Processing Window Queries in Wireless Sensor Networks

Yingqi Xu* Wang-Chien Lee *
Pennsylvania State University
{yixu, wlee}@cse.psu.edu

Jianliang Xu†
Hong Kong Baptist University
xujl@comp.hkbu.edu.hk

Gail Mitchell
BBN Technologies
gmitchell@bbn.com

Abstract

The existing query processing techniques for sensor networks rely on a network infrastructure for query propagation and data collection. However, such an infrastructure is very susceptible to network topology transients that widely exist in sensor networks. In this paper, we propose an infrastructure-free window query processing technique for sensor networks, called itinerary-based window query execution *(IWQE), in which query propagation and data collection are combined into one single stage and executed along a well-designed itinerary inside a query window. We study the parameters for setting up an itinerary (e.g.,* width *and* route*) and incorporate into IWQE three data collection schemes based on different performance trade-offs. Finally we demonstrate, by extensive simulations, the superior energy-time efficiency, robustness, and accuracy of IWQE over the current state-of-the-art techniques in supporting window queries under various network conditions.*

1 Introduction

Wireless sensor networks are revolutionizing the ways of collecting information from the physical world [3]. Due to unattended and untethered node deployments, most sensor network-based applications specify their interests using geographical predicates (i.e., spatial queries). Window query, one of the most important types of spatial queries, specifies a geographical region (e.g., a two-dimensional rectangular window) for retrieval of the sensed data from the nodes residing in the specified region. Window queries are often combined with aggregation functions (e.g., MAX, AVG, COUNT, etc.) to support analysis of complex phenomena. An example is "retrieving the average temperature from region w_1".

The window query processing techniques in traditional database research typically assume data are available in a centralized database and focus on improving the disk-access performance. However, collecting all sensor readings into a database is infeasible for a sensor network given its scarce resources (including energy, bandwidth, storage and processing). Recently, a number of studies [6, 10, 14, 15, 19, 21] have explored distributed query execution based on in-network data storage. They typically split a query execution into two stages: *query propagation* in which a query is propagated to the sensor nodes along some hierarchical infrastructure (e.g., a tree or a cluster), and *data collection* in which data are aggregated (usually in a reversed order) along the same infrastructure. Counting on a network infrastructure, these query processing techniques are classified as *infrastructure-based query execution*. Such techniques are susceptible to network topology changes, yet change is the norm (due to node movement or transient switching between active and sleeping modes for power saving) rather than the exception in sensor networks [25]. Maintaining a stable infrastructure in a dynamic network incurs excessive update messages and can cause message collisions, packet losses and transmission delays. The network dynamics deteriorate the performance of infrastructure-based query processing techniques and shorten the lifetime of a sensor network. These deficiencies call for novel techniques for window query processing that are not dependent on any network infrastructure.

In this paper, we propose such an infrastructure-free query processing technique for wireless sensor networks, called *Itinerary-based Window Query Execution* (IWQE). In IWQE, data are aggregated with query propagation along a well-designed itinerary. Specifically, starting from an initial sensor node in the query window, a query is propagated to the next node on the itinerary after collecting data from a current sub-region inside the window. This procedure is repeated until the last node on the itinerary is reached, from which the result is returned to the sink node. In contrast to the infrastructure-based techniques, IWQE does not maintain any network infrastructure and hence improves energy efficiency. It also reduces the query latency by combining data collection with query propagation.

Many unique and challenging research issues arise in IWQE, such as itinerary settings (i.e., width and route), query window coverage, in-network data processing, and handling of packet losses. This paper provides a thorough study of these research issues. More specifically, we make the following contributions in this paper:

*Wang-Chien Lee and Yingqi Xu were supported in part by National Science Foundation grant IIS-0328881.

†Jianliang Xu's work was partially supported by grants from the Research Grants Council of the Hong Kong SAR, China (Project No. HKBU 2115/05E and HKBU FRG/04-05/II-26).

- We introduce the IWQE which, to our best knowledge, is the first infrastructure-free window query processing technique for wireless sensor networks. IWQE, without relying on any infrastructure, performs query propagation and data collection in an integrated manner, thus improving the network's energy efficiency.
- The study reveals profound insights into guidance for designing an energy-aware and robust IWQE. Two key parameters for setting up an itinerary, namely *itinerary width* and *itinerary route*, are studied. The *maximum* itinerary width that ensures a full coverage of a query window is mathematically derived. Three itinerary routes with different design goals are explored.
- In order to avoid message collisions during data collection, three data collection schedules are developed based on the tradeoff between energy efficiency and query latency.
- We evaluate, with an extensive simulation study, the performance of IWQE against three existing query processing techniques. IWQE exhibits a superior performance in terms of *energy efficiency*, *query latency* and *query accuracy* under various network conditions. Indeed, IWQE soundly outperforms all compared protocols.

The rest of the paper is structured as follows. We review the related research work in Section 2. Section 3 presents the basic design of IWQE and Section 4 further explores IWQE by taking into account adversary network conditions. We report our performance evaluation and simulation results in Section 5. Finally, Section 6 concludes the paper and discusses the future work.

2 Related Work

IWQE, to our best knowledge, is the first infrastructure-free window query processing technique for sensor networks. Nevertheless, our work is inspired by a number of related research efforts.

In the existing solutions, a window query is typically processed based upon an infrastructure (e.g., a tree or a cluster based structure [6, 14, 19, 21]) by two steps. First, a query is propagated toward the specified query window. After the query covers the query window, data collection takes place along the infrastructure. The sensor nodes falling into the query window recursively report their readings to the nodes from which they received the query, such that partial results are returned level-by-level up the infrastructure until reaching the root node. Based on the *size* of the infrastructure, this approach can be further divided into *network spanning infrastructure* (NSI) and *window spanning infrastructure* (WSI). Examples of one NSI and two WSIs are shown in Figures 1(a) - 1(c).

An NSI is usually built when deploying the network and is maintained during the entire network lifetime [6, 14]. Due to its large scale, NSI incurs significant construction and maintenance overhead. For window query, NSI involves many irrelevant sensor nodes outside the query window for query processing. Moreover, NSI initiates the query processing from the root node (e.g., tree's root, or cluster head), such that queries inserted from other nodes have to be first routed to the root node, which costs extra resources. On the other hand, in WSI, the query is first forwarded to the query window by an end-to-end routing (e.g., geo-routing) protocol. Once the query reaches the window, an infrastructure within the query window is built along with the query propagation [19, 21]. By constructing an infrastructure for each query, the WSI approach also incurs noticeable overhead. Moreover, the query latency is long due to the two-staged query execution.

Geo-routing protocols (e.g., [11, 12, 22]) have been widely adopted in sensor network design (e.g., [13, 20]). However, as pointed out in [24], geo-routing, crafted for end-to-end routing, is not efficient for query propagation/data collection inside a region.

The itinerary-based idea, although not yet adopted by existing query processing techniques, has appeared in unicast routing [17], data fusion [18] and network surveillance [7] for sensor networks. Different from those studies, our proposal explores the nature of an itinerary (in terms of itinerary width and itinerary route), and investigates research issues in query propagation and data collection, such as in-network processing and robustness to packet losses.

Many techniques designed for ad hoc and sensor networks can complement IWQE. For example, approximate in-network data aggregation [4, 16] achieves a satisfactory query accuracy with a reasonable energy consumption for duplicate-sensitive queries. Such techniques can be used to enhance IWQE's resilience to packet losses, as will be discussed in Section 4.2.

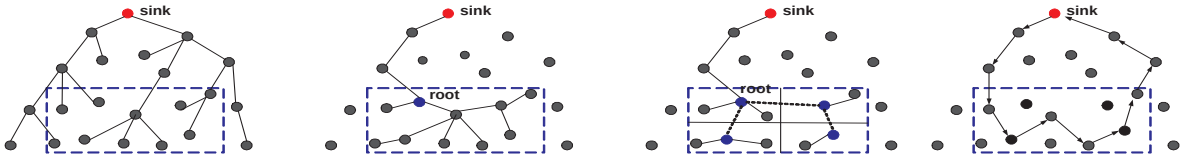
Finally, our definition of query accuracy for window queries, which provides a strict yet intuitive criterion for measuring the robustness of a query processing technique to dynamic network conditions, is inspired by [2].

3 Design of IWQE

In this section, we first describe the assumptions used in our study, then present the basic idea of IWQE, and finally discuss the detailed design of IWQE.

3.1 Assumptions

A window query retrieves the sensed data from the sensor nodes falling within a *query window*, which is a spatial area of interest specified by the user. In this paper, for clarity of the presentation, we assume the query window is shaped as a two-dimensional rectangle; the proposed techniques can be extended for other window shapes and for three-dimensional space. Thus, a window query is defined by a *spatial predicate* that specifies the location and size of the query window, an *operational function* that specifies the operation over collected sensor readings, and a *query lifetime* T during which the query is valid. In this paper, we consider only snapshot queries, which expect to obtain the query result only once during its lifetime. The sensor nodes are location-aware. The network could be dynamic in that



(a) NSI: tree (b) WSI: tree (c) WSI: cluster (d) IWQE

Figure 1. Window query processing

the sensor nodes may move inside the network and/or may occasionally enter into a sleep mode to save power. Each node maintains a neighbor table via periodic exchange of beacon messages.

3.2 Basic Idea of IWQE

As we pointed out, an infrastructure-based technique is vulnerable to network dynamics and thus we examine infrastructure-free window query processing techniques. A *naive* infrastructure-free approach is to flood the query to all sensor nodes inside the network. The nodes inside the query window, upon receiving the query, report their readings to the sink node by an end-to-end routing (e.g., geo-routing) protocol. Although not affected by network topology changes, the naive approach consumes excessive network resources for query processing and does not support in-network aggregation.

IWQE incorporates in-network data aggregation while providing robust and effective query processing under transient network topologies. To minimize the number of unnecessary node visits, IWQE forwards the query toward the query window by a geo-routing protocol (shown in Figure 1(d)). Inside the query window, a set of sensor nodes called *Query nodes* (i.e., Q-nodes) are chosen for query dissemination. Once receiving a query, a Q-node broadcasts a *probe* message that includes the query and information about the itinerary (e.g., itinerary width and itinerary route). Upon hearing the probe message, the neighbor nodes that are qualified to reply the query, called *Data nodes* (i.e., D-nodes), report their sensed data back to the Q-node. After aggregating the data from all D-nodes and the partial result received from the previous Q-node, the current Q-node selects the next Q-node based on the itinerary and a query forwarding heuristic, and forwards this new partial query result to the selected next Q-node. After the query traverses the entire query window, the aggregated result is returned back to the sink node, again by a geo-routing protocol. As can be seen, by performing data collection along with query propagation at each Q-node, IWQE does not need any static network infrastructure.

It is a challenge to design such an infrastructure-free window query processing technique to be robust, yet energy efficient. Since a query is executed along an itinerary, the itinerary is extremely important to the performance; thus the key parameters that determine an itinerary need an in-depth study. Moreover, data collection from multiple D-nodes needs to be scheduled to avoid excessive message collisions and processing delays. The remainder of this section addresses these issues in detail. Other research challenges, including low network density and packet losses, will be discussed in Section 4.

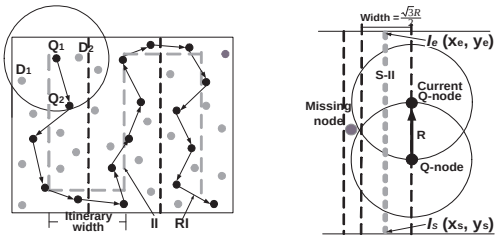
3.3 Itinerary

Intuitively, an itinerary is one or a set of curves that pass through the query window. As the network topology is not known a priori, the itinerary formed by the Q-nodes (called *real itinerary* (RI)) may not exactly match a planned itinerary (called *ideal itinerary* (II)). Figure 2(a) shows an example of IWQE, in which node Q_1 and Q_2 are two adjacent Q-nodes and D_1 and D_2 are the D-nodes for Q_1. All the arrows connecting the black nodes (i.e., the Q-nodes) form the RI and the grey dashed plotline is the II. An II (or RI) can be further split into a set of parallel *sub ideal itineraries* (S-II's) (or *sub real itineraries* (S-RI's)). The distance between two adjacent S-II's is defined by an *itinerary width* or simply *width*. Furthermore, an *S-II zone* is defined as the region with an S-II as the center line and the *itinerary width* as the width. Figure 2(a) shows three S-II zones; they are separated by two black dashed lines.

Given an S-II, IWQE aims to collect all sensor readings falling into the corresponding S-II zone, such that the query window is fully *covered* (i.e., all the sensor readings inside the window are collected) when all S-II's are traversed. However, the RI may deviate from the II. Without proper control of the deviation, the RI may not be able to cover the entire query window, thus hurting IWQE's query accuracy. This problem can be addressed by carefully planning the itinerary. In the following, by studying two key design parameters for planning an itinerary, *itinerary width* and *itinerary route*, we demonstrate this full window coverage can be achieved. Itinerary width reflects the closeness between adjacent S-II's. Generally speaking, a smaller itinerary width results in denser itineraries and smaller S-II zones, thus is more tolerant of the deviation of RI from II and favorable to improve the query accuracy. On the other hand, a small S-II zone incurs unnecessary transmissions and message collisions, which obviously defeat the goal of energy efficiency. The itinerary route represents the shape of the query processing itinerary, which has an impact on energy efficiency and query latency.

3.3.1 Itinerary Width

As we argued earlier, the II has to be designed in such a way that a query propagated along the RI reaches every sensor node in the query window at least once. Although with this requirement some sensor nodes may receive the query more than once, the redundancy provides a full coverage



(a) Itinerary Route (b) Itinerary Width

Figure 2. IWQE Itinerary

of the query window. In this section, we derive the *maximum itinerary width* (i.e., MIW) that ensures query window coverage. While MIW is not necessarily the optimal choice of itinerary width, it provides important guidance for balancing the tradeoff between energy efficiency and query accuracy.

In order to derive MIW, we assume the network is dense enough such that Q-nodes that make geographical progress on the itinerary are always available. The sensor network with reduced node density is considered in Section 4.1.

Theorem: *Let R denote the transmission range of the sensor nodes. In order to guarantee full coverage of a query window, the itinerary width must be less than $\frac{\sqrt{3}R}{2}$ (i.e., $MIW = \frac{\sqrt{3}R}{2}$).*

Proof: We prove the above theorem by showing: 1) when itinerary width equals MIW, all sensor nodes can be covered; 2) when itinerary width is larger than MIW, a node may not be covered.

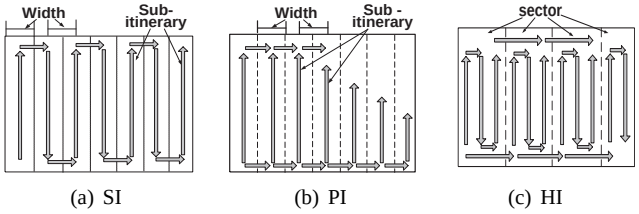
We assume that inside a rectangular S-II zone, there are a total of n Q-nodes for query propagation. Denote these Q-nodes as Q_i (with coordinates (x_i, y_i)), where $1 \leq i \leq n$. Without loss of generality, we assume that S-II is a vertical straight line; thus each forwarding progress is made along the y-axis. Assuming the query is propagated in the order of $Q_1, Q_2, \cdots, Q_n$, we have $y_i < y_{i+1}$ and $|Q_i - Q_{i+1}| = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \leq R$, where $1 \leq i \leq n-1$. We also assume the S-II starts at I_s ((x_s, y_s)) and ends at I_e ((x_e, y_e)), where I_s and I_e are on the boundary of the query window (see Figure 2(b)). To ensure the coverage of the S-II zone, we must have $|Q_1 - I_s| \leq \frac{R}{2}$ and $|Q_n - I_e| \leq \frac{R}{2}$.

Now suppose there exists a node D located at (x_d, y_d) within the current S-II zone that does not hear the query from any Q_i ($1 \leq i \leq n$) when the query is propagated. Given the itinerary width $\frac{\sqrt{3}R}{2}$, the following property holds:

$$|x_d - x_i| < \frac{\sqrt{3}R}{2}.$$

Since the node D does not receive the query from any Q_i, where $1 \leq i \leq n$, we have $\sqrt{(x_d - x_i)^2 + (y_d - y_i)^2} > R$. Combining the above two equations, we obtain for any $1 \leq i \leq n$:

$$|y_d - y_i| > \frac{R}{2}. \quad (1)$$



(a) SI (b) PI (c) HI

Figure 3. Itinerary Route and Width

Without loss of generality, consider Q_j ($1 \leq j < n$) whose distance to the node D satisfies $\frac{R}{2} < y_d - y_j < R$. Since $y_{j+1} - y_j \leq R$, $|y_{j+1} - y_d| < \frac{R}{2}$ is true, which is contradictory to Equation (1). Finally, suppose that Q_n satisfies $y_d - y_n > \frac{R}{2}$. Since $|Q_n - I_e| \leq \frac{R}{2}$, the node D outside the query window does not need to be covered. Therefore, we have proved when the itinerary width equals MIW, all sensor nodes are covered.

Meanwhile, for the case that itinerary width is larger than MIW (i.e., $\frac{\sqrt{3}R}{2}$), Figure 2(b) shows a simple example in which a node inside the query window (the grey node in the figure) could be missed. □

As noted earlier, MIW is not necessarily the optimal width, but is a conservative setting to ensure the window coverage. An application can increase the itinerary width to achieve a higher energy efficiency at the cost of a possibly lower query accuracy, and vice versa. We further study this tradeoff in our performance evaluation in Section 5.

3.3.2 Itinerary Route

This section proposes three different routes along which IWQE processes the query, namely *sequential itinerary* (SI), *parallel itinerary* (PI) and *hybrid itinerary* (HI), aiming at energy optimization, query latency reduction and a balance of those, respectively.

The sequential itinerary (SI) implies that the query is sequentially propagated along one itinerary (see Figure 3(a)). The sequential query execution minimizes the communication collisions since, at any time, only one copy of a query is processed within the window. However, with a single itinerary, the query execution along SI could suffer from a long query latency, especially for a large query window.

In order to speed up query execution, the parallel itinerary (PI) parallelizes the query propagation inside the query window. Figure 3(b) shows an example of PI, in which multiple threads of the query are spread horizontally and then propagated vertically along S-II's; the query can be processed along multiple parallel S-II's at the same time. The partial query result collected from a S-II zone is forwarded to its adjacent S-II zone (see Figure 3(b)), where the results from both S-II zones are further aggregated. Eventually, the final result is collected by the last Q-node on the *rightmost* S-II. The PI reduces the query processing delay by allowing several query executions to take place concurrently. However, as multiple copies of a query are processed simultaneously, communications in adjacent S-II zones may collide with each other. Thus, PI could require more energy consumption than SI.

In order to avoid the transmission collisions while maintaining a low query latency, we combine PI with SI to form a hybrid itinerary (HI). More specifically, HI divides the query window into several sectors as shown in Figure 3(c). Inside each sector, a query is executed along a sequential itinerary. Meanwhile, by forwarding the query to other sectors, the query execution is parallelized among multiple sectors. Therefore, the size of a sector determines the concurrency degree of query execution and the possibility for transmission collisions caused by the concurrent query executions in different sectors. If HI uses fewer sectors with larger sizes, the SI in each sector is longer, which results in a longer query latency. On the other hand, if the sector size is too small, the query executions in two different sectors are more likely to interfere each other. Thus, the sector size should be carefully selected to avoid the transmission collisions; its width should be at least *R + itinerary width*, where R is the transmission range of sensor nodes. In this way, even if S-RI departs from the S-II by the maximum deviation (i.e., half the itinerary width), the S-RI's in two adjacent sectors are still R away from each other, thus not causing interference.

3.4 Itinerary Traversal

Based upon the planned itinerary, this section describes the detailed query processing algorithm, including data collection schedule and query forwarding scheme.

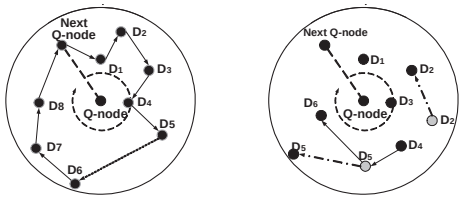
3.4.1 Data Collection

In IWQE, a query issued by a sink node is first forwarded toward the query window by a geo-routing protocol (e.g., GPSR [11]). For simplicity, we assume the query forwarding is destined to the *corner* of the query window that is closest to the sink. Since a node is not necessarily located at the corner, with a geo-routing protocol, the query is actually forwarded to the node (i.e., the first Q-node) inside the query window that is closest to the corner.

Upon receiving a query, the current Q-node broadcasts a *probe message*. The neighbor nodes that are inside the current S-II zone but have not received the query before (i.e., the D-nodes) report their readings to the Q-node. This requires each sensor node to track the queries it has received such that no duplicated queries are processed. Since each Q-node probably has more than one D-node, a schedule is needed for these D-nodes to avoid collisions when they respond to the Q-node. The schedule is supposed to set a total order among all these D-nodes, i.e., during a given time period, only one D-node reports its readings to the Q-node. The precedence of a D-node is determined by the following rule. A line connecting the current Q-node and a geographical point (randomly chosen by the Q-node and specified in the probe message) rotates clockwise; the D-node hit by the line earlier has a higher precedence. We study three different implementation schemes based on this idea.

The first scheme is a *TDMA-based scheme* in which the Q-node obtains a schedule by sequentializing all the D-nodes in its neighbor table. This schedule is broadcast with the probe message. Upon receiving the probe, a D-node decides its reply time based on the assigned precedence and transmission delay. This scheme ensures a collision-free data collection, but incurs transmission overhead for broadcasting the schedule. Moreover, this scheme is not robust to network topology changes. Even though beacon messages are periodically exchanged between nodes to update the cached information, the neighbor states cached at Q-nodes could be still out-of-date.

The second scheme, *token ring-based scheme* employing a *virtual* token, is designed to remove the schedule broadcast in the TDMA-based scheme. As Figure 4(a) shows, all D-nodes, based on their reply precedences, form a ring. Although not knowing its *absolute* reply precedence, a D-node, based on its neighbor table, can derive its *relative position* on the ring, and figure out the D-node that should reply immediately before itself. Therefore, a reply message from one D-node is a virtual token for the next D-node that should reply. For instance in Figure 4(a), a reply message from D_3, which is immediately in front of D_4 on the ring, is the token for D_4 to reply to the Q-node. However, there are several problems with this scheme. First, the ring could be disconnected due to limited transmission ranges of sensor nodes. In Figure 4(a), D_6 cannot receive the token from D_5, since it is outside D_5's transmission range. Second, the ring could be disordered due to the staleness of cached neighbor states. Figure 4(b) shows that D_3 stores the location of D_2 as the grey node; however upon receiving the probe, D_2 has moved to the new location shown as the black node. Therefore, both D_3 and D_2 believe they should reply after D_1, which causes a transmission collision. Third, the token could be missing due to transmission collisions, node dynamics and node failures. In Figure 4(b), the token from D_4 to D_5 is missing since D_5 moves out of D_4's transmission range during the data collection. All the above problems can be resolved by getting the current Q-node involved. For example, by issuing another token, the Q-node re-initiates the interrupted data collection. However, the involvement of Q-nodes increases the algorithm complexity and transmission overhead.



(a) Scenario I (b) Scenario II

Figure 4. Token Ring-Based Data Collection

The third solution, a *contention-based scheme*, solves the problems associated with the TDMA- and token ring-based schemes. In this scheme, each D-node determines its reply precedence independently without assistance from the Q-node or knowledge about its neighbor nodes. Each D-node, upon receiving the probe message that contains

a reference line emanating from the current Q-node, sets a timer, which can be calculated by a simple heuristic: $timer = max\_delay \times (\frac{\alpha}{2\pi})$, where α is the angle formed by the specified reference line and the line connecting the current Q-node and the current D-node, and max_delay is the maximum time that the Q-node is allowed to complete its data collection. A D-node does not respond to the Q-node until its timer expires. Moreover, to prevent two D-nodes from having the same timer, a small jitter is added to each timer, which has been proved to be a simple yet effective technique by many communication protocols in sensor networks [10, 11].

3.4.2 Query Forwarding

After the current Q-node receives the data from all its D-nodes, it calculates a partially aggregated query result and selects a next Q-node to forward the result. We place two requirements on the selection of the next Q-node. First, in order to proceed with query processing, the next Q-node should make geographical progress along the S-II. Second, the next Q-node must reside inside the current S-II zone, which constrains the contour of RI and ensures the coverage of query propagation. The above two conditions can be determined purely based on the Q-node's neighbor table (with cached neighbors' locations) and the planned itinerary route and width. Among all sensor nodes that satisfy the above two requirements, the next Q-node can be determined by some heuristic. In this study, we consider: 1) Closest to Itinerary (CTI): in order to minimize the deviation between RI and II, the node closest to the II is chosen as the Q-node, which could reduce the potential communication collisions between adjacent S-RI's for IWQE with PI and HI routes; 2) Most Progress on Itinerary (MPI): in order to minimize the total number of Q-nodes and reduce the query propagation delay, the node that maximizes the spatial progress of query propagation on the itinerary is chosen as the Q-node.

4 Additional Research Issues

Section 3 discussed the basic design of IWQE with fair network density and lossless network communications. In this section, we investigate how IWQE performs without jeopardizing the performance when the ideal network conditions are not met.

4.1 Itinerary Void

In the previous sections, we assumed that the sensor network is dense enough that the next Q-node can always be found for query forwarding. However, considering such obstacles as non-uniform node distribution and low-connectivity areas, this assumption is not always valid. The above problems, generalized as *itinerary void*, can impact the performance of IWQE.

We adapt the localized perimeter forwarding algorithm developed by GPSR [11][1] to handle the itinerary void. The algorithm operates on a planar graph, which eliminates the crossing edges. When a Q-node, now called *stuck node*, encounters the itinerary void, the query propagation switches to a perimeter forwarding mode that forwards the query on the progressively closer face of the planar graph; each of such faces is crossed by the S-II. On each face, the relay node, called B-node, is selected based on the *right-hand rule*, a long-known graph traversal that walks a cycle in a graph. The stop condition of perimeter forwarding is modified as follows: if a B-node is inside the S-II zone and makes progress on the itinerary over the stuck node, the perimeter forwarding ends and query forwarding is resumed. In the case the itinerary void extends outside the query window (i.e., the B-node is outside the query window), the perimeter forwarding is stopped and the query is forwarded to the adjacent S-II zone. During the perimeter forwarding, data collection is not performed since all B-nodes are either outside the current S-II or have been queried.

[1] Similar algorithms can also be found in [5, 12].

4.2 Robustness to Packet Losses

Considering the harsh and dynamic sensor network environments, packet losses frequently happen. However, retransmitting each lost packet expends energy, while delaying the query response. IWQE takes advantage of the wireless broadcast medium to avoid some of the retransmissions without hurting query accuracy.

In essence, we only allow retransmissions between Q-nodes; the messages for data collection between a Q-node and D-nodes are not retransmitted in the case of packet losses. When a D-node replies its data to the Q-node, all its neighbor nodes can overhear the packet. The neighbor nodes, instead of ignoring the packet, cache the data and aggregate it partially with their own readings. Thus, the data sent by a D-node is the aggregated partial result from multiple sensor nodes. As such, when data from a D-node to the Q-node is lost, its reading can still be collected from other D-nodes which cached and aggregated the reading, thus greatly improving the robustness to the packet losses.

For duplicate-insensitive queries (e.g., MAX and MIN), the above robust approach is able to improve IWQE's resilience to packet losses without deteriorating the energy efficiency. However, this approach may lead to errors for duplicate-sensitive queries (e.g., SUM, COUNT and AVG). Fortunately, recent research work [2, 4, 16] has proposed several algorithms which, with minor energy overhead, are able to acquire the *approximately* correct query results without double-counting the duplicate (redundant) sensor readings. In this paper, we focus on evaluation of duplicate-insensitive queries, and will report the performance for duplicate-sensitive queries in future work.

5 Performance Evaluation

We have developed a simulator based on ns-2 [1] (with the CMU wireless extensions). In this section, we study the impact of different protocol design choices (i.e., itinerary width, itinerary route and data collection method) on the performance of IWQE and examine IWQE's sensitivity to various system parameters, such as window size, node mobility, and packet loss rate.

Parameters	Values	Parameters	Values
R (m)	40	Number of nodes	100
Window size (m^2)	90×90	Number of trials	25
Network size (m^2)	160×160	V_{max} (m/s)	15
Packet size(bytes)	32	Pause time (s)	0
Number of queries	40	Node degree	15
Sector width in HI (m)	69.2	Beacon interval (s)	1.0

Table 1. Default Parameter Settings

5.1 Metrics and Settings

Similarly to [10, 11], we implemented IWQE atop the ns-2 802.11 MAC layer. Initially, the sensor nodes are randomly distributed in the simulated field. The mobility of sensor nodes is modeled by the *random waypoint* (RWP) model: the node selects a destination at random in the field, and moves to the destination at a speed randomly chosen from a configured range $[0, V_{max}]$; upon arrival, it pauses for a random period and selects a new destination. Because the original RWP model fails to achieve a stable mobility pattern, our simulator adopts an improved RWP model [23], which ensures a stable average moving speed during the simulation period. In our simulation, the pause time is set at zero so that sensor nodes move constantly. The mobility of sensor nodes is controlled by varying the maximum moving speed V_{max}. By default, $V_{max} = 15m/s$.

Given a transmission range R of $40m$ [9], and an average node density ρ, the average number of neighbors for each sensor node (called *node degree*) can be obtained as $\rho\pi R^2$. By fixing the number of sensor nodes (i.e., 100 sensor nodes) and varying the simulated field from $160m \times 160m$ to $317m \times 317m$, the node degree ranges from 5 to 20. The queries are issued from 10 sink nodes which are randomly distributed in the network and do not move during the simulation. A duplicate-insensitive aggregate function (e.g., MAX) is used by the queries. Each simulation run lasts for 100 seconds of simulated time. The results are obtained by averaging the performances over 25 simulation runs. Table 1 summarizes the default parameters used in our simulations. The following three metrics are employed in our evaluation:

Energy consumption: The average amount of energy consumed by a window query.

Query Latency: The average elapsed time between the time a query is issued by a sink and the time that the sink receives the query result.

Query Accuracy: Similar to [2], we define the *valid nodes* for a window query as the nodes that reside inside the query window and are reachable by the sink during the query lifetime T. A window query processing technique expects to collect data from *all* valid nodes. Thus, we define the *query accuracy* as the ratio of the number of valid nodes involved in the query result to the total number of valid nodes. To make a fair comparison, we set T with the longest query latency among all processing techniques under study such that no queries are dropped due to query expiration.

We compare IWQE to the *naive* approach (described in Section 3), a NSI approach (denoted as *GRT*) [6] and a WSI approach (denoted as WSI:Tree) [19]. For the query forwarding heuristics in IWQE, our experimental results show that MPI is more effective than CTI in terms of energy consumption and query latency, and has a similar query accuracy to CTI. Due to space limitation, the results are not shown here. The rest of the experiments use MPI as the default forwarding heuristic. Moreover, we use the contention-based approach as our default data collection algorithm, which demonstrates better performance than other schemes (will be shown in Section 5.2.2). The max_delay in contention-based approach is set to $0.15s$ by default.

In GRT, a network-spanning tree, similar to R-tree [8], is built for query execution. Each sensor node maintains a minimum bounding rectangle (MBR), the smallest rectangle which encloses all the sensor nodes in the subtree rooted at the node itself. Given a window query, if the query window intersects with its MBR, the node sends the query to the child nodes; otherwise, the query is dropped. To support multiple sinks by one tree, all queries are forwarded to the root node which in turn initiates the query processing. Unlike GRT, WSI:Tree first forwards the query toward the query window by GPSR protocol [11].[2] Once the query arrives, a tree infrastructure is built *inside* the window along the query propagation. The detailed query processing has been described in Section 2. GRT and WSI:Tree both face a problem that the timer for a parent node to collect data may have expired before it hears from all its child nodes. There are two ways to deal with the late data reports from the child nodes: 1) the parent node discards these late data reports, which conserves the energy but impedes the query accuracy; 2) the parent forwards the late data reports to its parent, which improves the query accuracy at the cost of higher energy expense. In our implementation, GRT adopts the first method, while WSI:Tree uses the latter. As all the compared query processing techniques (including the naive approach) require a beacon mechanism for sensor nodes to maintain information about their neighbor nodes, we do not measure the cost of beacon messages, which is usually of lower order than the application data traffic [20].

5.2 Parameter and Heuristic Selection of IWQE

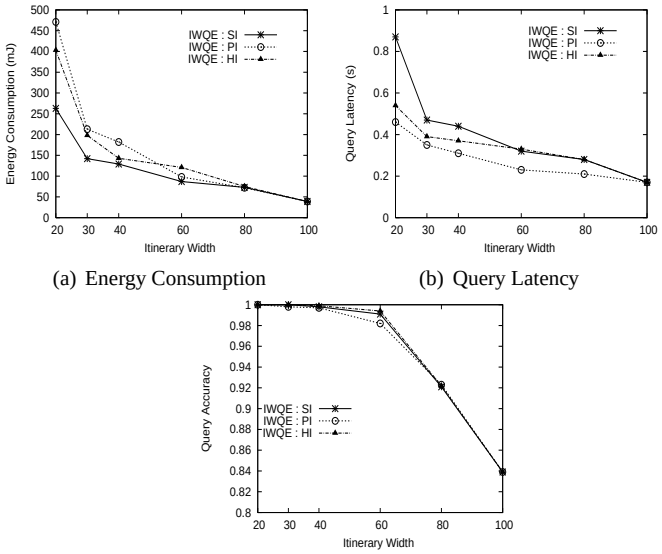
Our first set of experiments studies the itinerary width and various query execution heuristics. To be focused, we create an ideal network scenario for the following two experiments in which sensor nodes are stationary and transmissions are reliable, unless explicitly mentioned.[3]

5.2.1 Selection of Itinerary Width

Figure 5 compares the performance of IWQE with sequential itinerary (denoted by IWQE:SI), IWQE with parallel itinerary (denoted by IWQE:PI) and IWQE with hybrid itinerary (denoted by IWQE:HI) by varying the itinerary width from $20m$ (i.e., $\frac{R}{2}$) to $100m$ (i.e., $2.5R$), which is longer than the window's width(i.e., $90m$).

[2] Instead of the *directional flood* proposed by [19], we adopt GPSR for query forwarding which is more energy efficient.

[3] The transmission losses caused by the collisions are included.



(a) Energy Consumption (b) Query Latency

(c) Query Accuracy

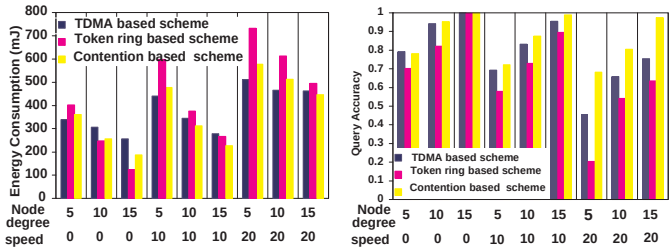
Figure 5. Study of Itinerary Width

Figure 5(a) shows the energy consumption. When the itinerary width is very small (i.e., $20m$), all approaches have noticeably high energy consumption because of the excessive overlap of transmission ranges between adjacent S-RI's, causing transmission overhead and collisions (especially for IWQE:PI). As the itinerary width is increased to $30m$, the energy consumption of IWQE drops dramatically ((because of fewer S-II's used and less overlap of data transmissions between adjacent S-RI's). After that, the energy consumption continuously decreases, since the window coverage of query propagation decreases and fewer sensor nodes participate in query execution. A similar trend is observed for query latency as shown in Figure 5(b). Figure 5(c) plots the query accuracy. All approaches maintain a very high and competitive query accuracy when the itinerary width is smaller than $40m$. However, when the itinerary width is larger than $60m$, the query accuracy dramatically decreases, since the window is not fully covered and not all the sensor nodes inside the window receive the query (recall that the transmission range is $40m$). According to the derivation in Section 3.3.1, $MIW = 34.6m$. The experimental results demonstrate that IWQE with MIW is able to achieve very high query accuracy due to its full window coverage. Thus, in the following experiments, we set the itinerary width to MIW. Nevertheless, the setting of itinerary width can be adapted to application requirements based on performance objectives.

5.2.2 Data Collection Heuristics

We study the data collection heuristics proposed in Section 3.4.1 by varying V_{max} from $0m/s$ to $20m/s$ and varying the node degree from 5 to 15. The energy consumption and query accuracy are shown in Figures 6(a) and 6(b), where IWQE:HI is used.

Several observations can be made. First, the TDMA-based and token ring-based approaches are noticeably affected by the network dynamics, since cached neighbor states, which determine the data collection schedule, could be inaccurate due to network topology changes. Second, the performance of token ring-based approach deteriorates in a sparse network, due to ring disconnection and missing tokens. Furthermore, the TDMA-based approach works best in sparse and static networks, owing to less overhead for broadcasting the data collection schedule; the token ring-based approach consumes the least energy with a competitive query accuracy in static and dense networks. The contention-based approach works fairly well under all conditions, especially in dynamic networks, since the D-nodes decide the timing of response based only on their current locations, instead of the knowledge about their neighbors. Therefore, we choose the contention-based approach as our default data collection algorithm.



(a) Energy Consumption (b) Query Accuracy

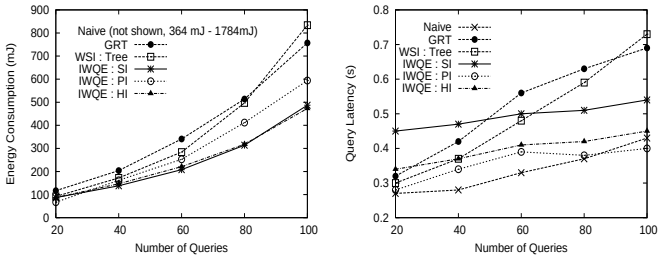
Figure 6. Data Collection Heuristics

5.3 Impact of Querying Load

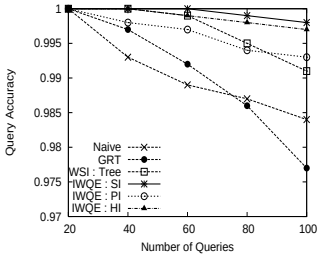
We now compare the IWQE family of approaches to the naive, GRT and WSI:Tree. We vary the total number of snapshot queries issued during the experiment from 20 to 100 to simulate different network traffic loads. Again, in this set of experiments, sensor nodes are stationary and transmissions are reliable.

Figure 7(a) shows the energy consumption of all approaches under comparison, except the one for the naive approach, which has very high energy consumption (i.e., ranging from 364mJ to 1784mJ) due to its query flooding scheme. The energy consumption of GRT is higher and grows faster than that of IWQE, since GRT incurs many unnecessary node visits. As more queries are issued, WSI:Tree, suffering from late data reports (discussed in Section 5.1), quickly increases its energy consumption, which eventually exceeds GRT's. Among the IWQE approaches, IWQE:PI costs more energy than IWQE:HI and IWQE:SI, due to more collisions between adjacent S-RI's. However, IWQE:PI shows the lowest query latency among all the IWQE approaches in Figure 7(b). The query latency of GRT increases dramatically as more queries are propagated along the same tree structure, which incurs excessive message collisions and retransmissions. As a result of more late data reports from child nodes, the query latency of WSI:Tree worsens when query load increases.

Figure 7(c) depicts the query accuracy. With a heavier network traffic load, the naive approach has a decreasing query accuracy, since more data packets are dropped



(a) Energy Consumption (b) Query Latency



(c) Query Accuracy

Figure 7. Impact of Querying Load

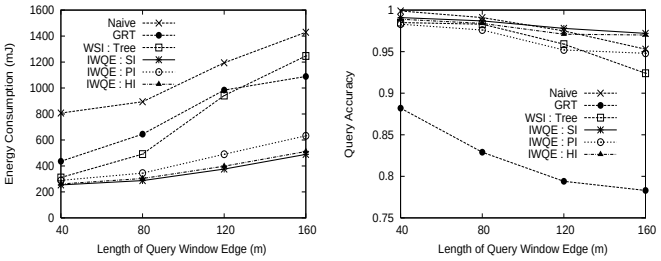
due to the increasing number of collisions. Under GRT, more late data reports from child nodes are discarded as network traffic increases, which leads to less accurate query results. Nevertheless, WSI:Tree maintains a much more stable query accuracy at the cost of transmitting the late data reports individually. For IWQE, the retransmission mechanism between Q-nodes and the caching mechanism used for D-nodes make its query accuracy stable as well.

5.4 Impact of Query Window Size

This section investigates the impact of window size on the performance. We vary the window size from $40m \times 40m$ to $160m \times 160m$. Since a query window may not be fully contained in the simulated field, we consider only the part falling inside the field for such a query. The sensor nodes move with a maximum speed of $15m/s$.

As shown in Figure 8(a), the energy consumption of GRT increases with a larger window almost as fast as that of the naive approach. This is because more MBRs intersect with the query window and more nodes are involved in query propagation. In the worst case, GRT involves all the sensor nodes in query propagation, which is the same as query flooding in the naive approach. However, with in-network aggregation, GRT receives less communication in the data collection stage than the naive approach. Due to more late data reports with an expanding query window, the energy consumption of WSI:Tree increases even faster than that of GRT. Among the three routes of IWQE, as before, IWQE:PI has the highest energy cost.

Figure 8(b) plots the query accuracy. All query processing techniques have a decreasing query accuracy as the window increases. This is reasonable since the larger the region covered by the query, the higher the probability of transmission errors and packet losses. However, the IWQE approaches are less affected due to the robust error resilience techniques proposed in Section 4.2. It is interesting to



(a) Energy Consumption (b) Query Accuracy

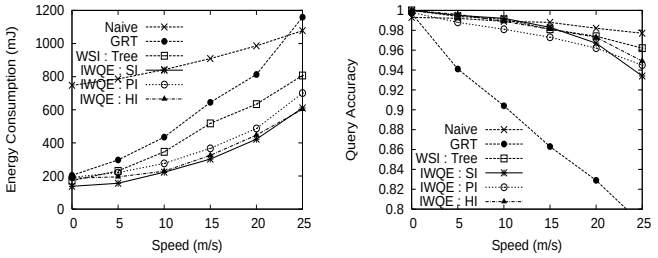
Figure 8. Study of Query Window Size

observe that the naive approach becomes worse than the IWQE approaches when the query window expands. The reason is that when a larger region is queried, more sensor nodes *independently* reply their readings to the sink node in the naive approach, which results in significant message collisions. This leads to a decreasing query accuracy as explained in Section 5.3.

5.5 Impact of Network Dynamics

In this section, we evaluate the impact of network dynamics on the performance of query processing techniques by varying V_{max} from $0m/s$ to $25m/s$.

Figure 9(a) shows that all query processing techniques tend to consume more energy when sensor nodes move faster. However, compared with the naive, WSI:Tree and IWQE approaches, GRT is affected most by this dynamics, since many more MBR update messages are created network-wide to maintain the up-to-date tree infrastructure. For a similar reason, as explained in Section 5.4, the increasing amount of update messages degrades GRT's query accuracy as shown in Figure 9(b). The accuracy of IWQE is also affected by the network dynamics. However, by integrating the data collection with query propagation, IWQE, not relying on any network infrastructure, is much more stable than GRT in dynamic networks.



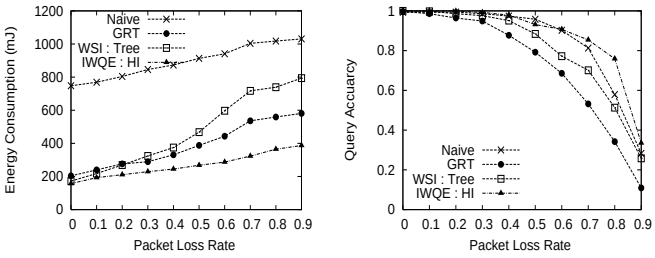
(a) Energy Consumption (b) Query Accuracy

Figure 9. Impact of Network Dynamics

5.6 Impact of Packet Losses

In this section, we examine the impact of packet losses. We simulate stationary sensor nodes and vary the rate of a packet being lost during transmission from 0.0 to 0.9. The maximum number of retransmissions at the MAC layer is 7. We show the simulation results of IWQE:HI only; IWQE:SI and IWQE:PI have similar performance to IWQE:HI.

As shown in Figure 10(a), all processing techniques consume more energy when the packet loss rate increases.



(a) Energy Consumption (b) Query Accuracy

Figure 10. Impact of Packet Losses

Since IWQE constrains the retransmissions between two Q-nodes only (not between a Q-node and a D-node), its energy consumption increases much more slowly than GRT and WSI:Tree, which retransmit all the lost packets. Figure 10(b) shows that IWQE is more robust to the packet losses than GRT, as IWQE allows D-nodes to cache and aggregate the data they overhear from other D-nodes. For GRT, considering its inherent packet losses caused by excessive communication collisions, retransmitting each lost packet deteriorates its performance. A higher packet loss rate results in more retransmissions and more late data reports in WSI:Tree; thus the energy consumption of WSI:Tree quickly increases and surpasses that of GRT.

6 Summary and Future Work

Existing techniques for processing window queries rely on network infrastructures and thus are very vulnerable to network dynamics. Our proposal, IWQE, conducts query propagation and data collection along a well-designed itinerary in an integrated manner that does not require the support of a network infrastructure. We derived an upper bound for itinerary width for achieving a full window coverage and studied three different designs for itinerary routes. In addition, three data collection schemes were explored. An extensive performance evaluation was conducted to compare the performance of IWQE and other competing protocols, including naive, GRT-based approach [6] and WSI tree approach [19]. The experimental results showed that IWQE exhibits a superior performances in terms of energy consumption, query latency, and query accuracy, and soundly outperforms all the compared protocols under various network conditions. IWQE with itinerary width as MIW is able to ensure window coverage and achieves a very high query accuracy. IWQE with HI route shows a satisfactory tradeoff between energy efficiency and query latency. Moreover, the contention-based data collection scheme works fairly well with various degrees of network density and dynamics.

IWQE has been shown as a promising window query processing algorithm for wireless sensor networks. As for the future work, we will investigate approximate spatial queries that trade query accuracy for energy efficiency. We plan to extend our design of IWQE in support of more complicated spatial queries combined with selection functions (e.g., range selection). Finally, we plan to implement IWQE on real sensor motes.

7 References

[1] The network simulator (NS-2). http://www.isi.edu/nsnam/ns/.

[2] M. Bawa, A. Gionis, H. G.-Molina, and R. Motwani. The price of validity in dynamic networks. In *Proceedings of ACM SIGMOD Conference*, Paris, France, Jun. 2004.

[3] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Pers. Commun.*, 7, 2000.

[4] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of ICDE*, Boston, MA, Mar. 2004.

[5] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of ACM MobiHoc*, Long Beach, CA, Oct. 2001.

[6] 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 GeoSensor Networks Workshop*, Portland, Maine, 2003.

[7] C. Gui and P. Mohapatra. Virtual patrol: A new power conservation design for surveillance using sensor networks. In *Proceedings of IEEE/ACM Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, Apr. 2005.

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

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *ACM SIGPLAN Notices*, 35(11):93–104, 2000.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM MobiCom*, pages 56–67, Boston, MA, 2000.

[11] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM MobiCom*, pages 243–254, Boston, MA, Aug 2000.

[12] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of ACM MobiHoc*, Annapolis, MD, Jun. 2003.

[13] J. Li and P. Mohapatra. LAKER: Location aided knowledge extraction routing for mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, New Orleans, LA, Mar. 2003.

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[15] 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 ACM SIGMOD Conference*, pages 491 – 502, San Diego, CA, June 2003.

[16] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of ACM SenSys*, Baltimore, MD, Nov. 2004.

[17] D. Niculescu and B. Nath. Trajectory based forwarding and its applications. In *Proceedings of ACM MobiCom*, pages 260–272, San Diego, CA, 2003.

[18] S. Patil, S. R. Das, and A. Nasipur. Serial data fusion using space-filling curves in wireless sensor networks. In *Proceedings of International Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, Oct 2004.

[19] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey. Data funneling: Routing with aggregation and compression for wireless sensor networks. In *Proceedings of IEEE Sensor Network Protocols and Applications*, Anchorage, AL, 2003.

[20] 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. *Mobile Networks and Applications*, 8(4):427–442, 2003.

[21] Y. Xu and W.-C. Lee. Window query processing in highly dynamic geosensor networks: Issues and solutions. *GeoSensor Networks, edited by A. Stefanidis and S. Nittel*. CRC Press LLC., 2004, ISBN: 0-41532-404-1, pp. 31-52.

[22] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell. PSGR: Priority based stateless geo-routing in wireless sensor networks. In *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Washington, DC, Nov. 2005.

[23] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *Proceedings of ACM MobiCom*, San Diego, CA, Sept. 2003.

[24] 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.

[25] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of ACM SenSys*, pages 1–13, Los Angeles, CA, Nov. 2003.