

Window Query Processing in Highly Dynamic Geo-Sensor Networks: Issues and Solutions

Yingqi Xu Wang-Chien Lee

Department of Computer Science and Engineering

Pennsylvania State University

University Park, PA 16802

E-Mail: {yixu, wlee}@cse.psu.edu

Abstract

*Wireless sensor networks have recently received a lot of attentions due to a wide range of applications such as object tracking, environmental monitoring, warehouse inventory, and health care [16, 29]. In these applications, physical data is continuously collected by the sensor nodes in order to facilitate application specific processing and analysis. A database-style query interface is natural for development of applications and systems on sensor networks. There are projects pursuing this research direction [14, 15, 25]. However, these existing works have not yet explored the **spatial** property and the **dynamic** characteristics of sensor networks.*

*In this paper, we investigate how to process a **window query** in **highly dynamic geo-sensor networks** and propose several innovative ideas on enabling techniques. The networks considered are highly dynamic because the sensor nodes can move around (by self-propelling or attaching themselves to moving objects) as well as turn to sleeping mode. The geo-sensor nodes are assumed to be location-aware. There exist many research issues in executing a window query in such sensor networks. The dynamic characteristics make those issues non-trivial. A critical set of networking protocols and access methods need to be developed. In this paper, we present a location-based stateless protocol for routing a window query to its targeted area, a space-dividing algorithm for query propagation and data aggregation in the queried area, and a solution to address user mobility issue when the query result is returned.*

1 Introduction

The availability of low-power micro-sensors, actuators, embedded processors, and RF radios has enabled distributed wireless sensing, collecting, processing, and dissemination of complex environmental data in many civil and military applications. In these applications, queries are often inserted into a network to extract and derive information from sensor nodes. There are a lot of research efforts aiming at building sensor network based systems to leverage the sensed data to applications. However, most of existing works are based on design and requirements of some specialized application. Thus, they can not be easily extended for other applications. To facilitate rapid development of systems and applications on top of sensor networks, building blocks, programming models and service infrastructures are necessary to bridging the gap between underlying sensor networks and upper layer systems and applications.

A database style query interface is natural for development of applications and systems on sensor networks. The declarative, ad hoc query languages used in traditional database systems can be used to formulate queries to exploit various functionality of sensor nodes and retrieve data from the physical world. In deed, database technology, after many years of development, has matured and contributed significantly to the rapid growth of business and industry. Commercial, research, and open-source development tools are available to facilitate rapid implementations of applications and systems on databases. Thus, a query layer on top of the sensor networks will allow database developers to leverage their experience and knowledge and to use existing tools and methodologies for designs and implementations of sensor network based systems and applications.

Sensor databases such as Cougar [25] and TinyDB [14, 15] have been proposed. However, these existing works have not yet explored the *spatial* property and the *dynamic* characteristics of sensor networks. In this paper, we investigate how to process a spatial *window query* in *highly dynamic sensor networks* (HDSN) and present several innovative ideas on enabling techniques for query processing. The network is highly dynamic because sensor nodes may go to *sleeping* mode to save energy as well as move around by self-propelling or attaching themselves to moving objects (e.g. vehicles, air, water). In addition to the capacities static sensor nodes typically possess (e.g. computation, storage, communication and sensing ability), here we assume that sensor nodes are location-aware via GPS or other positioning techniques [7, 17]. The spatial property of sensor nodes is important since sensor networks are deployed and operated in a geometric space after all. We are particularly interested in window query because it is one of the most fundamental and important queries supported in spatial databases. A window query on sensor database retrieves the physical data falling within specified *query window*, a 2- to 3-dimensional area of

interest specified by its user. There are obviously many new challenges for processing spatial window queries in HDSNs. In this paper, we use the following query execution plan as a vehicle to examine various research issues.

1. Routing the query towards an area specified by the query window;
2. Propagating the query within the query window;
3. Collecting and aggregating the data sensed in the query window;
4. Returning query result back to the query user (who is *mobile*).

Many technical problems need to be answered in order to carry out this execution plan. For example, how to route the query to the targeted area by taking energy, bandwidth, and latency into account; how to ensure a query reaches all the sensors located within window; how to collect and aggregate data without relying on a static or fixed agent; and how to deal with user mobility. To realize this execution plan, a critical set of networking protocols and access methods need to be developed. Although there are some existing work investigating either the window query processing or HDSNs, none of them provides a complete solution for window query processing in a HDSN. We have proposed innovative ideas and several enabling techniques and solutions.

Our proposals prevail for window query processing in HDSNs in the following aspects:

- Sensor nodes are able to make wise query routing decision without state information of other nodes or the network. The proposed stateless protocol, namely, *spatial query routing* (SQR) enables efficient query routing in HDSN where the topology frequent changes.
- In stead of serialized forwarding, pipelining techniques are deployed in our work and effectively reduces the delay of forwarder selection.
- Propagating query inside the query window is done in an energy-efficient way. The query propagation is ensured to cover the whole query window.
- Query results are aggregated in certain geographical region instead of at some pre-defined sensor node, which adopts well to the dynamics of HDSNs. Query results are processed and aggregated inside the query window thus the number of transmissions is reduced.
- User mobility is accommodated by utilizing the static property of geographical region as well. Query result is delivered back to the mobile user, even if she moves during query processing.

The rest of this paper is organized as follows. Section 2 presents the backgrounds and the assumptions for our work and discusses various performance requirements. In Section 3, research challenges arising in the context of this study is investigated. Section4 describes our main designs including spatial query

routing algorithm, spatial propagation and aggregation techniques and a strategy for returning the query results back to mobile users. Related work is reviewed and compared with our proposals in Section 5. Finally, Section 6 concludes this paper and depicts future research directions.

2 Preliminaries

In this section, we provide some backgrounds and discuss challenges faced in processing window queries in HDSNs. We first describe the assumptions we based on, followed by a review of HDSNs and window query. At the end, we give a list of performance metrics that need to be considered for evaluating query processing in sensor networks.

2.1 Assumptions

We assume that the sensor network is a pulled-based, on-demand network. In other words, the network only provides data of the monitored area upon users' requests. While the types of events and sensed data (e.g. temperature, pressure or humidity) are pre-defined and accessible from the sensor nodes, no sensing or transmission actions are taken by the nodes until a query is inserted into the network. This assumption is based on the fact that most of the sensor networks stay in low power mode in order to conserve energy and prolong the network lifetime. Nevertheless, a push-based network can be emulated by executing a long-running query in an on-demand network. We further assume that users are able to insert their queries from any sensor node, instead of through one or more stationary access points in the sensor networks. Finally, a user, who moves at will, is able to receive the query result back at different locations of the network.

2.2 Highly Dynamic Sensor Networks

Here we characterize the HDSNs. General speaking, the sensor nodes in HDSNs have the same functionalities of sensing, computation, communication and storage as the static sensor nodes commonly considered in the literature [1, 8]. Nevertheless, HDSNs also have the following important properties:

- **Node Mobility:** The sensor nodes in HDSN are mobile. They may drive themselves by self-propelling (via wheels, micro-rockets, or other means) or by attaching themselves to certain transporters such as water, wind, vehicles and people. With self-propelling sensor nodes, a HDSN is self-adjustable to achieve better area coverage, load balances, lifetime, and other system functionalities and requirements. These intelligent sensor networks can be controlled by the network administrator and adaptable to the queries or commands from the applications. On the other hand, for the sensor

nodes attaching to transporters, their moving patterns are dependent on the transporters. The applications may have little control or influence on the movement of these sensor nodes.

- **Energy Conservation:** Sensor nodes may switch between *sleeping* mode and *active* mode in order to conserve energy and extend the lifetime of networks. Thus, a sensor node is not always accessible. From the viewpoint of the network, the sensor node joins and leaves the network periodically or asynchronously based on sleeping schedules derived from various factors such as node density, network size, bandwidth contention, and etc.
- **Unreliable Links and Node Failures:** Another factor that contributes to dynamics of sensor networks is node and communication failures. This has a different impact from energy conservation because the available sensor nodes within the network will continue to decrease.

Sensor nodes with some or all of the above described properties form a dynamic sensor network. While nodes sleep, node failure, and unreliable communication exist in most of sensor networks, here we stress on the high mobility of sensor nodes. We argue that the mobility of sensor nodes is essential in a wider range of applications. For example, a sensor network for air pollution test, where all sensors are scattered in the air and transported by the wind; and a vehicle network, where sensor nodes are carried by moving vehicles. Applications are able to collect the data from the sensors about air pollution and traffic conditions. In addition, HDSNs may provide application layer solutions to some existing issues in the network layer. Take network topology adaptivity as an example: when an application observes that the density or the number of sensor nodes in Region X is not sufficient to satisfy the application requirements, it could command redundant or idle sensor nodes in Region Y to move to Region X.

2.3 Location Awareness

In the context of this paper, we assume that the sensor nodes are location-aware via GPS and other positioning techniques. The location awareness of sensor nodes is very important since sensor networks are deployed and operated in a geometric space after all. Since the sensor nodes in HDSNs are mobile, location information is crucial not only for certain kinds of spatial queries but also for the sensor readings to be meaningful. In addition to the time, sensor ID and readings, location information is frequently used in query predicates and requested by the applications. Moreover, location is frequently used in routing, dissemination and location-based query [4, 9, 11, 20, 21, 27, 28].

A location needs to be specified explicitly or implicitly for its use. Location models depend heavily on the underlying location identification techniques employed in the system and can be categorized as follows:

- **Geometric Model:** A location is specified as an n -dimensional coordinate (typically $n = 2$ or 3), e.g., the latitude/longitude pair in the *GPS*. The main advantage of the geometric model is its compatibility across heterogeneous systems. However, providing such fine-grained location information may involve considerable cost and complexity.
- **Symbolic Model:** The location space is divided into disjointed zones, each of which is identified by a unique name. Examples are Cricket [18] and the cellular infrastructure. The symbolic model is in general cheaper to deploy than the geometric model because of the lower cost of the coarser location granularity. Also, being discrete and well-structured, location information based on the symbolic model is easier to manage.

The geometric and symbolic location models have different overheads and levels of precision in representing location information. The appropriate location models to be adopted depends on applications. In this paper, we only consider the geometric location model.

2.4 Window Query

Due to the mobility of sensor nodes, querying the physical world based on IP addresses or IDs of the sensor nodes is not practical. For many applications of sensor networks which need to extract data from a specific geometric area, spatial queries such as window query and nearest neighbor search are essential. In this paper, we focus on window queries.

Window query enables users to retrieve all the data falls within the *query window*, a 2- to 3-dimensional area of interest defined by users. For example, consider a sensor network for an air pollution test, in which all sensors are scattered in the air and transported by the wind. Possible queries are: "What is the *average* pollution index value in a 10-meter space surrounding me?" or "Tell me if the *maximum* air pollution index value in Region X is over α ". In the first query, the query originates from inside the query window, but the latter one is issued from outside the window. In addition, in a vehicle network where sensors are carried by cars. A user may decide to change her driving route dynamically by issuing a query like "How many cars are waiting at the entrance of George Washington Bridge?". As seen in the above examples, practical window queries usually are coupled with aggregation functions, such as *AVG*, *SUM*, *MAX*, and etc. Thus, aggregation is an important operation to be carried out by the sensor networks. Aggregation algorithms are important not only to provide computational support for those functions but also to reduce the number of messages and energy consumption in the network. How to efficiently aggregate and compute the functions in network is an actively pursued research topic in sensor database. We do not provide specific algorithms for aggregation functions, but focus on issues

and strategies in enabling aggregation operation.

2.5 Performance Requirements

In order to assess the various enabling techniques for processing window queries in HDSNs, evaluation criteria need to be considered. In the following, we discuss some performance requirements:

- *Energy efficiency.* Sensor nodes are driven by a extremely frugal battery resource, which necessitates the network design and operation be done in an energy-efficient manner. In order to maximize the lifetime of sensor networks, the system needs a suite of aggressive energy optimization techniques, ensuring that energy awareness is incorporated not only into individual sensor nodes but also into groups of cooperating nodes and the entire sensor network. Based on this remark, our work studies message routing, sensor node communications and cooperations, data flow diffusion and aggregation by taking energy efficiency into consideration. These concepts are not simply juxtaposed, but fitting into each other and justify an integrative research topic.
- *Total message volume:* Recent studies show that transmitting and receiving messages dominate the energy consumption on sensor nodes [19, 23]. Therefore, controlling the total message volume has a significant effect on reducing the energy consumption (in addition to traffic) of the network. Furthermore, it also reflects the effectiveness of the aggregation and filtering of sensor readings. We expect that aggregation and filtering inside the network can reduce the total message volume tremendously.
- *Access latency:* This metric, indicating the freshness of query results, is measured as the average time between the moment a query is issued and the moment the query result is delivered back to the user. In addition to the lifetime of the sensor networks, access latency is important to most of applications, especially the ones with critical time constraints. Usually there are tradeoffs between energy consumption and access latency.
- *Result accuracy and precision:* The other performance factors trading off with energy consumption and access latency are result accuracy and precision. High accuracy and precision in query result may require powerful sensing ability and high sampling rate, localized cooperation among sensor nodes, and larger packet size for transmissions. Approximate answer and query result with less precision may sometimes be acceptable by the applications. Network should achieve as high accuracy and precision of query result as other constraints allow.
- *Query success rate:* Query success rate is the ratio of the number of successfully completed queries against the total number of query issued by applications. This criteria shows how effective the

employed query processing algorithms and network protocols are.

3 Research Issues

Although there exist some studies on various related issues of processing window query in highly dynamic sensor networks, they only addresses some partial aspects of the problems. To the best of our knowledge, this paper presents the first effort to provide a complete suite of solution/strategies to processing window query in HDSNs. In the following, we investigate the issues by considering the following query execution plan:

1. **Routing a query towards the area specified by its query window.** Once a user (or an application) issues a window query, the first question needs to be answered is how to bring the query to the targeted area in order to retrieve data from sensor nodes located there. There exist many routing protocols based on state information of the network topology or state information of neighborhood to a routing node. However, the mobility of sensor nodes in HDSNs makes those protocols infeasible. In HDSNs, the state information changes so frequently that maintaining state consistency represents a major problem. It is very difficult (if not impossible) to obtain a network-wide state in order to route a query efficiently. Thus, *stateless* strategies need to be devised. Here we exploit the location-awareness of sensor nodes to address the need of stateless routing.

An intuitive stateless routing strategy is to *flood* the network. Since each sensor node is aware of its own location, it can easily decide whether itself is within the query window or not. If a sensor node receives a query and finds itself locating within the specified query window, it may return its sensed data back to the sender for processing while re-broadcasting the query to its neighbors. Flooding does not require the sensor nodes to have knowledge of their neighbors and the network in order to route a query to targeted sensor nodes, so it meets the constraints of HDSNs very well. However, all the drawbacks of flooding such as implosion, overlap and resource inefficiency are inherited. In addition, data is very difficult to be aggregated by flooding.

Considering the spatial nature of window queries and the location-awareness of sensor nodes, a class of protocols, called *geo-routing protocols*, that make routing decisions based on locations of sensor nodes and their distances to the destination looks promising. However, almost all of existing geo-routing protocols still require some knowledge of neighbors' locations to the sensor nodes making routing decision. In this paper, we propose a truly stateless geo-routing protocol, called *stateless spatial query routing (SQR)*, which takes the strength of geo-routing protocols and employs various

heuristics to fine tune the query routing decisions. Based on SQR, a window query is routed towards the area specified by query window based on sensor nodes' locations and energy awareness, without any state information of neighbors or network topology.

2. Propagating the query to sensor nodes located within the query window. Once a query arrives in one of the sensor nodes located in the area specified by query window, the sensor node may decide whether to start the *query propagation* mode right there or pass the duty to a more suitable sensor node (e.g., send the query to a node located at the center of the window). An algorithm for query propagation should try to satisfy the following two requirements: (1) cover all the sensor nodes located in the window; and (2) terminate query propagation when all the nodes received the query. A strict enforcement of requirement (1) can ensure that no sensor node misses the query. Any miss may lead to an inaccurate query result. However, this requirement is sometimes difficult to satisfy due to the dynamic nature of the sensor networks considered here. Thus, this requirement can be weakened, based on various specifications, to accept an approximated answer or an answer with less precision. Enforcing requirement (2) is critical because the query propagation process should stop once all the sensor nodes in the window receive the query.

Conventional flooding algorithms can be modified to satisfy the above two requirements. Each sensor node maintain a query cache, which records all the queries it receives. When a sensor node receives a query, it first checks its query cache to see if there is a matching query. If yes, the query will be simply dropped; otherwise, the query is retransmitted to all its neighbors. In this way, query propagation terminates when all the sensor nodes inside the query window have the query in their caches. While the query cache can terminate the query propagation process, overhead inherited from flooding still exists. Furthermore, during the query propagation, sensor nodes may be still in move. Should the new nodes join the query processing? Should the nodes leave the window quit the query? The semantics and implied operations of queries will need to be clearly defined.

3. Collecting and aggregating the data in the query window. As we pointed out earlier, aggregation is an important operation to be supported in sensor networks for computation of aggregation functions and for reducing the number of message transmissions and energy consumptions in the networks. Thus, instead of having all the sensor nodes located inside a query window send back their readings to the user for further processing, it is more efficient to process the data in network and only deliver the result back. To process sensor readings based on certain aggregation functions and filtering predicates, the common wisdom is to assign a sensor node located inside the query

window as an *aggregation leader*, who collects and processes the readings (or partially computed results) from other nodes. This approach may work for a static network. However, in our scenario, sensor nodes may move constantly so fixed or static leader may not exist. Therefore, how to locate the leader in order to process the sensor readings locally and correctly is a challenge. In this paper, we introduce a concept of *leading region* to accommodate the mobility of aggregation leaders. Based on spatial space-division, we propose a solution, called *spatial propagation and aggregation* (SPA), for query propagation and data aggregation.

4. **Returning the query result back to the user.** After the query is processed, the result need to be delivered back to the user. Due to the user mobility, delivering the query result back to the user is not a trivial task. One intuitive approach is to route the result message based on sensor ID. However, in a highly dynamic sensor network, an ID-based routing implies flooding and thus imposes expensive energy and communication overheads. In this paper, we combine the geo-routing and a message forwarding strategy to solve the problem.

4 Proposed Solutions

In this section, we present several innovative solutions that we proposed to address the problems discussed in the previous section. We first describe *SQR*, a stateless spatial query routing method to route a window query towards the area specified by its query window. Then, we present *SPA*, an spatial space-division based approach for query propagation and sensor data aggregation within the query window. Finally, we discuss our solution for returning the query result to a user with mobility.

4.1 Spatial Query Routing

In HDSN, it is difficult for a *sender*, the sensor node which currently holds a query message, to make query routing decisions without even knowing whether there exists a neighbor. Thus, an innovative idea is to let the potential *query forwarders*, the sensor nodes reachable from the sender, decide whether they would voluntarily forward the query message based on their own state information, such as the distances from the sender and query window, their remaining energy levels, moving directions, speeds, and etc. To facilitate the potential volunteers to make timely and proper decisions, the sender provides information such as its own location, the query window (specified by two points), the size of message, and other auxiliary information to prioritize the volunteer query forwarders. *SQR* consists of two primary tasks: 1) determining a volunteer to serve as the query router; and 2) setting up the next-hop query router based on *overhearing*. The goal of the second task is to reduce forwarding delays.

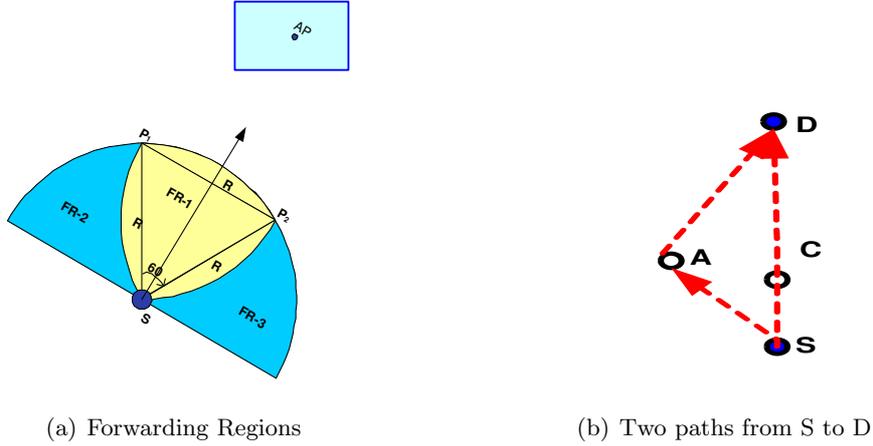


Figure 1. Spatial Query Routing

4.1.1 Volunteer Forwarders

To simplify the presentation, we use Figure 1 to depict a snapshot of the spatial query routing. A sender S , looking for a query forwarder, will broadcast messages to a space of radio range R . The routing of a query message is directed by aiming at a point in the query window, called *Anchor Point* (AP), determined by the application. In the figure, the AP is set to be the center of the query window. Here we assume that all the communications are bidirectional.

Once a sensor node receives a query and becomes the sender, it first decides its *forwarding region* (FR) based on the AP and its current position. The forwarding region is the upper part of the circle which is vertical to the line between AP and the sender. The FR is further divided by the sender into three parts:

- **FR-1** is the area with vertex S, P_1 and P_2 , and surrounded by three curves. The curve connecting any two vertices is the partial circumference of the circle centered at the remaining vertex. For example, the curve between P_1 and S is on the circle with center P_2 . Therefore, any sensor nodes located inside FR-1 can hear each others' communications with the sender.
- **FR-2 and FR-3** are the two regions inside the FR, but are falling outside of FR-1. In other words, sensor nodes in these regions can communicate with the sender S , but not necessarily aware of on-going communications between sender and other sensor nodes in FR. FR-2 and FR-3 are separated by FR-1 (shown as the dark area in Figure 1(a)). Here we number the forward regions based on their forwarding priorities (as explained below).

There are many schemes to assign forwarding priorities to the regions. Here, the sender calculates 3 priority levels based on the shortest distances of the forwarding regions to the query window. As illustrated in the figure, FR-1 has a higher priority than FR-2 which in turn has a higher priority than

FR-3¹. To find the next query forwarder, the sender broadcasts a *Forwarder_Request* message to all its neighbors. The message contains the current location of the sender, query window, location of AP, and three timer ($Response_timer_i$ where $1 < i < 3$) for the three FRs described above. $Response_timer_i$ specifies the allocated period of time sensor nodes located in FR- i should respond to the sender. The responding priorities of sensor nodes in the same FR are determined by parameters such as its energy level, distance to sender and destination, etc.

Once receiving the message, a potential forwarder checks whether itself is located inside an FR. If not, the sensor node simply drops the message. Otherwise, based on which FR it is located and the parameters considered below, the node computes and holds itself for a certain period of *holding time*. Before the holding time is expired, if the node does not hear from other neighbor nodes or the sender, it sends a *ACK_Forwarder* message back to the sender; otherwise, it will drop the message and cancel the holding timer. A node N inside FR may use heuristics based on the following parameters to determine the holding time:

- **Remaining energy on the node N** , which aims to balance the energy consumption in the network. The node with higher energy remaining has shorter holding time and thus suppresses the acknowledgements from others nodes in the same FR with less energy resource.
- **The distance between S and N** . The longer distance between S and N , the shorter holding time. The metric tries to reduce the number of hops between the source and the destination by forwarding the message as far away from S as possible. However to calculate the distance, S has to attach its geometric location with the message. Moreover, using this metric may cause longer path from the source to the destination. For example in Figure 1(b), S and D represent the source and the destination respectively. There are two possible paths from S to D : either $S \rightarrow C \rightarrow D$ or $S \rightarrow A \rightarrow D$. Based on this metric, the packet will follow the path of $S \rightarrow A \rightarrow D$. Obviously it is not the best path from S to D , because longer transmission distance costs more energy.
- **The distance between N and the query window**. To overcome the side effect in the previous metric, the nodes can also take account into the distance between itself and the query window². The N who has longer distance from the distance has longer timer.
- **Forwarding Regions**. The sensor nodes located in FR-1 have higher priority than those located in FR-2 and FR-3 to serve as a forwarder. The sensor nodes in FR-1 have to respond to the sender before $Response_timer_1$ is expired. The sensor nodes in FR-2 will need to respond after

¹The distance from FR2 to the lower left corner of query window is shorter than the distance from FR-3 to the window.

²Depending on derived heuristics, the distance can be between N and the center, the nearest points, or AP of the window.

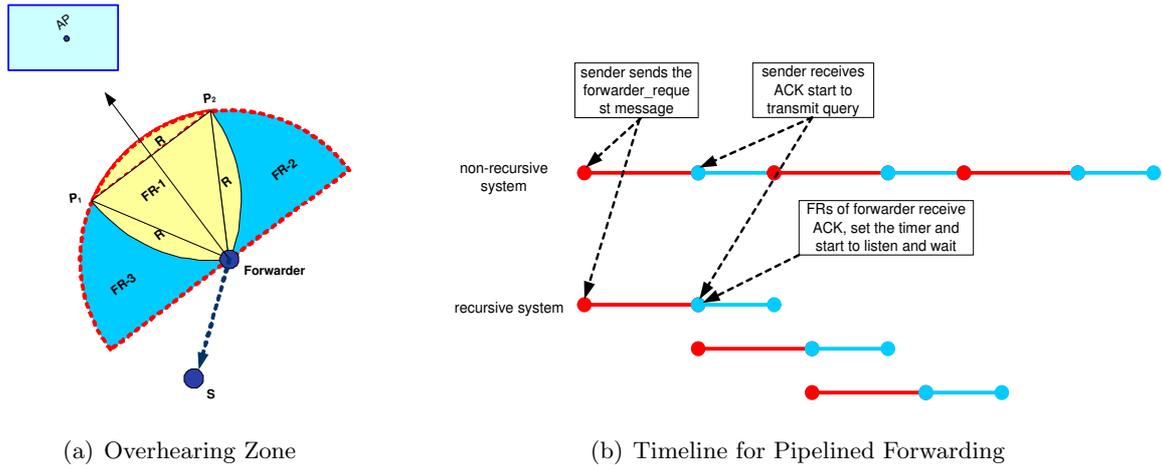


Figure 2. Pipelined Forwarding

$Response_timer_1$ and before $Response_timer_2$ is expired. The sensor nodes in FR-3 follow the same rule. Thus, nodes in FR-1 have shorter holding timers than the ones in FR-2, which has shorter timer than the ones in FR-3.

The situation where the the sender does not receive any $Ack_Forwarder$ message from any of FRs implies that there is no node in FRs of the sender or that there exist some nodes there but none of them are willing to serve as a forwarder. This well-known *forwarding hole problem* can be resolved by existing solutions [11] or by re-try after waiting for a short period of time (since the network is dynamic).

4.1.2 Pipelined Forwarding

In the forwarder volunteering process described above, the query forwarder have to wait for quite a while (i.e., after sending a $ACK_Forwarder$ message to the sender and received the query message from the sender) before it can look for the next forwarder. In this section, we introduce a pipelined forwarding strategy based on overhearing to reduce the delay of query forwarding.

When the next forwarder acknowledges the sender by sending an $ACK_Forwarder$ message, the nodes inside forwarder's FRs also can overhear the message. To speed up the query forwarding process, we propose to have a potential forwarder to attach query forwarding information (which was given in $Forwarder_Request$ message) in the $ACK_Forwarder$ message. Thus, the $ACK_Forwarder$ message is not only serving as the acknowledgement to the sender but also used to find the next forwarder. For example in Figure 2(a), when S hears an ACK from the forwarder, the sensor nodes in FRs circled by the red dotted line can also hear the ACK, which contains enough information for those sensor nodes to decide whether to volunteer as the next forwarder. The Figure 2(b) shows the timeline for three times transmis-

sions. Here is a simple analysis of the idea (assuming no conflicts happen). If the average *Response_Timer* is RT seconds, the average query transmission time is QT seconds, and the query message is forwarded M times before arriving the *DP*, the non-pipelined system takes $M \times (RT + QT)$ seconds for query forwarding, but pipelining system only needs $M \times (RT + QT) - (M - 1) \times QT = M \times RT + QT$, and saved $(M - 1) \times QT$ seconds. We feel this idea is feasible and currently developing more detailed analysis to validate it.

4.2 Spatial Propagation and Aggregation

HDSN brings major challenges to query propagation and data aggregation. Due to the mobility of sensor nodes and frequently changing network topology, the static aggregation leader and fixed infrastructure that is based on by existing query propagation and data collection algorithms may not exist [14, 25, 28]. In this section, we propose a query propagation and data collection algorithm, called *spatial propagation and aggregation* (SPA), based on spatial space division and the concept of *leading regions* (LR), which are designate regions for dispatching query and processing aggregation. One of the sensor nodes in the leading region will serve as a *propagation and aggregation leader* (PAL) to decide how to propagate a query to space-divided subareas and to collect data from those subareas for aggregation. The size and location of a leading region can be preset or adapted during run-time. The PAL of a leading region can be determined by volunteering (based on various constraints of the sensor nodes), voting (based on group decision), or designating (e.g., the first node receives the query). In addition, the PAL can adjust its LR before it proceeds to propagate queries. The size of LR can be determined by the mobility of PAL and density of nodes in the area. The main difference between the leading area here and the leader in static sensor networks is that a leading area addresses a fixed geographical area instead of some specific nodes which are dynamic in HDSNs.

Based on the innovative ideas discussed above, we use Figure 3 to further illustrate our SPA algorithm. While the leading areas can be of any shape, we represented them as circles to distinguish from the sub-windows. Once a query received by a sensor node in the query window, the sensor node can decide to 1) serve as the PAL for the window by specify an LR and start the SPA algorithm; or 2) forward the query to a temporary LR (e.g., center of the window) and let more qualified nodes there to volunteer. Assume that a sensor node in the temporary LR decides to serve as the PAL, it will specify its LR and then start the SPA algorithm.

The SPA algorithm we proposed is recursive. Once the PAL in charge of the whole window and its LR are determined, the query window W is divided into four sub-windows, namely W_1, W_2, W_3 and W_4 . For

each sub-window W_i where i is the subscript of the sub-window, a circle area in the middle is temporarily assigned as the leading region, denoted as LR_i . The query will be propagated to the four leading regions by using SQR protocol described earlier. After a query arrives at the temporary leading region in the sub-window, a PAL in charge of the sub-window is determined and its refined LR are obtained. Thus, each sub-window (W_i) is further divided into even smaller windows, i.e., W_{i1}, W_{i2}, W_{i3} and W_{i4} . The recursive splitting and forwarding procedure is repeated until a stop condition is satisfied, which will be discussed later in this section. Once the splitting and forwarding process stops, sensor nodes report their readings back to their leading region where the PAL will collect and aggregate the data. Once the PAL decides that there is no more reports from the area it leads, it sends the aggregated data to the leading region at an immediate higher level. This aggregation process is repeated until the PAL for the whole query window receives the aggregation result from all of its children LRs.

There are still several questions need to be answered in the above seemingly simple recursive algorithm:

1. **How large the leading region should be?** In SPA algorithm, the leading region is set by the PAL based on mobility of the PAL and the density of network, so that there will always be at least one sensor node in the region. Further analysis is needed to obtain the optimal leading region. To address the rare scenario that there is no sensor nodes in the leading region, one possible solution is to extend the routing protocol such that the data may be send to the upper-level leading areas.
2. **How to deal with the scenario that the PAL moves out of the leading region when the aggregation is not completed?** Once the PAL realizes that it is close to the boundary of leading region and probably will move out of the region, it floods the leading region to find a new PAL. When a new PAL is found, the old PAL hands off the partially completed aggregation result and other needed information (e.g., the higher level leading region, information about lower level region) to the new PAL. As such, the data collection and aggregation process is not interrupted.
3. **What is the stop condition for recursive splitting and forwarding?** Many heuristics can be used. For example, the size of sub-windows or the ratio of window size vs LR size. Once a PAL stops window splitting and query propagation, it floods the query inside of the sub-window that it is in charges and waits for the data from the sensor nodes inside its window. Since flooding is constrained in a very small region, duplicated transmissions are reduced.

4.3 Returning the query result back to the (mobile) user

Once the PAL at the highest level obtains the query result, it will return the result back to the user (i.e., the query issuer). Due to the unlimited mobility, a user may move while the query is processed.

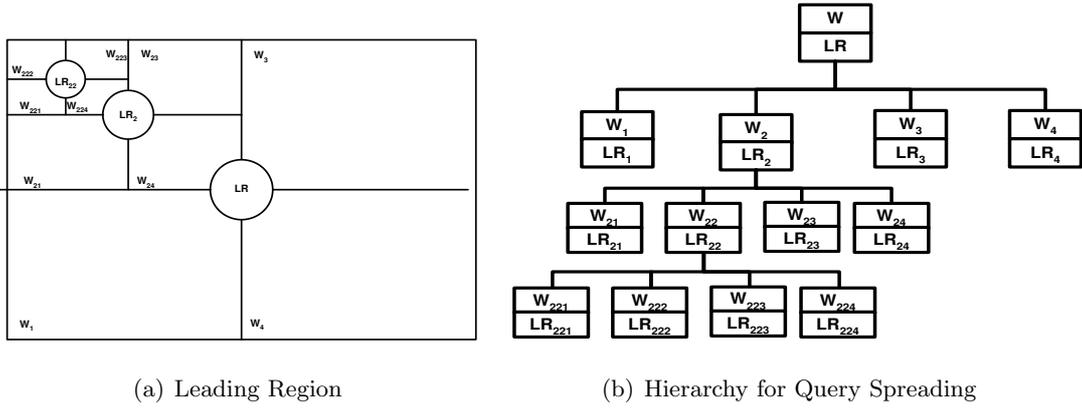


Figure 3. Spatial Propagation and Aggregation

Therefore the location where the query was issued may not be the exact location where the result should be delivered back to.

To address this problem, an *issuer region*, which defines the region where the user may stay in (while moving) during query processing, will be included as part of the query routed towards the query window. Thus, instead of routing the query result based on the user’s identifier or exact location, the stateless SQR protocol (as described in Section 4.1) is used to send the result back to the issuer region. Once the query result reaches the issuer region, it is flooded to the whole region.

Similar to deciding a leading region in SPA algorithm, a user should specify a precise area that will accommodate her future movement. However, in the rare occasion that the user will move out of the issuer region, it will specify a new region which may or may not overlap with the original region, called *extended issuer region* (EIR), and pass this information to a sensor node, called *forwarding agent* (FA), located in the old region. If the FA has to move out of the old issuer region, it will find a substitute to serve the role of forwarding agent and pass the information of EIR to the new FA. This forwarding mechanism will be established if the user has to move out of her new EIRs too. Thus, the query result may go across several forwarding agents until reaching the user.

5 Related Work

To the best of our knowledge, window query process in highly dynamic sensor networks has not yet been studied in the sensor network and database research community. However, our work has been informed and influenced by a number of research efforts.

Energy-efficient data dissemination is among the first set of related research issues being addressed. SPIN is one of the early work that is designed to address the deficiencies of classic flooding by negotiations

and resource adaption [5, 13]. Directed Diffusion takes data-centric naming approach to enable in-network data aggregation [10]. By employing initial low-rate data flooding and gradual reinforcement of better paths, directed diffusion accommodates certain level of sensor nodes dynamics. TTDD proposed by Ye et.al targets at scalable and efficient data delivery to multiple mobile sinks in a large sensor network [27]. TTDD enables mobile sinks to continuously receive data on the move at the cost of building up a grid structure for each data source within the whole network. Instead of purely flooding data or request, rumor routing algorithm proposed by Braginsky et.al is a logical compromise [3]. A source sends out "agents" which randomly walk in the sensor network to set up event paths. Queries also randomly walk in the sensor field until they meet an event path. Such previous work investigates energy-efficient data dissemination protocols in a stationary or low-mobility sensor networks, since frequent movements of upstream nodes on the route may cause a new route discovery procedure or high route maintenance expense. In our proposal, sensor nodes make decision to route a query without any state information of other nodes or networks, thus suitable for the sensor networks with unstable topology.

In addition, recent work has explored the routing algorithm for window query in static sensor networks [28]. Specifically GEAR showed how the geographic forwarding technique can help achieve energy efficiency for routing the query towards the window and spreading the query inside the window. Our ideas not only explore the query forwarding and dissemination in a stateless way, but also the data collection and aggregation and delivering the results back to the users in HDSN.

Some of the inspiration for our studies comes from [26], which enables the distributed and localized sensor operations to improve the scalability. Ye et.al proposed an original forwarding schemes for maintaining relatively constant working node density, thus prolonging the system lifetime.

Many of the energy-aware techniques developed for improving sensor network performance can be adopted by our study. For example LEACH [6] proposes a clustering based protocol that utilizes randomized rotation of local cluster heads to evenly distributed the energy load among the sensors in the network. Our proposal has the additional degree of freedom in being able to use application semantics to achieve further efficiency.

Our work borrows heavily from the literature on ad hoc routing. Specifically, it is close related to the geographical routing algorithm proposed in [11] where each sensor node greedily forwards the packet based on the knowledge of its neighbors. As a state-based algorithm, it suffers expensive upkeep of neighbor changes due to the system dynamics. LAR, in its attempt to constrain the flooding into smaller possible region instead of the whole network, is probably the closest related work [12]. However the effectiveness

of the estimation for requested zone is restricted by the destination's previous known location and its known mobility pattern. Another relevant work is IGF proposed by B.M. Blum et.al. [2]. By utilizing high node density and location awareness, IGF develops a routing algorithm which is robust and works without knowledge of the existence of neighboring nodes.

Our work is influenced by the data management researches in sensor networks as well. In Moving Objects Database (MOD), a base station (gateway) outside the network collects the sensor readings and answers user queries [22, 24]. However the centralized approach incurs heavy transmission overhead and consumes extensive energy. Our work focuses on extracting and aggregating the data locally inside the network, thereby reducing the duplicate and unnecessary transmissions. In both TinyDB and Cougar projects, the query is executed directly on the sensor nodes instead of on the gateway. Therefore, the sensor readings are only extracted out when a query is inserted into the network. TinyDB develops the techniques for query result aggregation and filtering on site by building up a semantic tree [14, 15]. Cougar project studies the query execution plan and query optimization in the sensor networks [25].

6 Conclusion

Motivated by future sensor network applications, we studied the problem of window query execution in highly dynamic sensor networks, where network topology changes frequently. The proposed suite of protocols forwards the query towards the query window by utilizing some heuristics without any state information; propagates the query inside the query window recursively and ensures the coverage of the window; aggregates the sensor readings at certain geographical region before delivering it back (which adapts to the mobility of sensor nodes and reduces the amount of transmissions); guarantees that the query issuer receives the result even if she is in movement. Our protocols explore the static property of geographical location to bear with the mobility of both sensor nodes and users.

Some issues remain to be further exploited such as the heuristics for forwarder selection and the size of the leading regions. Other future work includes developing simulations and implement prototypes to validate our proposals.

References

- [1] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy. Wireless integrated network sensors: Low power systems on a chip. In *European Solid State Circuits Conference*, Hague, Netherlands, 1998.
- [2] B.M.Blum, Tian He, S. Son, and John A. Stankovic. IGF: A robust state-free communication protocol for sensor networks. Technical Report CS-2003-11, University of Virginia, 2003.

- [3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31, Atlanta, Georgia, USA, September 2002.
- [4] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Symposium on Operating Systems Principles*, pages 146–159, 2001.
- [5] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the fifth ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, WA, USA, 1999.
- [6] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, 2000.
- [7] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.
- [8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. *ACM SIGOPS Operating Systems Review*, 34(5):93–104, 2000.
- [9] T. Imielinski and S. Goel. Dataspace - querying and monitoring deeply networked collections in physical space. *IEEE Personal Communications*, 7(5):4–9, October 2000.
- [10] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 56–67, 2000.
- [11] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, 2000.
- [12] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom98)*, pages 66–75, 1998.
- [13] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8(2-3):169–185, 2002.
- [14] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. Tag: Tiny aggregate queries in ad-hoc sensor networks. In *ACM Symposium on Operating System Design and Implementation (OSDI)*, Dec. 2002.

- [15] Sam Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of the 18th International Conference on Data Engineering*, San Jose, CA, USA, February 2002.
- [16] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM Press, 2002.
- [17] N. Patwari, A. III, M. Perkins, N. Correal, and R. O’Dea. Relative location estimation in wireless sensor networks. In *IEEE Transactions on Signal Processing, Special Issue on Signal Processing in Networks*, Oct 2002.
- [18] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 32–43, 2000.
- [19] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.
- [20] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu. Data-centric storage in sensor networks. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, 2002.
- [21] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, Georgia, USA, September 2002.
- [22] A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain, and Son Dao. Modeling and querying moving objects. In *13th International Conference on Data Engineering (ICDE)*, pages 422–432, 1997.
- [23] WINS project. Electrical engineering department, ucla. <http://www.janet.ucla.edu/WINS/>.
- [24] Ouri Wolfson, Bo Xu, Sam Chamberlain, and Liqin Jiang. Moving objects databases: Issues and solutions. In *Statistical and Scientific Database Management*, pages 111–122, 1998.
- [25] Yong Yao and Johannes Gehrke. Query processing for sensor networks. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research*, pages 21–32, 2003.
- [26] F. Ye, S. Lu, and L. Zhang. Gradient broadcast: A robust, long-lived large sensor network. Technical report, <http://irl.cs.ucla.edu/papers/grab-tech-report.ps>, April 2001.
- [27] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the eighth annual international conference on Mobile computing and networking*, pages 148–159. ACM Press, 2002.

- [28] 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.
- [29] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. In *IEEE Signal Processing Magazine*, March 2002.