Energy Efficient Processing of K Nearest Neighbor Queries in Location-aware Sensor Networks*

Julian Winter Yingqi Xu Wang-Chien Lee
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802
Email: {jwinter, yixu, wlee}@cse.psu.edu

Abstract

The k nearest neighbor (KNN) query, an essential query for information processing in sensor networks, has not received sufficient attention in the research community of sensor networks. In this paper, we examine in-network processing of KNN queries by proposing two alternative algorithms, namely the GeoRouting Tree *(GRT) and the* KNN Boundary Tree *(KBT). The former is based on a distributed spatial index structure and prunes off the irrelevant nodes during query propagation. The latter is based upon ad-hoc geographic routing and first obtains a region within which at least k nearest sensor nodes are enclosed and then decides the k nearest nodes to the query point. We provide an extensive performance evaluation to study the impact of various system factors and protocol parameters. Our results show that GRT yields a good tradeoff between energy consumption and query accuracy in static scenarios. On the other hand, KBT achieves better energy efficiency while being more tolerant to network dynamics.*

1 Introduction

Sensor networks have been adopted by a wide range of applications for collecting, processing and disseminating environmental data and thus have attracted considerable research attention. Many research projects focus on different aspects of sensor networks (e.g., index structures, data storage, routing algorithms, data dissemination and aggregation techniques) in order to extract and leverage the sensed data for various applications [5, 12, 13, 14, 17, 26]. Spatial queries that extract sensed data from specific locations in the network are particularly important since the data is often geographically distributed. One of the essential classes of spatial queries is the k *nearest neighbor (KNN)* query which has particular relevance for sensor networks. For instance, in order to make the tradeoff between the energy efficiency, system performance and operation fidelity, the application usually requires the readings from the k sensor nodes closest to the location of an event [11, 15]. KNN queries give sensor network operators a way to sample the environment around the location of an event through the nearest neighbors of the location with a specific sample size (k). In this paper we focus on providing an energy-aware and robust solution for locating and querying these k nearest sensor nodes. In general, the KNN problem is defined as follows:

Definition: (k nearest neighbor problem).
Given a set M of N sensor nodes and a geographical location (denoted as a query point q), find a subset M' of k nodes ($M' \subseteq M$, $k \leq N$) such that $\forall n_1 \in M'$ and $\forall n_2 \in M - M'$, $dist(n_1, q) \leq dist(n_2, q)$.[1]

In this paper, a *KNN query* retrieves sensed data from the k sensor nodes nearest to the given query point q.

KNN queries have been extensively studied in traditional spatial databases [18, 20, 22]. Such centralized databases utilize indices for which efficient algorithms for executing KNN queries have been presented. However the wireless sensor network environment raises new research challenges over these database systems: 1) Sensor nodes operate on an extremely frugal energy budget. Traditional centralized query execution plans which require all sensor nodes to periodically report their readings to a central node (e.g. base station), could quickly drain the energy resources of sensor nodes; 2) Sensor nodes have limited computation, storage and communication abilities. Complex centralized index structures are not applicable to sensor networks as they re-

*Wang-Chien Lee was supported in part by US National Science Foundation grant IIS-0328881.

[1] Function $dist(\cdot, \cdot)$ denotes the Euclidean distance between two geographical locations.

quire a large amount of storage space and excessive communication among sensor nodes; 3) Sensor networks are mainly characterized by a dynamic topology due to node failures (e.g. energy depletion), unattended and untethered operations and mobility of sensor nodes [24, 25]. The network dynamics make the query executions that rely on network infrastructure unfeasible for sensor networks as the infrastructure maintenance cost may overwhelm the network performance.

Thus, energy efficiency, scalability and robustness to dynamic network topologies have been taken as the design goals of our proposed KNN query algorithm. More specifically, an efficient KNN query algorithm should 1) minimize the number of sensor nodes involved with query execution since the radio operations dominate the energy consumption of sensor networks; 2) minimize the total amount of data transmitted; 3) evenly distribute the responsibility of query execution to all involved sensor nodes. Motivated by these requirements, we propose two KNN query processing algorithms, *GeoRouting Tree* (GRT) and *KNN Bounded Tree* (KBT) which reflect two different design philosophies[2]. GRT, inspired by R-tree [6], is a long-running, network-sized distributed index structure which capitalizes on the spatial proximity of the sensor nodes. When a KNN query propagates along the GRT, the GRT algorithm tries to prune out the nodes that are definitely not one of the k nearest neighbors based on their geographical location. We observe that by reducing the number of nodes queried the GRT algorithm reduces energy consumption in static networks. However, it is not suitable for dynamic networks due to the high maintenance cost of its index structure. Compared with GRT, KBT is a short-lived and smaller tree infrastructure. The KBT algorithm first estimates a region which is as small as possible while still large enough to contain the k nearest neighbors for the given query point. Inside the estimated region a KBT is built along which the query propagation and data collection are conducted.

This paper presents the following major contributions:

- We propose a KNN search algorithm based upon GRT index which intelligently prunes off irrelevant nodes during query propagation, thus reducing the energy consumption while maintaining high query accuracy.
- As an alternative for dynamic scenarios, we propose the KBT algorithm which adopts an indexless approach and minimizes the responsibility of individual sensor nodes for query execution. Our design results in a robust KNN algorithm that gives high energy efficiency and low query latency.
- Simulations have been conducted to evaluate the performance of both KNN search algorithms. GRT demonstrates a good tradeoff between energy consumption and query accuracy in static networks. The indexless algorithm, KBT exhibits superior energy efficiency and query latency with reasonable query accuracy.

[2]Without confusion, we let GRT and KBT denote both the KNN query processing algorithms and the network infrastructures they are based on.

The rest of the paper is organized as follows. In Section 2 we describe our assumptions as well as introduce the GRT [4] and GPSR routing algorithm [9] which are used in our research. Section 3 and Section 4 present the detailed algorithm designs for the GRT algorithm and the KBT algorithm, respectively. The performance of both approaches under different circumstances is examined in Section 5. Section 6 discusses the related research work, and finally Section 7 gives the conclusive remark and future work.

2 Preliminaries

In this section we present the assumptions made in our study. Then we give a brief review for the GRT [4] structure and GPSR algorithm [9] which are adopted by our GRT algorithm and KBT algorithm, respectively.

2.1 Assumptions

We assume that a wireless network consisting of N sensor nodes is deployed in a two-dimensional fixed area. Each sensor node is location-aware. Due to the mobility of sensors as well as their tendency to fail, we assume that the topology of the sensor networks changes dynamically.

A KNN query can be issued by any sensor node instead of by one or more stationary access points in the networks. The application expects the query results from the same sensor node which issued the query. Finally, we assume that sensing data are stored locally on the sensor nodes which implies that in order to extract the data from the k nearest sensor nodes, KNN queries have to be propagated to those nodes.

2.2 GeoRouting Tree (GRT)

The GRT, a tree infrastructure similar to R-tree, was proposed for executing queries on sensor networks in [4]. In the GRT (shown in Figure 1), each sensor node X maintains a minimum bounding rectangle (MBR) for its child Y. The MBR for node Y is the smallest rectangle which encloses the geo-locations of all the sensor nodes in the tree rooted at Y (including Y). MBRs are constructed from the leaf nodes up and MBRs for leaf nodes are geographical points. GRT captures the spatial properties of sensor nodes and does not require the minimum and maximum number nodes in one MBR.

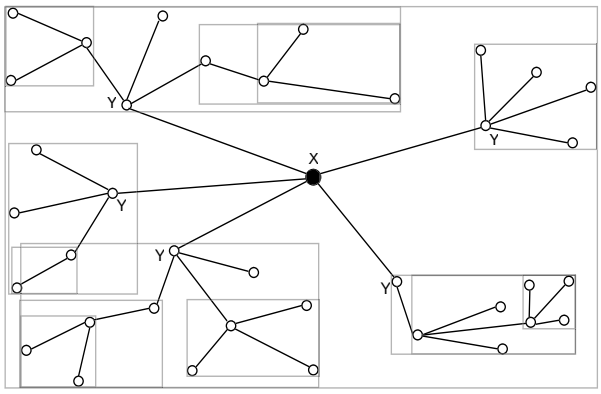


Figure 1. GeoRouting Tree

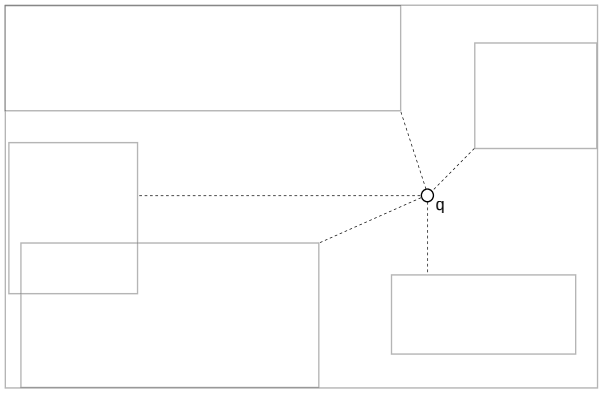


Figure 2. MINDIST

To accommodate to network topology changes, we adopt a simple beacon algorithm [9] for GRT maintenance. Periodically, each sensor node broadcasts a beacon message that includes its identifier, position and MBR. The parent X recalculates Y's MBR after receiving its child Y's beacon. If the changes on Y's MBR affects the correctness of X' MBR, X has to update its MBR and report the update to its parent as well. If a parent node does not hear from a child after a period of time the parent deletes this child and updates the MBR. On the other hand, if a child does not hear from its parent, it selects a new parent from its neighbor table and informs the new parent which involves recomputing the MBRs as well. In this paper, we designed a KNN search algorithm based on GRT. The detailed algorithm will be presented shortly in Section 3.

2.3 GPSR Algorithm

KBT, our second KNN algorithm, is built upon GPSR [9], a geographical multi-hop routing algorithm. The rationale of this selection is explained in Section 4. GPSR works in two modes: *greedy* mode and *perimeter* mode. In greedy mode, the forwarding node forwards the message to the neighbor closest to the destination. If no such neighbor exists, the algorithm switches to the perimeter mode, which given a planarized graph of the network topology routes messages around voids in the network. GPSR returns to greedy mode from perimeter mode when the packet reaches a node closer to the destination than the node at which the packet entered the perimeter node. GPSR routes packets to the *nearest neighbor* (NN) of the destination which may not be associated with a specific sensor node. This property has been used in other studies; interested readers are referred to [21].

3 KNN Algorithm Over GeoRouting Tree

In this section, we describe the KNN algorithm operating on GRT which exists during the network lifetime and spans the entire network. For the sake of presentation we assume the algorithm starts from the root node[3].

3.1 Data Structures for GRT Algorithm

We first present several data structures required by the GRT algorithm. For a given query point q and a sensor node $\mathcal{S}$,

- $MINDIST(\mathcal{S}, q)$ is the minimum distance from q to $\mathcal{S}$'s MBR. The formal definition of MINDIST is given by [18]. Figure 2 shows an example of MINDIST denoted by dashed lines from q to MBRs. Intuitively, MINDIST determines the minimum distance from q to the MBR that encloses the subtree rooted at node $\mathcal{S}$.
- $NTable(\mathcal{S})$ is the neighbor table for sensor node $\mathcal{S}$. A neighbor table stores the information about the nodes one-hop away from $\mathcal{S}$ including their id and their current geographical location. The size of the $NTable$ is the total number of neighbors of $\mathcal{S}$ denoted by m.
- $KTable(\mathcal{S}, q)$ is maintained by sensor node $\mathcal{S}$. From $\mathcal{S}$ and $\mathcal{S}$'s neighbor nodes, the k closest sensor nodes to q are selected. Their id and their distance to the query point (i.e., $dist(q, id)$ are stored in $KTable(\mathcal{S}, q)$. If $k > m$, the remaining $(k - m)$ entries have their id set as NULL and their $dist(q, id) = \infty$. The $Ktable$ is sorted based on $dist(q, id)$ in ascending order. For the

[3]Queries inserted from non-root nodes have to be routed to the root node first. Other solutions, such as building multiple trees rooted at different sensor nodes or starting the GRT algorithm from any sensor nodes, are possible but outside the scope of this research.

simplicity of the presentation we denote $dist^i_{\mathcal{S}}(q, id)$ as the $dist(q, id)$ of i^{th} entry in $KTable(\mathcal{S}, q)$.

- D_{max} records the maximum distance between the the query point q to sensor nodes from the current k closest sensor nodes. D_{max} is disseminated along with the query and is updated as the query is propagated.

3.2 GRT Algorithm

The KNN search algorithm over GRT works as follows. Once the root node receives the query from the application, it computes its $MINDIST(0, q)$, forms its $KTable(0, q)$, and initializes D_{max} as the $dist^k_0(q, id)$ which is the $dist$ of the last entry in $KTable(0, q)$[4]. The root node then broadcasts the query along with the D_{max} value. Any child node $\mathcal{S}$ that receives the query computes its $MINDIST(\mathcal{S}, q)$ and compares $MINDIST(\mathcal{S}, q)$ with the received D_{max}. If $MINDIST(\mathcal{S}, q) > D_{max}$ node $\mathcal{S}$ drops the query, since all sensors in $\mathcal{S}$'s subtree have a distance farther than D_{max} from the query point q. Otherwise, node $\mathcal{S}$ forms $KTable(\mathcal{S}, q)$ based on its $NTable$, sets $D_{max} = Min\{D_{max}, dist^k_{\mathcal{S}}(q, id)\}$, and broadcasts the query along with the new D_{max}. The process is repeated until the query reaches the leaf nodes. Once the propagation procedure stops, the sensor node returns its $KTable$ and query execution results to its parent. The parent at each level aggregates all the $KTables$ received from its children and extracts the k entries which have smallest $dist(q, id)$ as a new $KTable$ as well as the query results from these k nodes and reports them to its parent. This aggregation process is repeated until eventually the root receives results from all its children and forms a new $KTable$ that contains the k nearest sensor nodes to the query point along with their associated query results.

The above KNN algorithm over GRT, usually referred as the *branch-and-bound* technique [18], prunes the subtrees that definitely do not have nodes within k closest nodes to the query point thereby reducing overall communication and conserving energy. However, GRT algorithm raises several performance issues. First, as a long-lasting network-sized infrastructure, GRT may incur excessive construction cost as well as maintenance cost when the network topology changes frequently. Second, the KNN algorithm over GRT requires substantial storage space since many data structures (e.g., parent table, children table, $KTable$ and $NTable$) have to be stored. This storage requirement would increase significantly with either increasing network density or the value of k.

Observing these problems, we designed an alternative KNN search algorithm over KBT, a smaller-sized and short-lived tree structure triggered by each query. Compared with the GRT algorithm, KBT effectively reduces the cost of constructing and maintaining the tree structure as well as the storage requirements. Meanwhile, KBT is able to reduce the number of nodes involved with query execution as well.

[4]Without loss of generality, we set the node ID for the root node to 0.

4 KNN Algorithms over KBT

We argue that the KNN search algorithm has to be 1) more distributed so that the responsibility of individual nodes is minimized, making the network less vulnerable to single node failure, and 2) more localized, so that the number of nodes involved in query processing is minimized. Meanwhile, we are aware of geographical routing algorithms that approach shortest-path routing that can reach any node in the network as long as its geographical location is known. These routing algorithms require a periodic beacon message to keep the list of neighbors of each node updated.

Our proposed KNN search algorithm over KBT is facilitated by the Greedy Perimeter Stateless Routing (GPSR) algorithm [9]. The basic idea of KBT is to find the KNN sensor nodes by searching an estimated region of the network. This region (called the *KNN search region*) is a circular region centered at the query point that is estimated to be as small as possible while still large enough to enclose the k nearest sensor nodes. Clearly, the performance of the KBT algorithm heavily depends on the radius of the circular KNN search region which determines the number of nodes to be accessed for query execution. Additionally, the techniques of disseminating the query and collecting data inside the KNN search region are also important for determining the query latency. In the following, we discuss the above issues in detail by breaking the KBT algorithm into several phases: 1) forwarding the query toward the query point and estimating the radius size of the KNN search region; 2) propagating the query inside the KNN search region and locating the k nearest sensor nodes; 3) returning the query results to the application.

4.1 Estimation of KNN Search Region

Upon receiving a query from the application, the sensor node forwards the query toward the *home node* (the nearest neighbor to the query point) using the GPSR routing algorithm. The query point is specified by the application and not necessarily associated with any sensor node. The home node is responsible for estimating the size of the KNN search region based on the information collected during the query propagation. In this paper, we investigate four approaches for estimating the radius of the KNN search region.

Our first approach, named SUMDIST, was previously proposed in [23]. Briefly, this technique records the location information of k relay nodes that are closest to q along the relay path. Upon receiving the query, the home node combines and sorts these k relay nodes with its neighbor nodes based on the their distance to q. The radius of KNN search region is set as the distance of the kth nearest node to q.

The second approach, called MHD (named MHD-2 in [23]), avoids transmitting the coordinates of k relay nodes towards the home node. Instead, only one maximum hop distance (MHD) value that is calculated as the maximum distance between two hops along the relay path is transmitted. Assuming the home node has m neighbors and $m < k$, the KNN search region is estimated as a circle with radius $(k - m) \times MHD$. Compared with SUMDIST, MHD dramatically reduces the transmission overhead for query propagation.

However the above two approaches are not expected to perform well when k becomes large as the estimated region may quickly expand to the entire sensor network. Therefore, we propose a third approach called NeighborClass. First, the source node selects the members of its NeighborClass as the $Min\{m, k\}$ closest nodes to q (m denotes the number of neighbor nodes). The NeighborClass distance is computed as the furthest distance between the query point and any member of the NeighborClass. These nodes become a new NeighborClass and their IDs and the NeighborClass distance value are passed along with the query packet to the next relay node. Once a relay node receives the query packet, it again selects the $Min\{m, k\}$ closest nodes from its list of neighbors that do not belong to any NeighborClass in the query packet it received from the previous relay. These newly selected node IDs and the maximum distance become a new NeighborClass and are added to the query packet and transmitted to the next forwarding node. When the home node eventually receives the query packet, it includes its NeighborClass and then sets the radius of the KNN search region distance by iterating backwards through the list of NeighborClasses and summing the number of members of each NeighborClass until the sum is larger than k and the KNN search region radius is set as the NeighborClass distance of the last NeighborClass searched. By considering the neighbor nodes of all relay nodes, we expect the KNN search region estimated by NeighborClass to be much smaller than those obtained by SUMDIST and MHD and less affected by increasing k.

However NeighborClass still incurs a significant amount of traffic overhead along the query path since the IDs of the NeighborClass members are transmitted. Therefore, we developed a modified version of NeighborClass, called NeighborClass2 that avoids transmitting the IDs of NeighborClass members along the forwarding path. Each relay node (each having a NeighborClass) only passes the count of member nodes and the maximum distance to the query point q from the furthest member node. This approach cannot guarantee that the radius of the KNN search region will encompass the KNN sensor nodes, but may serve as a good heuristic technique since sensors near the relay route may be double counted (i.e. sensors may be members of more than one NeighborClass). This double counting may be advantageous since it can represent sensors that are inside the KNN search region but are not members of any NeighborClass.

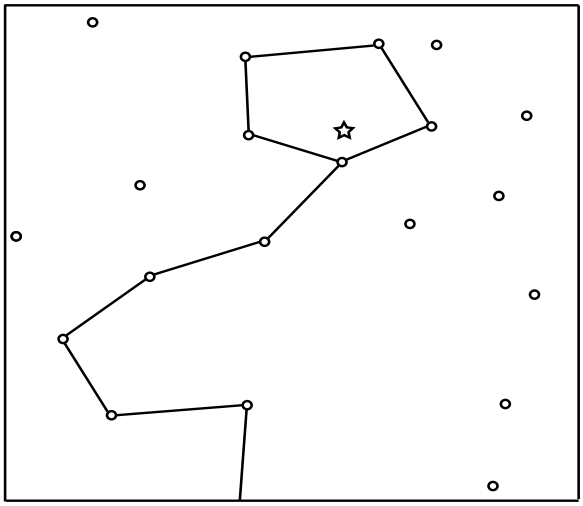


Figure 3. KBT home node and perimeter

After the query packet has reached the candidate home node, it is transmitted around the perimeter of the query point in perimeter mode. This is necessary to verify the home node and we consider it part of the relay path in order to exploit the perimeter nodes and their neighbors for reducing the boundary radius. Figure 3 depicts the routing phase of the KBT KNN algorithm.

4.2 Find $k - 1$ Nearest Neighbors

Given that the query has reached the home node, the next step is to find the remaining $k - 1$ nearest neighbors (the home node is the nearest neighbor). Before we describe the details of the KBT algorithm, we first discuss a naive algorithm which simply floods within the KNN search region. Upon receiving the query, all sensor nodes inside the search region report their readings back to the home node using GPSR algorithm. Although simple, the flooding approach is expected to be efficient for small k since the KNN search region is small. In this case, constructing an infrastructure for disseminating the query and collecting data inside the small KNN search region is unnecessary compared with the cost of flooding. Moreover, flooding reduces the dependence on

any specific nodes inside the search region (other than the home node), thus is more tolerable to individual node failure which makes flooding likely to have good accuracy and robustness. A drawback of flooding is that the query results have to be returned back to the home node individually, thus losing the opportunity for aggregating the results at intermediate nodes.

To address the drawbacks of flooding, we proposed the KBT tree structure which provides the opportunity for local data aggregation. For naming purposes, we refer to this technique as the *single root* (SR) KBT technique since the tree is rooted at the home node. As the query is broadcasted, nodes select their parent based on their geographical proximity. After choosing a parent the child sets a timer based on the difference between an estimated value of the height of the tree and its level in the tree. Once the timer expires, the nodes aggregate results from their children and respond to their parents. The $TreeHeight$, the estimate of the height of the tree, is set before the query is issued. A counter, increased by one at every hop, is passed along with the query and recorded by the internal nodes as their *level*. The timer of the internal node is set as $2 \times Max\{1, TreeHeight - level\} \times MessageDelay$ where $MessageDelay$ is the estimated message propagation delay between two neighbor nodes. The advantage of the tree approach is that results can be aggregated thereby reducing total amount of data transmitted. The drawback is that poorly set timers can either reduce the accuracy of the KNN results or unnecessarily increase the query latency.

A third approach is to use multiple trees rooted at perimeter nodes [9]. The perimeter nodes are determined by routing around the home node with GPSR perimeter mode. We refer to this technique as the *perimeter tree* (PT). The home node still serves as an overall root. The goal of this approach is to attempt to balance the tree to improve query accuracy since the timers are set based on a fixed estimate of the height of the tree. $TreeHeight$ estimates close to the actual tree give better query accuracy. Furthermore, smaller $TreeHeight$ estimates mean shorter latency since the timers expire sooner. When constructing the tree, the root transmits the query back around the perimeter. The circular region is divided into slices defined by the midpoints between the hops as shown in Figure 4. This approach assigns more responsibilities to internal nodes but may be able to improve query accuracy for large values of k. This is because the height of the tree will be smaller than the SR tree since PT trees are more balanced. However, PT has more overhead than SR since an additional transmission around the perimeter is required and the midpoint data must be included in the broadcast query packet.

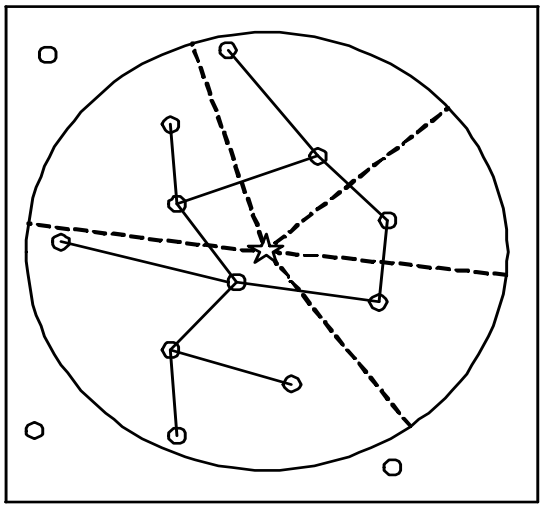


Figure 4. KBT Perimeter Tree

4.3 Return Results

After the home node has received the KNN results, they are aggregated into a single message and returned to the source node using GPSR. If a KBT tree has been constructed inside the KNN search region it is automatically dissolved after the query finishes. In other words, the lifetime of KBT is only as long as the KNN query takes to process.

5 Performance Evaluation

In this section, we explore the performance of both the GRT algorithm and KBT algorithm in terms of the energy consumption, query latency and query accuracy. We first study the different techniques for estimating the KNN search region for the KBT algorithm by varying the value of the application parameter k. We then investigate the impact of varying network conditions and application specifications such as the network density, the rate of node failure, and the mobility of sensor nodes in order to test the sensitivity of the GRT algorithm and KBT algorithm to these factors.

5.1 Simulation Settings

We implemented the GRT and KBT algorithms on CSIM [19] which allows for custom simulation design and supports network scalability. By default, 1000 sensors are deployed uniformly randomly inside a 500m $\times$ 500m region. Each of these sensors has a transmission radius of 40 meters and has approximately 18 neighbors within its transmission

radius. The query point q for a given query is chosen randomly with a k value selected randomly from between 10 and 100. Nodes are by default stationary and nodes do not fail (therefore the cost for beacon messages is not needed and not counted for both GRT and KBT). For each experiment we run five KNN queries back to back. The simulations run until all five queries have returned successfully or have been dropped by a network failure, which typically takes between 10s - 20s. We assume a $MessageDelay$ of 30ms. The experiment results represent the average of 50 trials.

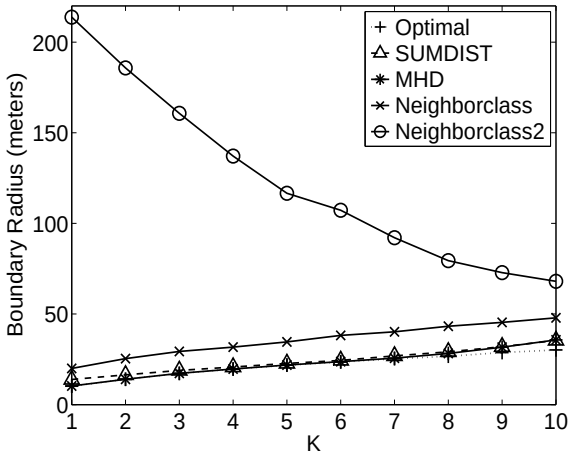
For each experiment we measure the following:

- **Transmission Energy Consumption (Joules):** The total energy consumed for transmitting packets during the simulation time.
- **Query Latency (ms):** The average elapsed time between a query being issued and results being received.
- **Query Accuracy:** The percentage ratio of the number of nodes which are k nearest nodes to q reporting their results over k. The query accuracy for a failed query due to network failures is 0.

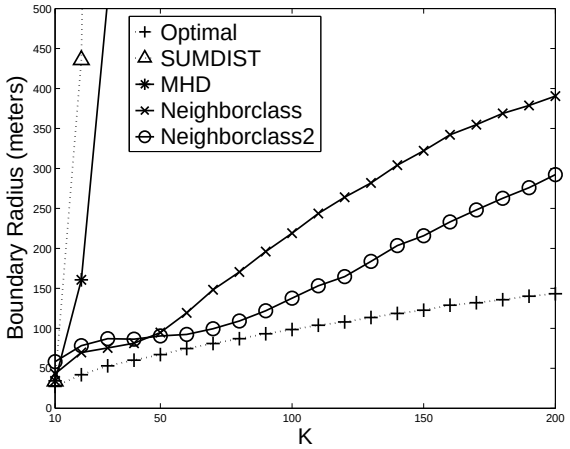
5.2 Estimating KNN Search Boundary

We first study the different heuristics for estimating the KNN search region in the KBT algorithm which determine the size of the estimated region, thus directly impacting the energy efficiency of KBT. For clear comparison we introduce the *optimal* KNN search region which contains k nearest neighbor nodes with a *minimum* radius.

Figure 5 shows the impact of query parameter k (the driving factor on the radius size) on the different proposed boundary estimation techniques where the Y-axis demonstrates the radius of KNN search boundary. Figure 5(a) shows that the SUMDIST and MHD have the best performance for very small k (i.e., $k \leq 10$) and come very close to approaching the optimal boundary radius. However as k increases, the region estimated by SUMDIST and MHD grow radically as shown in Figure 5(b) (note that the X-axis is in logarithmic scale). This is because when k is larger than the number of hops on the query forwarding path, the information collected by both MHD and SUMDIST approach is insufficient for making accurate estimations about the KNN search region. As we expected, the NeighborClass2 boundary method consistently gives boundary values closest to the optimal boundary size for large value of k (i.e., $k > 10$). For the sake of clarity, we only simulate KNN queries with $k > 10$ and employ the NeighborClass2 boundary technique for KBT algorithm in the following experiments since it gives the best performance and has low overhead.



(a) $k < 10$



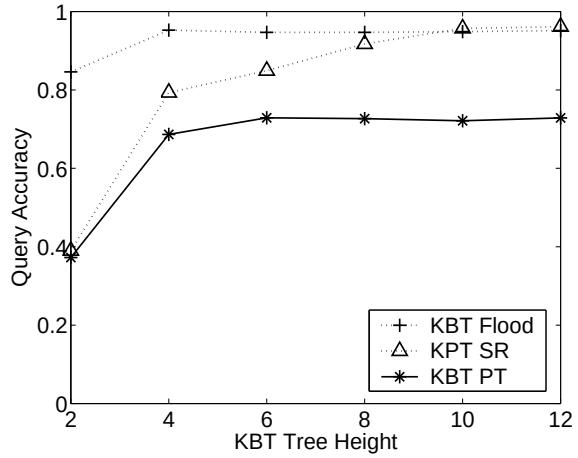
(b) $k > 10$

Figure 5. KBT Boundary Techniques

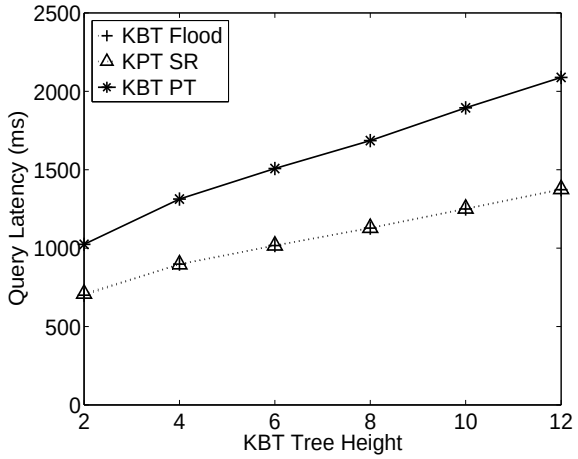
5.3 Impact of KBT TreeHeight

An important tradeoff in KBT design is selecting an appropriate $TreeHeight$ value since the internal nodes of the tree must wait long enough to obtain results from their children without unnecessarily increasing the query latency. The $TreeHeight$ value is an estimate of the height of the KBT tree.

Figure 6(a) shows the effect of varying $TreeHeight$ from 2 to 12 on the KBT query accuracy. The figure shows that the accuracy of all KBT techniques increase with the tree height, as less datum are dropped due to expired timers. Figure 6(b) shows the natural increase in the query latency as the tree height increases. Based on this experiment, we selected a tradeoff value of 6 as the default $TreeHeight$ value.



(a) Query Accuracy



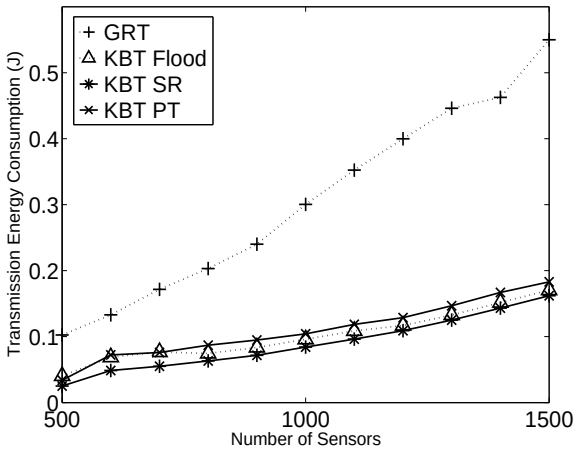
(b) Query Latency

Figure 6. Impact of KBT Tree Height

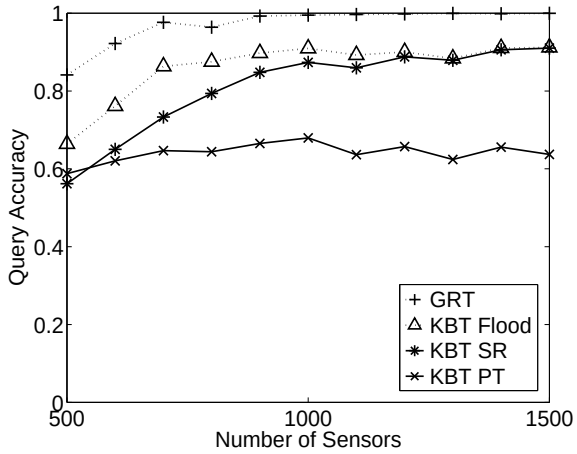
5.4 Impact of Sensor Density

In this experiment we measure the effect of sensor node density on the performance of GRT and KBT by varying the number of sensors inside the fixed region from 500 to 1500 (the average number of neighbors varies from 10 to 25) which is demonstrated in Figure 7.

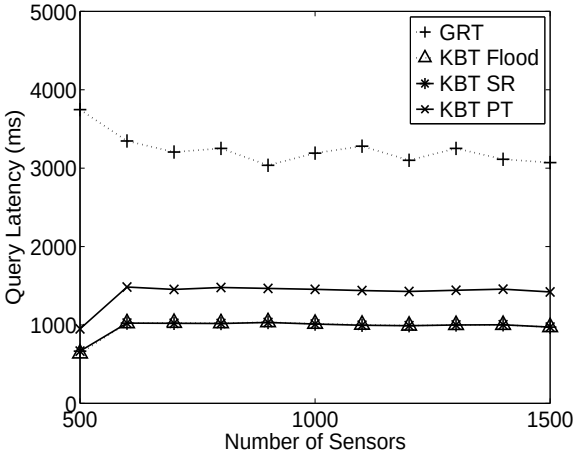
Figure 7(a) shows a linear energy increase as the network becomes denser, as more sensor nodes get involved with query execution for both GRT algorithm and KBT algorithm. The reason that the energy consumption of GRT algorithm increases faster than KBT algorithm is because the number of sensors participating in the query execution and GRT construction cost grows faster. The query accuracy depicted in Figure 7(b) improves as more sensor nodes are deployed since fewer routing failures are likely to happen in a denser network. Moreover, KBT boundary meth-



(a) Transmission Energy Consumption (J)



(b) Query Accuracy



(c) Query Latency

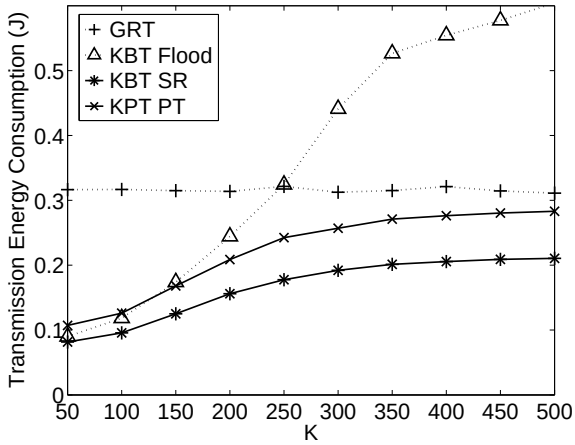
Figure 7. Impact of Sensor Density

ods benefit from high network density, since during query forwarding, additional neighbor information of the relay nodes are collected which give more accurate KNN search region estimates. Figure 7(c) demonstrates the query latency for the different approaches. Although the latency is only slightly affected by the sensor density, it shows that both KBT Flooding and KBT SR have consistently about half the latency of the GRT. This trend is consistent with the query latency of other experiments although they are not presented due to space constraints. Like KBT, GRT internal nodes set timer values based on their tree level. However, since GRT spans the entire network, the GRT $TreeHeight$ is typically larger than KBT's thus producing longer waiting times at GRT internal nodes for propagating the results back to the root node. KBT Flood and KBT SR have the same latency since they use the same routing techniques and timer values. KBT PT has a slightly longer latency due to an extra trip around the perimeter.

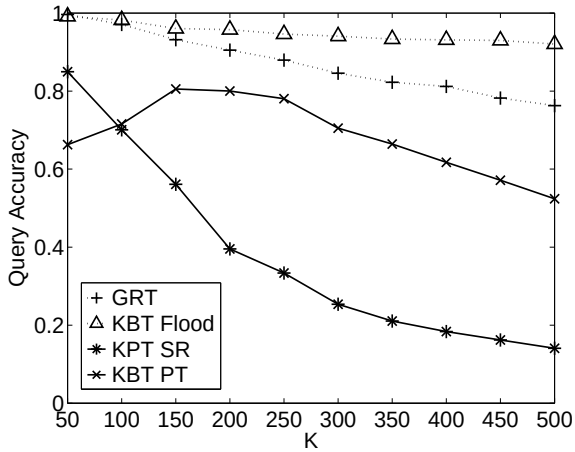
5.5 Impact of k

In this section we investigate the application requirement of k since directly affects the number of nodes involved with the query. Figure 8 varies k from 50 to 500 (5% to 50% of the total number of sensors). Figure 8(a) shows that the transmission energy consumed by the GRT algorithm stays fairly constant while the KBT method increases since the size of KBT search region grows along with k. The KBT Flood energy grows more rapidly since larger KNN search regions will require more multiple hop transmissions for sensors to transmit results back to the home node whereas KBT SR and PT use tree infrastructures that utilize aggregation. In fact, this lack of aggregation and additional relaying cost causes KBT Flood to be worse than GRT for $k > 250$. KBT SR and KBT PT give the best energy performance although they also consistently get worse as k increases, which is natural since the KNN search region increases along with k (see Figure 5). KBR SR has the consistently lowest energy consumption by taking advantage of data aggregation to reduce data transmission and has lower overhead than KBT PT.

Figure 8(b) shows the effect of varying k on query accuracy. KBT Flood gives the best accuracy, followed by GRT, both of which decline slowly as k increases. The slow decline can be attributed to the use of fixed timer values while the number of nodes involved in executing the query increases. The accuracy of KBT SR decreases as k increases since sensors have unrestricted parent selection causing the height of the SR trees grow rapidly as more sensors are included and with fixed $TreeHeight$ values fewer results can return to the parent before the timer expires for large k. KBT PT is less affected by this problem since the division into subtrees limits the height of the tree.



(a) Transmission Energy Consumption (J)

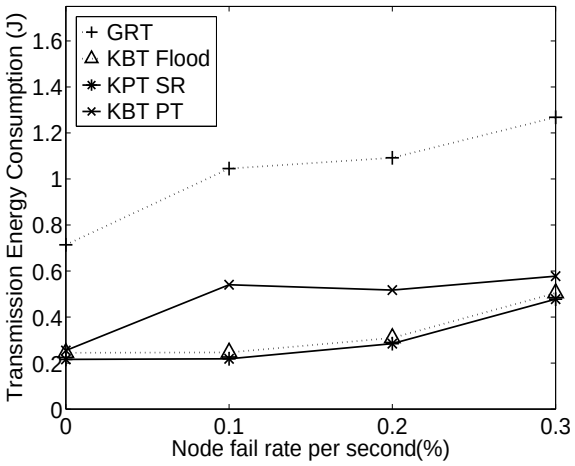


(b) Query Accuracy

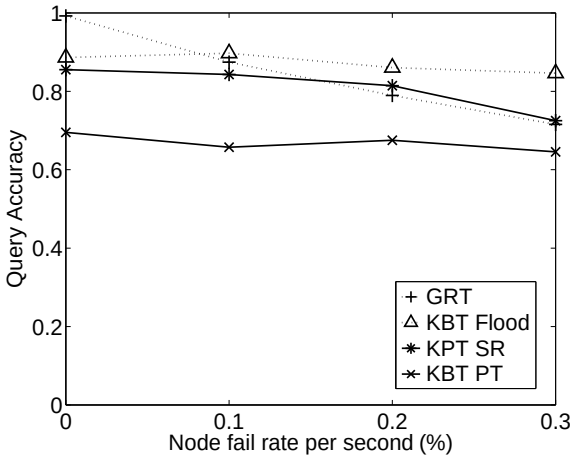
Figure 8. Impact of k (number of expected answers)

5.6 Impact of Sensor Failure

Another property of sensor networks that can affect query processing performance is the rate of failure of sensor nodes. Figure 9 shows the effect of node failure on energy consumption and query accuracy by varying the percentage rate of node failure per second from 0% to .3%. A beacon period (the duration between two beacon messages) of 3 seconds is used for this experiment. The transmission energy consumption depicted in Figure 9(a) increases for GRT since more MBR updates are necessary and more sensors need to find new parents when existing parents fail. The KBT techniques increase slightly since a simple retransmission timer was used to help improve reliability. As expected, the accuracy shown in Figure 9(b) decreases some-

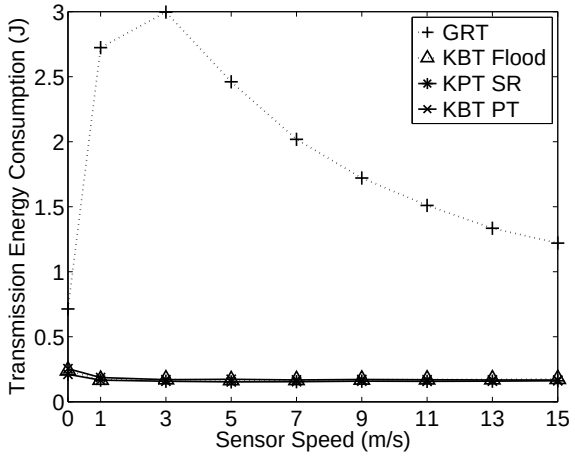


(a) Transmission Energy Consumption (J)

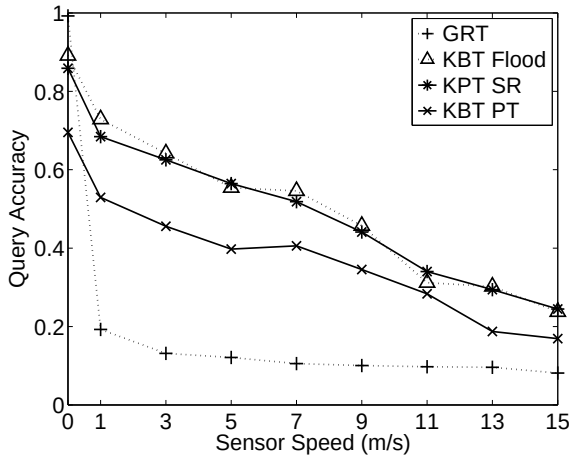


(b) Query Accuracy

Figure 9. Impact of Sensor Failure



(a) Transmission Energy Consumption (J)



(b) Query Accuracy

Figure 10. Impact of Sensor Mobility

what for all techniques although GRT decreases faster since more nodes have responsibilities critical for query success. More specifically if a parent node fails in GRT, the results from its subtree are lost. KBT Flood evenly distributes the responsibility evenly among all nodes inside the KNN search region and thus is least affected by node failure.

5.7 Impact of the Mobility of Sensor Nodes

In this section, we study the impact of sensor movements on KNN algorithms. Sensor mobility is modelled using the Random Waypoint mobility model [1] with a pause time of 0.8 seconds. The node speed varies from 0 to 15 meters per second (m/s). Figure 10(a) shows that the KBT techniques are hardly affected by mobility. However, GRT first dramatically increases in energy consumption and at speeds over 3 meters per second GRT decreases in energy. The GRT increase can be attributed to the additional maintenance cost that occurs as neighbors, children and parents are more likely to move out of range and require updating. As sensor nodes move faster, connections between parent nodes and children nodes are more vulnerable. Therefore more queries and data are dropped before they reach the desired destination due to the disconnection in GRT. In other words, the increasing network dynamics downgrade the query accuracy of the GRT algorithm, however it also reduces the energy consumption as fewer nodes are involved in query execution. The same reasons also apply to the KBT algorithm (hence KBT query accuracy decreases as well), however there are fewer critical transmissions in KBT than GRT. Figure 10(b) depicts how sensor mobility affects query accuracy. This decrease can also be attributed to dropped queries that occur frequently in GRT in mobile sensor networks. Both KBT and GRT have decreased ac-

curacy as sensor speeds increase, but GRT has a more dramatic drop in accuracy. These figures show that GRT is not tolerant to sensor mobility and affirms that maintaining a distributed index structure in a mobile environment is not usually a good idea.

The results presented in this section show that GRT often gives the best query accuracy in static networks, however, the energy and latency are typically worse than the KBT techniques. Furthermore, in dynamic networks the KBT techniques far outperform GRT since the cost of maintaining a distributed index structure is very high and performance is still limited. The KBT Flood, SR and PT techniques all perform differently in different circumstances. KBT Flood is the simplest with the fewest critical nodes and consistently gives the best accuracy and latency but has consistently the highest cost. KBT SR typically gives the best energy performance but trades off with the worst accuracy. KBT PT serves as a hybrid between them by installing some critical nodes in the network that improves the accuracy of KBT SR with less cost than KBT Flood. Other experiments were conducted but not included due to space constraints.

6 Related Work

To our knowledge, KNN queries have not been well explored in wireless sensor networks. However our work has been driven by a wide range of other research efforts on distributed sensor networks.

Distributed sensor networks have been an active research subject. Research efforts have been conducted for building an efficient and robust network infrastructure to support various higher-level protocols and applications. A routing tree [12], similar to our KBT tree, spans the entire network for query executions incurs considerable construction and maintenance cost of the tree structure. Clustering [3, 7] is an alternative network infrastructure where a cluster head is responsible for the operations of the sensor nodes in its cluster. With small-scale centralized control from the cluster head, the cluster infrastructure is suitable for achieving cooperation among sensor nodes. Another well known approach called Directed Diffusion [8] represents an alternative solution for propagating queries and collecting results. A route between the sink and source is built on demand and maintained during the lifetime of the query. However, without considering the geographical nature of sensor networks, this end-to-end route is not beneficial for spatial queries. Other traditional index structures such as the R-tree [6], have demonstrated advantages to researchers in sensor networks. For instance, one work named Peer-tree [2] employs the R-tree in their proposal for sensor networks and discusses the execution plan for various queries upon their modified R-tree. However their work only focuses on one aspect of query execution on the traditional structure and lacks the details needed to handle the unique spatial properties and limitations of sensor networks.

In addition, recent work has pointed out that the location-awareness of nodes can significantly facilitate designing energy-efficient and robust network paradigms and protocols. For instance, Geo-routing protocols [9, 10, 24] have been widely accepted as efficient routing algorithms and have been adopted by many research works (e.g [16, 27]). Other works [25] take advantage of the geographical information of sensor nodes to create a suite of stateless protocols that are especially important for dynamic sensor networks where the network topology changes frequently.

Window queries, another essential spatial query also attracts increasing research attention. GEAR [27] focuses on spatial window queries and considers the issue of how to diffuse the query inside the window so that all the sensor nodes residing within the window receive the query. Even though GEAR does not require any network infrastructure for spreading the query, it does not consider how to collect results without the infrastructure, and therefore it is hard to evaluate the execution cost for window queries. Another work aimed at dynamic sensor networks that made progress on this problem proposed a complete set of query execution plans including forwarding the query toward the query window, propagating the query inside the window, collecting the results and eventually returning the result to the application [25].

7 Conclusion and Future Work

In this paper we presented the novel problem of KNN queries in sensor networks and proposed two KNN algorithms that represent two different design philosophies of sensor network algorithms to address the research issues with this problem. Our extensive performance evaluation of both techniques showed that both KBT and GRT showed promise and defined the environment and circumstances under which each performed the best. GRT had a good trade-off between energy and query accuracy while KBT had far better query latency and energy consumption. In dynamic networks, KBT was far more robust due to minimized sensor responsibilities.

For our future work we will investigate the use of KBT for window queries in dynamic sensor networks. We also plan to investigate the effect of overlapping MBRs on the effect of KNN query processing on the GRT. We also intend to analyze the advantages and drawbacks of processing KNN queries on subtrees in parallel versus sequentially. Adding retransmission timers to both GRT and KBT could easily make communication more reliable. Finally, we intend to support KNN queries for sensor networks that can locate and sample the k nearest events in a sensor network.

8 References

[1] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[2] M. Demirbas and H. Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing*, Linkping, Sweden, Sept. 2003.

[3] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263–270, Seattle, WA, Aug. 1999.

[4] D. Goldin, M. Song, A. Kutlu, H. Gao, and H. Dave. Georouting and delta-gathering: Efficient data propagation techniques for geosensor networks. In *Proceedings of NSF Workshop on GeoSensor Networks*, Portland, ME, Oct. 2003.

[5] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A distributed index for features in sensor networks. In *Proceedings of the IEEE ICC Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, Apr. 2003.

[6] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD*, pages 47–57, Boston, MA, 1984.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 8020–8029, Maui, HI, Jan. 2000.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking*, pages 56–67, Boston, MA, Aug. 2000.

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

[10] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wirel. Netw.*, 6(4):307–321, 2000.

[11] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2), 2002.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[13] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the ACM SIGMOD*, San Diego, CA, Jun. 2003.

[14] D. Papadias, J. Zhang, N. Mamounis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the Very Large Databases Conferences*, Berlin, Germany, Sept. 2003.

[15] H. Qi, X. Wang, S.S. Iyengar, and K. Chakrabarty. High performance sensor integration in distributed sensor networks using mobile agents. *International Journal of High Performance Computer Applications*, 16(3):325–335, 2002.

[16] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *ACM/Kluwer Journal of Mobile Networks and Applications*, 8(4):427–442, 2003.

[17] 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 International Workshop on Sensor Networks and Applications*, pages 78–87, Atlanta, GA, Sept. 2002.

[18] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of ACM SIGMOD*, pages 71–79, San Jose, CA, 1995.

[19] H. Schwetman. *CSIM user's guide (version 18)*. Mesquite Software, Inc., http://www.mesquite.com, 1998.

[20] T. Seidl and H. Kriegel. Optimal multi-step k-nearest neighbor search. In *Proceedings of ACM SIGMOD*, pages 154–165, Seattle, WA, 1998.

[21] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *ACM SIGCOMM Computer Communication Review*, 33(1):137–142, 2003.

[22] Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *Proceedings of International Symposium on Spatial and Temporal Databases*, pages 79–96, Los Angeles, CA, Jul. 2001.

[23] J. Winter and W.C. Lee. KPT: A dynamic KNN query processing algorithm for location-aware sensor networks. In *Proceedings of International Workshop on Data Management for Sensor Networks*, pages 119–125, Toronto, Canada, 2004.

[24] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell. PSGR: Priority-based stateless geo-routing in highly dynamic sensor networks. Technical Report CSE 04-021, Pennsylvania State University, Sept. 2004. http://www.cse.psu.edu/pda/SDB/pubs/PSGR_TR.pdf.

[25] Y. Xu and W.C. Lee. Window query processing in highly dynamic geo-sensor networks: Issues and solutions. In *Proceedings of NSF Workshop on GeoSensor Networks*, Portland, ME, Oct. 2003.

[26] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proceedings of Conference on Innovative Data Systems Research*, Asilomar, CA, Jan. 2003.

[27] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, May 2001.