Toward the Optimal Itinerary-Based KNN Query Processing in Mobile Sensor Networks

Shan-Hung Wu, Kun-Ta Chuang, Chung-Min Chen, *Member*, *IEEE*, and Ming-Syan Chen, *Fellow*, *IEEE*

Abstract—The K-Nearest Neighbors (KNN) query has been of significant interest in many studies and has become one of the most important spatial queries in mobile sensor networks. Applications of KNN queries may include vehicle navigation, wildlife social discovery, and squad/platoon searching on the battlefields. Current approaches to KNN search in mobile sensor networks require a certain kind of indexing support. This index could be either a centralized spatial index or an in-network data structure that is distributed over the sensor nodes. Creation and maintenance of these index structures, to reflect the network dynamics due to sensor node mobility, may result in long query response time and low battery efficiency, thus limiting their practical use. In this paper, we propose a maintenance-free itinerary-based approach called Density-aware Itinerary KNN query processing (DIKNN). The DIKNN divides the search area into multiple cone-shape areas centered at the query point. It then performs a query dissemination and response collection itinerary in each of the cone-shape areas in parallel. The design of the DIKNN scheme takes into account several challenging issues such as the trade-off between degree of parallelism and network interference on query response time, and the dynamic adjustment of the search radius (in terms of number of hops) according to spatial irregularity or mobility of sensor nodes. To optimize the performance of DIKNN, a detailed analytical model is derived that automatically determines the most suitable degree of parallelism under various network conditions. This model is validated by extensive simulations. The simulation results show that DIKNN yields substantially better performance and scalability over previous work, both as k increases and as the sensor node mobility increases. It outperforms the second runner with up to a 50 percent saving in energy consumption and up to a 40 percent reduction in query response time, while rendering the same level of query result accuracy.

Index Terms—Indexing methods, query processing, distributed databases, sensor networks, mobile environments, wireless communication.

1 Introduction

Over the past years, mobile sensor networks have received a significant amount of attention as they support a wide range of applications, e.g., the Intelligent Transportation Systems (ITS) [1], wildlife conservation systems [2], and battlefield surveillance systems [3]. Sensor nodes move and are periodically queried by an external source for summaries and statistical information about the underlying physical process. The K-Nearest Neighbors (KNN) query, which invokes finding for KNN around a query point q, has been of significant interest in many studies and become one of the most important spatial queries in mobile sensor networks [4], [5], [6], [7]. The applications of KNN queries may include vehicle navigation, wildlife social discovery, and squad/platoon searching on the battlefields. KNN queries may also be applied to the emergency situations such as tracing the insurgent of a traffic accident, discovering the impact of a forest fire, and seeking for the nearby survivals around a battlefield stronghold, to name a few.

The problem of efficient KNN search in a spatial or multidimensional database is a major research topic in the literature [5], [8], [9], [10], [11], [12]. Traditional KNN query processing techniques assume that location data are available in a centralized database and focus on improving the index performance [13], [5], [14], [15]. In new applications where data sources are geographically spread (e.g., sensor networks [16], [2], wireless ad hoc networks [17], ITS [1], etc.), pulling data from a large number of data sources (e.g., sensor nodes, laptops, vehicles, etc.) is generally infeasible due to high energy consumption, high communication cost, or long latency [18], [19]. Recently, a number of studies have explored "in-network" KNN query processing techniques for sensor networks [20], [9], [10], [11], [12]. These techniques rely on certain in-network infrastructure—index or data structures (e.g., clustered indices or spanning trees) distributed among the sensor nodes—to select KNN candidates, propagate queries, and aggregate the result.

Although these in-network approaches avoid the overhead of periodical data gathering from a large number of sources, they still face several challenges if they are to be deployed in large-scale mobile sensor networks. First, the distributed indexing structures may become too costly to maintain when the number of nodes increases, due to the communication overhead among the nodes. Second, certain

- *S.-H. Wu is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, and Telcordia Applied Research Center, Taipei, Taiwan, ROC. E-mail: brandonwu@research.telcordia.com.*
- *K.-T. Chuang is Synopsys Inc., Taipei, Taiwan, ROC. E-mail: doug@arbor.ee.ntu.edu.tw.*
- *C.-M. Chen is with Telecordia Applied Research Center, Taipei, Taiwan, ROC. E-mail: chungmin@research.telecordia.com.*
- *M.-S. Chen is with the Research Center of Information Technology Innovation, Academia Sinica, Taipei, Taiwan, and the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC. E-mail: mschen@cc.ee.ntu.edu.tw.*

Manuscript received 12 July 2007; revised 30 Dec. 2007; accepted 8 Feb. 2008; published online 18 Apr. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-07-0358.
Digital Object Identifier no. 10.1109/TKDE.2008.80.

techniques [20], [9] require some nodes to act as supernodes, for example, as clusterheads or data aggregation points. These supernodes may easily turn into a bottleneck of the system. Furthermore, current in-network-based KNN techniques [20], [9], [10], [11], [12] have all assumed a fixed network topology where sensor nodes are stationary and never fail. This assumption makes them inappropriate for a sensor environment where sensor nodes are mobile and packet loss is the norm rather than an exception [18], [21], [22], as the maintenance overhead of the in-network indexing structure could be considerable.

In this paper, we propose a Density-aware Itinerary KNN query processing (DIKNN) for mobile sensor networks that does not rely on any sort of in-network indexing structure. The key idea of DIKNN is to let sensor nodes collect partial results and propagate the query along a well-devised *conceptual* itinerary structure. No physical maintenance of this itinerary structure is required. The DIKNN divides (conceptually) a circular search boundary centered at the query point q into multiple cone-shape areas. It then performs concurrent itinerary traversal, based on a predefined itinerary structure, to the nodes in each of these areas. The traversal length is adjusted dynamically according to the node distribution information it collects as the traversal proceeds.

To the best of our knowledge, DIKNN is the first KNN processing technique for mobile sensor networks that does not rely on any in-network indexing structure support: No constant maintenance or fixed data aggregation point is needed. Because of this, DIKNN is able to avoid potential bottleneck and survive rapid changes of network topology. It also reduces query response time (or latency for short) by combining data collection with query propagation in a well-devised itinerary.

Several challenging issues arise in the design of DIKNN, such as the trade-off between concurrent traversal and network interference, the estimated search radius, and the design of an efficient itinerary. We investigate and present solutions to each of these issues. To ensure the optimal performance of DIKNN, we derive a detailed analytical model that automatically determines the most suitable degree of concurrency under various network conditions. This model is validated by extensive simulations. The simulation results also show that DIKNN yields substantially better performance and scalability over previous work, both as k increases and as the sensor node mobility increases. In particular, it outperforms KPT [11], [12] with up to a 50 percent saving in energy consumption and up to a 40 percent reduction in query response time, while rendering the same level of query result accuracy.

The rest of the paper is organized as follows: Section 2 reviews the previous studies on KNN query processing and related work to DIKNN. Section 3 presents the design of DIKNN. An analytic model is derived for parallel itinerary traversal. In Section 4, algorithms determining the KNN boundary are introduced. We also discuss how DIKNN may interact with network environments. Section 5 reports our performance evaluation based on simulation results. Section 6 concludes the paper. An analytic model is derived in Section 5 for parallel itinerary traversal. Section 6 reports our performance evaluation based on simulation results. Finally, Section 7 concludes the paper.

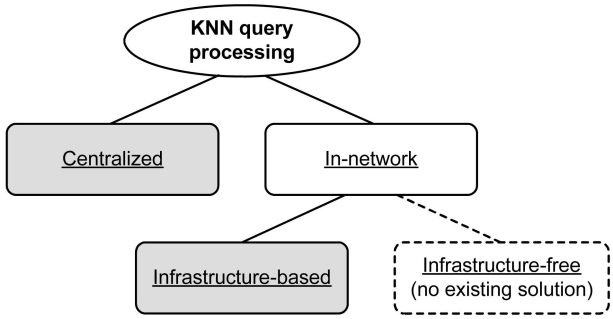

Fig. 1. Categorization of previous studies.

2 Related Work

Previous work on KNN or window (range) query processing in sensor networks can be generally classified into two categories: the *centralized* and *in-network* approach, as shown in Fig. 1. The centralized approach performs the queries in a centralized database containing locations of all the sensor nodes [13], [5], [14], [15]. These location data are usually maintained in an R-tree variant index modified to handle mobile objects. In contrast, the in-network approach does not rely on a centralized index, instead, it propagates the query directly among the sensor nodes in the network and collects relevant data to form the final result [23], [20], [9], [10], [11], [12]. This approach is favored when maintenance of a centralized index is expensive or may impose high energy consumption on the sensor nodes. And, it happens, for example, when the locations of the nodes change frequently or when there is no direct wireless link between the nodes and the centralized index. The centralized indices may lead to substantial message routing/relay overhead.

The in-network approach can be further divided into two subcategories: those relying on a certain sort of in-network infrastructure and those that are infrastructure free. The term "infrastructure" refers to a data structure distributed among the sensor nodes that is created, either once on-the-fly or to be updated constantly, to support the query processing. The works in [20], [9], [10], [11], and [12] are representatives of this kind targeting at KNN queries for fixed sensor networks. Maintenance of the in-network data structure could become costly prohibitive, if not infeasible, when the sensor nodes become mobile. To eliminate this problem, an infrastructure-free method was proposed in [19], but it applies to window queries only. Unlike range queries, the search boundary of a KNN query is not predetermined by the user. It is challenging to estimate a proper KNN search boundary that balances the trade-off between power efficiency and query result accuracy. Recently, a number of works have addressed continuous queries using in-network techniques [23], [5], [15]. These methods are good for constant monitoring of queries of long-standing interest but do not suit well for on-demand queries (one time only) that is the focus of our work.

We will briefly describe the Peer-tree [20], DSI [9], and KPT [11], [12] as they are most relevant to our work. They will also be used as a comparison point in our performance evaluation. The Peer-tree [20] and DSI [9] decentralize the index structures (e.g., R-tree [13]) to distributed environ-

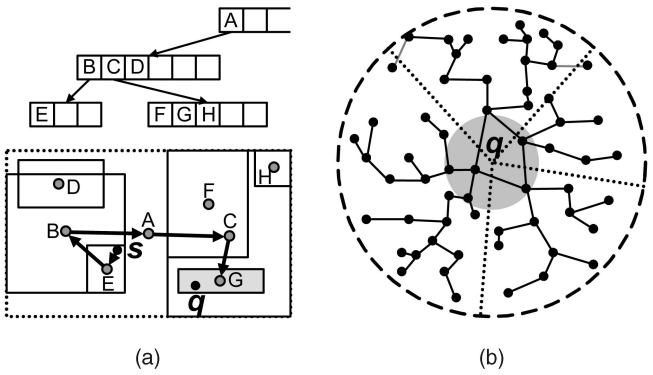

Fig. 2. Related work: The decentralized R-tree approaches and KPT.

ments. As shown in Fig. 2a, a network is partitioned into a hierarchy of Minimum Bounding Rectangles (MBRs). Each MBR covers a geographical region, including all sensor nodes located inside. An MBR in the higher hierarchy (say, region A in Fig. 2a) covers all the regions of the sub-MBRs in the child hierarchy (regions B, C, and D in Fig. 2a). One specific node is designated as a clusterhead (i.e., distributed index) in each MBR. A clusterhead knows locations and identities (IDs) of all the other nodes within the MBR. It also knows locations and IDs of its parent and child clusterheads. To handle an NN query, the source node s routes the query message to its clusterhead (node E in Fig. 2a). Upon receiving the message, the clusterhead forwards it upward in the hierarchy until that the query point q is covered by the MBR of a clusterhead (in this case, node A in Fig. 2a). The clusterhead then forwards the message downward in the hierarchy looking for a child clusterhead (node G in Fig. 2a) that contains q with minimal MBR. After that, the location of NN of q can be determined, and the NN is informed of the query message by unicast. Supporting of KNN queries is more complicated that needs multiple clusterheads to find and to propagate the query message in different MBRs. Since every query message goes through the clusterheads, the major problem of these approaches is that index nodes become system bottlenecks easily. Such approaches are vulnerable to index failure. In addition, there are many unnecessary hops from s to the KNN nodes because each query message is routed along the hierarchy of clusterheads, as depicted by the arrows in Fig. 2a. Such overhead becomes significant in the large-scale sensor networks, where the distance between clusterheads is long.

To address such issues, the KPT [11], [12] is proposed to handle the KNN query without fixed indexing. This work assumes each sensor node is location aware. After a query is issued from s, it is routed to the sensor node, named *home node*, closest to q. To avoid flooding the entire network, a conservative boundary containing at least k candidates is estimated by the home node. Multiple trees rooted at the home node are then constructed to propagate queries and to aggregate data, as shown in Fig. 2b. Upon aggregating data at the home node, it determines correct KNNs (by sorting locations) and transmits their query responses back to the source s. KPT assumes an optimal network condition where each node is stationary. It encounters two serious drawbacks in presence of mobility. First, constructing or maintaining the trees while sensor nodes are moving incurs considerable overhead. Partially collected data may be forwarded again and again between new and old tree nodes. Second, the conservative (large) boundary grows quadratically as k increases, which leads to high energy consumption and long latency. Although such a boundary is expected to cover at least k nodes in the worst case, sensor nodes may either move in or move out the boundary during tree construction and data aggregation. KPT returns poor query result accuracy.

In light of the above problems, we propose DIKNN, which to our best knowledge, is the first infrastructure-free KNN search method for mobile sensor networks. Nevertheless, the concept of itinerary traversal is inspired by a number of research efforts in unicast routing [24], data fusion [25], network surveillance [26], and window query processing [19]. Our main contribution lies on the origination of a sophisticated itinerary structure and its detailed analysis (taking into account the important factors such as itinerary width, data collection scheme, adaptive search boundary estimation, forwarding heuristics, etc.) that offers both high efficiency and high flexibility in parallel query dissemination and processing.

3 Design of DIKNN

In this section, we first give a formal definition of our problem. Then, we describe the three execution phases in DIKNN. Design and analysis of the itinerary/subitinerary traversal are detailed thereafter.

3.1 Definitions and Network Model

In this paper, we focus on snapshot queries, which expect to obtain the query result only once during their lifetimes. The KNN problem is defined as follows:

Definition 3.1 (KNN problem). *Given a set of sensor nodes S, a geographical location q (i.e., query point), and valid time T find a subset S' of S with k nodes (i.e., $S' \subseteq S, |S'| = k$) such that at time T, $\forall n_1 \in S', n_2 \in S - S' : DIST(n_1, q) \leq DIST(n_2, q)$, where $DIST$ denotes the euclidean distance function.*

Ideally, we would like to obtain the exact result set S' comprising the KNN of q at the given time T. However, due to node mobility and efficiency considerations [18], [19], we may accept an approximate result set. Query result accuracy is measured by the percentage ratio of the correct KNNs (at valid time T) returned. Depending on different application needs, the valid time T can be defined either as the time the query is issued (snapshot results are better) or the time the result set is received (newer results are better). In our evaluation, measurements of accuracy according to these two types of valid time are called *preaccuracy* and *postaccuracy*, respectively.

We assume the network is in ad hoc mode so nodes far away from their mutual coverage communicate with each other through multihops. We assume that all sensor nodes can store data locally and answer the queries individually. In addition, the moving speed and directions of sensor nodes are arbitrary. Each sensor node is aware of its geolocation. Beacons with locations and IDs are periodically broadcasted. Every sensor node also maintains a table including IDs and locations of neighbor nodes falling

within its radio range, r. Note this network scenario has been assumed in [19] and complies with IEEE standard 802.15.4 [16], the *Low Rate Wireless Personal Area Network* (LR-WPAN), to achieve maximum compatibility.

Note that Definition 3.1 does not specify the types of information returned with S'. If the node order in S' is desired, each node may return its geolocation, and the user receiving S' can simply determine the order of nodes by comparing their distances to the query point.

3.2 Execution Phases

The execution of DIKNN consists of three phases:

1. *Routing phase.* A query message Q is geographically routed from the sink node s to the nearest neighbor (i.e., the home node n_p, where p denotes the number of hops along the routing path) around the query point q. Information of the sensor network is gathered along with the routing procedure without the aid of any infrastructure.
2. *KNN boundary estimation phase.* Upon receiving Q and the collected information from the previous phase, the home node estimates a searching boundary, named KNN boundary, with radius R by using an efficient (specifically, linear time) KNNB algorithm. The estimated boundary is not fixed and will be dynamically adjusted (by the other nodes) as long as additional information is available in the next phase.
3. *Query dissemination phase.* The home node disseminates the query message to all sensor nodes inside the KNN boundary. Dissemination follows a concurrent itinerary structure, and query responses are aggregated along with multiple itineraries. At the end of dissemination, the aggregated query responses along each itinerary are bundled and routed back to the sink directly without the home node's involvement.

Next, we explore the main phase of DIKNN, the query dissemination phase, by assuming that the KNN boundary is given. The routing and KNN boundary estimation phases will be visited later.

3.3 Itinerary-Based Query Dissemination

Once the KNN boundary (and its radius R) is determined, the home node n_p enters the query dissemination phase aiming to inform all the sensor nodes inside the boundary of the query message Q and to collect their responses. As the infrastructure-based technique leads to considerable overhead in dynamic environments, we turn to explore an infrastructure-free technique. One naive infrastructure-free solution is to flood the query within the boundary. Each node inside the boundary, upon receiving Q, routes its response back to s end-to-end and then broadcasts Q again. This approach, however, is extremely resource-consuming and has poor scalability because of the excessive number of independent routing paths from sensor nodes to s [19]. In addition, serious degrees of collision and hidden terminal problem may also occur during the wireless transmission. To address these issues, DIKNN adopts an itinerary-based dissemination technique, which provides robust and effective query processing under transient network topologies.

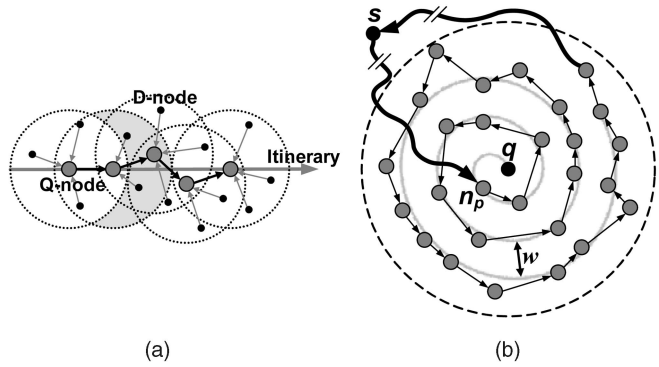

Fig. 3. Itinerary-based query dissemination.

The concept of itinerary query dissemination [24], [25], [26], [19] can be best understood by the illustration in Fig. 3a. A set of *Query nodes* (Q-nodes) in the KNN boundary are chosen for query dissemination. Upon receiving a query, a Q-node broadcasts a *probe* message that includes information about Q, R, and the itinerary (e.g., itinerary width and the number of sectors, which will be discussed later). When hearing the probe message, the neighbor nodes that are qualified to reply the query, called *Data nodes* (D-nodes), report their query response back to the Q-node. After obtaining the data from all D-nodes as well as the partial result received from the previous Q-node, the current Q-node selects the next Q-node based on the itinerary information and forwards this new partial query result to the selected next Q-node. This procedure repeats until the query traverses the entire KNN boundary along a predefined (say, spiral) itinerary structure, as shown in Fig. 3b. Responses of all nodes held by the last Q-node are then returned back to the sink node s in a single message.

Primitives of itinerary-based solution. Some useful primitives have been proposed in [19] to ensure correctness of itinerary execution. At first, the itinerary width w, as shown in Fig. 3b, specifies the minimum distance between different segments of an itinerary. Obviously, a small w results in denser itinerary traversal which ensures the KNN boundary to be fully covered by the traversal. On the other hand, the small w incurs unnecessary transmission and long latency because of the increased itinerary length.

Theorem 3.1. *Let r denote the transmission range of the sensor nodes. In order to guarantee full coverage of a KNN boundary, the itinerary width w must be less than $\sqrt{3}r/2$.*

It can be shown [19] that letting $w = \sqrt{3}r/2$ yields full coverage while ensuring the minimal itinerary length, a good balance on query accuracy and energy efficiency. Second, data collection from multiple D-nodes needs to be better scheduled to avoid collisions and delays. The *contention-based data collection scheme*[1] can be utilized to prevent serious contention and to sustain network dy-

1. As suggested by our simulation result, the data collection scheme introduced in this paper combines both the *token-ring-based* and *contention-based* scheme to achieve higher performance. For a detailed discussion on these two schemes, please refer to [19].

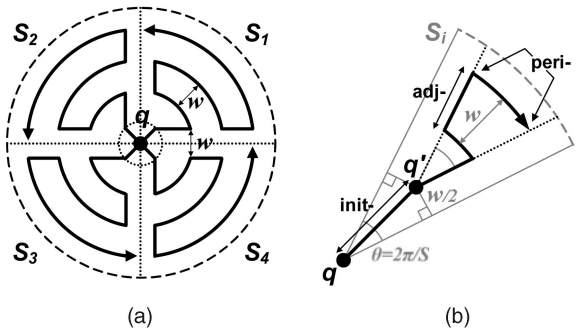

Fig. 4. Concurrent query dissemination.

namics. In this scheme, a reference line emanating form the current Q-node is included in the probe message. The probe message also contains a precedence list indicating the reply order of D-nodes. Upon receipt of the probe message, each D-node sets a timer with $timer = (\frac{\alpha}{2\pi})im$, where α is the angle formed by the specified reference line and the line connecting the current Q-node and the current D-node, i is the received precedence, and m is a time unit for the Q-node waiting for each D-node to report its data. A D-node does not respond to the Q-node until its timer expires. Discussions on the other issues such as fault tolerance and traveling in low-connectivity areas with *itinerary voids* (i.e., situations when a Q-node cannot find the next Q-node for query forwarding) can be found in [27] and [19].

In mobile environments, when the Q-node moves outside the current segment during the data collection phase, it may immediately forward the partial collected results to the node nearby its original location. The data collection can be resumed on the new Q-node with a new probe message targeting only those D-nodes that have not been reported yet. To avoid duplicated data, we let each D-node report its own ID along with the sensed data to the Q-node. By keeping the IDs of the nodes (including Q-nodes and D-nodes) collected so far in the partial query result, each Q-node is able to determine whether the coming data is duplicated or not. Note that since k is usually small, keeping these IDs may not result in too much overhead.

Clearly, the performance of dissemination solely depends on the structure of an itinerary, along which Q is propagated and responses are forwarded. Query latency can be significantly improved by considering the parallel dissemination. Nevertheless, concurrent dissemination may increase the likelihood of channel contention and collision at the data link and physical layers, causing degradation of network throughput. Parallelization should be exercised cautiously to avoid the overhead and should satisfy the following criteria. First, the number of routing paths leading back to the sink s, after dissemination, should be controllably small to prevent high energy consumption in large-scale sensor networks. Second, as concurrent itinerary traversals may incur channel interference in wireless transmission, the chance they meet should be as small as possible. Unfortunately, the only study [19] that mentions parallelization cannot scale well to a high concurrency level due to its simplified assumption upon the query range and itinerary structure.

Concurrent itinerary structures. To fulfill the above criteria, a KNN boundary is partitioned into multiple sectors, as shown in Fig. 4a. In each sector, the query is propagated along a subitinerary. The distance between subitineraries in adjacent sectors is w to ensure full coverage of the KNN boundary when $w \leq \sqrt{3}r/2$. Each subitinerary consists of three segments: the init-, adj-, and peri-segments, as illustrated in Fig. 4b. The init-segment is a portion of subitinerary that has a distance less than $w/2$ to either side of a sector's border. This segment is formed by a straight line to get rid of the interference as soon as possible. Specifically, let S be the number of sectors and l_{init} be the length of the init-segment. Then, we have $\sin(\theta/2) = \sin(\pi/S) = (w/2)/l_{init}$, which gives

$$l_{init} = \min\{w/(2\sin(\pi/S)), R\}. \quad (1)$$

Let q' denote the end of the init-segment. The peri-segments are portions of the subitinerary that together form perimeters of concentric circles centered at q'. Let l_{peri} be the total length of the peri-segments, then

$$l_{peri} = \sum_{i=1}^{\lfloor (R-l_{init})/w \rfloor} \frac{2\pi(iw)}{S}, \quad (2)$$

where $2\pi(iw)/S$ denotes the perimeter length of the ith concentric circle, and $\lfloor (R - l_{init})/w \rfloor$ denotes the number peri-segments. The adj-segments are portions of the subitinerary that are parallel to either side of the sector's border. It is clear that each adj-segment has the same length of w. The total length of the adj-segments l_{adj} therefore is equal to

$$l_{adj} = \lfloor (R - l_{init})/w \rfloor w. \quad (3)$$

Ideally, two subitineraries in adjacent sectors interfere with each other only at their init-segments. An important observation is that even if these two subitineraries are traversed in different speeds, extra interference can only occur at adj-segments, which are relatively short as compared to peri-segments. Such a cone-shape itinerary structure is highly adaptive to various degrees of parallelism. At an extreme, the shape of a subitinerary degenerates into a straight line if S is large enough. This allows the best efficiency when no interference can ever occur in the sensor network (e.g., when *Contention Free Period* (CFP) is exercised in the LR-WPAN).

4 KNN Boundary Estimation

It is a challenging issue to precisely estimate the KNN boundary without the aid of supernodes containing long-term monitored (and cached) information. This is because decisions must be made with very limited knowledge that can only be obtained from query propagation. DIKNN adopts a simple, yet effective, algorithm named KNNB, tailored for sensor nodes with limited ability.

4.1 Routing Phase

In the routing phase, a query Q is routed from sink node s to the nearest neighbor n_p (i.e., the home node) around the query point q, where p denotes the number of hops along the routing path. Any geographic face routing protocol [27], [28] is compatible with DIKNN. Ensuring correctness and efficiency of these routing protocols in the sensor network is

an orthogonal issue to DIKNN, which is studied in the literature [29], [30].

By utilizing the geographic face routing protocol, information collection is performed between hops. An additional list L is sent along with Q. On the ith $(1 \leq i < p)$ hop to the destination, the corresponding node (i.e., the sensor node triggering the ith hop) appends its own location loc_i and the number of newly encountered neighbors enc_i to L. To avoid duplicate information, enc_i can simply be counted by checking the number of neighbors having a distance larger than r from the corresponding node of the $(i-1)$th hop. Note that in comparison with the previous studies [23], [20], [9], such an information gathering technique consumes very few extra recourses since only nodes around the route is involved.

4.2 Linear KNNB Algorithm

Upon the receipt of the query message and the list L from the previous phase, n_q starts estimating the KNN boundary by determining its radius length R. As described previously, too large a boundary (as the one given by KPT [11]) incurs great energy consumption and long latency. In contrast, a small boundary loses the query accuracy. The determination of R must balance two conflicting factors: 1) increasing R to enclose correct KNN points as many as possible and 2) decreasing R to reduce the energy consumption.

In most of the previous work, sensor nodes are thought to be uniformly distributed in the network and sometimes even to form a grid. However, the recent investigation [31] argues that spatial irregularity happens in the majority of the cases. Regarding the limited energy and computing power on sensor nodes that may prohibit a complex analysis of the sensor distribution, for now, we content ourselves that KNNB adopts a weaker assumption: Sensor nodes are uniformly distributed *within the optimal KNN boundary* (i.e., the boundary containing exactly KNN). Of course, this assumption is blatantly violated when k is large. We will discuss a solution later to remit the bias of estimation.

Let $\widetilde{R}$ be the radius of the optimal KNN boundary and n_i be the corresponding node of the ith hop in the routing path that locates inside the optimal KNN boundary. We have $\overline{loc_i q} \leq \widetilde{R}$. Consider a circle centered at q with radius $\overline{loc_i q}$. From assumption of uniform distribution, we can estimate the number of nodes est_k locating inside the circle by using the equation:

$$D \approx \frac{est\_k}{\pi\left(\overline{loc_i q}\right)^2} \approx \frac{\sum_{j=i}^{p} L.enc_j}{Area\ covered\ from\ n_i\ to\ n_p}, \qquad (4)$$

where D denotes density of nodes $(nodes/m^2)$ within the optimal KNN boundary. Thus, we have

$$est\_k \approx \frac{\pi(\overline{loc_i q})^2 \sum_{j=i}^{p} L.enc_j}{Area\ covered\ from\ n_i\ to\ n_p},$$

while leaving the coverage area (the long-dotted line shown in Fig. 5b) along the routing path from n_i to n_p to be determined. Since correct evaluation of this area is too complicated to be executed on a sensor node given its limited computing power, we need an easy approximation.

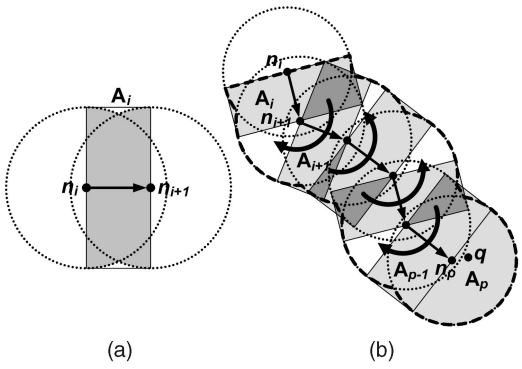

Fig. 5. KNN boundary estimation.

Observing that a sensor node will always find the next hop within its radio coverage, thus advance of a single hop must be less than r. As depicted in Fig. 5a, the shaded coverage area between two sensor nodes of successive hops is large enough to be approximated by using a rectangle A_i. Putting rectangles $A_i, A_{i+1}, \ldots, A_{p-1}$ from n_i to n_{p-1} together, the symmetric property shown by the arrows in Fig. 5b helps the doubly counted regions (heavily shaded in Fig. 5b) to complement the opposite areas against the routing path. Thus, summing these rectangles with an additional semi-circular region A_p yields an easily calculated but close approximation to the coverage area from n_i to n_p.

From the above, we now have est_k. Back to our problem of estimating the KNN boundary, applying (4), we have

$$D = \frac{k}{\pi\widetilde{R}^2} \approx \frac{est\_k}{\pi\left(\overline{loc_i q}\right)^2}.$$

$\widetilde{R}$ is unknown. As est_k approaches k, we have $R = \overline{loc_i q} \approx \widetilde{R}$. Algorithm 1 below shows the detailed steps of KNNB.

Algorithm 1. The KNNB algorithm: $\mathrm{KNNB}(L, q, r, k)$.
Require: information list L gathered from the first phase of DIKNN, the query point q, radius r of a sensor node, and number k of nearest neighbors to be found.
Ensure: returning radius length R of the KNN boundary.
1: $i = L.length - 1$;
2: $neighbors = L.enc_i$;
3: $approx\_area = \pi r^2/2$;
4: **while** $i \geq 0$ **do**
5: $d = \mathrm{DIST}(L.loc_i, q)$;
6: $est\_k = \pi d^2(neighbors/approx\_area)$;
7: **if** $est\_k \geq k$ **then**
8: return d;
9: **end if**
10: $neighbors+ = L.enc_{i-1}$;
11: $approx\_area+ = \mathrm{APPROX}(L.loc_i, L.loc_{i-1})$;
12: $i = i - 1$;
13: **end while**

The list L is indexed from one. In lines 4-13, KNNB iteratively approaches est_k to k by examining L from the tail. The function $DIST(a, b)$ at line 5 simply returns the distance between the given two positions a and b, and the function $APPROX(a, b)$ at line 11 gives the approximative rectangle by returning $r \cdot DIST(a, b)$. The complexity of

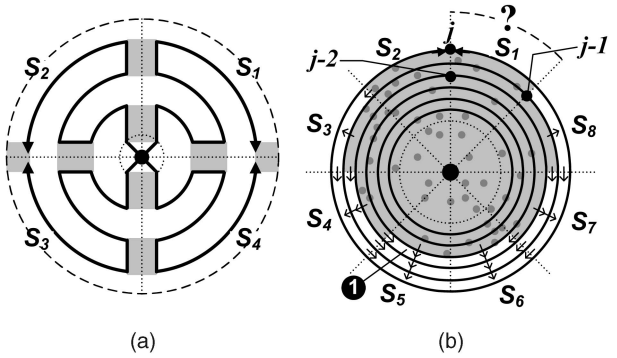

Fig. 6. Dynamic search boundary adjustment according to the spatial irregularity.

KNNB is $O(n)$, where n is the number of hops in the routing path.

Note that the experimental results reveal that the radius lengths returned by KNNB are generally $1/\sqrt{k\pi}$ of those returned by KPT [11], [12] given the same level of accuracy. An example finding for five nearest neighbors $(k = 5)$ is depicted in Fig. 2b. As we can see, the shaded KNN boundary determined by KNNB is much smaller than the long-dotted one estimated by KPT.

4.3 Interaction with Environments

Next, we discuss some mechanisms adopted by DIKNN to face spatial irregularity and mobility in real-world sensor environments.

Spatial irregularity. The KNNB algorithm estimates the KNN boundary by assuming that sensor nodes are uniformly distributed around the query point q. This assumption is valid for a small region due to the spatial locality; however, when k is large, the sensor nodes tend to irregularly spread, and their spatial density becomes unpredictable [31]. This effect, called *spatial irregularity*, may degrade the query accuracy. To handle this problem, we let the Q-nodes in different sectors adjust their own R during dissemination. Specifically, we inverse the direction of peri-segments in every interseptal sector. In such a configuration, the face-to-face adj-segments of different subitineraries together form *rendezvous segments* (as shown by the shaded area in Fig. 6a), in which two Q-nodes from adjacent subitineraries can, with a little cost of latency, meet with each other and exchange the latest statistics (e.g., total number of nodes explored so far). By repeating this procedure, the jth rendezvous segment in a subitinerary can obtain information from $2, 4, \ldots, \min\{2j, S\}$ nearby sectors at the jth, $(j-1)$th, $\ldots$, 1st level of the peri-segments, respectively (as depicted by the shaded area in Fig. 6b). Each sector, say, S_1 in Fig. 6b, can then infer how many nodes around q are explored so far (totally) and dynamically adjust R to stop or to continue the dissemination. With rendezvous, itinerary traversals can stop immediately if KNN are discovered before reaching the perimeter of the KNN boundary or continue if fewer nodes are found. Note this technique may result in shrinking of a KNN boundary, if there are nodes moving into the boundary after the KNN boundary estimation phase.

When S_1 is doing the inference, a simple bilinear interpolation can be used to complement the not-yet-exchanged information from the other sectors. For example, S_1 may estimate the density of an unseen segment (Fig. 6b(1)) by taking the advantages of spatial locality along the *radiate* and *concentric lines* centered at the query point. The bilinear interpolation is done by considering the known densities of the adjacent segments along these two lines, as shown by the small arrows in Fig. 6b.

Mobility concern. The mobility of sensor nodes degrades query accuracy because nodes may move in or move out the KNN boundary during dissemination. For applications to which discovering correct KNNs are the most important concern, support of flexible expansion of R to include more candidates is critical. One naive approach is to modify the KNNB algorithm so that $R' = c \cdot R$ is returned, where R' denotes the adjusted radius of the KNN boundary and c, $c > 1$, denotes a constant. Obviously, with the larger c, we may guarantee more correct KNNs with R'; however, more energy is consumed as well. This makes it very difficult for an application to determine a good value of c. In DIKNN, we address this issue in the query dissemination phase, where the last Q-node is obligated to determine how much farther a subitinerary should continue. Specifically, each application is allowed to specify an attribute, named *assurance gain* g, $0 \leq g \leq 1$, at the time when a KNN query is issued. By acquiring the moving speed of each sensor node along with data collection, each subitinerary can maintain a record μ specifying the fastest moving speed traced so far. Upon receiving this record, the last Q-node is able to measure the maximum shift of sensor nodes by $(t_s - t_e)\mu$, where t_s and t_e denote time stamps of the moment the node n_p receives the query and the query dissemination ends, respectively. Thus, an appropriate expansion of R can be obtained by $R' = R + g(t_e - t_s)\mu$.

To deal with the interaction between the mobility and spatial irregularity, it is reasonable to first adjust the KNN boundary according to the updated density information during the itinerary traversal, then make a final expansion (to guarantee the worst case performance) by considering the uncertainty introduced by the node mobility.

Fault tolerance. Wireless sensor networks encompass various types of packet losses, which may occur due to mobility of sensor nodes, environmental interference, low signal-to-noise ratios (SNR), contention of the channel access, etc. Xu et al. [19] have proposed a mechanism to deal with the packet losses. Upon receiving the probe message form a Q-node, each D-node, besides replying with its own response, relays all the responses it hears from the neighbor D-nodes to the Q-node as well. This approach causes redundant transmission and induces extra contention, as well as energy overhead. In this paper, we propose a new fault-tolerant data collection scheme that requires much less extra energy consumption and does not result in duplicate responses. Benefiting from the location awareness, a D-node receiving a probe message knows the relative reply precedence with its neighbor D-nodes. It monitors (i.e., caches) the response of the D-node having the precedence immediately before itself. If no ACK (from the Q-node to monitored D-node) is heard, the monitoring

TABLE 1
Notation Used in our Analysis

Param.	Description
R	radius length of KNN boundary (m)
D	node density within KNN boundary ($nodes/m^2$)
w	itinerary width (m)
m	time unit for data collection (s)
r	radio range of sensor node (m)
ρ	channel bit rate of senor node (bps)
d_{prob}	size of the probe message ($bits$)
d	size of query responses ($bits$)

All parameters can be obtained from the KNN boundary estimation phase.

D-node sends cached data together with its own response. This helps to overcome the loss of D-node responses. On the other hand, when sending the response, each D-node piggybacks the probe message upon its response, so the nearby D-nodes will have a chance to hear the probe again, if the original probe from Q-node is lost.

5 Optimal Concurrent Dissemination

As we have seen in Section 3, parallelizing the subitineraries may reduce query latency. However, when S is large, network throughput degrades due to the frequent occurrence of contentions and collisions at the data link and physical layers. Such degradation of the network throughput can, reversely, increase the latency. As a consequence, how to determine an appropriate S is vital to DIKNN performance. In this section, we derive an analytic model that is able to suggest the most appropriate degree of concurrency under various network conditions.

Given the parameters shown in Table 1, our first step is to derive a model that gives the expected dissemination time T under a certain S. For simplicity, we assume the actual dissemination path in each sector proceeds exactly along the subitinerary, no itinerary void encountered during the dissemination, and no packet loss occurs due to the environmental interference, low SNR, and mobility of sensor nodes. We do not consider the node mobility in our analysis. In IEEE 802.15.4 wireless transmission aspect, we assume the slotted *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) mechanism is enabled, no CFP is exercised in a *Superframe*, and the payload (e.g., queries and responses) from upper layers can be transmitted within a *Contention Access Period* (CAP). Note the derivation requires a prerequisite condition that $S \geq 4$ to ensure the function $\sin^{-1}$ being valid.

Without ambiguity, we refer to the "itinerary" as the subitinerary of a sector interchangeably. Suppose it takes $E[hops]$ hops between Q-nodes to finish itinerary traversal, we are able to express T as

$$T = \sum_{i=1}^{E[hops]} (E[TD_i] + E[TP_i]), \tag{5}$$

where $E[TD_i]$ and $E[TP_i]$ denotes expected time for the ith Q-node to collect data and to forward partial query result to the next Q-node, respectively.

Derivations of $E[hops]$, $E[TD_i]$, and $E[TP_i]$ are basically straightforward. Let $l_{itinerary}$ be the length of itinerary and $E[Adv]$ be the expected distance between hops, then

$$E[hops] = \frac{l_{itinerary}}{E[Adv]}. \tag{6}$$

Recalling the contention-based data collection scheme in which the Q-node broadcasts a probe message and waits m seconds for each D-node to response the query, we can obtain $E[TD_i]$ by summing the time the Q-node spends in transmitting probe message and waiting for responses from D-nodes. That is,

$$E[TD_i] = \frac{d_{prob}}{\rho} + \pi r^2 Dm, \tag{7}$$

where d_{prob} denotes the size (in bits) of the probe message.

Assume that every sensor node has the same size of query response, d (in bits). Each Q-node on the subitinerary (except the first one) encounters aD new neighbors (i.e., D-nodes) after a hop, where

$$a = \pi r^2 - 2r^2 \cos^{-1}(E[Adv]/(2r)) + E[Adv]\sqrt{r^2 - (E[Adv]/2)^2}$$

is the area that is not covered by the radio range of the precedent Q-node, as shown by the shaded area in Fig. 3a. The amount of data the ith Q-node transfers to the $(i+1)$th Q-node is $(\pi r^2 D + (i-1)aD)d$. Thus,

$$E[TP_i] = \frac{(\pi r^2 D + (i-1)aD)d}{\rho}. \tag{8}$$

However, (6), (7), and (8) are valid only in the optimal case. $E[TD_i]$ and $E[TP_i]$ are susceptible to interference of wireless communication from adjacent sectors. Next, we give detailed derivations of $E[hops]$, $E[TP]$, and $E[TD]$ by taking the network dynamics into account.

Derivation of $E[hops]$. Recall that each subitinerary consists of init-, peri-, and adj-segments. From (6), we have

$$E[hops] = \frac{l_{init} + l_{peri} + l_{adj}}{E[Adv]}.$$

To proceed with the analysis, we determine how much advance a Q-node can make on each hop by the following lemma.

Lemma 5.1. *A Q-node is expected to find the next Q-node at distance $E[Adv] = r^2\sqrt{D}/(1 + r\sqrt{D})$ along the itinerary.*

The proof of this lemma is shown in the Appendix. Combining (1), (2), (3), and the lemma, $E[hops]$ can be given by

$$E[hops] = \frac{(1 + r\sqrt{D})}{r^2\sqrt{D}} \left(\min\{w/(2\sin(\pi/S)), R\} + \sum_{i=1}^{\lfloor \frac{R - l_{init}}{w} \rfloor} \frac{2\pi(iw)}{S} + \lfloor (R - l_{init})/w \rfloor w \right). \tag{9}$$

Derivation of $E[TP]$. In the LR-WPAN CAP, all sensor nodes contend with each other for channel access. Serious contention can lead to degradation of the network

throughput and growth of $E[TP_i]$ and $E[TD_i]$. To evaluate the amount of degradation, we follow an analytic model proposed in [32] to derive a normalized throughput $\widetilde{S}(u)$,[2] where $0 \leq \widetilde{S}(u) \leq 1$ denotes the fraction of time the channel is used to successfully transmit payload bits when there are u stations contending for channel access. By multiplying ρ with $\widetilde{S}(u)$, we can obtain the actual channel rate of a sensor node under a certain level of contention.

To determine the contention level u, we define the *active Q-nodes* as a set of Q-nodes in different sectors which perform data collection at a specific time. Now, consider the following lemma.

Lemma 5.2. *Given an active Q-node, n_i, in the ith sector, the number of the active Q-nodes in neighbor sectors that have distances less than ε from n_i can be expressed by*

$$U(\varepsilon, \widehat{\Delta}) = \begin{cases} S, & \text{if } \widehat{\Delta} \leq \varepsilon/2, \\ 2\left\lfloor [S \cdot \sin^{-1}(\varepsilon/(2\widehat{\Delta}))]/\pi \right\rfloor, & \text{otherwise,} \end{cases}$$

where $1 \leq i \leq S$, and $\widehat{\Delta}$ denotes the distance between n_i and query point q.

The proof of this lemma is also given in the Appendix. Let $\Delta(i)$ be a function returning the distance between the ith Q-node of the current sector and the query point. There exist $U(r, \Delta(i))$ Q-nodes from adjacent sectors that may result in contention. Thus, $\widetilde{S}(1 + U(r, \Delta(i)))\rho$ denotes the actual rate at which the query message and partial result is transmitted from the ith to the $(i+1)$th Q-node. With this new rate, we can rewrite (8) as

$$E[TP_i] = \frac{(\pi r^2 D + (i-1)aD)d}{\widetilde{S}(1 + U(r, \Delta(i)))\rho}. \tag{10}$$

Derivation of $E[TD]$. Applying the degraded channel rate $\widetilde{S}(1 + U(r, \Delta(i)))\rho$, the data collection time specified by (7) now becomes

$$E[TD_i] = \frac{d_{prob}}{\widetilde{S}(1 + U(r, \Delta(i)))\rho} + \pi r^2 Dm. \tag{11}$$

In addition to the channel rate concern, the *hidden terminal problem* can also affect $E[TD_i]$ due to the lack of RTS/CTS mechanism in IEEE 802.15.4. Consider a scenario in which two probe messages are broadcasted simultaneously from two active Q-nodes of different sectors. Some D-nodes, which locate inside the overlapped coverage of two Q-nodes, will not be successfully informed of the probe due to radio interference. We can see that this phenomenon occurs only when the distance between two active Q-nodes is less than $2r$. By applying the above lemma, $U(2r, \Delta(i))$ now represents the number of nearby active Q-nodes that may result in the hidden terminal problem with the ith Q-node in current sector.

To obtain the probability a probe message is hidden by the others, consider the random back-off scheme, where the time is slotted and data transmission on each station can only occur at the beginning of each slot time. A back-off counter is decremented as long as the channel sensed idle. When the back-off counter reaches zero, data transmission is then launched. The study [32] has derived τ_u denoting the probability a sensor node transmits data (after its random back-off) at a particular slot time, given $u-1$ neighbor Q-nodes contending for the channel access. Let $P_h(i)$ be the probability that the transmission of the probe message from the ith Q-node is hidden by active Q-nodes in other sectors. Since the transmission attempts on different sensor nodes are independent, $P_h(i)$ can be expressed by the probability that the ith Q-node and at least one of the $U(2r, \Delta(i))$ neighbor Q-nodes transmits probe messages at the same slot time, conditioned on the fact that the ith Q-node transmits, i.e.,

$$\begin{aligned} P_h(i) &= \frac{\tau_n\left(1 - (1-\tau_n)^{n-1}\right)}{\tau_n} \\ &= 1 - (1-\tau_n)^{n-1}, \end{aligned}$$

where $n = 1 + U(2r, \Delta(i))$. Suppose every time a probe message is hidden, the active Q-node broadcasts the probe message again and reperforms data collection, then (11) can be rewritten as

$$\begin{aligned} E[TD_i] &= M + \sum_{j=1}^{\infty} P_h(i)^j M \\ &= \frac{M}{1 - P_h(i)}, \text{where} \\ M &= \frac{d_{prob}}{\widetilde{S}(1 + U(r, \Delta(i)))\rho} + \pi r^2 Dm. \end{aligned} \tag{12}$$

Putting (9), (10), and (12) together, we finally obtain (5).

Model validation. The optimal concurrency level derived by our analytic model is verified by extensive simulation results. We demonstrate a typical scenario finding for $k = 300$ nearest neighbors in 256 $\times$ 256 m^2 simulation fields, where 1,000 stationary nodes are randomly located. A heavy query load 20 $queries/s$ is issued. For clear presentation, we make the analytic response time T continuous by omitting the floor functions in the model. As illustrated in Fig. 7, the optimal number of sectors ($S_{opt} = 8$) suggested by the model is very close to the one ($S_{opt} = 7$) revealed by the simulation results. Note the simulated response time T grows dramatically as S increases. This is due to the fact that the Q-node

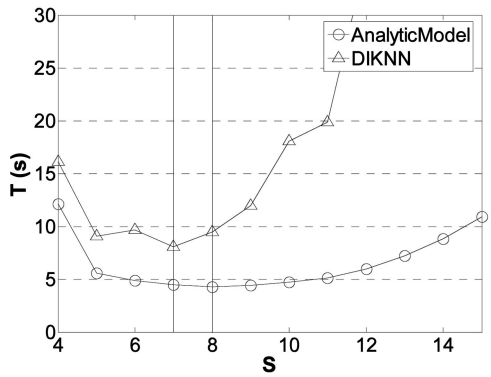

Fig. 7. Comparison of the optimal number of sectors determined by the analytic model ($S_{opt} = 8$) to simulation results ($S_{opt} = 7$).

2. Due to the space limitation, we do not derive $\widetilde{S}(u)$ in this paper. Interested readers can refer to [32] for the detailed discussion.

may not always find the next Q-node along the itinerary, especially when S is large, and the sectors are narrow (at this time, a traversal may go into adjacent sectors). Since we do not target on returning the correct T, our model can efficiently determine the best number of sectors by precisely tracing the trend of T. With this model, a mapping table from S to T can be constructed and stored in each sensor node. By simply looking up the table, a home node can determine the optimal degree of concurrency in $O(1)$ time and start query dissemination thereafter.

6 Performance Evaluation

In this section, we explore performance of DIKNN in terms of query accuracy, query latency, and energy efficiency. The simulation environment is developed based on ns-2 [33]. We study the impact of varying application specifications and network conditions, such as k, the query load, the mobility of sensor nodes, and the packet loss rate.

6.1 Settings and Performance Metrics

We simulate the LR-WPAN environment at 2.4 GHz by disabling the RTS/CTS mechanism and setting the channel rate 250 kbps. The georouting protocol GPSR [27] and DIKNN are implemented above the ns-2 802.11 MAC layer. The optimal number of sectors is determined by our analytic model (in this configuration, $S_{opt} = 8$). Mechanisms (i.e., rendezvous, mobility assurance, and fault-tolerant scheme) coping with network dynamics are enabled. Initially, the sensor nodes are randomly distributed in the simulated field. The mobility of sensor nodes is modeled by the *random waypoint* (RWP) model, in which each sensor node selects an arbitrary destination and moves to the destination at a random speed ranging from 0 to $\mu_{\max}$. Upon arrival, the node selects a new destination and walks again. In our configuration, the mobility of sensor nodes is controlled by varying the maximum moving speed $\mu_{\max}$. By default, $\mu_{\max} = 10\, m/s$. There are 200 sensor nodes in the simulation field, and each with radio range 20 m [16], [34]. By fixing the number of sensor nodes and varying the simulated field from 200 $\times$ 200 to 115 $\times$ 115 m^2, the node degree (i.e., neighbor count of each sensor node) ranges from 5 to 20. The time unit for data collection is 0.018 s, and the query response size of each sensor node is 10 $bytes$. Every simulation run lasts for 100 seconds of simulated time. The performances are obtained by averaging the result over 20 simulation runs. Our confidence level was at 95 percent with the confidence interval of $(X - 1.96\sigma/3.16, X + 1.96\sigma/3.16)$, where X is the mean, and σ is the standard deviation of the samples. Table 2 summarizes the default parameters. Note that we control k such that a query may effect at most 1/3 of the nodes in a network. This prevents a serious bias when the query point is issued near the network border, where nodes may not move freely within the full coverage of the KNN boundary.

To evaluate the simulation result, three performance metrics are employed:

- **Query latency.** The elapsed time (in second) between the time a query is issued by the sink and the time the query responses are returned.
- **Energy consumption.** Amount of energy (in $Joule$) consumed by the nodes' wireless modules in packet transmission and reception during a simulation run.
- **Query accuracy.** As discussed in Section 3.1, the *preaccuracy* and *postaccuracy* are measured separately in our experiments.

TABLE 2
Default Parameters Used in Our Simulation

Parameters	Value	Parameters	Value
Node number	200	r	20 m
Network size	115×115 m^2	Sector number	8 (by model)
Node degree	20	$\mu_{\max}$	10 m/s
Response size	10 bytes	Beacon interval	0.5 s
Channel rate	250 kbps	RTS/CTS	off
m	0.018 s	Query interval	4 s
Rendezvous, FT scheme	enabled	Assurance gain	0.1

We focus on comparison between DIKNN and the in-network executions: the naive approaches (KPT) [11], [12] and the Peer-tree approach [20]. Note the KNN boundary estimation techniques proposed in [11] and [12] lead to quadratic growth of the boundary area as k increases. The query execution can easily flood the entire network. For example, when $k = 20$ and $MHD = 15$, the returned radius length $R = 20 \cdot 15 = 300$ exceeds twice the field edge, resulting that the boundary area is six times larger than the network size. For fair comparisons, we simulate KPT in which the KNNB algorithm is adopted for boundary estimation, and a spanning tree is constructed for data collection after the boundary is determined. In Peer-tree, a global index structure, R-tree [13], is built to preserve the MBR hierarchy as described in Section 2. To avoid a skew index, we partition the network into a 5 $\times$ 5 grid. Every cell represents an MBR within which a stationary clusterhead is prelocated and its address is known by every sensor node. Each sensor node periodically sends a notification of existence to its closest clusterhead. If a clusterhead does not hear from a child after a period of time, it deletes the node and updates the MBR record.

6.2 Observations

Before studying the DIKNN performance, we discuss some observations from our simulation result that are worthy to be mentioned. We apply DIKNN to some large-scale sensor distributions obtained in [35]. The trace format of ns-2 is modified so that the query execution can be visualized. Fig. 8 demonstrates a scenario for finding $k = 500$ caribous around an arbitrary query point. The concurrent itinerary traversals are illustrated in Fig. 8a, where successive hops between Q-nodes are connected with lines. As pointed out by the arrows, we can see that the itinerary void appears occasionally. When a void is encountered, the itinerary traversal switches to the perimeter forwarding mode, which bypasses the vacancy by walking into the nearby segments or sectors. During the perimeter forwarding, some sensor nodes, as surrounded by the rectangle in Fig. 8b, may not be informed of the query message because they are isolated within a sector. Such a phenomenon can, empirically, cause 0.2 percent to 1 percent degradation of both the post- and prequery accuracy. Fortunately, from mobility of sensor nodes, this effect can be alleviated by issuing a serious of

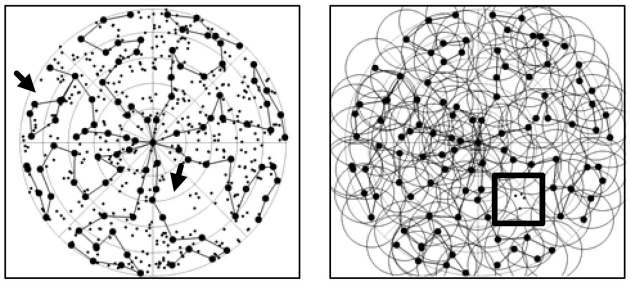

Fig. 8. Visualization of DIKNN execution over the real-world sensor distributions [35].

queries within a time interval. Our demonstration verifies the applicability of DIKNN to real deployments.

6.3 Scalability

The application specified k directly affects the number of nodes involved with the query. In this section, we investigate the impact of k by varying k from 20 to 100. The parameter $\mu_{\max}$ is set to 10 m/s, and the query interval is exponentially distributed with mean 4 s. Fig. 9a shows that both Peer-tree and KPT grow faster than DIKNN as k increases. This is due to the overhead of Peer-tree to route query messages between different levels of the R-tree hierarchy that incurs many unnecessary node visits, and the overhead of KPT to construct and maintain the spanning tree during data collection. Such overhead also leads to the higher energy consumption, as shown in Fig. 9b. KPT consumes more energy than the others when $k = 100$ due to a serious degree of collisions and large retransmissions of data in the tree. Fig. 9c shows that Peer-tree has the postaccuracy below average since the cluster-heads may not obtain the most current position of each sensor node. The postaccuracy of KPT also degrades when k is large because of the long latency in data collection. The preaccuracy of DIKNN and KPT, as depicted in Fig. 9d, varies when $k \leq 60$. This occurs due to the error in KNN boundary estimation, since when k (correspondingly, R) is small, the error rate is relatively large. However, when $k > 60$, boundary error shrinks and DIKNN becomes precise; while the others continuously degrade due to their long latency. Among these results, DIKNN exhibits superior improvements in query latency and energy efficiency while preserving a high level of accuracy.

6.4 Impact of Query Load

The impact of query load is studied in this section. The query arrival rate is varied by changing the mean of the exponentially distributed interval from 10 to 1 s. We set $k = 40$ and $\mu_{\max} = 10\ m/s$. In Figs. 10a and 10b, we can see that Peer-tree has high latency and consumes large energy under all loads because of the routes in R-tree hierarchy and mass information updates, respectively. As depicted in Fig. 10a, KPT responds faster than DIKNN under small query loads, since DIKNN has to wait a time unit ($m = 0.018\ s$) for every D-node to collect data. However, as query load increases, the latency of KPT grows because of serious collisions and retransmissions. This overhead also affects the energy consumption of KPT, as shown in Fig. 10b, which results in the highest growing rate among all work. Note the energy consumption of DIKNN also grows. This is because we evaluate the total energy consumption in each simulation run, which encompasses more query processing when the load increases. Thus, DIKNN precludes excessive overhead and has a stable performance under heavy query load.

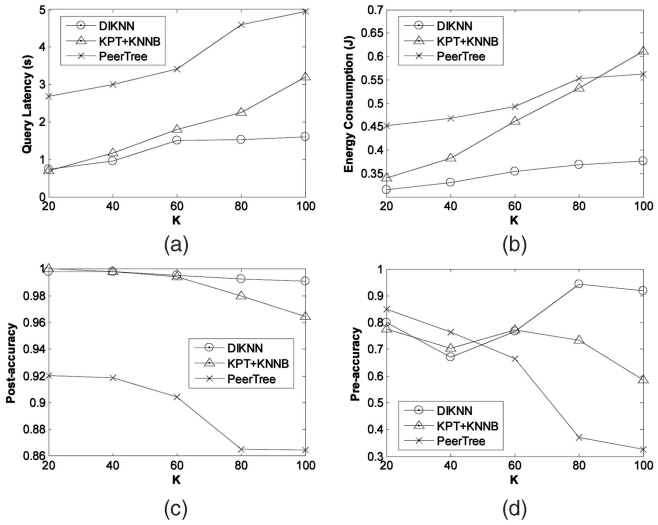

Fig. 9. Scalability of DIKNN.

6.5 Robustness

The impact of packet loss rate (from 0.1 to 0.6) is given in Figs. 10c and 10d. We examine the robustness of DIKNN by varying the packet loss rate from 0.1 to 0.6. We set $k = 40$, $\mu_{\max} = 10\ m/s$, and mean query interval 4 s as the default. Two versions, DIKNN (with fault-tolerant scheme) and DIKNNwoFT (without fault-tolerant scheme) are considered. Fig. 10d shows that the accuracy of all mechanisms falls when the packet loss rate increases. However, benefiting from the fault-tolerant scheme, DIKNN remains to have a high accuracy when the packet loss rate is less than 0.4 and outperforms all the other works, including DIKNNwoFT. As depicted in Fig. 10c, the fault-tolerant data collection scheme consumes relatively small energy and thus is suggested to be

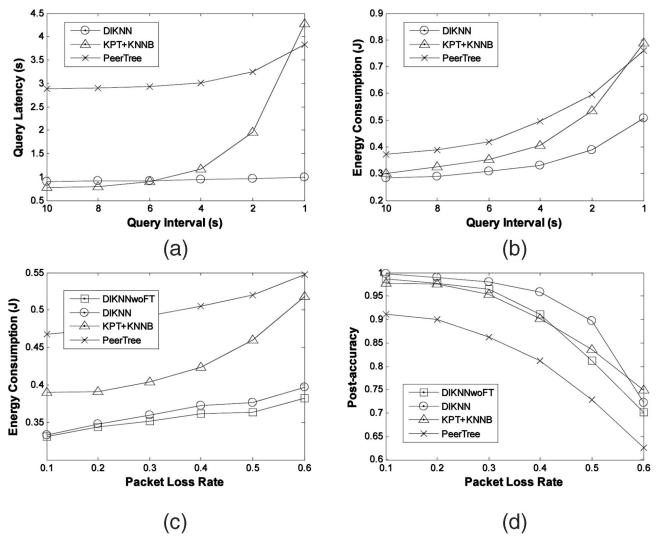

Fig. 10. Impact of query load and packet loss rate.

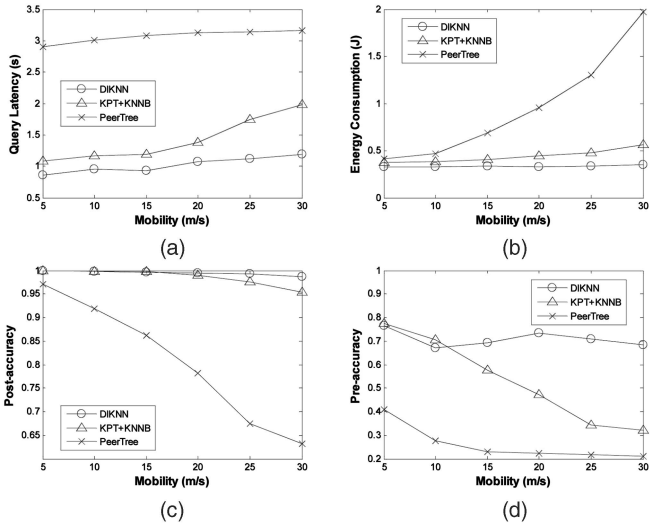

Fig. 11. Impact of mobility.

turned on to sustain network dynamics. DIKNN reveals great feasibility to the error-prone sensor environments.

6.6 Impact of Network Dynamics

In this section, we study the impact of sensor movements by varying $\mu_{\max}$ from 5 to 30 m/s. We set $k = 40$, and the query interval is exponentially distributed with mean 4 s. Fig. 11a shows that Peer-tree has high latency in all moving speeds because of the routes in hierarchy, as stated in the previous section. The latency of KPT grows because of tree maintenance overhead. Data at child levels of the tree will have to wait to be relayed to the root until the structures of the parent layers are settled. In Fig. 11b, we can see that the energy consumption of Peer-tree increases rapidly, because there are more sensor nodes moving across MBRs, which results in excessive information updates. The post- and preaccuracy of Peer-tree shown in Figs. 11c and 11d degrade dramatically because the latest position of each sensor node can hardly be traced by the clusterheads under high mobility. A clusterhead simply drops packets (i.e., the queries) if they cannot be routed to the destinations in the MBR record. The preaccuracy of KPT also degrades due to its latency. Benefiting from the novel itinerary traversals that maintain no infrastructure, DIKNN has stable performance under various mobility conditions. Again, DIKNN offers prominent advantages over energy efficiency and query latency in mobile sensor networks while rendering a high level of accuracy.

7 Conclusion

In this paper, we proposed a cost-effective solution, DIKNN, for handling the KNN queries in mobile sensor networks. DIKNN integrates query propagation with data collection along a well-designed itinerary traversal, which requires no infrastructures and is able to sustain rapid change of the network topology. A simple and effective KNNB algorithm has been proposed to estimate the KNN boundary under the trade-off between query accuracy and energy efficiency. Dynamic adjustment of the KNN boundary has also been addressed to cope with spatial irregularity and mobility of sensor nodes. Determination of the optimal concurrency for query disseminations is modeled in this paper. From extensive simulation results, DIKNN exhibits a superior performance in terms of energy efficiency, query latency, and accuracy in various network conditions.

Appendix A

A.1 Proof of Lemma 2

Let X be a random variable, which denotes the distance between the current Q-node and a neighbor along the itinerary. Since we assume that sensor nodes are uniformly distributed within the KNN boundary, X has uniform distribution with CDF:

$$F_X(x) = P\{X \le x\} = x/r, 0 \le x \le r.$$

Let $Adv = max\{X_1, X_2, \ldots, X_i\}$ be an ith-ordered statistic of X, which denotes the maximum of distances between the current Q-node and i neighbors along the itinerary, where $i \in \mathbb{N}$. Assume the location of each sensor node is independent with others, then the CDF of Adv can be expressed by

$$\begin{aligned} F_{Adv}(x) &= P\{Adv \le x\} \\ &= P\{X_1 \le x \wedge X_2 \le x \wedge \ldots \wedge X_i \le x\} \\ &= (x/r)^i, \end{aligned}$$

leading to the pdf

$$f_{Adv}(x) = dF_{Adv}(x)/dx = (i/r)(x/r)^{i-1}.$$

Given the parameter D, there exist $r\sqrt{D}$ sensor nodes that locate exactly on the itinerary within the radio coverage of the current Q-node. Since a Q-node will always choose the farthest neighbor to be the next Q-node, by replacing i with $r\sqrt{D}$, we can derive $E[Adv]$ by

$$\begin{aligned} E[Adv] &= \int_0^r x f_{Adv}(x)dx = \frac{r\sqrt{D}x(x/r)^{r\sqrt{D}}}{(1+r\sqrt{D})}\Big|_0^r \\ &= r^2\sqrt{D}/(1+r\sqrt{D}). \end{aligned}$$

□

A.2 Proof of Lemma 3

Given the assumption that the sensor nodes are uniformly distributed within the boundary and every node has the same capability with same data size, the relative expected location of an active Q-node in a sector will be the same with that in all the other sectors at any particular time during dissemination. Since the structures of subitineraries are identical in different sectors, it follows two direct consequences: 1) all active Q-nodes in different sectors are expected to have the same distance from the query point; 2) let $\widehat{u}$, $1 \le \widehat{u} < S/2$, be an integer, the expected angle $\theta_{\widehat{u}}$, included by the line connecting q and the active Q-node in the ith sector and the line connecting q and the active node in the $(i+\widehat{u})$th sector, is fixed by the value $\theta_{\widehat{u}} = \widehat{u}(2\pi/S)$.

From consequence 1, these active Q-nodes together form a circle that has diameter of length $2\widehat{\Delta}$. If $2\widehat{\Delta} \le \varepsilon$, then all active Q-nodes has mutual distances less than ε, yielding $U(\varepsilon, \widehat{\Delta}) = S$, and we finish the proof.

Otherwise, let $d_{\widehat{u}}$ be the distance between two active Q-nodes of the ith sector and the $(i+\widehat{u})$th sector. From

consequence 2, we have $\sin(\theta_{\widehat{u}}/2) = (d_{\widehat{u}}/2)/\widehat{\Delta}$, leading to $\sin(\widehat{u}\pi/S) = (d_{\widehat{u}}/2)/\widehat{\Delta}$. Thus, $d_{\widehat{u}}$ can be expressed as

$$d_{\widehat{u}} = 2\widehat{\Delta}\sin\left(\frac{\widehat{u}\pi}{S}\right).$$

We can see that if $d_{\widehat{u}}$ is less than ε, the active Q-node in the $(i+\widehat{u})$th sector will have a distance less than ε from n_i and so does the active Q-nodes in the $(i+\widehat{u}-1)\text{th}, \ldots, (i+1)\text{th}$ sectors. By giving constraint $d_{\widehat{u}} \le \varepsilon$, we have $2\widehat{\Delta}\sin(\widehat{u}\pi/S) \le \varepsilon$, which yields

$$\widehat{u} \le \frac{S \cdot \sin^{-1}(\varepsilon/(2\widehat{\Delta}))}{\pi} < \frac{S}{2}.$$

Thus, there are $\left\lfloor S\sin^{-1}(\varepsilon/(2\widehat{\Delta}))/\pi \right\rfloor$ active Q-nodes from the $(i+1)$th sector to the

$$\left(i + \left\lfloor S\sin^{-1}(\varepsilon/(2\widehat{\Delta}))/\pi \right\rfloor\right)\text{th}$$

sector, which have a distance less than ε from n_i. Note that $\sin(\widehat{u}\pi/S)$ is a monotone increasing function with inverse larger than 0 and less than $\pi/2$, since $1 \le \widehat{u} < S/2$ implies that $0 < \widehat{u}\pi/S < \pi/2$. As a consequence, $\widehat{u}$ is less than $S/2$. By considering the sectors in both clockwise and counterclockwise directions, we have

$$U(\varepsilon, \widehat{\Delta}) = 2\left\lfloor [S \cdot \sin^{-1}(\varepsilon/(2\widehat{\Delta}))]/\pi \right\rfloor.$$

□

ACKNOWLEDGMENTS

The work was supported in part by the National Science Council of Taiwan, under Contracts NSC93-2752-E-002-006-PAE. The authors are grateful to Professor Wang-Chien Lee for his suggestions and Yingqi Xu for discussions over the ns-2 simulation. K.-T. Chuang did this work while with National Taiwan University.

REFERENCES

[1] U.D. of TraNsportation, "Intelligent Transportation System Joint Program Office Home," http://www.its.dot.gov, 2006.
[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS),* 2002.
[3] F. of Am. Scientists, "Remote Battlefield Sensor System (Rembass)," http://www.fas.org, 2000.
[4] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. Abbadi, "Approximate Nearest Neighbor Searching in Multimedia Databases," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* 2001.
[5] A.A.H.D. Chon and D. Agrawal, "Range and KNN Query Processing for Moving Objects in Grid Model," *Mobile Networks and Applications,* vol. 8, no. 4, 2003.
[6] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," *Proc. ACM SIGMOD,* 1995.
[7] Z. Song and N. Roussopoulos, "K-Nearest Neighbor Search for Moving Query Point," *Proc. Int'l Symp. Spatial and Temporal Databases (SSTD),* 2001.
[8] M.-S. Chen, P.S. Yu, and K.-L. Wu, "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 1, pp. 161-173, Jan./Feb. 2003.
[9] W. Lee and B. Zheng, "DSI: A Fully Distributed Spatial Index for Location-Based Wireless Broadcast Services," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS),* 2005.
[10] B. Liu, W. Lee, and D. Lee, "Distributed Caching of Multi-Dimensional Data in Mobile Environments," *Proc. Int'l Conf. Mobile Data Management (MDM),* 2005.
[11] J. Winter and W. Lee, "KPT: A Dynamic KNN Query Processing Algorithm for Location-Aware Sensor Networks," *Proc. Int'l Workshop Data Management for Sensor Networks (DMSN),* 2004.
[12] J. Winter, Y. Xu, and W. Lee, "Energy Efficient Processing of k Nearest Neighbor Queries in Location-Aware Sensor Networks," *Proc. Ann. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous),* 2005.
[13] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. ACM SIGMOD,* 1984.
[14] G. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," *ACM Trans. Database Systems,* vol. 24, no. 2, 1999.
[15] M. Mokbel, X. Xiong, and W. Aref, "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-Temporal Databases," *Proc. ACM SIGMOD,* 2004.
[16] IEEE Std. 802.15.4-2003, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks,* 2003.
[17] IEEE Std. 802.11-1997, *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* 1997.
[18] M. Bawa, A. Gionis, H.G. Molina, and R. Motwani, "The Price of Validity in Dynamic Networks," *Proc. ACM SIGMOD,* 2004.
[19] Y. Xu, W. Lee, J. Xu, and G. Mitchell, "Processing Window Queries in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* 2006.
[20] M. Demirbas and H. Ferhatosmanoglu, "Peer-to-Peer Spatial Queries in Sensor Networks," *Proc. Int'l Conf. Peer-to-Peer Computing (ICP2PC),* 2003.
[21] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom,* 2000.
[22] J. Kahn, R. Katz, and K. Pister, "Next Century Challenges: Mobile Networking for Smart Dust," *Proc. ACM MobiCom,* 1999.
[23] R. Cheng, B. Kao, S. Prabhakar, A. Kwan, and Y. Tu, "Adaptive Stream Filters for Entity-Based Queries with Non-Value Tolerance," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* 2005.
[24] D. Niculescu and B. Nath, "Trajectory Based Forwarding and Its Applications," *Proc. ACM MobiCom,* 2003.
[25] S. Patil, S. Das, and A. Nasipuri, "Serial Data Fusion Using Space-Filling Curves in Wireless Sensor Networks," *Proc. Conf. Sensor and Ad Hoc Comm. and Networks (SECON),* 2004.
[26] C. Gui and P. Mohapatra, "Virtual Patrol: A New Power Conservation Design for Surveillance Using Sensor Networks," *Proc. Int'l Symp. Information Processing in Sensor Networks (IPSN),* 2005.
[27] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. ACM MobiCom,* 2000.
[28] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric Ad-Hoc Routing: Of Theory and Practice," *Proc. Ann. ACM Symp. Principles of Distributed Computing (PODC),* 2003.
[29] Y. Kim, R. Govindan, B. Karp, and S. Shenker, "On the Pitfalls of Geographic Face Routing," *Proc. Discrete Algorithms and Methods for Mobile Computing and Comm. (DIALM),* 2005.
[30] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing," *Proc. ACM MobiHoc,* 2003.
[31] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin, "Coping with Irregular Spatio-Temporal Sampling in Sensor Networks," *ACM SIGCOMM Computer Comm. Rev.,* vol. 34, no. 1, 2004.
[32] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 3, 2000.
[33] *The Network Simulator,* http://www.isi.edu/nsnam/ns, 2000.
[34] I. Howitt and J. Gutierrez, "IEEE 802.15.4 Low Rate—Wireless Personal Area Network Coexistence Issues," *Wireless Comm. and Networking,* vol. 3, nos. 16-20, 2003.
[35] "Caribou Population Distribution in Gros Morne National Park Greater Ecosystem," http://www.pc.gc.ca/apprendre-learn/prof/sub/eco/itm5/fi-lr6/caribou_E.asp, 2003.

Shan-Hung Wu received the BS degree from the Department of Information Management, National Central University, Jhongli, Taiwan, and the MS degree from Department of Computer Science and Information, National Taiwan University, Taipei. He is currently a PhD candidate in the Department of Electrical Engineering, National Taiwan University, Taipei, and a research staff member at Telcordia Technologies. His research interests include distributed data management, spatial/temporal databases, wireless and sensor networks, and performance modeling.

Kun-Ta Chuang received the BS degree from the National Taiwan Normal University, Taipei, in 2000, and the PhD degree in communication engineering from the National Taiwan University, Taipei, in 2006. He is currently serving as a software engineer in SYNOPSYS Inc. to develop physical verification tools. His research interests include data mining, mobile data management, and electronic design automation.

Chung-Min Chen received the BS degree in computer science from the National Taiwan University and the PhD degree in computer science from the University of Maryland, College Park. He is currently a director and senior scientist at Telcordia Technologies. His research interests include data management, network management, and their applications. He is a member of the IEEE and the IEEE Computer Society.

Ming-Syan Chen received the BS degree in electrical engineering from National Taiwan University, Taipei, and the MS and PhD degrees in computer, information and control engineering from the University of Michigan, Ann Arbor, in 1985 and 1988, respectively. He is now a Distinguished Research Fellow and the director of the Research Center of Information Technology Innovation (CITI) in the Academia Sinica, Taiwan, and is also a distinguished professor in the Electrical Engineering Department, National Taiwan University. He was a research staff member at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996, the director of the Graduate Institute of Communication Engineering from 2003 to 2006, and also the president/CEO of the Institute for Information Industry (III), Taiwan, from 2007 to 2008. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 260 papers in these research areas. In addition to serving as program committee members in many conferences, he served as an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* (TKDE) from 1997 to 2001. He is currently on the editorial board of the *Very Large Data Base (VLDB) Journal*, the *Knowledge and Information Systems (KAIS) Journal*, and the *International Journal of Electrical Engineering.* He is a distinguished visitor of the IEEE Computer Society for Asia-Pacific from 1998 to 2000 and also from 2005 to 2007 (invited twice). He served as the international vice chair for INFOCOM 2005, program chair of PAKDD 2002, program cochair of MDM 2003, program vice chair of IEEE ICDE 2008, CEC/EEE 2006, ICDE 2006, ICDCS 2005, ICPP 2003, and VLDB 2002, and many other program chairs and cochairs. He was a keynote speaker on Web data mining in the International Computer Congress in 1999 and IEEE ISM in 2007 and a tutorial speaker on Web data mining in DASFAA 1999 and on parallel databases in the 11th IEEE ICDE in 1995. He was also a guest coeditor for *IEEE TKDE* special issue for data mining published in the December 1996 issue. He holds, or has applied for, 18 US patents and seven ROC patents on data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He is a recipient of the National Science Council (NSC) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, K.-T. Li Research Penetration Award for his research work, and the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions and patent applications. He is a fellow of the ACM and also a fellow of the IEEE.