Towards Scalable and Efficient Processing of Probabilistic Spatial Queries in Mobile Ad Hoc and Sensor Networks

Dominique Dudkowski*, Tobias Drosdol*, and Pedro José Marrón
Universität Stuttgart, Institut für Parallele und Verteilte Systeme (IPVS)
{dudkowski|drosdol|marron}@informatik.uni-stuttgart.de

Abstract: With the proliferation of sensor technology and advances in wireless communication, gathering, processing, and querying context information in mobile ad hoc and sensor networks becomes attractive and feasible. To support context-aware applications in such networks, efficient processing techniques for frequently used functionality, such as query and event management, are highly beneficial. In this paper, we discuss the challenges in efficient processing of spatial queries in mobile ad hoc and sensor networks. These comprise advanced query semantics based on inaccurate position information, efficient protocols and algorithms for data storage and query resolution, scalability with respect to network size, and support for mobile network nodes. We outline our concepts for solving these issues and show how they can be implemented in a suitable software architecture. Using current evaluation results, we show that our prototype implementation fulfills the stated challenges.

1 Introduction

The integration of sensor technology into mobile communication devices allows capturing the dynamic state of objects and phenomena in the physical world. Large amounts of context information become available allowing the realization of context-aware applications. Because various functionality, such as query processing, event management, or road navigation, is often used by such applications, their repeated implementation becomes a tedious task. An underlying data management facility that provides such fundamental services greatly simplifies application design. For location-aware applications in particular, *spatial queries*, such as range and k-nearest neighbor queries, are frequently used to retrieve data items by referencing to their location. They may be used, for example, to retrieve all taxis in a particular region or the nearest tools to one's own position in a factory building. While our Nexus platform ([Gr05]) already supports spatial queries on a large scale for infrastructure-based networks, an adequate query management facility does not exist for mobile ad hoc and sensor networks (MASNs). In such networks, nodes have integrated sensing, communication, and storage capabilities, but have to manage acquired data autonomously in a highly distributed manner. Due to limited resources of network nodes concerning energy, computing power, memory, and communication bandwidth, designing efficient strategies for query processing poses new challenges.

*funded by the German Research Foundation (DFG) within the Center of Excellence (SFB) 627.

The focus of this paper is to discuss the major challenges in supporting spatial queries in MASNs and to give an overview of our concepts for solving these issues. We do not intend to describe algorithmic details, which we address in other work. Rather than that, our aim is to give new impulses to context management issues of comparable complexity in related fields of context-aware systems for mobile networks.

The rest of our paper is structured as follows: In Sec. 2 we describe the system model that we use in the following discussions. Sec. 3 identifies the challenges related to query management in MASNs and Sec. 4 discusses related work. Our approach to tackle the stated problems and the integration of our concepts into a software architecture are described in Sec. 5 and 6, respectively. We discuss current evaluation results in Sec. 7 and conclude our paper in Sec. 8 with a summary and implications for future work.

2 System Model

The system model comprises a MASN operating in a service area that we denote by A. Two types of entities are located inside of A. *Perceivable objects* (PO_j) are an abstraction for physical objects, persons, or services that may be observed by sensor nodes. They may move by self-propulsion (persons, robots) or by being attached to carriers (persons, carts, conveyors). Mobile *sensor nodes* (SN_i) are part of the MASN and they may assume different roles (observers, data servers and clients), possibly multiple ones at a time. *Observers* are sensor nodes that capture the state of perceivable objects in their vicinity. They assemble a corresponding data object o_j consisting of the object identifier, object type, and observed location of the corresponding perceivable object, as well as the observation time. The observed location may be estimated, for example, from the observer's own position and the interpolated distance between the observer and the perceivable object. *Data servers* (DS_k) store a subset of data objects. Finally, *clients* provide access to the observed data by issuing spatial queries. Our goal is to implement spatial queries based on the given system model using only ad hoc communication between sensor nodes. Next, we discuss the challenges that arise from sensor data acquisition to the assembly of the final query result.

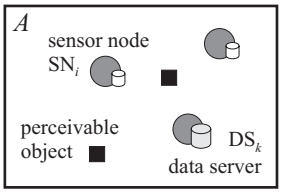

Figure 1: System model.

3 Challenges

The characteristics of MASNs lead to three principal challenges that must be solved with respect to the realization of a general spatial query management facility. First, *algorithm scalability* is essential to assure operation in networks with increasing size. Second, *node mobility* leads to the effect of moving data servers that must be compensated to retain the integrity of the data store. That is, it must be guaranteed that all objects relevant for query

resolution can still be found. Third, we must support advanced *query semantics* to account for inaccurate position information obtained from sensor observations.

With increasing size of the operated service area and number of sensor nodes, maintaining efficient operation of the overall query management facility is essential. Appropriate update propagation and storage protocols for acquired sensor data and strategies for aggregating query results from a highly distributed object store are required. Spatial queries are inherently suitable in that they support the design of a scalable solution. Because they access data based on location information, a geometric index can be designed that has a natural spatial relation to the coordinate system of the service area. In the case of range queries, it is intuitively suitable to store data near the specified range. The same is true for k-nearest neighbor queries, where objects nearest to a particular position are requested. Storing data close to its origin consequently allows to confine the resolution of a spatial query to a limited geometric region and provides the necessary scalability. Different mechanisms that provide a tradeoff between update propagation and query aggregation costs as well as the granularity by which the data model is partitioned and replicated across sensor nodes must be evaluated.

The increasing relevance of mobility in wireless network scenarios is a fundamental new issue when conceiving a general query management platform. Because data from sensor observations is stored directly on particular sensor nodes, data travels along with them and eventually loses its spatial coherence. If no countermeasures are taken, the communication overhead for finding particular data items increases constantly and eventually leads to network congestions and to the depletion of the nodes' energy. Relocation algorithms are required to "keep data in place" with minimum overhead in terms of communication costs and latency to assure continuous data availability for the aggregation of query results.

Physical processes involved in every sensor measurement inevitably lead to the acquisition of inaccurate sensor data. In the case of spatial data, the exact position of mobile objects is generally unavailable. In addition, limited update rates and the communication skew on each hop in a MASN aggregate to even greater inaccuracies. As a consequence, expressing the position of an object solely by using point coordinates is an inadequate simplification. In the case of a range query, the inclusion relation used to determine if the object is inside of the specified range is limited to a point inclusion test. However, due to localization errors, the object may be found in a particular *location area* that might only partially overlap with the given range. A threshold decision may then be much more appropriate to decide whether the object is considered to be inside or outside of the range. If the position data of individual objects is too inaccurate, some applications may even want to completely exclude these objects from the query result. The impact of inaccuracy on the outcome of a query might be even stronger if inhomogeneous position distributions are taken into account. The natural fact of position inaccuracies must be addressed by advanced query semantics, leading to *probabilistic spatial queries* that combine location areas with corresponding probability density functions.

4 Related Work

We classify existing work in the field of spatial query management into three categories: location and query semantics, algorithms for spatial queries in *stationary* sensor networks, location management and content location in mobile ad hoc networks.

In the first category, different query semantics were proposed based on interval calculations. The authors of [CP03] and [Ch04] introduce the notion of probabilistic threshold queries. In [CKP03] a classification of probabilistic queries is given, including definitions of probabilistic range and nearest neighbor queries. While the authors do not define spatial queries in more than one dimension, their concepts constitute a theoretical basis for the definition of spatial query semantics. Our work supports advanced query semantics for range and k-nearest neighbor queries based on inaccurate locations that also allow for excluding objects from query evaluation that do not possess a minimum accuracy.

In the second category, the Distributed Index for Features in Sensor Networks (DIFS, [Gr03]) and the Distributed Index for Multidimensional Range Queries in Sensor Networks (DIM, [Li03]) implement range queries in sensor networks for one and more dimensions, respectively. Both approaches apply localized storage for the geometric domain, which is beneficial for network scalability and query efficiency. Further, the Peer-Tree proposed in [DF03] was developed based on a centralized R-Tree that is used, e.g., in [SR01]. It is suitable for different types of spatial queries, in particular, for nearest neighbor queries. Most of these approaches concentrate on a single query type, and only the Peer-Tree may potentially be used for several types of spatial queries. Further, mobility is not addressed by any of the work, since all approaches are tailored to stationary networks. Last, only point coordinates are supported in the resolution of a spatial query, which does not allow to use more complex query semantics based on inaccurate positions.

In the third category, work that explicitly considers mobile nodes can be subdivided according to two purposes: location management and content location. Location management contains various approaches to query the position of network nodes for the purpose of geometric routing. Representative work includes the Grid Location Service (GLS, [Li00]) and the dead-reckoning-based location service proposed in [KD04]. Recent work in content location in mobile ad hoc networks is provided in [SH04] and [TV04]. These approaches allow to locate content by ID and have no support for spatial queries. As a consequence, they require a very different type of index structure with no relation to one required for spatial queries. Their work can be viewed supplementary to ours to provide a full-featured location service supporting position queries in addition to spatial queries.

To the best of our knowledge, all previous contributions do not address the challenges discussed in Sec. 3 in their combination. While spatial queries are implemented in the second category, node mobility is not considered. The third category supports mobility, but does not implement spatial queries. Finally, all approaches use just point coordinates and do not consider probabilistic query semantics. We will now address each of the challenges and show how they are solved in our architectural framework in Sec. 6.

5 Approach

5.1 Data Placement

We begin by defining a data placement strategy that is the basis for localized storage and constitutes the primary element for network scalability. Next, we show how data storage makes explicit use of that strategy during the observation of perceivable objects.

We recall from Sec. 2 that an observer captures the state of a perceivable object PO_j in form of a data object o_j. Each o_j contains the object identifier $o_j.\mathrm{id}$, object type $o_j.\mathrm{type}$, observed location $o_j.\mathrm{loc}$, and observation time $o_j.\mathrm{time}$. The observed location is composed of a location area and location probability density function (location pdf). The object is known to be located somewhere within this location area, and the location pdf defines the probability of being located in any subset of that area. Next, we subdivide the service area A into disjunct data sectors (Figure 2.a) and define two associations. First, we associate one data server DS_k that is located inside of a data sector S_r with that sector. Second, we associate data object o_j with each data sector S_r that overlaps with $o_j.\mathrm{loc}$. Note that multiple associations for the same object are possible (Figure 2.b). Finally, we define that o_j is mapped to data server DS_k iff both DS_k and o_j are associated with S_r.

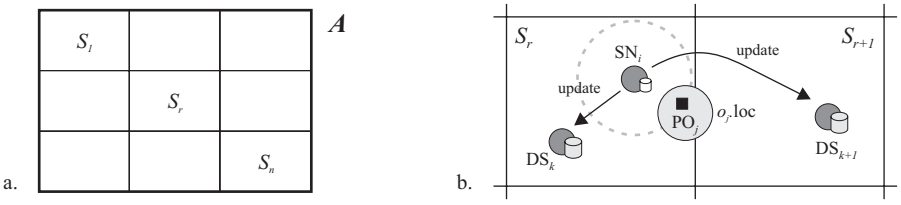

Figure 2: Subdivision of service area and placement of data objects.

In the first step of the data storage protocol, an observer constructs a data object o_j of the form $(o_j.\mathrm{id}, o_j.\mathrm{type}, o_j.\mathrm{loc}, o_j.\mathrm{time})$ that encapsulates the state of the corresponding perceivable object. Next, all data sectors are determined that overlap with the location area $o_j.\mathrm{loc}$. In Figure 2.b, $o_j.\mathrm{loc}$ overlaps with the data sectors S_r and S_{r+1}. According to the associations defined previously, copies of the data object are sent to data servers DS_k and DS_{k+1}. For that, a two-step routing is applied. In the first step, geometric routing, such as Greedy Perimeter Stateless Routing (GPSR, [KK00]), delivers the data object to a node inside of the respective data sector. Then, a local routing strategy is applied to finally reach the data server associated with that sector. At each data server, the object is inserted into the local database if a previous copy does not exist or its observation time is more current[1] than that of the existing copy. Note that each data object is annotated with a lifetime (counting from the observation time) and considered stale once its lifetime has elapsed. This way, copies that are not updated anymore (e.g. because the corresponding perceivable object is no longer detected by any observer) will eventually be deleted.

[1] We assume that sufficiently synchronized clocks exist, e.g., provided by GPS or a synchronization algorithm for mobile networks ([Röm01]).

5.2 Data Relocation

To maintain an efficient mapping between data servers and sectors, we use *data relocation* to move the data of an old server $\mathrm{DS}_{\mathrm{old}}$ to a new server $\mathrm{DS}_{\mathrm{new}}$. A *changeover condition* decides when relocation is initiated. A good strategy is to combine spatial and temporal predicates. If a server moves too far away from its associated data sector, the condition should trigger to limit the overhead for update and query routing. If a server remains nearly stationary inside of its associated sector, a timeout should trigger to avoid that single sensor nodes keep their server role for a long time and their energy is exhausted. In the second step, an election algorithm is used to determine a suitable sensor node that is eligible to become $\mathrm{DS}_{\mathrm{new}}$. For example, we may elect the node closest to the center of the corresponding sector, or the node that is expected to remain inside of that sector for a maximum time. In the final step, the contents of the database of $\mathrm{DS}_{\mathrm{old}}$ are transferred to $\mathrm{DS}_{\mathrm{new}}$. Our strategy transfers only those data objects that we do not consider stale, that is, whose lifetime has not yet elapsed. After relocation is complete, $\mathrm{DS}_{\mathrm{new}}$ takes over and thus guarantees storage locality.

5.3 Query Resolution Algorithms

We now describe two algorithms to resolve *probabilistic* range and k-nearest neighbor queries. Both algorithms involve an initial step in which the query is sent from the client to a *proxy node* that resolves the query on behalf of the client. For range and k-nearest neighbor queries, the node located closest to the center of the query range R and the query target position p_{NN} respectively, is an appropriate choice. Using GPSR, this node may be reached using greedy forwarding first, and then selecting the node visited twice during perimeter mode, which is guaranteed to be the closest node to the respective position.

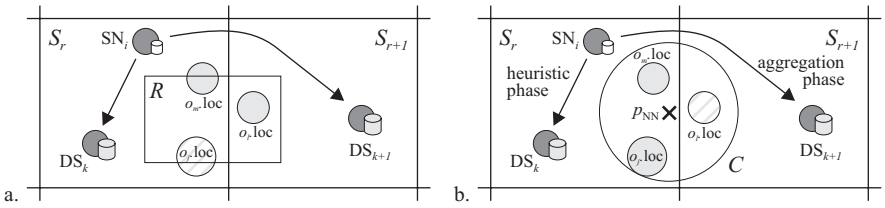

Figure 3: Query algorithms for range and k-nearest neighbor query resolution.

Probabilistic range queries (*p*RQs) for a geometric range R are resolved at the proxy node by aggregating the result from a number of *partial p*RQs. In the beginning, the proxy determines the set of data sectors overlapping with R (Figure 3.a). Each data server associated with one of these sectors is queried using a partial *p*RQ. Similar to data storage, routing to the data sectors is performed in two steps. At each data server, the partial *p*RQ is resolved locally. The algorithms for local resolution of partial *p*RQs depend on the precise query semantics and are out of the scope of this paper. Each partial result is returned to

the proxy that aggregates it into the final result. A timeout at the proxy terminates result aggregation if not all partial *p*RQs succeed. Finally, the proxy sends the result to the client.

The resolution of probabilistic k-nearest neighbor queries (*p*NQs) at the proxy node involves two phases: the *heuristic phase* finds initial nearest neighbor candidates, and the *aggregation phase* creates the final result. During the heuristic phase, the goal is to find k candidate objects that are close to the actual nearest neighbors. This is done by issuing a partial *p*NQ in the data sector that contains the query position p_{NN} (Figure 3.b). If no k candidates are returned, circumjacent data sectors are queried with increasing distance to p_{NN} until k candidates are available. From these candidates, our algorithm constructs the circle denoted by C with center p_{NN} that completely contains the location areas of the candidates. This property guarantees that all other objects that might be nearer to p_{NN} than any of the candidates are considered. During the aggregation phase, partial *p*NQs are sent to all data sectors that overlap C. At each data server, a partial *p*NQ is resolved that determines nearest neighbors from the objects stored in the server's local database. Up to k objects are returned to the proxy for each partial *p*NQ. The proxy aggregates received objects into the final query result. Again, a timeout terminates the aggregation if any partial result cannot be determined. Finally, the query result is returned to the client.

6 Software Architecture

Figure 4 shows the software architecture of our data management framework that implements the protocols and algorithms described in the previous section. The architecture is composed of two main parts: *routing* and *data management*. Routing comprises two layers with increasing level of abstraction with respect to the specification of routing goals. The *Routing Executive Layer* (REL) is responsible for carrying out low-level routing. It implements variations of geometric routing and extends GPSR ([KK00]). It is able to route a message to the node nearest to a geometric region (Location Routing Module) or to a specific node (ID Routing Module). It implements scoped flooding used to announce the location of database nodes, where the information is in turn used by the ID Routing Module. The *Routing Convergence Layer* (RCL) interfaces with the REL and data management and realizes complex routing goals. It provides the two-step routing used for data storage and query resolution by invoking the RCL multiple times.

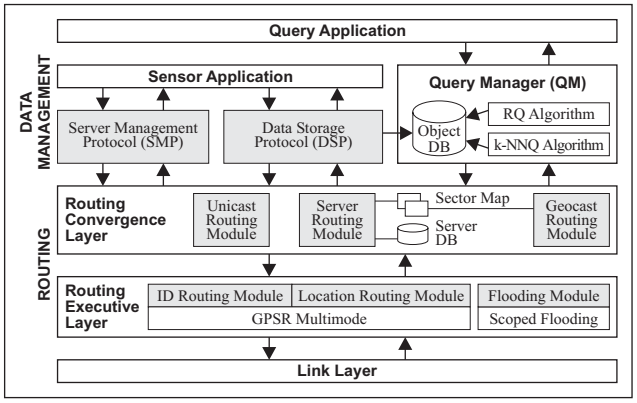

Figure 4: Software Architecture.

Data management implements the protocols and algorithms for data placement, data relocation, and query resolution. The *Data Storage Protocol* (DSP) implements the functionality described in Sec. 5.1. The DSP invokes the RCL possibly multiple times to route copies of data objects to the data servers to which the object maps, and updates each copy in the local database (Object DB) of these servers. The *Server Management Protocol* (SMP) performs data relocation as described in Sec. 5.2. Finally, the *Query Manager* (QM) integrates implementations of query algorithms, like the range and k-nearest neighbor query algorithm. It is modular and allows to add and remove implementations as needed. Each query algorithm can be invoked by applications and performs the resolution of a particular query instance by invoking the RCL each time a partial query is resolved.

7 Performance Analysis

In this section we state quantitative results to discuss the performance of our architecture. We have conducted our experiments in the ns-2 simulation environment using a service area of 400 $\times$ 400 m. We deployed 80 sensor nodes and 160 perceivable objects in the service area. Nodes and objects move according to the random waypoint mobility model with a fixed default speed of 1.5 m/s and 30 seconds of pause time. The transmission range of nodes is 150 m. We assume that a node is able to detect a perceivable object while its physical distance to the object is less than or equal to 20 m. Further, the location computed during an observation is assumed to be inaccurate by a disc with a radius of 10 m.

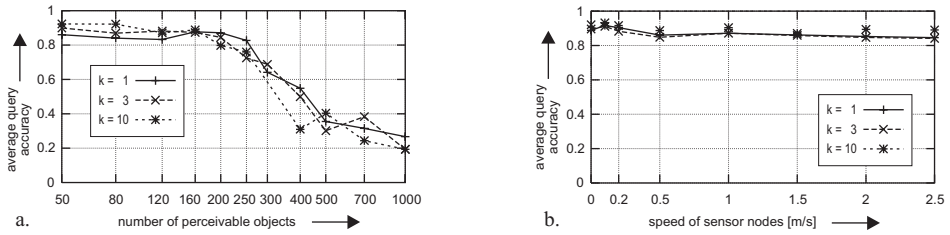

Figure 5: Average query accuracy for k-nearest neighbor queries and different values of k.

Figure 5.a shows the average *query accuracy* in relation to the number of perceivable objects for k-nearest neighbor queries. Query accuracy is the fraction of the objects returned by our algorithm that would also be returned from a global data structure containing the most recently observed location of all perceivable objects. With increasing number of these objects, the average distance between them decreases. Thus, there are more objects with a similar distance to the query reference position p_{NN}. As a consequence, ordering objects by that distance is much more influenced by inaccurate position information. This leads to a decrease in query accuracy with a growing number of objects, which may only be compensated by higher observation rates or additional location fusion strategies (which is out of the scope of this paper). Figure 5.b shows query accuracy as a function of node speed. The results confirm that mobility has virtually no effect on query accuracy for typical pedestrian speeds of up to 2.5 m/s.

Figure 6 shows results of various metrics as a function of the number of data sectors for k-nearest neighbor queries. Minimal values of the metrics, which indicate optimal performance of the algorithms, are found for small numbers of sectors. Average *query latency*, defined as the time necessary from issuing the query to returning the result to the client, is shown in Figure 6.a. It increases with the number of data sectors because more partial queries are required to resolve a nearest neighbor query. Since we do not send partial queries redundantly, the probability increases that at least one of them fails due to unresolved routing. This leads to an increasing number of cases where query timeouts occur at the proxy and thus, average query latency increases.

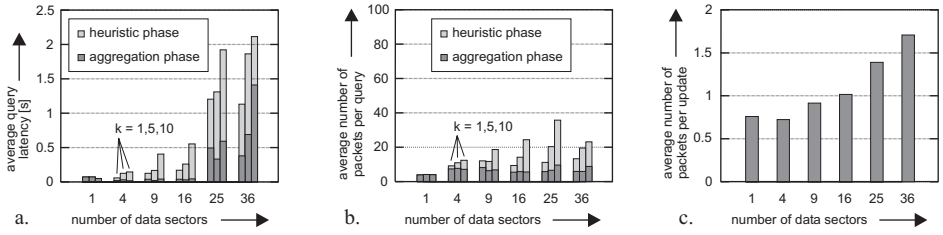

Figure 6: Average query latency, query costs, and update costs.

Figure 6.b shows the average *query costs*, defined as the total number of packets required to resolve a nearest neighbor query. Costs increases with the number of data sectors, because more partial queries are required for a larger number of data sectors. Note that with increasing k, query latency and query costs in Figure 6.a and 6.b increase because more partial queries are required. This is a direct consequence from the circle C that is determined according to Sec. 5.3, whose radius increases and which overlaps more data sectors with larger values of k. Figure 6.c shows the *update costs* for increasing number of data sectors, that is, the number of packets required to update a data object at potentially multiple servers. In the case of 4 sectors, update costs are optimal. For more data sectors, update costs increase because the location area computed by the data storage protocol according to Sec. 5.1 overlaps with more data sectors. Note that the average number of packets may be less than 1, since some observations are done by a data server that is responsible for storing the data object it has created.

8 Conclusion

In this paper, we discussed the challenges for efficient and scalable processing of spatial queries in mobile ad hoc and sensor networks. We have described our general approach that solves the problems of algorithm scalability, node mobility, and query semantics by exploiting localized data storage and efficient data relocation while taking into account general query semantics defined on location areas and probability density functions. We presented our software architecture that incorporates the algorithms into a general framework for spatial query processing in mobile ad hoc and sensor networks. Finally, we showed by quantitative analysis that our architecture achieves the stated goals.

Future work includes extension and optimization aspects. Extensions concern the inclusion of a topologic model to account for restricted movement of objects, for example, in urban areas. A more complex static data model requires partitioning approaches to consider the limited storage capabilities of sensor nodes. Further, the incorporation of position queries must be accomplished to provide all essential queries used in location-aware applications. Optimizations are to be done to packet aggregation on the update side, which we expect will tremendously decrease network traffic. On the query side, we see the potential for many optimizations when considering different timeout strategies together with partial query planning based on object densities in different parts of the service area. Further, redundant sending of partial queries may be used to increase query accuracy. Last, we have already designed a suitable cleanup protocol that increases consistency of multiple copies corresponding to the same perceivable object.

References

[CKP03] Cheng, Kalashnikov, Prabhakar: Evaluating Probabilistic Queries over Imprecise Data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 551–562, San Diego, USA, 2003.

[CP03] Cheng, Prabhakar: Managing Uncertainty in Sensor Databases. *ACM SIGMOD Record. Special Issue: Special Section on Sensor Network Technology & Sensor Data Managment*, 32(4):41–46, 2003.

[Ch04] Cheng et al.: Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data. In *Proc. 30th VLDB Conf.*, pp. 876–887, Toronto, Canada, 2004.

[DF03] Demirbas, Ferhatosmanoglu: Peer-to-Peer Spatial Queries in Sensor Networks. In *Proc. 3rd Int. Conf. on Peer-to-Peer Computing*, pp. 32–39, Linköping, Sweden, 2003.

[Gr03] Greenstein et al.: DIFS: A Distributed Index for Features in Sensor Networks. In *Proc. 1st IEEE Int. Workshop on Sensor Network Protocols & Applications*, pp. 163–173, Anchorage, USA, 2003.

[Gr05] Großmann et al.: Efficiently Managing Context Information for Large-scale Scenarios. In *Proc. 3rd IEEE Conf. on Pervasive Computing & Comm.*, Kauai Island, USA, 2005.

[KD04] Kumar, Das: Performance of Dead-Reckoning-Based Location Service for Mobile Ad Hoc Networks. *Wireless Comm. & Mobile Computing*, 4(2):189–202, 2004.

[KK00] Karp, Kung: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. 6th Int. Conf. on Mobile Computing & Networking*, pp. 243–254, Boston, USA, 2000.

[Li00] Li et al.: A Scalable Location Service for Geographic Ad Hoc Routing. In *Proc. 6th Int. Conf. on Mobile Computing & Networking*, pp. 120–130, Boston, USA, 2000.

[Li03] Li et al.: Multi-Dimensional Range Queries in Sensor Networks. In *Proc. ACM Conf. on Embedded Networked Sensor Systems*, Los Angeles, USA, 2003.

[Röm01] Römer: Time Synchronization in Ad Hoc Networks. In *Proc. 2nd ACM Int. Symp. on Mobile Ad Hoc Networking & Computing*, pp. 173–182, Long Beach, USA, 2001.

[SH04] Seada, Helmy: Rendezvous Regions: A Scalable Architecture for Service Location and Data-Centric Storage in Large-Scale Wireless Networks. In *Proc. 18th Int. Parallel & Distributed Processing Symp.*, p. 218a, Santa Fe, USA, 2004.

[SR01] Song, Roussopoulos: K-Nearest Neighbor Search for Moving Query Point. In *Proc. 7th Int. Symp. on Advances in Spatial & Temporal Databases*, pp. 79–96, Redondo Beach, USA, 2001.

[TV04] Tchakarov, Vaidya: Efficient Content Location in Wireless Ad Hoc Networks. In *Proc. IEEE Int. Conf. on Mobile Data Management*, pp. 74–85, Berkeley, USA, 2004.