Efficient Algorithms for Probabilistic Spatial Queries in Mobile Ad Hoc Networks

Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel
University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS)
{dudkowski|marron|rothermel}@informatik.uni-stuttgart.de

***Abstract*— With the proliferation of wireless communication and sensor technology, the importance of location-based applications has tremendously increased. In order to support these applications, services that implement algorithms for the efficient processing of spatial queries, such as range and k-nearest neighbor queries, are definitely beneficial. In this paper, we propose definitions of probabilistic range and k-nearest neighbor query semantics that take into account the inaccurate position information acquired from positioning systems. We introduce efficient algorithms for distributed storage of the dynamic information captured by positioning sensors on mobile network nodes, and for processing the aforementioned queries in mobile ad hoc networks. We show by evaluation that the studied algorithms incur small communication costs, and that the query algorithms return accurate results in a timely manner.**

I. Introduction

The design and implementation of applications for mobile ad hoc networks (MANETs) creates new challenges for computer scientists and developers since they need to provide efficient algorithms for data processing even in the presence of mobility [11], [17], [29]. In this setting there are a variety of application scenarios such as search-and-rescue operations, autonomous building surveillance, inter-vehicle networking or urban community networks (see also [21]) that motivate the need for novel approaches. In such scenarios, location-based applications and services that explicitly take into account the position of mobile users play a crucial role. In addition, the proliferation of sensors integrated on mobile devices allows to capture objects in the vicinity of mobile nodes, which tremendously increases the amount of dynamic data that must be handled in the mobile ad hoc network.

More specifically, spatial queries such as range and k-nearest neighbor queries, are frequently used in location-based applications to retrieve objects inside of a geometric region, or objects that are closest to a given position. The combination of mobility and the need for efficient algorithms for the evaluation of spatial queries and the underlying data storage give rise to two fundamental challenges, both of which are addressed in this paper.

First, since every sensor measurement is subject to inaccuracy, values that represent the position of objects in mobile environments are also inaccurate. Therefore, it is necessary to reconsider the semantics of position information (usually point coordinates in the literature) and to redefine them in order to take into account the degree of accuracy provided by physical sensors. For example, GPS [9] and WLAN-based systems like [28] typically allow us to determine the position of a mobile object only within a few meters which, depending on the scenario, unless explicitly handled could lead to erroneous results and limitations in the interpretation of query results. For example, consider a worker in a factory scenario using range queries to determine which robots are located within a restricted area. If the application uses point coordinates and does not take into account the confidence of each position reading, the user is unable to estimate the level of confidence of the answer and, in the end, she might not be able to know for sure which robots are located in the area. In this case, it seems clear that point coordinates do not provide enough information and that extensions towards more expressive location semantics, such as *probabilistic* spatial query semantics, are needed.

The second challenge deals with the design and implementation of spatial query algorithms that leverage the information contained in these new semantics, while at the same time, providing scalability and efficiency even in the presence of mobility. Since MANETs are expected to be composed of large numbers of mobile devices that cover a service area, it is essential to design algorithms that scale with both, the number of devices and the size of the service area. Two issues are relevant in this context. First, data is generally only relevant within a certain scope (locality of information) which, in some cases, might be defined by the user. Secondly, spatial queries operate on the accounts of position information. Therefore, algorithms for the evaluation of spatial queries need to take the geometry of the service area into account. Under these circumstances, *localized* data storage and query processing can be then used to provide the necessary scalability. Additionally, in order to account for node mobility it is possible to design efficient handoff algorithms that ”keep the data in place”.

In this paper, we provide new definitions for probabilistic spatial query semantics and present novel algorithms for the evaluation of probabilistic range and k-nearest neighbor queries in mobile ad hoc network. We further present algorithms that efficiently store data gathered by mobile nodes on dedicated data servers that tightly integrate with the proposed query algorithms to achieve overall efficient operation. As shown in the experimental evaluation section, our algorithms scale to large networks and handle node mobility in an efficient way.

The rest of our paper is structured as follows. Sec. II discusses related work in the fields of spatial queries. In

Sec. III and IV, we define the semantics of range and k-nearest neighbor queries based on inaccurate position information and present our system model. Sec. V describes our algorithms for data storage and the processing of range and k-nearest neighbor queries in MANETs. We analyze the performance of our update and k-nearest neighbor query algorithm in Sec. VI. Sec. VII concludes this paper and states implications for future work.

II. Related Work

To the best of our knowledge, no previous work exists that integrates algorithms for data storage with suitable algorithms for processing probabilistic spatial queries in MANETs.

Related Work	**Network Model**	**Query Type**	**Object Locations**
RLS [2] GLS [15] HLS [14]	MANET, mobile WSN	position queries	N/A
GHT [20] RR [24] GCLP [26]	stationary WSN MANET, mobile WSN MANET	ID-based queries ID-based queries ID-based queries	 N/A
DIFS [12] DIM [16] Peer-Tree [7] STQP Frame-work [6] KPT [27]	 stationary WSN	1D range queries multi-dimensional range queries k-NNQ; reverse, constrained NNQ historical range queries k-NNQ	 point coordinates
NNQ [22] k-NNMP [25] CNN [10] RNN [1] GNN [18]	centralized systems	k-NNQ k-NNQ constrained NNQ reverse, k-NNQ group NNQ	 point coordinates
FNNQ [23]	federated databases	k-NNQ	accurate coordinates
Our Work	mobile WSN, MANET	range, k-NN queries	location areas and pdfs

TABLE I
Related work in the field of spatial queries.

Query algorithms can be divided into several groups according to Table I. The first group (Randomized Location Service (RLS) [2], Geographic Location Service (GLS) [15], and Hierarchical Location Service (HLS) [14]), contains representative work on location services suitable for mobile ad hoc networks (MANETs) and also WSNs. For the purpose of location management and communication based on node addresses, only position queries are supported. For the same reason, only network nodes may be target objects, which simplifies the management of position information significantly. We additionally support the observation of objects in the physical world by sensors integrated into the network nodes and contribute suitable algorithms for the storage of data gathered by network nodes and for processing range and k-nearest neighbor queries based on that data.

In the second group, we have included algorithms for ID-based queries. Geographic Hash Tables (GHT) [20], Rendezvous Regions (RR) [24], and the Geographic Content Location Protocol (GCLP) [26] are designed for ID-based queries. RR are a generalization of the GHT for *mobile* networks using regions instead of points for rendezvous between data updates and queries. GCLP allows to locate content in a MANET based on intersections between update and query message paths. As in the case of location management, ID-based queries require an index that builds on the ID domain, as opposed to spatial queries. In both of the previously mentioned groups, location semantics are not relevant to query processing and thus omitted. In our work, we define query semantics that build on more complex location semantics since they are essential during the processing of spatial queries.

The third group contains work on spatial queries in wireless sensor networks where sensor nodes are considered stationary. Whereas the Distributed Index for Features in Sensor Networks (DIFS) [12] and the Distributed Index for Multi-Dimensional Range Queries (DIM) [16] support only range queries in one or more dimensions, the Peer-Tree approach [7] is suitable for spatial queries in general. The authors of [6] and [27] propose algorithms for processing historical range and k-nearest neighbor queries, respectively. Their approaches use simple data collection strategies that may potentially lead to very inefficient processing of the query itself. Common to all approaches is that they were designed for *stationary* networks. On the other hand, we concentrate on MANETs, which requires explicit handling of mobility so that the integrity of the data store is retained.

Exemplary work on nearest neighbor queries for centralized systems is summarized in the fourth group. Variants include the classic k-NNQ found in [1], [22], [25], constrained NNQ [10], reverse NNQ [1], and group NNQ [18]. The Peer-Tree algorithm [7] is a direct translation of these centralized P2P indexes to wireless sensor networks and therefore, does not address node mobility. Obviously, these approaches build on a centralized system, which is fundamentally different from the dynamic distributed network model we consider in our work.

Finally, the authors of [23] present distributed algorithms for processing *federated* k-nearest neighbor queries (FNNQ) over loosely coupled data sources connected through the internet, hence, they do not consider mobile nodes that play the role of data servers. The data model used only includes accurate locations of stationary two-dimensional objects and does not consider inaccurate positions of mobile objects.

The discussed approaches have in common that locations, either in one or two dimensions, are given in terms of accurate coordinates. In contrast, we will define probabilistic range and k-nearest neighbor queries in the next section that are fully implemented in our algorithms presented in Sec. V.

III. Concepts

Let us now formalize the notion of position in the presence of sensor measurements that lead to inaccurate position information and then define our semantics for probabilistic range and k-nearest neighbor queries.

The general notion of value uncertainties is well established in the literature and in the field of probabilistic queries used in [3], [4] and [5]. We apply these concepts to the case of

two dimensions in Cartesian space and provide Definitions 3.1 to 3.3 as the basis for our query semantics. We define the *location* of an object as the combination of a *location area* and a *location probability density function (pdf)* for an object o_j, as illustrated in Fig. 1.a.

Definition 3.1: The **location area** L_j of an object o_j is the smallest subset $L_j \subseteq \mathbb{R}^2$ inside of which object o_j is located with probability 1.

Definition 3.2: The **location pdf** $\varrho_j(X)$ of an object o_j with $X = (x, y) \in L_j$ is any two-dimensional pdf over L_j that satisfies the condition $\int_{L_j} \varrho_j(X) dL_j = 1$.

Note that the equality holds because of our assumption that object o_j is always located inside of L_j. The location pdf can be used to compute the probability P_j that an object o_j is located inside a subset of its location area $L'_j \subseteq L_j$, or written formally, $P_j = \int_{L'_j} \varrho_j(X') dL'_j$.[1]

Definition 3.3: The **location** of an object o_j is defined as the pair composed of location area L_j and location pdf ϱ_j of o_j,

$$\mathrm{loc}_j := (L_j, \varrho_j)$$

Example 3.1 (location): A concrete example of a location is the use of circular-shaped location areas and two-dimensional uniform location pdfs. Then, the location of an object may be specified as $\mathrm{loc} = (L, \varrho) := ((X - M)^2 \leq r^2, 1/\pi r^2)$ where X is any position inside of L, and M and r denote the center and radius of L, respectively.

A. Probabilistic Range Query

Using the notion of location, we now define semantics for probabilistic range queries. Previous work [3]–[5] has not considered the fact that the position of some objects may be too imprecise to be able to return an appropriate answer. We make the extension of Equation (1) in [8] to two dimensions in Cartesian space. We further introduce an additional query parameter that allows us to omit data objects from query processing if the degree of inaccuracy in the position information of these objects is too large. Our query semantics consists of two parts: the *inclusion condition* and the *inaccuracy threshold*. We have illustrated these concepts in Fig. 1.b. By R we denote the range query region.

Definition 3.4: The **inclusion condition** for an object o_j with positions $X \in L_j$ is satisfied iff the probability that o_j is located inside of a geometric region R is greater than or equal to the *inclusion threshold* $0 < P_{\mathrm{PRQ}} \leq 1$, and is given by:

$$P(X \in R) = \int_R \varrho_j(X) dR \geq P_{\mathrm{PRQ}}$$

Example 3.2 (inclusion condition): Assume the inclusion threshold $P_{\mathrm{PRQ}} = 0.5$. In Fig. 1.b, the inclusion condition is never satisfied for L_1 because $L_1 \cap R$ is empty. It is always true for L_3 and L_4 because their intersection with R is completely contained in R. For L_2, the inclusion condition is not satisfied for $P_{\mathrm{PRQ}} = 0.5$.

[1]Without loss of generality, we make the assumption of independent probabilities in the remainder of this paper, which does not impact the formulation of our concepts or the algorithms presented in Sec. V.

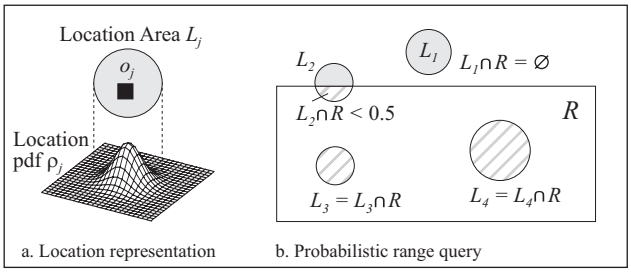


Fig. 1. Locations and range queries.

Definition 3.5: The **inaccuracy function** $F_\alpha(\mathrm{loc}_j) = F_\alpha(L_j, \varrho_j)$, whose argument is a location composed of a location area and a location pdf according to 3.3, evaluates to a single value $\alpha_j \geq 0$ that describes the degree of inaccuracy of object o_j's location. The value $\alpha_j = 0$ represents an accurate location.

Example 3.3 (inaccuracy function): Let the location of an object, defined as in Example 3.1, be $\mathrm{loc} = (L, \varrho)$ with $L := (X - M)^2 \leq r^2$ and $\varrho := 1/\pi r^2$. The degree of inaccuracy is defined using the location area only, because the uniform distribution does not give any additional information on the degree of inaccuracy. Therefore, we may define the inaccuracy function to be $F_\alpha(\mathrm{loc}) = F_\alpha(L, \varrho) = F_\alpha(L) = \sqrt{L/\pi}$. That is, we use the radius of the location area to quantify the degree of inaccuracy of location loc. While $F_\alpha(L) = 0$ represents a point coordinate (with zero radius), any positive number may be used to express potentially high degrees of inaccuracy. The maximum degree of inaccuracy may be bound by the maximum error of a particular localization technology.

Definition 3.6: The **inaccuracy threshold** $\alpha \geq 0$ defines the maximum degree of inaccuracy allowed for an object to take part in the evaluation of a spatial query.

The inaccuracy threshold is used to allow an application to specify that only objects whose position is not beyond a maximum degree of inaccuracy will be considered in a query.

Example 3.4 (inaccuracy threshold): In Example 3.1, we may define the inaccuracy threshold for the given location by a maximum tolerable radius of a given location area, for example, $\alpha = 5$.

Definition 3.7: Given the range query region R, the inclusion threshold P_{PRQ}, the inaccuracy function $F_\alpha(\mathrm{loc}_j)$ according to Definition 3.5, and the inaccuracy threshold α, the **probabilistic range query result**, denoted by $\mathrm{R}_{\mathrm{PRQ}}$, is defined as:

$$\mathrm{R}_{\mathrm{PRQ}} = \{o_j \mid P(X_j \in R) \leq P_{\mathrm{PRQ}} \wedge F_\alpha(\mathrm{loc}_j) \leq \alpha\}$$

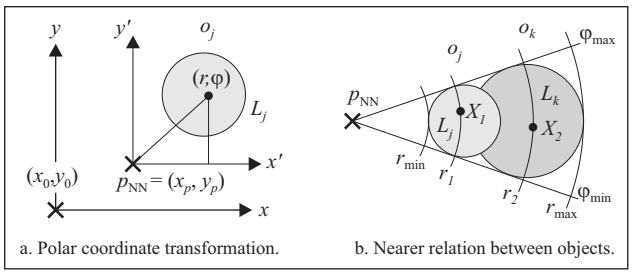
Intuitively, any object that is contained in R with the given minimum probability P_{PRQ} and whose degree of location inaccuracy is smaller than or equal to the inaccuracy threshold is included in the query result $\mathrm{R}_{\mathrm{PRQ}}$.

B. Probabilistic k-Nearest Neighbor Query

In [3] the authors define a type of nearest neighbor query such that the result set *definitely* contains the 1-nearest neighbor, which might lead to an indefinite and potentially large number of returned data objects. The proposed query semantics do not cover the case where $k > 1$ and they do not allow to specify a limit in the degree of inaccuracy that is allowed for objects to take part in the query evaluation. In the following, we propose semantics for k-nearest neighbor queries that defines exactly k objects that are considered nearest to a reference location given an inaccuracy function and threshold.

For that purpose, we define the *nearer* relation for two objects o_j and o_k based on Definitions 3.1 to 3.3. This relation allows us to determine *exactly* k nearest objects relative to a position p_{NN}.

Let $p_{\mathrm{NN}} = (x_p, y_p)$ be the reference coordinate from which the k nearest objects are to be determined. To simplify our notation, we introduce a polar coordinate system whose origin is p_{NN}. Fig. 2.a illustrates the location area L_j of an object in a polar coordinate system. The location pdf is then rewritten as $\varrho_j(X')$, with $X' = (r, \varphi)$, where $r^2 = (x-x_p)^2+(y-y_p)^2$ and $\tan\varphi = (y - y_p)/(x - x_p)$.



a. Polar coordinate transformation. b. Nearer relation between objects.

Fig. 2. Notion of nearer.

The definition of the nearer relation is illustrated in Fig. 2.b for the location areas L_j, L_k of two objects o_j, o_k. Intuitively, o_j is closer to p_{NN} than o_k if o_j is closer in "most of the cases". In Fig. 2.b, o_j is closer to p_{NN} for all pairs of positions $X_1 \in L_j, X_2 \in L_k$ for which X_1 is closer to p_{NN} than X_2. By integrating over products $\varrho_j \cdot \varrho_k$ of the location pdfs of o_j, o_k for all pairs of positions for which o_j is nearer to p_{NN} than o_k, we obtain the following definition:

Definition 3.8: The probability $P_{jk}|_{p_{\mathrm{NN}}}$, by which an object o_j is nearer to position p_{NN} than o_k, is defined as:

$$P_{jk}|_{p_{\mathrm{NN}}} = \int_{\substack{r_1=r_{\min}\\ \varphi_1=\varphi_{\min}}}^{\substack{r_1=r_{\max}\\ \varphi_1=\varphi_{\max}}} \varrho_j(X_1) \left[\int_{\substack{r_2=r_1\\ \varphi_2=\varphi_{\min}}}^{\substack{r_2=r_{\max}\\ \varphi_2=\varphi_{\max}}} \varrho_k(X_2) dR_2 \right] dR_1,$$

where $X_1 = (r_1, \varphi_1)$ and $X_2 = (r_2, \varphi_2)$, as well as $dR_1 = r_1 dr_1 d\varphi_1$ and $dR_2 = r_2 dr_2 d\varphi_2$.

Definition 3.9: For two objects o_j, o_k, object o_j is nearer to the coordinate p_{NN} than o_k if the following holds:

$$o_j < o_k|_{p_{\mathrm{NN}}} \iff P_{jk} > 0.5$$

In other words, o_j is nearer to p_{NN} than o_k iff the probability that o_j is nearer than o_k is greater than 0.5. It is also possible that o_j and o_k have the same distance to p_{NN}:

Definition 3.10: Two objects o_j, o_k, have the same distance to p_{NN} if the following holds:

$$o_j = o_k|_{p_{\mathrm{NN}}} \iff P_{jk}|_{p_{\mathrm{NN}}} = P_{kj}|_{p_{\mathrm{NN}}} = 0.5$$

The definition of the k-nearest neighbor query result R_{PNQ} is based on Definition 3.9 and 3.10. In addition, we use the inaccuracy threshold of Definition 3.6 to only include objects in the query processing whose location is not beyond a particular inaccuracy threshold α.

Definition 3.11: The **probabilistic k-nearest neighbor query result**, denoted by R_{PNQ}, is defined as:

$$\begin{aligned} &R_{\mathrm{PNQ}} = \{o_1, \ldots, o_k\} \ : \\ &\forall o_r \in R_{\mathrm{PNQ}} : F_\alpha(\varrho_j) \leq \alpha \ \wedge \\ &\quad \forall o_r \in R_{\mathrm{PNQ}}, o_s \notin R_{\mathrm{PNQ}} : \\ &\quad \alpha_s \leq \alpha \Rightarrow o_r \leq o_s\,|_{p_{\mathrm{NN}}} \end{aligned}$$

Note that if the property in Definition 3.10 holds for some objects the query result may be non-unique.

IV. System Model

Our system model considers two different types of entities: *perceivable objects* and *mobile nodes*, both located inside of a fixed geographic *service area*, which we denote by A. For ease of discussion and without loss of generality, we assume a fixed number of objects and nodes inside of A.

Perceivable objects, denoted by PO_j with $1 \leq j \leq m$, are an abstraction for any kind of entity whose state can be perceived. They may correspond to physical objects and people, or even to mobile nodes themselves whose state, in particular, location, may be of interest. Every PO_j has a physical position $q_j(t) \in A$ expressed by means of Cartesian coordinates. Due to mobility, positions are a function of time. Mobile nodes MN_i, with $1 \leq i \leq n$, are located at physical positions $p_i(t) \in A$. Mobile nodes may assume any of the following roles: observers, data servers, or query clients. *Observers* are able to perceive objects PO_j in their vicinity and create a corresponding data object o_j that contains a unique object identifier $o_j.\mathrm{id}$, the observed location $o_j.\mathrm{loc}$, and the observation time $o_j.\mathrm{tobs}$; *data servers*, denoted by DS_k are able to store copies of data objects in a local database; *query clients* are able to execute spatial queries. Multiple roles may be assumed by mobile nodes at the same time.

V. Algorithms

The goal of this section is to design an integrated set of algorithms for the storage of imprecise position information and the processing of probabilistic range and k-nearest neighbor queries that fulfills the following properties:

- support for location semantics as defined in 3.3;

- algorithm scalability as the size of the service area and the number of nodes increase;
- algorithm efficiency by localized data storage and query evaluation;
- resilience of the distributed data store to node mobility.

Concerning the family of spatial queries, we can identify the following additional important characteristics. On one hand, the argument of a range query is a fixed geometric region, which allows us to immediately determine the geometric region in the network where objects may have been observed. On the other hand, determining a fixed geometric range that includes the k objects of a k-nearest neighbor query is not possible, because it is unknown in advance how far these objects are located from the query position. This important difference is addressed by our data storage algorithm to guarantee efficient processing of both types of queries.

In Sec. V-A we introduce a mapping as the basis for our algorithms that defines which data objects are stored on which data servers, and we introduce our data storage algorithm. Sec. V-B describes our relocation strategy to tackle the problem of data server mobility. In Sec. V-C and V-D, we describe in detail our algorithms for processing probabilistic range and k-nearest neighbor queries, respectively.

A. Data Storage

The data mapping approach used is based on a subdivision of the service area A into *data sectors* $S_r \subseteq A$ similar to existing approaches (e.g. [15], [24]). The difference of our approach is that we allow *arbitrary* geometric shapes of data sectors so that adaptation to particular underlying topographies or node distributions (e.g. in urban scenarios) is possible. We assume that A is completely covered by the union of the data sectors and that sectors do not overlap.

Data objects are mapped to data servers in two steps. First, we define that each data sector S_r is associated with exactly one data server DS_k at any point in time. In general, the association holds if $S_r \cap L_i \neq \varnothing$, where L_i is the location area of DS_k according to Def. 3.1, but it is possible that DS_k is temporarily located outside of its associated data sector. Second, we define that each data object o_j is associated with data sector S_r if the location area L_j of o_j overlaps with S_r, that is, if $L_j \cap S_r \neq \varnothing$. Finally, we define that object o_j will be stored in the local database of data server DS_k if both are associated with the same data sector S_r.

Our data storage algorithm delivers data objects generated from observations of perceivable objects to data servers according to the defined data mapping. Let MN_i be an observer capturing the state of a perceivable object PO_j. An *observation* may occur in two situations: either MN_i *discovers* PO_j when they approach each other, or PO_j was not observed for a fixed time interval while being within sensing range of MN_i. Upon observation, MN_i creates o_j from the perceived state of PO_j. Then, MN_i determines which data servers DS_k will store a copy of o_j according to the data mapping. Fig. 3 shows the situation where L_j overlaps with two data sectors S_r and S_{r+1}. It follows that two updates, each containing a copy of o_j, are sent to DS_k and DS_{k+1}. For that purpose, we implemented a suitable unicast protocol based on GPSR [13] that sends an update in two steps. First, the update is sent towards the data sector S_r associated with o_j. This requires only a geometric position of the target sector, from which we assume that it is known to all nodes. Then, a suitable local routing protocol, such as AODV [19], may be used to forward the packet to DS_k. Upon reception of the copy of o_j at a data server DS_k, the object is inserted into the local database if the observation time is more current than that of an existing copy of o_j, or if a copy does not exist.

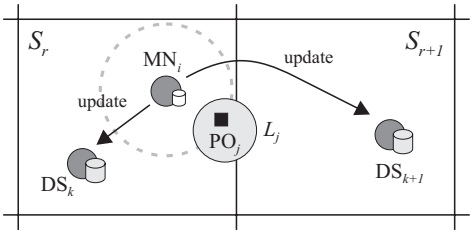


Fig. 3. Data storage algorithm.

B. Data Relocation

Relocating the contents of a server's local database is essential for keeping data 'in place' while servers move between data sectors. Due to the lack of space, we will describe only briefly our relocation strategy. Based on a *changeover condition* a data server DS_i decides to relocate its database contents to another server DS_j. To consider different scales of server mobility, the predicate of the changeover condition combines spatial and temporal aspects. For low-mobile servers, a timeout limits the amount of time a server remains responsible for its associated data sector. For the case of higher mobility, a spatial condition triggers when the server reaches a critical distance from its associated data sector. Electing a new data server DS_j is based on similar characteristics. Currently, DS_i elects the node closest to the center of the data sector with which it is associated. DS_i is able to determine this node solely based on position information stored in its local database. Finally, the database contents of DS_i are transferred to the newly elected server using a robust relocation protocol. Only those objects will be relocated whose observation time is sufficiently new. This lifetime strategy is suitable in our context, because position data is usually highly dynamic. Data server DS_j takes over as soon as it will have fully built its local database from DS_i's database contents.

C. Range Query Processing

Let R and α denote the query range and inaccuracy threshold according to Definition 3.6.

The processing of a probabilistic range query (*p*RQ) on a mobile node MN_i starts by determining a list S that contains identifiers of all data sectors that must be queried from a globally known list S_{global} of sector identifiers. For that, an overlap test is used to select all data sectors that may be

associated with data objects contained in R. *Partial p*RQs are then sent to every data sector in S. In Fig. 4, MN_i sends partial *p*RQs to two data sectors, because R overlaps with both S_r and S_{r+1}.

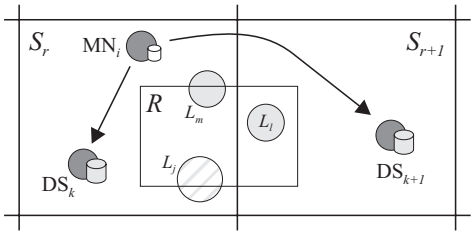


Fig. 4. Range query algorithm.

It is suitable to select a *proxy node* that is located near the query range to process the range query on behalf of the query client. A simple yet effective method is to select the mobile node nearest to the center of the query range. Using the concept of a proxy node, the number of hops that are required to send a partial query and, consequently, the required total communication costs for a query, decreases significantly.

According to the two-step mapping described previously, those data servers that are associated with a sector in S will receive a partial *p*RQ. On reception, a data server uses a local processing strategy for the partial query that returns objects in R according to the particular location semantics used. If the local strategy is correct, our data mapping scheme guarantees that the processing of the overall query is also correct. The result is then sent back to MN_i. Partial query results $\mathrm{Res}_{\mathrm{part}}$ received by MN_i are aggregated into the final result.

D. Nearest Neighbor Query Processing

Processing probabilistic k-nearest neighbor queries (*p*NQs) bears the difficulty that a bounding area that contains the k nearest objects is unknown beforehand. However, our data mapping approach allows for an efficient processing strategy in this situation. For that, we use a two-phase approach. During the *heuristic phase*, k objects o_j are preselected to determine the maximum range in which additional objects must be considered to compute the final query result. The selection strategy is realized by using partial *p*NQs executed in a sequence that allows us to find k candidate objects quickly and whose distance to the real k nearest neighbors is small. This step is essential to minimize additional communication costs required to complete the query. During the *aggregation phase*, all data servers that may contain objects that are nearer to p_{NN} than any of the objects determined during the heuristic phase are queried by means of a partial *p*NQ. The final result is then aggregated by using the *nearer* relation in Definition 3.9 and 3.10. In the following, MN_i is the proxy node processing the query on behalf of the client and p_{NN} denotes the query reference location.

The processing of a *p*NQ begins in the PROCESS-PNQ procedure. Argument k is the query parameter. To limit the search range of a *p*NQ in case object density is low or for large values of k, we use a *maximum search radius*, denoted by r. It may be specified by the query client and upper bound by the algorithm to avoid the dissemination of partial queries across the whole network.

```
PROCESS-PNQ(k, p_NN, α, r)
1  Res ← ∅
2  S ← S_global.ORDER-BY-DISTANCE-TO(p_NN)
3  SEND-PARTIAL-PNQ(k, p_NN, α, S.REMOVE(0))
4  SET-HEURISTIC-TIMEOUT(t_heu)
```

After initializing the final result Res to the empty set, the algorithm creates a copy of the global sector list that contains all sectors ordered by their distance to p_{NN} (line 2). The heuristic phase starts by sending a partial *p*NQ to the first sector in that list (line 3), which is the data server associated with data sector S_r where $p_{\mathrm{NN}} \in S_r$. Note that this sector must be queried in any case, because it may always contain an object nearer to objects in any other data sector. Fig. 5 illustrates the sending of the partial query during the heuristic phase.

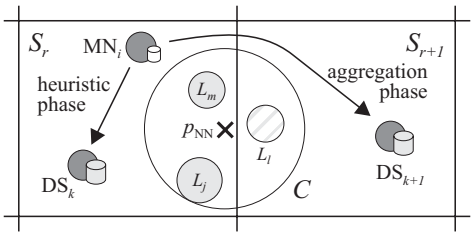


Fig. 5. Nearest neighbor query algorithm.

Similar to the case of a range query, a partial *p*NQ is processed locally at DS_k, by considering only data objects whose location area overlaps S_r. The implementation of the local processing strategy again depends on the query semantics used, making our algorithm independent of the semantics used.

Any partial result received by MN_i is handled in the RECEIVE-PARTIAL-PNQ-REPLY procedure:

```
RECEIVE-PARTIAL-PNQ-REPLY(Res_part)
 1  update final with partial query result
 2  for i ← 0 to Res_part.SIZE()
 3     do if not Res.CONTAINS(Res_part[i].id)
 4           then if Res.SIZE() < k
 5                   then Res.ADD(Res_part.REMOVE(i))
 6                   else if ∃j : Res_part[i].loc <
 7                              Res[j].loc | p_NN ∧
 8                              ∀k : Res[k] ≤ Res[j] | p_NN
 9                              then Res[j].REPLACE(
10                                     Res_part.REMOVE(i))
11
12  if phase = heuristic
13     then if Res.SIZE = k
14             then AGGREGATE-PNQ(k, p_NN, a, r)
15             else if DISTANCE(S[0], p_NN) < r
16                     then SEND-PARTIAL-PNQ(
17                            k, p_NN, a, S.REMOVE(0))
18                     else FINISH-PNQ
19     else if phase = aggregation
```

```
20          then S.GET(Res_part.s). status = received
21               terminate aggregation if known that
22               all partial queries returned
23               if ∀s ∈ S : s. status = received
24                  then FINISH-PNQ
```

For both the heuristic and aggregation phase, the incoming partial query result is merged into the final result (line 2-10). For this, every object o' in $\mathrm{Res}_{\mathrm{part}}$ is tested for containment in Res (line 3). Only if a copy o does not already exist in Res and Res contains less than k objects, o' is inserted (line 5). If Res already contains k objects, o' replaces o in Res only if o' is nearer to p_{NN} than o and if o is the object furthest away from p_{NN} from all other objects in Res (line 9-10).

The heuristic phase continues by testing if k objects were found (line 13). If so, the aggregation phase begins (line 14), otherwise, it is checked whether there exist more data sectors that may contain data objects whose distance to p_{NN} is less than the maximum search radius r (line 15). If so, MN_i queries other sectors S'_r with increasing distance to p_{NN} (line 16-17), otherwise, the query terminates and returns less than k objects (line 18).

Based on the result Res generated in the heuristic phase, node MN_i determines those sectors that must additionally be queried to complete the final result. For that, the smallest circle C is constructed that contains the location areas of all determined objects and whose center is p_{NN}. This is shown in the AGGREGATE-PNQ procedure (line 2-5). In Fig. 5 two objects with location areas L_j, L_m were retrieved from DS_k in the heuristic phase for a 2-nearest neighbor query. In the example, C contains another object's location area L_l that was not found during the heuristic phase. All sectors that were already queried in the heuristic phase are left out in the aggregation phase (line 8-10). For each remaining sector, we send partial *p*NQs (line 13-14). In Fig. 5, only Sector S_{r+1} is queried in the aggregation phase to retrieve the object whose location area is L_l.

```
AGGREGATE-PNQ()
 1  compute circle C inside of which we must further look
 2  C.r_agg ← 0
 3  for each o ∈ Res
 4     do d_max ← MAX_{p∈o.loc}(DISTANCE(p, p_NN))
 5        C.r_agg ← MAX(C.r_agg, DISTANCE(d_max, p_NN))
 6
 7  prune list of sectors that still overlap with C
 8  for each s ∈ S
 9     do if not OVERLAPS(s, C)
10           then S.REMOVE(s)
11
12  send partial queries
13  for each s ∈ S
14     do SEND-PARTIAL-PNQ(k, p_NN, a, s)
15
16  SET-AGGREGATION-TIMEOUT(t_agg)
```

Each result returned during the aggregation phase is merged at MN_i into the final result Res and each sector queried during the aggregation phase is annotated with the *received* state (line 20 in the RECEIVE-PARTIAL-PNQ-REPLY procedure). This allows us to determine if all partial results were collected (line 23), and the aggregation phase may terminate before the aggregation timeout occurs. Upon completion of the aggregation phase, the query result is returned to the query client.

VI. EVALUATION

To validate the performance of our algorithms we have conducted a set of experiments using the 802.11 implementation of the network simulator ns-2. We simulate 80 nodes and 160 objects located within a service area of 400 × 400 m. Both mobile nodes and perceivable objects move using the random waypoint mobility model with a fixed speed of 1.5 m/s and a pause time of 30 seconds. To analyze the impact of the data mapping approach discussed in Sec. V-A, we assume that a mobile node is able to determine the location of a perceivable object with an accuracy of 10 m. Table II summarizes the parameters that impact algorithm performance.

System Parameter	Default Value
simulation time	360 s
medium bandwidth	11 Mbit/s
service area	400 m × 400 m
number of sectors	4 (100 m × 100 m)
number of nodes	80
node speed	1.5 m/s
transmission range	100 m
sensing range	25 m (≈ RFID)
position inaccuracy	10 m (≈ GPS)
number of objects	160
object speed	1.5 m/s
observation interval	3 s (+ discovery)
object lifetime	10 s (soft state)
query rate	$1/30\ n\ \mathrm{s}^{-1}$
query parameter k	3

TABLE II
SYSTEM PARAMETERS

We focus on the analysis of the storage and k-nearest neighbor query algorithm and concentrate on the impact of system parameters on these algorithms. Note that the detailed analysis of algorithm-specific properties, such as strategies of timeout selection and partial query scheduling are optimizations of our proposed algorithms and are not discussed in this paper. In the following, we discuss measurement results with respect to the performance metrics query accuracy and latency as well as query and storage cost.

The **query accuracy** metric describes the quality of the result returned by the k-nearest neighbor query. Because comparison to related work is impractical, we have performed comparative measurements with respect to the *model-based optimum result*. That result reflects the situation where a query is processed based on the most current data at query time that was observed by mobile nodes, which corresponds to the case of a single centralized storage. This is the optimum case that can be achieved in the MANET at all. Let $R_{\mathrm{mod}} = \{o_1, \ldots, o_s\}$ and $R = \{o_1, \ldots, o_t\}$ denote the model-based

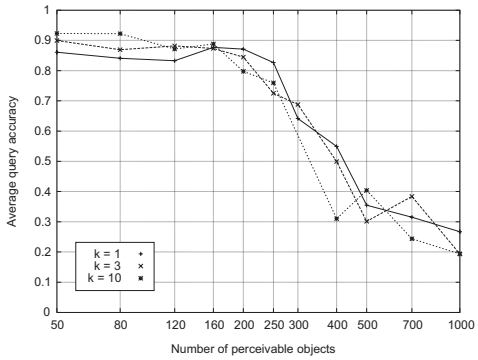


Fig. 6. Average query accuracy with respect to the model-based optimum as a function of the number of perceivable objects.

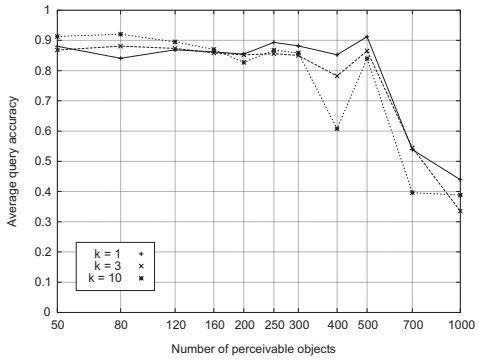


Fig. 7. Average query accuracy with respect to the model-based optimum as a function of the number of perceivable objects for point coordinates.

optimum result and the result returned by our query algorithm, respectively. The accuracy of a k-nearest neighbor query with respect to the model-based optimum result, denoted by A_{mod}, is defined as $A_{\mathrm{mod}} = |R \cap R_{\mathrm{mod}}| \ / \ |R_{\mathrm{mod}}|$.

Fig. 6 shows the results for the average query accuracy with respect to the model-based optimum result. We observe that the maximum achieved degree of accuracy is somewhat above 92 %. The deviation from the optimum is due to the soft state approach that we have chosen. For the selected object lifetime value of 10 seconds stale copies of object observations may exist on data servers in different sectors. Therefore, partial queries may be evaluated on inconsistent data. When the number of objects is beyond 250, we observe a sharp drop-off. The reason lies in the fact that in the model-based optimum, a single data server is assumed to receive object updates with optimum (zero) latency, whereas in the real system the propagation of each object update takes a certain amount of time. In addition, when the number of objects increases, the distance between them decreases and more and more objects have a similar distance to the position p_{NN}. Information not up-to-date has a greater influence on the ordering of objects. Together with the time-lag between object updates in the model-based optimum result and the simulated system, query accuracy inevitably decreases. These effects may only be compensated if the observation interval is increased. We observe that the dependence of the average query accuracy from different values of k may be neglected. Although we do not send redundant partial queries, the maximum accuracy is larger than 92%. This shows that our query algorithm performs well under the given system constraints and bears additional potential for optimizations. Fig. 7 shows the average query accuracy when point coordinates are used instead of inaccurate position information. The query accuracy for about 80 perceivable objects is virtually the same as for the respective results in Fig. 6. With increasing number of objects though, accuracy is greater for point coordinates. The reason is related to the fact that when location areas are used, the mapping introduced in Sec. V-A leads to a greater number of object copies stored at data servers in different data sectors than in the case of point coordinates. This way, partial queries

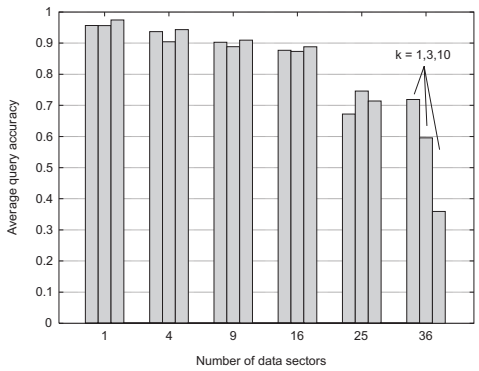


Fig. 8. Average query accuracy with respect to the model-based optimum as a function of the number of data sectors.

evaluate on copies of data objects that are inconsistent to a larger degree. We note that this fact can be addressed with an additional lightweight synchronization protocol that removes old copies updated by newer copies. We leave the details of this optimization to future work.

Fig. 8 shows the average query accuracy as a function of the number of data sectors. These results are significant as they may be used to calibrate the deployed algorithms, that is, to find the optimum sector granularity. We have displayed 3 sequences of values for the query parameter $k = 1, 3, 10$. The optimum with respect to query accuracy seems to occur for a single data sector. In that case, only a single data sector is queried, and a partial k-nearest neighbor query is able to provide the final query result already.

Fig. 9 shows that the mobility of nodes does not have a significant impact on query accuracy in the range of typical pedestrian's speeds. However, the achieved accuracy strongly depends on the particular routing protocols used. In our case, we used a modified version of GPSR, and whose beaconing interval was set to a suitable value to address the range of speeds encountered for pedestrians. We have observed virtually constant graphs for our other metrics as well in the given range of speeds.

We next define **query latency** as the time between the query was issued and the result was returned. Fig. 10 shows the average query latency as a function of the number of

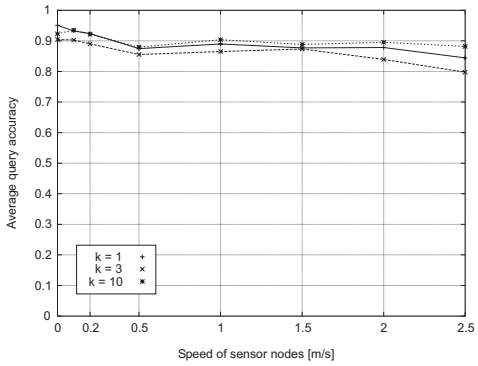


Fig. 9. Average query accuracy with respect to the model-based optimum as a function of the node speed.

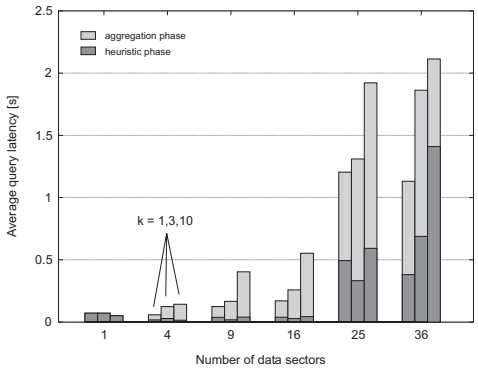


Fig. 10. Average query latency as a function of the number of data sectors.

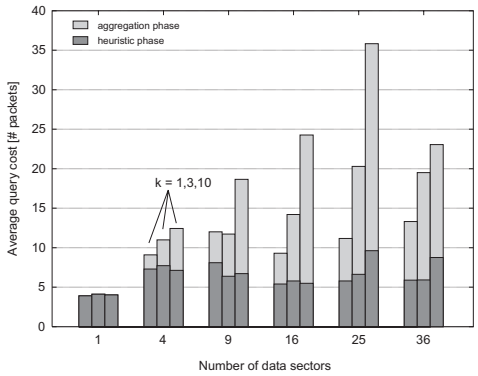


Fig. 11. Query costs as a function of the number of data sectors.

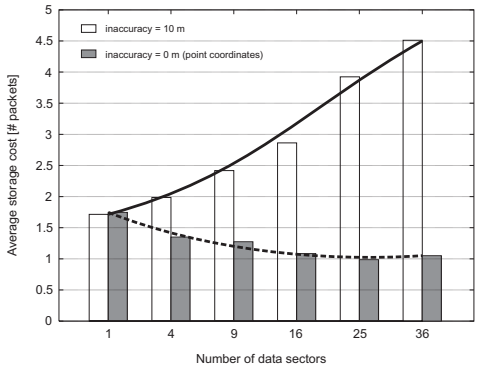


Fig. 12. Storage cost as a function of the number of data sectors.

data sectors. For small numbers of data sectors, the average latency is low. The reason lies in our strategy to process a k-nearest neighbor query. If only a small number of data sectors exist, the heuristic phase completes well before the timeout. With increasing number of sectors, the heuristic phase must query more data sectors to find candidate objects, which also increases with parameter k. Because we do not send redundant partial queries, some may fail due to unsuccessful routing attempts. The heuristic or aggregation phase cannot complete prior to a timeout in more and more cases, leading to a greater average query latency. The time needed for the aggregation phase decreases, because the longer the heuristic phase takes, the less sectors remain to be queried. Note that the aggregation phase times out after 2 seconds in our implementation.

For a single k-nearest neighbor query, the **query costs** are the sum of all packets sent to process the query. In Fig. 11, we depict the results for query costs as a function of the number of data sectors. We have split the columns so the costs required for partial queries and reply packets during the heuristic and aggregation phase can be separately observed. Up to 25 sectors, query costs increase because more and more partial queries are required to process a single query. If the number of sectors is increased further, costs begin to decrease again. This is related to the heuristic phase, which times out more and more frequently, because it occurs more often that at least one partial query fails. Our choice is to select small numbers of sectors for the deployed algorithms. Query costs increase with larger values of k because more partial queries are required as the circle C computed as described in Sec. V-D increases and covers more data sectors.

We quantify **storage cost** by the average number of packets sent per object update from observers to database nodes. Fig. 12 shows both the plots for point coordinates and location areas that are inaccurate to 10 m. In the case of location areas, storage costs increase because a finer sector granularity leads to more overlaps between a location area and data sectors, which results in more updates sent to data servers. If point coordinates are used, only a single overlap with a data sector is possible, independent of the sector granularity. Thus, with a growing number of data sectors, object updates require fewer hops, which results in a decrease of the average number of packets per update.

From the presented measurements we conclude that the optimum number of data sectors for the given service area (400 m $\times$ 400 m) is between one and four. If a larger service area is considered, a subdivision with the same granularity simply results in more data sectors. Because our storage and query processing algorithms operate locally, their scalability is guaranteed for *arbitrary* network sizes.

VII. Conclusion

In this paper, we have defined semantics for range and k-nearest neighbor queries based on the inaccurate nature

of object positions. We have introduced efficient algorithms for data storage and processing range and k-nearest neighbor queries in mobile ad hoc networks that consider inaccurate object positions. Our evaluation showed that the communication costs for object updates and partial queries are small thanks to our strategy of mapping data objects to data servers. Also, we showed that the algorithm for k-nearest neighbor queries yields good results even if partial queries are not sent redundantly. Future work includes the extension of the query semantics to other types of spatial queries, such as group nearest neighbor queries [18]. Concerning our algorithms, we are working on replication mechanisms to increase availability of data in sparse MANETs, both by replicating data associated with one data sector on additional servers in the same sector or on servers of different sectors. We are also investigating in detail the redundant sending of partial queries to increase accuracy and further minimize latency of k-nearest neighbor queries by using different timeout strategies. Our goal is to analyze the interaction between different algorithm parameters and their impact on the performance metrics to provide optimal storage and query performance.

REFERENCES

[1] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02)*, pages 44–53, Edmonton, Canada, July 2002. IEEE Computer Society.

[2] S. Bhattacharya. Randomized location service in mobile ad hoc networks. In *Proceedings of the 6th International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '03)*, pages 66–73, San Diego, California, USA, Sept. 2003. ACM Press.

[3] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 551–562, San Diego, California, USA, June 2003. ACM Press.

[4] R. Cheng and S. Prabhakar. Managing uncertainty in sensor databases. *ACM SIGMOD Record. Special Issue: Special Section on Sensor Network Technology and Sensor Data Managment*, 32(4):41–46, Dec. 2003.

[5] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proceedings of the 30th VLDB Conference*, pages 876–887, Toronto, Canada, Aug. 2004.

[6] A. Coman, M. A. Nascimento, and J. Sander. A framework for spatio-temporal query processing over wireless sensor networks. In *Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN 2004)*, Toronto, Canada, Aug. 2004.

[7] M. Demirbas and H. Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 32–39, Linköping, Sweden, Sept. 2003. IEEE Computer Society.

[8] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the 30th VLDB Conference*, pages 588–599, Toronto, Canada, Aug. 2004.

[9] P. Enge and P. Misra. Special issue on global positioning system. *Proceedings of the IEEE*, 87(1):3–15, Jan. 1999.

[10] H. Ferhatosmanoglu, I. Stanoi, D. Agrawal, and A. E. Abbadi. Constrained nearest neighbor queries. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2001)*, volume 2121 / 2001, pages 257–278, Redondo Beach, California, USA, Jan. 2001. Springer-Verlag.

[11] S. Goel, T. Imielinski, K. Ozbay, and B. Nath. Grassroots - a scalable and robust information architecture. Technical Report DCS-TR-523, Department of Computer Science, Rutgers University, Piscataway, New York, USA, June 2003.

[12] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A distributed index for features in sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 2003)*, pages 163–173, Anchorage, Alaska, USA, May 2003. IEEE Communications Society.

[13] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 243–254, Boston, Massachusetts, USA, Aug. 2000. ACM Press.

[14] W. Kieß, H. Füßler, J. Widmer, and M. Mauve. Hierarchical location service for mobile ad-hoc networks. *Mobile Computing and Communications Review*, 8(4):47–58, 2004.

[15] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 120–130, Boston, Massachusetts, USA, Aug. 2000. ACM Press.

[16] X. Li, Y.-J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, Los Angeles, California, USA, Nov. 2003. ACM Press.

[17] S. Nittel, M. Worboys, A. Stefanidis, I. Cruz, M. Egenhofer, D. Goldin, A. Howard, A. Labrinidis, S. Madden, and A. Voisard. Report from the first workshop in geo sensor networks. *ACM SIGMOD Record*, 33(1):141–144, Mar. 2004.

[18] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *Proceedings of the 20th International Conference on Data Engineering (ICDE 2004)*, pages 301–312, Boston, Massachusetts, USA, Mar. 2004. IEEE Computer Society.

[19] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop of Mobile Computing Systems and Applications*, pages 90–100, New Orleans, Louisiana, USA, Feb. 1999.

[20] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 78–87, Atlanta, Georgia, USA, Sept. 2002. ACM Press.

[21] K. Römer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004.

[22] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 71–79, San Jose, California, USA, May 1995. ACM Press.

[23] T. Schwarz, M. Iofcea, M. Grossmann, N. Hnle, D. Nicklas, and B. Mitschang. On efficiently processing nearest neighbor queries in a loosely coupled set of data sources. In *Proceedings of the 12th Annual ACM International Workshop on Geographic information Systems (GIS '04)*, pages 184–193, Washington, DC, USA, nov 2004. ACM Press.

[24] K. Seada and A. Helmy. Rendezvous regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*, page 218a, Santa Fe, New Mexico, USA, Apr. 2004. IEEE Computer Society.

[25] Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2001)*, pages 79–96, Redondo Beach, California, USA, 2001. Springer-Verlag.

[26] J. B. Tchakarov and N. H. Vaidya. Efficient content location in wireless ad hoc networks. In *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM 2004)*, pages 74–85, Berkeley, California, USA, Jan. 2004. IEEE Computer Society.

[27] J. Winter and W.-C. Lee. KPT: A dynamic knn query processing algorithm for location-aware sensor networks. In *Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN 2004)*, Toronto, Canada, Aug. 2004.

[28] Z. Xiang, S. Song, J. Chen, H. Wang, J. Huang, and X. Gao. A wireless LAN-based indoor positioning technology. *IBM Journal of Research and Development*, 48(5/6):617–626, Sept. 2004.

[29] Y. Xu and W.-C. Lee. Window query processing in highly dynamic geosensor networks: Issues and solutions. In *Proceedings of NSF Workshop GeoSensor Networks (GSN03)*, Portland, Maine, USA, Oct. 2003.