Abstract. In this paper we describe the architecture and world model of the NEXUS platform for location- and context-aware applications. An important aspect of the NEXUS world model is that it also describes highly mobile objects, such as persons or vehicles. Therefore, we look in more detail at the management of location information in the NEXUS platform and the NEXUS Location Service, which provides for tracking the location of mobile objects with a high accuracy. In particular, we present a comparison of and simulation results for different protocols that are suitable for the frequent updating of highly accurate location information.

1. Introduction

Recently, Location Based Services have received much attention as a promising area of application for mobile communication systems. To provide location-awareness these applications rely on a built-in model of the real world (see, for example, [4]). They use this model, for example, to show the current position of the user on a street map or to perform spatial queries, like finding the restaurant next to the user's current position, The level of detail of the underlying model determines what functionality can be provided by such an application.

Up to now, all location-aware applications use their own special world model, which is expensive to create and to maintain, especially in case of an up-to-date detailed 3D model. The goal of the NEXUS project is to develop a platform that provides a detailed model of the real world to location-aware applications, both for in-door and out-door usage (see [7]). This model, called the Augmented World Model (AW-model), consists of representatives for real world objects and of virtual objects that provide, among other things, links to external information spaces like the WWW and mobile objects like the users of the system and cars or trucks. The model is kept up-to-date by integrating the update of sensor systems, for example to obtain the current position of mobile objects. Location-aware applications may query the current state of this model by using the Augmented World Querying Language and receive as answer information about the model described by the Augmented World Modeling Language (see [13]). We envision the Nexus platform as a middleware that connects providers and clients of location-based services.

An important aspect of the NEXUS platform is the management of the highly dynamic location information of mobile objects for which a special component, the NEXUS Location Service (LS), is responsible. The LS is especially designed for a possibly very high update rate and the varying accuracy of the location information that is necessary to support a variety of sensor systems (e.g., GPS [18] or an Active Badge system [17]). Traditional Spatial Databases (e.g., [5]) and Geographical Information Systems [21], which are used for the storage of spatial data, are not suited for this purpose, as they are optimized for more static data with fewer updates.

If the location information is acquired on the mobile object itself (e.g.,via GPS [18]), it has to be transmitted from the mobile object to a location server using wireless communication. In case of a distributed service, where the location information is cached or replicated on a number of servers, the information also has to be transmitted between location servers. To control the transmission of location information, different update protocols with varying properties can be used. In this paper we will also discuss the characteristics of different classes of update protocols and compare their performance (i.e., the number of messages required to guarantee a certain accuracy) by means of a simulation.

The remainder of the paper is structured as follows: in Section 2 we describe the architecture of the NEXUS platform and the Augmented World Model that is managed by it. Section 3 discusses the architecture of the NEXUS Location Service, which is responsible for managing the location information of mobile objects, while in Section 4 we look at the location update protocols that are required for transmitting highly accurate location information. Section 5 discusses related work and, finally, Section 6 concludes the paper and gives a short outlook on future work.

2. The NEXUS Platform

2.1 Augmented World Model

The AW-model [13] is the central information model of the NEXUS platform, providing the applications with an integrated, homogenous view on the available data. The most important types of information are models of geographic objects (e.g., buildings, streets or cities), models of mobile objects (e.g., NEXUS users), and virtual objects. Examples for virtual objects are Virtual Information Towers [9] or Virtual PostIts. They don't have a counterpart within the real world, but can be visualized by an appropriate application. Virtual Objects provide additional information or services for NEXUS users.

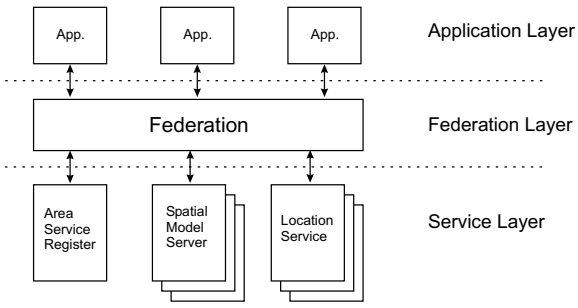

Fig. 1. Architecture of the NEXUS system.

We are using an object-oriented approach, which allows the AW-model to be easily extended. A subway station can, for example, be modelled as a subclass of *Station*, thus inheriting all attributes of *Station*. There are two categories of classes: members of the Standard Class Schema and the Extended Class Schema. The Standard Class Schema contains classes that we consider fundamental and have a well-known

structure, so they can be used by every application. There may be usage scenarios where additional classes are needed for special purposes. Such classes can be defined as sub-classes of members of the Standard Class Schema - as shown for the subway station above - and belong to the Extended Class Schema. As *SubwayStation* inherits all attributes of *Station*, it still can be used as a *Station*-object by applications that only know the Standard Class Schema, but special applications are needed to take advantage of the additional attributes of *SubwayStation*.

2.2 Architecture

Fig. 1 shows the main components of the NEXUS architecture. It is organized in three tiers: The Service Layer contains the servers that provide the data needed by the applications. Spatial Model Servers provide data about static geographic objects, and Location Servers return on demand the location of mobile objects. Each Spatial Model Server stores information about spatial objects within a particular area. For example there could be a Spatial Model Server that stores 2D and 3D-shapes of buildings located within the city of Stuttgart.

The Federation Layer contains NEXUS-nodes, which do not store data themselves, but receive requests from applications, look up appropriate servers, transmit the query to those servers and integrate the results into an homogenous result set (see [3]). Please recall that the WWW - the most popular open information system today - only provides the top (Web browsers) and the bottom layer (HTTP, FTP and other servers). Compared to the WWW, NEXUS needs the Federation Layer for two reasons: We want to present the data in a uniform and structured model with well-known semantics (which does not exist on the WWW) and we want to enable applications to perform spatial queries without knowledge about the respective servers. In contrast, a centralized approach would not necessarily need the federation layer, but its scalability would be poor and comprehension of different providers would be difficult. The functionality of the federation is implemented on a NEXUS-node, which also may host a Spatial Model or Location Server.

In brief, a query is processed by a NEXUS-node in the following way: Consider a client requesting data of users and buildings within a particular area. This query is sent to an arbitrary NEXUS-node. The NEXUS-node first has to distribute the query among the different services. Knowing that user is a subclass of *MobileObject* (stored at Location Servers) and building is a subclass of *StaticObject* (stored at Spatial Model Servers), the NEXUS-node creates two queries: One for users within the specified area and one for buildings within that area. The first query is sent to an arbitrary Location Server, which will contact additional Location Servers as necessary. The second query needs to be forwarded to the respective Spatial Model Servers. An additional component - the Area Service Register - is required to find out the addresses of those Spatial Model Servers. We will not discuss the Area Service Register further in this article, it can be regarded as a kind of geographic Domain Name System.

NEXUS-nodes and Spatial Model Servers are accessed via a query-response-protocol. The query contains a specification of the objects the application is interested in, e.g. objects within a particular area or of a particular type. The answer is a list of matching objects. We are using SOAP [22] as communication protocol because of its independency of the underlying transport protocol and programming language. As we will show in the following section, most transferred data is already encoded using XML.

3. NEXUS Location Service

The previous section has discussed the NEXUS platform for supporting location-aware applications. An important aspect of this platform is the management of the highly dynamic location information of mobile objects for which a special component, the Location Service (LS), is responsible. Within the NEXUS platform the Federation Layer calls the LS, whenever an AWQL-query concerns the position of such objects. An important requirement in the design of the LS is its scalability, as the NEXUS platform will

eventually have to manage a large number of objects, many of which will be mobile. To this end, the LS is being realized as a distributed service, whose servers are organized in a hierarchical architecture.

3.1 Functionality

Conceptually, the LS stores a location descriptor for each registered mobile object (called a *tracked object*). This location descriptor *ld(o)* consists of a position *ld(o).pos* and a guaranteed accuracy for the position information *ld(o).acc*, which depends on the sensor system and the update protocol (see Section 4). Together, these form a circular location area in which the actual position of the tracked object *rp(o)* is contained (see Fig. 2).

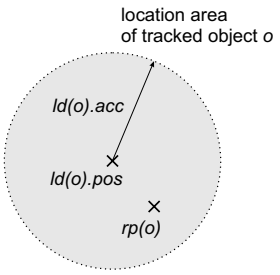

Fig. 2. Location area defined by *ld(o)*.

With the registration at the LS, a user can determine the maximum accuracy the LS is allowed to keep the location information with. Subsequently, the sensor systems only transfer their information with the minimum frequency to guarantee this accuracy. This is part of the security requirements for the LS.

For its tracked objects the LS can process the following basic queries, whose semantics were designed to consider the various accuracies.

A position query returns the location descriptor for the current position of a certain mobile object *o*. An example of its usage would be to determine the current position of a bus to estimate its delay.

$$posQuery(o) \rightarrow ld$$

A range query returns the identifiers and location descriptors for all objects that are inside of a given geographical area. It can be used, for example, to determine all persons currently waiting at a bus stop, in order to inform them about a delay.

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

Parameters of the range query are the queried area *a*, the accuracy that is requested of the location information *reqAcc*, and the degree to which their location descriptors must overlap with the queried area, *reqOverlap*. Mobile objects that do not meet these requirements are not included in the query results.

With a nearest neighbor query the tracked object that is nearest to a given position can be determined. Using a nearest neighbor query, a user will, for example, be able to determine the nearest taxi cab to his current position.

$$neighborQuery(p, reqAcc, nearQual) \rightarrow (nearestObj, nearObjSet)$$

The parameter *p* contains the queried position, *reqAcc* the accuracy that is required of the location information and *nearQual* the maximum distance from the nearest object with which an object is still considered a near object. Accordingly, the nearest object, considering the position in its location descriptor, is returned in *nearestObj*, and other near objects, as defined by *nearQual*, in *nearObjSet*.

Besides these queries, the LS will also be able to observe basic events, such as “object enters/leaves a room”, “object crosses a line” or “objects meet”.

3.2 Architecture

To be able to provide the scalability necessary for a possibly world-wide deployment of the NEXUS platform, the LS is organized in a hierarchical structure (similar to that of the Globe Location Service for software objects, see [16]). In [10] the architecture of the LS is described in detail. Each leaf server of this hierarchy is responsible for a certain geographical area, its service area, and stores the position and registration information for the mobile objects inside its area of responsibility. A tracked object sends updates of its location to the corresponding leaf server according to a protocol that guarantees a certain resulting accuracy (see [11]).

Higher-level servers have a service area which encompasses the service areas of all their children (see Fig. 3). For the mobile objects in this area, they store only a forwarding pointer to the branch that contains the position information of the object.

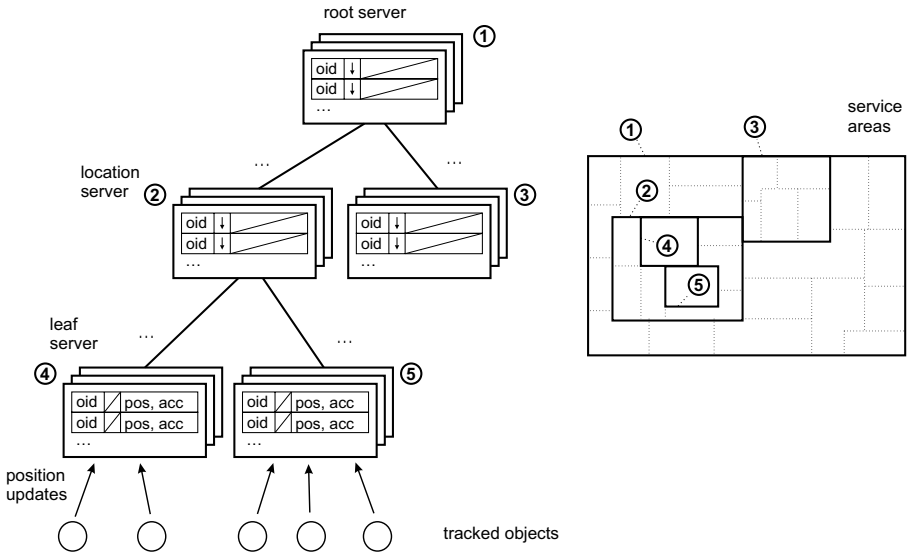

Fig. 3. The hierarchical architecture of the NEXUS Location Service.

When a tracked object moves from the service area of one leaf server to that of a new server, a handover is performed. Registration information and the management of the position information are transferred to the new server and the affected forwarding pointers on higher level servers are reset accordingly (see Fig. 4).

A query can be sent to any leaf server and is then forwarded to the appropriate server(s). In case of a position query, the query is forwarded upwards till a forwarding pointer to the queried object is found. It then follows the forwarding path to the appropriate leaf location server, whose location descriptor for that object is returned as the result of the query. A range query works similarly, using the overlap relationship between queried area and service areas to find the concerned leaf server(s).

Because updates and queries will have a high degree of locality, that is, they mostly concern nearby servers, the LS will be able to handle most queries without forwarding them upwards for more than a few levels.

3.3 Implementation

As a fast processing of queries and especially updates concerning location information is of great importance for the performance of the LS, the location information is managed in a special main-memory

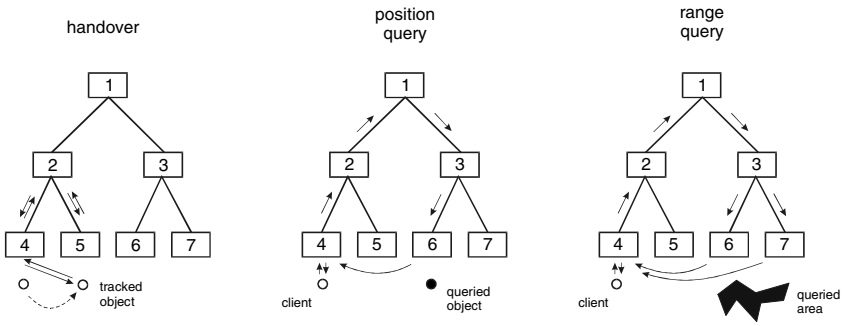

Fig. 4. Basic mechanisms of the NEXUS Location Service.

data-structure based on a Quadtree (see [14]), while the registration information is stored in a standard database. The volatile position information, which will be out-of-date after a server failure anyway, can be recovered from the mobile objects. A prototype of the LS has been implemented, which is able to process as many as 2000 remote updates and 1000 queries per second on a standard Solaris workstation.

4. Location Update Protocols

As the location information is required with a high accuracy on a location server and therefore has to be updated frequently, it is important to use a suitable update protocol. The components involved in updating the location information are as follows (see Fig. 5): The *source* of the location information (e.g., the mobile device of the user) receives information about the mobile object's position from an attached positioning sensor. The location information is required at a location *server*, where it can be queried by applications using the NEXUS platform. To this end, the location information can be either updated by the source or requested from the server. The goal is to use as few messages (updates and requests) as possible while still maintaining a requested accuracy of the position information available on the server.

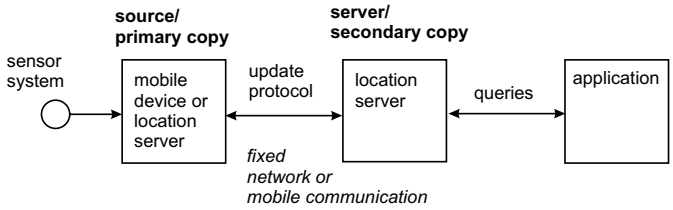

Fig. 5. Components involved in the updating of location information.

4.1 Protocol Classes

Update protocols for location information can be divided into three main classes, namely querying, reporting and combined protocols, where each class has a number of typical variants. A protocol is called a querying protocol, if the server decides when to request the location information from the source. In case of a reporting protocol, the source updates a local copy on the server. In Fig. 6 the possible variants of update protocols are shown; their most important representatives are described next:

Cached querying protocol: The cached querying protocol stores a cached copy of the last transmitted location information on the server. When the location information for the mobile object is queried by an

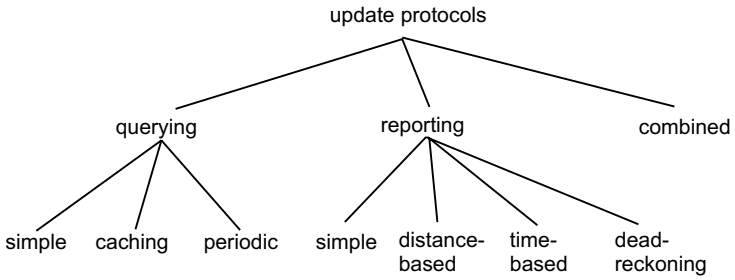

Fig. 6. Classification of protocols for updating location information.

application, the server estimates the accuracy of the cached copy and returns it, if it is considered to be accurate enough. Otherwise, the server has to send a request to the source like with the simple protocol. A *pessimistic cached querying protocol* (as discussed here) uses the distance, the mobile object may have traveled at its maximum speed, for the estimation of the accuracy.

Distance-based reporting protocol: The distance-based protocol sends an update of the location information, whenever the geographic distance between the current location of the mobile object and the last reported location becomes greater than a given distance threshold *D*. As this protocol sends more messages if the mobile object is moving fast and less messages if it is slower or stationary, it is often more efficient for objects that perform sporadic movements between periods of immobility. This is, for example, typical for users in an office environment or more so when tracking the location of pieces of office equipment.

Combined protocol: While a querying protocol can not be adjusted to different mobility characteristics of the mobile objects, a reporting protocol does not consider the query rate and the accuracy requested by the applications. With a combined protocol, which integrates the distance-based reporting protocol and the cached querying protocol, both of these features can be achieved. Similar to the distance-based reporting protocol, the source may update a secondary copy on the server to achieve a given spatial accuracy *D*. If the location information stored in the secondary copy is not accurate enough for a certain query, the server requests the information from the source as in a querying protocol. To minimize the total number of messages consisting of updates and queries, it has to be decided whether to update a secondary copy at all and what distance threshold *D* has to be used, depending on the mobility properties of the mobile object and on the queries by the applications. If the source monitors the mobility properties and the server those of the queries, they can adapt the properties of the protocol dynamically by increasing/decreasing the distance threshold *D* or by starting/stopping the updating of the secondary copy altogether.

Dead-reckoning: Dead-reckoning is an optimization of the distance-based protocol. Here, the server uses a prediction function to estimate the current location of the mobile object based on its last reported state, which contains its old location, speed and the direction of its movement and maybe also information about the future route of the object[1]. The source also calculates this estimated location using the same prediction function and sends an update when it differs from the actual location by more than a certain distance threshold *D*. Variants of the dead-reckoning protocol mainly differ in their prediction function. The *linear prediction dead-reckoning protocol* assumes that the mobile object keeps on moving along a line given by the reported position and direction with the reported speed. As shown in our simulations, this simple to implement protocol is in many cases already a big improvement compared to the distance-based update protocol, especially if the object is a car moving on a highway, reducing the number of required messages by up to 84%. More advanced dead-reckoning protocols have the mobile object report its future

[1] If speed and direction are not directly available, they can be inferred from the last n position sightings.

road and assume that is moving along this road with its last reported speed [20]. Or they match the mobile object's position and course to a map and assume that it is following the current road taking the most frequented direction at an intersection.

4.2 Simulation Results

To evaluate the presented update protocols, we have performed a number of simulations based on actual GPS traces, which we have recorded in the surrounding area of Stuttgart, Germany. As the performance of the protocols, especially that of the dead-reckoning protocol, depends to a great deal on the mobility characteristics of the mobile objects, we have recorded GPS traces for four different types of objects:

- a walking person
- a car in city traffic
- a car in inter-urban traffic
- a car on a freeway

The traces have been obtained using a Differential GPS receiver, which has an accuracy of 2-5 m and whose output has been written to a file every second.

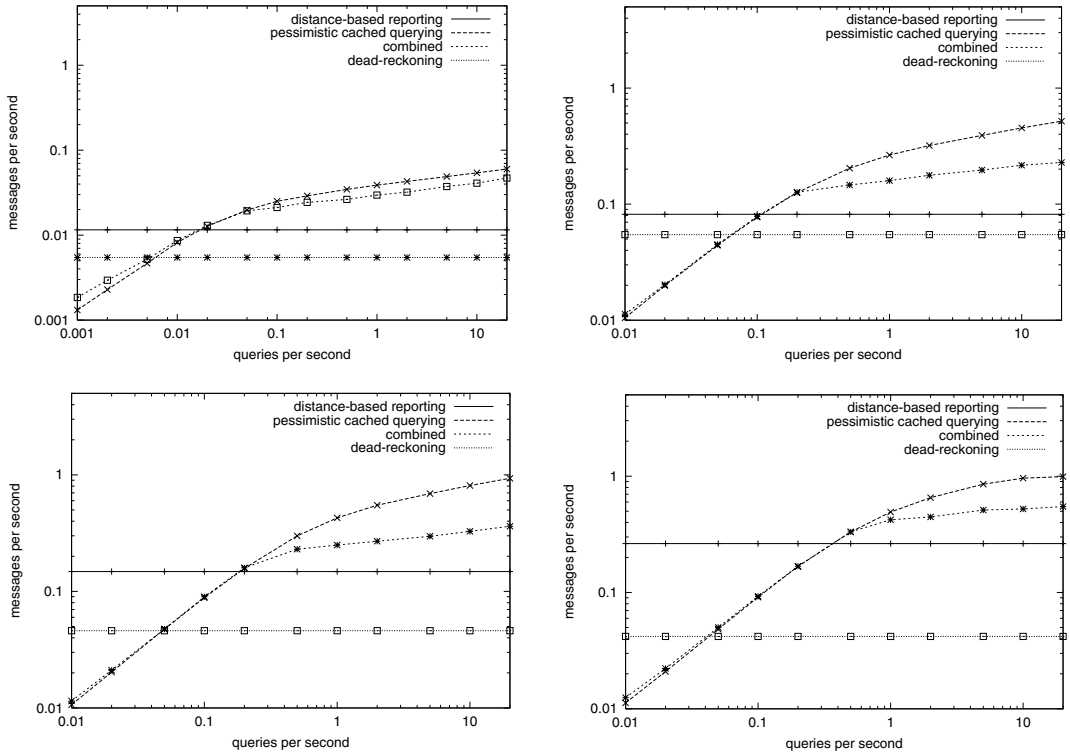

Fig. 7. Simulation results: A comparison of the number of messages transmitted with the distance-based reporting, the pessimistic cached querying, the combined and the linear prediction dead-reckoning protocol. From upper left to lower right for a walking user and a car in city, interurban or freeway traffic.

Fig. 7 shows the results of the simulation runs for the distance-based reporting, the pessimistic cached querying, the combined and the linear prediction dead-reckoning protocol. The results are shown seperately for the different types of mobile objects discussed above and with a requested accuracy of 100 m on the server. The simulation results show that the reporting protocols perform better for a higher query rate, while the querying protocols require less messages for a low query rate. The combined protocol, which has the same advantages as the querying protocol, namely that it can guarantee a per-query requested accuracy, also

requires less messages than the querying protocol for higher query rates (reducing the number of messages by up to 61%).

In all cases the dead-reckoning protocol is a great improvement on the similar distance-based reporting protocol, especially in case of a car on a freeway where it reduces the number of updates by 84%. It is also possible to use a dead-reckoning instead of a distance-based reporting protocol as basis for the combined protocol, which we have not done here, for reasons of simplicity.

5. Related Work

The management of location information is an important issue in the development of location-aware systems. A location service has been, for example, developed for location information of a local installation of the Active Badge system [17] or its much more accurate successor, Active Bat [6]. In the ParcTab system, which uses a location tracking technique similar to the Active Badge system, the location information of each user is managed and protected by a special user agent because of privacy considerations [15]. Leonhardt [12] proposes a wide-scale distributed location service that is independent of application and sensor systems. In his PhD-thesis, he examines fundamental issues of such a location service and classifies them in an abstract service model. In addition, he proposes a location model for the integration of different types of sensor data and requirements as well as policies for access control of a location service. The thesis also discusses architectural aspects of a global, general-purpose location service but does not propose or evaluate a specific architecture or update protocols.

Spatial data is usually stored in Spatial Databases [5] or Geographic Information Systems (GIS) [21]. While these are suitable for storing the static parts of the AW-model, they are, however, intended for large static data and complex queries and are not suited for the frequent updates necessary for the NEXUS Location Service.

An important aspect in the location management for the mobile phones in personal communication systems (PCS) is the management of the location information (e.g., [8]). In today's systems, for example GSM, the mobile phone reports its current location area to the PCS system each time it enters a new location area. Information about a mobile phone's current location area is stored in the Home Location Register (HLR). If a call arrives for a certain mobile phone, it is paged on a special channel in all communication cells belonging to the location area stored in the phone's HLR. Different protocols for updating and paging the location information have been discussed, for example a distance, a movement and a time-based protocol in [1] or the LeZi-update protocol in [2]. However, they only consider location information with a very low accuracy (a location area has a size of up to 40 km) and a dead-reckoning approach is not feasible in this context.

In the DOMINO project a database for the tracking of mobile objects is being developed (see [19]). One intended area of application is fleet management for the trucks of a transportation company. In [20] a dead-reckoning protocol for a previously known routes is described. Different variants of dead-reckoning strategies are compared with the goal of minimizing a given cost function. The results show, that under these conditions it is advantageous to use a protocol that adaptively adjusts the distance threshold at which the protocol sends a location update (*adaptive dead reckoning, adr*) as compared to a more simple strategy (*speed dead-reckoning, sdr*). To be able to deal with disconnections of the mobile communication link, a variant of these strategies has been designed that continuously decreases the distance threshold, if no updates are sent (*disconnection detection dead-reckoning, dtdr*). In our work, however, we do not assume a known route for the mobile object.

6. Conclusion

In this paper we described the NEXUS platform for location- and context-aware applications and discussed its architecture and world model. We looked in more detail at the management of location information and the NEXUS Location Service. In particular, we presented a comparison of and simulation results for different protocols that are suitable for updating highly accurate location information. Together they enable the NEXUS platform to track mobile objects with a high accuracy.

Future work in the area of update protocols will concentrate on developing and evaluating more advanced variants of the dead-reckoning protocols. This will, hopefully, result in an even higher reduction of the required number of update messages by relying on map information.

References

[1] A. Bar-Noy, I. Kessler and M. Sidi: Mobile Users: To Update or not to Update?, in *Wireless Networks*, 1(2), pp. 175-185, 1995.

[2] A. Bhattacharya and S. Das: 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, pp. 1-12, 1999.

[3] S. Busse, R.-D. Kutsche, U. Leser, H. Weber: *Federated Information Systems: Concepts, Terminology and Architectures*. Forschungsberichte des Fachbereichs Informatik, Bericht Nr. 99-9, Technische Universität Berlin.

[4] K. Cheverst, N. Davies, K. Mitchell and A. Friday: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project, in *Proceedings of the Sixth International Conference on Mobile Computing and Networking (MobiCom 2000),* pp. 20-31, ACM Press, 2000.

[5] J. Davis, *The IBM DB2 Spatial Extender: Managing Geo-Spatial Information within the DBMS*, IBM Cooperation, http://www.ibm.com/software/data/pubs/papers/spatial/, 1998.

[6] A. Harter, A. Hopper. P. Steggles, A. Ward and P. Webster, The Anatomy of a Context-Aware Application, in *Proceedings of the 5th Annual ACM/IEEE International Conference.on Mobile Computing and Networking (MOBICOM'99)*, Seattle, WA, USA, Seiten 59-68, 1999.

[7] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel and M. Schwehm: Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications, *Proceedings of the Fifth Annual ACM/ IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, Washington, pp. 249-255, ACM Press, 1999.

[8] D. Lam, Y. Cui, D. C. Cox and J. Widom: A Location Management Technique to Support Lifelong Numbering in Personal Communications Services, in *ACM Mobile Computing and Communications Review*, 2(1), pp. 27-35.

[9] A. Leonhardi, U. Kubach, K. Rothermel and A. Fritz: Virtual Information Towers - A Metaphor for Intuitive, Location-Aware Information Access in a Mobile Environment, in *Proceedings of the Third International Symposium on Wearable Computers (ISWC'99)*, San Fransisco, CA, USA, IEEE Press, 1999.

[10] A. Leonhardi, K. Rothermel: *Architecture of a Large-scale Location Service*, Technical Report, TR-2001-01, Faculty of Computer Science, University of Stuttgart, 2001.

[11] A. Leonhardi and K. Rothermel: A Comparion of Protocols for Updating Location Information, in *Special Issue on Internet Scalability: Advances in Parallel, Distributed and Mobile Systems, Baltzer Cluster Computing Journal,* 4(4), pp. 355-367*,* 2001 (to appear).

[12] U. Leonhardt: *Supporting Location-Awareness in Open Distributed Systems*, Dissertation, Imperial College of Science, Technology and Medicine, University of London.

[13] D. Nicklas, M. Großmann, T. Schwarz, S. Volz and B. Mitschang: A Model-Based, Open Architecure for Mobile, Spatially Aware Applications, in *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD '2001)*, 2001.

[14] H. Samet: *The Design and Analysis of Spatial Data Structures*, ISBN 0-201-50255-0, Addison Wesley,

Reading, MA, USA, 1990.

[15] M. Spreitzer and M. Theimer: Providing Location Information in a Ubiquitous Computing Environment, in *Proceedings of the 14th ACM Symposium on Operating System Principles (SOSP '93)*, Asheville, NC, USA, Seiten 270-283, 1993.

[16] M. v. Steen, F. Hauck, P. Homburg and A. Tanenbaum: Locating Objects in Wide-Area Systems, *IEEE Communications Magazine*, pp. 2-7, 1998.

[17] R. Want, A. Hopper, V. Falcao and J. Gibbons: The Active Badge Location System, *ACM Transactions on Information Systems,* 10(1), pp. 91-102, 1992.

[18] B. Hofmann-Wellendorf, H. Lichtenegger and J. Collins: *Global Positioning System: Theory and Practice*, Springer-Verlag, TELOS.

[19] O. Wolfson, B. Xu, S. Chamberlain and L. Jiang: Moving Objects Databases: Issues and Solutions, in *Proc. of the 10th Int. Conf. on Scientific and Statistical Database Management (SSDBM '98),* Capri, Italy, pp. 111-122, 1998.

[20] O. Wolfson, A. P. Sistla, S, Chamberlain and Y. Yesha: Updating and Querying Databases that Track Mobile Units, in *Distributed and Parallel Databases Journal,* 7(3), pp. 1-31, 1999, Kluwer Academic Publishers.

[21] M. F. Worboys, *GIS: A Computing Perspective*, Taylor & Francis, London, UK, 1995.

[22] World Wide Web Consortium: *Simple Object Access Protocol (SOAP) 1.1*, W3C Note 08 May 2000, http://www.w3.org/TR/SOAP.