Processing Multiple Aggregation Queries in Geo-Sensor Networks

Ken C.K. Lee[1], Wang-Chien Lee[1], Baihua Zheng[2], and Julian Winter[1]

[1] Pennsylvania State University, USA
{cklee, wlee, jwinter}@cse.psu.edu
[2] Singapore Management University, Singapore
bhzheng@smu.edu.sg

Abstract. To process aggregation queries issued through different sensors as access points in sensor networks, existing algorithms handle queries independently and perform in-network aggregation only at the query time. As a result of ad-hoc and independent execution of queries, no partial result is sharable and reusable among the queries. Consequently, scarce sensor network resources can be easily overconsumed, particularly, those sensors commonly accessed by queries. In this paper, we address this issue by examining strategies to maintain **M**aterialized **I**n-**N**etwork **V**iews (*MINV*s) that pre-compute and store commonly used aggregation results in the sensor network. With MINVs, aggregated sensed results for some spatial regions are available and sharable to queries. Thus, the number of sensor accesses is greatly reduced. Through simulations, we validate the effectiveness of proposed strategies.

1 Introduction

Sensor network applications are often interested in the sensed data in ceratin geographical regions (typically in form of spatial windows) rather than on some specific sensors. Examples of such applications include pollution monitoring and city road traffic control. Through sensor networks, those environmental data (i.e., pollution and traffic) are tracked and made available for querying. Due to the expensive energy cost of communication in wireless sensor networks, a summary of readings (i.e., aggregated readings) is preferred over a collection of all individual sensor readings. This kind of queries that collect aggregated readings from sensors within a geographical area is called *spatial aggregation query*. In such queries, aggregate functions such as sum, count, average, max and min are frequently used. Example queries include: "What is the average pollution index value in the 10-meter space surrounding me?" and "How many available parking slots in the car park?".

In-network aggregation has been studied in sensor database projects (for example, Cougar [1] and TinyDB [2]). These works focus on the construction and optimization of a routing tree, an ad-hoc network topology over which query results are aggregated and routed toward the root where the result is collected. However, the design of these works focuses only on a single query. For a large-scale sensor network, multiple queries may be issued from different locations with

M.L. Lee, K.L. Tan, and V. Wuwongse (Eds.): DASFAA 2006, LNCS 3882, pp. 20–34, 2006.

sensors close to the users serving as *access points*. With existing techniques, individual queries can be processed by forming independent routing trees with access points as the roots. Due to independent topologies, the aggregated results cannot be naturally shared with and reused by other queries. The scarce sensor resources (in particular, the battery power) are therefore easily overconsumed. Thus, the lifetime of a sensor network (or a certain portion of network) is quickly shortened when a large number of queries are issued.

Sharing results of multiple aggregation queries and optimizing the query performance presents some technical challenges that must be faced. An independent executing query does not take other queries into consideration. Therefore, it is difficult to (1) determine what partial aggregated result of a query will be reusable (if any) by other queries; (2) decide which sensors to store sharable results for later access; and (3) make other queries aware of the availability of the stored query results. Moreover, processing on-demand queries issued from arbitrary sensors is already a challenging issue.

In this paper, we address these challenges by maintaining **M**aterialized **I**n-**N**etwork **V**iews (*MINV*s) in the sensor networks. By identifying a set of frequently-used sensor readings at the planning stage, a MINV can be defined to store an aggregated result of readings from the set of sensors in support for processing queries at the run time. Obviously, the deployment of MINVs has several advantages. First, the study of query compatibility in multiple query optimizations is reduced to the matching between the view and the queries. A query can take full advantage of the view if the aggregation required by the query is the same as the view. Thus, multiple queries can be supported through the view. Second, the views are distributed in the sensor network, so it does not overload any single sensor and it does not require any *super sensor* (i.e., a more powerful sensor) for data storage or processing. Queries are executed by traversing sensors in the network to collect readings, either the raw data or aggregated results from the views, based on the real requirements.

The rest of the paper is organized as follows. Section 2 describes the system model our proposals are based on and reviews related work. Section 3 details the proposed schemes to support multiple spatial aggregation queries. Section 4 evaluates the impact of different factors on the performance through simulation. Finally, Section 5 concludes the paper.

2 Preliminaries

In this section, we first describe the characteristics of sensor networks and our assumptions. Then, we discuss spatial aggregation queries followed by the review of some approaches in data dissemination reported in literature.

2.1 System Model and Assumptions

We consider the sensor network formed by homogeneous and stationary sensors as shown in Fig. 1(a) where dots represent sensors. We assume that sensors are densely and uniformly distributed in a geographical area. Sensors have four

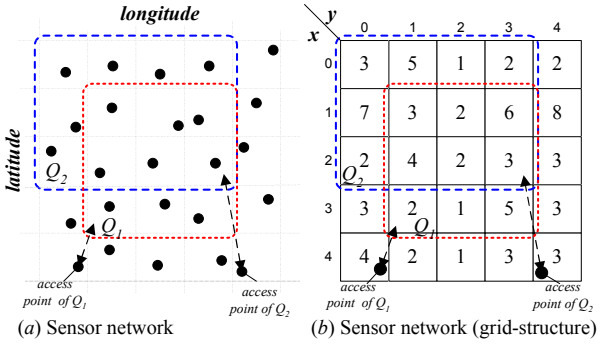


Fig. 1. Grid structure of sensor network

basic capabilities: (1) sensing, (2) storage, (3) computation, and (4) wireless communication. Sensors can sense and collect its readings, and serve as storage for its own collected readings and a partial content of a MINV. Computation enables some query processing tasks and wireless communication enables a sensor to relay messages from one sensor to its neighbors. All sensors are considered identical in terms of processing power, memory size, radio transmission coverage and energy. No task or data is sent to a specific *super* sensor for storage or processing. All sensors either operate independently or collaborate with others. We further assume the sensors are location-aware (i.e., each sensor obtains its geographical location) and time synchronized, (i.e., each sensor obtains the global clock) through GPS or other positioning techniques [3, 4].

As many well-known research in sensor data routing such as GAF [5] and data dissemination such as TTDD [6] and Comb-needle [7], we model the sensor network as a grid. We define the side length of a grid bound by $R/\sqrt{5}$ with R the transmission range of each sensor based on GAF [5]. Each cell is uniquely identified by a grid coordinate (x,y). For convenience, we refer to the cell at (x,y) as grid(x,y). We assume the cell side length is at most $R/\sqrt{5}$, therefore the signal of a sensor in a cell is conservatively reachable to other sensors in adjacent cells. To be specific, a message from a sensor located in grid(x, y) can be received by all other sensors within the same cell and neighboring cells (above, below, left and right) can also hear the message, but not sensors in diagonally adjacent cells. For simplicity, each cell is assumed to have only one sensor located at the center of grid(x, y) and denoted by $s_{x,y}$. Fig. 1(b) shows a logical presentation of the grid-structured sensor network. Further, we indicate a number in each cell as the reading of a corresponding sensor located inside the cell. Based on a grid-structured sensor network, we focus on the processing of spatial aggregation queries.

2.2 Spatial Aggregation Queries

Without loss of generality, we assume that each sensor maintains data in the following form: $\langle readings, loc\rangle$. Based on the grid structured sensor network, our spatial aggregation query is expressed in an SQL-like syntax as exemplified in the following:

SELECT **AGG**(*readings*) FROM SensorNetwork
WHERE *loc* IN $\langle [x_1, x_2], [y_1, y_2] \rangle$;

This expression means collecting (**AGG**(*readings*)), aggregated readings from sensors in a logical relation, SensorNetwork, whose locations (*loc*) are within a region specified by $\langle [x_1, x_2], [y_1, y_2] \rangle$ where x_1, x_2, y_1 and y_2 are the grid coordinates. Referring to the Fig. 1(a), two aggregation queries, denoted by Q_1 and Q_2, are issued at two different sensors (called *access points*) to aggregate readings received at all the sensors within the specified spatial window. Later, the query results are routed back to the user via the corresponding access points.

2.3 Related Work

A number of ongoing research projects focus on guiding the sensors to disseminate their measurements to interested users. In general, those works can be grouped in three major categories, namely, *pure push*, *pure pull* and *hybrid* approaches. They are briefly reviewed as follows.

Pure Push Approaches. Pure push approaches proactively propagate readings from individual sensors assuming that queries located in different parts of the sensor network may be interested in their readings. This approach is suitable when multiple queries are scattered in the network and their locations are not known in advance. Example push approaches include flooding, SPIN [8] and TTDD [6].

Pure Pull Approach. In pure pull approach, sensors are silent unless a request arrives. Queries play an active role to traverse the network to collect readings. After having been triggered by a query, interested sensors deliver their readings toward an access point (also called sink point) where the query is issued. Example pull approaches include directed diffusion [9], TAG [10] and Cougar [1].

Hybrid Push and Pull Approach. Hybrid approaches combine the advantages of both push and pull approaches. These approaches are composed of two steps. First, sensors push their readings to collection points determined by dissemination algorithms. Second, from collection points, queries pull the readings depending on requirements. Several distributed approaches are proposed such as geographic hash-tables (GHTs) [11], DIM [12] and Comb-needle [7].

Our approaches proposed in this paper are also hybrid approaches, but are very different from existing work. First, we consider multiple queries with more complicated aggregation, which have not been considered in sensor networks. Second, we assume each sensor can serve as an access point and queries can be issued at any access point.

3 Materialized In-Network View

Motivated by the needs of sharing query results for multiple queries, we examine the use of MINVs to support multiple spatial aggregation queries. The materialized view has been widely used in data warehouse and OLAP applications [13] to

improve the query response time. It computes aggregated values among a collection of disk resident operational data and stores the results as database views on disks. When a query is issued, the partial results of interested views are retrieved to ease query computation. We employ the similar idea in the context of sensor networks.

However, techniques designed for manipulating materialized views in a centralized database system cannot be directly applied to sensor networks due to the constraints of sensor networks. First, wireless communication is very energy expensive. Thus, both the message size and quantity need to be minimized. Second, each sensor has only a limited amount of memory and therefore a MINV needs to be distributed among sensors. Third, in-network view maintenance technique, rather than central approaches, are applied to keep the size of messages (intermediate view change) transferred among sensors compact.

Based upon the above factors, we propose three different approaches, namely, *full scanning*, *replication cluster*, and *prefix sum*, for managing a MINV and process spatial aggregation queries. Full scanning does not maintain any view. Replication cluster, as its name suggests, maintains a view of a pre-defined cluster and replicates the view to all the nodes inside the cluster. Prefix sum, good for supporting range sum queries, allows sensors to maintain cumulated readings over a range of sensors [13].

For the sake of simplicity, the following discussion focuses on the aggregation function, sum. Thus, a query accumulates sensor readings within a specified spatial window $\langle [x_1, x_2], [y_1, y_2] \rangle$. A given sum-aggregation query Q is to retrieve $\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} r(s_{i,j})$, with $r(s_{i,j})$ the reading received from a sensor $s_{i,j}$. Let us revisit the query Q_1 depicted in Fig. 1(b). The specified spatial window is $\langle [1, 3], [1, 3] \rangle$ and the corresponding sum is 28 ($= 2+1+5+4+2+3+3+2+6$).

3.1 Full Scanning

Full scanning is a pure pull approach. It maintains no view and serves as our baseline algorithm. Every query has to traverse all sensors within the query window to collect and aggregate readings. Initially, a query is routed from an access point to the closest sensor, named *originating node*, on the boundary of the spatial window. The query traversal follows two sorts of paths, namely, *border path* and *interior paths*. The border path runs along the window boundary and readings are recorded from encountered nodes until the originating node is met again. The interior paths are linear paths horizontally or vertically crossing the window. Employing both border and interior paths has the following advantages: (1) They ensure a complete coverage (and traversal) of all sensors in the window. (2) Parallel traversal of border path and interior paths improves response time. (3) The originating node where the scanning ends is the closest sensor to the access point; facilitating the efficient final result delivery.

The detailed description of scanning process of Q_1 is illustrated in Fig. 2. Initiated at the access point $(4, 0)$, Q_1 is first routed to the originating node $(3, 1)$ and traverses the border path in the clockwise direction. At $(2, 1)$, Q_1 finds a row of nodes ($(2, 1)$ through $(2, 3)$) not yet visited and forks a child query

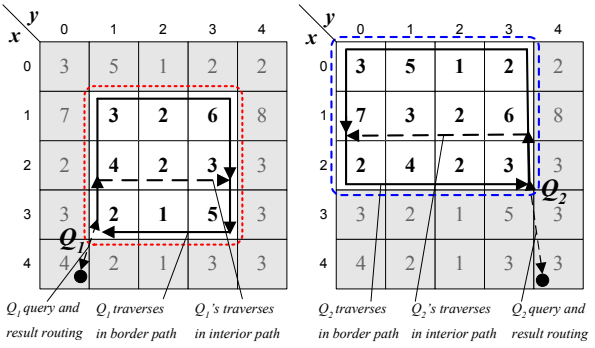


Fig. 2. Full scanning

Q_1' to traverse along the interior path to collect the reading of the sensor at $(2, 2)$. Meanwhile, Q_1 continues the traversal to $(1, 1)$, $(1, 2)$, and $(1, 3)$. Then, at $(2, 3)$, the intersection between the interior and border paths, Q_1 collects Q_1''s result (i.e., 2) and aggregates it with the tentative result (i.e., 20). Thereafter, Q_1 carries the aggregated reading of 22 and scans through $(3, 3)$, $(3, 2)$, and $(3, 1)$, with the final reading of 28. The final answer is sent back to the access point. Q_2 traverses in the same fashion but with a counter-clockwise border path. The description is omitted for space saving.

For every single query, full scanning takes one pass and visits all required sensors once. However, sensors within a common query-interested region could be accessed multiple times. In Fig. 2, sensors within the common region of Q_1 and Q_2 (i.e. $\langle[1, 2], [1, 3]\rangle$), are accessed twice. Therefore, the saving/sharing among queries can be obtained if a sharable partial result is available somewhere.

3.2 Replication Cluster

The replication cluster approach is motivated by spatial access locality of queries, i.e., sensors closely located are very likely to be accessed by a same query. Here, we assume that certain clusters are determined at the system planning stage based on analysis of query patterns and other system, network, and application factors. Further, this cluster information is assumed known to both sensors and issued queries via pre-programming. Our idea is to let each sensor within a cluster maintain (and replicate) the view of an entire cluster. Thereafter, a query fully covering a cluster can obtain the view by visiting any single sensor within the cluster. Thus, the traversal of all the member nodes of a cluster is replaced by one sensor visit, significantly reducing the number of sensor accesses.

To maintain the freshness of a view inside a cluster, every sensor knows other member sensors in the same cluster and they adopt a flooding mechanism to update the view. When a sensor obtains a new reading, it updates its own replica of the view and then broadcasts the change (the difference between previous and new readings for sum semantic) to intermediate neighbor nodes. The broadcast is marked by a unique tuple $\langle ID_s, ID_b\rangle$, with ID_s the ID of the sender sensor

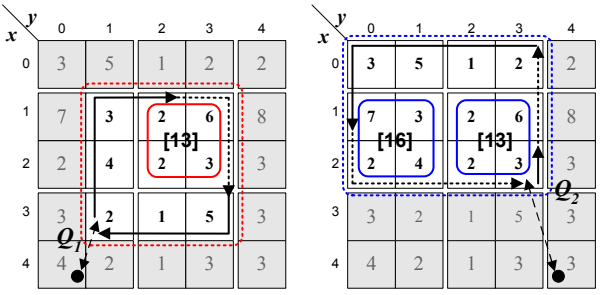


Fig. 3. Replication cluster

and ID_b the ID of the sender's broadcast. Once neighbor nodes receive the change, they update their own view replica and rebroadcast the same message. The flooding-based view update terminates at those nodes outside the cluster.

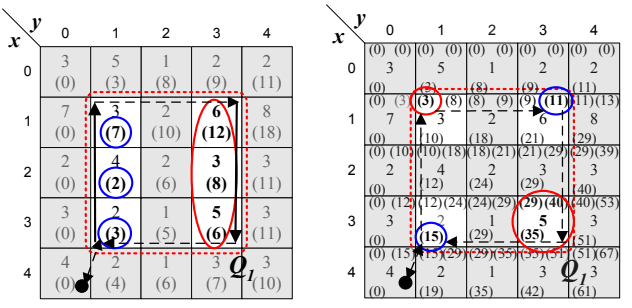
Fig. 3 shows query processing with replication cluster. Suppose two clusters are formed, located at $\langle[1,2],[0,1]\rangle$ and $\langle[1,2],[2,3]\rangle$. Like full scanning, Q_1 is first routed to node $(3,1)$ and starts traversal in the clockwise direction. However, a child query at $(2,1)$ can be avoided because the sensor at $(2,2)$ is part of a cluster fully covered by the query. Thus, it takes only the border path and aggregated result is 28 $(= 2+4+3+13+5+1)$. The result is sent back to the access point. Although Q_1 traverses both $(1,3)$ and $(2,3)$ (the path annotated by a dotted line), the query and the partial result are routed through the two sensors without invoking computation at the application level in sensors. Q_2 performs similarly. Note that the cluster $\langle[1,2],[2,3]\rangle$ is shared by both Q_1 and Q_2. Q_1 obtains the aggregated result (13) at node $(1,2)$ while Q_2 obtains it at node $(2,3)$. The workload of sensors within a cluster is also shared.

The cluster size has an impact on the performance. If the cluster size is relatively large to a query, a query will be less likely to be completely covered by a cluster; resulting in full scanning. If the cluster size is too small, the saving is limited since it still needs to fork child queries to scan interior sensors, and the sharing of aggregated readings is also limited.

3.3 Prefix Sum

The third approach is prefix sum view [13] which accumulates readings throughout the network. Each sensor maintains a partial cumulated sum of readings. We present two variants, namely, *1-dimensional (1-D) prefix sum* and *2-dimensional (2-D) prefix sum*. They employ the same aggregation concepts but are different in scopes of aggregation.

1-Dimensional Prefix Sum. With 1-D prefix sum, every sensor, $s_{i,j}$ in a sensor network maintains its own reading $r(s_{i,j})$, and a cumulated sum of readings denoted by $V_{i,j-1} = \sum_{k=0}^{j-1} r(s_{i,k})$ (or simply $V_{i,j-1} = r(s_{i,j-1}) + V_{i,j-2}$ where $V_{i,-1} = 0$). Sensor $s_{i,j}$ knows the sum of the readings of all the preceding nodes namely $s_{i,0}, s_{i,1} \cdots s_{i,j-1}$ in the row i. Fig. 4(a) depicts a sample sensor network



(a) 1-D prefix sum (b) 2-D prefix sum

Fig. 4. Linear Access on a Wireless Broadcast Channel

partitioned into 5 rows. In the top row, sensors $s_{0,j}$ keep readings: 3, 5, 1, 2 and 2 and the corresponding cumulated sums $V_{0,j-1}$ shown in braces: 0, 3, 8, 9 and 11 respectively, with $0 < j \leq 5$.

Instead of sending a sequence of readings along a row to initialize and update a view, in-network maintenance is used to propagate the sum of readings. At row i, the first node, $s_{i,0}$ starts propagating its own readings as $V_{i,0}$ $(= r(s_{i,0}))$ to the second node, $s_{i,1}$. Then $s_{i,1}$ keeps this received reading, and computes $V_{i,1}$, i.e., $V_{i,0} + r(s_{i,1})$ and sends it to $s_{i,2}$. The propagation repeats until the last sensor of the row is reached.

Upon request, a sensor $s_{i,j}$ can report any of the three readings: (1) its reading, $r(s_{i,j})$; (2) $V_{i,j-1}$, a stored cumulated sum of $r(s_{i,0})$ through $r(s_{i,j-1})$; and (3) $V_{i,j}$, a derived cumulated sum from readings of $s_{i,0}$ through $s_{i,j}$ (i.e., $V_{i,j-1} + r(s_{i,j})$). Thus, it is efficient to find out the cumulated sum from $s_{i,a}$ to $s_{i,b}$ $(a \leq b)$ in a row i by subtracting $V_{i,a-1}$ from $V_{i,b}$, since they are obtainable in $s_{i,a}$ and $s_{i,b}$. In Fig. 4(a), accessing sensors at $(0, 2)$ and $(0, 4)$ can get the cumulated sum from $(0, 2)$ to $(0, 4)$. The sensor at $(0, 2)$ provides $V_{0,1} = 8$ and the sensor at $(0, 4)$ reports $V_{0,4} = V_{0,3} + r(s_{0,4}) = 11 + 2 = 13$. The cumulated sum for the range is $V_{0,4} - V_{0,1} = 5$.

Spatial aggregation query processing is reduced to accessing the two ending nodes of rows covered by a query. For example, Q_1 in Fig. 4(a) covers three rows. Q_1 is first routed to the originating node at $(3, 1)$ to start clockwise border path traversal. It visits sensors at $(3, 1)$, $(2, 1)$ and $(1, 1)$ to collect $V_{3,0}(= 3)$, $V_{2,0}(= 2)$, and $V_{1,0}(= 7)$, respectively. Then the query is routed to $(1, 3)$, and it visits $(1, 3)$, $(2, 3)$, and $(3, 3)$ in sequence, collecting $V_{1,3}(= 18)$, $V_{2,3}(= 11)$, and $V_{3,3}(= 11)$. Finally the query result of 28 $(= -3 - 2 - 7 + 18 + 11 + 11)$ is sent back to the user.

2-Dimensional Prefix Sum. 2-D prefix sum view is in a 2-dimensional fashion. Before the discussion, we define the notation $W_{i,j}$ as the cumulated sum of reading of $s_{0,0}$ through $s_{i,j}$, i.e., $W_{i,j} = \sum_{a=0}^{i} \sum_{b=0}^{j} r(s_{a,b})$. Different from 1-D prefix sum view, each sensor $s_{i,j}$ has to maintain four readings, including (1) the local reading, $r(s_{i,j})$; (2) the cumulated sum $W_{i,j-1}$ provided by node $s_{i,j-1}$; (3) the cumulated sum $W_{i-1,j}$ provided by node $s_{i-1,j}$; and (4) the cumulated sum

$W_{i-1,j-1}$ from node $s_{i-1,j-1}$. With these four readings, $s_{i,j}$ can determine $W_{i,j}$ that is $r(s_{i,j})+W_{i,j-1}+W_{i-1,j}-W_{i-1,j-1}$. Illustrated in Fig. 4(b), a sensor, $s_{2,2}$, maintains four readings $r(s_{2,2})(= 2)$, $W_{2,1}(= 24)$, $W_{1,2}(= 21)$ and $W_{1,1}(= 18)$. The cumulated sum, $W_{2,2}$, is then $2 + 24 + 21 - 18 = 29$.

2-D prefix sum takes an entire 2D sensor network as a whole and maintain a network-wise view starting from $s_{0,0}$. When a sensor $s_{i,j}$ receives all the cumulated sums, i.e., $W_{i,j-1}$, $W_{i-1,j}$, and $W_{i-1,j-1}$ from its neighbors, it calculates $W_{i,j}$ and broadcasts two readings, $W_{i,j}$ and $W_{i,j-1}$ to sensors $s_{i,j+1}$ and $s_{i+1,j}$. The reason to broadcast $W_{i,j-1}$ is that $s_{i+1,j}$ may be located out of the transmission range of $s_{i,j-1}$ and may not receive $W_{i,j-1}$ during $s_{i,j-1}$'s broadcast.

The aggregated readings for a query can be efficiently derived with fewer sensor visits. For a region $\langle[x_1, x_2], [y_1, y_2]\rangle$, the aggregated reading is $W_{x_2,y_2} - W_{x_2,y_1-1} - W_{x_1-1,y_2} + W_{x_1-1,y_1-1}$. Since fewer nodes are visited, spatial aggregation queries can be performed at a much lower cost. As an example, Q_1 in Fig. 4(b) specifies $\langle[1, 3], [1, 3]\rangle$ as its traversal window. First, Q_1 visits the sensor at $(3, 1)$ where $W_{3,0}(= 15)$ is collected. Second, it visits the sensor at $(1, 1)$ to collect $W_{0,0}(= 3)$. Then Q_1 goes to sensors at $(1, 3)$ and $(3, 3)$ to collect $W_{0,3}(= 11)$ and $W_{3,3}(= 51)$. The final result is 28 $(= -15 + 3 - 11 + 51)$.

Discussion. Unlike replication cluster, prefix sum does not reinforce any cluster boundary. It provides better reuse of a MINV by simple addition and subtraction. With an 1-D prefix sum view, a range query can be answered by visiting nodes along two ending columns while with a 2-D prefix sum view, readings from four corner sensors are energy sufficient. 1-D prefix sum view has a slightly lower maintenance cost, and higher concurrency in view update than 2-D prefix sum view. Consider a sensor network of $m \times n$ nodes with insignificant signal interference. 1-D prefix sum view update performs a row-based update that involves at most $(n - 1)$ propagations. 2-D prefix sum view maintenance starts at $s_{0,0}$, the top-left grid, and terminates at $s_{m-1,n-1}$. It requires $(n-1)$ propagations to reach the top-right side of the sensor network and another $(m-1)$ propagations to reach the bottom. Consequently, the time to update the entire network is about the same as the time to propagate $(m + n - 2)$ messages. Besides, the storage overhead for 1-D prefix sum view is less than 2-D prefix sum view.

3.4 Extensions

In the above discussion, we have only considered the aggregate function sum. Here, we discuss the necessary extensions to the three approaches in support of other aggregate functions, such as max, min, count, average[1], and median, as shown in Table 1. To allow replication cluster to support max/min/median, the view definition can be modified such that all other possible sensor readings are maintained in the view. Keeping additional readings is useful to maintain the min/max and median. From the table we can observe that optimization is highly dependent on aggregate function types. Full-scanning can support all types of aggregate functions but it generally performs worse than the other two schemes

[1] average can be determined by the dividing sum by count.

Table 1. Support of aggregate functions

Approach	count/sum/average	max/min	median
Full scanning	Yes	Yes	Yes
Replication cluster	Yes	Additional readings required	
Prefix sum	Yes	No	No

due to the high traversal cost. Other approaches incur view update cost. Depending on the aggregate functions, appropriate approaches can be chosen or used in a combined way to evaluate a query. This observation inspires our future research work, i.e., how to combine different approaches to process aggregate queries.

4 Performance Evaluation

In this section, we provide performance evaluation of proposed approaches. We use simulation which is developed with CSIM18 [14] to examine the efficiency of our proposal in terms of total message costs, energy consumption and query response time. The system parameters are set as in Table 2. For simplicity, we assume sensor network, window queries, and clusters are squares, i.e. $m = n$, $q_x = q_y$ and $c_x = c_y$. We take sum as the aggregation function. Periodically or triggered by any sensor, update of a MINV is performed. We consider R_q as the ratio of the query issuing frequency (r_q) to the frequency of the update happening (r_u), i.e., the average number of queries issued between two consecutive updates. In addition to the proposed schemes, namely, full scanning (FS), replication cluster (RC), 1-D prefix sum (1DPS) and 2-D prefix sum (2DPS), we considered an alternative that all *raw* sensor readings are pulled out of the sensor network and maintained at a remote base station. We label this scheme **Base Station** (BS). All queries are routed to and processed at the base station. Assuming the base station is resident at grid(0,0) in the network, the message relay cost for a sensor at grid(i,j) is $i + j$.

To compare the performance of the three proposed approaches and the BS scheme, our simulation varies *query window size*, *ratio of query/update rates* and

Table 2. Parameters

Parameter	Notation	Settings	Default Value
Sensor network size	$m \times n$	$60^2, \cdots 140^2$	100^2
Query window size	$q_x \times q_y$	$10^2, \cdots 100^2$	20^2
Query rate (10^{-3} event per sec)	r_q	50, 5, 0.5, 0.05	0.5
Update rate (10^{-3} events per sec)	r_u	50	50
Ratio of query/update events	$R_q = r_u/r_q$	1, 10, 100, 1000	100
Cluster size	$c_x \times c_y$	2×2, 4×4	
Message Latency between two sensors		30ns	
Energy consumption per every message		21,600nj (send),3,600nj(receive)	

network size and study their impact on the *number of message relays*, *energy consumption* and *query response time*. The number of message relays counts the message passings between sensors. Energy consumption refers to the energy consumed per sensor in processing view updates and queries. For each sensor, the energy consumed to send a message takes 21,600 nano-joules (nj) and that to receive a message takes 3,600nj [15]. Query response time refers to the duration between query issue time and corresponding result collection time. On average, transmission of a message between sensors takes 30ms. In our evaluation, all results to present are averaged by the number of queries executed, so the average cost per query is reported. In the following subsections, the impacts of each factor is studied.

Evaluation 1. Impact of Query Window Size: Our first evaluation studies the impact of query window size which affects the query traversal costs. Generally speaking, a larger query window is expected to yield a larger number of messages transmitted for all proposed approaches. BS is however not affected by the query window size. We vary the query window size (side length in $10, \cdots 100$, with a step of 10) and fix the ratio of query rate and update rate (R_q) at 100 (by fixing query rate, r_q, at 0.5×10^{-3}/sec and update rate, r_u at 50×10^{-3}/sec).

The result is shown in Fig. 5. Firstly, Fig. 5(a) shows the number of message relays (in log scale). In the figure, we can see that 1DPS and 2DPS provide the least number of message relays, outperforming all other approaches. FS perform better than RC (both 2×2 and 4×4 clusters) only when small query sizes are experimented. As the query size larger than 30×30, FC and RC(2×2) are very

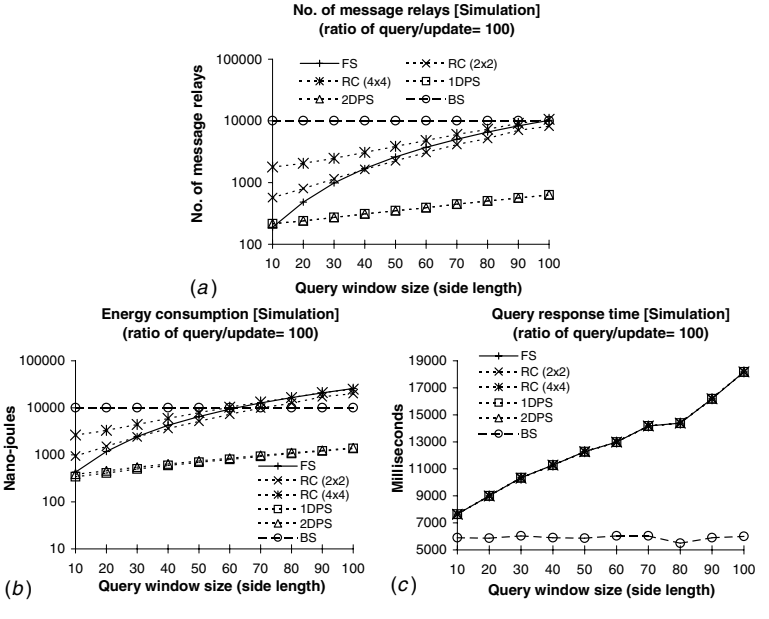


Fig. 5. Impact of query window size

close. Finally BS maintains the constant and largest number of message relays due to its high update costs.

Next, we study the impact of query size on energy consumption and query response time in Fig. 5(*b*) and Fig. 5(*c*), respectively. In Fig. 5(*b*), observation of energy consumption per sensor is very similar to that of message relays since message passing constitutes the major source of energy consumption. With this reason, we can see both 1DPS and 2DPS perform reasonably better than all others. BS maintains a constant level of energy consumption since it is independent of query size. In Fig. 5(*c*), query response time of all proposed schemes are the same since they involve the same longest query path, i.e., the border path for a same query window. Meanwhile, BS provides the shortest response time since queries do not need to traverse the query windows.

To sum up, this evaluation shows that the increase of query window size causes the message relays, energy consumption and response time increased for our proposed approaches. 1DPS and 2DPS perform the best and RC performs better than FC only when larger queries are experimented. BS provides constant performance and it performs the best in term of response time, otherwise, it is the weakest in the evaluation.

Evaluation 2. Impact of Ratio Between Query Rate and Update Rate: In the second evaluation, we study the effect of the ratio between query rate and update rate. The objective of this evaluation is to study the benefit of our proposed approaches when many queries are executed in a sensor network.

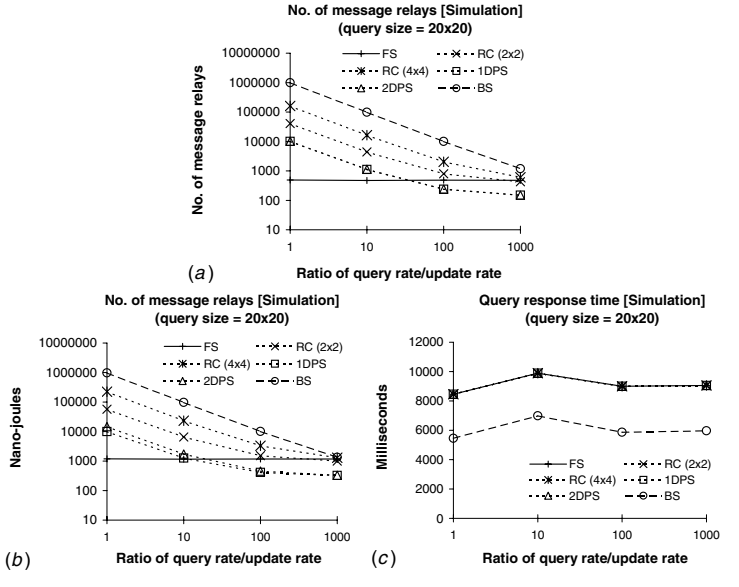


Fig. 6. Impact of query/update ratio

When the query rate is relatively high to update rate, the cost of MINV update is shared among queries for PS, RC and BS. FS is expected to have a fixed number of messages relays. We reuse the same settings as Evaluation 1 except that we vary the ratio of query rate and update rate between 1, 10, 100 and 1000 (by fixing query rate, r_q, at 50, 5, 0.5, 0.05 ($\times 10^{-3}$/sec) and fixing the update rate, r_u, at 50×10^{-3}/sec) and we retain the query window size at 20×20. Fig. 6 shows different performance metrics against the specified variation of ratio query/update rates. Fig. 6(a) shows the result in term of the number of message relays (in log scale). When a small query/update ratio (1 and 10) is experimented, all schemes except FS incurs higher costs due to heavy view update cost. Later, they decrease dramatically when higher query/update ratio (100 and 1000) is evaluated. Both 1DPS and 2DPS perform the same and better than RCs. BS is the weakest in this evaluation. Fig. 6(b) shows the energy consumption. The trends are similar to that of number of message relays because the message transmission is the major source of energy consumption. Fig. 6(c) depicts the query response time. BS performs the best and all other approaches have same query response time regardless of query/update ratio.

Evaluation 3. Impact of Network Size: The third evaluation investigates the scalability of our proposed approaches to the network size. We fix the query size and ratio of query/update rate at 20×20 and 100 respectively and we vary the network size from 60×60, to 80×80, to 100×100, to 120×120 and

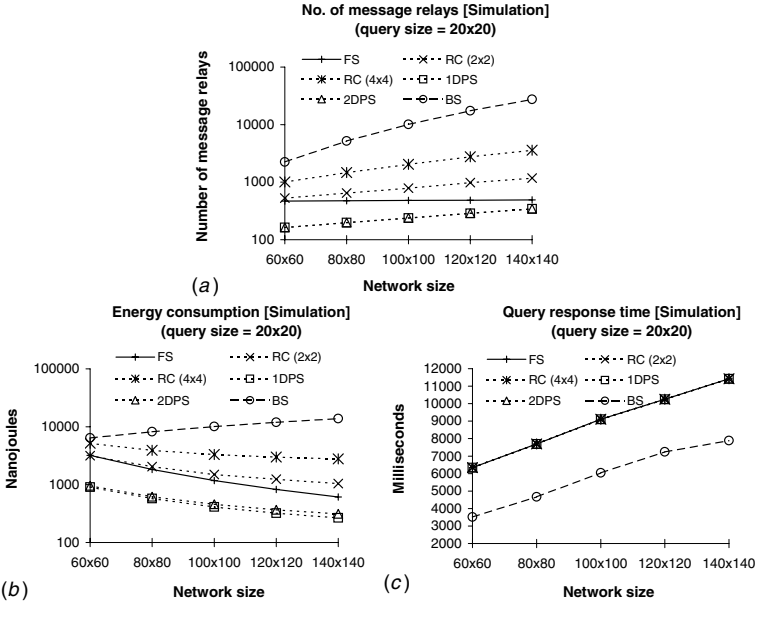


Fig. 7. Impact of network size

to 140×140. Fig. 7 plots the results. In Fig. 7(a), we can find the larger the network size is adopted, the larger the number of message relays produced by the schemes except for FS which does not maintain any view. Again, 1DPS and 2DPS perform the same and the best among all approaches while BS is the worst. Even worse, we can see that rate of increase of BS is higher than others because of increasing average distance between sensors and the base station; raising the average update costs. These results indicate that our 1DPS and 2DPS are good for various network size.

Fig. 7(b) depicts the impact of network sizes on energy consumption. It shows that the energy consumption all our proposed approaches generally decreases with the increase of the network size since queries (whose size equal to 20×20) become relatively small and they are scattered in the network. Therefore, the average energy consumed per node per query is much reduced. On the other hand, BS requires update propagation from every sensor, so that the update cost is increased with the network size. This observation points out the fact that the in-network query processing is more energy efficient. Fig. 7(c) shows the response time. This time all proposed approaches performs the same and BS remains the best.

5 Conclusion and Future Work

Aggregation queries are very important for sensor-network based systems. This paper identifies the limitations of existing ad-hoc based approaches for processing multiple aggregation queries and proposes materialized in-network views (MINVs) and associated access strategies, namely, *full scanning*, *replication cluster*, and *prefix sum*. Each approach has its own advantage and can support various types of the aggregation functions. Based on simulation, we compare their performance. Prefix sum provides the best performance in almost all the cases.

This is the first work addressing multiple spatial aggregation query processing in sensor networks. In the near future, we plan to work on a number of extensions. First, we are going to conduct in-depth analysis and extensive simulations and prototyping to validate our proposals and analysis. At the current stage, we consider queries issued in an one time fashion. To have continuous monitoring, sensors stream and aggregate their readings to the access points periodically or upon events of interest happen. As we have identified the minimum number of sensors to visit for a query using 1-D prefix sum and 2-D prefix sum in this paper, we will extend this model to register queries at those interested sensors in the spatial window. When changes of interested aggregated readings are detected at those query-registered sensors, the latest aggregated readings are propagated to the users. Further, as sensor memory is limited to accommodate a large number of views, we are studying the issue of memory management and selection criteria to maintain MINVs among sensors in sensor networks. Last but not least, as we explored in the paper, a query may include multiple aggregate functions and each approach has its own strength. It may be possible to execute the query using a combination of these proposed approaches.

Acknowledgements

In this research, Wang-Chien Lee and Ken C.K. Lee were supported in part by US National Science Foundation grant IIS-0328881.

References

1. Yao, Y., Gehrke, J.: Query Processing in Sensor Networks. In: CIDR, Asilomar, CA, USA, Jan 5-8. (2003)
2. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The Design of an Acquisitional Query Processor For Sensor Networks. In: SIGMOD Conf., San Diego, CA, USA, Jun 9-12. (2003) 491–502
3. Hightower, J., Borriello, G.: A Survey and Taxonomy of Location Systems for Ubiquitous Computing. (In: Technical Report UW-CSE 01-08-03, University of Washington)
4. MICA2 Environment/GPS Sensor Module MPR400/410/420, Crossbow Technology Inc. (http://www.xbow.com/Products/productsdetails.aspx?sid=72)
5. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed Energy Conservation for Ad Hoc Routing. In: MOBICOM, Rome, Italy. (2001) 70–84
6. Ye, F., Luo, H., Cheng, J., Lu, S., Zhang, L.: A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks. In: MOBICOM, Atlanta, GA. (Sep. 2002) 148–159
7. Li, X., Huang, Q., Zhang, Y.: Combs, Needles, Haystacks: Balancing Push and Pull for Discovery in Large-Scale Sensor Networks. In: ACM SenSys, Baltimore, MD. (Nov. 2004)
8. Heinzelman, W.R., Kulik, J., Balakrishnan, H.: Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In: MOBICOM, Seattle, WA. (Aug. 1999) 174–185
9. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks. In: MOBICOM, Boston, MA. (Aug. 2000) 56–67
10. Maddan, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In: OSDI. (Dec. 2002)
11. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S.: GHT: A Geographic Hash Table for Data-Centric Storage. In: WSNA, Altanta, GA. (Sep. 2002)
12. Li, X., Kim, Y.J., Govindan, R., Hong, W.: Multidimensional Range Queries in Sensor Networks. In: ACM SenSys, Los Angeles, CA. (Nov. 2004)
13. Ho, C.T., Agrawal, R., Megiddo, N., Srikant, R.: Range Queries in OLAP Data Cubes. In: SIGMOD Conf., Tucson, AZ. (May. 1997) 73–88
14. CSIM. (http://www.mesquite.com)
15. Lindsey, S., Raghavendra, C., Sivalingam, K.M.: Data Gathering Algorithms in Sensor Networks Using Energy Metrics. IEEE Transations on Parallel and Distributed Systems **13**(9) (2002)