden, die aktuelle Genauigkeit der Positionsinformationen zu ändern:

changeAcc(o, desAcc) $\rightarrow$ *offeredAcc*

Registrierung mit Garantie einer minimalen Genauigkeit: Es gibt eine Reihe von Anwendungen, die eine bestimmte minimale Genauigkeit der Positionsinformationen für die Realisie-

rung ihrer Funktionalität benötigen und für welche die einfache Registrierung nicht ausreichend ist. Wenn z. B. ein Verkehrsleitsystem den Abstand zwischen einzelnen Verkehrsteilnehmern kontrollieren soll, wird es während seiner gesamten Laufzeit eine Positionierungsgenauigkeit für alle Verkehrsteilnehmer benötigen, die deutlich unter dem einzuhaltenden Abstand liegt. Wenn die registrierende Instanz solche Anwendungen unterstützen will, muss sie mit dem LS eine minimale verfügbare Genauigkeit aushandeln.

Bei der zweiten Variante der Registrierungsfunktion wird daher eine minimal geforderte Genauigkeit $minAcc \geq desAcc$ mitgegeben:

register(s, oInfo, acl, desAcc, minAcc, regArea) →
(true, offeredAcc) | *(false, possibleArea)*

Ob *minAcc* garantiert werden kann hängt dabei von den im Dienstgebiet des LS verfügbaren und von dem mobilen Objekt unterstützten Positionierungssensoren ab[2]. Bei einem verteilten LS ist es allerdings nur mit hohem Aufwand möglich zu überprüfen, ob *minAcc* im gesamten Dienstgebiet eingehalten werden kann. Es ist außerdem zu erwarten, dass sich in einem größeren Dienstgebiet Teilgebiete befinden, in denen kein Sensorsystem verfügbar ist. Das mobile Objekt gibt daher bei der Registrierung im Parameter *regArea* das Gebiet an, in dem es sich voraussichtlich bewegen wird und für das *minAcc* garantiert werden soll. Ist dies von den innerhalb von *regArea* verfügbaren Sensorsystemen aus möglich, gibt der LS *true* zurück und in *offeredAcc* die verfügbare Genauigkeit aus dem geforderten Intervall [*desAcc, minAcc*].

Ist die Registrierung nicht erfolgreich, so wird *false* zurückgegeben. Der Grund dafür kann z. B. ein ungünstig gewähltes Registrierungsgebiet sein. Um eine nachfolgende Registrierung zu erleichtern, wird bei einer abgelehnten Registrierung zusätzlich das größte Teilgebiet $possibleArea \subseteq regArea$ zurückgegeben, in dem eine Garantie von *minAcc* möglich ist.

Wie bei der einfacheren Registrierung kann die verfügbare Genauigkeit auch nachträglich durch Aufrufen von *changeAcc* geändert werden, zum Beispiel, wenn das Objekt das zuvor angegebene Aufenthaltsgebiet *regArea* verlassen will:

changeAcc(o, desAcc, minAcc, regArea) →
(true, offeredAcc) | *(false, possibleArea)*

2. Von der verfügbaren Genauigkeit hängt auch in großem Maße ab, mit welcher Häufigkeit die Positionsinformationen aktualisiert werden müssen (siehe Kapitel 6). Ein weiterer, hier nicht betrachteter Grund, dass eine Registrierung scheitert, könnte daher auch in der zu hohen Last liegen, die eine bestimmte Genauigkeit erzeugen würde.

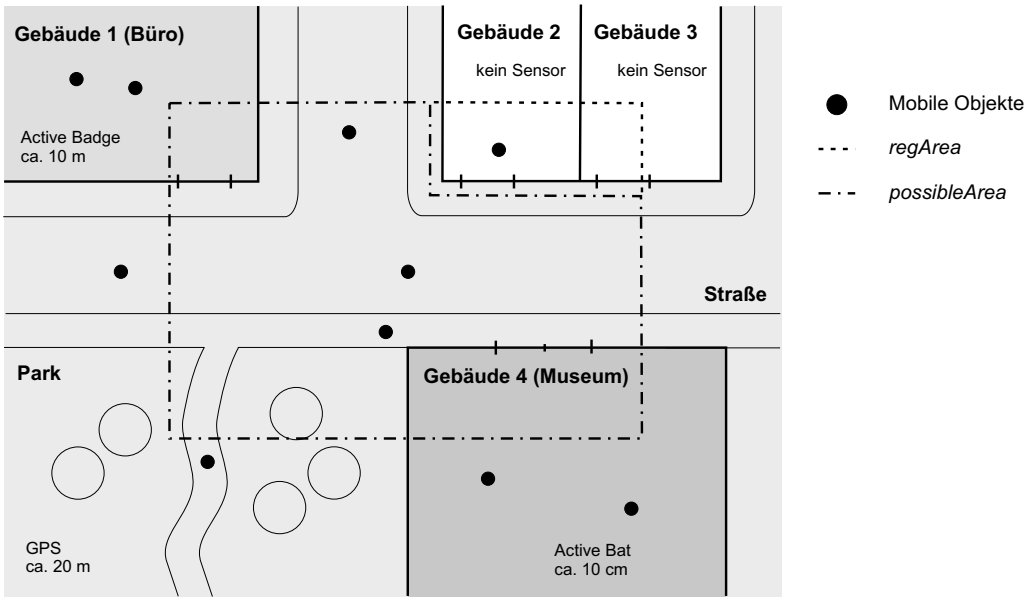


Abbildung 4-3. Beispiel für eine Einsatzumgebung des LS mit verschiedenen Sensorsystemen. Gezeigt ist eine fehlgeschlagene Registrierung mit einer geforderten Genauigkeit von 20 m und dem Registrierungsgebiet regArea *sowie dem zurückgegebenen möglichen Gebiet* possibleArea.

Positionsaktualisierung (engl. *position update*): Um die im LS gespeicherten Positionsinformationen zu aktualisieren, sendet das mobile Objekt (oder ein externes Sensorsystem) regelmäßig eine Positionsaktualisierungsnachricht (*update*) mit dem aktuellen Positionierungsdatensatz *s* an den LS. Für die Steuerung der Übertragung muss dabei ein Protokoll eingesetzt werden, das eine resultierende Genauigkeit der Positionsinformationen auf dem LS von *offeredAcc* garantiert. Verschiedene Varianten solcher Aktualisierungsprotokolle werden in Kapitel 6 ausführlich besprochen.

update(s)

Abmelden (engl. *deregister*): Durch Aufrufen der Funktion *deregister* wird schließlich das mobile Objekt beim LS abgemeldet, der daraufhin alle Kenntnisse über dieses Objekt verliert.

deregister(o)

4.3 Anfragen

Der LS unterstützt drei Typen von Anfragen: Positionsanfragen, Gebietsanfragen und Nachbarschaftsanfragen. Mit einer Positionsanfrage wird die aktuelle Positionsinformation für ein bestimmtes mobiles Objekt abgefragt. Ein Stadtinformationssystem kann dies beispielsweise verwenden, um die Position eines bestimmten Buses zu ermitteln und so dessen eventuelle Verspätung zu errechnen. Eine Gebietsanfrage dagegen liefert alle Objekte zurück, die sich zur Zeit in einem bestimmten Gebiet aufhalten. Das Stadtinformationssystem könnte damit alle Personen abfragen, die an einer bestimmten Bushaltestelle warten und diese über die Verspätung des Buses informieren. Bei einer Nachbarschaftsanfrage schließlich wird das mobile Objekt zurückgegeben, das sich am nächsten zu einer angefragten geographischen Position befindet. Ein Beispiel für eine Nachbarschaftsanfrage ist die Suche nach dem nächsten (freien) Taxi. Andere Anfragen lassen sich zumeist auf diese drei Anfragetypen sowie auf geometrische Operationen zurückführen.

Im Folgenden wird die Semantik der einzelnen Anfragen des LS beschrieben. Die geforderten Bedingungen gelten dabei immer zu dem Zeitpunkt, an dem die Anfrage auf dem LS ausgewertet wird.

Positionsanfrage (engl. *position query*): Eine Anwendung kann die aktuelle Position eines bestimmten Objekts *o* abfragen, indem sie die folgende Anfrage an den LS stellt:

$$posQuery(o) \rightarrow ld$$

Als Antwort wird der betreffende Lokationsbezeichner *ld(o)* zurückgegeben, falls es sich um ein vom Lokationsdienst verwaltetes Objekt handelt.

Gebietsanfrage (engl. *range query*): Eine Gebietsanfrage liefert die Lokationsbezeichner für alle vom LS verwalteten Objekte zurück, die sich innerhalb eines vorgegebenen Gebiets aufhalten. Da es sich bei den vom LS gespeicherten Positionsinformationen um (kreisförmige) Aufenthaltsbereiche und nicht um exakte Punkte handelt, ist in Grenzfällen nicht immer klar, welche Objekte in der Ergebnismenge enthalten sein sollen und welche nicht. In dem in Abbildung 4-4 gezeigten Beispielszenario ist sicher Objekt o_1 in der Ergebnismenge enthalten und o_2 nicht. Bei den Objekten o_3 und o_4, deren Aufenthaltsbereiche sich zu einem unterschiedlichen Grad mit dem angefragten Gebiet überschneiden, ist dies jedoch nicht so deutlich. Die Wahrscheinlichkeit, mit der sich diese Objekte im angefragten Gebiet befinden, hängt vom Grad der Überschneidung zwischen diesem Gebiet und dem Aufenthaltsbereich des Objekts ab[3]. Da Anwendungen diesbezüglich unterschiedliche Anforderungen haben können, erlaubt ihnen der LS den Grad der Überschneidung anzugeben, den ein Objekt haben muss, um zu der Ergebnismenge

der Gebietsanfrage zu gehören. Sei *a* das angefragte Gebiet und *ld(o)* der durch den Lokationsbezeichner gegebene (kreisförmige) Aufenthaltsbereich von o, dann ist der Grad der Überlappung folgendermaßen definiert:

$$Overlap(a, o) = SIZE(a \cap ld(o)) / SIZE(ld(o))$$

Die Funktion liefert damit einen reellen Wert aus dem Intervall [0,1].

Ein weiterer Parameter, der einen großen Einfluss auf das Ergebnis einer Anfrage hat, ist die Genauigkeit der Positionsinformationen. Angenommen die Genauigkeit einer Positionsinformation ist 200 m, wohingegen das angefragte Gebiet die Größe eines Raumes hat. In diesem Fall ist es schwierig, eine Aussage darüber zu treffen, ob sich das mobile Objekt in dem angefragten Gebiet befindet. Aus diesem Grund gibt ein weiterer Parameter, zusätzlich zu dem geforderten Grad an Überschneidung, einen Grenzwert für die minimale Genauigkeit der Positionsinformationen an. Objekte, deren Positionsinformationen nicht die geforderte Genauigkeit haben, werden bei der Berechnung des Anfrageergebnisses nicht berücksichtigt. In dem in Abbildung 4-5 gezeigten Beispiel wird o_5 nicht berücksichtigt, da seine Genauigkeit unterhalb des geforderten Grenzwerts liegt. Wird kein Wert für die geforderte Genauigkeit angegeben, so werden alle vom LS verwalteten Objekte berücksichtigt.

Eine Gebietsanfrage wird durch den Aufruf der folgenden Operation angestoßen:

$$rangeQuery(a, reqAcc, reqOverlap) \rightarrow objSet$$

Der Parameter *a* gibt dabei das angefragte geographische Gebiet an, das durch einen beliebigen Polygonzug beschrieben ist, während *reqAcc* und *reqOverlap* die geforderte Genauigkeit und den geforderten Grad an Überschneidung darstellen. Dabei muss *reqOverlap* aus dem Intervall (0,1] sein. In *objSet* werden Paare aus Objekt- und Lokationsbezeichner *(o, ld(o))* für die Objekte zurückgegeben, die sich in der Ergebnismenge der Anfrage befinden:

$$objSet = \{ (o, ld(o)) \mid o \in O \text{ and } Overlap(a, o) \geq reqOverlap > 0 \text{ and } ld(o).acc \leq reqAcc \}$$

3. Wir gehen dabei von einer gleichverteilten Aufenthaltswahrscheinlichkeit einen Objekts innerhalb seines Aufenthaltsbereichs aus.

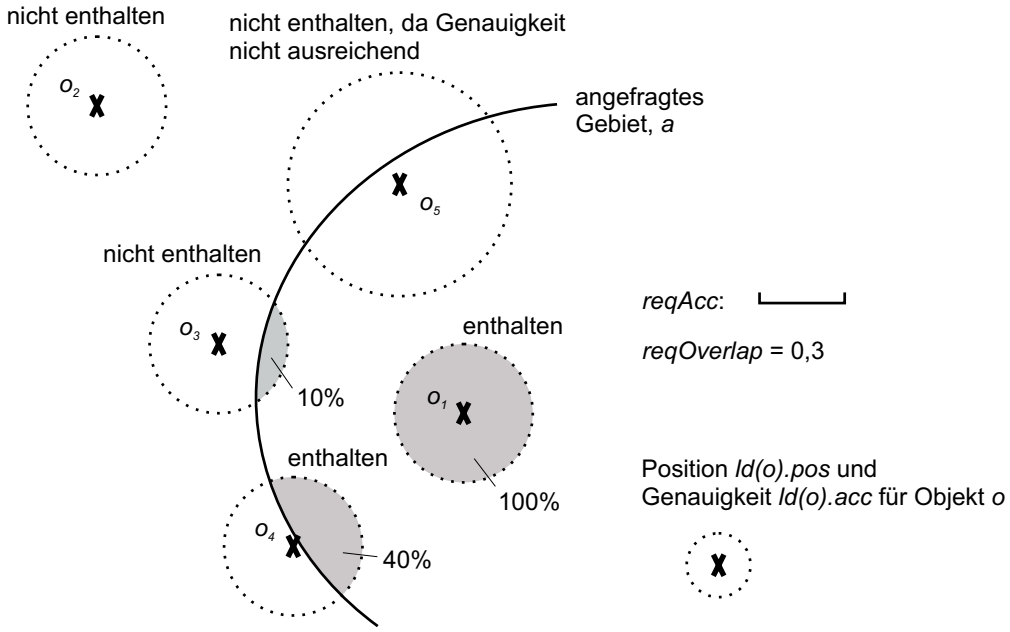


Abbildung 4-4. Beispielszenario für eine Gebietsanfrage.

Nachbarschaftsanfrage (engl. *nearest neighbor query*): Eine Nachbarschaftsanfrage ermittelt aus der Datenbank des LS das mobile Objekt mit der minimalen Entfernung zu einer gegebenen Position. Sie kann folgendermaßen aufgerufen werden:

neighborQuery(p, reqAcc, nearQual, maxDist) → *(nearestObj, nearObjSet)*

Für das entsprechende Objekt *o* ist in *nearestObj* das entsprechende (*o, ld(o)*)-Paar enthalten. Dabei gilt, dass bezüglich der vom LS in *ld(o).pos* gespeicherten Position kein anderes mobiles Objekt näher an der gesuchten Position *p* ist. Wie bei Gebietsanfragen kann eine Anwendung einen Schwellwert für die Genauigkeit *reqAcc* angeben, wobei Objekte mit einer geringeren Genauigkeit (d. h. mit einem Aufenthaltsbereich, dessen Radius größer als *reqAcc* ist) nicht berücksichtigt werden. Weiterhin sind Anwendungen einerseits oft nicht an Objekten interessiert, die mehr als eine gewisse Distanz entfernt sind, z. B. bei der Suche nach dem nächsten Taxi; andererseits nimmt der Aufwand für eine Nachbarschaftsanfrage auch mit der Größe des zu durchsuchenden Gebiets zu (siehe Abschnitt 5.4.5). Aus diesen Gründen erlaubt es der Parameter *maxDist*, den maximalen Abstand anzugeben, den ein mobiles Objekt der Ergebnismenge

von der gesuchten Position *p* haben darf. Das mobile Objekt, das als Ergebnis der Nachbarschaftsanfrage zurückgeliefert wird, hat folgende Eigenschaften:

$$nearestObj = (o, ld(o)) \text{ with } o \in O \text{ and}$$
$$DISTANCE(ld(o).pos, p) \leq DISTANCE(ld(o').pos, p)\ \forall o' \in O \text{ and}$$
$$ld(o).acc \leq reqAcc \text{ and } ld(o').acc \leq reqAcc \text{ and}$$
$$DISTANCE(ld(o).pos, p) \leq maxDist; \varepsilon \text{ else}$$

Infolgedessen hat auch kein anderes mobiles Objekt, dessen Positionsinformation die geforderte Genauigkeit aufweist, eine geringere Entfernung zu *p* als *DISTANCE(ld(o).pos, p) - reqAcc* (siehe Abbildung 4-5). Ein Klient kann dieser untere Grenze beispielsweise dazu verwenden, die maximale Sendestärke für eine drahtlose Kommunikation zu bestimmen, mit der kein anderer mobiler Sender gestört wird. Voraussetzung dafür ist, dass alle Sender ihre Position mit der geforderten Genauigkeit beim LS angemeldet haben.

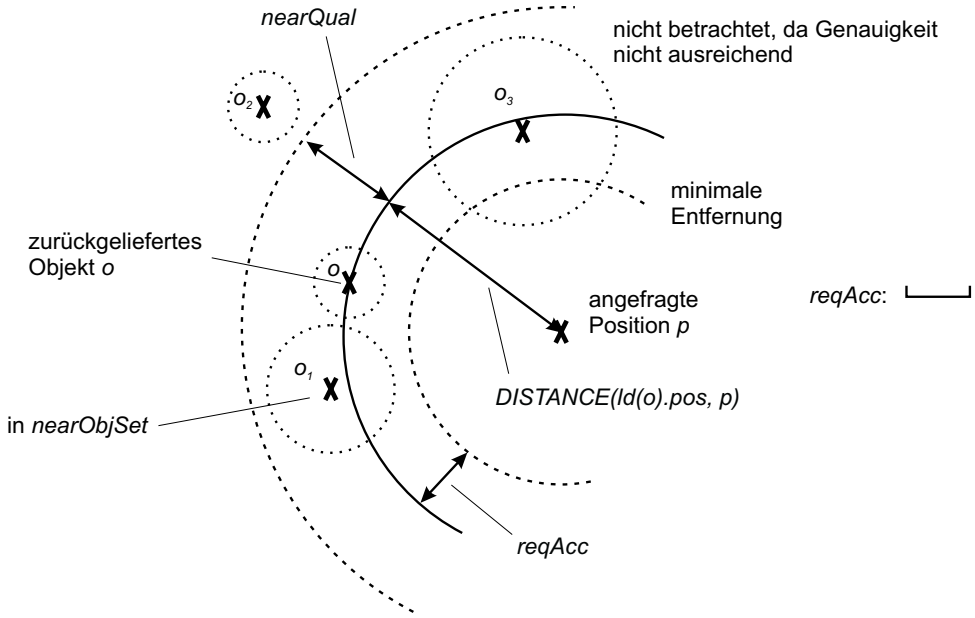


Abbildung 4-5. Beispielszenario für eine Nachbarschaftsanfrage.

Wegen der eingeschränkten Genauigkeit der im LS gespeicherten Positionsinformationen kann allerdings auch nicht garantiert werden, dass das so bestimmte Objekt tatsächlich der nächste Nachbar zu Punkt *p* ist. In dem in Abbildung 4-5 gezeigten Szenario kann zum Beispiel die wirkliche Position von Objekt o_1 näher bei *p* sein als die von *o*, wenn die Position von o_1 innerhalb der näher bei *p* liegenden Hälfte des entsprechenden Aufenthaltsbereichs liegt und die des

Objekts *o* in dem von *p* weiter entfernten Teil seines Aufenthaltsbereichs. Allerdings kann gezeigt werden, dass damit das Objekt ausgewählt wird, das mit der höchsten Wahrscheinlichkeit der nächste Nachbar zu *p* ist. Die Wahrscheinlichkeit hängt zwar neben der Entfernung auch von der Größe des Aufenthaltsbereichs ab, jedoch ist deren Einfluss sehr gering, wenn die Entfernung zu *p* vergleichsweise groß ist[4] (für Details siehe Anhang E).

Bedingt durch diese Ungenauigkeit werden einige Anwendungen nicht nur an dem nächsten Nachbarn, sondern auch an anderen “nahen” Nachbarn von *p* interessiert sein. Was “nahe” bedeutet wird dabei durch den Parameter *nearQual* bestimmt. Wenn *o* das als nächster Nachbar zu *p* bestimmte Objekt ist, dann ist *nearObjSet* folgendermaßen definiert:

$$\begin{aligned} nearObjSet = \{\ &(o', ld(o')) \mid\ o' \in O,\ o' \neq o \text{ and } DISTANCE(ld(o').pos, p) \leq \\ &(DISTANCE(ld(o).pos, p) + nearQual) \text{ and} \\ &ld(o').acc \leq reqAcc \text{ and} \\ &DISTANCE(ld(o').pos, p) \leq maxDist\ \} \end{aligned}$$

Wird *nearQual* auf null gesetzt, so ist *nearObjSet* gleich der leeren Menge. Durch Setzen von *nearQual* auf $2 \cdot reqAcc$ kann garantiert werden, dass alle Objekte, die sich möglicherweise näher an *p* befinden als *o*, in der Ergebnismenge der entsprechenden Anfrage enthalten sind. In dem Beispielszenario, das in Abbildung 4-5 dargestellt ist, gehört zum Beispiel Object o_1 zu *nearObjSet*. Objekt o_2 gehört jedoch nicht dazu, da die Position von o_2 außerhalb des Kreises um *p* liegt, der durch einen Radius entsprechend der Entfernung zwischen *o* und *p* + *nearQual* gebildet wird. Objekt o_3 wird wiederum nicht betrachtet, da es nicht den Genauigkeitsanforderungen entspricht.

4.4 Ereignisse

Anders als bei den bisher diskutierten Anfragen, die entsprechend einem klassischen Klient-Server-Modell ablaufen, ist bei ortsbezogenen Anwendungen oft eine Interaktion nach dem *Publish&Subscribe*-Modell sinnvoll, bei dem Klienten vom LS über das Eintreten bestimmter Ereignisse (engl. *events*) benachrichtigt werden (für einen Vergleich dieser beiden Interaktionsformen siehe z. B. COULOURIS, DOLLIMORE & KINDBERG (2001)). Ein Klient meldet sich dazu mit einer Beschreibung des ihn interessierenden Ereignisses beim LS an (engl. *subscribe*) und erhält

4. Wieder unter der Annahme, dass die Aufenthaltswahrscheinlichkeit innerhalb des Aufenthaltsbereichs gleichverteilt ist.

eine Benachrichtigung (engl. *notification*) wenn das betreffende Ereignis eintritt. Da Anwendungen mit Ortsbezug zumeist auf Veränderungen in der Umgebung des Benutzers reagieren (siehe auch SCHILIT, ADAMS & WANT (1994)), kann ein ereignisorientiertes Interaktionsmodell oft deren Entwicklung erleichtern.

Folgende Ereignisse sind im Zusammenhang mit mobilen Objekten und räumlichen Informationen vorstellbar. Ein Ereignis wird definiert, indem ein entsprechendes Prädikat beim LS registriert wird. Jedesmal wenn dieses Prädikat wahr wird, sendet der LS bis zu dessen Deregistrierung eine entsprechende Benachrichtigung zurück an den Klienten.

- *OnEnter*: Es wird eine Benachrichtigung erzeugt, sobald ein mobiles Objekt das durch den Parameter *a* beschriebene Gebiet betritt. Dieses Ereignis könnte z. B. verwendet werden, um Besucher eines Messestandes automatisch mit Informationsmaterial zu versorgen. Bei der Feststellung, ob dieses Ereignis eingetreten ist, gilt dieselbe Semantik wie bei der Gebietsanfrage mit den entsprechenden Parametern *reqAcc* und *reqOverlap*. Die Benachrichtigung enthält dann das betretene Gebiet und das betretende mobile Objekt *o*.

  Prädikat: *onEnter(a, reqAcc, reqOverlap)*
  Benachrichtigung: *notifyEnter(o, a)*

- *OnLeave*: Entspricht *onEnter* nur dass eine Benachrichtigung erzeugt wird, sobald ein mobiles Objekt das angegebene Gebiet verlässt.

  Prädikat: *onLeave(a, reqAcc, reqOverlap)*
  Benachrichtigung: *notifyEnter(o, a)*

- *OnCrossing*: Das *onCrossing*-Ereignis beschreibt das Übertreten der durch den Parameter *line* angegebenen Linie durch ein mobiles Objekt.

  Prädikat: *onCrossing(line, reqAcc, reqOverlap)*
  Benachrichtigung: *notifyCrossing(o, line)*

- *OnMeeting*: Dieses Ereignis wird ausgelöst, wenn sich die in der als erstem Parameter übergebenen Liste befindenden mobilen Objekte $(o_1, o_2, ...)$ mit einem maximalen Abstand von *dist* zusammenfinden, wobei deren Positionsinformation mit der Genauigkeit *reqAcc* verfügbar sein muss. Die bei einer Besprechung ausgetauschten oder entstandenen Dokumente könnten so beispielsweise automatisch mit Daten über die dabei anwesenden Personen annotiert werden (siehe LAMMING & FLYNN (1994)).

  Prädikat: $onMeeting(\{o_1, o_2, ...\}, dist, reqAcc)$
  Benachrichtigung: $notifyMeeting(\{o_1, o_2, ...\})$

- *ContPosUpdate*: Dieses Ereignis entspricht einer kontinuierlichen Positionsanfrage, wobei der Klient bis zur Abmeldung des Ereignisses regelmäßig über die Position eines bestimmten mobilen Objekts *o* informiert wird. Die Häufigkeit der Benachrichtigung ist dabei durch den Parameter *reportInterval* gegeben, der das zwischen aufeinander folgenden Benachrichtigungen liegende Zeitintervall festlegt.

  Prädikat: *contPosUpdate(o, reqAcc, reportInterval)*
  Benachrichtigung: *notifyPos(ld(o))*

- *ContAreaUpdate*: Entsprechend zu *contPosUpdate* für alle Objekte innerhalb eines bestimmten geographischen Gebiets *a*.

  Prädikat: *contAreaUpdate(a, reqAcc, reqOverlap, reportInterval)*
  Benachrichtigung: *notifyArea({(o_1, ld(o_1)), (o_2, ld(o_2)), ... })*

Darüber hinaus sind weitere, auch zusammengesetzte, Ereignisse denkbar. Die Betrachtung von Ereignismechanismen würde jedoch den Rahmen dieser Arbeit sprengen. Der interessierte Leser findet in BAUER (2000) und DUDKOWSKI (2002) weitere Informationen zu dieser Thematik sowie zu weiterführenden Arbeiten auf diesem Gebiet.

4.5 Zusammenfassung

In diesem Kapitel wurde ein Dienstmodell für einen Lokationsdienst entwickelt, das auf die speziellen Eigenschaften der Positionsinformationen mobiler Objekte und die Anforderungen von Anwendungen mit Ortsbezug ausgerichtet ist. In diesem Dienstmodell wird die Semantik der einzelnen vom LS anzubietenden Operationen beschrieben, unter Berücksichtigung der von den Sensorsystemen und den Sicherheitsbedenken der Benutzer vorgegebenen eingeschränkten Genauigkeit der Positionsinformationen sowie der von den Klienten des LS in den Anfragen geforderten Genauigkeit. Die nächsten Kapitel beschäftigen sich mit der Architektur eines diesem Dienstmodell entsprechenden verteilten Lokationsdienstes, der die für die Verwaltung genauer Positionsinformationen einer hohen Zahl mobiler Objekte notwendige Skalierbarkeit und Effizienz aufweist.

Kapitel 5

Architektur und Funktionsweise des Lokationsdienstes

Die für den LS geforderte Skalierbarkeit und Effizienz machen eine Realisierung durch eine verteilte Architektur erforderlich. In diesem Kapitel sollen die grundlegenden Mechanismen beschriebenen werden, die zur Datenhaltung der gespeicherten Positionsinformationen und die Ausführung der im vorigen Kapitel definierten Operationen in einer solchen verteilten Architektur notwendig sind. Die hier vorgestellten Konzepte wurden erstmals in LEONHARDI & KUBACH (1999) und LEONHARDI & ROTHERMEL (2002) veröffentlicht.

Nach einer Beschreibung des verwendeten System- und Fehlermodells in Abschnitt 5.1 gibt Abschnitt 5.2 zuerst einen Überblick über die Architektur des LS. Abschnitt 5.3 beschreibt die Datenhaltung und Wiederherstellung der Daten einzelner Lokations-Server. Im Detail werden die vorgestellten Algorithmen in Abschnitt 5.4 besprochen und Abschnitt 5.5 beschäftigt sich mit möglichen Optimierungen.

5.1 System- und Fehlermodell

Bei der Beschreibung der Architektur und Algorithmen des LS wird das folgende System- und Fehlermodell angenommen, das auf der Terminologie und Klassifizierung von JALOTE (1994) beruht. Das Systemmodell beschreibt dabei die Bestandteile eines Systems, das Fehlermodell die Annahmen über die Fehler, die in diesem System auftreten können.

Ein verteiltes System, wie es die am LS beteiligten Einheiten (Lokations-Server, mobile Objekte und Klienten) bilden, setzt sich aus mehreren autonomen Knoten (d. h. den dazugehörenden

Rechnern) und einem diese Knoten verbindenden Netzwerk zusammen. Bei den Knoten wird weiter unterschieden zwischen stationären Knoten (Lokations-Server) und mobilen Knoten (mobile Objekte). Klienten können sowohl stationär als auch mobil sein. Ein Knoten besteht aus Sicht des LS aus einem Prozessor und privatem flüchtigem und stabilem Speicher. Während die auf flüchtigem Speicher gehaltenen Daten bei einem Ausfall (engl. *fault*) verloren gehen, sind die auf stabilem Speicher nach einer Wiederherstellung des Knotens wieder verfügbar. Über das Kommunikationsnetzwerk können zwei beliebige Knoten Nachrichten miteinander austauschen (Punkt-zu-Punkt-Kommunikation). Für mobile Knoten setzen wir dabei einen geeigneten Mechanismus, wie beispielsweise *Mobile IP* (PERKINS (1996)) voraus, der die Kommunikation unabhängig von deren aktueller geographischer Position und Bewegung ermöglicht.

In unserem Fehlermodell gehen wir davon aus, dass Knoten nur Zusammenbruchsfehlern (engl. *crash fault*), wie in JALOTE (1994) definiert, unterliegen. Da der LS als Teil einer Infrastruktur für ortsbezogene Anwendungen eingesetzt werden soll, kann davon ausgegangen werden, dass die Lokations-Server nach einem Fehler repariert und neu gestartet werden. Wir nehmen daher weiterhin an, dass ein stationärer Knoten (Lokations-Server) nach einem Zusammenbruch nach endlicher Zeit wiederhergestellt wird und dass die auf stabilem Speicher gehaltenen Daten wieder verfügbar sind. Die mobilen Endgeräten, welche zumeist die Funktionalität der beim LS angemeldeten mobilen Objekte und der Klienten erbringen, sind allerdings im Vergleich zu stationären Servern wesentlich anfälliger für Fehler (siehe z. B. SATYANARAYANAN (1996)). Aus diesem Grunde gehen wir nicht davon aus, dass mobile Knoten (mobile Objekte und Klienten) in allen Fällen nach eine Fehler wiederhergestellt werden.

Durch Zusammenbruchsfehler in der Kommunikationsschicht, was wiederum häufiger bei Mobilfunkverbindungen als bei Festnetzverbindungen auftreten wird, können darüber hinaus zeitweilige Netzwerkpartitionierungen entstehen. Knoten innerhalb einer Partition können kommunizieren, Knoten unterschiedlicher Partitionen nicht. Wir gehen davon aus, dass innerhalb einer Netzwerkpartition ein zuverlässiger Nachrichtenaustausch möglich ist, der durch übliche Mechanismen wie Empfangsbestätigungen und das wiederholte Senden von Nachrichten erreicht werden kann.

5.2 Architektur des Lokationsdienstes

In diesem Abschnitt sollen zunächst die grundlegenden Konzepte und Mechanismen für die Architektur des verteilten Lokationsdienstes vorgestellt werden. Die hier nur kurz angerissenen Algorithmen für Registrierung, Zuständigkeitsübergabe und Anfragebearbeitung werden in

Abschnitt 5.4 näher betrachtet. Auf eine Darstellung von möglichen Optimierungen wurde dort verzichtet, sie werden zusammenfassend im darauf folgenden Abschnitt besprochen.

5.2.1 Hierarchische Struktur des Lokationsdienstes

Wenn der LS zukünftig als Teil eines ortsbezogenen Informationssystems eingesetzt werden soll, so ist z. B. in einer größeren Stadt mit hunderttausenden von mobilen Objekten und Benutzern zu rechnen. Um die Skalierbarkeit für einen solchen großflächigen Einsatz zu gewährleisten, sind die Lokations-Server, aus denen sich eine Installation des LS zusammensetzt, in einer hierarchischen Struktur organisiert. Vergleichbare hierarchische Architekturen wurden für die Lokationsverwaltung in Mobilfunknetzen (WANG (1993)) oder für den GLOBE Lokationsdienst für mobile Software-Objekte (VAN STEEN ET AL. (1998)) vorgeschlagen.

Ein Lokationsdienst wird so konfiguriert, dass er ein bestimmtes geographisches Gebiet abdeckt, sein sogenanntes Dienstgebiet. Der Lokationsdienst ist für die Verwaltung von mobilen Objekten zuständig, die sich innerhalb seines Dienstgebiets aufhalten; wenn eine mobiles Objekt dieses Dienstgebiet verlässt, wird es automatisch abgemeldet. Das Dienstgebiet des LS kann hierarchisch strukturiert werden, indem es in verschiedene Teildienstgebiete aufgeteilt wird, die wiederum selbst aus verschiedenen Teildienstgebieten bestehen können, und so weiter. Die Form eines Dienstgebiets wird dabei durch ein beliebiges zusammenhängendes Polygon beschrieben. Falls ein Dienstgebiet weiter unterteilt ist, muss dessen geographisches Gebiet vollständig durch das der untergeordneten Dienstgebiete abgedeckt sein. Die untergeordneten Dienstgebiete dürfen sich dabei nicht gegenseitig überlappen.

Einem Dienstgebiet ist jeweils ein Lokations-Server zugeordnet, der dafür zuständig ist, die mobilen Objekte zu verwalten, die sich innerhalb des entsprechenden geographischen Gebiets aufhalten. Der hierarchischen Struktur der Dienstgebiete entsprechend, sind die Lokations-Server ebenfalls in einer Hierarchie organisiert. Ein Wurzel-Server (engl. *root server*) ist für das Gesamtdienstgebiet des LS zuständig und hat demnach auch als einziger Lokations-Server keine Vorgänger. Blatt-Server (engl. *leaf server*) hingegen haben keine Nachfolger und sind für nicht weiter unterteilte Dienstgebiete zuständig. Dazwischen liegende Server werden als innere Server bezeichnet und haben sowohl Vorgänger als auch Nachfolger. Abbildung 5-1 zeigt ein Beispiel für einen Ausschnitt einer dreischichtigen Server-Hierarchie und die dazugehörenden Dienstgebiete.

Bei den Aufgaben der Lokations-Server kann grundlegend zwischen Blatt-Servern und Nicht-Blatt-Servern unterschieden werden:

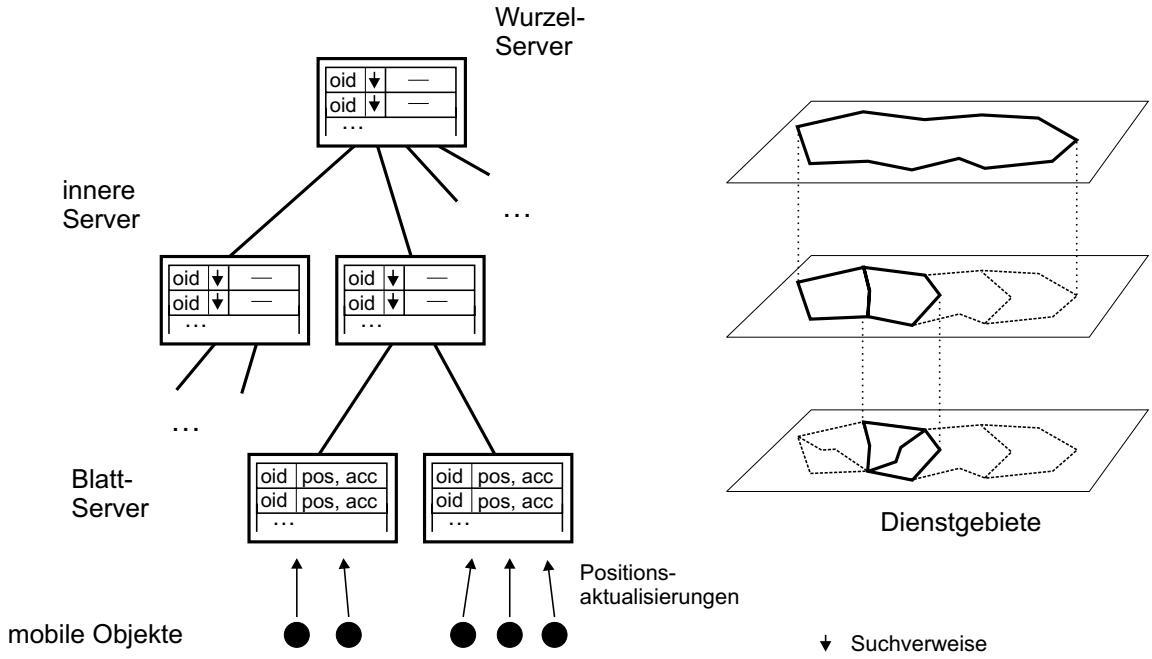


Abbildung 5-1. Hierarchische Architektur des Lokationsdienstes.

- *Blatt-Server* sind für die nicht weiter unterteilten Dienstgebiete auf unterster Ebene der Hierarchie zuständig. Ein Blatt-Server ist für die Verwaltung der Positionsinformationen der beim LS angemeldeten mobilen Objekte, die sich innerhalb seines Dienstgebiets aufhalten, verantwortlich. Da sich die Dienstgebiete einer Ebene nicht überlappen dürfen, ist zu jeder Zeit für ein bestimmtes mobiles Objekt nur ein Blatt-Server zuständig und empfängt von diesem die Positionsaktualisierungen. Wir nennen diesen Server den *Agenten* des mobilen Objekts (in Anlehnung an Mobile IP, PERKINS (1996)). Wenn sich das mobile Objekt von einem Dienstgebiet in ein anderes bewegt, muss die Zuständigkeit für die Positionsinformation an den entsprechenden Blatt-Server übergeben werden, der dann zum neuen Agenten des mobilen Objekts wird.
- Ein *Nicht-Blatt-Server* ist einem Dienstgebiet zugeordnet, das der Vereinigung der Dienstgebiete seiner Nachfolger entspricht. Die Größe der Dienstgebiete nimmt damit ausgehend von den Blatt-Server bis hin zum Wurzel-Server zu, dessen Dienstgebiet schließlich dem Gesamtdienstgebiet des LS entspricht. Die Nicht-Blatt-Server dieser Hierarchie sind für das Auffinden eines bestimmten Objekts zuständig und verwalten für jedes mobile Objekt innerhalb ihres Dienstgebiets einen *Suchverweis.* Der Suchverweis gibt denjenigen der Nachfolger des Nicht-Blatt-Servers an, der für das Teildienstgebiet zuständig ist, in dem

sich das betreffende mobile Objekt aufhält. Da sich die Dienstgebiete der Nachfolger nicht überlappen dürfen, ist dieser eindeutig bestimmt.

Die Positionsinformationen für ein mobiles Objekt werden demnach nur auf den Blatt-Servern gespeichert (in Form des zuletzt gemeldeten Positionierungsdatensatzes), während ein Nicht-Blatt-Server für das mobile Objekt einen Verweis auf denjenigen seiner Nachfolger besitzt, dessen Unterhierarchie den Blatt-Server mit der gesuchten Positionsinformation einschließt. Die Suchverweise bilden damit einen Suchpfad, der vom Wurzel-Server bis zum Agenten des mobilen Objekts reicht, der dessen Positionsinformationen vorhält.

Die aktuelle Positionsaktualisierung eines mobilen Objekts wird immer zum Agenten des Objekts gesendet. Wenn ein mobiles Objekt sein aktuelles Dienstgebiet verlässt und ein neues betritt, muss die Zuständigkeit für dieses auf den zugehörigen Lokations-Server übertragen und der Suchpfad entsprechend angepasst werden. Der Bearbeitungsvorgang muss dann sicherstellen, dass das mobile Objekt über seinen neuen Agenten informiert wird. Wieviele Server von einer Pfadaktualisierung betroffen sind, hängt von Höhe des niedrigsten Teilbaums ab, der sowohl den alten als auch den neuen Agenten des mobilen Objekts enthält. Auf Grund der hierarchischen Struktur der Dienstgebiete, betrifft eine Zuständigkeitsübergabe allerdings die oberen Ebenen wesentlich seltener als die unteren.

5.2.2 Mechanismen des Lokationsdienstes

In diesem Abschnitt soll ein Überblick über die Mechanismen zur Zuständigkeitsübergabe und Anfragebearbeitung in der hierarchischen Struktur des LS gegeben werden. Abbildung 5-2 zeigt dazu eine dreistufige Server-Hierarchie, die in den folgenden Beispielen verwendet wird. Die Algorithmen werden ausführlich in Abschnitt 5.4 besprochen.

In unserem Beispiel für eine Zuständigkeitsübergabe (Abbildung 5-2 links) empfängt Server 4 (kurz s_4) eine Positionsaktualisierung von einem mobilen Objekt *o*, dessen Position nicht mehr in seinem Dienstgebiet liegt. Er sendet daraufhin eine Anforderung für eine Zuständigkeitsübergabe an seinen Vorgänger. Dieser erkennt, dass sich *o* weiterhin in seinem Dienstgebiet befindet und leitet es daher an denjenigen seiner Nachfolger (in diesem Fall s_5) weiter, dessen Dienstgebiet die neue Position von *o* enthält, anstatt es nach oben weiterzugeben. Der Blatt-Server s_5 wird damit der neue Agent von *o* und sendet eine Bestätigung der Zuständigkeitsübergabe auf demselben Weg zurück an s_4, wodurch der Suchpfad für *o* aktualisiert wird. Davor benachrichtigt s_5 das mobile Objekt *o* über die Zuständigkeitsübergabe und meldet sich als dessen neuer Agent.

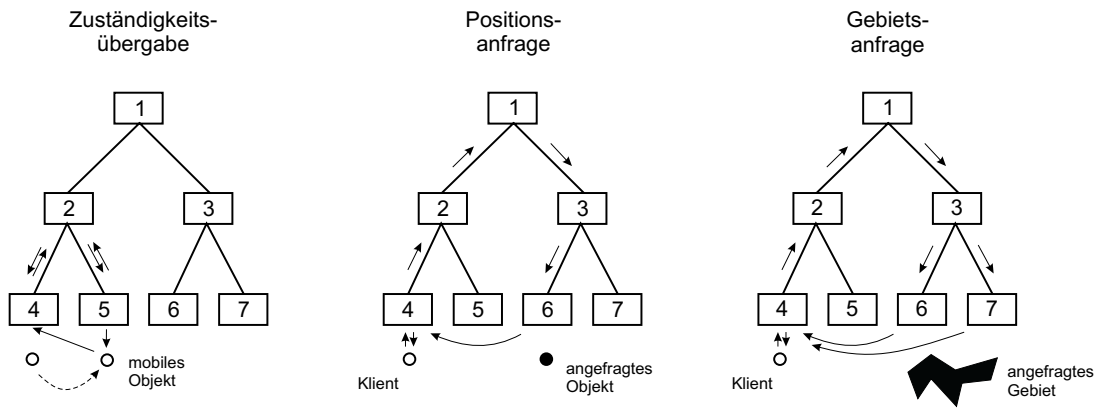


Abbildung 5-2. Grundlegende Mechanismen des Lokationsdienstes.

Anfragen können an einen beliebigen Blatt-Server des LS gestellt werden. Damit lokale Anfragen effizient bearbeitet werden können, sollte ein mobiler Klient dabei den Blatt-Server kontaktieren, dessen Dienstgebiet seine aktuelle Position enthält bzw. das zu seiner Position am nächsten liegt (siehe auch Abschnitt 5.4).

Bei der Bearbeitung einer Positionsanfrage wird der Suchpfad des betreffenden mobilen Objekts verwendet, um dessen Agenten zu finden. In unserem Beispiel gehen wir davon aus, dass eine Positionsanfrage an s_4 gesendet wurde und sich das gesuchte Objekt *o* gegenwärtig im Dienstgebiet von s_6 aufhält. Die Positionsanfrage wird in diesem Fall bis zum Wurzel-Server hochpropagiert, da erst dieser einen Suchverweis für *o* besitzt. Von dort ab wird die Anfrage entlang *o*'s Suchpfad bis zu s_6 weitergeleitet, der die aktuellsten Positionsinformation für *o* an s_4 zurückliefert. In vielen Fällen ist es jedoch nicht notwendig, die Server-Hierarchie in ihrer gesamten Höhe zu durchlaufen (siehe unten). Falls sich *o* im Dienstgebiet von s_5 befindet, erreicht die Anfrage nur den Server s_2 auf der nächsthöheren Ebene.

Für die Beschreibung der Gebietsanfrage nehmen wir an, dass sich das angefragte Gebiet *a* mit den Dienstgebieten von s_6 und s_7 überlappt. Anstatt der Suchpfade für die mobilen Objekte wird in diesem Fall die hierarchische Struktur der Dienstgebiete verwendet, um die zuständigen Server zu finden. Eine Anfrage wird ausgehend von s_4 nach oben weitergegeben, bis ein Server gefunden wurde, dessen Dienstgebiet das angefragte Gebiet vollständig enthält, in diesem Fall s_1. Von dort aus wird die Anfrage in allen Zweigen nach unten weitergegeben, deren Dienstgebiete sich mit *a* überlappen, solange bis s_6 und s_7 erreicht sind. Beide ermitteln die mobilen Objekte, die sich in den entsprechenden Teilgebieten von *a* aufhalten, und senden die Ergebnisse zurück an s_4. Dieser ist dafür verantwortlich, aus diesen Teilergebnissen die Antwort zusammenzusetzen, die dann an den Klienten übergeben wird.

Nachbarschaftsanfragen sind vergleichbar mit Gebietsanfragen und werden in Abschnitt 5.4.5 ausführlich besprochen.

Die Möglichkeit, die Funktionalität des LS auf mehrere Rechner zu verteilen, ist eine wichtige Voraussetzung für einen skalierbaren Dienst. In früheren Untersuchungen wurde belegt, dass unter Berücksichtigung von Positions- und Gebietsanfragen eine Verteilung anhand von Gebieten, wie sie durch die Zuordnung von Servern zu Dienstgebieten gegeben ist, wesentliche Vorteile gegenüber einer Verteilung nach Objekten, z. B. indem jeder Server für eine bestimmte Menge von Objekte zuständig ist, bietet (siehe KUBACH ET AL. (1999)). Die hier vorgestellte hierarchische Struktur des LS wurde dabei gewählt, um die Lokalität von Operationen ausnutzen zu können.

Positionsaktualisierungen und Zuständigkeitsübergaben sind grundsätzlich lokal, da ein mobiles Objekt natürlich nur in ein benachbartes Dienstgebiet überwechseln kann (falls kein Ausfall der Kommunikationsverbindung oder des Positionierungssensors auftritt). Darüber hinaus gehen wir ebenfalls davon aus, dass Anfragen in einem hohen Maße lokal sind, da sich die Benutzer einer Anwendung mit Ortsbezug häufig für mobile Objekte in ihrer Umgebung interessieren werden (z. B. alle Taxis, die in fünf Minuten bei ihnen sein können, oder alle Personen im selben Raum). Eine hohe Lokalität von Anfragen wurde für ortsbezogene Anwendungen (HARTER & HOPPER (1994)), für Anrufe in einem Mobilkommunikationsnetz (LAM ET AL. (1998)) und im Falle des Globe Lokationsdienstes für Software-Objekte (VAN STEEN ET AL. (1998)) beobachtet.

Ein weiterer Vorteil der hierarchischen Struktur ist, dass sie sich einfach auf eine hierarchische Organisationsstruktur abbilden lässt, die Voraussetzung für die Administration eines Internetweiten Dienstes ist (vgl. z. B. das DNS des Internets, MOCKAPETRIS & DUNLAP (1988)).

Die Leistungsfähigkeit des Systems wird durch die Höhe der Server-Hierarchie, deren Verzweigungsgrad (d. h. die Zahl der Nachfolger eines Lokations-Servers) und der Größe der (Blatt-) Dienstgebiete beeinflusst. Eine optimale Einstellung dieser Parameter ist abhängig von der Zahl der beteiligten mobilen Objekte im Gesamtdienstgebiet, von deren Verteilung innerhalb dieses Gebiets (d. h. an welchen Stellen sie sich befinden) und von deren Bewegungsverhalten (z. B. durchschnittliche Geschwindigkeit und Bewegungsradius). Die Leistungsfähigkeit von unterschiedlichen Konfigurationen des LS wird in Kapitel 7 mittels Experimenten an einer protoypischen Implementierung betrachtet.

Darüber hinaus hängt die Leistungsfähigkeit des LS natürlich auch davon ab, welche Anfragen mit welcher Häufigkeit gestellt werden, d. h. wie der Anfrage-Mix aussieht, der von den Klienten des LS, erzeugt wird. Ein weiterer Aspekt ist der Aufstellungsort der Server, der die Kommunikationskosten und Anwortzeiten betrifft. Wegen der angenommenen hohen Lokalität von

Positionsaktualisierungen und Anfragen, sollten die Blatt-Server möglichst nahe (in Bezug auf Kommunikationsbandbreite und -verzögerung) bei den ihnen zugeordneten Dienstgebieten aufgestellt werden.

5.2.3 Replikation und Partitionierung

Bei den bisherigen Betrachtungen wurde angenommen, dass einem Dienstgebiet jeweils nur ein einzelner Lokations-Server zugeordnet ist. Aus Gründen der Fehlertoleranz und der Verbesserung des Durchsatzes kann es jedoch sinnvoll sein, einem Dienstgebiet mehr als einen Lokations-Server zuzuordnen und damit eine Replikation oder Partitionierung der Datenspeicherung zu erreichen. Besonders wichtig ist dies beim Wurzel-Server und den Servern auf den oberen Ebenen, die entsprechend der zunehmenden Größe der Dienstgebiete eine große Anzahl von mobilen Objekten verwalten müssen. Replikation und Partitionierung von Daten wird in der Literatur ausführlich betrachtet (siehe z. B. COULOURIS, DOLLIMORE & KINDBERG (2001)). In dieser Arbeit wird diese Problematik daher nur angesprochen und nicht weitergehend diskutiert.

Wie im Folgenden besprochen, verwaltet jeder Lokations-Server eine Datenbank, in der er die Informationen zu den mobilen Objekten in seinem Dienstgebiet speichert. Wenn einem Dienstgebiet mehrere Lokations-Server zugeordnet sind, kann die entsprechende Datenbank sowohl im Ganzen auf diesen Servern repliziert, als auch partitioniert und auf die Server aufgeteilt werden (siehe Abbildung 5-3). Auch eine Kombination davon ist möglich.

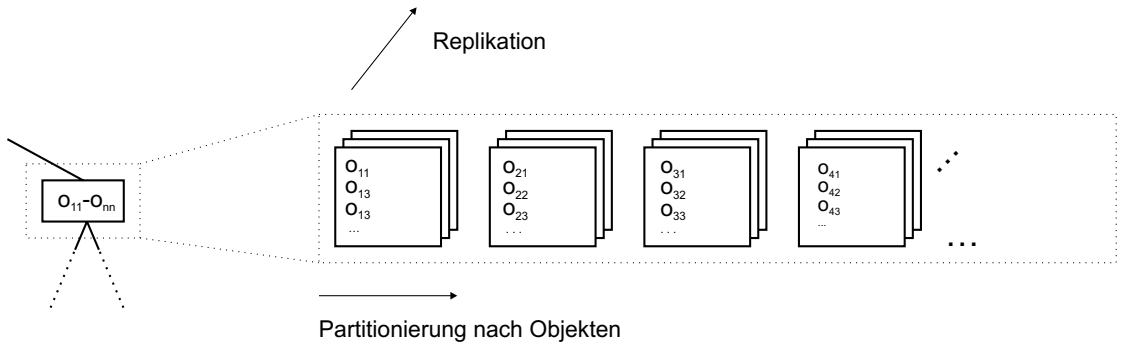


Abbildung 5-3. Replikation und Partitionierung der Daten eines Lokations-Servers.

Eine Replikation der Datenbank dient der Ausfallsicherheit und einer Verbesserung des Durchsatzes bei Anfragen. Da wir allerdings von einer häufigen Aktualisierung der Positionsinforma-

tionen ausgehen, würde deren Replikation zu einem hohen Aufwand führen. Es ist daher sorgfältig abzuwägen, ob und auf welcher Ebene eine Replikation der Datenbank sinnvoll ist.

Im zweiten Fall werden die mobilen Objekte, z. B. anhand des Objektbezeichners bzw. eines bestimmten Teils davon, auf die einzelnen Server verteilt (vgl. die Partitionierung des HLR in GSM, bei welcher der zuständige Server durch die ersten Ziffern der Mobilfunknummer bestimmt wird, MOULY & PAUTET (1992)). Damit kann die Belastung, die sich aus der Größe der zu verwaltenden Datenbank, der Häufigkeit der Aktualisierungen und der von Positionsanfragen (nicht aber Nachbarschafts- und Gebietsanfragen) ergibt, auf die einzelnen Server verteilt werden. Dies ist besonders für Server auf den höheren Ebenen sinnvoll.

5.3 Speicherung der Positionsinformationen

Um die im vorherigen Abschnitt beschriebene Funktionalität umzusetzten, muss ein Lokations-Server entsprechende Konfigurationsinformationen verwalten und in einer Datenbank Informationen über die mobilen Objekte in seinem Dienstgebiet speichern. Die dazu verwendeten Datenstrukturen und Mechanismen sollen im Folgenden besprochen werden.

5.3.1 Datenstrukturen

Jeder Lokations-Server *s* speichert einen Konfigurationsdatensatz *c* auf stabilem Speicher, der dazu dient, dessen Position in der Server-Hierarchie zu definieren und das dem Server zugeordnete Dienstgebiet mit den darin verfügbaren Sensorsystemen beschreibt. Ein Konfigurationsdatensatz enthält daher die folgenden Informationen:

c.sa: Der Eintrag *c.sa* beschreibt das Dienstgebiet, das dem Lokations-Server *s* zugeordnet ist.

c.parent: Dieser Eintrag identifiziert den in der Hierarchie direkt übergeordneten Server von *s*. Nur für den Wurzel-Server ist *s.parent* nicht definiert (ε).

c.children: In der Menge *c.children* ist jeder Nachfolger von *s* durch einen Datensatz beschrieben. Dieser Datensatz enthält im Feld *id* die Adresse des Nachfolgers und in *sa* dessen Dienstgebiet. Für einen Blattknoten ist *s.children* natürlich leer.

c.sensorMap: Der Eintrag *c.sensorMap* beschreibt die im Dienstgebiet von *s* verfügbaren Sensorsysteme.

Eine Sensorenkarte (engl. *sensor map*) wird für die Garantie einer minimalen Genauigkeit (siehe Abschnitt 4.2) benötigt und kann entfallen, wenn die Installation des LS die erweiterte Registrierung mit garantierter minimaler Genauigkeit nicht unterstützt. Die Sensorenkarte teilt das Dienstgebiet des Servers vollständig in disjunkte maximal große Teilgebiete auf, für die eine eindeutige Menge von verfügbaren Sensorsystemen angegeben werden kann. Für jedes dieser Teilgebiet enthält die Sensorkarte einen Eintrag der Form *(a* × *(sensorId* × *acc)*)*. Dieser Eintrag beschreibt die Ausdehnung des jeweiligen Teilgebiets *a* und enthält eine Liste mit einem eindeutigen Bezeichner *sensorId* und der Positionierungsgenauigkeit *acc* für jedes dort verfügbare Sensorsystem. Falls in einem Teilgebiet kein Sensor verfügbar ist, wird dies durch den Eintrag *(a, ∅)* beschrieben.

Diese Darstellung der Sensorkarte wurde gewählt, um eine effiziente Bestimmung der Überdeckung eines Teilgebiets durch Sensoren zu ermöglichen (siehe Abschnitt 5.4.1). Die Sensorkarte eines Lokations-Servers wird in der Initialisierungsphase aus einer Liste der im Dienstgebiet verfügbaren Sensorsysteme generiert und aktualisiert, wenn ein neues Sensorsystem hinzukommt oder eines wegfällt.

Ein beim LS angemeldetes mobiles Objekt wird als Besucher (engl. *visitor*) eines Lokations-Servers bezeichnet, wenn es sich zu dem entsprechenden Zeitpunkt im Dienstgebiet des Servers aufhält. Für die Informationen über seine Besucher verwaltet jeder Lokations-Server eine Besucherdatenbank (kurz: *visitorDB*), die für jeden Besucher einen sogenannten Besucherdatensatz (engl. *visitor record)* beinhaltet (vgl. das VLR in GSM, SCHILLER (2000)). Der Aufbau eines Besucherdatensatzes hängt davon ab, ob es sich bei dem Lokations-Server um einen Blatt- oder Nicht-Blatt-Server handelt.

Ein Besucherdatensatz *v* auf einem Blatt-Server des LS besteht aus folgenden Einträgen:

v.oId: Der (eindeutige) Bezeichner für das mobile Objekt.

v.offeredAcc: Beschreibt die Genauigkeit mit der die Positionsinformation für das mobile Objekt gegenwärtig im LS verfügbar ist.

v.regInfo: Registrierungsinformationen für das mobile Objekt bestehend aus den Felder *regInst, oInfo, acl, desAcc, minAcc, regArea* und *timeout*, welche die Adresse der registrierenden Instanz, Informationen über das mobile Objekt, die Zugriffskontrollliste, die gewünschte und minimale Genauigkeit, das Registrierungsgebiet sowie eine Zeitvorgabe für die automatische Abmeldung angeben, entsprechend zu den gleichnamigen Parametern der in Abschnitt 4.2 beschriebenen Registrierungsfunktion.

Auf einem Nicht-Blatt-Server enthält der Besucherdatensatz *v* anstelle der Registrierungsinformationen nur einen Verweis, der das Auffinden des zuständigen Blatt-Servers ermöglicht.

v.oId: Der Bezeichner des mobilen Objekts.

v.forwardRef: Ein Verweis auf den nächsten untergeordneten Server, der auf dem Suchpfad zu dem Agenten des betreffenden mobilen Objekts liegt.

Darüber hinaus speichert jeder Blatt-Server eine Datenbank mit den aktuellen Positionsinformationen der mobilen Objekte (kurz: *sightingDB*). Diese beinhaltet für jedes mobile Objekt den zuletzt übertragenen Positionierungsdatensatz.

5.3.2 Datenhaltung

Zur Verwaltung der im vorigen Abschnitt beschriebenen Datenstrukturen besitzt ein Lokations-Server die beiden Datenbanken *visitorDB* und *sightingDB*, die unterschiedlich behandelt werden (siehe Abbildung 5-4).

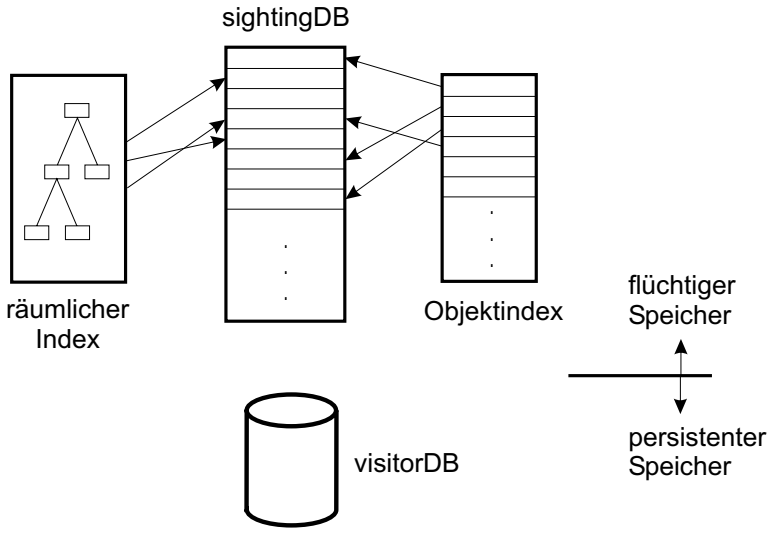


Abbildung 5-4. Komponenten der Datenhaltung auf einem Lokations-Server.

Die *visitorDB* mit den Registrierungs- und Suchpfadinformationen muss nur aktualisiert werden, wenn ein neues Objekt registriert, ein registriertes Objekt abgemeldet oder eine Zuständigkeitsübergabe durchgeführt wird, was im Vergleich zu Positionsaktualisierungen weit weniger häufig vorkommt. Sie wird daher auf stabilem Speicher gehalten. Dies ermöglicht es, nach ei-

nem Systemausfall den Suchpfad und die Registrierungsinformationen für ein mobiles Objekt wiederherzustellen.

Im Gegensatz dazu wird die *sightingDB* aus den zwei folgenden Gründen im flüchtigen Speicher gehalten. Da der LS mit dem Ziel entworfen wurde, möglichst aktuelle Positionsinformationen zu speichern, gehen wir erstens von einer häufigen Aktualisierung der Positionsinformationen aus, die deshalb im Besonderen effizient ausgeführt werden sollen (siehe die Messergebnisse zu unserem Prototyp in Abschnitt 7.2). Zweitens sind die gespeicherten Positionsinformationen damit auch nur für einen kurzen Zeitraum gültig und es macht keinen Sinn einen großen Aufwand in ihre Wiederherstellung zu investieren, da die wiederhergestellten Positionen mit hoher Wahrscheinlichkeit veraltet sind. Anstelle dessen werden die Positionsinformationen durch die von den mobilen Objekten empfangenen Positionsaktualisierungen automatisch wieder im flüchtigen Speicher hergestellt. Die Mechanismen zur Wiederherstellung der Positionsinformationen werden ausführlich im nächsten Abschnitt beschrieben.

Um Gebiets- und Nachbarschaftsanfragen effizient durchführen zu können, verwenden wir einen mehrdimensionalen Index über den Positionsinformationen in den Positionierungsdatensätzen. Für einen Überblick und eine Beschreibung verschiedener mehrdimensionaler Indizes siehe GAEDE & GÜNTHER (1998). Beispiele für solche mehrdimensionalen Indizes sind z. B. der Quadtree (SAMET (1990)) oder der R-Baum (GUTTMANN (1984)). Im Rahmen einer Diplomarbeit (HÄSSLER (2001)) wurden verschiedene repräsentative Vertreter von mehrdimensionalen Indexstrukturen im Hinblick auf ihren Einsatz in einer Hauptspeicherdatenbank des LS untersucht. Im Einzelnen wurden dort kd-Baum, Grid-File, Quadtree und R-Baum verglichen. Unter diesen speziellen Bedingungen hat sich der Quadtree, der auch für die Lokationsverwaltung beim Sentient Computing Projekt eingesetzt wird (ADDLESEE ET AL. (2001)), als am geeignetesten erwiesen (ähnlich effizient ist nur das Grid-File, das aber mit einem höheren Aufwand verbunden ist). In unserer Implementierung des LS haben wir daher einen Punkt-Quadtree[1] eingesetzt (siehe Abschnitt 7.1).

Für eine effiziente Ausführung von Positionsanfragen wird ein Hash-Index über die Objektbezeichner verwendet. Beide Indizes werden auf flüchtigem Speicher gehalten. Unsere in Abschnitt 7.2 beschriebenen Messergebnisse zeigen, dass selbst der mehrdimensionale Index sehr schnell aufgebaut werden kann (z. B. bei Eintreffen der von den mobilen Geräten nach einer Wiederherstellung eines Lokations-Servers gesendeten Aktualisierungsnachrichten).

1. Falls auch das Ausführen von dreidimensionalen Abfragen durch die Indexstruktur unterstützt werden soll (vgl. Abschnitt 4.1), muss hier anstelle des Quadtree ein entsprechend aufgebauter Octtree verwendet werden.

5.3.3 Wiederherstellung nach Systemausfällen

In diesem Abschnitt sollen Mechanismen zur Behandlung von Ausfällen mobiler Objekte und Lokations-Server betrachtet werden. Es wird dabei das in Abschnitt 5.1 beschriebene Fehlermodell vorausgesetzt und angenommen, dass diese nur von Zusammenbruchsfehlern betroffen sind.

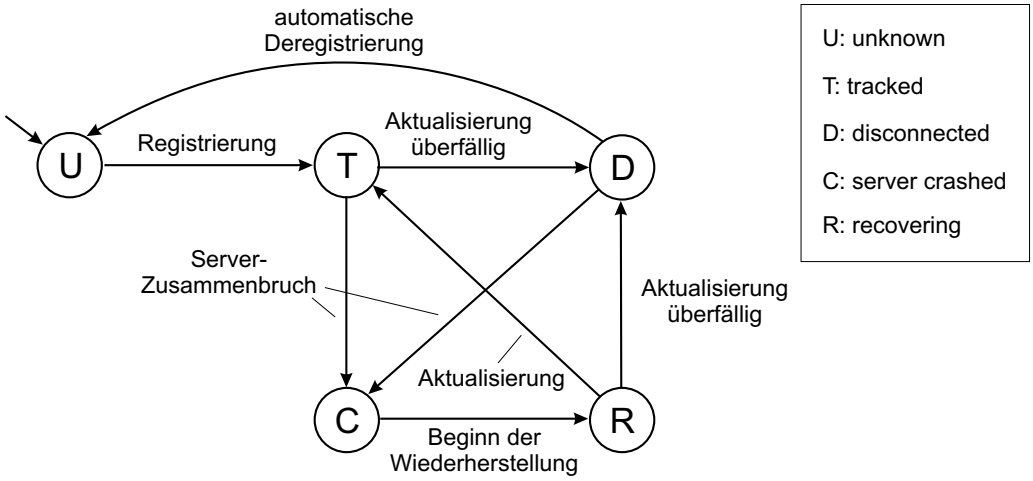


Abbildung 5-5. Zustände eines mobiles Objekt bei der Speicherung auf einem Lokations-Server.

Die im LS gespeicherten **mobilen Objekte** beachten dabei ein maximales Zeitintervall zwischen ihren Positionsaktualisierungen, *miu*. In Kapitel 6 werden verschiedene Protokolle zur Aktualisierung von Positionsinformationen betrachtet und es zeigt sich, dass es für alle Protokolle möglich ist, einen entsprechenden Wert für *miu* anzugeben. Falls der Agent des mobilen Objekts länger als *miu* auf die nächste Aktualisierungsnachricht warten muss, wird der Zustand des mobilen Objekts vom normalen *tracked* auf *disconnected* gesetzt, um anzugeben, dass dieses voraussichtlich nicht erreichbar und die Positionsinformation daher veraltet ist (siehe Abbildung 5-5). In diesem Zustand wird eine entsprechende Fehlermeldung als Ergebnis auf eine Positionsanfrage zurückgegeben. Bei der Bearbeitung von Gebiets- und Nachbarschaftsanfragen wird das Objekt ignoriert.

Falls sich das mobile Objekt länger als ein bestimmtes Zeitintervall *rtv* (engl. *registration timeout value*), der für jedes Objekt bei der Registrierung vorgegeben und in *v.regInfo.timeout* gespeichert wird[2], im Zustand *disconnected* befindet, wird es automatisch deregistriert. Damit halten wir uns an das sogenannte *Soft-State*-Prinzip (CLARK (1988)), um sicherzustellen, dass

die zu einem mobilen Objekt gehörenden Informationen schlussendlich wieder von den Lokations-Servern entfernt werden, selbst wenn das mobile Objekt nach einem Zusammenbruchsfehler nicht wiederhergestellt wird. Falls ein Objekt vor Ablauf des Zeitintervalls *rtv* wiederhergestellt wird, sendet es einfach eine neue Positionsaktualisierung an seinen Agenten und kehrt dort wieder zum Zustand *tracked* zurück. Falls es erst danach wiederhergestellt wird, muss es sich wieder beim LS anmelden.

Bei der Wiederherstellung nach einem Zusammenbruchsfehler lädt ein **Lokations-Server** seine Konfigurationsinformationen von stabilem Speicher und findet so seinen Platz in der Server-Hierarchie wieder. Lokations-Server stellen darüber hinaus die Suchverweise (Nicht-Blatt-Lokations-Server) bzw. die Registrierungsinformationen (Blatt-Lokations-Server) für die mobilen Objekte in ihrem Dienstgebiet wieder von stabilem Speicher her. Die Positionsinformationen der mobilen Objekte, die im Hauptspeicher gehalten werden, gehen jedoch bei einem Zusammenbruch verloren (siehe Abschnitt 5.3.2). Der Zustand der mobilen Objekte wird daher nach einer Wiederherstellung eines Blatt-Lokations-Servers auf *recovering* gesetzt. Um diese wiederherzustellen können zwei gegensätzliche Ansätze verwendet werden: (a) ein objektinitiierter und (b) ein Server-initiierter Ansatz.

Beim objektinitiierten Ansatz werden die Positionsinformationen durch die eintreffenden Positionsaktualisierungen der mobilen Objekte wiederhergestellt. Das mobile Objekt sendet dazu – nachdem es erkannt hat, dass sein Agent nicht mehr erreichbar ist (spätestens nach *miu*) – periodisch eine Positionsaktualisierungsnachricht mit einer geeigneten weniger häufigen Aktualisierungsrate *rui* (engl. *recovery update interval*) an seinen Agenten. Wie wir in Abschnitt 7.2 zeigen werden, ist das Aufbauen der Indizes nicht wesentlich langsamer als das reguläre Bearbeiten von Aktualisierungsnachrichten. Im Falle eines objektinitiierten Ansatzes muss der Blatt-Lokations-Server das Zeitintervall *rui* abwarten, bis er entweder die Positionsinformationen für die mobilen Objekte in seinem Dienstgebiet erhalten hat oder annimmt, dass die Verbindung zu ihnen abgebrochen ist.

Beim aufwändigeren Server-initiierten Ansatz fragt der wiederhergestellte Blatt-Lokations-Server die aktuelle Positionsinformation aktiv bei den mobilen Objekten ab, indem er die von stabilem Speicher wiederhergestellten Registrierungsinformationen verwendet. Dieser Ansatz setzt voraus, dass die zu den mobilen Objekten gehörenden Endgeräte auf Positionsanfragen reagieren können und führt zeitweise zu einer hohen Belastung des Netzwerks. Wir schlagen daher einen kombinierten Ansatz vor, bei dem die Positionsinformationen hauptsächlich durch die

2. Diese Angabe ist objektspezifisch, da die vom LS verwalteten mobilen Objekte sehr unterschiedliche Genauigkeiten der Positionsinformationen und Bewegungseigenschaften aufweisen können.

Positionsaktualisierungen der mobilen Objekte erlangt werden, wie im objektinitiierten Ansatz. Nur die von einer Anwendung angefragten Positionsinformationen werden direkt bei den mobilen Objekten angefordert. Dies ist zur Zeit nur für Positionsanfragen möglich, bei denen der Server das zum mobilen Objekt gehörende Endgerät kontaktieren kann, das ihm über die wiederhergestellten Registrierungsinformationen bekannt ist. Um dies auch für Gebiets- und Nachbarschaftsanfragen zu ermöglichen, bei denen alle mobilen Objekte in dem entsprechenden Gebiet abgefragt werden müssen, wird ein Mechanismus wie GeoCast benötigt (siehe NAVAS & IMIELINSKI (1997) oder COSCHURBA, ROTHERMEL & DÜRR (2002)). GeoCast erlaubt es, eine Nachricht effizient an alle mobilen Endgeräte innerhalb eines geographischen Gebiets auszuliefern.

5.4 Algorithmen

In diesem Abschnitt werden die wichtigsten Algorithmen, die für die Realisierung des hierarchisch verteilten Lokationsdienstes notwendig sind, ausführlich beschrieben. Im Einzelnen werden die Algorithmen für die Registrierung, die Positionsaktualisierung und die Zuständigkeitsübergabe (engl. *handover*) sowie für die Anfragebearbeitung besprochen. Bei der Beschreibung der Algorithmen gehen wir von einem mobilen Objekt mit angeschlossenen lokalen Positionierungssensoren aus; eine Erweiterung der Algorithmen auf externe Sensorsysteme ist allerdings leicht möglich. In diesem Kapitel werden weiterhin nur die grundlegenden hierarchischen Algorithmen behandelt. Optimierungsmöglichkeiten, die sich durch das Zwischenspeichern von Informationen ergeben, werden kurz in Abschnitt 5.5 vorgestellt.

Um die Funktionalität des LS zu benutzen, kontaktiert ein Klient einen nahegelegenen Blatt-Server, der im Folgenden als *Kontakt-Server* (engl. *entry server*) bezeichnet wird. Wir setzen dabei einen geeigneten Mechanismus zum Auffinden eines solchen Servers voraus, der z. B. durch Jini (JINI (2001)) oder SLP (GUTTMAN ET AL. (1999)) erbracht werden kann. Falls es sich – was oft der Fall sein wird – bei dem Klienten ebenfalls um ein mobiles Gerät handelt, das über eine Möglichkeit zur Bestimmung seiner Position verfügt, sollte es sich bei dem Kontakt-Server um den Blatt-Server handeln, in dessen Dienstgebiet sich der Klient gegenwärtig befindet, um die Lokalität von Anfragen bestmöglich auszunutzen. Falls der Klient gleichzeitig als mobiles Objekt beim LS angemeldet ist, ist ein geeigneter Kontakt-Server schon durch seinen Agenten, der durch den Zuständigkeitsübergabemechanismus bestimmt wird, gegeben.

Der Kontakt-Server sorgt für eine Weiterleitung der Anfrage zu den betroffenen Blatt-Lokations-Servern. Er sammelt dazu deren Teilergebnisse und integriert diese, bevor er das Gesamtergebnis an den Klienten zurücksendet. Eine Alternative hierfür wäre gewesen, dass die Blatt-

Lokations-Server die Teilergebnisse direkt an den Klienten zurückschicken, wodurch ein Kommunikationsschritt eingespart wird (vgl. auch unsere Messergebnisse in Abschnitt 7.3). Allerdings handelt es sich bei des Klienten des LS in vielen Fällen um mobile Endgeräte, die oft eine geringere Rechenleistung und eine schlechtere Kommunikationsanbindung aufweisen als stationäre Rechner. Bei der hier gewählten Vorgehensweise wird ihnen die Aufgabe, die Teilergebnisse zu integrieren, durch den Kontakt-Server abgenommen.

Bei der folgenden Beschreibung der Algorithmen wurde zum Zweck einer einfacheren Darstellung auf die Betrachtung der Fehlerbehandlung verzichtet und u. a. angenommen, dass angefragte Objekte beim LS registriert und angefragte Gebiete vollständig in dessen Gesamtdienstgebiet enthalten sind.

5.4.1 Registrierung

Um ein mobiles Objekt beim LS zu registrieren, sendet die registrierende Instanz eine Registrierungsanfrage (engl. *registration request*) *registerReq* an einen beliebigen Blatt-Server (siehe Abschnitt 4.2 für eine Beschreibung der Parameter) und wartet auf die Antwort *registerRes*, welche die Adresse des für das mobile Objekt zuständigen Agenten enthält. Im Folgenden wird der Einfachheit halber angenommen, dass es sich bei der registrierenden Instanz um das mobile Objekt selbst handelt. Der bei der Registrierung ablaufende Prozess ist in Algorithmus 5-3 beschrieben.

Der Registrierungsprozess besteht aus zwei Phasen, wobei die erste Phase beginnt, wenn der kontaktierte Server *registerReq* empfängt. In dieser Phase wird der für das Objekt zukünftig zuständige Blatt-Server gesucht, dessen Dienstgebiet die im mitgelieferten Positionierungsdatensatz *s* übergebene aktuelle Position des mobilen Objekts enthält. Falls erforderlich wird dazu die Registrierungsanfrage in der Server-Hierarchie Richtung Wurzel weitergegeben, solange bis ein Server gefunden wird, dessen Dienstgebiet *s.pos* enthält. Von dort aus wird die Anfrage nach unten weitergeleitet, solange bis der entsprechende Blatt-Server erreicht ist.

Eine Registrierung ist erfolgreich, wenn der LS eine Genauigkeit aus dem geforderten Intervall [*desAcc*, *minAcc*] bieten kann. Wenn keine geforderte minimale Genauigkeit *minAcc* angegeben ist, kann die vom LS angebotene Genauigkeit *offeredAcc* direkt aus den an der Position *s.pos* verfügbaren Positionierungssensoren und der maximal erlaubten Genauigkeit bestimmt werden. Im anderen Fall muss der Blatt-Server entsprechend Algorithmus 5-1 die maximal in dem angegebenen Gebiet verfügbare Genauigkeit aus der Sensorkarte berechnen. Abhängig von der Größe des Gebiets müssen dazu auch benachbarte Server kontaktiert werden[3]. Falls die Registrierung nicht erfolgreich war, d. h. die verfügbare Genauigkeit kleiner ist als *minAcc*, wird di-

```
(1)  function calcMinAcc(sensorMap, regArea, oInfo)
(2)      minAcc := 0
(3)      foreach subArea, subArea.a ∩ regArea ≠ ∅ in sensorMap do
(4)          minAcc := max( minAcc, min( { sens.acc |  sens ∈
                 subArea.sensorList ∧ sens.sensorId ∈  oInfo.sensorList } ) )
(5)      if ( regArea ∩ c.sa ≠ regArea ) then
             contact neighboring location servers
(6)      return minAcc
(7)  endfunc
```

Algorithmus 5-1. Bestimmung der maximalen Genauigkeit, die für ein gegebenes Gebiet garantiert werden kann.

rekt eine *registerFailed* Nachricht an die registrierende Instanz zurückgesendet. Diese beinhaltet das größtmögliche Gebiet um *s.pos*, für das die geforderte Genauigkeit noch garantiert werden kann, und das entsprechend zu der minimalen Genauigkeit aus der Sensorkarte errechnet wird (siehe Algorithmus 5-2).

```
(1)  function calcMaxPossArea(sensorMap, regArea, minAcc, oInfo)
(2)      possArea := ∅
(3)      foreach subArea, subArea.a ∩ regArea ≠ ∅ in sensorMap do
(4)          if ( min( { sens.acc |  sens ∈ subArea.sensorList ∧ sens.sensorId ∈
                 oInfo.sensorList } ) < minAcc ) then
(5)              possArea := possArea ∪ subArea.a
(6)          endif
(7)      if ( regArea ∩ c.sa ≠ regArea ) then
             contact neighboring location servers
(8)      return possArea
(9)  endfunc
```

Algorithmus 5-2. Bestimmung des maximalen Gebiets, für das eine vorgegebene Genauigkeit garantiert werden kann.

3. Um die Anzahl der betroffenen Server zu beschränken, kann es hier sinnvoll sein, eine obere Grenze für die Größe des Registrierungsgebiets vorzugeben.

Nur im Falle einer erfolgreichen Registrierung wird die zweite Phase gestartet, die ausgehend von dem Blatt-Server die Verweise, aus denen der Suchpfad für das Objekt besteht, aufbaut. Zu diesem Zweck wird eine *createPath* Anfrage in der Hierarchie bis zum Wurzel-Server nach oben gesendet (siehe Algorithmus 5-4). Jeder Server, der diese Nachricht empfängt, trägt einen entsprechenden Verweis in seine *visitorDB* ein. Der Blatt-Server speichert schließlich die Registrierungsinformationen für das mobile Objekt in seiner *visitorDB* und den mitgelieferten Positionierungsdatensatz *s* in seiner *sightingDB*. Er sendet dann eine *registerRes* Nachricht an die registrierende Instanz, um die Registrierung zu bestätigen. Diese Nachricht beinhaltet die Genauigkeit mit der die Positionsinformationen für das mobile Objekt vom LS angeboten werden (*offeredAcc*).

5.4.2 Positionsaktualisierung und Zuständigkeitsübergabe

Da Positionsaktualisierungen sehr häufig durchgeführt werden müssen, ist eine effiziente Bearbeitung von größter Bedeutung. Aus diesem Grund wurden in verschiedenen Forschungsbereichen Protokolle entwickelt, mit dem Ziel den Vorgang zur Positionsaktualisierung zu optimieren (für Mobilfunknetze siehe z. B. BAR-NOY, KESSLER & SIDI (1995) oder BHATTACHARYA & DAS (1999); für Mobile-Objekte-Datenbanken siehe WOLFSON ET AL. (1999)). Da sich dieses Kapitel thematisch mit Architekturfragen beschäftigt, wird hier ein einfacheres entfernungsbasiertes berichtendes Aktualisierungsprotokoll vorausgesetzt. Dieses und weitere Protokolle zur Aktualisierung von Positionsinformationen werden ausführlich in Kapitel 6 betrachtet.

Zur Positionsaktualisierung vergleicht ein beim LS angemeldetes mobiles Objekt seine aktuelle Position - wie sie von dem von ihm verwendeten Sensorsystem geliefert wird - ständig mit der Position, die es zuletzt an seinen Agenten übermittelt hat. Wenn die Entfernung zwischen diesen beiden Positionen größer ist als die vom LS angebotene Genauigkeit, sendet das mobile Objekt seinem Agenten eine *updateReq*-Nachricht, die den aktuell gültigen Positionierungsdatensatz *s* enthält. Der Ablauf bei der Bearbeitung einer solchen Positionsaktualisierung ist in Algorithmus 5-5 gezeigt.

Ein Agent, der eine Positionsaktualisierungsnachricht mit *s* empfängt, überprüft, ob *s.pos* noch in seinem Zuständigkeitsgebiet liegt. In diesem Fall aktualisiert er nur den entsprechenden Eintrag in seiner *sightingDB*. Im anderen Fall stößt er eine Übergabe der Zuständigkeit an, indem er eine entsprechende Anforderung, *handoverReq*, an den übergeordneten Lokations-Server sendet. Die entsprechende Nachricht enthält neben *s* auch die Registrierungsinformationen für das Objekt. Eine *handoverReq*-Nachricht wird Richtung Wurzel propagiert, solange bis die Position *s.pos* innerhalb des Zuständigkeitsgebiets des Empfängers liegt. Danach wird die Nach-

```
[ upon receiving registration request registerReq(s, regInst, oInfo, acl, desAcc,
minAcc, regArea, timeout) ]
(1)  if ( mobile object is in service area of server: s.pos ∈ c.sa ) then
(2)      if ( server is leaf server: c.children = ∅ ) then
             // determine maximal accuracy with which the location information
             // can be managed by the service inside regArea
(3)          acc := call calcMinAcc(c.sensorMap, regArea, oInfo);
(4)          if ( acc ≤minAcc ) then
                 // create visitor record
(5)              reg := (regInst, oInfo, acl, desAcc, minAcc, regArea, timeout);
(6)              insert into visitorDB values (oId = s.oId,
                     offeredAcc = max(acc, desAcc), regInfo = reg);
(7)              insert into sightingDB values s;
(8)              send registerRes(self, max(acc, desAcc))
                     to registering instance regInst
                 // registration successful
(9)              send createPath(s.oId) to parent server c.parent;
(10)         else
                 // registration not successful: determine maximum sub area of
                 // regArea for which minAcc can be guaranteed
(11)             possArea := call calcMaxPossArea(
                     c.sensorMap, minAcc, regArea, oInfo);
(12)             send registerFailed(self, possArea)
                     to registering instance regInst
(13)         endif
(14)     else
             // forward registration downwards
(15)         child := select child ∈ c.children with s.pos ∈ child.c.sa;
(16)         send registerReq(s, regInst, oInfo, acl, desAcc, minAcc, regArea,
                 timeout) to child
(17)     endif
(18) else
         // forward registration upwards
(19)     send registerReq(s, regInst, oInfo, acl, desAcc, minAcc, regArea,
             timeout) to parent server c.parent
(20) endif
```

Algorithmus 5-3. Ablauf des Registrierungsprozesses.

richt abwärts propagiert, bis sie einen Blattknoten erreicht, in dessen Zuständigkeitsgebiet *s.pos* enthalten ist. Dieser Lokations-Server wird der neue Agent für das Objekt, erzeugt einen entsprechenden Besuchereintrag in *visitorDB* und trägt *s* in *sightingDB* ein. Er sendet dann eine

```
[ upon receiving registration entry request createPath(oId) from server ls_f ]
(1)  insert into visitorDB values (oId = oId, forwardRef = ls_f)
(2)  if ( server is not root server: c.parent ≠ ε ) then
         // forward create path request upwards
(3)      send createPath(oId) to parent server c.parent
```

Algorithmus 5-4. Aufbauen des Suchpfades.

```
[ upon receiving position update update(s) from tracked object o ]
(1)  if ( position of sighting is outside of service area: s.pos ∉ c.sa ) then
         // initiate handover
(2)      send handoverReq(s, visitorDB(s.oId).reginfo)
             to parent server c.parent
(3)      receive handoverAck
         // remove visitor and sighting record from database
(5)      delete from visitorDB where oId = s.oId
(6)      delete from sightingDB where oId = s.oId
(7)  else
(8)      update sightingDB set s where oId = s.oId
(9)  endif
```

Algorithmus 5-5. Bearbeitung einer Positionsaktualisierungsnachricht.

entsprechende Benachrichtigung, *handoverRes*, an das mobile Objekt, die seine Adresse und die neue vom Lokationsdienst angebotene Genauigkeit enthält, um diesem die erfolgte Zuständigkeitsübergabe mitzuteilen.

Eine *handoverAck*-Nachricht wird nach erfolgter Zuständigkeitsübergabe schließlich entlang des Pfades zurückpropagiert, den die Anfrage genommen hat, um den Suchpfad für das mobile Objekt entsprechend umzusetzen. Nicht-Blatt-Server auf diesem Pfad, die diese Nachricht empfangen, aktualisieren ihre *visitorDBs* entsprechend. Das heißt, sie bauen den Teil des Weiterleitungspfads ab, der zu dem alten Agenten des Objekts führt und bauen einen neuen Teilpfad zu dessen neuen Agenten auf.

Da der LS hierarchisch organisiert ist und Bewegungen lokal sind, betreffen viele Zuständigkeitsübergaben nur einen Nicht-Blatt-Server. Die Wahrscheinlichkeit, dass ein bestimmter Nicht-Blatt-Server von einer Zuständigkeitsübergabe betroffen ist, nimmt abhängig vom Verzweigungsgrad der Server-Hierarchie von Blattknoten Richtung Wurzelknoten immer weiter ab.

```
[ upon receiving handover request handoverReq(s, regInfo) from server ls_f ]
(1)  if ( mobile object is in service area of server: s.pos ∈ c.sa ) then
(2)      if ( server is leaf server: c.children = ∅ ) then
             // insert object into local database
(3)          acc := determine maximum accuracy with which the location
                    information can be managed by the service
(4)          insert into visitorDB values (oId = s.oId,
                 offeredAcc = max(acc, regInfo.desAcc), regInfo = regInfo)
(5)          insert into sightingDB values s
(6)          send handoverRes(self, max(acc, regInfo.desAcc))
                 to tracked object o
(7)          send handoverAck to ls_f
(8)      else // server is not leaf server
             // forward handover downwards; create/reset forwarding reference
(9)          child := select child ∈ c.children with s.pos ∈ child.c.sa
(10)         send handoverReq(s, regInfo) to child
(11)         receive handoverAck
(12)         if ( visitorDB(s.oId) = ε ) then
(13)             insert into visitorDB values
                     (oId = s.oId, forwardRef = child)
(14)         send handoverAcc to ls_f
(15)     endif
(16) else
         // forward handover request upwards
(17)     send handoverReq(s, regInfo) to parent server c.parent
(18)     receive handoverAck
         // remove forwarding pointer
(19)     delete from visitorDB where oId = s.oId
(20)     send handoverAcc to ls_f
(21) endif
```

Algorithmus 5-6. Bearbeitung einer Zuständigkeitsübergabe.

Algorithmus 5-6 beschreibt die Bearbeitung einer Zuständigkeitsübergabeaufforderung, beim Empfangen einer entsprechenden Nachricht *handoverReq*.

5.4.3 Positionsanfrage

Ein Klient des LS startet eine Positionsanfrage bezüglich eines bestimmten mobilen Objekts *o*, indem er eine *posQueryReq* Nachricht an seinen Kontakt-Server sendet (siehe Algorithmus 5-7). Wenn der Kontakt-Server selbst den Besuchereintrag für *o* verwaltet, beantwortet er die An-

frage direkt, indem er auf seine *sightingDB* zugreift und dort die Position von *o* abfragt. Im anderen Fall, gibt er die Anfrage nach oben weiter und wartet auf die Antwort. Wenn er diese erhalten hat, liefert er den entsprechenden Lokationsbezeichner in einer *posQueryRes*-Nachricht an den anfragenden Klienten zurück.

```
(1)  procedure rtrvPosition(var ld, oId)
       // Retrieve position and offered accuracy for queried object from DBs
(2)    ld := select new ld(s.pos, v.offeredAcc) from sightingDB s, visitorDB v
            where s.oId = oId and v.oId = oId;
(3)  endproc
```

```
[ upon receiving position query posQueryReq(oId) from client o ]
(1)  if ( queried object is managed by server: visitorDB(oId) ≠ ε ) then
       // retrieve local position information
(2)    call rtrvPosition(ld, oId)
(3)  elseif ( server is not root server: c.parent ≠ ε ) then
       // forward query upwards
(4)    send posQueryFwd(oId, self) to parent server c.parent
(5)    receive position query results, posQueryRes(ld)
(6)  endif
(7)  send posQueryRes(ld) to client o
```

```
[ upon receiving position query forward posQueryFwd(oId, ls_e) ]
(1)  if ( queried object is managed by server:
           visitorDB(oId) ≠ ε ∧ c.children = ∅ ) then
       // retrieve local position information
(2)    call rtrvPosition(ld, oId)
(3)  elseif ( server has forwarding pointer: visitorDB(oId) ≠ ε ) then
       // forward packet downwards
(4)    send posQueryFwd(oId, ls_e) to child server visitorDB(oId).forwardRef
(5)  endif
(6)  send posQueryFwd(oId, ls_e) to parent server c.parent
```

Algorithmus 5-7. Ausführen und Weiterleiten einer Positionsanfrage.

Ein Nicht-Blatt-Server der eine weitergeleitete Positionsanfrage, *posQueryFwd*, empfängt, leitet die Anfrage solange an seinen übergeordneten Server weiter, solange er nicht selbst einen Besuchereintrag für *o* speichert. Andernfalls sendet er die Anfrage an den untergeordneten Server, der durch den Verweis in dem zu *o* gehörenden Besuchereintrag spezifiziert wird. Wenn ein Blattknoten eine weitergeleitete Positionsanfrage von seinem übergeordneten Server erhält, bestimmt er die Position von *o* aus seiner *sightingDB* und sendet den entsprechenden Lokations-

bezeichner direkt an den Kontakt-Server des Klienten, dessen Adresse in der weitergeleiteten Anfrage enthalten ist.

```
(1)  procedure rtrvObjectsInArea(var objects, area, reqAcc, reqOverlap)
       // retrieve objects in area out of own service area from DB
(2)    objects := select (oid, new ld(s.pos, v.offeredAcc))
                  from sightingDB s, visitorDB v
                  where Overlap(area, CIRCLE(s.pos, v.offeredAcc))
                    ≥ reqOverlap and v.offeredAcc < reqAcc
(3)  endproc
```

```
[ upon receiving range query rangeQueryReq(area, reqAcc, reqOverlap) from
client o ]
(1)  objects := ∅  // for collecting the results
(2)  covered := ∅  // for checking if all results have been received
(3)  if ( area overlaps with service area of the server:
       ENLARGE(area, reqAcc) ∩ c.sa ≠ ∅ ) then
       // retrieve local objects in area
(4)    call rtrvObjectsInArea(objects, area, reqAcc, reqOverlap)
(5)    covered := area ∩ c.sa
(6)  endif
(7)  if ( part of area lies outside service area:
         ENLARGE(area, reqAcc) −c.sa ≠ ∅ ) then
       // forward query upwards
(8)    send rangeQueryFwd(area, reqAcc, reqOverlap, self)
         to parent server c.parent
(9)    until area is entirely covered: covered = area do
(10)     receive range query results, rangeQuerySubRes(objs, a)
(11)     objects := objects ∪ objs; covered := covered ∪ a
(12)   end
(13) endif
(14) send rangeQueryRes(objects) to client o
```

Algorithmus 5-8. Ausführen einer Gebietsanfrage.

Insgesamt wird die Anfragenbearbeitung immer bei einem Blatt-Server begonnen, der wenn notwendig die Anfrage Richtung Wurzelknoten bis zu dem ersten Server, der einen entsprechenden Besuchereintrag speichert, weiterleitet. Alternativ dazu könnten Anfragen immer direkt an den (replizierten) Wurzel-Server der Hierarchie weitergeleitet werden. Wir haben uns jedoch gegen diese Alternative entschieden, da wir annehmen, dass die Klienten mehr an mobilen Objekten in ihrer nahen Umgebung interessiert sind als an weiter entfernten Objekten (sie-

he dazu die Diskussion zur Lokalität von Anfragen in Abschnitt 5.2). Die Entfernung zu dem Lokations-Server, der die benötigten Informationen speichert, ist in diesem Fall ausgehend von einem Blatt-Server im Durchschnitt kürzer ist als von der Wurzel aus. Zudem reduziert sich damit auch die Belastung der Wurzel, die sonst bei jeder Anfrage kontaktiert werden muss und schnell zu einem Flaschenhals des Systems würde.

5.4.4 Gebietsanfrage

Eine Gebietsanfrage wird gestartet, wenn ein Klient die Nachricht *rangeQueryReq* an seinen Kontakt-Server sendet (siehe Algorithmus 5-8). Im Gegensatz zu einer Positionsanfrage können mehrere Blatt-Server von einer solchen Anfrage betroffen sein, wenn das angefragte Gebiet über die Grenzen eines Dienstgebiets hinausragt. Wenn mehrere Server betroffen sind, senden diese ihre Teilergebnisse an den Kontakt-Server, der die Ergebnisse in der Variable *objects* sammelt. Eine *rangeQuerySubRes*-Nachricht, in der ein solches Teilergebnis zurückgeliefert wird, enthält, neben einer Beschreibung des Teils des angefragten Gebiets für das diese Ergebnisse gelten, ein Paar aus Objekt- und Lokationsbezeichner für jedes mobile Objekt innerhalb des Gebiets. Die Variable *covered* wird benutzt, um zu überprüfen, ob schon alle Teilergebnisse empfangen wurden. Nachdem der Kontakt-Server alle Teilergebnisse empfangen hat, sendet er sie in einer *rangeQueryRes*-Nachricht an den anfragenden Klienten zurück.

Wenn das Dienstgebiet des Kontakt-Servers das angefragte Gebiet nicht vollständig enthält, wird die Gebietsanfrage Richtung Wurzel propagiert, solange, bis ein Server gefunden wird, der das Gebiet vollständig überdeckt (siehe Algorithmus 5-9). Von diesem ausgehend wird die Anfrage zu allen Blatt-Servern weitergeleitet, die in den entsprechenden Unterhierarchien liegen und deren Dienstgebiete sich mit dem angefragten Gebiet überschneiden. Jeder dieser Blatt-Server fragt dann die entsprechenden mobilen Objekte unter Benutzung des mehrdimensionalen Indexes aus seiner *sightingDB* ab. Die Ergebnisse werden schließlich wiederum direkt an den Kontakt-Server zurückgeschickt.

Vor dem Vergleichen des angefragten Gebiets mit dem Dienstgebiet eines Lokations-Servers wird ersteres, durch Verwendung der Funktion *ENLARGE,* gleichmäßig an allen Seiten um den Betrag von *reqAcc* vergrößert. Andernfalls könnte es vorkommen, dass ein Server, der ebenfalls mögliche Kandidaten für die Ergebnismenge enthält, nicht befragt wird, da auch Objekte mit einer Position außerhalb des angefragten Gebiets aufgrund der Überlappung mit ihrem Aufenthaltsbereich zu der Ergebnismenge gehören können (siehe Abschnitt 4.3).

```
[ upon receiving range query forward rangeQueryFwd(area, reqAcc, reqOverlap,
ls_e) from forwarding server ls_f ]
(1)  if ( area overlaps with service area of the server:
           ENLARGE(area, reqAcc) ∩ c.sa ≠ ∅ ) then
(2)      if ( server is leaf server: c.children = ∅ ) then
           // retrieve local objects in area
(3)        call rtrvObjectsInArea(objects, area, reqAcc, reqOverlap)
(4)        send rangeQuerySubRes(objects, area ∩ c.sa) to entry server ls_e
(5)      else   // forward query downwards
(6)        foreach child in c.children do
(7)          if ( area overlaps with child service area:
                   ENLARGE(area, reqAcc) ∩ child.c.sa ≠ ∅ and
                   child was not forwarding server: child ≠ ls_f ) then
(8)            send rangeQueryFwd(area, reqAcc, reqOverlap, ls_e) to child
(9)      endif
(10) endif
(11) if ( area lies partially outside service area and
           message was not forwarded from parent:
           ENLARGE(area, reqAcc) –c.sa ≠ ∅ and c.parent ≠ ls_f ) then
         // forward query upwards
(12)     send rangeQueryFwd(area, reqAcc, reqOverlap, ls_e)
           to parent server c.parent
(13) endif
```

Algorithmus 5-9. Weiterleiten einer Gebietsanfrage.

Offensichtlich hängen die Kosten für die Bearbeitung einer Gebietsanfrage von der Anzahl der Blattknoten ab, die von ihr betroffen sind. Um diese Kosten zu beschränken, ist es denkbar, die Größe des Gebiets, das von einer Anwendung abgefragt werden kann, zu beschränken.

5.4.5 Nachbarschaftsanfrage

In diesem Abschnitt wird zuerst ein grundlegender Algorithmus für die Durchführung einer Nachbarschaftsanfrage besprochen, die zu einem vorgegebenen geographischen Ort das nächstgelegene mobile Objekt zurückliefert (für eine Beschreibung der Semantik siehe Abschnitt 4.3), dann wird kurz auf Optimierungsmöglichkeiten eingegangen.

Im Gegensatz zu Positions- und Gebietsanfragen ist bei einer Nachbarschaftsanfrage nicht von vorne herein klar, welches konkrete Objekt bzw. Gebiet davon betroffen ist. Eine entsprechende Anfrage muss daher in zwei Phasen durchgeführt werden.

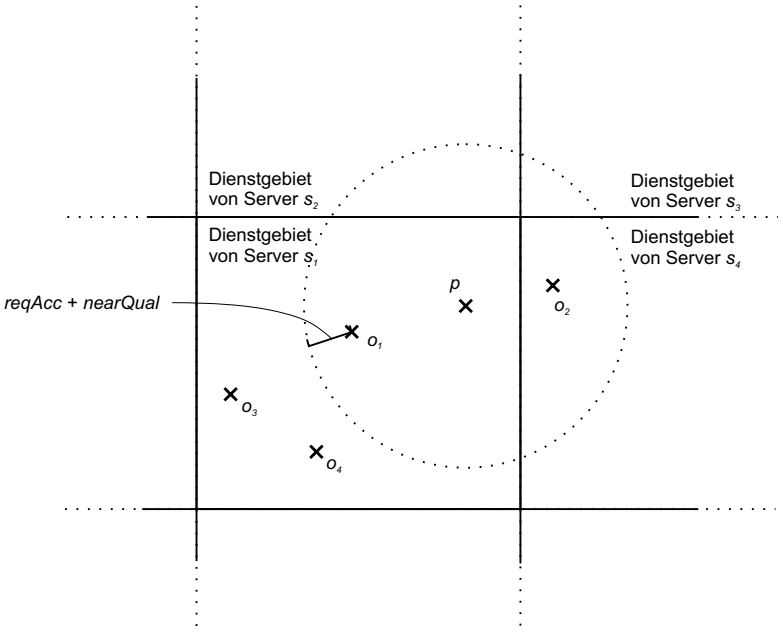


Abbildung 5-6. Beispielszenario für eine verteilte Nachbarschaftsanfrage.

```
(1)  procedure rtrvNeighbor(var nearObj, var nearObjSet, p, reqAcc, nearQual)
       // retrieve nearest object in own service area from DB
(2)    nearObj := select (s.oId, new ld(s.pos, v.offeredAcc))
         from sightingDB s, visitorDB v
         where min(DISTANCE(p, s.pos)) and v.offeredAcc < reqAcc
           and v.oId = s.oId;
       // retrieve objects in nearest objects set for own service area
(3)    nearObjSet := select (s.oId, new ld(s.pos, v.offeredAcc))
         from sightingDB s, visitorDB v
         where DISTANCE(p, s.pos) < DISTANCE(p, nearObj.pos) +
           nearQual and v.offeredAcc < reqAcc and v.oId = s.oId;
(4)  endproc
```

Algorithmus 5-10. Bestimmen der nächsten Nachbarn aus DBs für eine Nachbarschaftsanfrage.

In der ersten Phase einer Nachbarschaftsanfrage wird der Lokations-Server angefragt, der die gesuchte Position *p* enthält. In dem in Abbildung 5-6 gezeigten Beispiel wäre dies s_1. Dieser ermittelt nun über den mehrdimensionalen Index das mobile Objekt aus seiner *sightingDB*, das am nächsten zu *p* ist, in unserem Beispiel o_1 (angenommen, dass s_1 mindestens ein mobiles Ob-

jekt verwaltet). Allerdings muß o_1 nicht das nächste Objekt zu p sein. Wenn die gesuchte Position zum Beispiel nahe an einer Dienstgebietsgrenze liegt, können Objekte der Ergebnismenge in einem benachbarten Server gespeichert sein. In unserem Beispiel wäre eigentlich o_2 das zu p nächste Objekt.

```
[ upon receiving nearest neighbor query neighborQueryReq(p, reqAcc,
nearQual) from client o ]
(1)  waitingForResults := false     // flag
(2)  if ( position lies in service area of server: p ∈ c.sa ) then
         // retrieve qualifying objects from DB
(3)      call rtrvNeighbors(nearObj, nearObjSet, p, reqAcc, nearQual)
         // Determine maximum radius for objects that may qualify for query
(4)      r := DISTANCE(p, nearObj.pos) + reqAcc + nearQual
(5)      if ( other servers may contain qualifying objects:
                 CIRCLE(p, r) - c.sa ≠ ∅ ) then
(6)          objects := nearObj ∪ nearObjSet; covered := c.sa
             // initiate second phase
(7)          send neighborQueryPhase2(p, reqAcc, nearQual, CIRCLE(p, r),
                 c.self) to parent server c.parent
(8)          waitingForResults := true
(9)      endif
(10) else
(11)     objects := ∅; covered := ∅
(12)     send neighborQueryFwd(p, reqAcc, nearQual, c.self)
             to parent server c.parent
(13)     waitingForResults := true
(14) endif
(15) if ( waitingForResults ) then
(16)     repeat
(17)         receive neighbor query results, neighborQuerySubRes(objs, a_s, a_sa)
(18)         searched := a_s     // identical for every answer
(19)         objects := objects ∪ objs
(20)         covered := covered ∪ a_sa
(21)     until area is entirely covered: searched - covered = ∅
         select nearObj and nearObjSet from objects
(22) endif
(23) send neighborQueryRes(nearObj, nearObjSet) to client o
```

Algorithmus 5-11. Ausführen einer Nachbarschaftsanfrage.

Aus der Entfernung von o_1 zu p erhält s_1 allerdings durch Aufaddieren von *reqAcc* (um die Ungenauigkeit der Positionsinformationen miteinzubeziehen) und *nearQual* (um die Objekte der

Menge *nearObjSet* zu berücksichtigen) den Radius für das kreisförmige Gebiet um *p*, in dem sich alle Objekte der Ergebnismenge befinden müssen. In einer zweiten Phase werden dann alle Blatt-Server befragt, deren Dienstgebiete sich mit diesem Gebiet überschneiden. Wie s_1 schicken sie ihre Ergebnisse direkt an den Kontakt-Server zurück, der aus diesen das zu *p* nächste Objekt und die Objekte aus der *nearObjSet*-Menge ermittelt. Ein entsprechendes Vorgehen wird meist auch für Nachbarschaftsanfragen in baumartigen mehrdimensionalen Indexstrukturen verwendet (siehe z. B. FRIEDMANN, BENTLEY & FINKEL (1977) für kd-Bäume und ROUSSOPOULOS, KELLEY & VINCENT (1995) für R-Bäume).

```
[ upon receiving neighbor query forward neighborQueryFwd(p, reqAcc,
nearQual, ls_e) ]
(1)  if ( position lies in service area of server: p ∈ c.sa ) then
(2)      if ( server is leaf server: c.children = ∅ ) then
             // retrieve qualifying objects from DB
(3)          call rtrvNeighbors(nearObj, nearObjSet, p, reqAcc, nearQual)
             // Determine maximum radius for objects that may qualify for query
(4)          r := DISTANCE(p, nearObj.pos) + reqAcc + nearQual
(5)          objects := nearObj ∪ nearObjSet
(6)          if ( other servers may contain qualifying objects:
                     CIRCLE(p, r) - c.sa ≠ ∅ ) then
                 // initiate second phase
(7)              send neighborQueryPhase2(p, reqAcc, nearQual,
                     CIRCLE(p, r), ls_e) to parent server c.parent
(8)          endif
(9)          send neighborQuerySubRes(objects, CIRCLE(p, r), c.sa)
(10)     else // forward query downwards
(11)         child := select child ∈ c.children with s.pos ∈ child.c.sa
(12)         send neighborQueryFwd(p, reqAcc, nearQual, ls_e) to child
(13)     endif
(14) else // forward query upwards
(15)     send neighborQueryFwd(p, reqAcc, nearQual, ls_e)
             to parent server c.parent
(16) endif
```

Algorithmus 5-12. Weiterleitung einer Nachbarschaftsanfrage.

Wie bei Positions- und Gebietsanfragen geht wegen der angenommenen Lokalität der Anfragen (eine häufige Anfrage wird die nach dem nächsten mobilen Objekt zu der eigenen Position sein) eine Nachbarschaftsanfrage von einem nahegelegenen Kontakt-Server aus, an den ein Klient eine *neighborQueryReq*-Nachricht sendet (siehe Algorithmus 5-11). Damit beginnt die erste

Phase der Nachbarschaftsanfrage. Falls der Kontakt-Server die Anfrage nicht lokal beantworten kann, sendet er eine *neighborQueryFwd*-Nachricht an den ihm übergeordneten Server. Diese wird durch die Lokations-Server-Hierarchie bis zu dem Blatt-Server propagiert, dessen Dienstgebiet die Position *p* enthält (siehe Algorithmus 5-12). Der Blatt-Server ermittelt die Ergebnismenge aus seiner *sightingDB* (wie in Algorithmus 5-10 gezeigt) und sendet sie als Liste von (Objekt-, Lokationsbezeichner)-Paaren in einer *neighborQuerySubRes*-Nachricht an den Kontakt-Server zurück, dessen Adresse ihm aus der Weiterleitungsnachricht bekannt ist.

```
[ upon receiving second phase of neighbor query neighborQueryPhase2(p,
reqAcc, nearQual, a_s, ls_e) from forwarding server ls_f ]
(1)  if ( area to search overlaps with service area of the server:
           a_s ∩ c.sa ≠ ∅ ) then
(2)     if ( server is leaf server: c.children = ∅ ) then
           // retrieve qualifying objects from DB
(3)        call rtrvNeighbors(nearObj, nearObjSet, p, reqAcc, nearQual)
(4)        objects := nearObj ∪ nearObjSet
(5)        send neighborQuerySubRes(objects, a_s, c.sa) to entry server ls_e
(6)     else // forward query downwards
(7)        foreach child in c.children do
(8)           if ( area overlaps with child service area: a_s ∩ child.c.sa ≠ ∅ and
                    child was not forwarding server: child ≠ ls_f ) then
(9)              send neighborQueryPhase2(p, reqAcc, nearQual, a_s, ls_e)
                    to child
(10)    endif
(11) endif
(12) if ( part of area lies outside service area: a_s –c.sa ≠ ∅ and
           parent was not forwarding server: c.parent ≠ ls_f ) then
        // forward query upwards
(13)    send neighborQueryPhase2(p, reqAcc, nearQual, a_s, ls_e)
           to parent server c.parent
(14) endif
```

Algorithmus 5-13. Zweite Phase einer Nachbarschaftsanfrage.

Falls notwendig (siehe oben) startet der in der ersten Phase bestimmte Blatt-Server eine zweite Phase, indem er eine *neighborQueryPhase2*-Nachricht mit dem zu durchsuchenden Gebiet an den übergeordneten Server sendet, die wie eine Gebietsanfrageweiterleitung behandelt wird (siehe Algorithmus 5-13). In diesem Fall enthält seine Teilantwort an den Kontakt-Server auch das in der zweiten Phase zu durchsuchende Gebiet und das Teilgebiet, das er bereits abgedeckt hat. Blatt-Server, die eine solche *neighborQueryPhase2*-Nachricht erhalten, bestimmen die in

dem zu durchsuchenden Gebiet liegenden nächsten Objekte zu *p* und senden ebenfalls eine *neighborQuerySubRes*-Nachricht an den Kontakt-Server, in der wiederum auch das Gebiet angegeben ist, aus dem diese Ergebnisse stammen.

Währenddessen wartet der Kontakt-Server auf die einzelnen Teilergebnisse und sammelt sie in der Variable *objects*. Aus dem ersten empfangenen Teilergebnis erhält er das Gebiet a_s, das bei der Bearbeitung der Anfrage insgesamt durchsucht wird, und speichert es in der Variable *searched*. Die einzelnen Teilergebnisse enthalten darüber hinaus das Teilgebiet aus dem jeweils die darin enthaltenen Objektinformationen stammen. Der Kontakt-Server merkt sich in der Variable *covered*, welches Gebiet bereits durch die erhaltenen Teilergebnisse abgedeckt ist, indem er die Vereinigung von diesen bildet. Ist keine zweite Phase notwendig, so ist das zu durchsuchende Gebiet bereits durch das erste Teilergebnis abgedeckt. Andernfalls muss er solange auf weitere Teilergebnisse warten, bis das insgesamt abgedeckte Gebiet das zu durchsuchende Gebiet vollständig umfasst. Ist dies der Fall, bestimmt er aus *objects* das nächste Objekt zu *p*, *nearestObj* sowie *nearObjSet* und sendet diese in einer *neighborQueryRes*-Nachricht als Liste von (Objektbezeichner, Lokationsbezeichner)-Paaren an den anfragenden Klienten zurück.

Wie bei Gebietsanfragen hängen die Kosten für eine Nachbarschaftsanfrage unter anderem von der Anzahl der betroffenen Server und damit der Größe des zu durchsuchenden Gebiets ab. Auch hier kann es sinnvoll sein, die maximale Entfernung, die ein gesuchtes Objekt von der angefragten Position haben darf, und damit die Größe des zu durchsuchenden Gebiets zu beschränken.

Der oben beschriebene Algorithmus funktioniert gut für lokale Nachbarschaftsanfragen, bei denen nach dem nächsten mobilen Objekt zur aktuellen Position des Anfragenden, wie z. B. dem nächsten Taxi, gesucht wird. Es kann davon ausgegangen werden, dass ein großer Anteil der Nachbarschaftsanfragen aus solchen Anfragen bestehen wird. In diesem Fall kann die gesamte Anfrage oder zumindest die erste Phase meist komplett auf dem Kontakt-Server selbst durchgeführt werden. Für nicht-lokale Nachbarschaftsanfragen, die ein größeres Gebiet betreffen oder bei denen die gesuchte Position nahe einer Dienstgebietsgrenze liegt, ergibt sich allerdings eine höhere Antwortzeit durch die erforderliche zweite Phase.

Um bei nicht lokalen Nachbarschaftsanfragen die zweite Phase zu vermeiden, kann alternativ versucht werden, bereits während der ersten Phase eine Abschätzung für das Gebiet zu treffen, das alle mobilen Objekte der Ergebnismenge enthält. Es würden dann schon in dieser Phase alle Blatt-Server angefragt werden, deren Zuständigkeitsgebiet sich mit diesem Gebiet überlappt. Bei geschickter Wahl des Gebiets könnte die zweite Phase entfallen und daher eine bessere Antwortzeit erzielt werden. Jeder Nicht-Blatt-Server des Lokationsdienstes muss zu diesem Zweck

für jeden seiner untergeordneten Server zusätzlich eine Abschätzung der Dichte der mobilen Objekte in dessen Dienstgebiet *c.children.dens* haben.

Eine einfache Realisierung dieser Dichteabschätzung besteht darin, die Anzahl der Suchverweise, die auf den entsprechenden Kind-Server verweisen, durch den Flächeninhalt des Dienstgebiets dieses Kind-Servers zu teilen[4]. Mit Hilfe der Dichtefunktion lässt sich das Gebiet abschätzen, das durchschnittlich durchsucht werden muss, um das gesuchte Objekt zu finden. Wenn eine Gleichverteilung der mobilen Objekte im Dienstgebiet des jeweiligen Kind-Servers angenommen wird, ergibt sich dieses Gebiet zu einem Kreis um die angefragte Position mit einem Radius, welcher der durchschnittlichen Entfernung zum nächsten mobilen Objekt entspricht (unter Voraussetzung eines quadratischen Dienstgebiets und einer Verteilung der mobilen Objekte in einem gleichförmigen Gitter ist der Radius z. B. $1 / (\sqrt{2 \cdot dens})$ groß).

Überschneidet sich das durch die Dichteabschätzung gewonnene Gebiet bei der Abwärtspropagierung der Nachbarschaftsanfrage mit den Grenzen des Dienstgebiets des Servers, an den die Anfrage regulär weitergeleitet wird (d. h. der die angefragte Position enthält), so wird die Anfrage auch an die betroffenen Nachbarn weitergeleitet. Der Kontakt-Server für die Anfrage wählt schließlich das nächstgelegene Objekt aus und sendet es an den anfragenden Klienten zurück.

Soll eine genauere Dichteabschätzung verwendet werden, so kann das Dienstgebiet eines Lokations-Servers durch ein Raster mit einer vorgegebenen Granularität weiter unterteilt werden. Der ihm übergeordnete Server speichert für die Dichtefunktion dann ein Feld, dass zu jedem der so entstandenen Quadrate die Dichte an mobilen Objekten angibt. Änderungen dieser Dichte müssen dem übergeordneten Server regelmäßig gemeldet werden. Bei der Wahl der Granularität des Rasters und der Häufigkeit der Aktualisierungen muss zwischen dem dadurch entstehenden Aufwand und den zu erwartenden Optimierungsmöglichkeiten abhängig von der konkreten Einsatzumgebung des LS abgewägt werden.

5.5 Optimierungsmöglichkeiten

Bei der bisherigen Beschreibung der Algorithmen wurde aus Gründen der Übersichtlichkeit nicht auf mögliche Optimierungen durch das Zwischenspeichern (engl. *caching*) von Informationen eingegangen. Kann eine Anfrage nicht direkt von dem Lokations-Server beantwortet

4. Wesentlich aufwändiger ist eine solche Abschätzung jedoch, wenn auch spezielle Eigenschaften für das gesuchte Objekt gefordert werden. Die Abschätzung muss dann auch eine Komponente für die Selektivität der Anforderungen an die Eigenschaften des Objekts beinhalten.

werden, an den sie gestellt wurde (dem Kontakt-Server), so führt dies bisher dazu, dass die Hierarchie des LS traversiert wird, was sowohl den Durchsatz als auch die Antwortzeit des LS verringert (siehe Kapitel 7). In Verteilten Systemen werden in diesem Zusammenhang sehr oft Mechanismen zum Zwischenspeichern von häufig angefragten Informationen verwendet, um eine solche Traversierung zu vermeiden oder zu verkürzen. Die Skalierbarkeit und Leistungsfähigkeit des DNS (siehe MOCKAPETRIS & DUNLAP (1988)), zum Beispiel, beruht zu einem großen Teil auf entsprechenden Mechanismen.

Ein Zwischenspeichern von den für die Anfragebearbeitung benötigten Informationen kann im Wesentlichen auf den Klienten und den Blatt-Servern durchgeführt werden, da diese bei Anfragen als Kontakt-Server fungieren. Das Zwischenspeichern auf Blatt-Servern funktioniert entsprechend zu dem auf den Klienten selbst, ist dabei aber wirkungsvoller, da die gespeicherten Informationen für mehrere Klienten gleichzeitig genutzt werden können. Es soll daher im Folgenden nur das Zwischenspeichern von Informationen auf Blatt-Servern des LS betrachtet werden. Folgende Zuordnungsinformationen können dort auf flüchtigem oder stabilem Speicher zwischengespeichert werden.

(Blatt-Server, Dienstgebiet): Die Effizienz von Zuständigkeitsübergaben, Gebiets- und Nachbarschaftsanfragen kann erhöht werden, indem zu jedem lokal bekannten Blatt-Server dessen Dienstgebiet zwischengespeichert wird. Um diese Information zu verbreiten, wird beim Aussenden jeder Anfrage- und Antwortnachricht eine Beschreibung des Dienstgebiets des sendenden Blatt-Servers (Anfragen und Antworten gehen bei den im vorherigen Kapitel beschriebenen Algorithmen nur von Blatt-Servern aus) beigefügt. Ein Blatt-Server, der eine solche Nachricht empfängt, merkt sich die dort enthaltene Zuordnung in seinem Zwischenspeicher. Unter Verwendung dieses Mechanismus wird er bald Kenntnis über die Dienstgebiete anderer Blatt-Server, vor allem der in seiner Nachbarschaft, erlangen. Bevor ein Kontakt-Server eine Zuständigkeitsübergabe, eine Gebiets- oder eine Nachbarschaftsanfrage durchführt, die eine bestimmte Position oder ein bestimmtes Gebiet betrifft, überprüft er erst, ob er die/den dafür zuständigen Server aus seinem Zwischenspeicher ermitteln kann. Ist dies der Fall, versucht er diese(n) direkt zu kontaktieren, ohne die Hierarchie traversieren zu müssen. Da anzunehmen ist, dass sich die Dienstgebiete der Lokations-Server selten ändern, ist die Wahrscheinlichkeit, dass ein Eintrag aus dem Zwischenspeicher ungültig geworden ist, gering. Wir erwarten daher, dass die Leistungssteigerung den zusätzlichen (kleinen) Aufwand in nahezu allen Fällen deutlich übersteigt. Wegen der langen Gültigkeit dieser Einträge macht es auch Sinn, diese auf stabilem Speicher zu halten.

(Mobiles Objekt, aktueller Agent): Um die Bearbeitung von Positionsanfragen zu beschleunigen, kann ebenso die Adresse der aktuellen Agenten der mobilen Objekte zwischengespeichert

werden. Bei Empfang einer Antwort zu einer Anfrage speichert dazu ein Blatt-Server die Adresse des Servers, von dem die Antwort ausgeht, zusammen mit den Bezeichnern aller in der Antwort enthaltenen mobilen Objekte. Wenn eine Positionsanfrage durchgeführt werden soll, überprüft der Kontakt-Server wiederum, ob ein Eintrag für das entsprechende Objekt in seinem Zwischenspeicher enthalten ist. In diesem Fall versucht er direkt den entsprechenden Blatt-Server zu kontaktieren. Für mobile Objekte mit hoher Mobilität können die Einträge im Zwischenspeicher jedoch schnell ungültig werden, abhängig von der Größe der Dienstgebiete und der Geschwindigkeit und den Bewegungsmustern der mobilen Objekte. Es hängt demnach von der Häufigkeit von Zuständigkeitsübergaben zwischen Blatt-Servern ab, wie gut dieser Mechanismus funktioniert und ob es sinnvoll ist, die entsprechenden Einträge auf stabilem Speicher zu halten. Für den Fall, dass sich die Information über den aktuellen Agenten zu schnell ändert, wird in der Literatur vorgeschlagen, Verweise auf Server höherer Ebene zu speichern und so die Suchpfade zu verkürzen (siehe z. B. BAGGIO, BALLINITIJN & VAN STEEN (2000)), wobei eine geeignete Ebene adaptiv bestimmt wird. Die Dienstgebiete, die diesen Servern zugeordnet sind, sind größer und Zuständigkeitsübergaben demnach nicht so häufig.

(Mobiles Objekt, Lokationsbezeichner): Neben Informationen zum aktuellen Agenten eines mobilen Objekts ist es genauso gut möglich, dessen Lokationsbezeichner, der als Ergebnis einer vorangegangenen Anfrage zurückgeliefert wurde, zwischenzuspeichern. Falls ein Kontakt-Server eine weitere Positionsanfrage für ein zuvor abgefragtes mobiles Objekt erhält und die gespeicherte Positionsinformation noch genau genug ist (dies kann mit Hilfe von Gleichung 4-2 abgeschätzt werden), kann er die Anfrage direkt aus seinem Zwischenspeicher beantworten. Ob es überhaupt sinnvoll ist, Positionsinformationen zwischenzuspeichern, hängt von vielerlei Faktoren ab: Der Geschwindigkeit des mobilen Objekts, der Häufigkeit von Positionsanfragen und der in diesen geforderten Genauigkeit für dieses Objekt. Ebenso wie bei der Datenhaltung des Lokations-Servers selbst, lohnt sich das Speichern der Positionsinformationen auf stabilem Speicher nicht.

5.6 Zusammenfassung

In diesem Kapitel wurde die Architektur für einen verteilten skalierbaren Lokationsdienst entsprechend dem Dienstmodell aus Kapitel 4 vorgeschlagen. Der Aufbau des verteilten Dienstes und die verteilte Bearbeitung der Operationen wurde ausführlich diskutiert. Weiterhin wurde eine hauptspeicherbasierte Datenhaltungskomponente beschrieben, die dynamische Positionsinformationen effizient verwalten kann, und Mechanismen für deren Wiederherstellung nach einem Fehlerfall betrachten. Im nächsten Kapitel werden nun zuerst die dabei vorausgesetzten

Protokolle zur Aktualisierung von Positionsinformationen untersucht. In Kapitel 7 werden die hier vorgeschlagenen Mechanismen und Konzepte dann anhand von Messungen an einer prototypischen Implementierung untersucht.

Kapitel 6

Protokolle zur Übertragung von Positionsinformationen

Die in den vorigen Kapiteln beschriebenen Konzepte setzen an mehreren Stellen geeignete Protokolle für die Übertragung von Positionsinformationen voraus. In diesem Kapitel sollen entsprechende Protokolle nun ausführlich untersucht werden. Da die im LS gespeicherten Positionsinformationen, wegen der hohen geforderten Genauigkeit, sehr häufig aktualisiert werden müssen, ist die Verwendung von effizienten Protokollen von großer Bedeutung. Einerseits belegt das Übertragen der Positionsinformationen einen bestimmten Anteil der zur Verfügung stehenden Kommunikationsbandbreite, die im Falle von Mobilkommunikation teuer und knapp ist, andererseits belastet jede Positionsaktualisierung auch die Lokations-Server des LS. Unter diesen Gesichtspunkten wurden Protokolle zur Positionsaktualisierung bereits im Bereich der Lokationsverwaltung von Mobilfunknetzen behandelt, allerdings unter den Randbedingungen von diskreten Lokationsgebieten mit vergleichsweise großer Ausdehnung (in etwa zwischen 500 m und 35 km).

Weil der LS geometrische Positionsinformationen mit einer wesentlich höheren Auflösung verwalten soll, sind die dort erzielten Ergebnisse nicht direkt übertragbar. In diesem Kapitel wird daher zuerst eine Klassifizierung verschiedener Protokolle zur Aktualisierung von Positionsinformationen vorgenommen. Diese werden dann anhand der speziellen Anforderungen des LS betrachtet und miteinander verglichen. Anschließend wird näher auf die viel versprechende Klasse der Koppelnavigationsprotokolle eingegangen und ein neuartiges kartenbasiertes Koppelnavigationsprotokoll beschrieben. Ausführliche Simulationsergebnisse (basierend auf Bewegungen realer Objekte, die mit einem GPS-Sensor aufgezeichnet wurden) zeigen, dass dieses Protokoll die benötigten Aktualisierungsnachrichten gegenüber einem einfachen Koppelnavi-

gationsprotokoll, das bereits verglichen mit einem herkömmlichen Verfahren bis zu 80% weniger Nachrichtenaufwand verursacht, nochmal bis um die Hälfte reduziert.

Die in diesem Kapitel vorgenommene Untersuchung von Protokollen zur Übertragung von Positionsinformationen wurde erstmals in LEONHARDI & ROTHERMEL (2001) veröffentlicht. Mit der Betrachtung der Koppelnavigationsprotokolle hat sich die Diplomarbeit von Christian Nicu beschäftigt (NICU (2001)). Deren Ergebnisse wurden in LEONHARDI, NICU & ROTHERMEL (2002) veröffentlicht.

6.1 Problemstellung

Bei der folgenden Diskussion der Protokolle wird die Aktualisierung der Positionsinformation für ein einzelnes mobiles Objekt betrachtet, da deren Übertragung jeweils getrennt durchgeführt werden muss (vgl. die entsprechenden Algorithmen in Abschnitt 5.4.2). Die betrachtete Problemstellung umfasst daher die Übertragung der Positionsinformation von einer *Quelle*, an der diese mit Hilfe eines Positionierungssensors erfasst wird, zu einer *Senke*, die sie speichert und gegebenfalls weitergibt. Bei der Quelle handelt es sich beispielsweise um den mit einem GPS-Empfänger ausgerüsteten PDA des Benutzers. Die Senke, an der die Positionsinformation mit einer bestimmten Genauigkeit benötigt wird, ist in unserem Fall der für den Benutzer zuständige Lokations-Server (siehe Abbildung 6-1). Zur Übermittlung der Positionsinformationen wird eine (meist) drahtlose Kommunikationsverbindung verwendet. Protokolle, die zur Steuerung des entsprechenden Nachrichtenaustauschs eingesetzt werden können, werden in diesem Kapitel betrachtet.

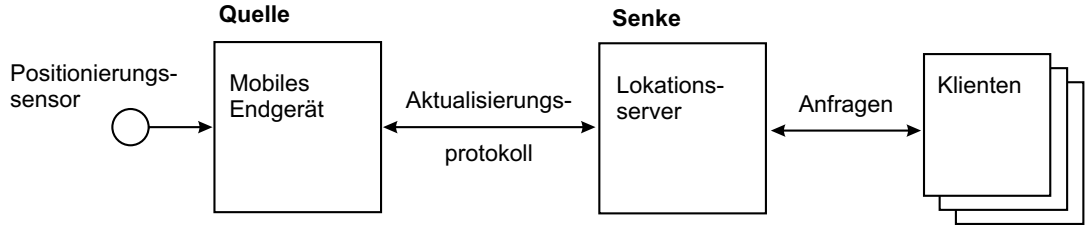


Abbildung 6-1. An der Übertragung von Positionsinformationen beteiligte Komponenten.

Da eine geforderte hohe Genauigkeit der Positionsinformationen zu einer hohen Zahl von ausgetauschten Nachrichten führt (z. B wird mehr als eine Positionsaktualisierung je zwei Sekun-

den bei einem einfachen Aktualisierungsprotokoll, einer geforderten Genauigkeit von 50 m und einem Fahrzeug mit einer Geschwindigkeit von 100 km/h benötigt), ist ein wichtiges Ziel beim Entwurf eines geeigneten Aktualisierungsprotokolls, die Anzahl der benötigten Nachrichten so weit wie möglich zu reduzieren.

Eine Reduzierung der Aktualisierungsnachrichten ist einerseits in Hinblick auf den Energieverbrauch sinnvoll. Da es sich bei den betrachteten mobilen Endgeräten in vielen Fällen um Geräte mit stark begrenztem Energievorrat handelt, kann sich ein häufiges Versenden von Positionsaktualisierungsnachrichten negativ auf deren Laufzeit auswirken. So wurden bei heutigen Mobilkommunikationssystemen große Anstrengungen unternommen, um den durch das Ankündigen des aktuellen Aufenthaltsbereichs eines Mobiltelefons entstehenden Energiebedarf zu verringern. Des Weiteren entstehen bei der Verwendung von heutigen Mobilkommunikationssystemen, wie sie u. A. bei der Verwaltung von Flotten von Lastkraftwagen (siehe z. B. das Fleetboard System, DAIMLERCHRYSLER AG (2002)) verwendet werden, z. T. beträchtliche Kosten durch die über SMS oder GPRS versendeten Aktualisierungsnachrichten. Durch eine Reduzierung von Aktualisierungsnachrichten lassen sich also auch Kosteneinsparungen erreichen.

Idealerweise soll es einem Klienten möglich sein, bei jeder Anforderung der Positionsinformation eine geforderte Genauigkeit anzugeben. Dies geht jedoch über die Anforderungen des LS hinaus, da dort die Genauigkeit der Positionsinformationen für ein mobiles Objekt insgesamt festgelegt wird (was aus Gründen des Datenschutzes wünschenswert ist). Zumindest muss daher eine minimale Genauigkeit für die Positionsinformation auf der Senke garantiert werden können.

Insgesamt werden die vorgestellten Protokolle also hinsichtlich der folgenden Gesichtspunkte betrachtet:

- Effiziente Übertragung der Positionsinformationen: Die Effizienz eines Protokolls ergibt sich durch die Anzahl der zur Übertragung der Positionsinformationen mit einer gewissen Genauigkeit benötigten Nachrichten.
- Effektivität der Übertragungen: Ob ein Protokoll seinen Einsatzzweck erfüllt, hängt davon ab, ob und bis zu welchem Grad es die geforderte Genauigkeit für die Positionsinformation einhalten kann. Diese Eigenschaft bezeichnen wir als die Effektivität des Protokolls.
- Toleranz gegenüber Verbindungsunterbrechung: Wegen der speziellen Eigenschaften einer drahtlosen Kommunikationsverbindung ist es wichtig, dass ein Protokoll in der Lage ist (zeitweilige) Unterbrechungen der Verbindung zu erkennen und mit diesen umzugehen.

Die in diesem Kapitel angestellten Betrachtungen sind zwar im Rahmen der Arbeiten am Lokationsdienst entstanden, sie können jedoch überall dort angewendet werden, wo (geometrische)

Positionsinformationen mit einer hohen Genauigkeit über eine Kommunikationsverbindung übertragen werden sollen.

6.2 Überblick

Protokolle zur Positionsaktualisierung können in drei wesentliche Klassen eingeteilt werden: anfragende, berichtende und kombinierte Protokolle, die wiederum in eine Reihe von Unterklassen unterteilt werden können. Jedes dieser Protokolle besitzt charakteristische Eigenschaften und ist damit für bestimmte Einsatzumgebungen und Anforderungen geeignet. Die Protokollklassen und ihre Beziehungen untereinander werden in den folgenden Abschnitten besprochen; eine Übersicht ist in Abbildung 6-2 dargestellt. Tabelle 6-1 auf Seite 93 fasst die Eigenschaften der einzelnen Protokolle noch einmal zusammen.

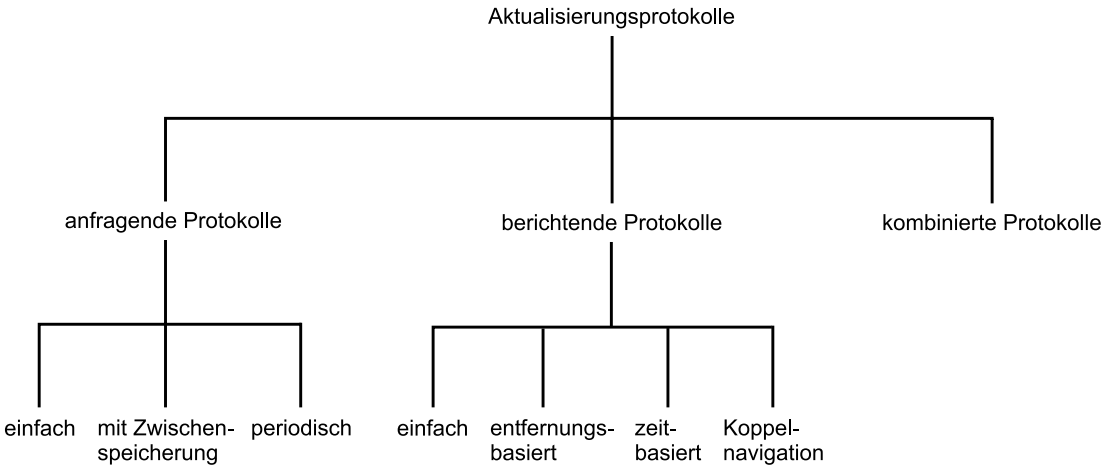


Abbildung 6-2. Klassifikation von Protokollen zur Übertragung von Positionsinformationen.

6.2.1 Anfragende Protokolle

Ein Protokoll wird als anfragendes Protokoll bezeichnet, wenn die Senke entscheidet, wann sie die Positionsinformationen von der Quelle anfordert. Die Senke kann bei dieser Entscheidung – anders als bei einem berichtenden Protokoll – die in einer Anfrage angegebene geforderte Genauigkeit beachten. Anfragende Protokolle haben zusätzlich den Vorteil, dass die Funktionalität

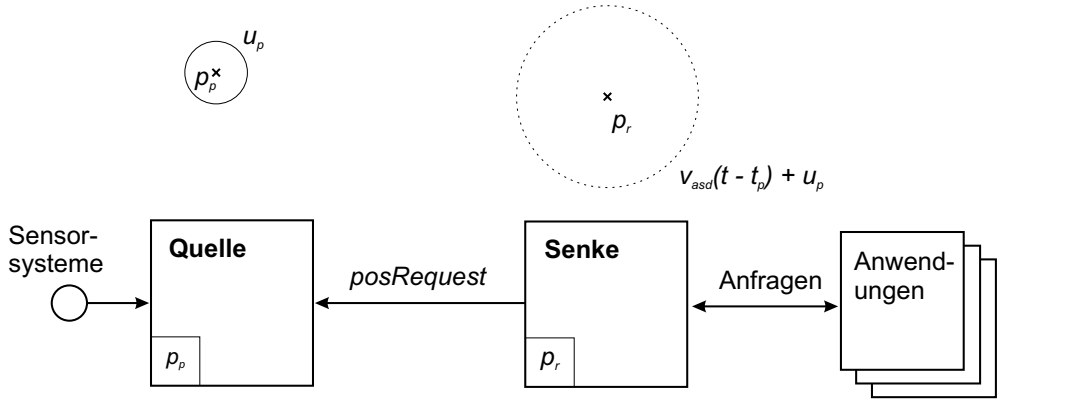
der Quelle sehr einfach zu realisieren ist, da dort weder umfangreiche Zustandsinformationen gehalten werden müssen noch komplizierte Algorithmen zu implementieren sind. Dies kann wichtig sein, wenn es sich dabei um ein einfaches mobiles Endgerät handelt; z. B. im Umfeld des Ubiquitous Computing (WEISER (1991)) um ein an einem mobilen Objekt befestigtes sogenanntes *Smart-Tag*. Das einfache anfragende Protokoll sowie das mit Zwischenspeicherung fordern die Informationen nur an, wenn eine Anwendung sie anfragt (*on-demand*). Es werden daher nur benötigte Informationen übertragen.

Einfaches anfragendes Protokoll (engl. *simple querying protocol*): In der einfachsten Version eines anfragenden Protokolls fordert die Senke die Informationen jedesmal bei der Quelle an, wenn sie von einer Anwendung benötigt werden (die auf der Quelle verfügbare Positionsinformation wird im Folgenden als p_p bezeichnet, ihre Ungenauigkeit als u_p). Die Senke muss dabei keine Kopie der Positionsinformationen halten und da immer die aktuellsten Information übertragen wird, wird auch die höchstmögliche Genauigkeit erreicht. Das *Mobile Location Protocol* (kurz MLP) für Mobilkommunikationssysteme definiert mit dem *Standard Location Immediate Request* z. B. ein solches Protokoll (siehe LIF (2001)). Bei einer hohen Anfragerate führt ein einfaches anfragendes Protokoll allerdings auch zu einer entsprechend hohen Anzahl von übertragenen Nachrichten. Die Antwortzeit der Senke gegenüber der Anwendung ist ebenfalls vergleichsweise hoch, da die Positionsinformationen jedesmal bei der Quelle abgefragt werden müssen. Wenn allerdings ein Benutzer aus Sicherheitsgründen nicht erlaubt, seine Positionsinformation außerhalb seines persönlichen Endgeräts zu speichern, ist auch nur dieses einfache Protokoll möglich. Ein Beispiel hierfür ist der in SPREITZER & THEIMER (1993) beschriebene Lokationsdienst, bei dem die Positionsinformationen jedes Benutzers jeweils von einem speziellen Benutzeragenten verwaltet werden, der jeden Zugriff darauf überwacht.

Anfragendes Protokoll mit Zwischenspeichern (engl. *cached querying protocol*): Beim einfachen Protokoll werden Positionsinformationen oft erneut angefragt, obwohl die zuletzt übermittelte Position noch die erforderliche Genauigkeit besessen hätte. Allerdings hat die Senke keine Möglichkeit dies eindeutig zu entscheiden, ohne zusätzliche Nachrichten mit der Quelle auszutauschen. Ist eine maximale Geschwindigkeit $\hat{v}$ für das mobile Objekt bekannt[1], so kann die Senke die minimale Genauigkeit der zuletzt übermittelten Positionsinformation zu einem späteren Zeitpunkt entsprechend Gleichung 4-2 berechnen. Beim anfragenden Protokoll mit Zwischenspeichern merkt sie sich daher die zuletzt übertragene Positionsinformation (in p_s) und

1. Diese kann entweder durch ein Beobachten des bisherigen Bewegungsverhaltens des mobilen Objekts ermittelt werden oder für bestimmte Objektklassen vorgegeben sein (so kann für ein Fahrzeug meist eine maximale Geschwindigkeit von 200 km/h angenommen werden).

fordert die Information nur neu an, wenn sie laut dieser Abschätzung nicht die geforderte Genauigkeit u'_q hat (siehe Abbildung 6-3). Bei einer *pessimistischen* Abschätzung, welche dazu die (tatsächlich) maximale Geschwindigkeit verwendet, kann so immer die geforderte Genauigkeit garantiert werden. Allerdings werden noch immer, vor allem wenn die maximale Geschwindigkeit deutlich über der durchschnittlichen Geschwindigkeit liegt, häufig Positionsinformationen übertragen, obwohl die Genauigkeit der gespeicherten Kopie noch ausreichen würde. Falls es nicht zwingend erforderlich ist, dass die geforderte Genauigkeit immer eingehalten wird, kann für eine *optimistische* Abschätzung eine Geschwindigkeit kleiner der Maximalgeschwindigkeit angenommen werden ($v_{asd} < \hat{v}$). Dabei kann es sich z. B. um die durchschnittliche Geschwindigkeit des mobilen Objekts handeln. Bei einem anfragenden Protokoll mit Zwischenspeichern sind die Antwortenzeiten davon abhängig, ob die Senke die Informationen bei der Quelle erfragt oder direkt die Kopie zurückliefern kann. Sie sind damit im Durchschnitt geringer als beim einfachen Protokoll.



on *posQuery*(*o, u'q*)
  if not valid p_r exists **then**
    (p_r, u_p, t_p) := **send** posRequest(o) **to** source
  endif
  return p_r
end

p_r is valid, if**:**
  $v_{asd}(t - t_p) + u_p < u'_q$

Abbildung 6-3. Funktionsweise des anfragenden Protokolls mit Zwischenspeichern.

Periodisch anfragendes Protokoll (engl. *periodic querying protocol*): Beim periodisch anfragenden Protokoll fordert die Senke die Positionsinformationen von der Quelle an, sobald jeweils ein bestimmtes Zeitintervall *T* seit der letzten Aktualisierung vergangen ist. Obwohl die

Initiative hierbei von der Senke ausgeht, hat dieses Protokoll dieselben Eigenschaften wie das zeitbasierte berichtende Protokoll, das im nächsten Abschnitt beschrieben ist.

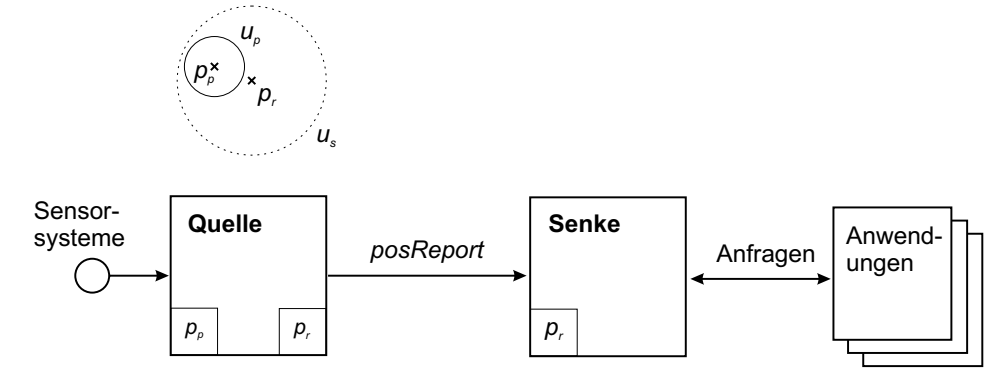
6.2.2 Berichtende Protokolle

Bei einem berichtenden Protokoll geht die Initiative, im Gegensatz zu den anfragenden Protokollen, von der Quelle aus. Diese merkt sich, welche Positionsinformationen sie zuletzt an die Senke geschickt hat, und kennt daher die Position, die diese aktuell für das mobile Objekt annimmt. Sie versendet eine Aktualisierungsnachricht, wenn zur tatsächlichen Position des Objekts ein gewisser Zeit- oder Entfernungsschwellwert überschritten wird. Die Senke speichert die zuletzt übertragene Positionsinformation und liefert bei einer Anfrage immer den Inhalt dieser Kopie zurück. Die Genauigkeit ihrer Kopie kann die Senke aus dem Zeit- oder Entfernungsschwellwert ermitteln. Bei einem ausschließlich berichtenden Protokoll kann daher die Positionsinformation nur mit dieser voreingestellten Genauigkeit zurückgeliefert werden, auch wenn in der Anfrage eine höhere Genauigkeit gefordert ist. Da die Quelle nicht kontaktiert wird, können allerdings kurze Antwortzeiten erreicht werden. Ein berichtendes Protokoll ist meist effizienter, wenn die Positionsinformationen oft angefragt werden (siehe Abschnitt 6.4), was z. B. der Fall ist, wenn die Senke das Eintreten eines Ereignisses überwachen soll.

Einfaches berichtendes Protokoll (engl. *simple reporting protocol*): Das einfachste berichtende Protokoll versendet die Positionsinformationen jedesmal, wenn sich diese auf der Quelle ändern. Die Informationen auf der Senke haben damit auch die höchstmögliche Genauigkeit. Die Anzahl der versendeten Nachrichten hängt in diesem Fall nur von der Aktualisierungsrate des Sensorsystems ab, die allerdings meist sehr hoch ist. Dieses einfache Protokoll wird aus diesem Grund hier nicht weiter betrachtet.

Zeitbasiertes berichtendes Protokoll: (engl. *time-based reporting protocol*): Bei einem zeitbasierten Protokoll wird die Positionsinformation periodisch übertragen, sobald ein bestimmtes Zeitintervall T vergangen ist. Die Aktualisierungsrate ist damit durch dieses Intervall festgelegt und hängt nicht vom Bewegungsverhalten des mobilen Objekts ab, was eine entsprechende zeitliche aber keine räumliche Genauigkeit garantiert. Wenn sich das Objekt langsam oder gar nicht bewegt, werden häufig Positionsaktualisierungen versendet, die sich nicht oder nur unwesentlich unterscheiden. Bewegt sich das Objekt hingegen schnell, so werden nicht genug Aktualisierungen erzeugt um eine benötigte (räumliche) Genauigkeit zu erreichen. Mit dem *Triggered Location Reporting Service* unterstützt der MLP-Standard beispielsweise auch ein zeitbasiertes Protokoll.

*Entfernungsbasiert*es *berichtendes Protokoll* (engl. *distance-based reporting protocol*): Bei einem entfernungsbasierten Protokoll versendet die Senke eine Aktualisierungsnachricht, wenn die Entfernung zwischen der zuletzt übertragenen Position und der aktuell gemessenen Position einen bestimmten Entfernungsschwellwert *D* überschreitet. Das Protokoll geht somit besser auf die Bewegungscharakteristik eines mobilen Objekts ein, indem es mehr Nachrichten versendet, wenn das Objekt sich schnell bewegt, und wenig oder gar keine, wenn es langsam bzw. stationär ist. Ein entfernungsbasiertes Protokoll ist daher besser geeignet für Objekte, die sich nur sporadisch bewegen und für längere Zeiten unbeweglich sind, als ein zeitbasiertes. So ein Verhalten ist z. B. typisch für Benutzer in einer Büroumgebung. Um eine bestimmte geforderte Abweichung u_s für die Positionsinformationen auf der Senke garantieren zu können, muss der Schwellwert für die Entfernung *D* auf u_s abzüglich der Ungenauigkeit des Sensorsystems u_p gesetzt werden ($D = u_s - u_p$). Abbildung 6-4 zeigt die Funktionsweise des entfernungsbasierten Protokolls im Detail. Es ist weiterhin einfach möglich, das zeit- und das entfernungsbasierte Protokoll zu integrieren, um eine Kombination ihrer Eigenschaften zu erhalten.



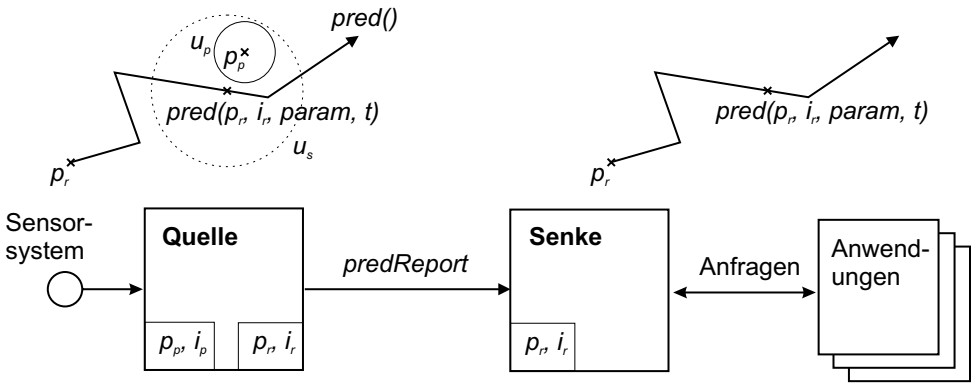

```
on sensorUpdate(p_p)
   if ( Distance(p_p, p_r) + u_p > u_s ) then
      send posReport(p_p) to server
      p_r := p_p
   endif
end
```

```
on posQuery(o)
   return p_r
end
```

Abbildung 6-4. Funktionsweise des entfernungsbasierten berichtenden Protokolls.

Koppelnavigationsprotokolle (engl. *dead-reckoning protocol*): Koppelnavigationsprotokolle sind eine Optimierung des entfernungsbasierten Protokolls, die sich die oft (streckenweise) vorhersagbaren Bewegungen mobiler Objekte zunutze machen. So herrschen bei einer Fahrt auf einer Autobahn meist geradlinige Bewegungen vor. Prinzipiell arbeiten alle Koppelnavigations-

protokolle nach dem folgenden Schema: Sowohl die Quelle als auch die Senke verwenden eine identische Vorhersagefunktion *pred()*, um die aktuelle Position des mobilen Objekts ausgehend von dessen zuletzt übermittelter Position p_r, weiteren Informationen über dessen Zustand i_r wie Geschwindigkeit $i_r.v$ und Bewegungsrichtung $i_r.dir$, allgemeinen Parametern *param* (z. B. Karteninformationen) sowie dem aktuellen Zeitpunkt *t* vorherzusagen. Falls die Quelle entdeckt, dass die vorhergesagte Position von der aktuellen Position des Objekts um mehr als einen vorgegebenen Entfernungsschwellwert *D* abweicht, wird wie beim entfernungsbasierten Protokoll eine Positionsaktualisierung versendet. Um eine maximale Abweichung der Positionsinformationen auf dem Server garantieren zu können, müssen sowohl Quelle als auch Senke eine identische Vorhersagefunktion verwenden. Die grundlegende Funktionsweise eines Koppelnavigationsprotokolls ist in Abbildung 6-5 gezeigt. Auf verschiedene mögliche Varianten wird in Abschnitt 6.3 detailliert eingegangen.



on sensorUpdate(p_p, i_p)
  if (Distance(p_p, *pred*(p_r, i_r, *t*)) +
      $u_p > u_s$) **then**
    send *predReport*(p_p, i_p) **to** server
    p_r, i_r := p_p, i_p
  endif
end

on *posQuery*
  return *pred*(p_r, i_r, *param*, *t*)
end

Lineare Vorhersage:
  pred(*p*, *i*, *param*, *t*) := *p* + *i.dir* · *i.v* · (*t* - *i.t*)

Abbildung 6-5. Funktionsweise eines Koppelnavigationsprotokolls.

6.2.3 Kombiniertes Protokoll

Sowohl anfragende als auch berichtende Protokolle haben ihre Nachteile. Während ein rein anfragendes Protokoll nicht auf unterschiedliche Bewegungsmuster des mobilen Objekts reagie-

ren kann, ist es bei einem berichtenden Protokoll nicht möglich, auf unterschiedliche von den Anwendungen geforderte Genauigkeiten einzugehen. Ein *kombiniertes Protokoll* (engl. *combined protocol*) verbindet die Eigenschaften sowohl der anfragenden als auch der berichtenden Protokolle. Entsprechend dem entfernungsbasierten berichtenden Protokoll, versendet die Quelle Nachrichten zur Positionsaktualisierung, um eine vorgegebene räumliche Genauigkeit *D* der lokalen Kopie auf der Senke zu garantieren. Falls diese nicht für die in einer Anfrage geforderte Genauigkeit ausreicht, fordert die Senke die aktuellen Positionsinformationen entsprechend eines pessimistischen anfragenden Protokoll mit Zwischenspeichern von der Quelle an. Das kombinierte Protokoll garantiert damit auch die Einhaltung der von der Anwendung geforderten Genauigkeit.

Das Verhalten des kombinierten Protokolls kann somit durch den Entfernungsschwellwert *D* gesteuert werden (siehe Abschnitt 6.4). Um die Anzahl der versendeten Nachrichten, bei denen es sich hier sowohl um Positionsaktualisierungen als auch das Abfragen der Positionsinformationen bei der Quelle handelt, zu minimieren, muss dieser Schwellwert abhängig von der Bewegungscharakteristik des mobilen Objekts und der Anfragecharakteristik der Anwendungen geeignet gewählt werden. Falls die beteiligten Komponenten die Bewegungen des mobilen Objekts und die Anfragen zu dessen Positionsinformation im Betrieb überwachen, kann über *D* das kombinierte Protokoll auch adaptiv an die aktuelle Situation angepasst werden. Wenn diese Entscheidung, ob die Genauigkeit der lokalen Kopie ausreicht oder ob deren Genauigkeit durch Verändern des Schwellwerts *D* angepasst werden soll, nur bei einer eintreffenden Positionsaktualisierung und/oder Anfrage getroffen wird, erhöht sich zwar der Aufwand zur Bearbeitung von Aktualisierungsnachrichten und Anfragen entsprechend, es wird aber kein zusätzlicher Überwachungsprozess benötigt.

Die Antwortzeit der Senke bei Einsatz eines kombinierten Protokolls ist wie bei einem anfragenden Protokoll mit Zwischenspeichern unterschiedlich und hängt davon ab, ob die Genauigkeit der lokalen Kopie noch ausreichend ist oder ob die Quelle kontaktiert werden muss. Die Funktionsweise eines kombinierten Protokolls ist als Kombination eines entfernungsbasierten berichtenden und eines anfragenden Protokolls mit Zwischenspeichern in Abbildung 6-6 gezeigt. Anstelle des entfernungsbasierten Protokolls könnte hier auch ein effizienteres Koppelnavigationsprotokoll eingesetzt werden, da dieses dieselben Eigenschaften besitzt, was hier allerdings aus Gründen der Übersichtlichkeit nicht betrachtet werden soll.

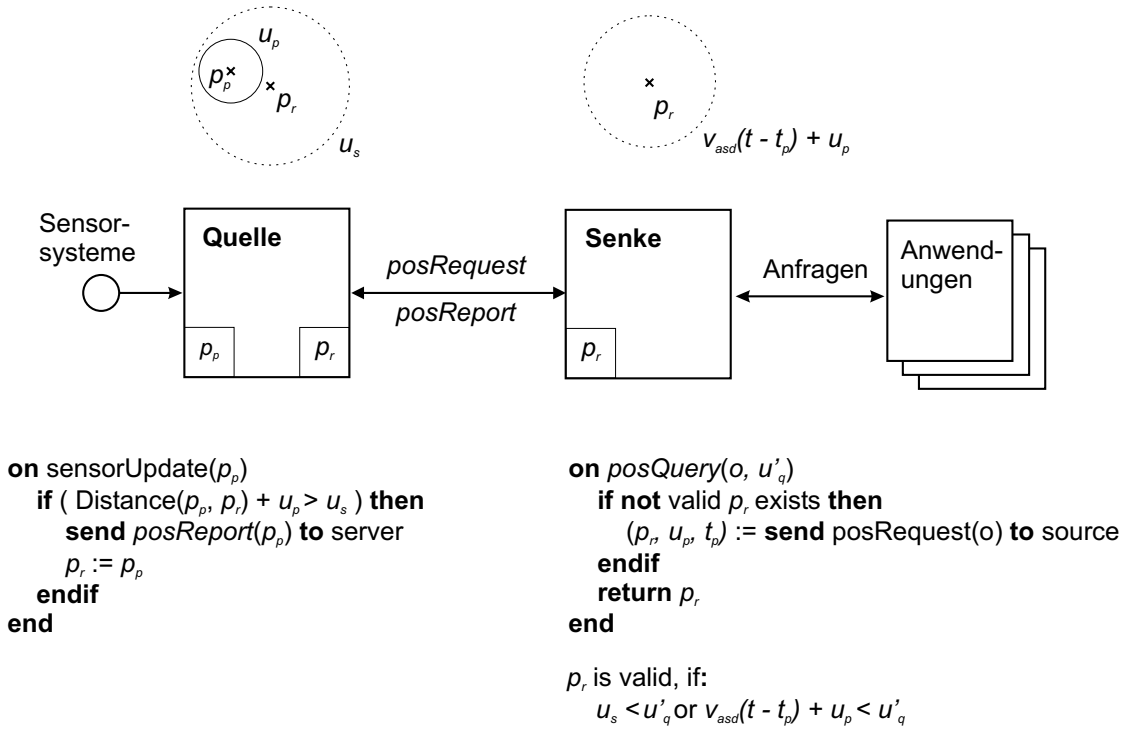


Abbildung 6-6. Funktionsweise des kombinierten Protokolls.

6.2.4 Verhalten bei Unterbrechung der Netzverbindung

Ein häufig auftretendes Problem bei mobilen Endgeräten und drahtloser Datenübertragung ist eine zeitweilige Unterbrechung der Kommunikationsverbindung (engl. *disconnection*, siehe z. B. SATYANARAYANAN (1996)), beispielsweise weil ein Fahrzeug durch einen Tunnel fährt. Es entsteht dadurch eine Netzpartitionierung, mit dem mobilen Gerät in der einen und dem restlichen Netzwerk in der anderen Partition (vgl. Abschnitt 5.1). Ein Protokoll, das für die Übertragung von Positionsinformationen mobiler Endgeräte eingesetzt werden soll, muss daher in der Lage sein, mit solchen Verbindungsunterbrechungen umzugehen.

In diesem Abschnitt betrachten wir die in den vorhergehenden Abschnitten vorgestellten Protokolle hinsichtlich ihres Verhaltens bei einer Verbindungsunterbrechung. Wichtig ist dabei einerseits, wie lange es dauert, bis die Senke eine Abweichung vom regulären Betrieb erkennen kann. Wie in Abschnitt 5.3.3 beschrieben setzt ein Lokations-Server dazu einen maximalen Zeitabstand zwischen zwei Positionsaktualisierungen *mui* voraus. Andererseits ist wichtig, wie stark die Genauigkeit der von der Senke innerhalb dieses Zeitintervalls zurückgelieferten Positions-

information von der erwarteten Genauigkeit maximal abweichen kann. Wo notwendig beschreiben wir auch entsprechende Modifikationen an den Protokollen, die eine Behandlung von Verbindungsunterbrechungen erlauben.

Anfragende Protokolle: Bei einem anfragenden Protokoll wird die Unterbrechung der Verbindung erkannt, sobald die Senke, als Reaktion auf eine Anfrage einer Anwendung, die Positionsinformation von der Quelle abfragt (bei einem Protokoll mit Zwischenspeichern nur dann, wenn die Genauigkeit der lokalen Kopie nicht ausreicht). In keinem Fall wird eine Positionsinformation mit geringerer Genauigkeit zurückgegeben, als bei bestehender Netzverbindung. Stattdessen kann eine entsprechende Fehler- oder Warnmeldung erzeugt werden.

Zeitbasiertes berichtendes Protokoll: Bei Verwendung eines zeitbasierten berichtenden Protokolls, erkennt die Senke eine Verbindungsunterbrechung, wenn das Zeitintervall *T* seit der letzten Aktualisierung abgelaufen und keine Aktualisierungsnachricht eingetroffen ist (d. h. *miu* = *T*). Eine zurückgegebene Positionsinformation hat daher auch in diesem Fall immer die geforderte Genauigkeit.

Entfernungsbasiertes berichtendes Protokoll: Wenn ausschließlich ein entfernungsbasiertes Protokoll eingesetzt wird, kann die Senke eine Unterbrechung der Verbindung nicht von sich aus erkennen. Sie nimmt hingegen an, dass sich das mobile Objekt um nicht mehr als den Entfernungsschwellwert *D* bewegt hat, und gibt die zuletzt übertragene Position als aktuelle Position zurück. Um Unterbrechungen der Verbindung erkennen zu können, kann das entfernungsbasierte mit dem zeitbasierten Protokoll kombiniert werden. Dazu wird ein oberes Zeitintervall $\hat{T}$ festgelegt, nach dessen Ablauf spätestens eine Positionsaktualisierung versendet werden muss. Eine Verbindungsunterbrechung wird also nach Ablauf von $miu = \hat{T}$ erkannt, woraus sich eine maximale Ungenauigkeit der zurückgelieferten Informationen von $\hat{T} \cdot \hat{v} + u_p$ ergibt. Ein geeigneter Wert für $\hat{T}$ muss durch eine Abwägung zwischen dem erhöhten Kommunikationsaufwand und der maximalen Ungenauigkeit, die ein Anwender bereit ist in Kauf zu nehmen, gefunden werden.

Koppelnavigationsprotokolle: Ein Koppelnavigationsprotokoll verhält sich gegenüber Verbindungsunterbrechungen wie das entfernungsbasierte Protokoll und kann durch einen entsprechenden Mechanismus erweitert werden.

Kombiniertes Protokoll: Wie sich ein kombiniertes Protokoll bei einer Verbindungsunterbrechung verhält hängt davon ab, wie oft es die Positionsinformationen von der Quelle erfragt und damit, welche Genauigkeit die auf der Senke gespeicherte lokale Kopie hat. Wenn keine genauere Positionsinformationen angefragt werden, verhält sich das kombinierte Protokoll wie das entfernungsbasierte berichtende Protokoll und liefert möglicherweise ungenaue Positionsinformationen zurück. Es ist jedoch auch hier möglich, ein maximales Zeitintervall $\hat{T}$ vorzugeben.

Tabelle 6-1 gibt zusammenfassend noch einmal einen Überblick über die Betrachtungen zu den verschiedenen Klassen von Übertragungsprotokollen, die im vorangegangenen Abschnitt beschrieben sind.

Protokoll	Kann obere Grenze für die Ungenauigkeit der zurückgelieferten Informationen garantieren.	Ermöglicht die geforderte Genauigkeit pro Anfrage vorzugeben.	Anzahl der Nachrichten ist abhängig von Bewegungscharakteristik des mobilen Obj.	Anzahl der Nachrichten ist abhängig von Anfragerate und geforderter Genauigkeit.	Ist in der Lage, (ohne Modifikationen) eine Unterbrechung der Verbindung zu erkennen.
anfragend:					
einfach	×	×	–	×	×
mit Zwischenspeichern					
pessimistisch	×	×	–	×	×
optimistisch	–	×	–	×	×
periodisch	–	–	–	–	×
berichtend:					
einfach	×	–	–	–	–
zeitbasiert	–	–	–	–	×
entfernungsbasiert	×	–	×	–	–
Koppelnavigation	×	–	×	–	–
kombiniert:	×	×	×	×	(×)

Tabelle 6-1. Zusammenfassung der Eigenschaften unterschiedlicher Protokolle zur Übertragung von Positionsinformationen.

6.3 Koppelnavigationsprotokolle

In diesem Abschnitt soll die im vorherigen Abschnitt beschriebene viel versprechende Klasse der Koppelnavigationsprotokolle näher betrachtet werden. Dazu wird zuerst ein Überblick über

verschiedene Varianten von Koppelnavigationsprotokollen gegeben und anschließend ein neuartiges kartenbasiertes Protokoll vorgestellt.

6.3.1 Übersicht

Verschiedene Varianten der Koppelnavigationsprotokolle unterscheiden sich hauptsächlich in ihrer Vorhersagefunktion. Während eine einfache Vorhersagefunktion nur den aktuellen Zustand des mobilen Objekts betrachtet (z. B. Bewegungsrichtung und Beschleunigung), gehen andere Vorhersagefunktionen davon aus, dass sich das mobile Objekt auf einer bestimmten Weg oder auf einem Wegenetzwerk bewegt. Abbildung 6-7 zeigt einen Überblick über die verschiedenen Varianten von Koppelnavigationsprotokollen, die im Folgenden näher besprochen werden.

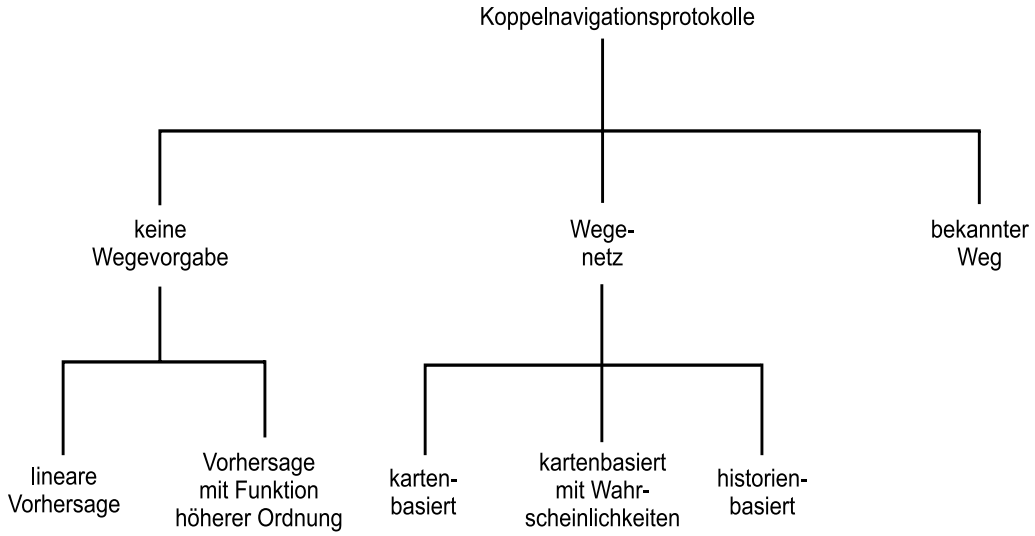


Abbildung 6-7. Einteilung verschiedener Koppelnavigationsprotokolle anhand der eingesetzten Vorhersagefunktion.

Lineare Vorhersagefunktion: Bei diesem einfachen Verfahren geht die Vorhersagefunktion davon aus, dass sich das mobile Objekt mit gleichbleibender Geschwindigkeit auf einer geraden Linie weiterbewegt, die aus der zuletzt gemeldeten Position und Bewegungsrichtung gebildet wird. Wie sich in unseren Simulationen gezeigt hat (siehe Abschnitt 6.5.3), erreicht dieses einfach zu implementierende Protokoll oft schon eine deutliche Verringerung der Aktualisierungshäufigkeit gegenüber dem verwandten entfernungsbasierten Protokoll. Außerdem kann es als

Rückfallstrategie für die komplizierteren Verfahren verwendet werden. Abbildung 6-8 zeigt ein Beispiel für die Funktionsweise des Koppelnavigationsprotokolls mit linearer Vorhersagefunktion.

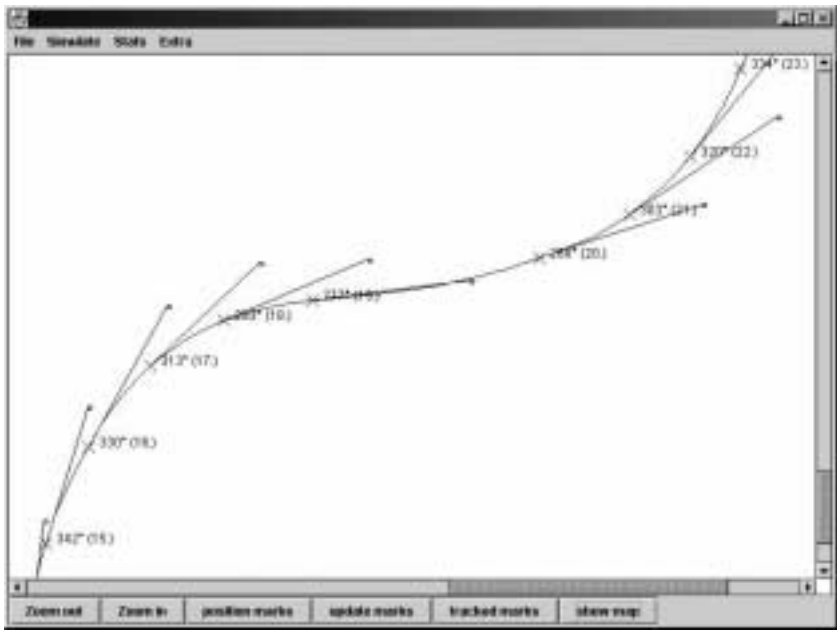


Abbildung 6-8. Beispiel für die Positionsaktualisierungen bei einem Koppelnavigationsprotokoll mit linearer Vorhersagefunktion.

Vorhersagefunktionen höherer Ordnung: Anstatt anzunehmen, dass sich das mobile Objekt auf einer geraden Linie weiterbewegt, könnte auch eine Funktion höherer Ordnung (z. B. eine *Spline*-Kurve, FOLEY ET AL. (1995)) für die Wegevorhersage verwendet werden. Dies hätte den Vorteil, dass damit auch die Bewegung in einer Kurve (siehe Abbildung 6-8) erfasst werden kann. Ähnliches könnte auch für die Geschwindigkeit angewendet werden, um Beschleunigen und Abbremsen zu berücksichtigen. Da allerdings die Beschleunigungs- bzw. Abbremsphasen im Straßenverkehr und noch mehr bei Fußgängern recht kurz sind, ist bei deren Berücksichtigung eine signifikante Verbesserung nur für sehr hohe geforderte Genauigkeiten zu erwarten. Wir haben uns gegen eine weitergehende Betrachtung von Vorhersagefunktionen höherer Ordnung zu Gunsten des im Folgenden vorgestellten kartenbasierten Protokolls entschieden, da dieses auch bereits eine bessere und weniger problematische Wegevorhersage beinhaltet.

Kartenbasierte Koppelnavigation: Mobile Objekte bewegen sich oft auf einem vorgegebenen Wegenetz, wie z. B. Personen, die durch die Straßen einer Stadt laufen, oder Kraftfahrzeuge auf einer Autobahn. Das kartenbasierte Koppelnavigationsprotokoll versucht daher die aktuelle Position des mobilen Objekts auf eine Straße aus einer integrierten Karte abzubilden (engl. *map-*

matching). Die Vorhersagefunktion nimmt dann an, dass sich das Objekt auf dieser Straße mit der zuletzt gemeldeten Geschwindigkeit weiterbewegt. An einer Kreuzung wird versucht die Abzweigung vorherzusagen, der das mobile Objekt mit der höchsten Wahrscheinlichkeit folgen wird. Grundsätzlich könnte auch nach jeder Abzweigung eine Aktualisierungsnachricht gesendet werden, was auf einer Autobahn, wo diese selten sind, auch gut funktionieren würde. Im Stadtverkehr würden die häufigen Abzweigungen allerdings zu vielen Aktualisierungsnachrichten führen. Das kartenbasierte Protokoll setzt eine detaillierte Straßenkarte voraus, die aber in einem Fahrzeugen bereits Bestandteil der sich zunehmender Beliebtheit erfreuenden Navigationssysteme ist. Weitere Informationen aus der Straßenkarte, wie Angaben über die Hauptstraßen oder Geschwindigkeitsbeschränkungen, können dazu verwendet werden, das kartenbasierte Verfahren weiter zu verbessern. Im nächsten Abschnitt gehen wir näher auf das kartenbasierte Koppelnavigationsprotokoll ein.

Kartenbasierte Koppelnavigation mit Wahrscheinlichkeitsinformationen: Um die Vorhersagegenauigkeit für die an einer Kreuzung genommene Abzweigungen zu erhöhen, ist eine Erweiterung des kartenbasierten Verfahrens denkbar, bei dem die Kanten der Karte mit Wahrscheinlichkeitsinformationen annotiert werden. Die den von einem Kreuzung abgehenden Abzweigungen zugeordneten Wahrscheinlichkeiten geben an, welcher Anteil der mobilen Objekte eine bestimmte Abzweigung nimmt (objektunabhängig) oder wie oft dies bei einem bestimmten Objekt der Fall ist (objektabhängig) und können durch eine Beobachtung des Bewegungsverhaltens der mobilen Objekte gewonnen werden. Die Vorhersagefunktion wählt daraufhin immer die Abzweigung mit dem höchsten Wahrscheinlichkeitswert aus. In wie weit diese Erweiterung eine Verbesserung gegenüber dem einfachen kartenbasierten Verfahren bringt, hängt von der Häufigkeit der Abzweigungen ab. Demgegenüber steht ein deutlich höherer Aufwand, der sich aus der Bestimmung und gegebenfalls Aktualisierung der Wahrscheinlichkeitswerte ergibt.

Historienbasierte Koppelnavigation: Falls keine Straßenkarte verfügbar ist, kann diese aus Aufzeichnungen von vergangenen Bewegungen der mobilen Objekte erstellt werden. Dabei wird wiederum davon ausgegangen, dass mobile Objekte häufig dieselben Wege verwenden, beispielsweise wenn ein Pendler zur Arbeit fährt. Falls diese Kartenerstellung auf einer ausreichend großen Datenbasis durchgeführt werden kann, ist das Ergebnis ein Wegenetz wie für das kartenbasierte Protokoll vorausgesetzt. Die Kartenerzeugung kann wiederum für alle Objekte gemeinsam (objektunabhängig) oder für ein bestimmtes Objekt (objektabhängig) erfolgen und wird ständig fortgeführt, während sich ein mobiles Objekte bewegt. Zusammenfassend verursacht die Erzeugung der Karten aus Historieninformationen einen großen Aufwand; die Ergebnisse werden bestenfalls (d. h. nach einer vollständigen Kartenerzeugung) dem des kartenbasierten Protokolls (mit objektabhängigen Wahrscheinlichkeitsinformationen) entsprechen. Sein

Vorteil liegt darin, dass es auch für Wege und Gebiete verwendet werden kann, für die keine Karteninformationen vorliegen. Im Folgenden wird das historienbasierte Verfahren allerdings nicht weiter im Detail betrachtet.

Koppelnavigation mit bekanntem Weg: Falls der Weg, den ein mobiles Objekt nehmen wird, vorab bekannt ist, muss bei einem entsprechenden Koppelnavigationsverfahren nur noch die Geschwindigkeit und nicht mehr die Bewegungsrichtung betrachtet werden (siehe WOLFSON ET AL. (1999)). Von der Leistungsfähigkeit entspricht das Koppelnavigationsprotokoll für bekannte Wege einem optimalen kartenbasierten Verfahren, das an jeder Kreuzung die richtige Abzweigung wählt. Bei unseren Betrachtungen sind wir jedoch nicht davon ausgegangen, dass die zukünftige Wege der mobilen Objekte dem System vorab bekannt sind. Dies ist z. B. nur der Fall, wenn ein Navigationssystem mit einer der geforderten Genauigkeit entsprechenden Auflösung verwendet wird und dessen Benutzer nicht von dem vorgegebenen Weg abweicht.

6.3.2 Ein kartenbasiertes Koppelnavigationsprotokoll

Da das kartenbasierte Verfahren gerade für die Fahrzeugnavigation einen guten Kompromiss zwischen Aufwand und Flexibilität darstellt, haben wir uns bei der Betrachtung von weiterführenden Koppelnavigationsprotokollen auf dieses konzentriert und beschreiben es in diesem Abschnitt ausführlicher (für weitere Details sei der interessierte Leser auf NICU (2001) verwiesen). Insgesamt unterscheidet sich das kartenbasierte Protokoll von dem grundlegenden Algorithmus für Koppelnavigationsprotokolle dadurch, dass bei der Erfassung der Sensorinformation die Position des mobilen Objekts auf eine Karte abgebildet wird und dass die Vorhersagefunktion Karteninformationen verwendet.

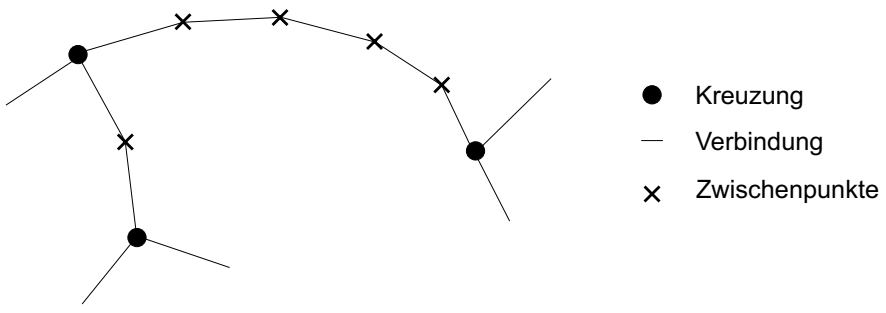


Abbildung 6-9. Beispiel für die Karteninformation, wie sie vom kartenbasierten Koppelnavigationsprotokoll verwendet wird.

Die vom kartenbasierten Koppelnavigationsprotokoll verwendeten Karteninformationen, die sowohl auf der Quelle als auch auf der Senke vorhanden sein müssen, haben den folgenden Aufbau: Alle erfassten Straßenkreuzungen sind durch einen eindeutigen Bezeichner und ihre geographische Position beschrieben. Die Kreuzungen können miteinander durch ebenfalls eindeutig identifizierbare gerichtete Verbindungen verknüpft sein, die den Straßen entsprechen. Um den Verlauf einer Straße genauer modellieren zu können, wird eine Verbindung meist durch Zwischenknoten in weitere Abschnitte unterteilt. Für die einfacheren Zwischenknoten ist neben ihrer Zugehörigkeit zu einer bestimmten Verbindung wiederum die geographische Position angegeben. Ein Ausschnitt einer solchen Karte ist als Beispiel in Abbildung 6-9 gezeigt. Die für unsere Simulationen verwendete Karte wurde aus einer für die Fahrzeugnavigation verwendeten Straßenkarte extrahiert[2].

Der Algorithmus zum Kartenabgleich wählt für ein mobiles Objekt eine aktuell nächstliegende Verbindung aus und berechnet eine korrigierte Position p_c, indem die Position rechtwinklig auf die Verbindung verschoben wird (siehe Abbildung 6-10). Eine Position kann auf eine Verbindung abgebildet werden, falls sie maximal die Entfernung u_m zu dem nächstgelegenen Punkt auf der Verbindung hat. Der Parameter u_m beschreibt damit, wie exakt der Kartenabgleich erfolgen soll und spiegelt die Genauigkeit des Positionierungssensors wider. Bei der Initialisierung werden potenzielle Verbindungen für die aktuelle Position eines mobilen Objekts mit Hilfe eines mehrdimensionalen Indexes über die Karteninformationen gefunden. Es wird dann die nächstgelegene Verbindung ausgewählt, falls sie nicht weiter als u_m entfernt ist. Eine Aktualisierungsnachricht wird – wie durch das grundlegende Protokoll vorgegeben – gesendet, wenn die Entfernung zwischen der wirklichen Position des mobilen Objekts und der von der Vorhersagefunktion (siehe unten) gelieferten Position den Betrag u_s überschreitet. Die Aktualisierungsnachricht enthält beim kartenbasierten Verfahren die korrigierte Position des mobilen Objekts p_c, seine Geschwindigkeit *i.v* und den Bezeichner der aktuellen Verbindung *i.l*.

Wenn die Quelle entdeckt, dass die Position des Objekts weiter als u_m von dessen aktueller Verbindung entfernt ist (d. h. seine Position kann nicht mehr auf die aktuelle Verbindung abgebildet werden), sucht sie nach der korrekten Verbindung durch entweder Vorwärts- oder Rückwärtsverfolgung.

Vorwärtsverfolgung: Wenn das mobile Objekt um mehr als eine Entfernung von u_m über das Ende B der aktuellen Verbindung hinausgelaufen ist (unter der Voraussetzung, dass es sich von A nach B bewegt), wird angenommen, dass es einen Kreuzungspunkt erreicht hat und eine Vorwärtsverfolgung wird eingeleitet. Zu diesem Zweck wird die Entfernung zwischen der Position

2. Der Medianwert für die Verbindungslänge beträgt bei den verwendeten Kartendaten in etwa 35 m.

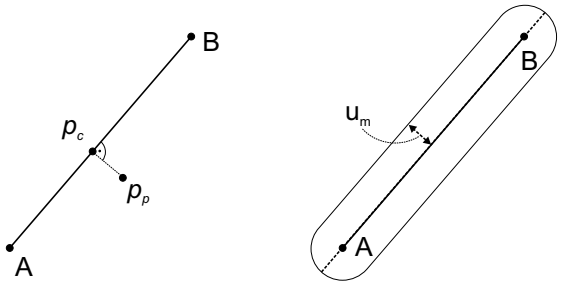


Abbildung 6-10. Kartenabgleich im kartenbasierten Koppelnavigationsverfahren: Abbildung der Position eines mobilen Objekts p_p *auf eine korrigierte Position* p_c *auf einer Verbindung (links) und maximaler Abstand* u_m*, den ein mobiles Objekt von seiner aktuellen Verbindung haben darf (rechts).*

des mobilen Objekts und allen von B abgehenden Verbindungen errechnet und die mit der kürzesten Entfernung als neue aktuelle Verbindung gewählt. Die Auswahl der neuen Verbindung wird durch dieses Vorgehen verzögert, da es wegen der Ungenauigkeit der Sensorinformationen nicht möglich ist, diese direkt am Punkt B zu bestimmen.

Rückwärtsverfolgung: Falls das Objekt seine aktuelle Verbindung verlässt ohne deren Ende B passiert zu haben, geht die Quelle davon aus, dass in einem vorherigen Schritt eine falsche Verbindung ausgewählt wurde und versucht dies durch Rückwärtsverfolgung zu korrigieren. Sie geht dann zurück zur zuletzt überquerten Kreuzung (wenn notwendig auch eine konfigurierbare Anzahl von Schritten weiter zurück) und überprüft die anderen ausgehenden Verbindungen auf entsprechende Weise.

Falls nach Vorwärts- oder Rückwärtsverfolgung keine passende Verbindung gefunden wurde, sendet die Quelle eine Aktualisierungsnachricht mit einem leeren Verbindungsfeld an die Senke. In diesem Fall wird das Koppelnavigationsprotokoll mit linearer Vorhersagefunktion als Rückfallposition verwendet. Die Quelle überprüft allerdings periodisch die Position des mobilen Objekts mit geeigneten Verbindungen der Straßenkarte (unter Verwendung des mehrdimensionalen Indexes), um zum kartenbasierten Verfahren zurückkehren zu können.

Die Vorhersagefunktion geht beim kartenbasierten Verfahren davon aus, dass das mobile Objekt weiterhin der zuletzt angegebenen Verbindung mit der dazugehörenden Geschwindigkeit folgt. Bei Erreichen einer Kreuzung wählt sie eine ausgehende Verbindung aus und nimmt an, dass es sich auf dieser mit gleichbleibender Geschwindigkeit weiterbewegt. In unserer Implementierung wird die Verbindung ausgewählt, deren Richtung sich am wenigsten von der der vorherigen Verbindung unterscheidet. Idealerweise würde die Vorhersagefunktion dabei die am meis-

ten benutzte Abzweigung (z. B. der Hauptstraße folgend) auswählen, diese Information konnte jedoch nicht auf einfache Weise aus der uns zur Verfügung stehenden Straßenkarte entnommen werden. In unseren Simulationen hat sich die von uns verwendete Alternative als eine gute Annäherung herausgestellt.

Abbildung 6-11 zeigt die Nachrichten zur Positionsaktualisierung, die mit dem beschriebenen kartenbasierten Koppelnavigationsprotokoll für das ebenfalls in Abbildung 6-8 verwendete Szenario erzeugt werden. Es ist deutlich zu erkennen, dass die Fähigkeit des kartenbasierten Verfahrens, dem Verlauf einer Straße zu folgen, im Vergleich zu dem einfacheren Verfahren mit linearer Vorhersagefunktion zu einer Reduzierung der Aktualisierungsnachrichten führt. Ein

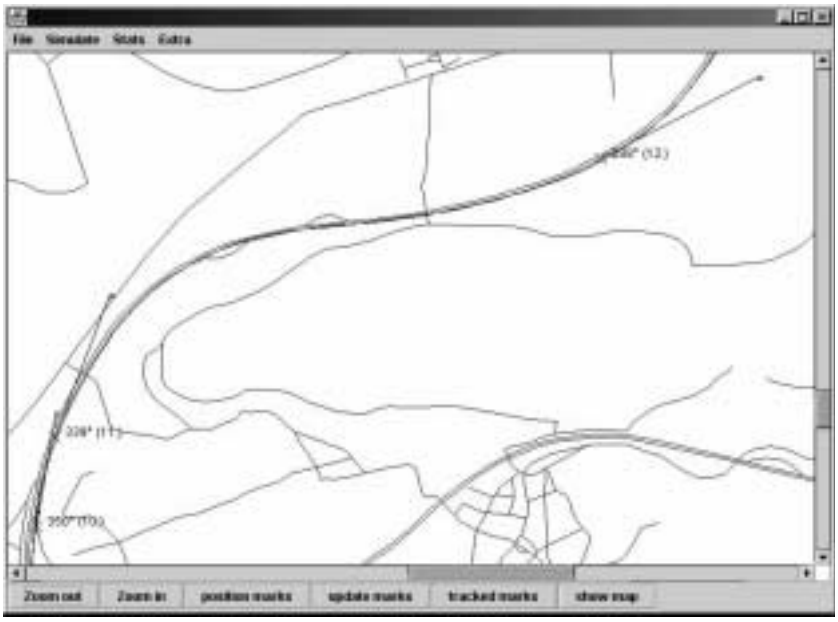


Abbildung 6-11. Beispiel für die Positionsaktualisierungen bei einem kartenbasierten Koppelnavigationsprotokoll für denselben Weg wie in Abbildung 6-8.

ausführlichere Betrachtung der Effizienz des kartenbasierten Koppelnavigationsprotokolls ist durch die Simulationen in Abschnitt 6.5.3 gegeben.

6.4 Analytische Betrachtung

Um einen besseren Vergleich zwischen den einzelnen Protokollen hinsichtlich ihrer Effizienz und Effektivität zu ermöglichen, sollen diese im folgenden Abschnitt analytisch betrachtet werden. Die erzielten Ergebnisse werden im darauffolgenden Kapitel zusätzlich mittels Simulatio-

nen, die auf den mit einem GPS-Empfänger aufgezeichneten Bewegungen realer mobiler Objekte (GPS-Protokolle) beruhen, verifiziert (in Abschnitt 6.5).

Bei den analytischen Betrachtungen werden jeweils die Anzahl *m* der pro Zeiteinheit übertragenen Nachrichten, bei denen es sich sowohl um Positionsaktualisierungen als auch um Positionsabfragen handeln kann[3], und die resultierende Genauigkeit ermittelt. Für die Genauigkeit wurde bisher immer die maximale Abweichung, gegeben durch den maximalen Abstand zwischen der auf der Senke verfügbaren Position und der wirklichen Position des mobilen Objekts $\hat{d}$, betrachtet, die z. B. wichtig ist, wenn bestimmt werden soll, in welchem Raum sich ein mobiles Objekt aufhält. Andere Anwendungen, die z. B. die Positionen mobiler Objekte auf einer Karte anzeigen, sind dagegen eher an der durchschnittlichen Abweichung $\bar{d}$ interessiert.

Bei der folgenden Analyse wird die Verzögerung, die für die Übermittlung der Nachrichten benötigt wird und die zu einer weiteren Abweichung bei den Positionsinformationen führen würde, der Einfachheit halber vernachlässigt. Da sie bei allen Nachrichten im gleichen Maße auftritt, hat sie wenig Einfluss auf den Vergleich der Protokolle. Weitere wichtige Einschränkungen, die im Folgenden nicht gesondert erwähnt werden, sind, dass die Genauigkeit der lokalen Kopie auf der Senke natürlich nicht genauer sein kann als die des Positionierungssensors ($u_s \geq u_p$) und dass es für eine Anwendung nicht sinnvoll ist, genauere Informationen anzufordern ($u_q \geq u_p$). Weiterhin wird angenommen, dass die Abweichung des Positionierungssensors im durchschnittlichen Fall halb so groß ist wie die maximale Abweichung ($u_p/2$). Da dies allerdings in hohem Maße vom jeweiligen Sensorsystem abhängt, kann es sich hierbei nur um eine Näherung handeln. Oft, wie z. B. bei GPS, ist die durchschnittliche Genauigkeit deutlich besser (d. h. $\bar{u}_p < u_p/2$), wodurch sich bei Berechnung der durchschnittlichen Genauigkeit eine Abschätzung nach oben ergibt.

In Anhang C sind alle in der Analyse verwendeten Variablen nochmals zusammengefasst.

6.4.1 Anfragende Protokolle

Beim *einfachen anfragenden Protokoll*, das hier wie gesagt nur zu Vergleichszwecken betrachtet werden soll, ist die Anzahl der übertragenen Nachrichten gleich der Anzahl der Anfragen *q*, da jede Anfrage an die Quelle weitergeleitet wird.

3. Im Folgenden wird der Einfachheit halber angenommen, dass eine Positionsaktualisierung denselben Aufwand verursacht wie eine Positionsabfrage. Sollte dies nicht der Fall sein, ist es einfach möglich, die beiden Komponenten entsprechend zu gewichten.

$$m = q \tag{6-1}$$

Wenn wir die Verzögerung bei der Nachrichtenübertragung vernachlässigen, hat die der Anwendung präsentierte Positionsinformation dieselbe Genauigkeit wie auf der Senke.

$$\hat{d} = u_p \qquad \bar{d} = \frac{u_p}{2} \tag{6-2}$$

Das 2 *anfragende Protokoll mit Zwischenspeichern* der Positionsinformationen benutzt Gleichung 6-3 (abgeleitet aus Gleichung 4-2), um die Genauigkeit der auf dem Server gespeicherten Informationen bei einer angenommenen Geschwindigkeit v_{asd} des mobilen Objekts abzuschätzen.

$$u(t) = v_{asd}(t - t_p) + u_p \tag{6-3}$$

Falls der so ermittelte Wert noch der geforderten Genauigkeit genügt ($u(t) < u_q$), wird direkt der Inhalt der gespeicherten Kopie zurückgegeben. Bei einem pessimistischen Ansatz wird v_{asd} wie in Gleichung 4-2 durch die Maximalgeschwindigkeit $\hat{v}$ des mobilen Objekts ersetzt, bei einem optimistischen Ansatz durch dessen durchschnittliche Geschwindigkeit $\bar{v}$. Im Vergleich zu dem einfachen Protokoll entfallen hier alle Nachrichten, bei denen die entsprechende Anfrage in das Zeitintervall fällt, in dem die zuletzt übertragene Information noch gültig ist, $\Delta t = (u_q - u_p) / v_{asd}$ (durch Umformung von Gleichung 6-3). Die resultierende Anzahl von benötigten Aktualisierungsnachrichten ist daher die Anzahl von Anfragen multipliziert mit dem Anteil an Anfragen, die nicht aus der lokalen Kopie bedient werden können, nämlich: $1 / (q\Delta t + 1) = 1 / (q \cdot (u_q - u_p) / v_{asd} + 1)$.

$$m = \frac{q}{q \cdot (u_q - u_p) / v_{asd} + 1} \tag{6-4}$$

Die maximale Abweichung ergibt sich durch den Zeitraum Δt, in dem eine lokale Kopie gültig ist, multipliziert mit der maximalen Geschwindigkeit des mobilen Objekts. Zu diesem Wert muss noch die initiale Ungenauigkeit der Sensorinformationen addiert werden.

Die durchschnittliche Abweichung setzt sich zusammen aus der Hälfte der Abweichung des Positionierungssensors, falls die Informationen bei der Quelle abgefragt werden müssen, und der halben Wegstrecke, die das mobile Objekt bei durchschnittlicher Geschwindigkeit in der Zeit

zurücklegt, in der die lokale Kopie gültig ist, im anderen Fall. In Gleichung 6-5 ist die entsprechende Formel vereinfacht dargestellt.

$$\hat{d} = \frac{u_q - u_p}{v_{asd}} \cdot \hat{v} + u_p \qquad \bar{d} = \frac{(u_q - u_p) / v_{asd} \cdot \bar{v} / 2}{v_{asd} / (u_q - u_p) / q + 1} + \frac{u_p}{2} \tag{6-5}$$

Wenn die für das mobile Objekt angenommene Geschwindigkeit v_{asd} dessen maximaler Geschwindigkeit $\hat{v}$ entspricht, ist eine Abweichung kleiner u_q für die Information, die einer anfragenden Anwendung übergeben wird, garantiert (durch Einsetzen in Gleichung 6-5). Das Protokoll kann also in diesem Fall immer die von der Anwendung geforderte Genauigkeit einhalten.

6.4.2 Berichtende Protokolle

Da das *zeitbasierte Protokoll* periodisch Nachrichten zur Positionsaktualisierung versendet, hängt die Anzahl der übertragenen Nachrichten nur von dem entsprechenden Zeitschwellwert T ab. Um eine bestimmte Genauigkeit u_s auf der Senke zu garantieren, darf T höchstens dem Zeitabschnitt entsprechen, den das mobile Objekt benötigt, um die durch u_s vorgegebene Entfernung (abzüglich der durch das Sensorsystem bedingten Genauigkeit) bei maximaler Geschwindigkeit zurückzulegen: $T = (u_s - u_p) / \hat{v}$.

$$m = \frac{1}{T} = \frac{\hat{v}}{u_s - u_p} \tag{6-6}$$

Die maximale Abweichung ist entsprechend dazu die Wegstrecke, die das mobile Endgerät bei maximaler Geschwindigkeit innerhalb des Zeitintervalls T zurückgelegt haben kann zuzüglich der durch das Sensorsystem bedingten Ungenauigkeit. Die durchschnittliche Abweichung erhält man, indem man die durchschnittliche Geschwindigkeit mit der Hälfte des Zeitintervalls T, die ja im Durchschnitt seit der letzten Positionsaktualisierung vergangen ist, multipliziert und die durchschnittliche Genauigkeit des Sensorsystems addiert.

$$\hat{d} = T \cdot \hat{v} + u_p = u_s \qquad \bar{d} = \frac{T}{2} \cdot \bar{v} + \frac{u_p}{2} = \frac{u_s - u_p}{2} \cdot \frac{\bar{v}}{\hat{v}} + \frac{u_p}{2} \tag{6-7}$$

Beim *entfernungsbasierten Protokoll* ist die Anzahl der übertragenen Nachrichten ebenfalls unabhängig von der Anfragerate. Um eine geforderte Abweichung u_s zu erreichen, muss, wie be-

reits gesagt, der Schwellwert für die Entfernung D auf u_s - u_p gesetzt werden. Die durchschnittliche Anzahl an übertragenen Nachrichten pro Zeiteinheit ergibt sich dann aus dem Inversen der Zeitdauer, die das mobile Objekt benötigt um eine Wegstrecke von D bei durchschnittlicher Geschwindigkeit zurückzulegen. Dabei wird angenommen, dass der Entfernungsschwellwert klein gegenüber dem Abstand zwischen größeren Richtungsänderungen ist.

$$m = \frac{\bar{v}}{D} = \frac{\bar{v}}{u_s - u_p} \qquad (6\text{-}8)$$

Da beim entfernungsbasierten Protokoll die Quelle die Notwendigkeit einer Positionsaktualisierung anhand der tatsächlichen Position des mobilen Objekts bestimmt, wird genau die bei der Senke geforderte Abweichung u_s erreicht, wenn keine Nachrichten verloren gehen (siehe Abschnitt 6.2.4). Die durchschnittliche Abweichung ergibt sich aus der Entfernung, die das mobile Objekt in der halben Zeit des Aktualisierungsintervalls zurücklegt, und ist halb so groß.

$$\hat{d} = u_s \qquad \bar{d} = \frac{u_s}{2} \qquad (6\text{-}9)$$

Das Verhalten der *Koppelnavigationsprotokolle* entspricht dem des entfernungsbasierten Protokolls, die Anzahl der versendeten Nachrichten ist daher auch unabhängig von der Anfragerate q. Die Anzahl von Nachrichten hängt allerdings von vielen Faktoren ab, so z. B. sehr stark von der Bewegungscharakteristik des mobilen Objekts und der verwendeten Vorhersagefunktion, und ist schwer analytisch zu erfassen. In vielen Fällen ist die benötigte Anzahl an Nachrichten deutlich kleiner als die beim entfernungsbasierten Protokoll, kann bei einer ungünstigen Vorhersagefunktion allerdings auch darüber liegen. Die Variable $gain_{dr}$ soll im Folgenden die Güte einer Vorhersagefunktion beschreiben. Sie ergibt sich aus dem Verhältnis zwischen der Anzahl der Aktualisierungsnachrichten, die bei einer bestimmten Vorhersagefunktion pro Zeiteinheit durchschnittlich benötigt werden, gegenüber der Zahl von Aktualisierungsnachrichten, die ohne Vorhersagefunktion (*pred(p, i, param, t) = p*) anfallen. Letzteres entspricht einem entfernungsbasierten berichtenden Protokoll, für das damit $gain_{dr} = 1$ gilt. Wenn $gain_{dr}$ die Einsparungen durch ein Koppelnavigationsprotokoll darstellt, die in Abschnitt 6.5 durch Simulationen bestimmt werden, ergibt sich die Anzahl der Nachrichten wie in Gleichung 6-10 gezeigt. Koppelnavigationsprotokolle werden ausführlich in unseren Simulationen diskutiert (siehe Abschnitt 6.5.3), ihre analytische Betrachtung soll daher hier nicht weiter ausgeführt werden.

$$m = gain_{dr} \cdot \frac{\bar{v}}{u_s - u_p} \tag{6-10}$$

Da für die Aktualisierung der Positionsinformationen genau dieselben Kriterien angewendet werden wie beim entfernungsbasierten Protokoll, ergibt sich die maximale und durchschnittliche Abweichung entsprechend:

$$\hat{d} = u_s \qquad \bar{d} = \frac{u_s}{2} \tag{6-11}$$

6.4.3 Kombiniertes Protokoll

Das kombinierte Protokoll setzt sich zusammen aus Elementen des entfernungsbasierten berichtenden und des pessimistischen anfragenden Protokolls mit Zwischenspeichern, was sich auch in seinem Verhalten widerspiegelt. Die versendeten Nachrichten sind hier sowohl Positionsaktualisierungen, die notwendig sind, um die Kopie auf der Senke aktuell zu halten, und Positionsabfragen, mit denen die Senke den aktuellen Zustand der Quelle abfragt, falls ihre eigene Kopie nicht genau genug ist. Die Zahl der Aktualisierungsnachrichten entspricht der im entfernungsbasierten berichtenden Protokoll. Bei der Anzahl von Positionsabfragen verringert sich die des anfragenden Protokolls um die Wahrscheinlichkeit, dass die geforderte Genauigkeit geringer als die der lokalen Kopie ist, $P(u'_q < u_s)$. Die entsprechende Wahrscheinlichkeitsverteilung hängt von der Art und Weise ab, wie die Positionsinformationen von den Anwendungen angefragt und genutzt werden.

$$m = \frac{\bar{v}}{u_s - u_p} + \frac{P(u'_q < u_s) \cdot q}{P(u'_q < u_s) \cdot q \cdot (u_q - u_p) / \hat{v} + 1} \tag{6-12}$$

Da das kombinierte Protokoll die Positionsinformationen jedesmal von der Quelle abfragt, wenn sie nicht in der geforderten Genauigkeit vorliegen, ist die durchschnittliche maximale Ungenauigkeit gleich der in den Anfragen geforderten. Die durchschnittliche Genauigkeit setzt sich wiederum aus den durchschnittlichen Genauigkeiten bei den entsprechenden anfragenden und berichtenden Protokollen zusammen.

$$\hat{d} = u_q \qquad \bar{d} = P(u'_q \geq u_s) \cdot \frac{u_s}{2} + P(u'_q < u_s) \cdot \left(\frac{(u_q - u_p) / \hat{v} \cdot \bar{v} / 2}{\hat{v} / (u_q - u_p) / q + 1} + \frac{u_p}{2} \right) \quad (6\text{-}13)$$

6.4.4 Diskussion

Ausgehend von den Analysen aus dem vorherigen Abschnitt, sollen in diesem Abschnitt die Eigenschaften der unterschiedlichen Protokolle diskutiert und verglichen werden. Koppelnavigationsprotokolle werden hier allerdings nur am Rande betrachtet, da sich Abschnitt 6.5 ausführlich mit diesen beschäftigt. Wie aus den bisherigen Betrachtungen deutlich geworden ist, hängt das Verhalten der Protokolle in großem Maße von den Bewegungscharakteristika der mobilen Objekte ab. Wir betrachten hier daher vier typische Klassen von Objekten, die unterschiedliche durchschnittliche und maximale Geschwindigkeiten aufweisen. Die Werte für die durchschnittliche Geschwindigkeit sind dabei aus den GPS-Protokollen für reale Objekte entnommen, die auch für die Simulation der Protokolle in Abschnitt 6.5 verwendet werden.

- o_1: Ein Fußgänger mit durchschnittlicher Geschwindigkeit $\bar{v}$ = 1,3 m/s (4,7 km/h) und angenommener maximaler Geschwindigkeit $\hat{v}$ = 2,8 m/s (≈ 10 km/h).
- o_2: Ein Auto im Stadtverkehr mit durchschnittlicher Geschwindigkeit $\bar{v}$ = 10 m/s (36 km/h) und angenommener maximaler Geschwindigkeit $\hat{v}$ = 22,5 m/s (≈ 80 km/h).
- o_3: Ein Auto auf einer Landstraße mit durchschnittlicher Geschwindigkeit $\bar{v}$ = 17 m/s (61 km/h) und angenommener maximaler Geschwindigkeit $\hat{v}$ = 44,5 m/s (≈ 160 km/h).
- o_4: Ein Auto auf einer Autobahn mit durchschnittlicher Geschwindigkeit $\bar{v}$ = 29 m/s (104 km/h) und angenommener maximaler Geschwindigkeit $\hat{v}$ = 55 m/s (≈ 200 km/h).

Als Sensorsystem wurde jeweils ein DGPS-Empfänger mit einer Genauigkeit von 5 Metern (u_p = 5 m) angenommen, den wir auch für das Erzeugen der GPS-Protokolle verwendet haben. Die geforderte Genauigkeit ist einheitlich auf durchschnittlich 100 Meter (u_q = 100 m) festgelegt.

Zuerst betrachten wir die Effizienz der Protokolle, d. h. die Gesamtzahl der Nachrichten, die zwischen Quelle und Senke ausgetauscht werden. Für die anfragenden und berichtenden Protokolle ist die Anzahl der Nachrichten abhängig von der Zahl der Anfragen pro Sekunde und für die unterschiedlichen Klassen mobiler Objekte in Abbildung 6-12 dargestellt.

Die Anzahl der benötigten Nachrichten steigt dabei generell mit einer größeren durchschnittlichen und maximalen Geschwindigkeit des mobilen Objekts. Insgesamt benötigt das entfernungsbasierte berichtende Protokoll weniger Nachrichten als das zeitbasierte und das optimis-

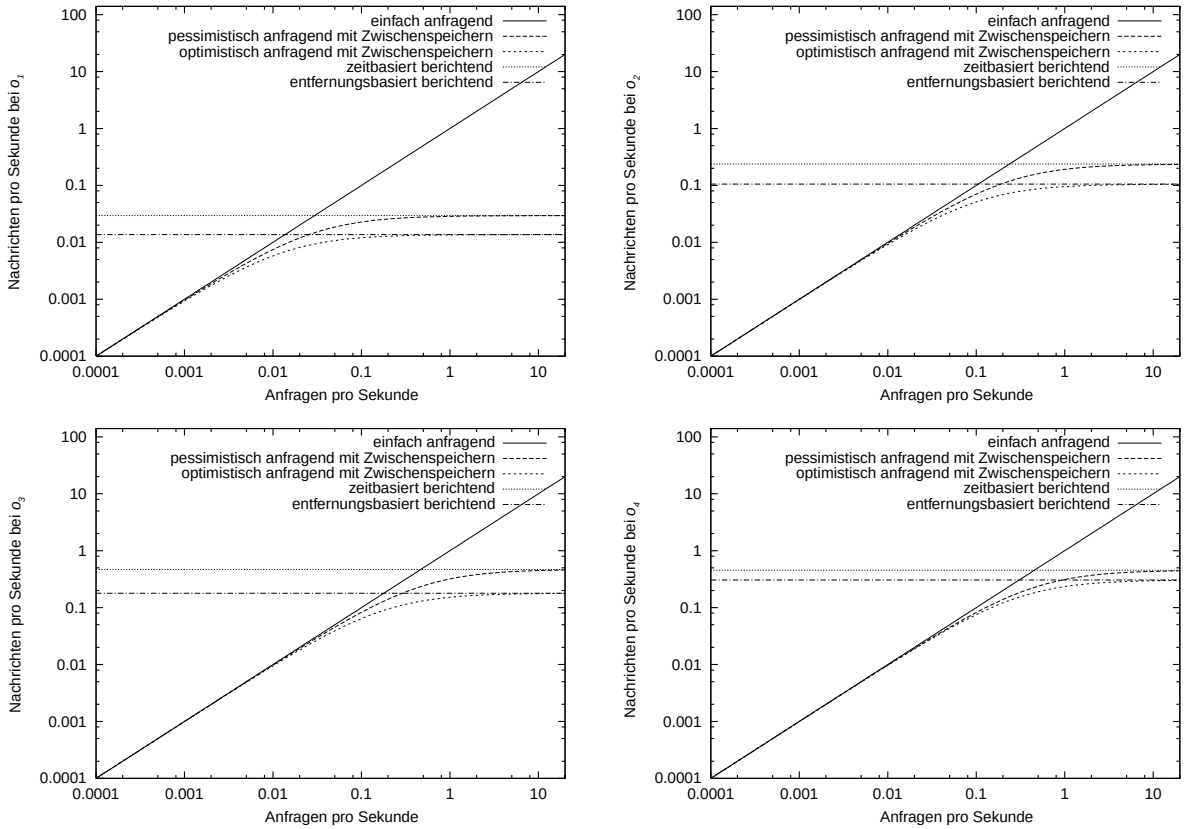


Abbildung 6-12. Ergebnis der Analyse: Vergleich der Anzahl von Nachrichten m, *die bei Verwendung der verschiedenen anfragenden und berichtenden Protokolle ausgetauscht werden. Von oben links nach unten rechts für die Objektklassen* o_1 *bis* o_4.

tische anfragende Protokoll mit Zwischenspeichern weniger als das pessimistische. Die Unterschiede sind um so deutlicher, je größer das Verhältnis zwischen maximaler und durchschnittlicher Geschwindigkeit ist (z. B. für Objekt o_3). Obwohl das optimistische anfragende Protokoll generell am wenigsten Nachrichten benötigt, kommt es für die meisten Anwendungen nicht in Frage, da die Genauigkeit der zurückgelieferten Positionsinformationen weit schlechter als die geforderte Genauigkeit sein kann (siehe unten).

Während die Nachrichtenrate bei den berichtenden Protokollen unabhängig von der Anzahl der Anfragen ist, nimmt sie bei anfragenden Protokollen für höhere Anfrageraten zu. Durch Gleichsetzen von Gleichung 6-4 und Gleichung 6-8 lässt sich der in Abbildung 6-12 ersichtliche Punkt bezüglich der Anfragerate berechnen, an dem z. B. ein entfernungsbasiertes berichtendes Protokoll, weniger Nachrichten benötigt als ein vergleichbares pessimistisches anfragendes Proto-

koll mit Zwischenspeichern. Für den Vergleich wurden diese beiden Protokolle gewählt, da beide eine vorgegebene Genauigkeit der Informationen garantieren können.

$$q > \frac{1}{(u_s - u_p)/\ \bar{v} - (u_q - u_p)/\ \hat{v}} \qquad (6\text{-}14)$$

Im Falle einer fest vorgegebenen geforderten Genauigkeit von $u_s = u_q$ und einer im Vergleich dazu geringen Ungenauigkeit u_p der Sensorinformationen, kann die Gleichung vereinfacht wie folgt dargestellt werden:

$$q > \frac{\hat{v} \cdot\ \bar{v}}{u_q \cdot\ (\hat{v} - \bar{v})} \qquad (6\text{-}15)$$

Wenn es sich um ein mobiles Objekt mit sporadischen Bewegungen handelt, und daher $\bar{v}$ im Vergleich zu $\hat{v}$ klein ist, ist damit das entfernungsbasierte berichtende Protokoll schon für geringe Anfrageraten besser als das anfragende.

Der andere wichtige Aspekt bei der Bewertung von Aktualisierungsprotokollen ist deren Effektivität, d. h. bis zu welchem Grad die Senke die in einer Anfrage geforderte Genauigkeit erfüllen kann. Die maximale und durchschnittliche Abweichung ($\hat{d}$ und $\bar{d}$) zwischen zurückgelieferter und tatsächlicher Position des mobilen Objekts ist für die zwei anfragenden und die zwei berichtenden Protokolle in Abbildung 6-13 dargestellt. Die Abweichung ist dort abhängig von dem Geschwindigkeitsverhältnis zwischen durchschnittlicher und maximaler Geschwindigkeit des mobilen Objekts ($\bar{v}/\ \hat{v}$) angegeben, welches einen guten Eindruck von dessen Mobilitätscharakteristik vermittelt. Ein mobiles Objekt mit einem niedrigeren Verhältnis bewegt sich mehr sporadisch (z. B. o_3 im Vergleich zu o_4), während ein Objekt mit einem Verhältnis von nahezu 1 sich sehr gleichförmig bewegt. Für die Darstellung sind wir von einer Anfragerate q von 0,1 Anfragen pro Sekunde ausgegangen.

Abbildung 6-13 zeigt, dass bis auf das optimistische anfragende Protokoll mit Zwischenspeichern alle Protokolle die geforderte Genauigkeit einhalten können. Dieses Protokoll ist im Besonderen nicht geeignet für mobile Objekte mit ungleichmäßigem Bewegungsverhalten, da die Abweichung bei einem niedrigen Geschwindigkeitsverhältnis sehr hoch wird. Unter den anderen Protokollen hat speziell das pessimistische anfragende Protokoll mit Zwischenspeichern eine durchgängig niedrigere durchschnittliche Abweichung, vor allem für Objekte mit einem geringeren Geschwindigkeitsverhältnis, als das vergleichbare entfernungsbasierte Protokoll.

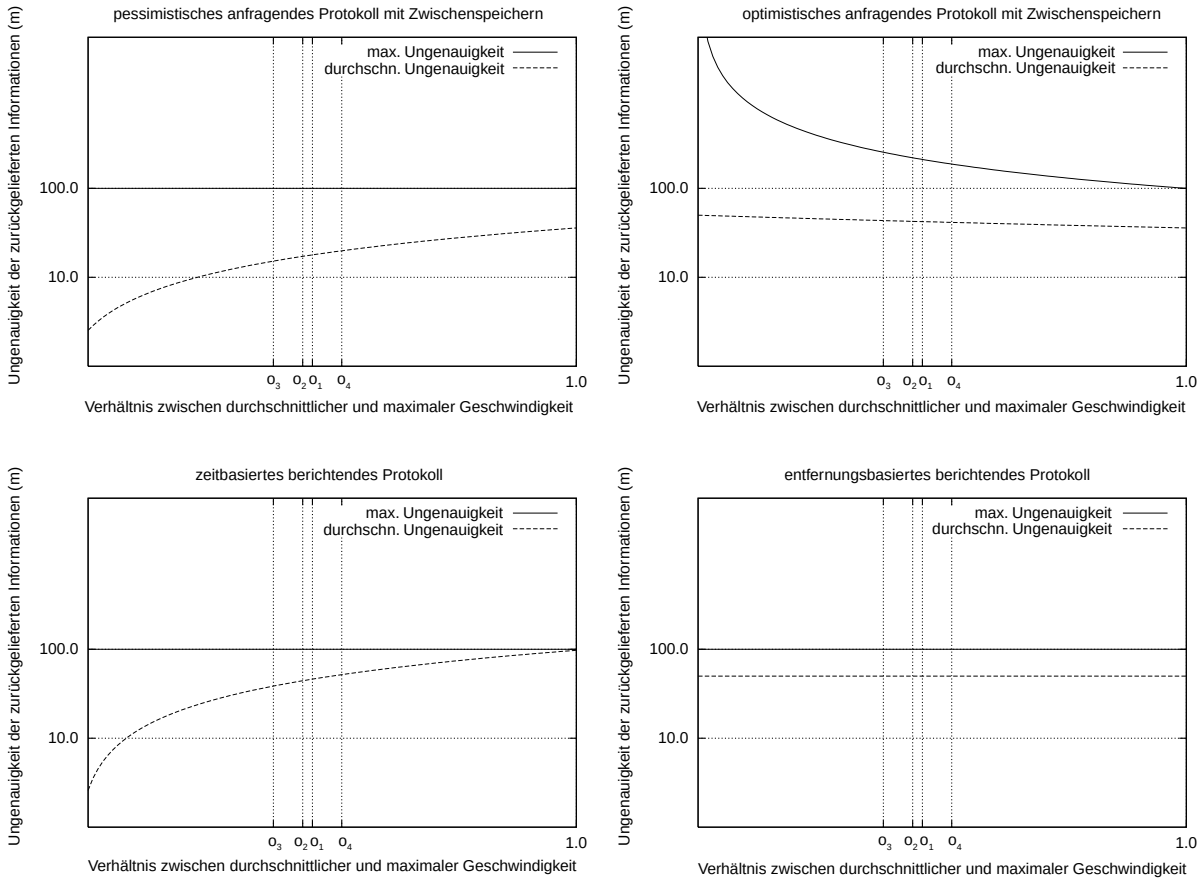


Abbildung 6-13. Ergebnis der Analyse: Maximale und durchschnittliche Abweichung zwischen der bei einer Anfrage zurückgelieferten Positionsinformation und der aktuellen Position des mobilen Objekts (logarithmische Skala) für das pessimistische und optimistische anfragende Protokoll sowie das zeit- und das entfernungsbasierte berichtende Protokoll abhängig vom Verhältnis zwischen durchschnittlicher und maximaler Geschwindigkeit (lineare Skala).

Wie bereits aus dem vorherigen Abschnitt deutlich geworden, hängt die Effizienz des kombinierten Protokolls sowohl von der für die lokale Kopie auf der Senke eingestellten Genauigkeit u_s als auch von der Anfragerate q ab. In Abbildung 6-14 (a) ist die Anzahl der übertragenen Nachrichten für Objekt o_4 abhängig von u_s und q gezeigt. Für die in den Anfragen geforderte Genauigkeit haben wir eine Gaussverteilung mit einem Erwartungswert von ebenfalls u_q = 100 m und einer Standardabweichung von 20 m angenommen. Eine entsprechende Gaussverteilung wird in den Naturwissenschaften oft zur Beschreibung reellwertiger Parameter verwendet. Alle weiteren Parameter haben die gleichen Werte wie bisher.

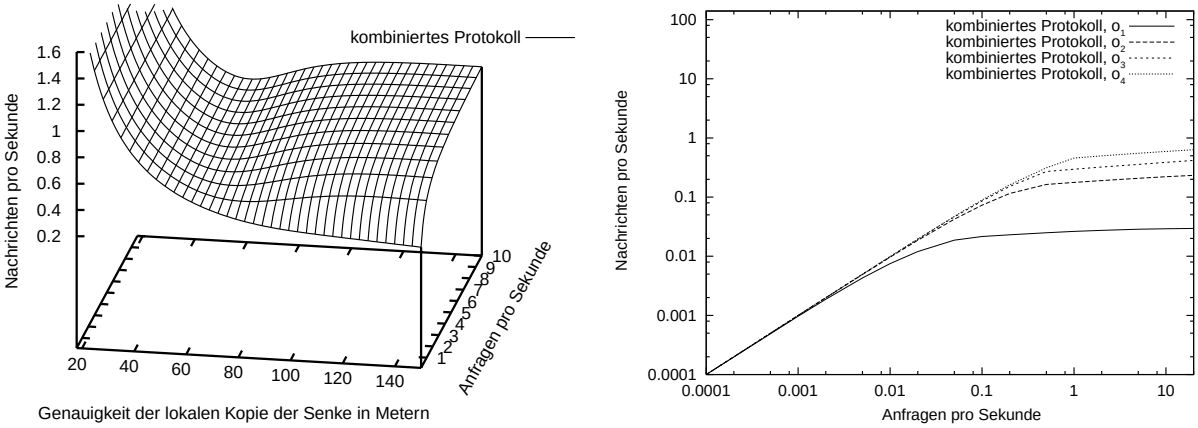


Abbildung 6-14. Ergebnis der Analyse: Anzahl der Nachrichten, die bei Verwendung des kombinierten Protokolls (a) für Objekt o_4 *abhängig von der Genauigkeit der Kopie der Senke* u_s *sowie der Anzahl von Anfragen pro Sekunde* q *und (b) für die Objekte* o_1 *bis* o_4 *bei optimaler Wahl von* u_s *abhängig von der Zahl der Anfragen benötigt werden.*

Abbildung 6-14 (a) zeigt, dass es für höhere Anfrageraten (in diesem Fall $q \geq 1$) einen optimalen Wert für u_s gibt, an dem die Anzahl der übertragenen Nachrichten am geringsten ist. Im Gegensatz zur Senke, welche die Notwendigkeit einer Positionsaktualisierung über die maximale Geschwindigkeit des mobilen Objekts abschätzen muss, kann die Quelle dazu dessen tatsächliche Geschwindigkeit verwenden, wodurch bei höheren Anfrageraten weniger Nachrichten benötigt werden (siehe auch Abbildung 6-12). Es ist daher sinnvoll, auf der Senke eine Kopie der Positionsinformation mit der Genauigkeit u_s durch einen berichtenden Protokollanteil von der Quelle aktualisieren zu lassen, wenn genügend Anfragen die Positionsinformation mit einer entsprechend hohen Genauigkeit benötigen. Für kleinere Anfrageraten werden hingegen weniger Nachrichten benötigt, wenn keine Kopie auf der Senke aktualisiert wird ($u_s = \infty$), da dort der durch Zusatzaufwand, der durch die Aktualisierung der Kopie entsteht, nicht durch den geringeren Bearbeitungsaufwand für Anfragen wettgemacht wird. Anhand von Gleichung 6-12 kann ein optimaler Wert für u_s abhängig von der Anfragerate q ermittelt werden. Abbildung 6-14 (b) zeigt schließlich die für die unterschiedlichen Klassen von Objekten o_1 bis o_4 benötigten Nachrichten pro Sekunde bei einer optimalen Wahl von u_s.

Wie erwartet verbindet das kombinierte Protokoll die positiven Eigenschaften der einfacheren anfragenden und berichtenden Protokolle. Obwohl es für hohe Anfrageraten etwas mehr Nachrichten benötigt als das entfernungsbasierte berichtende Protokoll, ist es wie die anfragenden Protokolle wesentlich effizienter für niedrige Anfrageraten. Anders als das berichtende Proto-

koll kann es darüber hinaus auch Anfragen beantworten, in denen eine Genauigkeit höher als u_s gefordert ist.

Zusammenfassend ist das kombinierte Protokoll für die meisten Anwendungsfälle am Besten geeignet, besonders wenn ein Protokoll gefordert ist, das in verschiedenen Umgebungen eingesetzt werden oder sich sogar auf veränderliche Bedingungen einstellen soll. Der Nachteil des kombinierten Protokolls ist, dass eine geeignete Einstellung für die Genauigkeit der Kopie auf der Senke u_s gefunden werden muss. Von den anfragenden und berichtenden Protokollen sind das pessimistische anfragende Protokoll mit Zwischenspeichern und das entfernungsbasierte berichtende Protokoll zu bevorzugen, da sie eine obere Grenze für die Genauigkeit der zurückgelieferten Positionsinformationen mit einer guten Effizienz kombinieren. Von diesen beiden ist das letztere deutlich besser bei Objekten mit ungleichmäßigem Bewegungsverhalten und bei höheren Anfrageraten. Es erlaubt allerdings nicht die geforderte Genauigkeit je Anfrage anzugeben.

6.5 Simulation

Um die Ergebnisse unserer analytischen Betrachtungen in Abschnitt 6.4 zu verifizieren und um die Eigenschaften der im vorangegangenen Abschnitt besprochenen Koppelnavigationsprotokolle, die analytisch nicht im Detail untersucht werden konnten, zu betrachten, wurde eine Reihe von Simulationen auf der Basis von GPS-Protokollen der realen Bewegungen mobiler Objekte durchgeführt. In diesem Abschnitt werden zuerst die dazu verwendeten GPS-Protokolle und die Simulationsumgebung beschrieben, bevor die Ergebnisse der Simulationen besprochen werden.

6.5.1 Simulationsumgebung

Als Grundlage für die Simulationen wurden im Rahmen dieser Arbeit eine größere Zahl von GPS-Bewegungsprotokollen für die Bewegungen von Fußgängern und Kraftfahrzeugen in der Umgebung von Stuttgart erstellt. Die Bewegungsprotokolle lassen sich entsprechend den Betrachtungen in Abschnitt 6.4 in vier Klassen mit unterschiedlichen Charakteristika einteilen:

- o_1, ein Fußgänger,
- o_2, ein Auto im Stadtverkehr,
- o_3, ein Auto auf einer Landstraße und
- o_4, ein Auto auf einer Autobahn.

In Tabelle 6-2 sind die Eigenschaften der verschiedenen Klassen von GPS-Bewegungsprotokollen zusammengefasst[4]. Die Bewegungsprotokolle wurden mit einem DGPS-Empfänger (Garmin 25, Empfang des D-Signals über den ALF-Dienst der Deutschen Telekom) erzeugt, der eine Genauigkeit von 2 bis 5 m aufweist. Die von diesem ausgegebene Position wurde dazu sekündlich in einer Datei protokolliert.

	Gesamtlänge	Gesamtdauer	durchschn. Geschw.	max. Geschw.
o_1, Fußgänger	10 km	2:08 h	4,6 km/h	7,2 km/h
o_2, Auto in Stadtverkehr	89 km	2:25 h	34 km/h	65 km/h
o_3, Auto auf Landstraße	99 km	1:39 h	60 km/h	116 km/h
o_3, Auto auf Autobahn	163 km	1:35 h	103 km/h	155 km/h

Tabelle 6-2. Charakteristika der bei der Simulation verwendeten GPS-Protokolle für verschiedene Klassen mobiler Objekte.

Für die Durchführung der Simulationen haben wir ein Simulationswerkzeug implementiert, welches für ein mobiles Objekt eine Quelle und eine Senke der Positionsinformationen realisiert. Die von der Quelle erzeugten Positionsinformationen werden dabei aus einem vorgegebenen GPS-Protokoll ausgelesen. An der Senke werden in zufälligen, exponentialverteilten Zeitabständen die Positionsinformationen mit einer im Fall eines anfragenden oder kombinierten Protokolls zufällig gaussverteilten Genauigkeit ansonsten mit einer feststehenden Genauigkeit angefordert.

Die Exponentialverteilung, die hier zur Bestimmung der Zeitabstände zwischen aufeinanderfolgenden Positionsanfragen verwendet wird, beschreibt die Intervalle zwischen den Ereignissen eines Poisson-Prozesses. Ein Poisson-Prozess wird typischerweise verwendet, um die Ankunftszeiten von Anfragen an ein Rechnersystem zu beschreiben (siehe z. B. MATHAR & PFEIFER (1990)). Der Erwartungswert für die Exponentialverteilung entspricht in unserem Fall dem Inversen der in unseren Simulationen variierten Anfragerate q.

Bei den berichtenden Protokollen, bei denen die geforderte Genauigkeit nicht pro Anfrage angegeben werden kann, wurde für die Positionsinformationen eine feststehende Genauigkeit von

4. Wegen der Ungenauigkeit der Sensorinformationen kann aus einem GPS-Protokoll ein Wert für die maximale Geschwindigkeit nur näherungsweise bestimmt werden.

100 m verlangt. Bei den anfragenden und dem kombinierten Protokoll wird, wie bei der Analyse in Abschnitt 6.4.3, die geforderte Genauigkeit hingegen durch eine Gaussverteilung mit einem Erwartungswert von ebenfalls 100 m und einer Standardabweichung von 20 m beschrieben.

In diese Simulationsumgebung können die unterschiedlichen Protokollmodule eingesetzt werden, die steuern, wie die Quelle und Senke auf ankommende Positionsaktualisierungen und Anfragen reagieren. Wie in unserer Analyse wird beim Nachrichtenaustausch die Verzögerung nicht betrachtet. Das Simulationswerkzeug ist darüber hinaus in der Lage, Karteninformationen und Positionsaktualisierungen zu visualisieren (siehe Abschnitt 6.3) und wurde auch zur Überprüfung und Entwicklung der Protokolle eingesetzt.

6.5.2 Grundlegender Vergleich der Protokolle

Im Folgenden sind die Ergebnisse der durchgeführten Simulationen dargestellt. Aus Gründen der Übersichtlichkeit werden hier nur jeweils die wichtigsten Protokolle verglichen. Für weitere Simulationsergebnisse wird der interessierte Leser auf LEONHARDI & ROTHERMEL (2001) sowie NICU (2001) verwiesen.

Abbildung 6-15 zeigt die Anzahl der bei einer Simulation des pessimistischen anfragenden Protokolls mit Zwischenspeichern, des entfernungsbasierten und des kombinierten Protokolls sowie des Koppelnavigationsprotokolls mit linearer Vorhersage übertragenen Nachrichten zusammenfassend für die unterschiedlichen Klassen von GPS-Bewegungsprotokollen.

Insgesamt entsprechen die in Abbildung 6-15 gezeigten Simulationsergebnisse weitestgehend den Ergebnissen der analytischen Betrachtung. Lediglich beim anfragenden Protokoll mit Zwischenspeichern und dem kombinierten Protokoll ist die Anzahl der übertragenen Nachrichten etwas höher als vorhergesagt. Grund dafür ist, dass die Positionsinformationen wegen der begrenzten Abtastrate f_p des Sensorsystems (in unserem Fall f_p gleich 1 pro s) nicht genau dann bestimmt werden, wenn sie die Senke bei der Quelle abfragt, sondern dass diese bereits ein durchschnittliches Alter von $T_p/2 = 1/(2 \cdot f_p)$ haben. Die auf der Senke gespeicherte lokale Kopie muss daher um $T_p/2$ früher als angenommen aktualisiert werden, was sich bei höheren Anfrageraten bemerkbar macht. Das zum Vergleich gezeigte Koppelnavigationsprotokoll mit linearer Vorhersage erweist sich als bereits sehr viel versprechend und benötigt jeweils deutlich weniger Nachrichten als die vergleichbaren anderen Protokolle. Koppelnavigationsprotokolle werden daher im nächsten Abschnitt noch ausführlicher betrachtet.

Weitere Simulationen wurden durchgeführt, um die maximale und durchschnittliche Ungenauigkeit der zurückgelieferten Positionsinformationen beurteilen zu können. Die Ergebnisse die-

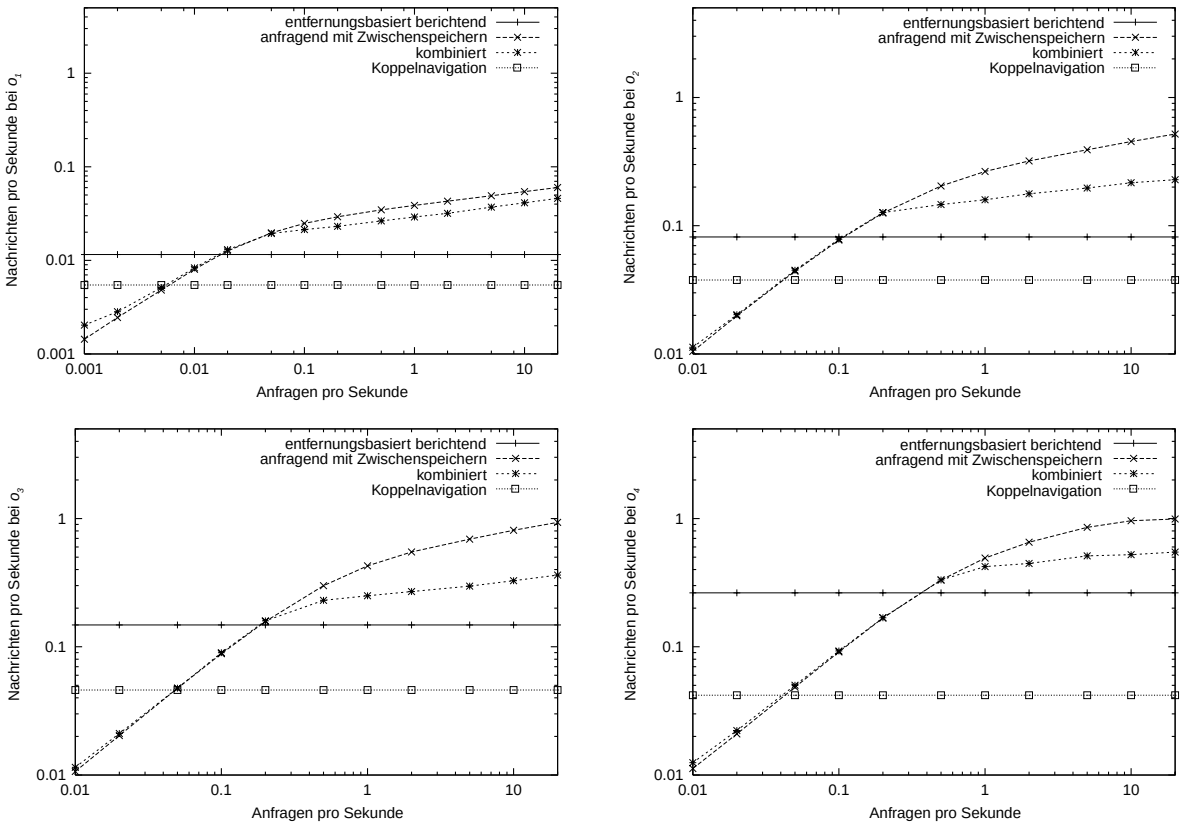


Abbildung 6-15. Simulationsergebnis: Vergleich der bei Simulation eines anfragenden Protokolls mit Zwischenspeichern, eines entfernungsbasierten berichtenden, eines kombinierten Protokolls sowie eines Koppelnavigationsprotokolls mit linearer Vorhersage resultierenden Anzahl von Nachrichten. Von oben links nach unten rechts für die Objektklassen o_1 bis o_4.

ser Simulationen sind für die vier betrachteten Klassen von mobilen Objekten, das pessimistische und optimistische anfragende Protokoll, das entfernungs- und zeitbasierte berichtende Protokoll sowie das kombinierte Protokoll in Tabelle 6-3 gezeigt. Da die Koppelnavigationsprotokolle dieselbe Aktualisierungsstrategie haben wie das entfernungsbasierte Protokoll, sind die entsprechenden Ergebnisse hier nicht gesondert aufgeführt. Die Anfragerate wurde dabei auf einen festen mittleren Wert von 0,1 pro Sekunde gesetzt, während die anderen Parameter dieselben Werte wie zuvor haben.

Weil wir für die maximalen Geschwindigkeit der mobilen Objekte eine obere Grenze je Objektklasse (wie in Abschnitt 6.4.4 beschrieben) angenommen haben (an Stelle der wirklichen maximalen Geschwindigkeit innerhalb der GPS-Bewegungsprotokolle (siehe Tabelle 6-2)), ist die maximale Ungenauigkeit bei dem pessimistischen anfragenden Protokoll mit Zwischenspei-

Obj.-klasse	Geschw.-verhältnis	pess. anfr. Protokoll mit Zwischensp.		optim. anfr. Protokoll mit Zwischensp.		entf.basiertes berichtendes Protkoll		zeitbasiertes berichtendes Protokoll		kombiniertes Protokoll	
		max	∅	max	∅	max	∅	max	∅	max	∅
o_1	0,46	64,21	19,20	126,73	39,59	99,23	54,29	60,67	24,09	64,92	19,41
o_2	0,44	66,63	8,37	178,99	24,67	98,74	48,35	83,13	19,18	66,66	8,39
o_3	0,38	41,66	6,00	141,77	17,61	99,00	45,47	92,55	24,25	44,05	6,03
o_4	0,53	42,92	6,27	94,09	11,21	94,01	40,68	89,54	35,06	41,13	6,26

Tabelle 6-3. Simulationsergebnis: Maximale und durchschnittliche Ungenauigkeit der zurückgelieferten Positionsinformationen für das pessimistische und optimistische anfragende Protokoll, das entfernungs- und zeitbasierte berichtende sowie das kombinierte Protokoll.

chern, dem zeitbasierten berichtenden Protokoll und dem kombinierten Protokoll deutlich geringer als die geforderte Genauigkeit u_s (entsprechend zu $v_{asd}/\ \hat{v}$). Ansonsten spiegeln die Werte die Ergebnisse der Analyse wider und nur das optimistisch anfragende Protokoll mit Zwischenspeichern kann die geforderte Genauigkeit von 100 m nicht bereitstellen. Im Vergleich zu der Zahl der übertragenen Nachrichten schwanken die Ergebnisse vor allem für die maximale Genauigkeit stark bei kurzfristigen Auslassungen bei den Sensorinformationen. Dies kann dazu führen, dass die Ungenauigkeit signifikant über 100 m ansteigt, wenn die Sensorinformationen nicht entsprechend vorverarbeitet werden.

6.5.3 Koppelnavigationsprotokolle

Da sich die Klasse der Koppelnavigationsprotokolle, die im vorherigen Abschnitt nur zu Vergleichszwecken präsentiert wurde, als sehr viel versprechend herausgestellt hat, wurden speziell im Hinblick auf diese weitere Simulationen durchgeführt.

In Abbildung 6-16 bis Abbildung 6-19 ist auf der linken Seite jeweils die Anzahl von Nachrichten pro Sekunde dargestellt, die das kartenbasierte Koppelnavigationsprotokoll, das Koppelnavigationsprotokoll mit linearer Vorhersage und das diesen von den Eigenschaften her entsprechende entfernungsbasierte berichtende Protokoll für die in Tabelle 6-2 beschriebenen Bewegungscharakteristika verursachen. Auf der rechten Seite sind die Werte für einen besseren Vergleich noch einmal als prozentuale Anteile verglichen mit dem entfernungsbasierten Protokoll gezeigt. In den Abbildungen wurde die geforderte Genauigkeit von 20 m bis 500 m für Fahrzeuge und von 20 m bis 250 m für einen Fußgänger variiert. Wie beim entfernungsbasierten Protokoll hängen die Ergebnisse der Koppelnavigationsprotokolle nicht von der Anfragerate ab.

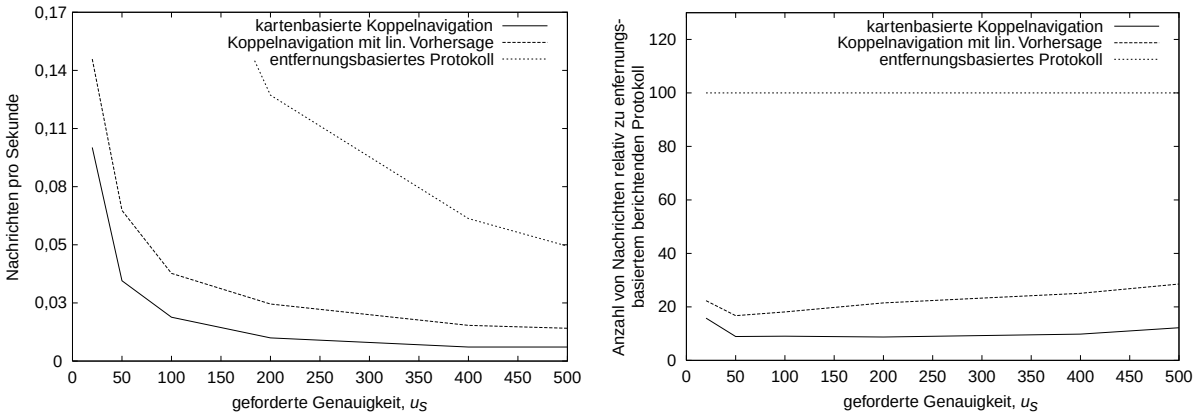


Abbildung 6-16. Fahrzeug auf Autobahn: Anzahl von Positionsaktualisierungsnachrichten absolut (links) bzw. relativ zu einem entfernungsbasierten Protokoll (rechts) abhängig von der geforderten Genauigkeit für ein kartenbasiertes Koppelnavigationsprotokoll und eines mit linearer Vorhersage.

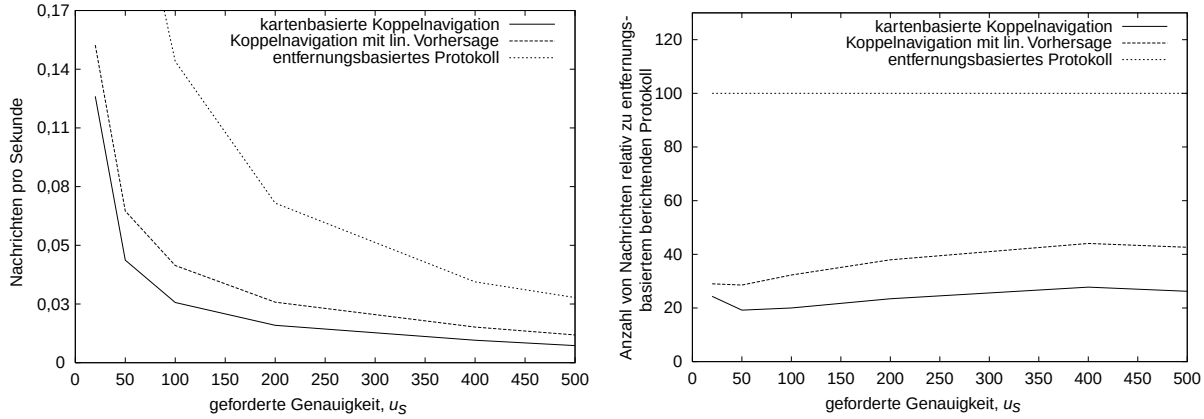


Abbildung 6-17. Fahrzeug auf Landstraße: Anzahl von Positionsaktualisierungsnachrichten absolut (links) bzw. relativ zu einem entfernungsbasierten Protokoll (rechts) abhängig von der geforderten Genauigkeit für ein kartenbasiertes Koppelnavigationsprotokoll und eines mit linearer Vorhersage.

Die für die Koppelnavigationsprotokolle benötigten Informationen über Geschwindigkeit und Bewegungsrichtung des mobilen Objekts wurde im Falle eines Fahrzeugs auf der Autobahn aus zwei aufeinander folgenden der im Sekundentakt vom Positionierungssensor gelieferten Positionen interpoliert, bei einem Fahrzeug auf der Landstraße oder in der Stadt aus vier und bei einem Fußgänger aus acht Positionen. Diese Werte, die von der Genauigkeit des Positionierungssensors und der Geschwindigkeit des mobilen Objekts abhängen, haben sich in weiteren Simulationen als optimal herausgestellt. Für Details siehe NICU (2001).

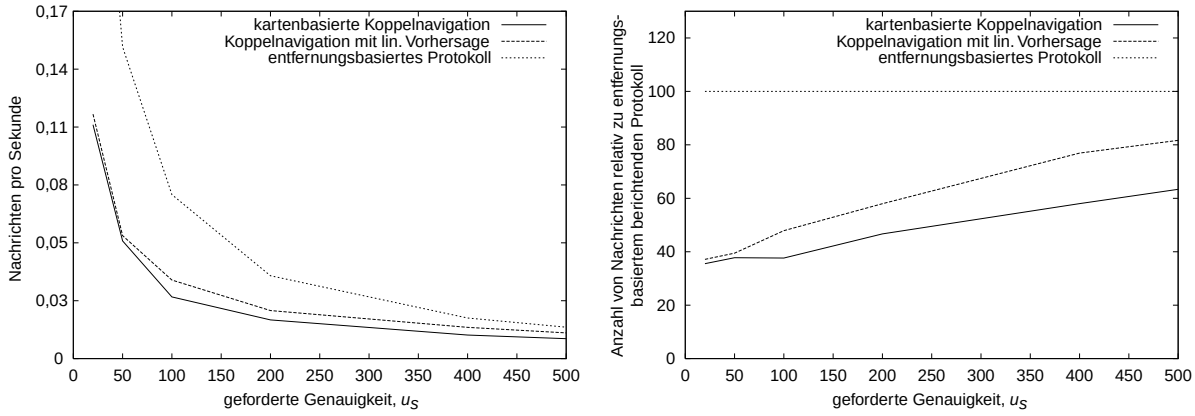


Abbildung 6-18. Fahrzeug im Stadtverkehr: Anzahl von Positionsaktualisierungsnachrichten absolut (links) bzw. relativ zu einem entfernungsbasierten Protokoll (rechts) abhängig von der geforderten Genauigkeit für ein kartenbasiertes Koppelnavigationsprotokoll und eines mit linearer Vorhersage.

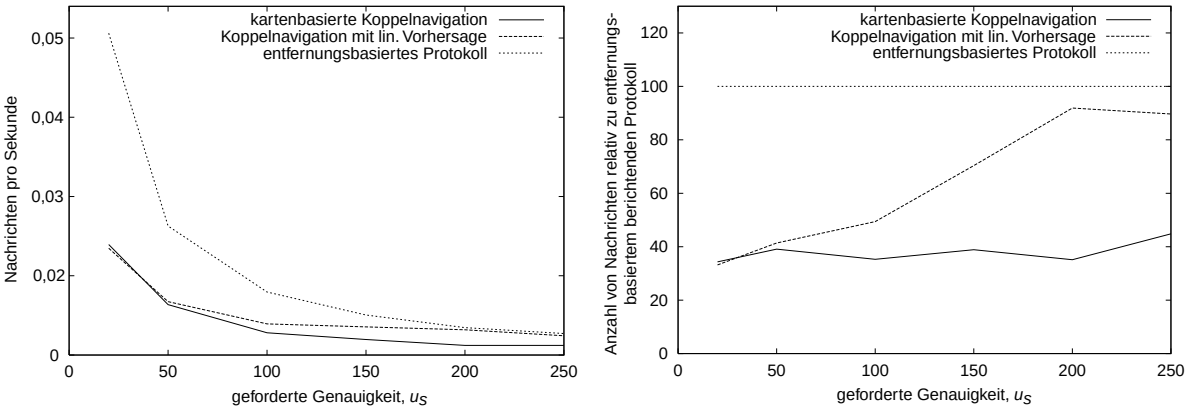


Abbildung 6-19. Fußgänger: Anzahl von Positionsaktualisierungsnachrichten absolut (links) bzw. relativ zu einem entfernungsbasierten Protokoll (rechts) abhängig von der geforderten Genauigkeit für ein kartenbasiertes Koppelnavigationsprotokoll und eines mit linearer Vorhersage.

Wie zu erwarten fallen die Verbesserungen, die schon das Koppelnavigationsprotokoll mit linearer mit Vorhersagefunktion gegenüber dem berichtenden Protokoll aufweist, für ein Fahrzeug auf der Autobahn höher aus (bis zu 83%) als z. B. für den Stadtverkehr (bis zu 63%). Der Grund dafür ist der deutlich gleichförmigere Verkehr, in Bezug auf Geschwindigkeits- und Richtungsänderungen, sowie die geringere Anzahl von Kreuzungspunkten auf der Autobahn. Bei allen Bewegungscharakteristika nehmen die Verbesserungen mit einer höheren geforderten Genauigkeit ab, da es mit einem zunehmenden Zeitabstand zwischen Aktualisierungsnachrichten immer

wahrscheinlicher wird, dass das mobile Objekt Geschwindigkeit oder Bewegungsrichtung ändert. Bei dem geringsten betrachteten Wert für die geforderte Genauigkeit von 20 m gibt es entgegen diesem Trend z. T. eine leichte Abnahme der Verbesserung, da sich dort die Ungenauigkeit des GPS-Empfängers bemerkbar macht.

Das kartenbasierte Koppelnavigationsverfahren verringert die benötigten Positionsaktualisierungen im Falle des Autobahnverkehrs um bis zu weiteren 60% verglichen mit dem Koppelnavigationsverfahren mit linearer Vorhersagefunktion. Nur für einen Fußgänger und die höchste geforderte Genauigkeit benötigt das kartenbasierte Verfahren mehr Nachrichten. Darüber hinaus kann das kartenbasierte Koppelnavigationsprotokoll, im Gegensatz zu dem mit linearer Vorhersage, die Verbesserungen auch für eine geringere geforderte Genauigkeit aufrecht erhalten (was besonders deutlich in Abbildung 6-19 zu sehen ist), da es besser in der Lage ist, die zukünftige Position des mobilen Objekts vorherzusagen.

6.6 Zusammenfassung und Erweiterungsmöglichkeiten

In diesem Abschnitt wurde ein umfassender Überblick über verschiedene Protokollklassen gegeben, die sich zur Übertragung von Positionsinformationen mit hoher Genauigkeit eignen. Ein Einsatz dieser Protokolle ist dabei nicht auf den LS beschränkt, sondern ist überall dort möglich, wo Positionsinformationen kontinuierlich übertragen werden sollen. Mit Hilfe des hier vorgestellten Vergleichs dieser Protokolle hinsichtlich ihrer Eigenschaften kann für eine vorgegebene Aufgabenstellung ein geeignetes Protokoll ausgewählt werden. Eine analytische Betrachtung der Protokolle ermöglicht weiterhin eine Abschätzung für die durch ein Protokoll verursachte Netzwerklast, die resultierende mittlere Ungenauigkeit der übertragenen Positionsinformationen und die Angabe einer oberen Grenze für deren maximale Ungenauigkeit. Simulationsergebnisse, die auf GPS-Protokollen der Bewegungen realer Objekte basieren, bestätigen die Ergebnisse der Analyse.

Aufbauend auf die zwei grundlegenden Klassen der anfragenden und berichtenden Protokolle haben wir ein kombiniertes Protokoll vorgeschlagen, das deren positive Eigenschaften vereint und für die meisten Einsatzzwecke geeignet sein sollte. Insbesondere kommt es im Gegensatz zu den berichtenden Protokollen auch mit einer je Anfrage geforderten Genauigkeit der Positionsinformationen zurecht. Für das kombinierte Protokoll kann die Analyse geeignete Parametereinstellungen für eine bestimmte Einsatzumgebung liefern. Eine hier nicht betrachtete Erweiterungsmöglichkeit für das kombinierte Protokoll stellen Mechanismen zur adaptiven Anpassung an die jeweilige Einsatzumgebung dar, die z. B. auf einer Minimierung der durch Gleichung 6-12 abgeschätzten Netzwerklast beruhen.

Auf die viel versprechende Klasse der Koppelnavigationsprotokolle wurde im Detail eingegangen und ein Überblick über mögliche Varianten gegeben. Darauf basierend wurde ein neuartiges kartenbasiertes Verfahren vorgestellt. Durch umfangreiche Simulationen für unterschiedliche Bewegungscharakteristika mobiler Objekte haben wir das kartenbasierte Koppelnavigationsprotokoll, ein einfacheres Koppelnavigationsprotokoll mit linearer Vorhersagefunktion und ein Nicht-Koppelnavigationsprotokoll verglichen. Es hat sich gezeigt, dass für alle Bewegungscharakteristika schon das einfachere Koppelnavigationsprotokoll eine deutliche Verbesserung gegenüber den Nicht-Koppelnavigationsverfahren darstellt. Durch Einsatz des kartenbasierten Verfahrens kann die Zahl der notwendigen Aktualisierungsnachrichten z. T. nochmal um mehr als die Hälfte reduziert werden (insgesamt um bis zu 91%). Wir sind der Überzeugung, dass sich Koppelnavigationsprotokolle für spezielle Einsatzgebiete noch weiter optimieren lassen und dass sie ein wichtiger Baustein für das Verwalten von hochaktuellen Positionsinformationen in Anwendungen mit Ortsbezug sind.

Eine weitere Verbesserung der Koppelnavigationsprotokolle könnte durch eine Einbeziehung von Geschwindigkeitsänderungen in die Vorhersagefunktion erreicht werden. So sind viele der verbleibenden Positionsaktualisierungen durch Geschwindigkeitsänderungen bedingt, in einem extremen Beispiel wenn ein Fahrzeug an einer Ampel halten muss. Bei einem kartenbasierten Verfahren könnte beispielsweise zusätzlich die Information über Geschwindigkeitsbeschränkungen verwendet werden, um die angenommene Geschwindigkeit für ein mobiles Objekt entsprechend anzupassen. Die Vorhersagefunktion könnte weiterhin auch um eine Berücksichtigung der Beschleunigungs- und Abbremsphasen der mobilen Objekte erweitert werden. Da diese allerdings eine im Vergleich zu den in unseren Simulationen betrachteten Aktualisierungsintervallen kurze Dauer haben, macht dies voraussichtlich jedoch nur für eine sehr hohe geforderte Genauigkeit Sinn und auch nur dann, wenn auch die erwartete Zielgeschwindigkeit mitberücksichtigt wird.

Kapitel 7

Experimente

In diesem Kapitel sollen die in Kapitel 5 vorgestellte Architektur und Algorithmen anhand von Messungen an einer prototypischen Implementierung des LS evaluiert werden. Da die Leistungsfähigkeit des LS zu einem großen Teil davon abhängt, wie effizient Anfragen von den beteiligten Lokations-Servern bearbeitet werden können, haben wir uns für eine Bewertung durch Messungen entschieden. Durch eine Simulation, z. B. unter Verwendung des weit verbreiteten Netzwerksimulators NS-2 (BAJAJ ET AL. (1999)), hätte nur der Kommunikationsaufwand und nicht die Bearbeitungszeiten auf den Servern berücksichtigt werden können.

Für die Experimente mit dem verteilten LS wurde ein an der Abteilung Verteilte Systeme vorhandenes Testsystem bestehend aus acht PCs (siehe HERRSCHER, LEONHARDI & ROTHERMEL (2002) oder NEL (2001)) verwendet. Sieben davon standen für die Messungen als einzelne Lokations-Server in verschiedenen zu untersuchenden Konfigurationen zur Verfügung, da einer der Rechner für die Steuerung der Experimente vorgesehen ist. Ein geplantes erweitertes System mit über 32 Knoten, verbunden über ein konfigurierbares Hochgeschwindigkeitsnetzwerk, war leider zum Zeitpunkt dieser Arbeit noch nicht verfügbar.

7.1 Prototypische Implementierung des LS

Für die Messungen wurde eine prototypische Implementierung des LS erstellt. Die Funktionalität dieses Prototyps umfasst die in Abschnitt 4.2 beschriebene einfache Registrierung von mobilen Objekten, Positionsaktualisierungen und die Zuständigkeitsübergabe zwischen Lokations-Servern. Weiterhin werden die in Abschnitt 4.3 beschriebenen Positions-, Gebiets- und Nachbarschaftsanfragen in einer leicht vereinfachten Form, bei der nur die in den Lokationsbe-

zeichnern gespeicherten Positionsinformation und nicht die Aufenthaltsbereiche berücksichtigt werden, unterstützt. Der Prototyp beherrscht weiterhin eine verteilte Bearbeitung der jeweiligen Funktionen entsprechend den Algorithmen aus Abschnitt 5.4. Die in Abschnitt 5.5 vorgestellten Mechanismen zur Optimierung der Algorithmen durch Zwischenspeichern von Informationen werden hier nicht betrachtet.

Der Prototyp des LS wurde unter Verwendung der Programmiersprache Java 1.3 (SUN (2001)) implementiert und baut auf die in der Diplomarbeit WÜNSCHE (1999) entstandenen Vorarbeiten auf. Der Prototyp ist sowohl auf PCs unter den Betriebssystemen Microsoft Windows (getestet mit Windows 98, NT und 2000) oder unter Linux als auch auf Sun Ultra Rechnern unter Solaris lauffähig.

Wie in Abschnitt 5.3 beschrieben wurde für die Speicherung der Positionsinformationen eine Hauptspeicherdatenbank basierend auf einem Punkt-Quadtree (SAMET (1990)) verwendet, der sich im Vergleich mit anderen Indexstrukturen (siehe HÄSSLER (2001)) als am geeignetsten für diesen Zweck herausgestellt hat. Registrierungsinformationen und Suchverweise, die auf stabilem Speicher gehalten werden sollen, können auf einer über die Java Datenbankschnittstelle JDBC angebundenen IBM DB2 Datenbank (IBM (2001)) gespeichert werden.

Die Kommunikationsprotokolle, die zum Austausch von Nachrichten zwischen Klienten, mobilen Objekten und Lokations-Servern benötigt werden, sind in einer Studienarbeit FREI (2000) entstanden und bauen auf dem leichtgewichtigen UDP-Protokoll (siehe COMER (2000)) auf. Da die Komponenten des LS zumeist nur einzelne Nachrichtenpakete für Anfragen und Ergebnisse austauschen (z. B. zur Positionsaktualisierung), ist durch diese Realisierung eine effiziente Klienten/Server- sowie Server/Server-Kommunikation gegeben. Fehlerbehandlung und Flusskontrolle können durch einfache Mechanismen (wie Sequenznummern und Packetbestätigungen) sichergestellt werden.

7.2 Durchsatz der Datenhaltungskomponente

Um die Leistungsfähigkeit der von uns vorgeschlagenen Hauptspeicherdatenbank zu evaluieren, wurden Messungen auf unterschiedlichen Zielsystemen durchgeführt, die den Durchsatz bei der lokalen Bearbeitung von verschiedenen für den LS benötigten Operationen zeigen. Die Ergebnisse dieser Messungen sind in Tabelle 7-1 für die gewählte Hauptspeicherlösung auf einem einfachen PC mit 800 Mhz CPU und 256 MB Speicher unter Linux und einem PC-Server mit vier 700 Mhz CPUs und 4 GB Speicher unter Windows 2000 dargestellt. Als Vergleich enthält die letzte Spalte von Tabelle 7-1 den Durchsatz einer entsprechenden Realisierung auf Basis einer IBM DB2 Datenbank mit Spatial Extender (DAVIS (1998)) in der Version 7.2, der eben-

falls auf dem PC-Server gemessen wurde. Die für die Datenbanklösung verwendeten SQL-Anfragen sind in Anhang D beschrieben.

Bei den durchgeführten Experimenten wurde mit einer leeren Datenhaltungskomponente begonnen, die für ein Dienstgebiet der Größe 10 mal 10 km zuständig ist. In diese wurden zuerst nacheinander 25.000 mobile Objekte mit zufällig ausgewählten Positionen aus diesem Dienstgebiet eingefügt. Anschließend wurden jeweils 10.000 (im Falle der Hauptspeicherlösung) bzw. 100 (im Falle der Datenbanklösung) Positionsaktualisierungen für zufällige Positionen sowie Positionsanfragen für zufällige Objekte durchgeführt. Schließlich wurden ebenso viele Gebietsanfragen für zufällig innerhalb des Dienstgebiets gewählte Gebiete mit jeweils drei verschiedenen Größen sowie Nachbarschaftsanfragen für zufällige Positionen an die Datenhaltungskomponente gestellt. Die entsprechenden Messergebnisse sind für die unterschiedlichen Operationen in Tabelle 6-1 gezeigt. Die Messungen zeigen allerdings nur die für das Speichern und Abfragen der Informationen auf einem einzelnen Rechner benötigte Zeit. Nicht berücksichtigt sind hier der für das Abfragen von einem entfernten Rechner aus benötigte zusätzliche Aufwand, der sich durch Einschränkungen der Netzwerkschnittstelle, das Ein- und Auspacken der Parameter und Ergebnisse sowie das Überprüfen der Parameter ergibt. Ergebnisse, die dies mit berücksichtigen, sind im nächsten Abschnitt enthalten.

Unsere Ergebnisse zeigen, dass mit der hauptspeicherbasierten Lösung vor allem eine sehr effiziente Bearbeitung von Positionsaktualisierungen erreicht werden kann. Positionsanfragen, bei denen nicht auf den mehrdimensionalen Index zugegriffen werden muss, können sogar noch schneller bearbeitet werden. Der Aufwand für eine Gebietsanfrage hängt natürlich von der Größe des angefragten Gebiets ab, da diese bestimmt, welcher Anteil des mehrdimensionalen Indexes durchsucht werden muss. Aber selbst für die größten Gebiete, mit einer Seitenlänge von 1 km, können auf dem PC-Server mehr als 2000 Anfragen pro Sekunde durchgeführt werden. Unsere Ergebnisse zeigen weiterhin, dass sich der mehrdimensionale Index sehr schnell aufbauen lässt (in unserem Fall mit über 25.000 Einfügeoperationen auf dem PC-Server). Da der mehrdimensionale Index bei einem Zusammenbruchsfehler eines Lokations-Servers verloren geht, ist dies für das Wiederherstellen des Servers besonders wichtig. Der Index wird dabei, wie in Abschnitt 5.3.3 beschrieben, aus den wieder eintreffenden Positionsaktualisierungen aufgebaut.

Im Vergleich dazu ist die auf Basis einer DB2 Datenbank implementierte experimentelle Datenhaltungskomponente um mehr als den Faktor 100 langsamer, besonders, wenn es sich um Operationen mit räumlichen Daten handelt. Wie bereits erwähnt sind herkömmliche räumliche Datenbanken auf komplexe Anfragen mit umfangreichen Daten ausgelegt und erzeugen auch für einfache Anfragen, wie sie im LS benötigt werden, einen beträchtlichen Zusatzaufwand. Während eine Datenbank zwar einen stabilen Speicher für die Positionsinformationen mit sich

bringt, würde besonders die niedrige Aktualisierungsrate zu einem Flaschenhals für die Implementierung eines effizienten Lokationsdienstes werden.

Zusammenfassend haben unsere Experimente gezeigt, dass mit einer hauptspeicherbasierten Datenhaltungskomponente meist weit über 1000 Aktualisierungen und Anfragen pro Sekunde selbst für Gebietsanfragen mit großen angefragten Gebieten bearbeitet werden können. Bei Verwendung der in dieser Arbeit vorgeschlagenen hauptspeicherbasierten Ansatzes sollte, im Gegensatz zu einer auf einer Datenbank basierenden Lösung, die Datenhaltungskomponente auch dann nicht zu einem Flaschenhals werden, wenn sehr dynamische Positionsinformationen vieler mobiler Objekte verwaltet werden müssen.

Operationstypen	Operationen pro Sek. Hauptspeicherdatenbank auf einfachem PC	Operationen pro Sek. Hauptspeicherdatenbank auf PC-Server	Operationen pro Sek. DB2 Datenbank mit Spatial Extender auf PC-Server
Index erstellen	13.804 pro s	29.103 pro s	47,10 pro s
Positionsaktualisierungen	16.393 pro s	70.921 pro s	55,25 pro s
Positionsanfragen	131.578 pro s	645.161 pro s	266,67 pro s
Gebietsanfragen (10 × 10 m)	5.583 pro s	24.630 pro s	172,71 pro s
Gebietsanfragen (100 × 100 m)	4.387 pro s	22.831 pro s	31,12 pro s
Gebietsanfragen (1 × 1 km)	496 pro s	2.424 pro s	4,73 pro s
Nachbarschaftsanfragen	2.538 pro s	15.600 pro s	27,58 pro s

Tabelle 7-1. Durchsatz der Datenhaltungskomponente eines Lokations-Servers mit einem Dienstgebiet der Größe 10 km × 10 km und 25.000 registrierten mobilen Objekten.

7.3 Antwortzeiten und Gesamtdurchsatz der Operationen des LS

Die bisherigen Untersuchungen haben sich lediglich mit der Leistungsfähigkeit der Datenhaltungskomponente des LS beschäftigt. In diesem Abschnitt soll die Antwortzeit und der Durchsatz für die einzelnen Operationen in einem verteilten LS betrachtet werden, wie sie von einem

entfernten Klienten, der über die UDP-Schnittstelle auf den LS zugreift, wahrgenommen werden.

Um die Gesamtantwortzeit sowie den Gesamtdurchsatz für die vom LS unterstützten Anfragen für unterschiedliche Konfigurationen des LS zu ermitteln, haben wir auf dem oben erwähnten PC-Cluster zwei verschiedene Testkonfigurationen eingerichtet. Beide Konfigurationen haben dieselbe Anzahl von vier Blatt-Servern, die jeweils für ein Viertel des Gesamtdienstgebiets von 1,5 mal 1,5 km zuständig sind (siehe Abbildung 7-1). Bei der ersten Konfiguration sollte der maximale Verzweigungsgrad erreicht werden, weshalb ein fünfter Rechner direkt als Wurzel-Server und Vorgänger der Blatt-Server fungiert. Mit der zweiten Konfiguration soll hingegen eine tiefere Server-Hierarchie betrachtet werden. Dem Wurzel-Server sind hier zwei weitere innere Lokations-Server untergeordnet, die jeweils Vorgänger von zwei benachbarten Blatt-Servern sind. Die PC-Rechner, auf denen die Lokations-Server der Testkonfigurationen ausgeführt werden, entsprechen dem im vorigen Kapitel betrachteten einfachen PC und haben damit jeweils eine 800-Mhz CPU, 256 MB Hauptspeicher und sind durch ein 100 MBit Ethernet Netzwerk verbunden.

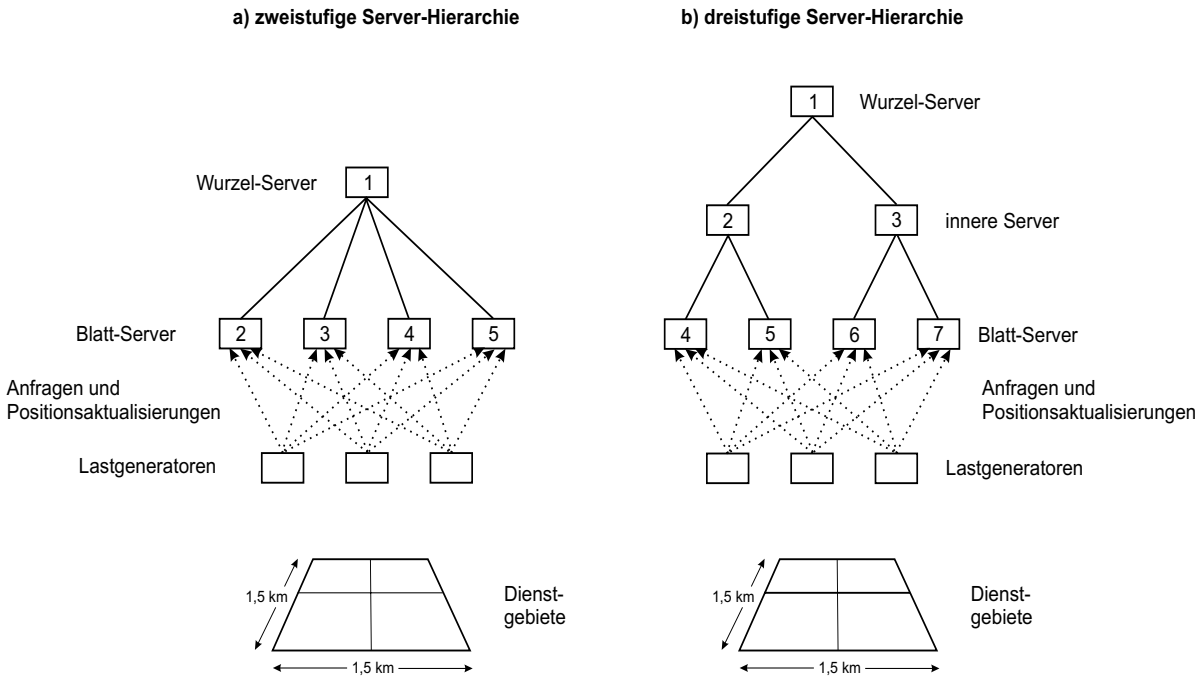


Abbildung 7-1. Testkonfigurationen des LS bei der Messung von Antwortzeit und Durchsatz von verschiedenen verteilten Operationen.

Für diese Konfigurationen des LS haben wir die Antwortzeiten und den Gesamtdurchsatz für Positionsaktualisierungen sowie für lokale und entfernte Positions-, Gebiets- und Nachbarschaftsanfragen gemessen. Das in den Gebietsanfragen abgefragte Gebiet hatte dabei die einem größeren Raum entsprechende mittlere Größe von 50 mal 50 Metern. Dazu haben wir zuerst 20.000 mobile Objekte mit zufällig gewählten Positionen beim LS registriert. Für die Messung der Antwortzeit hat ein einzelner Rechner, der ebenfalls mit 100 MBit/s an das Testsystem angebunden ist, als Lastgenerator jeweils 10.000 Anfragen der unterschiedlichen Anfragetypen an einen entsprechenden Blatt-Server gesendet (siehe unten), so schnell wie diese vom Blatt-Server abgearbeitet werden können.

In Tabelle 7-2 sind die Ergebnisse unserer Messungen für Positionsaktualisierungen und Positionsanfragen, in Tabelle 7-3 für Gebiets- und in Tabelle 7-4 für Nachbarschaftsanfragen gezeigt. Die in diesen Tabellen angegebenen Antwortzeiten ergeben sich durch das durchschnittliche Zeitintervall, das jeweils zwischen Absenden der Anfrage und dem vollständigen Eintreffen der Ergebnisse beim Testklienten vergangen ist.

Bei der Messung des Gesamtdurchsatzes haben drei Rechner zur Erzeugung der notwendigen Last an Anfragen oder Positionsaktualisierungen gedient (wie in Abbildung 7-1 gezeigt). Diese haben Anfragen und Aktualisierungsnachrichten in mehreren parallelen Prozessen so schnell wie möglich und zu gleichen Teilen an die vier Blatt-Lokations-Server der Testkonfiguration gesendet. Die Blatt-Server haben damit jeweils einen gleich großen Anteil der Gesamtlast zu bearbeiten gehabt. Zur Bestimmung des in Tabelle 7-2 bis Tabelle 7-4 gezeigten Gesamtdurchsatzes wurden die von den einzelnen Blatt-Servern pro Sekunde bearbeiteten Positionsaktualisierungen oder Anfragen aufaddiert. Die Ergebnisse sind dabei der Mittelwert aus einem jeweils 5 Minuten dauernden Experiment.

Um die beim Traversieren der Server-Hierarchie entstehenden Kosten zu ermitteln, haben wir in der ersten und zweiten Testkonfiguration zwischen lokalen Anfragen sowie entfernten Anfragen, bei denen eine Traversierung von einer bzw. zwei Ebenen der Server-Hierarchie notwendig sind, unterschieden. Bei Positionsaktualisierungen (die in unserer Architektur immer lokal sind) sowie für lokale Positions-, Gebiets- und Nachbarschaftsanfragen wurde die Anfrage immer an den Blatt-Lokations-Server gestellt, der diese vollständig bearbeiten kann. Für entfernte Anfragen wurde ein Kontakt-Server gewählt, der, wenn eine Hierarchieebene durchlaufen werden soll, im selben Teilbaum wie der Ziel-Server gelegen ist, bzw. im anderen Teilbaum, wenn zwei Hierarchieebenen traversiert werden sollen.

Weiterhin haben wir auch den Zusatzaufwand untersucht, der sich aus entfernten Gebietsanfragen ergibt, von denen mehr als einen Blatt-Lokations-Server betroffen ist. Zu diesem Zweck haben wir in der ersten Testkonfiguration ein Gebiet abgefragt, das (1) komplett im Dienstgebiet

eines Blatt-Servers liegt, das (2) sich mit den Dienstgebieten von zwei Blatt-Servern überlappt und das (3) sich mit den Dienstgebieten aller vier Blatt-Server überschneidet.

Der Gesamtdurchsatz an Positionsaktualisierungen, den der LS in unseren Testkonfigurationen erreicht, würde ausreichen, um die Positionsinformationen von mehr als 55.000 mobilen Objekten mit einer durchschnittlichen Geschwindigkeit von 3 km/h und einer Genauigkeit von 25 m selbst bei einem einfachen Aktualisierungsprotokoll zu verwalten[1]. Lokale Anfragen können vergleichsweise effizient bearbeitet werden, wobei Durchsatz und Antwortzeiten bei Nachbarschaftsanfragen und vor allem bei Gebietsanfragen schlechter sind als bei Positionsanfragen. Der Grund hierfür ist der größere Aufwand bei der Bearbeitung dieser Anfragen sowie die größere Ergebnismenge.

Operation	Antwortzeit	Durchsatz
Positionsaktualisierung (mit ACK)	2,5 ms	1.946 pro s
Lokale Positionsanfrage	2,6 ms	1.754 pro s
Entfernte Positionsanfrage (1 Hierarchieebene)	11,2 ms	393 pro s
Entfernte Positionsanfrage (2 Hierarchieebenen)	16,8 ms	238 pro s

Tabelle 7-2. Antwortzeit und Gesamtdurchsatz für Positionsaktualisierungen sowie lokale und entfernte Positionsanfragen in den Testkonfigurationen des LS.

Obwohl Durchsatz und Antwortzeiten für entfernte Anfragen immer noch gut sind, sind besonders entfernte Gebietsanfragen deutlich teuerer als lokale Anfragen. Der Grund hierfür liegt darin, dass von einer entfernten Anfrage immer mindestens drei Lokations-Server betroffen sind. Hier wirkt sich zunächst auch der zusätzliche Kommunikationsschritt aus, der sich aus unserer Entscheidung ergibt, die Teilergebnisse erst zum Kontakt-Server der Anfrage zu schicken, bevor sie gesammelt dem Klienten übermittelt werden (siehe Abschnitt 5.4). Das Durchlaufen von weiteren Hierarchiestufen verlängert in unseren Testkonfigurationen die Antwortzeit nur noch um je nach Anfrage etwa 5,5 ms. Eine deutliche Verbesserung sollten auch die in Abschnitt 5.5

1. Um zukünftig eine möglichst hohe Anzahl von Knoten zu ermöglichen, besteht das verwendete Testsystem nur aus einfachen PC-Rechnern. Ein deutlich höherer Durchsatz sollte sich durch den Einsatz von entsprechenden Server-Rechner erreichen lassen (vgl. auch Tabelle 7-1). In früheren Messungen, deren Ergebnisse in LEONHARDI & ROTHERMEL (2002) beschrieben sind, wurde auf einem System aus SUN Rechnern ein meist mehr als doppelt so hoher Durchsatz erzielt.

skizzierten hier nicht berücksichtigten Optimierungen bringen, die das Durchlaufen der Server-Hierarchie vermeiden.

Operation	Antwortzeit	Durchsatz
Lokale Gebietsanfrage	15,9 ms	471 pro s
Entfernte Gebietsanfrage (1 Hierarchieebene, 1 Server)	49,2 ms	143 pro s
Entfernte Gebietsanfrage (1 Hierarchieebene, 2 Server)	49,5 ms	116 pro s
Entfernte Gebietsanfrage (1 Hierarchieebene, 4 Server)	48.4 ms	100 pro s
Entfernte Gebietsanfrage (2 Hierarchieebenen, 1 Server)	53,9 ms	110 pro s

Tabelle 7-3. Antwortzeit und Gesamtdurchsatz für lokale und entfernte Gebietsanfragen in den Testkonfigurationen des LS.

Operation	Antwortzeit	Durchsatz
Lokale Nachbarschaftsanfrage	4,2 ms	1.138 pro s
Entfernte Nachbarschaftsanfrage (1 Hierarchieebene)	16,5 ms	246 pro s
Entfernte Nachbarschaftsanfrage (2 Hierarchieebenen)	22,9 ms	191 pro s

Tabelle 7-4. Antwortzeit und Gesamtdurchsatz für lokale und entfernte Nachbarschaftsanfragen in den Testkonfigurationen des LS.

7.4 Verzweigungsgrad der Server-Hierarchie und Lokalität der Anfragen

Während die bisherigen Ergebnisse die Verarbeitungsgeschwindigkeit für einzelne Operationen und die Auswirkungen einer unterschiedlichen Höhe der Server-Hierarchie zeigen, soll im Folgenden der Verzweigungsgrad der Hierarchie in Bezug auf die Lokalität der Anfragen betrachtet werden. Zu diesem Zweck wurde für verschiedene Testkonfigurationen mit einem Verzweigungsgrad zwischen drei und fünf (siehe Abbildung 7-2) der Durchsatz für einen vorgegeben Anfrage-Mix mit einem variierten Anteil von lokalen und entfernten Anfragen gemessen. Falls nicht ausdrücklich erwähnt, entsprechen die Rahmenbedingungen dieser Experimente denen der im vorigen Abschnitt beschriebenen.

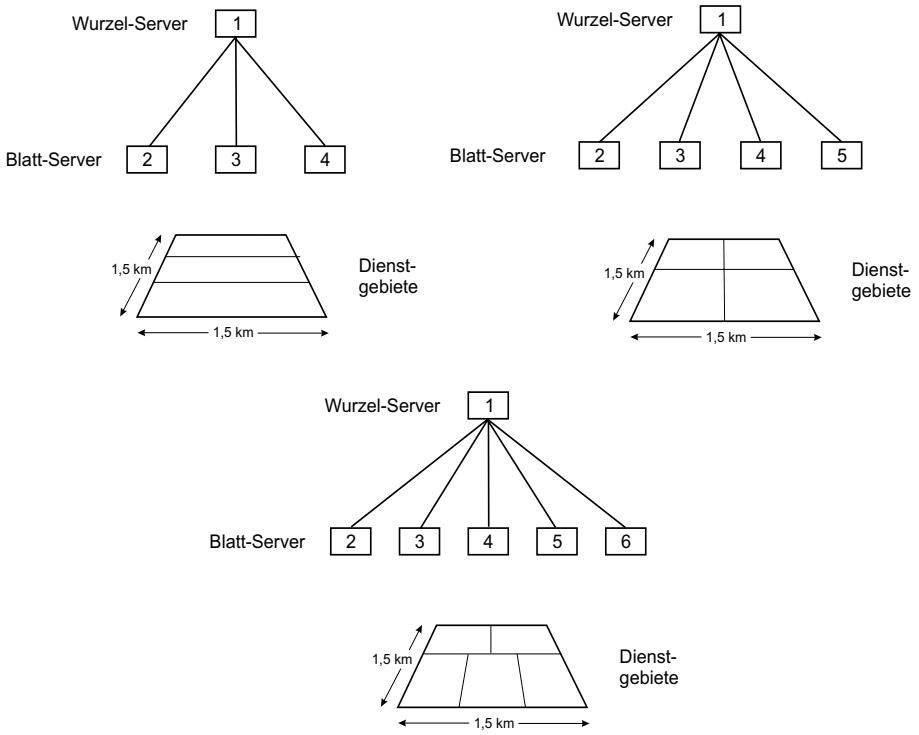


Abbildung 7-2. Testkonfigurationen des LS bei der Messung der Auswirkungen von Verzweigungsgrad und Lokalität der Anfragen.

Bei der Häufigkeit der verschiedenen Anfragetypen innerhalb dieses Anfrage-Mixes wurde ein Verhältnis von 20% zu 40% zu 40% zwischen Positions-, Gebiets- und Nachbarschaftsanfragen gewählt. Dieses Verhältnis ist durch Erfahrungen mit einer prototypischen Realisierung einer auf den LS aufsetzenden ortsbasierten Anwendung als Erweiterung des VIT-Systems (siehe LEONHARDI ET AL. (1999) oder FRITZ (1999)) motiviert. Dort hat sich gezeigt, dass Gebiets- und Nachbarschaftsanfragen häufig für periodisch wiederkehrende Systemfunktionen verwendet werden, wie das Anzeigen der Positionen anderer Benutzer auf einer Straßenkarte. Positionsanfragen werden hingegen hauptsächlich für weniger häufige Benutzeranfragen, wie das Setzen eines ortsbezogenen Lesezeichens, verwendet. Allerdings weisen die verschiedenen Anfragetypen auch ein ähnliches Verhältnis für den Durchsatz im lokalen und entfernten Fall auf (siehe vorheriger Abschnitt). Eine andere Häufigkeit der Anfragetypen innerhalb des Anfrage-Mixes sollte daher keinen wesentlichen Unterschied beim Verhalten des LS für verschiedene Verzweigungsgrade machen.

Als Hintergrundlast wurden von zwei zusätzlichen Rechnern für jeden Blatt-Server 5000 mobile Objekte vor dem Experiment beim LS angemeldet und ihre Positionsinformationen während

des Experiments entsprechend einem sogenannten *Random-Waypoint*-Bewegungsmodell aktualisiert. Bei diesem Modell wählt ein mobiles Objekt jeweils einen zufälligen Zielpunkt aus dem gesamten Dienstgebiet und bewegt sich mit einer Geschwindigkeit von in unserem Fall durchschnittlich 3 km/h auf direktem Wege dorthin. Dort angekommen wählt es einen neuen Zielpunkt. Ein solches Random-Waypoint-Modell bzw. ähnliche Modelle werden oft verwendet, um die Bewegungen mobiler Objekte für die Bewertung von Mobilkommunikationsarchitekturen oder Netzwerkprotokollen mit mobilen Teilnehmern zu beschreiben (wie z B. in BROCH ET AL. (1998)).

Da hier nur eine Hintergrundlast an Positionsaktualisierungen und Zuständigkeitsübergaben erzeugt werden sollte, ist ein einfaches Bewegungsmodell ausreichend. Mit einer durchschnittlichen Geschwindigkeit von 3 km/h wird die Bewegung von Fußgängern nachgebildet. Die Anzahl von Positionsaktualisierungen, die bei einer anderen durchschnittlichen Geschwindigkeit (z. B. der eines Fahrzeugs) anfällt, lässt sich mit Hilfe der Formeln aus Abschnitt 6.4 errechnen. Die Anzahl der mobilen Objekte, die der LS in der betrachteten Konfiguration und einer jeweiligen Anfragelast verwalten kann, wird sich dann um den entsprechenden Faktor verringern bzw. erhöhen.

In Abbildung 7-3 sind die Ergebnisse unserer Experimente gezeigt. Der Anteil an entfernten Anfragen wurde dabei in Schritten von 0,1 von 0 (nur lokale Anfragen) auf 1 (nur entfernte Anfragen) erhöht. Um die Ergebnisse der verschiedenen Konfigurationen mit einer unterschiedlich großen Anzahl von Blatt-Servern vergleichbar zu machen, ist der Durchsatz jeweils auf den Fall mit nur lokalen Anfragen normiert.

Abbildung 7-3 zeigt, dass der Durchsatz bei einem größeren Anteil an entfernten Anfragen mit einem höheren Verzweigungsgrad der Server-Hierarchie immer weiter abnimmt. Der Grund dafür liegt in der höheren Belastung des Wurzel-Servers, da dieser die entfernten Anfragen für eine größere Anzahl von Kind-Servern weiterzuleiten hat. Die Ergebnisse aus dem vorigen Abschnitt haben hingegen gezeigt, dass eine mehrstufige Hierarchie für einen Teil der Anfragen Einbußen bei Antwortzeiten und Durchsatz mit sich bringt und dass unter diesem Gesichtspunkt eine möglichst flache Hierarchie zu bevorzugen ist. Bei der Konfiguration des LS für eine konkrete Einsatzumgebung muss also auf diese zwei Gesichtspunkte Rücksicht genommen werden. Ausgehend von der Anzahl der zu unterstützenden mobilen Objekte und damit der Anzahl benötigter Blatt-Lokations-Server sowie der zu erwartenden Lokalität der Anfragen, die sich aus den eingesetzten Anwendungen ergibt, muss so eine geeignete Höhe und der passende Verzweigungsgrad für die Server-Hierarchie des LS gefunden werden.

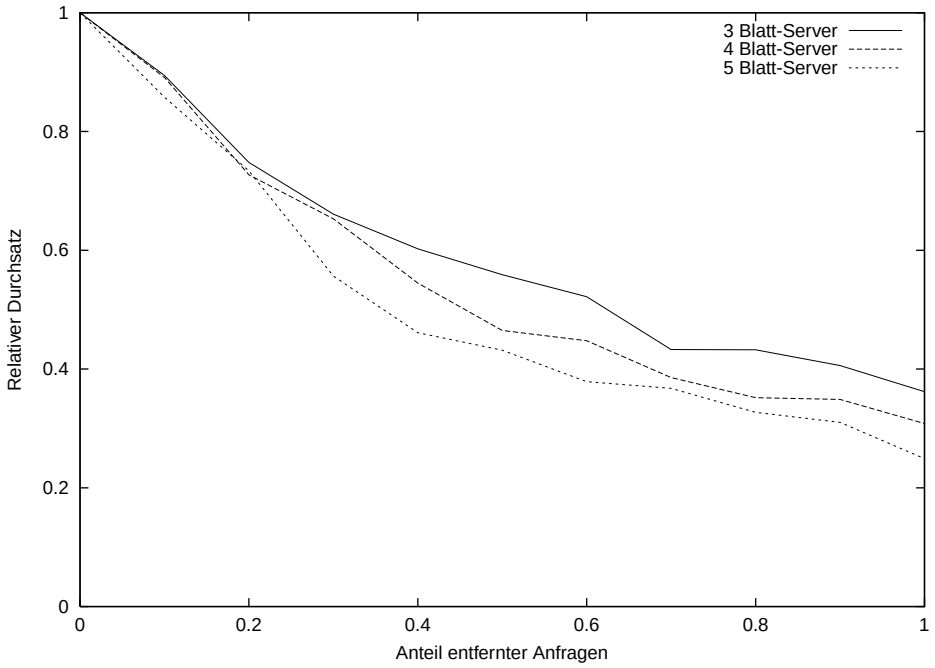


Abbildung 7-3. Durchsatz des LS bei verschiedenen Testkonfigurationen mit einem Verzweigungsgrad zwischen 3 und 5 für einen unterschiedlich hohen Anteil an entfernten Anfragen.

7.5 Konfiguration des LS

Wie in Abschnitt 5.2.2 angesprochen sind die Parameter, die eine Konfiguration des LS bestimmen, 1) die Größe und Form der Dienstgebiete der Blatt-Server, 2) der Verzweigungsgrad der Server-Hierarchie sowie 3) deren Höhe. Aus den Ergebnissen der in diesem Kapitel beschriebenen Experimente an einer prototypischen Implementierung des LS lassen sich einige Anhaltspunkte für eine optimale Einstellung dieser Parameter ziehen. Diese werden im Folgenden zusammengefasst. Eine weiterführende Betrachtung geht über den Rahmen dieser Arbeit hinaus und sollte das Ziel zukünftiger Untersuchungen sein (siehe auch Abschnitt 9.3).

Die Größe eines Blattdienstgebiets wird beschränkt durch die Leistungsfähigkeit des ihm zugeordneten Blatt-Servers, der die Positionsaktualisierungen der darin befindlichen mobilen Objekte und Anfragen hinsichtlich ihrer Position verarbeiten muss. Unsere in Abschnitt 7.3 beschriebenen Experimente zeigen den Durchsatz für die entsprechenden Operationen anhand einer prototypischen Implementierung des LS. Für eine bestimmte Server-Hardware kann die Leistungsfähigkeit über entsprechende Benchmark-Tests ermittelt werden. Die Anzahl von Po-

sitionsaktualisierungen, die für ein bestimmtes Dienstgebiet zu verarbeiten sind, ergibt sich abhängig vom eingesetzten Positionsaktualisierungsprotokoll aus der Zahl der mobilen Objekte in diesem Dienstgebiet[2] sowie deren durchschnittlicher Geschwindigkeit entsprechend den Gleichungen aus Abschnitt 6.4. Die Menge der zu verarbeitenden Anfragen hängt ebenfalls z. T. von der Anzahl der mobilen Objekte im Dienstgebiet ab, vor allem, wenn diese auch gleichzeitig Klienten des LS sind. Darüber hinaus wird sie allerdings stark von den Anwendungen beeinflusst, die den LS nutzen, und muss daher für das jeweilige Einsatzgebiet gesondert ermittelt werden. Die Form der Blattdienstgebiete sollte schließlich so gewählt werden, dass möglichst wenig Zuständigkeitsübergaben benötigt werden. Bei einer gleichförmigen Verteilung und Bewegung der mobilen Objekte würde dies bedeuten, dessen Umfang so gering wie möglich zu halten.

Die Auswirkungen des Verzweigungsgrads der Server-Hierarchie des LS hängen stark von der Lokalität der Anfragen ab, wie in Abschnitt 7.4 gezeigt wurde. Die Lokalität der Anfragen ist wiederum stark abhängig von der Nutzung des LS durch ortsbezogene Anwendungen und muss für den jeweiligen Einsatzzweck ermittelt werden. Grundsätzlich ist ein hoher Verzweigungsgrad für den LS vorzuziehen, da dieser eine niedrige Höhe der Server-Hierarchie ermöglicht und damit die Kosten für deren Traversierung verringert (siehe unten). Die in Abschnitt 7.4 beschriebenen Untersuchungen zeigen allerdings, dass bei einer geringeren Lokalität der Anfragen (mehr als 20% an entfernten Anfragen) die Leistungsfähigkeit des LS mit einem höheren Verzweigungsgrad abnimmt. Ein hoher Verzweigungsgrad ist daher nur bei einer sehr hohen Lokalität der Anfragen sinnvoll. Wegen der eingeschränkten Zahl von zur Verfügung stehenden Testrechnern konnte dieser Zusammenhang zwischen der Lokalität von Anfragen und dem Verzweigungsgrad der Server-Hierarchie allerdings nur tendenziell aufgezeigt werden und sollte in weiteren Arbeiten ausführlicher untersucht werden.

Wenn der Verzweigungsgrad der Server-Hierarchie und die Größe der Blattdienstgebiete vorgegeben sind, kann die benötigte Höhe aus der Größe des abzudeckenden Gesamtdienstgebiets unter Verwendung bekannter Gleichungen für Baumstrukturen berechnet werden. Die Kosten für die Operationen des LS, die bei der Traversierung einer zusätzlichen Hierarchieebene entstehen, können aus den Unterschieden in Durchsatz und Antwortzeiten für lokale und entfernte Operationen, wie sie in Abschnitt 7.3 beschrieben sind, ersehen werden.

2. Bestimmt durch den Flächeninhalt des Dienstgebiets multipliziert mit der darin vorhandenen Dichte an mobilen Objekten.

7.6 Zusammenfassung

Unsere Experimente mit einer prototypischen Realisierung des LS haben zum einen gezeigt, dass mit der vorgeschlagenen hauptspeicherbasierten Datenhaltung eine sehr deutliche Leistungsverbesserung gegenüber einer herkömmlichen räumlichen Datenbank erzielt werden kann. Zum anderen kann auch die verteilte Architektur des LS Anfragen und vor allem Positionsaktualisierungen effizient ausführen. Dies ist vor allem für lokale Anfragen der Fall, die ja nach unseren Annahmen einen hohen Anteil der Anfragen ausmachen werden. Entfernte Anfragen weisen immer noch eine gute Antwortzeit und einen guten Durchsatz auf, die allerdings gegenüber lokalen Anfragen abhängig von der Höhe der Server-Hierarchie und deren Verzweigungsgrad z. T. deutlich abnimmt. Eine Verbesserung der Leistungsfähigkeit des LS sollte sich in vielen Fällen durch den Einsatz der in Abschnitt 5.5 betrachteten Mechanismen zum Zwischenspeichern von Informationen ergeben (siehe auch Abschnitt 9.3). Voraussetzung für eine solche Erweiterung der Architektur des LS ist allerdings auch eine detailliertere Untersuchung verschiedener Konfigurationen, für die eine Testumgebung mit einer größeren Anzahl von Knoten benötigt wird. Schließlich geben die in diesem Kapitel beschriebenen Ergebnisse einige Anhaltspunkte dafür, wie eine optimale Konfiguration des LS für einen bestimmtes Einsatzgebiet aussehen muss.

Kapitel 8

Sicherheits- und Datenschutzaspekte

Bei den Positionsinformationen, die im LS gespeichert sind, handelt es sich um äußerst sensitive Informationen. So kann z. B. über die Positionsinformation auf die aktuelle Tätigkeit eines Benutzers geschlossen oder, durch deren Beobachtung über einen längeren Zeitraum hinweg, ein sehr detailliertes Profil seiner Gewohnheiten erstellt werden. Mögliche Benutzer des LS werden daher nur zulassen, dass dieser ihre Position speichert, wenn sie sicher sind, dass die entsprechende Information vertraulich behandelt wird. Der LS muss also ein wirksames Konzept für eine Zugriffskontrolle und die Garantie des Datenschutzes bieten.

Beim Entwurf von entsprechenden Zugriffskontrollmechanismen für Positionsinformationen soll daher darauf geachtet werden, dass die Kontrolle über die Herausgabe dieser Informationen schlussendlich bei dem betreffenden Benutzer selbst liegt. Dabei gilt es zwei unterschiedliche Aspekte zu beachten:

- Der Benutzer vertraut dem LS selbst.
- Der Benutzer vertraut darüber hinaus bestimmten Klienten oder Klientengruppen.

In SPREITZER & THEIMER (1993) wird vorgeschlagen, zum Zweck der Zugriffskontrolle die Positionsinformationen eines Benutzers in einem sogenannten Benutzeragenten zu kapseln. Dieser wird auf einem Rechner ausgeführt, der sich unter Kontrolle des Benutzers befindet, und überwacht sämtliche Anfragen bezüglich dessen Position. Dieser Ansatz ist zwar für Positionsanfragen möglich, erschwert allerdings die Ausführung von Gebiets- und Nachbarschaftsanfragen sowie die Überwachung von entsprechenden Ereignissen beträchtlich. Für den LS, bei dem die effiziente Ausführung von letzteren von großer Bedeutung ist, trägt ein solcher Ansatz daher nicht.

Der LS muss deshalb Zugriffskontrollmechanismen beinhalten, die es seinen Benutzern erlauben, die Genauigkeit der über sie gespeicherten Positionsinformationen insgesamt zu steuern und andererseits zu bestimmen, welche Benutzer bzw. Benutzergruppen mit welcher Genauigkeit auf die Positionsinformationen zugreifen dürfen.

Die Sicherheitsrelevanz einer Positionsinformation hängt dabei, wie ihr Informationsgehalt auch, von ihrer Genauigkeit ab. Während ein Benutzer es zulassen mag, dass seine Kollegen erfahren, ob er sich an seiner Arbeitsstelle aufhält oder zu Hause, will er vielleicht nicht, dass sie erfahren, wo genau er sich innerhalb des Bürogebäudes befindet (z. B. in der Kaffeeecke anstatt in seinem Büro). Die im Folgenden beschriebenen Schutzmechanismen sollen daher den Benutzern erlauben, den gewünschten Grad an Sicherheit anhand der Genauigkeit der Positionsinformationen (siehe auch LEONHARDT (1998)) vorzugeben.

In diesem Kapitel wird ein einfaches Konzept vorgestellt, das die Sicherheits- und Datenschutzaspekte im LS adressiert. Dies betrifft vor allem Mechanismen für eine Kontrolle des Zugriffs auf die im LS gespeicherten Positionsinformationen. Innerhalb des Nexus-Projekts werden die Sicherheitsaspekte von den Projektpartnern am Institut für Kommunikationsnetze und Rechnersysteme (IKR) behandelt (siehe HAUSER & KABATNIK (2001) und HAUSER, LEONHARDI & KÜHN (2002)). Die hier beschriebenen Mechanismen sind Ergebnisse einer entsprechenden Zusammenarbeit. Zukünftige Arbeiten am IKR haben die weiterführende Untersuchung von Schutzmechanismen für Positionsinformationen zum Ziel.

8.1 Sicherheitsinfrastruktur

Für die im Folgenden beschriebenen Mechanismen wird eine Sicherheitsinfrastruktur vorausgesetzt, die auf asymmetrischen Schlüsseln basiert (engl. *public key infrastructure*, kurz PKI). Jede beteiligte Komponente hat dazu sowohl einen geheimen Schlüssel, der nur ihr selbst bekannt ist, als auch einen öffentlichen Schlüssel, den die anderen beteiligten Komponenten kennen. Die Kommunikation zwischen mobilen Objekten, Klienten und dem LS wird mit deren Hilfe verschlüsselt, um die Geheimhaltung und Integrität der Kommunikation zu gewährleisten und zu verhindern, dass ein Angreifer die ausgetauschten Positionsinformationen durch Abhören der Kommunikation ausspähen oder verfälschen kann (siehe z. B. STALLINGS (1995)).

Die PKI erlaubt weiterhin eine Authentifizierung der an einer Kommunikation beteiligten Parteien, die beispielsweise verhindert, dass sich ein Angreifer fälschlicherweise als ein berechtigter Klient oder ein mobiles Objekt ausgibt. In diesem Zusammenhang muss die Zugehörigkeit eines öffentlichen Schlüssels zu einer bestimmten Person oder einem bestimmten Dienst garantiert sein. Zu diesem Zweck werden eine oder mehrere sogenannte vertrauenswürdige dritte Par-

teien (engl. *trusted third party*, kurz TTP) benötigt, die diese Zuordnung kennen und zertifizieren. Über eine TTP kann auch eine Abrechnung für im Zusammenhang mit der Verwendung des LS in Anspruch genommenen Dienste durchgeführt werden, ohne dass die wirkliche Identität des Zahlenden bekannt sein muss.

8.2 Einschränkungen gegenüber dem Lokationsdienst

Gegenüber dem LS soll das Prinzip der Datensparsamkeit berücksichtigt werden (siehe MÜLLER & RANNENBERG (1999)), d. h. es werden dort nur die Daten gespeichert, die auch wirklich benötigt werden. So ist es nicht notwendig, die Positionsinformation eines Benutzers im LS unter dessen vollständiger Identität zu speichern; statt dessen wird ein Pseudonym *P* verwendet, das von einer TTP vergeben wird, die auch die Identität des dazugehörenden Benutzers kennt. Ein Benutzer meldet sich unter diesem Pseudonym beim LS an, das dort dann dem eindeutigem Objektbezeichner *oID* entspricht. Ortsbezogene Dienste und Personen, denen der Benutzer den Zugriff auf seine Positionsinformationen erlauben will, übergibt er eine Referenz auf dieses Pseudonym, gegebenenfalls zusammen mit einem entsprechenden Autorisierungszertifikat (siehe unten). Über die Referenz kann daraufhin auf die vom LS für den Benutzer verwaltete Positionsinformation zugegriffen werden.

Wie schon in Kapitel 4 beschrieben und in den Schnittstellen, der Architektur sowie den Mechanismen des LS berücksichtigt, kann der Benutzer zusätzlich vorgeben mit welcher Genauigkeit die Positionsinformation über ihn im LS gespeichert sein soll. Der LS erhält die Positionsinformation dann nur noch mit der eingestellten Genauigkeit übermittelt. Die Aktualisierungsprotokolle, die zur Übertragung der Positionsinformationen mit einer vorgegebenen Genauigkeit benötigt werden, wurden in Kapitel 6 beschrieben. Bei den dort betrachteten Protokollen werden zwar die Positionsinformationen bei der Übertragung mit der Genauigkeit des Positionierungssensors übermittelt, allerdings können auch hier vor der Übertragung die in Abschnitt 8.4 beschriebenen Mechanismen zur Umrechnung in ungenauere Positionsinformationen eingesetzt werden. Dies funktioniert natürlich nur, wenn ein lokales Positionierungssystem verwendet wird, das sich unter der Kontrolle des Benutzers befindet.

Bei einem externen Positionierungssystem ist es aus Datenschutzgesichtspunkten wünschenswert, dass dieses von einem anderen Betreiber kontrolliert wird, als der LS selbst. Ein Benutzer muss darauf vertrauen können, dass dessen Betreiber die Positionsinformationen nicht mit einer höheren Genauigkeit an den LS weitergibt, als vereinbart. Dies kann durch entsprechende Vertragsbedingungen und gegebenfalls durch darin vereinbarte Sanktionen geregelt werden. Dies bedeutet, dass hier nicht technische, sondern einzig vertragliche Maßnahmen erfolgverspre-

chend sind. Lokale Positionierungssysteme sind aus diesem Grund gegenüber externen zu bevorzugen.

8.3 Mechanismen zur Zugriffskontrolle

Um eine flexible Zugriffskontrolle auf die im LS gespeicherten Positionsinformationen zu ermöglichen (vgl. HAUSER & KABATNIK (2001)), beruhen die folgenden Mechanismen auf dem in ELLISON (1999) und AURA (1999) vorgeschlagenen Konzept eines sogenannten Autorisierungszertifikats (engl. *authorization certificate*). Dieses wird von einer registrierenden Instanz für die von ihr verwalteten mobilen Objekte, gegebenenfalls nach Rückfrage mit dem Benutzer oder einem Systemverwalter, vergeben. Bei der registrierenden Instanz wird es sich, wie bereits erwähnt, in vielen Fällen um das mobile Endgerät des Benutzers selbst handeln. Im Fall eines externen Positionierungssystems oder sehr einfachen mobilen Endgeräten (wie z. B. den in HOLMQUIST ET AL. (2001) beschriebenen *Smart-its*), die nicht direkt einem Benutzer zugeordnet sind, kann die registrierende Instanz aber auch eine zentrale Verwaltungseinheit sein, die von einem Systemverwalter administriert wird.

Ein Autorisierungszertifikat hat das im Folgenden beschriebene Format und ist mit dem geheimen Schlüssel der registrierenden Instanz signiert. Um das Autorisierungszertifikat geheim zu halten, kann es mit dem bekannten öffentlichen Schlüssel des LS verschlüsselt werden.

authCert = { *oID, (**all** | clientId | groupId),*
maxAcc, startDate, expirationDate,
*[**noPositionQuery** ,] [**noRangeQuery** ,] [**noNeighbourQuery** ,]*
Signatur durch K_{RI}^{-1} } verschlüsselt mit K_{LS}

K_{RI}^{-1}: geheimer Schlüssel der registrierenden Instanz

K_{LS}: öffentlicher Schlüssel des LS

Ein Autorisierungszertifikat erlaubt es allgemeinen Klienten (beschrieben durch das Schlüsselwort ***all***), einem bestimmten Klienten (beschrieben durch dessen Pseudonym, *clientId*) oder einer Gruppe (*groupId*) mit der angegebenen Genauigkeit (*maxAcc*) auf die Positionsinformationen eines bestimmten mobilen Objekts (*oID*) zuzugreifen. Spezifischere Einstellungen, z. B. einen Klienten betreffend, überschreiben dabei allgemeinere, die eine Gruppe oder beliebige Klienten betreffen.

Einerseits lässt sich hier einstellen, dass die Positionsinformationen nur mit einer bestimmten Genauigkeit, die geringer sein kann als die vom LS angebotene Genauigkeit *(maxAcc ≥ offeredAcc*), zurückgegeben werden sollen.

Andererseits sind auch die grundsätzlich unterschiedlichen Anfrageklassen berücksichtigt. Um eine Positionsanfrage durchführen zu können, muss ein Klient den Bezeichner für das betreffende Objekt kennen. Mit Hilfe von Gebiets- und Nachbarschaftsanfragen können dagegen die Pseudonyme (*oID*) für bisher unbekannte mobile Objekte ausgehend von einer Position oder einem Gebiet gefunden werden. Die optionalen Parameter ***noPositionQuery***, ***noRangeQuery*** und ***noNeighbourQuery*** bieten daher noch die Möglichkeit, bestimmte Klassen von Anfragen für das mobile Objekt einzuschränken. Im ersten Fall wird bei einem Versuch auf die Positionsinformationen zuzugreifen eine entsprechende Fehlermeldung zurückgegeben. Bei einer Gebiets- oder Nachbarschaftsanfrage ist das mobile Objekt in der Ergebnismenge statt unter seinem eindeutigen Pseudonym unter einem allgemeinen anonymen Pseudonym (oID_{anon}) enthalten. Dies erlaubt es einer Anwendung zwar zu bestimmen, dass sich mobile Objekte in einem gewissen Gebiet aufhalten bzw. dass sich ein mobiles Objekt in der Nähe einer gewissen Position befindet, sie kann aber nicht über deren Pseudonyme Rückschlüsse auf deren Identität ziehen. Dies würde es z. B. dem Betreiber eines Messestandes ermöglichen, über eine periodische Gebietsanfrage zu bestimmen, wieviele Besucher sich für seinen Stand interessieren und gegebenenfalls, welche Exponate von besonderem Interesse sind. Er kann aber die Ergebnisse nicht verwenden, um die Identität der Besucher festzustellen und ihnen Werbung zuzusenden.

Schließlich enthält ein Autorisierungszertifikat den Zeitpunkt (*startDate*) an dem es gültig wird und den Zeitpunkt an dem es seine Gültigkeit verliert (*expirationDate*).

Bei der Registrierung eines mobilen Objekts beim LS (siehe Abschnitt 4.2) wird eine Zugriffskontrollliste (engl. *access control list*, kurz ACL) übergeben, die eine Menge von vorab durch die registrierende Instanz vergebenen Autorisierungszertifikaten darstellt. Sie wird vom LS als Teil der Registrierungsinformationen gespeichert und kann jederzeit durch die registrierende Instanz wieder geändert werden. Jede Zugriffskontrollliste sollte dabei ein Autorisierungszertifikat für allgemeine Benutzer enthalten, um festzulegen, wie sich der LS gegenüber unbekannten Anfragenden verhalten soll.

Die zwei möglichen Abläufe einer Anfrage an den LS sind in Abbildung 8-1 gezeigt. Eine Anfrage an den LS enthält im einfacheren Fall (Abbildung 8-1 (a)), neben dem Typ der Anfrage und den benötigten Parametern, eine Signatur, welche die Identität des Anfragenden bezeugt. Wenn der LS eine solche Anfrage bearbeitet, bestimmt er anhand der Identität des Anfragenden und der betroffenen mobilen Objekte die zutreffenden Autorisierungszertifikate aus seinen Zugriffskontrolllisten. Bei der Zusammenstellung der Ergebnismenge berücksichtigt er die in die-

sen enthaltenen Einschränkungen, wobei er gegebenenfalls Positionsinformationen, die ihm mit einer höheren Genauigkeit vorliegen, entsprechend in ungenauere Positionsinformationen umrechnen muss (siehe nächster Abschnitt).

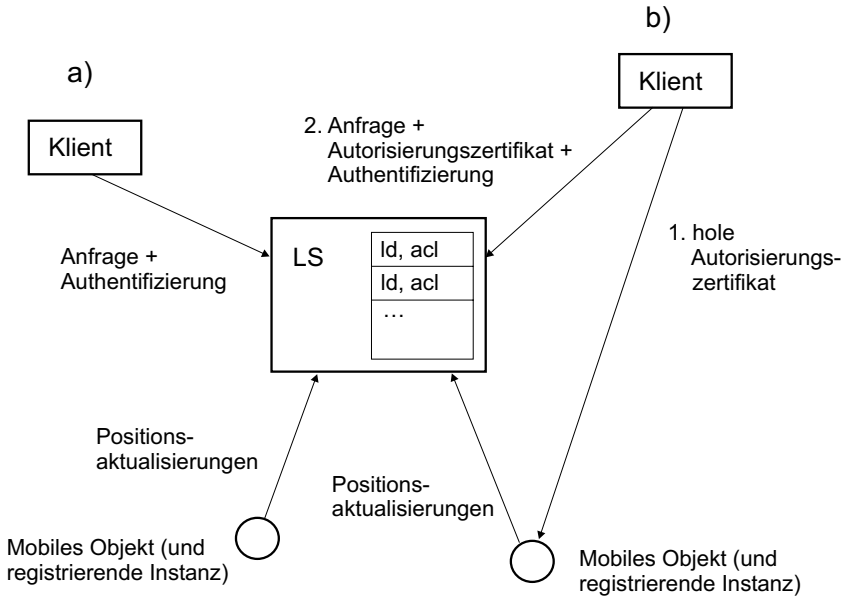


Abbildung 8-1. Sicherheitsmechanismen zur Zugriffskontrolle auf die im LS gespeicherten Positionsinformationen.

Um die Flexibilität der Zugriffskontrolle zu erhöhen und damit die Zugriffskontrolllisten nicht zu umfangreich werden, kann ein Klient ein Autorisierungszertifikat auch direkt bei der registrierenden Instanz erfragen und der entsprechenden Anfrage mitgeben (Abbildung 8-1 (b)). Da die Zertifikate von der jeweiligen registrierenden Instanz zertifiziert werden, ist es egal, auf welchem Weg sie zum LS gelangen. Der LS überprüft daraufhin den Typ der Anfrage, die betroffenen mobilen Objekte und die Identität des Anfragenden nicht nur mit den Angaben in der ACL, sondern auch mit den Angaben in den der Anfrage mitgegebenen Autorisierungszertifikaten. Aus diesem Grund enthält ein Autorisierungszertifikat auch den Bezeichner des mobilen Objekts, was allein für die Zugriffskontrolllisten, die bei der Registrierung des betreffenden Objekte festgelegt werden, nicht notwendig wäre. Um die Kommunikation zwischen Klienten und Lokations-Servern geheim zu halten, kann die Anfrage mit dem bekannten öffentlichen Schlüssel des LS verschlüsselt werden. Dieser verschlüsselt die Antwort dann wiederum mit dem öf-

fentlichen Schlüssel des jeweiligen Klienten. Insgesamt hat eine Anfrage an den LS daher das folgende Format:

query = (*queryType, parameters,*
[authCert,]
Signatur durch K_{Cl}^{-1}) verschlüsselt mit K_{LS}

K_{Cl}^{-1}: geheimer Schlüssel des Klienten

K_{LS}: öffentlicher Schlüssel des LS

Autorisierungszertifikate, die der LS über die Anfragen erhält, kann er in seiner Zugriffskontrollliste zwischenspeichern, um den Aufwand für nachfolgende Anfragen zu veringern.

8.4 Sicherheitsbedingte Verringerung der Genauigkeit von Positionsinformationen

Die in den vorherigen Abschnitten besprochenen Mechanismen zur Kontrolle des Zugriffs auf Positionsinformationen setzen in vielen Fällen voraus, dass einem Klienten bzw. dem LS Positionsinformationen nur mit einer vorgegebenen geringeren Genauigkeit übergeben werden (beschrieben durch den Lokationsbezeichner ld_{gen}), obwohl diese ursprünglich mit höherer Genauigkeit vorliegen (ld_{orig}, mit $ld_{gen}.acc \geq ld_{orig}.acc$). Daraus ergibt sich das Problem, die Positionsinformationen so umzurechnen, dass diese insgesamt nur noch die geforderte Genauigkeit hat, ohne dass ein Angreifer auf die höhere Genauigkeit zurückschließen kann.

Zwei grundlegende Möglichkeiten für eine solche Umrechnung sind in Abbildung 8-2 gezeigt. Im ersten Fall wird aus einem Gitter jeweils das Quadrat zurückgegeben, in dem sich die im LS gespeicherte genauere Position des mobilen Objekts befindet. Das Gitter hat dabei einen Linienabstand, welcher der gewünschten Genauigkeit entspricht und einen feststehenden Ursprung. Nachteil dieser Umrechnung ist, dass ein Angreifer durch Überwachen der zurückgelieferten Positionen den Zeitpunkt ermitteln kann, an dem ein mobiles Objekt eine Gitterlinie überschreitet. In Kombination mit verschiedenen Genauigkeitsstufen ergeben sich dadurch Angriffspunkte für ein Rückschließen auf die ursprünglichen genaueren Positionsinformationen.

Der zweite Ansatz beruht darauf, dass ein kreisförmiges Aufenthaltsgebiet mit dem Radius der gewünschten Genauigkeit erzeugt wird, welches das ursprüngliche Aufenthaltsgebiet vollständig enthält und welches einen zufällig gewählten Mittelpunkt hat. Um diesen zu bestimmen,

wird auf jede Koordinate des Mittelpunkts des ursprünglichen Aufenthaltsgebiets eine Zufallszahl aus dem Bereich [0, $ld_{gen}.acc$ - $ld_{orig}.acc$] aufaddiert. Durch den Einsatz von kryptographisch sicheren Zufallszahlengeneratoren, bei denen im Gegensatz zu üblichen Zufallszahlengeneratoren kein Rückschließen auf den erzeugenden Algorithmus durch Beobachten einer Folge von erzeugten Zufallszahlen möglich ist (siehe z. B. SCHNEIER (1995)), kann verhindert werden, dass sich ein Angreifer Regelmäßigkeiten eines Zufallszahlengenerators für einen Angriff zu Nutze macht. Wenn ein Angreifer allerdings die Möglichkeit hat, die Positionsinformationen innerhalb kurzer Zeit sehr häufig abzurufen (z. B. indem er mit anderen Angreifern zusammenarbeitet), kann er das ursprüngliche Aufenthaltsgebiet einfach durch Übereinanderlegen und Mitteln der zurückgegebenen Aufenthaltsgebiete bestimmen.

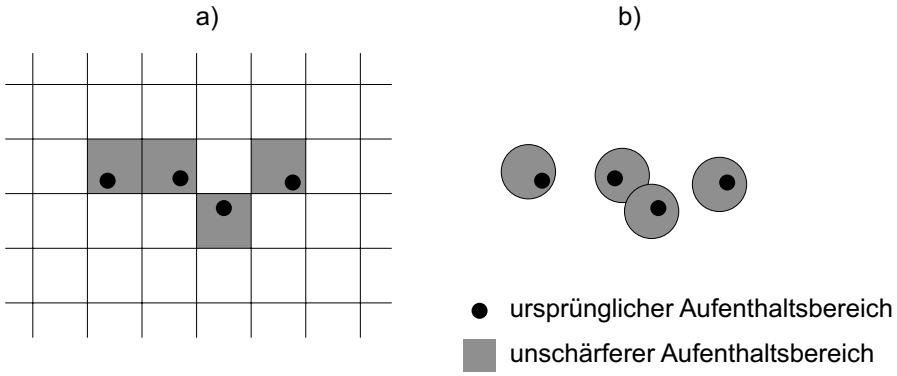


Abbildung 8-2. Zwei grundlegende Mechanismen um genaue in ungenauere Positionsinformationen umzurechnen.

Um dies zu verhindern, wird ein sogenannter zeitabhängiger Zufallszahlengenerator verwendet, bei dem der Betrag, um den sich zwei aufeinander folgende Zufallszahlen unterscheiden, von der zwischen deren Erzeugung vergangenen Zeit abhängt (siehe HATAMIAN (1995)). Die Stärke der Abhängigkeit, d. h. um welchen Betrag sich die Zufallszahlen innerhalb eines bestimmten Zeitabschnitts maximal verändern, kann durch verschiedene Parametereinstellungen bestimmt werden.

Wenn der Zufallszahlengenerator *tdran(t, param)* eine solche, von der aktuellen Zeit *t* abhängige Zufallszahl aus dem Bereich [0, 1] erzeugt, kann der Mittelpunkt für den ungenaueren Lokationsbezeichner ld_{gen} folgendermaßen aus dem originalen Lokationsbezeichner ld_{orig} errechnet werden:

$$ld_{gen}.pos.x := ld_{orig}.pos.x + (ld_{gen}.acc - ld_{orig}.acc) \cdot (2 \cdot tdran(t, param) - 1)$$

$$ld_{gen}.pos.y := ld_{orig}.pos.y + (ld_{gen}.acc - ld_{orig}.acc) \cdot (2 \cdot tdran(t, param) - 1)$$

Die Ausgabe eines zeitabhängigen Zufallszahlengenerators ist in Abbildung 8-3 gezeigt. Dadurch dass die Positionsinformation innerhalb eines kurzen Zeitintervalls nur noch geringfügig verändert wird, ist die oben beschriebene Art des Angriffs nicht mehr möglich. Ein Angreifer müsste die Positionsinformation über eine lange Zeit hinweg beobachten, um auf die wirkliche Position des mobilen Objekts zurückschließen zu können (dieser Zeitraum ist abhängig von den Parametereinstellungen des Zufallszahlengenerators). Innerhalb dieses Zeitraums kann er allerdings nicht mehr zwischen einer wirklichen Bewegung des mobilen Objekts und einer durch den Zufallszahlengenerator aufaddierten Veränderung unterscheiden.

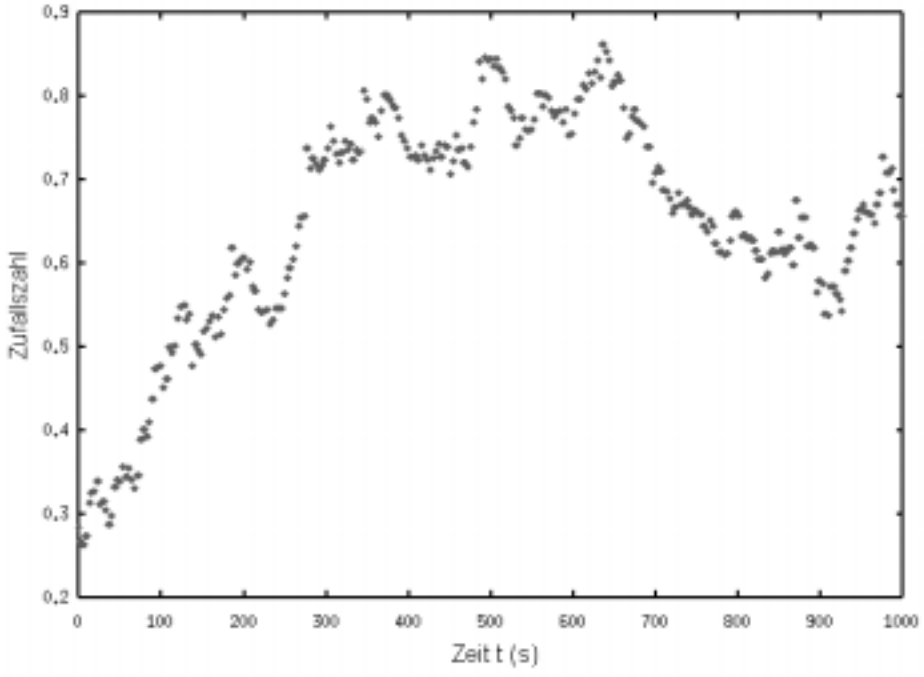


Abbildung 8-3. Funktionsweise eines zeitabhängigen Zufallszahlengenerators.

8.5 Zusammenfassung und Erweiterungsmöglichkeiten

In diesem Kapitel wurden die Sicherheitsaspekte behandelt, die sich aus den im LS gespeicherten Positionsinformationen ergeben. Es wurde dazu ein Konzept für eine einfache Zugriffskontrolle vorgestellt, welche es den jeweiligen Benutzern erlaubt, zu bestimmen, welche Personen oder Dienste Informationen über ihre Position beim LS abfragen dürfen. Die Möglichkeit, anhand der Genauigkeit, mit der einzelne Klienten auf die Positionsinformationen zugreifen dürfen, einen unterschiedlichen Grad an Vertrauen auszudrücken, bietet die Grundlage für eine flexible Vergabe von Zugriffsrechten. Die hier angestellten einfachen Betrachtungen zeigen, dass

eine benutzergesteuerte flexible Zugriffskontrolle für die Positionsinformationen im LS möglich ist.

Über die hier beschriebenen Zugriffskontrollen für Positionsinformationen hinaus sind weitergehende Mechanismen zur Zugriffskontrolle denkbar, die auch die aktuelle Position des angefragten mobilen Objekts und evtl. die Position des Anfragenden mit einbeziehen. Ein Anpassen der Sicherheitsanforderungen an die Position des Objekts kann mit den oben beschriebenen Mechanismen zwar dadurch erreicht werden, dass die Zugriffskontrollliste entsprechend dessen aktueller Position jeweils neu gesetzt wird. Da dies jedoch bedeutet, dass die registrierende Instanz die Position des mobilen Objekts ständig überwachen muss und sich die Zugriffskontrolllisten häufig ändern, wäre hier eine flexiblere Definition der Zugriffskontrollanforderungen wünschenswert. Entsprechende Mechanismen werden zum Beispiel in LEONHARDT (1998) diskutiert, gehen allerdings über den Rahmen dieser Arbeit hinaus.

In HAUSER & KABATNIK (2001) wird weiterhin vorgeschlagen, dass ein Benutzer seine Positionsinformation unter verschiedenen Pseudonymen (z. B. $P_1, P_2 \neq P$) bekannt machen kann, um zu verhindern, dass durch Kombination der Ergebnisse unterschiedlicher Anfragen und die Kooperation von verschiedenen Klienten Wissen über ihn angesammelt wird. Der Benutzer sollte daher angeben können, welche Klienten auf welche Informationen unter einem bestimmten Pseudonym zugreifen können. Die Pseudonyme sind so beschaffen, dass nur der LS und nicht die Klienten bestimmen können, ob zwei Pseudonyme dasselbe mobile Objekt beschreiben. Darüber hinaus ist angedacht, dem Benutzer eine Rückmeldung darüber zu geben, welche Informationen bereits über ein Pseudonym verfügbar geworden sind und ihm zu ermöglichen, dieses Pseudonym gegebenfalls wieder zu löschen. Dies entspricht dem in JENDRICKE & GERD TOM MARKOTTEN (2000) oder CLAUSS & KÖHNTOPP (2001) beschriebenen Konzept eines Identitätsmanagers. Davon abgeleitete Mechanismen für den LS werden gegenwärtig von unseren Projektpartnern am IKR konzipiert.

Kapitel 9

Resümee

9.1 Zusammenfassung

In dieser Arbeit wurden die für einen verteilten skalierbaren Lokationsdienst benötigten Konzepte behandelt, der für die Verwaltung sehr genauer und damit sehr dynamischer Positionsinformationen einer großen Anzahl mobiler Objekte geeignet ist. Der Zugriff auf solch genaue Positionsinformationen ist eine wichtige Voraussetzung für die einfache Entwicklung von fortgeschrittenen Anwendungen mit Ortsbezug. Eine Betrachtung bisheriger Ansätze hat gezeigt, dass kein existierendes System die an den Lokationsdienst gestellten speziellen Anforderungen in zufriedenstellender Weise erfüllen kann.

In einem ersten Schritt wurde die Funktionalität des Lokationsdienstes betrachtet, ausgehend davon wie Positionsinformationen in Anwendungen mit Ortsbezug typischerweise verwendet werden. Als Ergebnis ist ein Dienstmodell für den gewünschten Lokationsdienst entstanden. Hervorzuheben ist besonders die in den Semantiken der vom LS angebotenen Operationen berücksichtigten unterschiedlichen Genauigkeiten der Positionsinformationen, die sowohl durch unterschiedliche Sensorsysteme als auch durch verschieden starke Sicherheitsanforderungen der Benutzer bedingt sind.

Um die in diesem Dienstmodell geforderte Funktionalität auf eine skalierbare Art und Weise realisieren zu können, wird für den LS eine hierarchisch verteilte Architektur vorgeschlagen. In der hier betrachteten Architektur ist jeweils ein Blatt-Lokations-Server für ein disjunktes geographisches Dienstgebiet zuständig und verwaltet die Registrierungs- und Positionsinformationen für die sich in diesem befindenden mobilen Objekte. Eine Zuständigkeitsübergabe muss stattfinden, wenn ein mobiles Objekt aus diesem Dienstgebiet in das eines anderen Blatt-Lokations-Servers wechselt. Eine den Blatt-Servern übergeordnete hierarchische Struktur von Loka-

tions-Servern sorgt dafür, dass die für die Bearbeitung einer Anfrage benötigten Blatt-Server gefunden werden. Die Algorithmen zur Durchführung der Registrierung mobiler Objekte sowie der Bearbeitung von Positionsaktualisierungen, Positions-, Gebiets- und Nachbarschaftsanfragen, wurden in dieser Arbeit im Detail beschrieben.

Als Teil der Architektur des LS wurde eine spezielle Organisation der Datenhaltungskomponente für Positionsinformationen innerhalb eines Lokations-Servers vorgeschlagen. Bei dieser werden die dynamischen Positionsinformationen in einer Hauptspeicherdatenbank gehalten, während die umfangreicheren Registrierungsinformationen und die Suchverweise in einer Standarddatenbank gespeichert sind. Dadurch wird es möglich, vor allem Positionsaktualisierungen sehr effizient zu bearbeiten. Das Verfahren zur Wiederherstellung der Hauptspeicherdatenbank nach einem Systemausfall, verwendet dazu die von den mobilen Objekten periodisch gesendeten Positionsaktualisierungsnachrichten.

Messungen an einer prototypischen Implementierung des LS haben die Machbarkeit und Leistungsfähigkeit der Konzepte für die Datenhaltungskomponente und die verteilten Algorithmen gezeigt. Ein prototypischer Lokations-Server ist so bereits auf einem einfachen PC in der Lage, mehr als 500 Positionsaktualisierungen und Anfragen in der Sekunde zu bearbeiten.

Ausführlich betrachtet wurden ebenfalls die für die Übertragung von sehr genauen Positionsinformationen benötigten Protokolle. So wurden grundlegende Protokollalternativen beschrieben und ihre Eigenschaften sowohl analytisch als auch durch Simulationen für verschiedene typische Bewegungscharakteristika mobiler Objekte verglichen. In der Klasse der Koppelnavigationsprotokolle wurde darüber hinaus ein neuartiges kartenbasiertes Protokoll vorgestellt, das gegenüber einem einfachen Protokoll bis zu 91% der benötigten Aktualisierungsnachrichten einsparen kann.

Schließlich wurden Sicherheits- und Datenschutzaspekte behandelt, die sich aus einen Lokationsdienst ergeben, der den aktuellen Aufenthaltsort von Personen speichern soll. Deren Adressierung ist für die Akzeptanz eines solchen Dienstes von großer Bedeutung. Aufbauend auf einer geforderten Sicherheitsinfrastruktur, die für die Geheimhaltung, Integrität und Authentizität der übertragenen Nachrichten zuständig ist, wurden Mechanismen zur Kontrolle des Zugriffs auf Positionsinformationen vorgestellt. Durch die Vergabe von Autorisierungszertifikaten kann ein Benutzer festlegen, welche Klienten mit welcher Genauigkeit auf die über ihn gespeicherten Positionsinformationen zugreifen dürfen.

9.2 Bewertung

Die in dieser Arbeit beschriebenen Konzepte und unsere Experimente mit einer prototypischen Implementierung des LS zeigen, dass ein Lokationsdienst realisierbar ist, der die Positionsinformationen einer Vielzahl von mobilen Objekten mit einer sehr hohen Genauigkeit verwalten kann. Wir erwarten, dass die Verfügbarkeit eines solchen Dienstes die Entwicklung von ortsbezogenen Anwendungen, die auf die Funktionalität eines solchen Dienstes aufbauen können, deutlich vereinfacht und daher auch zu einer weiteren Verbreitung solcher Anwendungen führen wird.

Der hier beschriebene Lokationsdienst kann zum einen als generische Komponente für ortsbezogene Anwendungen, beispielsweise in einem ortsbezogenen Stadtführer oder in einer Telematikanwendung zur Verkehrsüberwachung eingesetzt werden. Zum anderen kann er auch als Teil der Infrastruktur eines Mobilkommunikationssystems bereitgestellt werden, wo er von verschiedenen Anwendungen mit Ortsbezug, die innerhalb dieses Mobilkommunikationssystems angeboten werden, genutzt werden kann.

Für einen globalen Einsatz des LS, ähnlich dem des DNS im Internet, müssen noch organisatorische Fragen zum Betrieb des Dienstes geklärt werden, d. h. wie die administrativen Zuständigkeiten für die einzelnen Lokations-Server zwischen verschiedenen Betreibern aufgeteilt werden können. So erlaubt die hierarchische Architektur des LS, dass jeweils ein Betreiber für einen bestimmten Teilbaum des LS zuständig ist. Dieser kann einzelne Unterteilbäume dann wiederum an andere Betreiber weitervergeben. Wie beim DNS ist damit die Verwaltung des (verteilten) Wurzel-Servers von großer Bedeutung, da dort sämtliche mobile Objekte registriert sein müssen. Voraussetzung für einen globalen Einsatz ist auch eine Verwendung der in Abschnitt 5.5 beschriebenen Mechanismen zum Zwischenspeichern von Informationen, die ein häufiges Traversieren der Server-Hierarchie vermeiden.

Eine wichtige Einschränkung des LS ist, dass er in der hier diskutierten Form ganz auf die Verwaltung der Positionsinformationen mobiler Objekte ausgelegt ist. Eine Anwendung mit Ortsbezug, die den LS nutzen will, wird daher für die meiste anzubietende Funktionalität zusätzlich auf weitere Komponenten, wie eine Quelle für digitale Karten oder einen Verzeichnisdienst, zugreifen müssen. Diese Trennung ermöglicht zwar eine sehr gezielte Optimierung des LS hinsichtlich der Eigenschaften von Positionsinformationen und erlaubt es eine ortsbezogene Anwendung gezielt aus den Komponenten aufzubauen, die für sie benötigt werden, erfordert aber ein getrenntes Ansprechen dieser Komponenten und verhindert eine komponentenübergreifende Optimierung (siehe den Ausblick im folgenden Abschnitt).

Protokolle zur Übertragung von Positionsinformationen wurden in dieser Arbeit, über ihre Anwendung im LS hinaus, ausführlich betrachtet und untersucht. Die hier angestellten Betrachtungen zu Eigenschaften, Effektivität und Effizienz dieser Protokolle lassen sich überall dort anwenden, wo Positionsinformationen mit einer hohen Genauigkeit kontinuierlich übertragen werden sollen. Anhand der Koppelnavigationsprotokolle konnte ferner gezeigt werden, dass es durch ein geeignetes Übertragungsprotokoll möglich ist, die Anzahl der zur Übertragung von Positionsinformationen mit einer bestimmten Genauigkeit benötigten Nachrichten deutlich zu reduzieren. Das kartenbasierte Koppelnavigationsprotokoll eignet sich dabei besonders für den Einsatz in Telematikanwendungen auf Fahrzeugen, da es unter den in diesen Anwendungen gegebenen Voraussetzungen ein sehr hohes Einsparpotenzial bietet.

Die in Kapitel 8 vorgestellten Mechanismen für die Kontrolle des Zugriffs auf die im LS gespeicherten Positionsinformationen bieten zwar noch nicht die maximal mögliche Flexibilität bei der Vergabe von Zugriffsrechten, sie zeigen jedoch schon, dass es möglich ist, eine Zugriffskontrolle für den LS zu realisieren, die den Benutzern eine flexible Vergabe von Zugriffsrechten auf Basis unterschiedlicher Genauigkeiten erlaubt. Die hier beschriebenen Sicherheitsmechanismen sollten – ein entsprechendes Vertrauen in den Betreiber des Dienstes vorausgesetzt – ausreichen, um Sicherheitsbedenken hinsichtlich der Herausgabe von Positionsinformationen an den LS zu zerstreuen.

9.3 Ausblick

Wie in Abschnitt 7.5 besprochen konnten in dieser Arbeit nur Anhaltspunkte dafür gegeben werden, wie eine optimale Konfiguration einer Server-Hierarchie des LS für ein gegebenes Einsatzgebiet aussehen muss. Eine weitere Untersuchung der Auswirkung dieser Konfiguration auf die Leistungsfähigkeit des LS sollte daher ein wichtiges Ziel weiterer Arbeiten sein. Für eine entsprechende Untersuchung ist vor allem ein detaillierteres Modell für die Bewegungen der beim LS angemeldeten mobilen Objekte und für die Anfragen ortsbezogener Anwendungen zu erstellen. Dieses Modell muss hinsichtlich der für die Einsatzumgebung des LS wichtigen Eigenschaften, wie die durchschnittliche Geschwindigkeit mobiler Objekte oder die Lokalität von Anfragen, parametrisierbar sein. Zu diesem Zweck werden geeignete Metriken für die entsprechenden Größen benötigt. Ausgehend von diesem Modell sind in einer geeigneten Testumgebung verschiedene Konfigurationen des LS hinsichtlich Durchsatz und Antwortzeit der Operationen zu untersuchen, um abhängig von den Umgebungsparametern die Auswirkung der Konfigurationsmöglichkeiten (z.B. Höhe und Verzweigungsgrad) auf die Leistungsfähigkeit des LS zu ermitteln.

Ziel einer Erweiterung des LS über die hier betrachtete grundlegende Funktionalität hinaus ist ein Ausbau von dessen API, um Anwendungen umfangreichere und komfortablere Zugriffsmöglichkeiten auf die im LS gespeicherten Positionsinformation zu bieten, und eine weitere Optimierung der Mechanismen des LS, um dessen Leistungsfähigkeit zu erhöhen. Neben einer Erweiterung der in Abschnitt 4.3 beschriebenen Anfrageschnittstelle um eine Einbeziehung von statischen Daten (siehe unten) ist, wie bereits in Abschnitt 4.4 erwähnt, vor allem eine Unterstützung für die Benachrichtigung über eingetretene Ereignisse sinnvoll. Um das Eintreten dieser räumlichen Ereignisse effizient beobachten zu können, müssen entsprechende Mechanismen in die Datenhaltungskomponente des LS integriert werden, die sich ebenfalls den mehrdimensionalen Index zu Nutze machen. Da dies alleine nur die Beobachtung von Ereignissen auf einem einzelnen Lokations-Server erlaubt, muss für Ereignisse, die das Dienstgebiet mehrerer Lokations-Server betreffen bzw. die auch statische Daten miteinbeziehen, eine Möglichkeit für eine verteilte Beobachtung von Ereignissen geschaffen werden. Ein Bespiel für ein solches Ereignis ist die Benachrichtigung, dass der Benutzer gerade an einem Schuhgeschäft vorbeigekommen ist.

Ziel weiterführender Arbeiten sollte auch eine Optimierung der verteilten Algorithmen des LS durch die in Abschnitt 5.5 angesprochenen Mechanismen zum Zwischenspeichern von Informationen sein. Ähnliche Mechanismen wurden auch schon im Zusammenhang mit der Lokationsverwaltung in Mobilfunknetzen untersucht. Eine genauere Betrachtung verdienen dabei Mechanismen bei denen die Positionsinformationen nicht nur auf den Blatt-Servern der Server-Hierarchie gespeichert werden, sondern auch, mit einer geringeren Genauigkeit, auf den inneren Lokations-Servern. Eine Anfrage, in der nur eine geringe Genauigkeit der Positionsinformationen gefordert wird, muss dabei nicht mehr vollständig durch die Server-Hierarchie bis zu den betreffenden Blatt-Servern weitergeleitet werden, sondern wird schon vorher von einem zentraler gelegenen inneren Server bearbeitet. Bei Gebiets- und Nachbarschaftsanfragen kann der zusätzliche Vorteil auftreten, dass die Anfrage so von weniger Servern bearbeitet werden muss, als bei der Weiterleitung bis zu den Blatt-Servern. Allerdings hängt die Wirkung dieser Mechanismen stark vom Anfrageverhalten der Klienten des LS ab. Sie wurden daher nicht in die allgemeinere Betrachtung der Architektur mit aufgenommen.

Bei den Betrachtungen in dieser Arbeit stand immer die Verwaltung von Positionsinformationen im Vordergrund. Das Zusammenspiel mit statischen nicht-räumlichen Daten, die für eine genauere Beschreibung der gesuchten mobilen Objekte benötigt werden (wie z. B. zum Auffinden des nächstgelegenen Taxis), wurde hier nicht speziell behandelt. Bisher müssen solche, gegebenfalls umfangreichere Daten über Eigenschaften und Gestalt eines mobilen Objekts bei einer zweiten, separaten Datenbasis, wie einem Verzeichnisdienst oder einer Datenbank, abge-

fragt werden. Innerhalb des Projekts Nexus sind dies z. B. spezielle Umgebungsmodell-Daten-Server. Die Föderationskomponente fragt für eine Anfrage die dazu benötigten Teilinformationen bei den betroffenen Umgebungsmodell-Daten-Servern sowie dem LS ab. Aus den Teilergebnissen setzt sie schließlich die Anfrageergebnisse zusammen.

Besonders bei der Bearbeitung von Nachbarschaftsanfragen und bei der Behandlung von Sicherheitsmechanismen hat sich allerdings gezeigt, dass eine integrierte Verwaltung von Positionsinformationen und statischen Daten sinnvoll wäre. Damit werden u. a. flexiblere Mechanismen zur Anfrageoptimierung (siehe z. B. HÄRDER & RAHM (1999)) möglich, die entscheiden, in welcher Reihenfolge einzelne Teile einer Anfrage bearbeitet werden sollen. Oft kann so der Gesamtaufwand für eine Anfrage reduziert werden. Um diese Integration zu erreichen, muss die Datenhaltungskomponente des LS durch eine Erweiterung um Datenbankmechanismen, wie der Unterstützung von weiteren ein- und mehrdimensionalen Indizes, für die Verwaltung größerer Datenmengen ausgelegt werden. Um die gewünschten Verbesserungen zu erzielen, ist es weiterhin notwendig, existierende Mechanismen zur Anfragebearbeitung um eine Berücksichtigung der speziellen Eigenschaften von Positionsinformationen zu erweitern.

Schließlich wurden in der hier beschriebenen Realisierung des LS, wegen deren allgemeineren Verwendbarkeit, geometrische Koordinaten zur Beschreibung der Positionen mobiler Objekte verwendet. Da ein Benutzer mit den geometrischen Koordinaten selbst meist nicht viel anfangen kann, benötigt eine Anwendung, die auf den LS zugreift, für die Interpretation dieser Daten oft zusätzliche Karteninformationen, mit deren Hilfe sie die Koordinaten in Beziehung zu Objekten in der Umgebung des Benutzers setzen kann. Wenn ein Benutzer beispielsweise abfragen will, welche Personen sich gegenwärtig in einem durch eine bestimmte Raumnummer beschriebenen Raum aufhalten, werden Karteninformationen benötigt, um die Raumnummer auf die geometrischen Koordinaten des Raumes abzubilden. Weil für komplexere Anwendungen mit Ortsbezug diese Karteninformationen sowieso benötigt werden und selbst einfachere Anwendungen in ihrer Benutzungsoberfläche meist eine Kartendarstellung anbieten, ist dies jedoch keine wesentliche Einschränkung.

Für einfache Anwendungsszenarien, in denen keine Karteninformationen verfügbar sind, kann es allerdings sinnvoll sein, nur mit symbolischen Koordinaten zu arbeiten. Diese symbolischen Koordinaten sind für die Benutzer intuitiv verständlich und meist hierarchisch strukturiert (z. B. nach Stadt, Straße, Gebäude, Stockwerk und Raumnummer). Die hierarchische Strukturierung der Koordinaten erlaubt es, einfache Operationen, wie das Ermitteln einer Enthaltenseinsbeziehung, direkt auf den Koordinaten selbst durchzuführen. Wenn diese symbolischen Koordinaten beispielsweise über Infrarotbaken ausgesendet werden, können sie direkt von einem entsprechenden Sensor ausgelesen werden. Einfache Anwendungen mit Ortsbezug lassen sich damit

ohne größeren Aufwand erstellen und einführen. Für einen Lokationsdienst kann es daher wünschenswert sein, beide Formen von Koordinaten zu unterstützen. Neben Veränderungen an seinen Schnittstellen würde dies hauptsächliche eine Veränderung der Datenhaltung mit sich bringen, da deren räumlicher Index für hierarchische symbolische Koordinaten nicht benötigt wird. Die verteilten Algorithmen des LS können im Wesentlichen bestehen bleiben, da sie nur auf allgemeinen Vergleichen der Positionsinformationen, wie Enthaltensein oder Überlappung, beruhen.

Anhang A

Abkürzungsverzeichnis

ACL	Access Control List
API	Application Programming Interface
DB	Datenbank
DGPS	Differential Global Positioning System
DNS	Domain Name System
EOTD	Enhanced Observed Time Difference
GIS	Geographic Information System
GMLC	Gateway Mobile Location Center
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HLR	Home Location Register
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	Local Area Network
LBS	Location Based Services
LS	Location Service, Lokationsdienst

MAN	Metropolitan Area Network
MLP	Mobile Location Protocol
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
QoS	Quality of Service
RFC	Request for Comment
RPC	Remote Procedure Call
SLP	Service Location Protocol
SMS	Short Message Service
TCP	Transmission Control Protocol
TTL	Time to Live
TTP	Trusted Third Party
UDP	Universal Datagram Protocol
UMTS	Universal Mobile Telecommunications System
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
VLR	Visitor Location Register
WAN	Wide Area Network
WGS84	World Geodetic System 1984
WWW	World Wide Web
XML	Extensible Markup Language

Anhang B

Glossar

Agent
Als Agent eines mobilen Objekts wird derjenige Blatt-Lokations-Server bezeichnet, in dessen Dienstgebiet es sich aktuell aufhält und der demnach die Registrierungs- und Positionsinformationen für dieses speichert.

Aufenthaltsbereich (engl. *location area*)
Das durch einen Lokationsbezeichner definierte Gebiet, das die Ungenauigkeit der Positionsbestimmung beschreibt und in dem die Position des betreffenden mobilen Objekts enthalten ist.

Autorisierungszertifikat (engl. *authorization certificate*)
Eine Datenstruktur, welche die Berechtigung eines bestimmten Klienten beschreibt, auf die Positionsinformation eines mobilen Objekts mit einer vorgegebenen Genauigkeit zugreifen zu dürfen.

Blatt-Lokations-Server
Ein Lokationsserver, der keine Nachfolger in der LS-Hierarchie hat und für die Verwaltung der Registrierungs- und Positionsinformationen der mobilen Objekte in seinem Dienstgebiet zuständig ist.

Dienstgebiet (engl. *service area*)
Das geographische Gebiet für das der Lokationsdienst bzw. ein einzelner Lokations-Server zuständig ist und für das er die Positionsinformationen der darin enthaltenen mobilen Objekte verwaltet.

Gebietsanfrage (engl. *range query*)
Eine Anfrage an den Lokationsdienst, welche die Positionsinformationen für alle Objekte zurückgibt, die sich in einem angegebenen geographischen Gebiet aufhalten.

Nicht-Blatt-Lokations-Server
Die den Blatt-Servern übergeordnete Lokations-Server, die für das Auffinden des zu einem angefragten mobilen Objekts (bei Positionsanfragen) bzw. Gebiets (bei Gebiets- und Nachbarschaftsanfragen) gehörenden Blatt-Servers zuständig sind.

Kontakt-Server (engl. *entry server*)
Eine Anfrage kann an einen beliebigen Blatt-Lokations-Server gestellt werden, der dann der Kontakt-Server für diese Anfrage wird und für das Weiterleiten der Anfrage und das Zusammenstellen und Zurückliefern der Ergebnisse zuständig ist.

Koppelnavigationsprotokoll (engl. *dead-reckoning protocol*)
Ein Protokoll zur Positionsaktualisierung, bei dem Quelle und Senke die Position eines mobilen Objekts mit einer identischer Vorhersagefunktion berechnen und die Quelle nur dann eine Aktualisierungsnachricht versendet, wenn diese von dessen aktueller Position um mehr als einen bestimmten Betrag abweicht.

Lokationsbezeichner (engl. *location descriptor*)
Ein Lokationsbezeichner beschreibt den aktuellen Aufenthaltsort eines mobilen Objekts und die Ungenauigkeit dieser Information durch einen Aufenthaltsbereich.

Lokationsdienst (engl. *location service*)
Ein Dienst, der die dynamischen Positionsinformationen mobiler Objekte speichert und diesbezügliche Anfragen beantwortet.

Lokations-Server
Die Funktionalität des Lokationsdienstes wird von einem verteilten System von Lokations-Servern erbracht, die jeweils für ein bestimmtes Dienstgebiet zuständig sind und für die mobilen Objekte in diesem Dienstgebiet entweder direkt die Positionsinformationen (Blatt-Server) oder Suchverweise auf einen zuständigen Kind-Server speichern.

Mobiles Objekt (engl. *tracked object*)
In diesem Zusammenhang ein mobiles Objekt der realen Welt (z. B. eine Person oder ein Fahrzeug), dessen Position über einen Positionierungssensor ermittelt werden kann und dessen Positionsinformation vom Lokationsdienst verwaltet wird.

Nachbarschaftsanfrage (engl. *nearest neighbor query*)
Eine Anfrage an den Lokationsdienst, die das sich zu einer vorgegebenen geographischen Position am nächsten befindende mobile Objekt zurückgibt.

Ortsbezogene Anwendungen (engl. *location-aware applications*)
Eine Anwendung, die ihren Benutzern Informationsangebote und Dienste abhängig von deren aktuellem geographischem Aufenthaltsort anbietet.

Positionierungsdatensatz (engl. *sighting record*)
Ein von einem Positionierungssensor gelieferter Datensatz, der für einen gegebenen Zeitpunkt den Aufenthaltsort eines bestimmten mobilen Objekts und die Genauigkeit dieser Information durch ein Aufenthaltsgebiet beschreibt.

Positionierungssensor (engl. *positioning sensor*)
Ein Sensor, der die aktuelle Position eines mobilen Objekts mit einer bestimmten Genauigkeit bestimmen kann.

Positionsanfrage (engl. *position query*)
Eine Anfrage an den Lokationsdienst, welche die aktuelle Position eines angegebenen mobilen Objekts zurückgibt.

Registrierende Instanz
Diese meldet ein mobiles Objekt beim Lokationsdienst an und kontrolliert die Genauigkeit, mit der dieser dessen Positionsinformation speichert.

Suchverweis (engl. *forwarding pointer*)
Verweist auf denjenigen Nachfolger eines Nicht-Blatt-Lokations-Servers, auf dem entweder die Positionsinformation selbst oder der nächste Suchverweis zu einem bestimmten mobilen Objekt gespeichert sind.

Wurzel-Lokations-Server (engl. *root location server*)

Der Lokations-Server an der Spitze der Server-Hierarchie, der für das Gesamtdienstgebiet des LS zuständig ist.

Anhang C

Mathematische Bezeichner und Funktionen

Die in dieser Arbeit verwendeten mathematischen Bezeichner und Funktionen sind mit ihren Einheiten bzw. Ergebnismengen im Folgenden zusammengefasst.

Mengen

Pos	Menge aller möglichen geographischen Koordinaten.
A	Menge aller zusammenhängenden geographischen Gebiete ($A \subseteq Pos^*$).
O	Alle vom Lokationsdienst verwalteten mobilen Objekte.

Funktionen

DISTANCE(x,y)	Berechnet die Entfernung zwischen den beiden geographischen Koordinaten *x* und *y* in Metern *(*m*).*
SIZE(a)	Berechnet den Flächeninhalt des geographischen Gebiets *a* in Quadratmetern (m^2).
CIRCLE(p,r)	Erzeugt die Beschreibung für einen Kreis um Mittelpunkt *p* mit Radius *r* (*A*).
ENLARGE(a, d)	Vergrößert das Gebiet *a*, indem dessen Grenzen an jedem Punkt um den Betrag *d* senkrecht nach außen verschoben werden (*A*).

Mobile Objekte und Sensorinformationen

rp	Tatsächliche Position des mobilen Objekts ($rp \in Pos$).

$\hat{v}$	Maximale Geschwindigkeit des mobilen Objekts (m/s).
$\bar{v}$	Durchschnittliche Geschwindigkeit des mobilen Objekts (m/s).
dens	Durchschnittliche Dichte an mobilen Objekten je Quadratmeter ($1/m^2$).
p_p	Die von einem Positionierungssensor für ein mobiles Objekt bestimmte Position ($p_p \in Pos$).
t_p	Der Zeitpunkt an dem die Positionsinformation p_p durch den Positionierungssensor bestimmt wurde.
u_p	Die Genauigkeit der von dem Positionierungssensor bestimmten Positionsinformation p_p (m).

Anfragen zu Positionsinformationen

q	Durchschnittliche Anzahl von Anfragen pro Zeiteinheit (pro s).
u_q	Die bei Anfragen durchschnittlich für die Positionsinformationen geforderte Genauigkeit (m).
u'_q	Die bei einer bestimmten Anfrage geforderte Genauigkeit (m).

Parameter der Positionsaktualisierungsprotokolle

D	Der Entfernungsschwellwert bei einem entfernungsbasierten oder kombinierten Positionsaktualisierungsprotokoll (m).
T	Der Zeitschwellwert bei einem zeitbasierten Positionsaktualisierungsprotokoll (s).
$\hat{T}$	Maximales Zeitintervall, das zwischen zwei Positionsaktualisierungen verstreichen darf (s).
u_s	Die Genauigkeit mit der die Positionsinformation bei einem berichtenden oder kombinierten Protokoll auf der Senke vorgehalten wird (m).
v_{asd}	Die für das mobile Objekt angenommene (maximale) Geschwindigkeit bei einem Protokoll mit Zwischenspeichern (m/s).
u_m	Die beim Abgleich mit einer Karte maximal akzeptable Entfernung zwischen der Position eines mobilen Objekts und einer Verbindung der Straßenkarte (m).
pred()	Vorhersagefunktion, die bei einem Koppelnavigationsprotokoll von Quelle und Senke verwendet wird.

Eigenschaften der Positionsaktualisierungsprotokolle

m	Die Anzahl der für die Übertragung der Positionsinformationen pro Zeiteinheit durchschnittlich benötigten Nachrichten (pro s).
$\hat{d}$	Die maximale Abweichung zwischen der zurückgegebenen und der tatsächlichen Position für ein mobiles Objekt (m).
$\bar{d}$	Die durchschnittliche Abweichung zwischen der zurückgegebenen und der tatsächlichen Position (m).
p_r	Die bei der letzten Positionsaktualisierung übertragene Position ($p_r \in Pos$).
p_c	Die beim kartenbasierten Koppelnavigationsprotokoll durch Abgleich mit einer Karte korrigierte Position des mobilen Objekts ($p_c \in Pos$).
$gain_{dr}$	Verhältnis zwischen der bei Verwendung einer Vorhersagefunktion eines Koppelnavigationsprotokolls pro Zeiteinheit benötigten Aktualisierungsnachrichten und der entsprechenden Anzahl ohne Vorhersagefunktion.

Anhang D

SQL-Anweisungen für die Vergleichsmessungen mit DB2

In Abschnitt 7.2 wurden die Ergebnisse des Vergleichstests zwischen der vorgeschlagenen hauptspeicherbasierten Datenhaltung für Positionsinformationen und einer Realisierung auf Basis einer kommerziellen DB2 Datenbank von IBM mit der räumlichen Erweiterung „Spatial Extender” gezeigt. Für letztere wurden dabei die im Folgenden vereinfacht dargestellten SQL-Anweisungen verwendet.

Erstellung der Tabelle und des mehrdimensionalen Indexes:

```
CREATE TABLE LOCINFO (oid VARCHAR(20), p db2gse.st_point, acc DOUBLE);

CREATE UNIQUE INDEX OINDEX ON LOCINFO(oid);

CREATE INDEX LINDEX ON LOCINFO(p) EXTEND USING
    db2gse.spatial_index(50, 250, 1000);
```

Beispiel für Einfügen eines mobilen Objekts:

```
INSERT INTO LOCINFO VALUES ('test00001', db2gse.st_point(1000.0, 1100.0,
    db2gse.coordref()..srid(0)), -1.0);
```

Beispiel für Positionsaktualisierung:

```
UPDATE LOCINFO SET p = db2gse.st_point(1100.0, 1200.0, db2gse.coordref()..srid(0))
    WHERE oid = 'test00001';
```

Beispiel für Positionsanfrage:

```
SELECT p FROM LOCINFO WHERE oid = 'test00001';
```

Beispiel für Gebietsanfragen mit unterschiedlich großen Gebieten:

```
SELECT oid FROM LOCINFO WHERE db2gse.st_contains(db2gse.st_polyfromtext(
    'polygon((10.0 10.0, 20.0 10.0, 20.0 0.0, 10.0 0.0, 10.0 10.0))', db2gse.coor-
    dref()..srid(0)), p) = 1;

SELECT oid FROM LOCINFO WHERE db2gse.st_contains(db2gse.st_polyfromtext(
    'polygon((100.0 100.0, 200.0 100.0, 200.0 0.0, 100.0 0.0, 100.0 100.0))',
    db2gse.coordref()..srid(0)) , p) = 1;

SELECT oid FROM LOCINFO WHERE db2gse.st_contains(db2gse.st_polyfromtext(
    'polygon((100.0 1000.0, 1100.0 1000.0, 1100.0 0.0, 100.0 0.0, 100.0 1000.0))',
    db2gse.coordref()..srid(0)) , p) = 1;
```

Beispiel für Nachbarschaftsanfrage (da bei einer Realisierung nur über die Distanzberechnungsfunktion *st_distance* der mehrdimensionale Index nicht berücksichtigt wird, wurde hier eine deutlich schnellere Variante unter Verwendung einer Gebietsanfrage mit *st_contains* eingesetzt, wobei eine geforderte maximale Entfernung *maxDist* von 50 m angenommen wurde):

```
SELECT oid, db2gse.st_distance(p, db2gse.st_point(500.0, 100.0,
    db2gse.coordref()..srid(0))) AS dist FROM LOCINFO WHERE db2gse.st_contains(
    db2gse.st_polyfromtext('polygon((450.0 50.0, 550.0 50.0, 550.0 150.0, 450.0 150.0,
    450.0 50.0))', db2gse.coordref()..srId(0)),p) = 1 ORDER BY dist ASC;
```

Anhang E

Entfernung einer Position zu einem kreisförmigen Aufenthaltsbereich

Für die Semantik der in Abbildung 4.3 beschriebenen Nachbarschaftsanfrage wird in Kapitel 4 die durchschnittliche Entfernung zwischen allen Positionen in einem kreisförmigen Aufenthaltsbereich *U* und einem vorgegebenen Punkt *p*, der außerhalb von *U* liegt, benötigt. Diese ist allerdings nur bei einer großen Entfernung zwischen Kreis und Ausgangspunkt gleich der Entfernung zu dessen Mittelpunkt, wie dort angenommen wird. Bei einer gleichförmigen Verteilung der Aufenthaltswahrscheinlichkeit innerhalb des Kreises ergibt sich die durchschnittliche Entfernung aus dem Flächenintegral über die Entfernung von *p* zu allen Positionen innerhalb des Gebiets geteilt durch den Flächeninhalt des Aufenthaltsgebiets:

$$\overline{DISTANCE(U,\ p)} = \frac{\int\int_{U} DISTANCE(z,\ p)\ dA}{|U|} \qquad z \in U$$

Zur Vereinfachung der Berechnung des in dieser Formel enthaltenen Flächenintegrals soll im Folgenden o. B. d. A. angenommen werden, dass *p* auf dem Ursprung des verwendeten Koordinatensystems liegt und der Mittelpunkt des kreisförmigen Aufenthaltsbereichs auf dessen x-Achse. Wenn *d* der Entfernung zwischen *p* und dem Mittelpunkt von *U* entspricht und *r* dem Radius von *U,* haben alle Positionen innerhalb des Aufenthaltsbereichs eine Entfernung zu *p* von *d* - *r* bis *d* + *r*. Alle Positionen mit derselben Entfernung *e* zu *p* liegen dann auf einem Kreisbogen, der sich ergibt, wenn ein Kreis um *p* mit dem entsprechenden Radius *e* mit dem Aufenthaltsbereich *U* geschnitten wird (siehe Abbildung 9-1). Mit α als dem Winkel zwischen einer

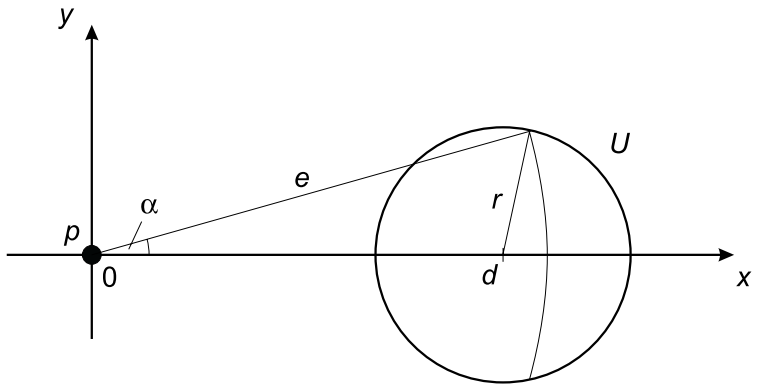


Abbildung 9-1. Berechnung der durchschnittlichen Entfernung zwischen einem Punkt p und den Positionen eines kreisförmigen Aufenthaltsbereichs A.

Geraden durch diesen Schnittpunkt und den Ursprung sowie der x-Achse und somit einer Länge des Kreisbogens von $2e\alpha$ ergibt sich das obige Flächenintegral zu:

$$\int\int_{U} DISTANCE(z,\ p)\ dA = \int_{d-r}^{d+r} e \cdot\ 2e\alpha de$$

Der Winkel α läßt sich dabei durch den Kosinussatz im Dreieck bestimmen. Damit ist die Formel für die gesuchte durchschnittliche Entfernung:

$$\overline{DISTANCE(U,\ p)} = \frac{\int_{d-r}^{d+r} 2e^2 \operatorname{acos}\left(\frac{e^2 + d^2 - r^2}{2ed}\right) de}{2\pi r^2}$$

Leider gibt es für dieses Integral keine geschlossene Lösung, es kann nur numerisch gelöst werden. Abbildung 9-2 zeigt das mit dem Mathematikprogramm Maple berechnete Verhältnis zwischen der gesuchten durchschnittlichen Entfernung und der Abschätzung durch die einfachen Entfernung zwischen *p* und dem Mittelpunkt von *A* abhängig von dem Faktor, um den *p* weiter vom Mittelpunkt von *A* entfernt ist als dessen Radius *r (d/r)*. Wie in Abbildung 9-2 zu sehen, wird der Unterschied schnell vernachlässigbar, wenn *d* groß im Vergleich zu *r* ist. Selbst bei einer ähnlichen Größenordnung wird der Unterschied nie größer als 15%.

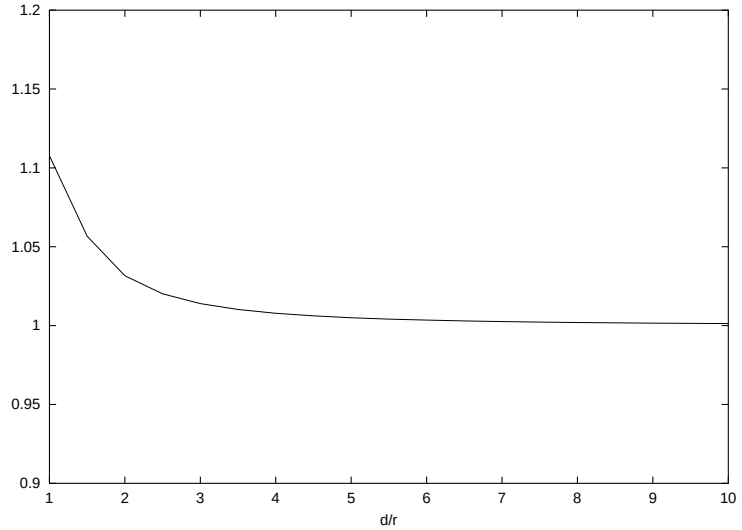


Abbildung 9-2. Verhältnis zwischen der durchschnittlichen Entfernung eines Punkts p *zu den Punkten eine kreisförmigen Aufenthaltsbereiches* A *und der Abschätzung durch* p*'s Entfernung zu* A*'s Mittelpunkt* d*, abhängig vom Verhältnis zwischen* A*'s Radius und* d*.*

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

ABUTALEB & LI (1997)

Abutaleb, A. und Li, V. O. K. (1997), „Location Update Optimization in Personal Communication Systems", in ACM/Baltzer Wireless Networks Journal (WINET) 3, 3, Seiten 205-216.

ADDLESEE *ET AL.* (2001)

Addlesee, M. und Curwen, R. und Hodges, S. und Newman, J. und Steggles, P. und Ward, A. und Hopper, A. (2001), „Implementing a Sentient Computing System", in IEEE Computer 34, 8, Special Issue on Location Aware Computing, Seiten 50-56.

ADIJIE-WINOTO *ET AL.* (1999)

Adijie-Winoto, W. und Schwartz, E. und Balakrishnan, H. und Lilley, J. (1999), „The Design and Implementation of an Intentional Naming System", in Proc. of the 17th ACM Symp. on Operating Systems Principles (SOSP '99), Kiawah Island Resort, SC, USA, Seiten 186-201.

ALONSO, BARBARA & GARCIA-MOLINA (1990)

Alonso, R. und Barbara, D. und Garcia-Molina, H. (1990), „Data Caching Issues in an Information Retrieval System" in ACM Transactions on Database Systems 15, 3, Seiten 359-384.

AURA (1999)

Aura, T. (1999), „Distributed Access-rights Management with Delegation Certificates", in Secure Internet Programming: Security Issues for Distributed and Mobile Objekts, LNCS 1603, Springer, Seiten 211-235.

BAGGIO, BALLINITIJN & VAN STEEN (2000)

Baggio, A. und Ballintijn, G. und van Steen, M. (2000), „Mechanisms for Effective Caching in the Globe Location Service", in Proc. of the 9th ACM SIGOPS European Workshop, Kolding, Dänemark, Seiten 55-60.

BAHL & PADMANABHAN (2000)

Bahl P. und Padmanabhan, V. N. (2000), „RADAR: An In-Building RF-Based User Location and Tracking System", in Proc. of the 19th IEEE Conf. on Computer Communications (InfoCom 2000), Tel Aviv, Israel, Seiten 775-784.

BAJAJ *ET AL.* (1999)

Bajaj, S. und Breslau, L. und Estrin, D. und Fall, K. und Floyd, S. und Handley, M. und Haldar, P. und Helmy, A. und Heidemann, J. und Huand, P. und Kumar, S. und McCanne, S. und Rejaie, R. und Sharma, P. und Varadhan, K. und Xu, Y. und Yu, H. und Zappala, D. (1999), „Improving Simulation for Network Research", Technischer Bericht Nr. 99-702, University of Southern California, CA, USA.

BAR-NOY, KESSLER & SIDI (1995)

Bar-Noy, A. und Kessler, I. und Sidi, M. (1995), „Mobile Users: To Update or not to Update?", in Wireless Networks 1, 2, Seiten 175-185.

BAUER (2000)

Bauer, M. (2000), „Event-Management für mobile Benutzer", Diplomarbeit Nr. 1836, Fakultät Informatik, Universität Stuttgart, Deutschland.

BAUMANN *ET AL.* (2001)

Baumann, J. und Coschurba, P. und Kubach, U. und Leonhardi, A. (2001), „Metaphors for Context-Aware Information Access", in Springer Personal Technologies Journal 5, 1, Seiten 16-19.

BHATTACHARYA & DAS (1999)

Bhattacharya, A. und Das, S. (1999), „LeZi-update: An Information-theoretic Approach to Track Mobile Users in PCS Networks", in Proc. of the 5th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '99), Seattle, WA, USA, Seiten 1-12.

BROCH *ET AL.* (1998)

Broch, J. und Maltz, D. und Johnson, D. und Hu, Y.-C. und Jetcheva, J. (1998), „A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", in Proc. of the 4th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '98), Dallas, TX, USA, Seiten 85-97.

BUTZ *ET AL.* (2001)

Butz, A. und Baus, J. und Krüger, A. und Lohse, M. (2001), „A Hybrid Indoor Navigation System", in Proc. of the 2001 International Conference on Intelligent User Interfaces (IUI 2001), Santa Fe, NM, USA, Seiten 25-32.

CAMBRIDGE POSITIONING SYSTEMS (2001)

Cambridge Positioning Systems (2001), „CPS Home Page", Web-Seite, URL: http://www.cursor-system.com.

CHEVERST *ET AL.* (2000)

Cheverst, K. und Davies, N. und Mitchell, K. und Friday, A. (2000), „Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project", in Proc. of the 6th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom 2000), Boston, MA, USA, Seiten 20-31.

CLARK (1988)

Clark, D. (1988), „The Design Philosophy of the DARPA Internet Protocols", in Proc. of the ACM SIGCOMM Symp. on Communications Architectures and Protocols (SIGCOMM '88), Stanford, CA, USA, Seiten 106-114.

CLAUSS & KÖHNTOPP (2001)

Clauß, S. und Köhntopp, M. (2001), „Identity Management and its Support of Multilateral Security", in Computer Networks 37, 1, Special Issue on Electronic Business Systems, Elsevier Science Publishers, Seiten 205-219.

COMER (2000)

Comer, D. E. (2000), „Internetworking with TCP/IP - Principles, Protocols, and Architectures", ISBN 0-13-018380-6, Prentice Hall, Upper Saddle River, NJ, USA, Vierte Auflage.

COULOURIS, DOLLIMORE & KINDBERG (2001)

Coulouris, G. und Dollimore, J. und Kindberg, T. (2001), „Distributed Systems – Concept and Design", ISBN 0201-61918-0, Addison-Wesley, Harlow, UK, Dritte Auflage.

COSCHURBA, KUBACH & LEONHARDI (2000)

Coschurba, P. und Kubach, U. und Leonhardi, A. (2001), „Research Issues in Developing a Platform for Spatial-Aware Applications", in Proc. of the SIGOPS European Workshop, Kolding, Dänemark, Seiten 153-158.

COSCHURBA, ROTHERMEL & DÜRR (2002)

Coschurba, P. und Rothermel, K. und Dürr, F. (2002), „A Fine Grained Addressing Concept for GeoCast", in Proc. of the Int. Conf. on Architecture of Computing Systems, Trends in Network and Pervasive Computing (ARCS '02), Karlsruhe, Deutschland, Seiten 101-113.

CZERWINSKI *ET AL.* (1999)

Czerwinski, S. und Zhao, B. Y. und Hodes, T. und Joseph, A. und Katz, R. (1999), „An Architecture for a Secure Service Discovery Service", in Proc. of the 5th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '99), Seattle, WA, USA, Seiten 24-35.

DAIMLERCHRYSLER AG (2002)

DaimlerChrysler AG (2002), „Mercedes-Benz Fleetboard", Web-Seite, URL: http://www.fleetboard.de.

DAVIS *ET AL.* (1996)

Davis, C. und Vixie, P. und Goodwin, T. und Dickinson, I. (1996), „A Means for Expressing Location Information in the Domain Name System", Internet Request for Comment, RFC 1876, Internet Engineering Task Force, URL: http://www.ietf.org/rfc/rfc1876.txt.

DAVIS (1998)

Davis, J. (1998), „The IBM DB2 Spatial Extender: Managing Geo-Spatial Information within the DBMS", Technischer Bericht, IBM Cooperation, URL: http://www.ibm.com/software/data/pubs/papers/spatial/.

DUDKOWSKI (2002)

Dudkowski, D. (2002), „Ereignisse im Nexus Lokationsdienst", Studienarbeit Nr. 1825, Fakultät Informatik, Universität Stuttgart, Deutschland.

ELLISON (1999)

Ellison, C. (1999), „The Nature of a Usable PKI", in Computer Networks 31, 8, Special Issue on Computer Network Security, Elsevier Science Publishers, Seiten 823-830 (auch in RFC 2693).

ESRI (2001)

ESRI Cooperation (2001), „ArcGIS Main Page", Web-Seite, URL: http://www.esri.com/software/arcgis/index.html.

FEINER *ET AL.* (1997)

Feiner, S. und MacIntyre, B. und Höllerer, T. und Webster, T. (1997), „A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment", in Proc. of the 1st Int. Symp. on Wearable Computers (ISWC '97), Cambridge, MA, USA, Seiten 74-81.

FITZMAURICE (1993)

Fitzmaurice, G. (1993), „Situated Information Spaces and Spatially Aware Palmtop Computers", in Communications of the ACM 36, 7, Seiten 38-49.

FOLEY *ET AL.* (1995)

Foley, J. und van Dam, A. und Feiner, S. und Hughes, J. (1995), „Computer Graphics", ISBN 0-2018-4840-6, Addison-Wesley, Reading, MA, USA, Zweite Auflage.

FORLIZZI *ET AL.* (2000)

Forlizzi, L. und Güting, R. H. und Nardelli, E. und Schneider, M. (2000), „A Data Model and Data Structures for Moving Objects Databases", in Proc. of the 19th ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD 2000), Dallas, TX, USA, Seiten 319-330.

FREI (2000)

Frei, T. (2000), „Entwurf und Implementierung einer Komponente für die Übermittlung von Positionsinformationen innerhalb des Nexus Lokalisierungsdienstes", Studienarbeit Nr. 1795, Fakultät Informatik, Universität Stuttgart, Deutschland.

FRIEDMANN, BENTLEY & FINKEL (1977)

Friedmann, J. H. und Bentley, J. J. und Finkel, R. A. (1977), „An Algorithm for Finding Best Matches in Logarithmic Expected Time", in ACM Transactions on Mathematical Software 3, 3, Seiten 209-226.

FRITZ (1999)

Fritz, A. (1999), „Positionsabhängiger Zugriff auf WWW-Inhalte", Diplomarbeit Nr. 1709, Fakultät Informatik, Universität Stuttgart, Deutschland.

GAEDE & GÜNTHER (1998)

Gaede, V. und Günther, O. (1998), „Multidimensional Access Methods" in ACM Computing Surveys 30, 2, Seiten 170-231.

GROSSMANN *ET AL*. (2001)

Grossmann, M. und Leonhardi, A. und Mitschang, B. und Rothermel, K. (2001), „A World Model for Location-Aware Systems", in Informatik - Zeitschrift der schweizerischen Informatikorganisationen 8, 5, Seiten 22-25.

GUTTMAN *ET AL*. (1999)

Guttman, E. und Perkins, C. und Veizades, J. und Day, M. (1999), „Service Location Protocol, Version 2", Internet Request for Comment, RFC 2608, Internet Engineering Task Force, URL: http://www.ietf.org/rfc/rfc2608.txt.

GUTTMANN (1984)

Guttmann, A. (1984), „R-Trees: A Dynamic Index Structure for Spatial Searching", in Proc. of the 13th ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD '84), Boston, MA, USA, Seiten 47-57.

HÄRDER & RAHM (1999)

Härder, T. und Rahm, E. (1999), „Datenbanksysteme, Konzepte und Techniken der Implementierung", ISBN: 3-540-65040-7, Springer-Verlag, Berlin, Deutschland.

HARTER & HOPPER (1994)

Harter, A. und Hopper, A. (1994), „A Distributed Location System for the Active Office" in IEEE Network 36, 1, Seiten 62-70.

HARTER *ET AL*. (1999)

Harter, A. und Hopper, A. und Steggles, P. und Ward, A. und Webster, P. (1999), „The Anatomy of a Context-Aware Application", in Proc. of the 5th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MOBICOM'99), Seattle, WA, USA, Seiten 59-68.

HÄSSLER (2001)

Hässler, A. (2001), „Untersuchung von Datenbanktechnologien für hochdynamische Positionsinformationen mobiler Objekte", Diplomarbeit Nr. 1889, Fakultät Informatik, Universität Stuttgart, Deutschland.

HAUSER & KABATNIK (2001)

Hauser, C. und Kabatnik, M. (2001): „Towards Privacy Support in a Global Location Service", in Proceedings of the IFIP Workshop on IP and ATM Traffic Management (WATM/EUNICE 2001), Paris, Frankreich, Seiten 81-89.

HAUSER, LEONHARDI & KÜHN (2002)

Hauser, C. und Leonhardi, A. und Kühn, P. J. (2002), „Sicherheitsaspekte in Nexus - einer Plattform für ortsbezogene Anwendungen", in it + ti - Informationstechnik und Technische Informatik, 5/2002, Oldenbourg Verlag, München, Deutschland, Seiten 268-277.

HATAMIAN (1995)

Hatamian, S. T. (1995): „Application Note: A Fast Time-Correlated Gaussian Random Number Generator", Web-Seite, URL: http://www.fortran.com/fortran/fm_gauss.html.

HERRSCHER, LEONHARDI & ROTHERMEL (2002)

Herrscher, D. und Leonhardi, A. und Rothermel, K. (2002): „Modeling Computer Networks for Emulation", in Proc. of the Int. Conf. on Parallel and Distributed Processing Techniques and Applications, PDPTA '02, Las Vegas, NV, USA, Seiten 1725-1731.

HIGHTOWER & BORRIELLO (2001)

Hightower, J. und Borriello, G. (2001), „Location Systems for Ubiquitous Computing", in IEEE Computer 34, 8, Special Issue on Location Aware Computing, Seiten 57-66.

HO & AKYILDIZ (1997)

Ho, J. S. M. und Akyildiz, I. F. (1997), „Dynamic Hierarchical Database Architecture for Location Management in PCS Networks", in IEEE/ACM Transactions on Networking 5, 5, Seiten 646-661.

HOHL *ET AL.* (1999)

Hohl, F. und Kubach, U. und Leonhardi, A. und Rothermel, K. und Schwehm, M. (1999), „Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications", in Proc. of the 5th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '99), Seattle, WA, USA, Seiten 249-255.

HOFMANN-WELLENDORF, LICHTENEGGER & COLLINS (1997)

Hofmann-Wellendorf, B. und Lichtenegger, H. und Collins, J. (1997), „Global Positioning System: Theory and Practice", ISBN: 3211-83534-2, Springer-Verlag, Wien, Österreich, Fünfte Auflage.

HOLMQUIST *ET AL.* (2001)

Holmquist, L. E. und Mattern, F. und Schiele, B. und Alahuhta, P. und Beigl, M. und Gellersen, H.-W. (2001), „Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts", in Proc. of the Int. Conf. on Ubiquitous Computing (Ubicomp 2001), Atlanta, GA, USA, Seiten 116-122.

IBM (2001)

IBM Cooperation (2001), „DB2 Universal Database", Web-Seite, URL: http://www.ibm.com/software/data/db2/udb/.

IEEE COMPUTER SOCIETY (1997)

IEEE Computer Society LAN MAN Standards Committee (1997), „Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification", The Institute of Electrical and Electronics Engineers, New York, IEEE Standard 802.11.

ITU/ISO (1997)

ITU/ISO (1997), „Recommendation X.500 (08/97): Open Systems Interconnection - The Directory: Overview of Concepts, Models and Services", Int. Telecommunications Union.

JALOTE (1994)

Jalote, P. (1994), „Fault Tolerance in Distributed Systems", ISBN: 0-13-301367-7, Prentice Hall, Englewood Cliffs, NJ, USA.

JENDRICKE & GERD TOM MARKOTTEN (2000)

Jendricke, U. und Gerd tom Markotten, D. (2000), „Usability meets Security - The Identity Manager as Your Personal Security Assistant for the Internet", in Proc. of the 16th Annual Computer Security Conf. (ACSAC 2000), New Orleans, LA, USA, Seiten 11-15.

JINI (2001)

Sun Microsystems Inc. (2001), „Jini Network Technology", Web-Seite, URL: http://www.sun.com/jini.

KLEEMANN (1992)

Kleemann, L. (1992), „Optimal Estimation of Position and Heading for Mobile Robots Using Ultrasonic Beacons and Dead-Reckoning", in Proc. of the 1992 IEEE Int. Conf. on Robotics and Automation, Nice, France, Seiten 2582-2587.

KO & VAIDYA (1998)

Ko, Y.-B. und Vaidya, N. H. (1998), „Location-Aided Routing (LAR) in Mobile Ad Hoc Networks", in Proc. of the 4th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '98), Dallas, TX, USA, Seiten 66-75.

KOVACS (1999)

Kovacs, E. (1999), „Vermittlung und Management von Diensten in offenen Systemen", Dissertation, Fakultät Informatik, Universität Stuttgart, Deutschland, ISBN 3-8300-0068-5, Verlag Dr. Kovac.

KUBACH *ET AL.* (1999)

Kubach, U. und Leonhardi, A. und Rothermel, K. und Schwehm, M. (1999), „Analysis of Distribution Schemes for the Management of Location Information", Technischer Bericht Nr. 1999/01, Fakultät Informatik, Universität Stuttgart, Deutschland.

LAM *ET AL.* (1998)

Lam, D. und Cui, Y. und Cox, D. C. und Widom, J. (1998), „A Location Management Technique to Support Lifelong Numbering in Personal Communications Services", in ACM Mobile Computing and Communications Review 2, 1, Seiten 27-35.

LAMMING & FLYNN (1994)

Lamming, M. und Flynn, M. (1994), „Forget-me-not: Intimate Computing in Support of Human Memory", in Proc. of FRIEND 21: Int. Symp. on Next Generation Human Interfaces, Tokyo, Japan, Seiten 125-128.

LEONHARDI *ET AL.* (1999)

Leonhardi, A. und Kubach, U. und Rothermel, K. und Fritz, A. (1999), „Virtual Information Towers - A Metaphor for Intuitive, Location-Aware Information Access in a Mobile Environment", in Proc. of the 3rd IEEE Int. Symp. on Wearable Computers (ISWC '99), San Fransisco, CA, USA, Seiten 15-20.

LEONHARDI & KUBACH (1999)

Leonhardi, A. und Kubach, U. (1999), „An Architecture for a Universal, Distributed Location Service", in Proc. of the European Wireless '99 Conf., München, Deutschland, ITG Fachbericht, VDE Verlag, Berlin, Deutschland, Seiten 351-355.

LEONHARDI & ROTHERMEL (2001)

Leonhardi, A. und Rothermel, K. (2001), „A Comparison of Protocols for Updating Location Information", in Baltzer Cluster Computing Journal 4, 4, Seiten 355-367.

LEONHARDI & ROTHERMEL (2002)

Leonhardi, A. und Rothermel, K. (2002), „Architecture of a Large-scale Location Service", Short Paper in Proc. of the 22nd Int. Conf. on Distributed Computing Systems (ICDCS 2002), Wien, Österreich, Seiten 465-466 (auch als Technischer Bericht Nr. 2001/01, Fakultät Informatik, Universität Stuttgart, Deutschland).

LEONHARDI, NICU & ROTHERMEL (2002)

Leonhardi, A. und Nicu, C. und Rothermel, K. (2002), „A Map-based Dead-reckoning Protocol for Updating Location Information", in Proc. of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (IPDPS-WPIM 2002), Ft. Lauderdale, FL, USA.

LEONHARDT (1998)

Leonhardt, U. (1998), „Supporting Location-Awareness in Open Distributed Systems", Dissertation, Imperial College of Science, Technology and Medicine, University of London, UK.

LI *ET AL.* (2000)

Li, J. und Jannotti, J. und De Couto, D. und Karger, D. und Morris, R. (2000), „A Scalable Location Service for Geographic Ad-hoc Routing", in Proc. of the 6th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom 2000), Boston, MA, USA, Seiten 120-130.

LIF (2001)

Location Inter-operability Forum (2001), „Mobile Location Protocol Specification", Approved Specification, LIF TS 101 v2.0.0, URL: http://www.locationforum.org/publicdownload/LIF-TS-101-v2.0.0.zip.

LOCUST (2001)

MIT Media Lab (2001), „The Locust Home Page", Web-Seite, URL: http://lcs.www.media.mit.edu/projects/wearables/locust/.

LUTGE (2001)

Lutge, G.(2001), „Jetzt, immer und überall", Die Zeit, Nr. 38.

MAASS (1997)

Maaß, H. (1997), „Location-Aware Mobile Applications based on Directory Services", in Proc. of the 3rd Int. Conf. on Mobile Computing and Networking (MobiCom '97), Budapest, Ungarn, Seiten 22-33.

MATHAR & PFEIFER (1990)

Mathar, R. und Pfeifer, D. (1990), „Stochastik für Informatiker", ISBN: 3-519-02240-0, B. G. Teubner Verlag, Stuttgart, Deutschland.

MOCKAPETRIS & DUNLAP (1988)

Mockapetris, P. und Dunlap, K. (1988), „Development of the Domain Name System", in Proc. of the ACM SIGCOMM Symp. on Communications Architectures and Protocols (SIGCOMM '88) , Stanford, CA, USA, Seiten 123-133.

MOULY & PAUTET (1992)

Mouly, M. und Pautet, M.-B. (1992), „The GSM System for Mobile Communications", Cell & Sys, Frankreich.

MÜLLER & RANNENBERG (1999)

Müller, G. und Rannenberg, G. (Eds.) (1999), „Multilateral Security in Communications", ISBN: 3-8273-1360-0, Addison-Wesley, Reading, MA, USA.

MURATORE (2001)

Muratore, F. (2001), „UMTS: Mobile Communications for the Future", ISBN: 0-471-49829-7, Wiley, Chichester, England.

NAGUIB & COULOURIS (2001)

Naguib, H. und Coulouris, G. (2001), „Location Information Managment", in Proc. of the Int. Conf. on Ubiquitous Computing (UbiComp 2001), Atlanta, GA, USA, Seiten 35-41.

NAVAS & IMIELINSKI (1997)

Navas, J. und Imielinski, T. (1997), „Geocast - Geographic Addressing and Routing", in Proc. of the 3rd Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '97), Budapest, Ungarn, Seiten 66-76.

NEL (2001)

Network Emulation Lab (2001), „The Network Emulation Testbed Homepage", Web-Seite, Fakultät Informatik, Universität Stuttgart, Deutschland, URL: http://www.informatik.uni-stuttgart.de/ipvr/vs/en/projects/net/index.html.

NEXUS (2002)

Nexus Forschergruppe (2002), „Research Group Nexus", Web-Seite, Universität Stuttgart, Deutschland, URL: http://www.nexus.uni-stuttgart.de.

NICKLAS *ET AL.* (2001)

Nicklas, D. und Grossmann, M. und Schwarz, T. und Volz, S. und Mitschang, B. (2001), „A Model-Based Open Architecture for Mobile, Spatially-Aware Applications", in Proc. of the 7th Int. Symp. on Spatial and Temporal Databases (SSTD 2001), Redondo Beach, CA, USA, Seiten 117-135.

NICKLAS & MITSCHANG (2001)

Nicklas, D. und Mitschang, B. (2001), „The Nexus Augmented World Model: An Extensible Approach for Mobile Spatially-Aware Applications", in Proc. of the 7th Int. Conf. on Object-Oriented Information Systems (OOIS 2001), Calgary, Canada, Seiten 392-401.

NICU (2001)

Nicu, C. (2001), „Untersuchung von Koppelnavigations-Protokollen für die Übertragung von Positionsinformationen", Diplomarbeit Nr. 1919, Fakultät Informatik, Universität Stuttgart, Deutschland.

NIMA (1997)

National Imagery and Mapping Agency (1997), „DoD World Geodetic System 1984, its Definition and Relationship with Local Geodetic Systems", National Imagery and Mapping Agency, 8350.2, Dritte Auflage.

NOKIA (2001)

Nokia Networks (2001), „Nokia Artuse Gateway Mobile Location Center", Web-Seite, URL: http://www.nokia.com/pc_files/GMLC_datasheet.pdf.

OPEN GIS (2001)

Open GIS Consortium (2001), „Open GIS Consortium Home Page", Web-Seite, URL: http://www.opengis.org.

ORACLE (1997)

Oracle Cooperation (1997), „Oracle8 Spatial Cartridge, Advances in Relational Database Technology for Spatial Data Management", Technischer Bericht, URL: http://otn.oracle.com/products/oracle8/htdocs/xsdotwp3.htm.

PERKINS (1996)

Perkins, C. (Editor) (1996), „IP Mobility Support", Internet Request for Comment, RFC 2002, Internet Engineering Task Force, URL: http://www.ietf.org/rfc/rfc2002.txt.

PERKINS, ROYER & DAS (1999)

Perkins, C. und Royer, E. und Das, S. R. (1999), „Ad hoc On demand Distance Vector (AODV) Routing", Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-04.txt.

PITOURA & SAMARAS (2001)

Pitoura, E. und Samaras, G. (2001), „Locating Objects in Mobile Computing", in IEEE Transactions on Knowledge and Data Engineering 13, 4, Seiten 571-592.

PRIYANTHA, CHAKRABORTY & BALAKRISHNAN (2000)

Priyantha, N. B. und Chakraborty, A. und Balakrishnan, H. (2000), „The Cricket Location-Support System", in Proc. of the 6th Int. Conf. on Mobile Computing and Networking (MobiCom 2000), Boston, MA, USA, Seiten 32-43.

REGTP (2001)

Regulierungsbehörde für Telekommunikation und Post (2001), „Marktbeobachtung - Mobilfunkdienste", Web-Seite, URL: http://www.regtp.de/schriften/start/fs_08.html.

RIZZO, LININGTON & UTTING (1994)

Rizzo, M. und Linington, P. und Utting, I. (1994), „Integration of Location Services in the Open Distributed Office", Technischer Bericht 14-94, Computing Laboratory, University of Kent, UK.

ROUSSOPOULOS, KELLEY & VINCENT (1995)

Roussopoulos, N. und Kelley, S. und Vincent, F. (1995), „Nearest Neighbor Queries", in Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD '95), San Jose, CA, USA, Seiten 71-79.

SALTENIS *ET AL.* (2000)

Saltenis, S. und Jensen, C. S. und Leutenegger, S. T. und Lopez, M. A. (2000), „Indexing the Positions of Continuously Moving Objects", in Proc. of the 19th ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD 2000), Dallas, TX, USA, Seiten 331-342.

SAMET (1990)

Samet, H. (1990), „The Design and Analysis of Spatial Data Structures", ISBN 0-201-50255-0, Addison-Wesley, Reading, MA, USA.

SATYANARAYANAN (1996)

Satyanarayanan, M. (1996), „Fundamental Challenges in Mobile Computing", in Proc. of the 15th ACM Symp. on Principles of Distributed Computing (PODC '96), Philadelphia, PA, USA, Seiten 1-7.

SCHNEIER (1995)

Schneier, B. (1995), „Applied Cryptography: Protocols, Algorithms, and Source Code in C", ISBN: 0-471-12845-7, John Wiley & Sons, New York, NY, USA, 2. Auflage.

SCHILIT, ADAMS & WANT (1994)

Schilit, B. N. und Adams, N und Want, R. (1994), „Context-Aware Computing Applications", in Proc. of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, Seiten 85-90.

SCHILIT & THEIMER (1994)

Schilit, B. und Theimer, M. (1994), „Disseminating Active Map Information to Mobile Hosts", in IEEE Network Journal 8, 5, Seiten 22-32.

SCHILLER (2000)

Schiller, J. (2000), „Mobilkommunikation", ISBN 3-8273-1578-6, Addison-Wesley, München, Deutschland.

SEYBOLD (2001)

Seybold, C. (2001), „Positioning of Mobile Devices Using Network Information from GSM Devices", Diplomarbeit Nr. 1905, Fakultät Informatik, Universität Stuttgart, Deutschland.

SKALOUD & SCHWARZ (2000)

Skaloud, J. und Schwarz, K. P. (2000), „Accurate Orientation for Airborne Mapping Systems", in ISPRS Journal of Photogrammetry and Remote Sensing 66, 4, Seiten 393-402, International Society for Photogrammetry and Remote Sensing.

SPREITZER & THEIMER (1993)

Spreitzer, M. und Theimer, M. (1993), „Providing Location Information in a Ubiquitous Computing Environment", in Proc. of the 14th ACM Symp. on Operating System Principles (SOSP '93), Asheville, NC, USA, Seiten 270-283.

STALLINGS (1995)

Stallings, W. (1995), „Network and Internetwork Security Principles and Practice", ISBN: 0-02-415485-7, Prentice Hall, Englewood Cliffs, NY.

SUN (2001)

Sun Microsystems (2001), „Java™ 2 Platform, Enterprise Edition", Web-Seite, URL: http://java.sun.com/j2ee/.

TAYEB, ULUSOY & WOLFSON (1998)

Tayeb, J. und Ulusoy, O. und Wolfson, O. (1998), „A Quadtree-Based Dynamic Attribute Indexing Method", in Computer Journal 41, 3, Seiten 185-200.

UPNP (2001)

Universal Plug and Play Forum (2001), „UPnP Homepage", Web-Seite, URL: http://www.upnp.org.

VAN STEEN *ET AL.* (1998)

van Steen, M. und Hauck, F. und Homburg, P. und Tanenbaum, A. (1998), „Locating Objects in Wide-Area Systems", in IEEE Communications Magazine 36, 1, Seiten 104-109.

VIAG INTERKOM (2001)

Viag Interkom (2001), „Genion Produkte", Web-Seite, URL: http:// www.genion.de.

WANG (1993)

Wang, J. Z. (1993), „A Fully Distributed Location Registration Strategy for Universal Personal Communcation Systems", in IEEE Journal on Selected Areas in Communications 11, 6, Seiten 850-860.

WANT *ET AL.* (1992)

Want, R. und Hopper, A. und Falcao, V. und Gibbons, J. (1992), „The Active Badge Location System", in ACM Transactions on Information Systems 10, 1, Seiten 91-102.

WARD, JONES & HOPPER (1997)

Ward, A. und Jones, A. und Hopper, A. (1997), „A New Location Technique for the Active Office", in IEEE Personal Communications 4, 5, Seiten 42-47.

WEISER (1991)

Weiser, M. (1991), „The Computer for the 21st Century", in Scientific American 265, 3, Seiten 94-104.

WOLFSON *ET AL.* (1998)

Wolfson, O. und Xu, B. und Chamberlain, S. und Jiang, L. (1998), „Moving Objects Databases: Issues and Solutions", in Proc. of the 10th Int. Conf. on Scientific and Statistical Database Management (SSDBM '98), Capri, Italy, Seiten 111-122.

WOLFSON *ET AL.* (1999)

Wolfson, O. und Sistla, A. P. und Chamberlain, S. und Yesha, Y. (1999), „Updating and Querying Databases that Track Mobile Units", in Distributed and Parallel Databases Journal 7, 3, Kluwer Academic Publishers, Seiten 1-31.

WORBOYS (1995)

Worboys, M. F. (1995), „GIS: A Computing Perspective", ISBN: 0-7484-0065-6, Taylor & Francis, London, UK.

WÜNSCHE (1999)

Wünsche, W. (1999), „Detailentwurf eines allgemeinen Location-Service und Erstellung eines einfachen Prototyps", Diplomarbeit Nr. 1759, Fakultät Informatik, Universität Stuttgart, Deutschland.

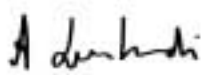
ZHAO (1997)

Zhao, Y. (1997), „Vehicle Location and Navigation Systems", ISBN: 0-89-00686-15, Artech House Publishers, Norwood, MA, USA.

ZIEGERT (2000)

Ziegert, T. (2000), „Nutzung von Verhaltensmustern zur Lokalisierung mobiler Objekte", Dissertation, Fakultät Informatik, Technische Universität Dresden, Deutschland.

Ich erkläre hiermit, dass ich, abgesehen von den ausdrücklich bezeichneten Hilfsmitteln und den Ratschlägen von jeweils namentlich aufgeführten Personen, die Dissertation selbstständig verfasst habe.



(Alexander Leonhardi)