On the Performance of Location-Centric Storage in Sensor Networks

Kai Xing & Xiuzhen Cheng
Computer Science
The George Washington University
Washington, DC 20052, USA
{kaix,cheng}@gwu.edu

Jiang Li
Systems & Computer Science
Howard University
Washington DC, 20059, USA
lij@scs.howard.edu

Abstract

In this paper, we study the performance of location-centric storage (LCS) theoretically. Our results indicate that LCS utilizes network resource efficiently. In particular, the storage load of sensors is independent of the network size, and is evenly distributed across the network. Moreover, the communication distance for getting event information is small. Therefore, the protocol has great scalability. In addition, we also propose algorithms for data retrieval in LCS. Our analysis shows that both the number of queries and the response cost involved in the algorithms could approach to the theoretical lower bound.

Keywords: sensor networks, location-centric storage, LCS

1 Introduction

There exists four data storage methods proposed in the context of sensor networks. In *Local Storage (LS)*, short-lived data is stored locally at the home sensor. In *External Storage (ES)*, data is sent to an outside access point where it can be further processed as needed. In *Data-Centric Storage (DCS)*, data is stored according to name/location. A data centric storage scheme [9] based on geographic hash tables [10] maps the data of the same type (name) to a fixed location in the sensor network. The performance of these three methods has been extensively studied [6,9–12]. These studies indicate that no one outperforms the other two in all situations. The fourth storage method is termed *Location-Centric Storage (LCS)*, which is complement to DCS, LS, and ES.

The basic concept of LCS has been applied to the one-dimensional sensor network mimicking a unidirectional highway for safety warning [15]. The generalization to roadways with intersections has been reported in [15]. Two-Dimensional LCS has been proposed in [14]. Compared to LS, ES, and DCS, LCS utilizes a completely different concept. In fact, LCS is relatively more context-aware. In LCS, events are replicated at multiple locations based on the associated parameter *intensity*. The intensity of an event is a function of the event type, the significance and the location of the event, the application scenario, etc. The higher the intensity, the more number of replications the event will have, and the farther away from the home location the event will be stored.

In this paper, we will study the performance of LCS in terms of storage and query theoretically. The rest of the paper is organized as follows. First we briefly overview the location-centric storage protocol for sensor networks in Section 2. The theoretical performance analysis on storage and query is given in Section 3 and 4, respectively. We conclude this paper by Section 5 with a comparison of the four storage methods.

2 Overview of Location-Centric Storage

In this section, we briefly overview the basic concept of LCS. One-dimensional LCS was first introduced in [15] for roadway safety warning. LCS for a general sensor network was proposed in [14].

We assume that sensors can obtain their own geometric coordinates (S_x, S_y) using GPS or other techniques, such as those proposed in [1,7,13]. We further assume that a robust broadcasting protocol is in place such that event records can be properly disseminated.

When detecting an event, the home sensor[1] creates a record with the following five fields:

- The *time* indicating when the event occurs.
- The *location* (i.e. the coordinates (S_x, S_y)) of the event. For simplicity, we assume an event collocates

[1]An event is usually detected by multiple sensors simultaneously but one sensor will be designated for reporting the event [2]. We term this sensor the "home sensor" of the event.

with its home sensor. Note that the *time* and *location* fields together uniquely identifies an event record.

- An integral *intensity* value (σ) that characterizes the event. Intensity values are application-specific. If the event is a car crash [15], the intensity value characterizes the time needed to clear the road. In context-aware facility query, the intensity value indicating the availability of a gas station may be proportional to the price the owner would like to pay. Generally speaking, the higher the intensity, the wider area the record should be dispatched; The closer the sensor to the event location, the higher the probability of the sensor storing the record.

- A *Time-To-Live* (TTL) as the expiration time (relative to the current moment) of the record. Records are purged from the database when their TTL values reach 0.

- The type of the event.

In LCS, when a sensor receives an event record, it computes its distance to the event location and checks whether it is "close enough"[2] to the event location. Thus each sensor is able to locally and independently determine whether it should drop or store the received event record. When a user query is received, a response is generated based on the information stored in the sensor's database. This procedure can be formally defined by the following LCS protocol.

1. When detecting an event, the home sensor *S* creates, stores and broadcasts an event record.

2. When receiving an event record, a sensor stores the record if (a) its X coordinate $\in \{x+2^0, x+2^1, x+2^2, \cdots, x+2^{\sigma-1}\}$, and (b) its Y coordinate $\in \{y+2^0, y+2^1, y+2^2, \cdots, y+2^{\sigma-1}\}$, where σ and (x,y) are the intensity value and the event location, respectively. Otherwise, the record is dropped. In both cases, the sensor broadcasts the record if its distance to the event location is less than $2^{\sigma-1}$ in both X and Y dimensions.

3. After a record is stored, its TTL value decreases by the clock tick. The entry containing the record will be purged out of the database immediately when TTL reaches 0.

4. When receiving a user query, a response to the user based on the information stored in the sensor's database will be generated.

[2]Here "close enough" means that this sensor is the closest among its neighboring sensors to one of the ideal locations where the record should be stored.

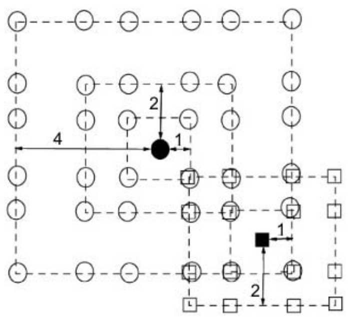

Figure 1. An Example of Location-Centric Storage Scenario. All circles store the event record for the event at the solid dot, whose intensity value is 3; And all squares keep a copy of the event record for the event at the solid square, whose intensity value is 2.

It should be noticed that for any particular pair of (i,j), where $i,j \in \{0,1,2,...,\sigma-1\}$, there is probably no sensor on the exact point of $(x \pm 2^i, y \pm 2^j)$. In this case, the sensor closest to the point in the neighborhood takes the place and keeps a copy of the record.

From the LCS protocol, it can be easily seen that records are stored in exponentially expanding frames, where the distance between the i-th and $i+1$-th frame is 2^i. Besides, the larger the intensity value, the more expanding frames that will contain the information, and thus the further the information can reach. As an example, Fig. 1 shows a sensor network with two events. The event detected at the solid dot has an intensity of 3. Therefore, its record is stored in a sensor whose horizontal and vertical distances to the solid dot are members of the set $\{1,2,4\}$ (corresponding to 2^0, 2^1, and 2^2, respectively). Similarly, the other event detected at the solid square has an intensity of 2. In this case, less sensors in a smaller area store the record.

3 Storage Performance Analysis

Due to the simple structure, our protocol has several well-defined properties that lead to promising performance. The following theorems provide the analysis.

Theorem 3.1 *Given two records* A *and* B *produced by two nodes at different locations* (A_x, A_y) *and* (B_x, B_y)*, respectively. Let* σ_A *and* σ_B *be their corresponding intensity values.*

1. *If* $A_x \neq B_x$ *and* $A_y \neq B_y$*, at most* 16 *nodes store both records.*

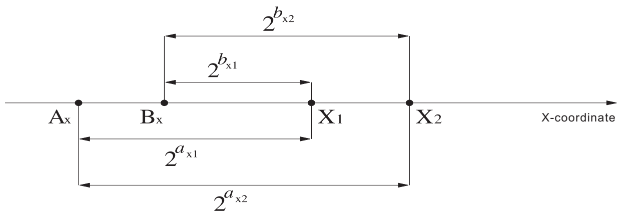

Figure 2. Two nodes at $(x1, y1)$ **and** $(x2, y2)$**,** $x1, x2 \in (B_x, \infty)$**.**

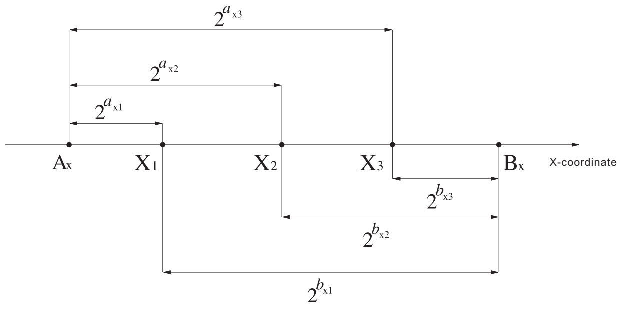

Figure 3. Three nodes at $(x1, y1)$**,** $(x2, y2)$ **and** $(x3, y3)$**,** $x1, x2, x3 \in [A_x, B_x]$**.**

2. *If $A_x = B_x$ or $A_y = B_y$, at most $4(2\sigma+1)$ nodes store both records, where $\sigma = min\{\sigma_A, \sigma_B\}$ is the smaller intensity value among the two.*

Proof. We assume that both records are alive at the same time (since otherwise no nodes will store both of them). The storage locations for record *A* are then $\{(A_x \pm 2^i, A_y \pm 2^j)|i, j \in \{0, 1, \cdots, \sigma_A - 1\}\}$. Similarly, we can determine the storage locations for *B*.

Case 1: $A_x \neq B_x$ **and** $A_y \neq B_y$**.**

Without loss of generality, we assume $A_x < B_x$. Consider the X coordinate only. A_x and B_x partition the X axis into three intervals: $(-\infty, A_x), [A_x, B_x], (B_x, \infty)$. Using the reduction to absurdity approach, we will prove that there exists at most one X coordinate in the right (left) interval such that nodes with this X coordinate will store both records of *A* and *B*.

Given two nodes at $(x1, y1)$ and $(x2, y2)$ with $x1, x2 \in (B_x, \infty)$. For contradiction we assume that both nodes store both records. Without loss of generality, we further assume $x2 > x1$. Let $a_{x1}, b_{x1}, a_{x2}, b_{x2}$ (Fig. 2) be the values such that

$$x1 = A_x + 2^{a_{x1}} = B_x + 2^{b_{x1}} \quad (1)$$
$$x2 = A_x + 2^{a_{x2}} = B_x + 2^{b_{x2}} \quad (2)$$

It is easily seen that $a_{x2} > a_{x1}$ and $b_{x2} > b_{x1}$ since $x2 > x1$. Further, $A_x < B_x$ induces $a_{x1} > b_{x1}$ and $a_{x2} > b_{x2}$. From Eqs. (1) and (2), we obtain

$$2^{a_{x1}} - 2^{b_{x1}} = 2^{a_{x2}} - 2^{b_{x2}} \quad \Rightarrow$$
$$(2^{a_{x1}-b_{x1}} - 1) = 2^{b_{x2}-b_{x1}}(2^{a_{x2}-b_{x2}} - 1) \quad (3)$$

In Eq. (3), the left side value is odd while the right side value is even, which is impossible. Therefore, the assumption that there exist two nodes storing both records can not be held true. Thus we have proved that at most one x in (B_x, ∞) such that the node at (x, y) stores both records.

With a very similar derivation, the same conclusion holds for the interval $(-\infty, A_x)$. In the following, we will prove that for those nodes whose X coordinates are in $[A_x, B_x]$, at most two of them may store both records of *A* and *B*, again using reduction to absurdity.

For contradiction we assume three such nodes at different locations $(x1, y1)$, $(x2, y2)$ and $(x3, y3)$ store both records. Let $a_{xi}, b_{xi} (i \in \{1, 2, 3\})$ (Fig. 3) be the values such that

$$x1 = A_x + 2^{a_{x1}} = B_x - 2^{b_{x1}} \quad (4)$$
$$x2 = A_x + 2^{a_{x2}} = B_x - 2^{b_{x2}} \quad (5)$$
$$x3 = A_x + 2^{a_{x3}} = B_x - 2^{b_{x3}} \quad (6)$$

Without loss of generality, we assume $x1 < x2 < x3$. Hence we have $b_{x1} > b_{x2} > b_{x3}$. Also, from the above equations, we obtain

$$2^{b_{x1}-b_{x2}}(2^{a_{x2}-b_{x1}} - 1) = (2^{a_{x1}-b_{x2}} - 1) \quad (7)$$
$$2^{b_{x1}-b_{x3}}(2^{a_{x3}-b_{x1}} - 1) = (2^{a_{x1}-b_{x3}} - 1) \quad (8)$$
$$2^{b_{x2}-b_{x3}}(2^{a_{x3}-b_{x2}} - 1) = (2^{a_{x2}-b_{x3}} - 1) \quad (9)$$

Eq. (7) is true if and only if $a_{x2} = b_{x1}$ and $a_{x1} = b_{x2}$ (otherwise the parity of the two sides would be different). Similarly, Eq. (8) and Eq. (9) are true if and only if $a_{x3} = b_{x1}$, $a_{x1} = b_{x3}$, $a_{x3} = b_{x2}$, $a_{x2} = b_{x3}$. Therefore $a_{x1} = a_{x2} = a_{x3} = b_{x1} = b_{x2} = b_{x3}$, and thus $x1 = x2 = x3$, which contradicts the assumption.

From the above analysis, we conclude that there are at most four different X coordinates such that the nodes with these coordinates will store both records. The same argument holds true for the Y coordinate. Therefore there are at most 16 positions at which the nodes store both records.

Case 2: $A_x = B_x$ **or** $A_y = B_y$**.**

Consider the case of $A_x = B_x$ and $A_y \neq B_y$. From the above analysis, we know that at most 4 different Y coordinates whose nodes store records for both events. Further, there are at most $2\sigma + 1$ different X coordinates whose nodes store both records, where $\sigma = min\{\sigma_A, \sigma_B\}$. Therefore at most $4(2\sigma + 1)$ nodes store both records of *A* and *B*. A similar discussion holds for the case when $A_x \neq B_x$ and $A_y = B_y$. ■

Corollary 1 *Assume all event records have the same intensity value σ. Given two nodes at (A_x, A_y) and (B_x, B_y), respectively,*

1. *If $A_x \neq B_x$ and $A_y \neq B_y$, they store at most 16 records in common.*
2. *If $A_x = B_x$ or $A_y = B_y$, they store at most $4(2\sigma + 1)$ records in common.*

Proof. Consider a node at (S_x, S_y). According to our protocol, the node can only store event records generated by nodes at $(S_x \pm 2^i, S_y \pm 2^j)(i, j \in \{0, 1, 2, \ldots, \sigma - 1\})$. Therefore, we can reverse the roles of records and nodes in the proof of the Theorem 3.1, which leads to the corollary. ■

Remark: Theorem 3.1 indicates that no matter how big the intensity value of a record is, there will be a fixed number of sensors that store the same pair of records in the network, as long as the two event locations are not colinear in X and Y directions. However, when these two locations are colinear in either X or Y direction, the intensity value does matter. Particularly, intensity values determine how many copies of the records can be stored and what distance the records can be propagated. Therefore, they affect the storage space at each sensor, as indicated by Theorem 3.2. Corollary 1 shows that records are distributed among all nodes instead of converging onto some of them. Thus no hot spots will be created.

Theorem 3.2 *Assume broadcast is instantaneous. Let the average intensity value of records be σ, and the average TTL value be T (assumed as an integer). Also assume that at any node, the number of events detected during a unit time, denoted by N, follows a Poisson distribution with a mean of λ. If N is independent node-wise and time-wise, the average number of records stored at each node is $\lambda(4\sigma^2 + 4\sigma + 1)T$.*

Proof. Given a node at (S_x, S_y), it is easily seen that at any time t, the node stores the records generated at (x, y) (where $x = S_x \pm 2^i$ and $y = S_y \pm 2^j$ for $i, j \in \{0, 1, \ldots, \sigma - 1\}$) during the time interval $[t - T, t]$. Let $N_k^{x,y}$ be the number of events for which the node at (x, y) generates records during the kth unit time interval $[t - T + k - 1, t - T + k]$ ($k \in \{1, 2, \ldots, T\}$). The average number of records generated by this node during the time interval $[t - T, t]$ is thus $W_{x,y} = \sum_{k=1}^{T} N_k^{x,y}$. Consequently, at any time t, the number of records stored in the node at (S_x, S_y) is $W = \sum_{x,y} W_{x,y} = \sum_{x,y} \sum_{k=1}^{T} N_k^{x,y}$. Since $N_k^{x,y}$'s are Poisson distributed and independent from each other, W follows the Poisson distribution with the mean of $\lambda(2\sigma + 1)^2 T = \lambda(4\sigma^2 + 4\sigma + 1)T$. ■

Remark: Note from Theorem 3.2 that the average number of records stored in each node at any instant time is independent of the network size. This independency also implies the bounded broadcast of records. Therefore, our protocol is efficient in terms of storage requirement, power consumption, and bandwidth utilization. It is thus highly scalable.

Theorem 3.3 *Let σ be the intensity value in an event record. Assume the radio range of each sensor is set to be one unit, then the record will be broadcasted at most $(2^\sigma - 1)^2$ times. With a careful broadcast scheduling, this upper bound can be reduced to $2\sigma \times (2^\sigma - 2) + (2^\sigma + 1)$.*

Proof. According to our protocol, a record with the intensity σ generated at (x, y) is propagated within the area of $[x - 2^{\sigma-1}, x + 2^{\sigma-1}]$ and $[y - 2^{\sigma-1}, y + 2^{\sigma-1}]$. Imagine a grid laid on the area centered at (x, y), and each grid cell is sized 1×1. Since the radio range of each sensor is one unit of distance, only the nodes on (or closest to) the crossings of the virtual grid lines need to participate in the broadcast. Also note that the broadcast stops on the boundary of the area. Therefore, the total number of intermediate nodes participating in the broadcast is at most $(2^\sigma - 1)^2$.

This upper bound can be improved if the record is propagated horizontally and vertically only when necessary. To be specific, each of the sensors at $(x \pm i, y)$, where $i = 0, 1, \cdots, 2^{\sigma-1}$, needs to broadcast once; and each of the sensors at $(x \pm 2^i, y \pm j)$, where $i = 0, 1, \cdots, \sigma - 1$ and $j = 0, 1, \cdots, 2^{\sigma-1} - 1$, needs to broadcast once. Therefore the total number of broadcastings is at most $2\sigma \times (2^\sigma - 2) + (2^\sigma + 1)$. ■

Remark: From Theorems 3.2 and 3.3, we observe that LCS is efficient in network resource (power, bandwidth, memory) utilization. Further, LCS is fair to all nodes in storage space, as long as the records are uniformly and independently generated. This is an intrinsic difference compared with DCS [10,12], which creates storage hot spot even when the number of events in the network is low.

Theorem 3.4 *Suppose (x, y) is the location of a user and (S_x, S_y) is the location of an event whose record has an intensity value of σ. Let $d_x = |x - S_x|$, and $d_y = |y - S_y|$. If the user is in the broadcast region of this event, i.e., $x \in [S_x - 2^{\sigma-1}, S_x + 2^{\sigma-1}]$ and $y \in [S_y - 2^{\sigma-1}, S_y + 2^{\sigma-1}]$,*

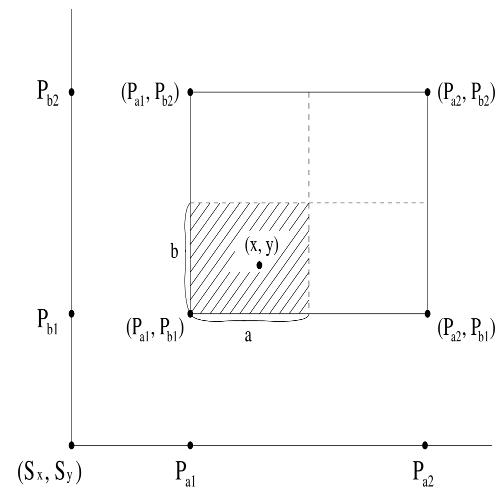

Figure 4. For a user at (x, y) in the shaded area, the record of the event at (S_x, S_y) will be provided by the node at (P_{a1}, P_{b1}). The query distance is thus the distance from (x, y) to (P_{a1}, P_{b1}). When the user is closer to (P_{a1}, P_{b2}), (P_{a2}, P_{b1}) or (P_{a2}, P_{b2}), the calculation is similar.

the average query distance d_q is:

$$d_q = \begin{cases} \frac{\sqrt{a^2+b^2}}{3} + \frac{a^2 \ln(\frac{\sqrt{a^2+b^2}+b}{a})}{6b} + \frac{b^2 \ln(\frac{\sqrt{a^2+b^2}+a}{b})}{6a}, \\ \quad \text{if } x \neq S_x \pm 2^i \text{ and } y \neq S_y \pm 2^j, \\ \quad \text{where } i, j = 0, 1, \cdots, \sigma - 1; \\ \\ \frac{\sqrt{a^2+b^2}}{2}, \qquad \text{otherwise.} \end{cases}$$

where

$$\begin{cases} a = (2^{\lceil \log_2 d_x \rceil} - 2^{\lfloor \log_2 d_x \rfloor})/2 \\ b = (2^{\lceil \log_2 d_y \rceil} - 2^{\lfloor \log_2 d_y \rfloor})/2 \end{cases}$$

Proof.

We denote

$$P_{a1} = S_x + 2^{\lfloor \log_2 d_x \rfloor},\ P_{a2} = S_x + 2^{\lceil \log_2 d_x \rceil}$$
$$P_{b1} = S_y + 2^{\lfloor \log_2 d_y \rfloor},\ P_{b2} = S_y + 2^{\lceil \log_2 d_y \rceil}$$

Therefore, $a = (P_{a2} - P_{a1})/2, b = (P_{b2} - P_{b1})/2$. Note that $a = 0$ indicates that $x = S_x \pm 2^i$ for $i = 0, 1, \cdots, \sigma - 1$, and $b = 0$ indicates that $y = S_y \pm 2^j$ for $j = 0, 1, \cdots, \sigma - 1$. There are four different cases:

Case 1: $a \neq 0, b \neq 0$.

In this case, the user at (x, y) chooses the closest point from (P_{ai}, P_{bj}) $(i, j = 1, 2)$, and sends the query to the node at that point. Whichever point the user chooses, the situation is similar. Therefore, we will only consider the situation when (P_{a1}, P_{b1}) is the closest to the user, as shown in Fig. 4.

Since (x, y) can be any point in the shaded square with the same probability, the average query distance d_q is

$$\begin{aligned} d_q &= \frac{\int_0^a \int_0^b \sqrt{x^2 + y^2} dx dy}{ab} \\ &= \frac{\sqrt{a^2 + b^2}}{3} + \frac{a^2 \ln(\frac{\sqrt{a^2+b^2}+b}{a})}{6b} \\ &\quad + \frac{b^2 \ln(\frac{\sqrt{a^2+b^2}+a}{b})}{6a} \end{aligned}$$

Case 2: $a \neq 0, b = 0$.

In this case, (x, y) is on the line between (P_{a1}, P_{b1}) and (P_{a2}, P_{b1}). The user will choose the closer point from (P_{a1}, P_{b1}) and (P_{a2}, P_{b1}) for the query. Therefore, the average query distance is

$$\frac{a}{2} = \frac{\sqrt{a^2}}{2} = \frac{\sqrt{a^2 + b^2}}{2} \ (\because\ b = 0)$$

Case 3: $a = 0, b \neq 0$.

The derivation is similar to case 2.

Case 4: $a = 0, b = 0$.

In this case, the user is at (P_{a1}, P_{b1}). Therefore, the query distance is $\frac{\sqrt{a^2+b^2}}{2} = 0$. ■

Remark: Theorems 3.4 and its proof reveal that when the user resides in the broadcast region of an event, the query distance is no more than the distance between the user and the home location of this event. In fact, in most cases, the former is much smaller than the latter, resulting in a low query delay.

It is obvious that using our protocol, the information of an event can only be propagated to the furthest distance of $2^{\sigma-1}$, where σ is the intensity value of the record corresponding to the event. Therefore, a user can only be notified of the events that occur within certain distance from the user. Usually, a user can communicate with any node within the neighborhood area, called the *user communication area*. If the information of an event can be obtained by the user, we say that *the information is covered by the user communication area*, and we call an area with covered information a *covered area.*

Theorem 3.5 *Assume all events have the same intensity value σ. Suppose an event occurs at an arbitrary location (S_x, S_y) in the network, but a user at (x, y) can only communicate with nodes within an area of $l \times l$ (denoted by $\mathcal{H}$)*

centered at (x, y). Let $d_x = |S_x - x|$, $d_y = |S_y - y|$, and $\beta = \min(\sigma - 1, \lfloor \log_2 l \rfloor)$. The user is notified of the event if $(S_x, S_y) \in \mathcal{A}, \mathcal{B}$ or $\mathcal{C}$, where area $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$ are defined as follows,

$$\begin{cases} (S_x, S_y) \in \mathcal{A} & \text{if } d_x \leq 2^\beta + l/2 \text{ and } d_y \leq 2^\beta + l/2 \\ (S_x, S_y) \in \mathcal{B} & \text{if } (d_x \leq 2^\beta + l/2 \\ & \text{and } 2^\beta + l/2 < d_y \leq 2^{\sigma-1} + l/2), \\ & \text{or} \\ & (2^\beta + l/2 < d_x \leq 2^{\sigma-1} + l/2 \\ & \text{and } d_y \leq 2^\beta + l/2) \\ (S_x, S_y) \in \mathcal{C} & \text{if } 2^\beta + l/2 < d_x, d_y \leq 2^{\sigma-1} + l/2. \end{cases}$$

Otherwise, the user cannot be notified,

Proof. In the case of $(S_x, S_y) \in$ area $\mathcal{A}, \mathcal{B}$ or $\mathcal{C}$, since the information of any event occurs in area $\mathcal{A}, \mathcal{B}$ or $\mathcal{C}$ is recorded by some nodes in $\mathcal{H}$, and the user communicates with all nodes in $\mathcal{H}$, the user will be notified of the event.

In the case that the event (S_x, S_y) occurs in the area other than area $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$, since the information of the event is not recorded by any node in $\mathcal{H}$, the user won't be able to receive the event information. ■

4 Query Performance Analysis

In a typical sensor network, query a number of (if not all) sensors to retrieve gathered data is a central task for monitoring and control. A naive but inefficient way is to flood queries to all sensors in the monitored area.

When LCS is applied as the data storage method, a simple and efficient approach for data retrieval is readily available. Both the number of queries needed and the response cost could approach to the theoretical lower bound. In the following, we propose method for data retrieval in a one-dimensional network, and then extend the idea to the two-dimensional case.

4.1 Definitions

Given a graph $G = (V, E)$, a *dominating set* D of G is a subset of V such that for $\forall u \in V - D$, there is a $v \in D$ for which $(u, v) \in E$ (i.e., u is dominated by v). A dominating set with a minimum cardinality is called a *minimum dominating set*. Computing a minimum dominating set is an NP-complete problem [4]. Given a dominating set D of graph G, if each vertex of G is dominated by exactly one element in D, then D is called a *perfect dominating set* of G. A perfect dominating set is necessarily a minimal dominating set.

Let n and m be k-bit binary numbers. The *Hamming distance* $h(n, m)$ of n and m is the number of bit positions in which n and m differ. A *k-dimension hypercube* H_k is an undirected graph with $N = 2^k$ vertices. In H_k, each vertex is uniquely identified by a k-bit binary expansion of some integer $u \in \{0, 1, ..., N - 1\}$, and an edge connects two vertices u and v if and only if $h(u, v) = 1$.

Let C be an *error-correcting code* consisting of N codewords, in which each codeword consists of n letters taken from an alphabet $\mathcal{A}$ of length q, and every two distinct codewords differ in at least $d = 2e + 1$ places. Then C is said to be *perfect* if for every possible word w_0 of length n with letters in $\mathcal{A}$, there is a unique code word w in C in which at most e letters of w differ from the corresponding letters of w_0. An example perfect code is the (k, t)-Hamming code, in which a codeword of length k contains t check bits.

Computing a perfect dominating set D of H_k can be transformed to the problem of computing a perfect single-error-correcting code [8]:

Lemma 4.1 *Given a k-dimension hypercube H_k, a perfect dominating set D of H_k is precisely a perfect binary single-error-correcting code with 2^k codewords.*

4.2 Field Data Gathering in a One-Dimension Sensor Network

Assume a one-dimension network contains N sensors, denoted by S_0, S_1, S_2, S_3, ..., S_{N-1}, placed at locations $0, 1, 2, 3, ..., N - 1$, respectively, in a straight line, where $N = 2^k$ and $min_{i=0}^{N-1}\{\sigma_{S_i}\} \geq k$. A graph $G(V, E)$ can be constructed by setting V=$\{S_0, S_1, S_2, S_3, ..., S_{N-1}\}$ and an edge $e(S_u, S_v) \in E$ if and only if the distance between S_u and S_v equals 2^i, where $i \in \{0, 1, \cdots, k - 1\}$. Based on this graph model, a minimum dominating set of G stores all event information recorded in the whole network, which means that querying this subset suffices in order to retrieve all events.

4.2.1 Case I: $k = 2^t - 1$

For simplicity, we consider the case of $k = 2^t - 1$ for a non-negative integer t first. Since computing a minimum dominating set is a NP-complete problem [5], a computationally efficient algorithm in this section is sought to find out a good approximation. We first identify a hypercube H_k as a subgraph of G; Then map the (k, t)-Hamming code, a perfect single-error correction code, to a perfect dominating set of H_k. We prove that the size of this perfect dominating set is at most twice of that of a minimum dominating set of G.

Lemma 4.2 *$G(V, E)$ defined above contains a k-dimension binary hypercube H_k as a subgraph.*

Proof. We prove this lemma by constructing $H_k(V', E')$ by setting $V' = V$ and an edge $e(S_u, S_v) \in E'$ if and only if $e(S_u, S_v) \in E$ and $h(u, v) = 1$, since $S_u(S_v)$ resides at position $u(v)$, every pair of S_u and S_v satisfying $h(u, v) = 1$ has a corresponding edge in E. ■

Lemma 4.2 proves the existence of H_k in G. According to Lemma 4.1, the problem of finding a perfect dominating set in a hypercube H_k can be transformed to the problem of finding a perfect single-error correction code. The following theorem maps the (k, t)-Hamming code to a perfect dominating set in H_k when $k = 2^t - 1$.

Theorem 4.3 *Given a k-dimension hypercube H_k, if $k = 2^t - 1$, where $t \in \{0, 1, ...\}$, H_k has a perfect dominating set containing exactly $\frac{2^k}{k+1}$ nodes.*

Proof. For a (k, t)-Hamming codeword $b_1 b_2 \cdots b_k$, the bit positions b_i satisfying $i = 2^j$ with $j = 0, 1, \cdots, t - 1$ are the check bits while others are data bits. Therefore the (k, t)-Hamming code is used to correct a single error of data words with length $k - t$. Since the (k, t)-Hamming code is a perfect single-error correction code, the subset of vertices in H_k whose binary representations correspond to the valid Hamming codewords of all data words with length $k - t$ is a perfect dominating set, based on Lemma 4.1. The size of this perfect dominating set is 2^{k-t}, which equals to $\frac{2^k}{k+1}$. ■

Theorem 4.4 *When $k = 2^t - 1$ for a non-negative integer t, the size of the perfect dominating set found via the construction of a (k, t)-Hamming code is at most $2 \cdot OPT$, where OPT is the cardinality of a minimum dominating set of the original graph $G(V, E)$ formed from the one-dimension LCS network.*

Proof. Note that each node stores records from at most $2k$ other sensors, namely dominates at most $2k$ nodes. Therefore the size of a minimum dominating set of $G(V, E)$ is lower-bounded by
$\frac{N}{2k} = \frac{N}{2log_2 N}$, which means that $OPT \geq \frac{N}{2log_2 N}$. According to Theorem 4.3, the size of the perfect dominating set computed based on the construction of a (k, t)-Hamming code is $\frac{2^k}{k+1} = \frac{N}{k+1} = \frac{N}{log_2 N+1} \leq \frac{N}{log_2 N} \leq 2 \cdot OPT$. ■

Algorithm 1 summarizes the procedure of finding a small subset of node to query in the one-dimension LCS network when $N = 2^k$ and $k = 2^t - 1$ in order to retrieve all event information.

For example, given a one-dimension LCS network with a network size of $N = 128$. A hypercube H_7 can be derived easily according to Lemma 4.2. Since $k = 7$, we have $t = 3$. The $(7, 3)$-Hamming code contains the following valide codewords: {0000000, 0000111, 0011001, 0011110, 0101010, 0101101, 0110011, 0110100, 1001011, 1001100, 1010010, 1010101, 1100001, 1100110, 1111000, 1111111}. Therefore, we need to query only 16 sensors (S_0, S_7, S_{25}, S_{30}, $\cdots$) in order to retrieve all event information within the one-dimension LCS network of 128 nodes.

Algorithm 1:

```
1. Construct $G(V, E)$ to model the one-dimension LCS network;
2. Compute the Hypercube $H_k$ as a subgraph of $G$ based on Lemma 4.2;
3. Compute the $(k, t)$-Hamming code;
4. Map the Hamming codewords to the corresponding nodes in the hypercube $H_k$ and $G$.
```

4.2.2 Case II: $k \neq 2^t - 1$

When $k \neq 2^t - 1$, we can partition the network and map it to several hypercubes and find corresponding perfect dominating set in each hypercube. This problem can also be solved via constructing leafy spanning tree of hypercube [3].

4.3 Two-Dimension LCS network

In this section we present how to extend the results in the one-dimension LCS network to the case of two-dimension LCS. Assume the user-queried area is $N \times N$ and $N = 2^k$, where $k = 2^t - 1$ for a non-negative integer t. The following algorithm identifies the dominating set to query for event information retrieval.

Algorithm 2:

```
1. Apply Algorithm 1 to the one-dimension LCS network with size $N$. Find the dominating set $D = \{d_1, d_2, \cdots\}$.
2. For a sensor $i$ at location $(a, b)$ in the two-dimension network, if $a \in D$ and $b \in D$, then $i$ will be selected into the query set S.
```

Theorem 4.5 *All of the information in the user-queried area $N \times N$ is covered by the queried sensors in the set S.*

Proof. It is obvious that the information stored by sensor i at (a, b) is covered by the sensors in the set S if $a \in D$ and $b \in D$, namely $(a, b) \in S$.

Now we consider the cases of $a \in \overline{D}$ or $b \in \overline{D}$ or both, namely $(a, b) \in \overline{S}$. Following the results obtained for the one-dimension LCS network, we observe that a is dominated by some node d_i in D, b is dominated by some node d_j in D, which indicate that the information stored at (a, b) is covered by (d_i, d_j). Therefore all of the information in the user-queried area is covered by the sensors in S. ■

Based on Theorem 4.5, the size of the dominating set found via **Algorithm 2** is $\frac{N^2}{(log_2 N+1)^2}$. Table 1 reports several example results for the case of two-dimension LCS network. It indicates that **Algorithm 2** is able to greatly reduce the number of nodes to be queried.

Network Size $N \times N$	The Number of Queried Sensors
8×8	4
128×128	256
$2^{15} \times 2^{15}$	$4,194,304$

Table 1. The network size and the corresponding number of sensors to be queried.

5 Comparison Study

Note that LCS has better scalability and stronger resilience against node failures compared with DCS. Nevertheless, LCS and DCS can coexist in a sensor network since they target different application scenarios. DCS is designed for a large-area data dissemination. The designated storage locations for an event type in DCS decreases query overhead since no flooding is involved. On the other hand, LCS is designed for scenarios when the event information is needed only when a user is approaching the event location. The query of a specific event in LCS requires global flooding.

Compared to LS, a group of sensors in LCS store an event generated roughly at the geometric center of the group, while in LS a sensor stores its local observations. LCS has better resilience against node failures, and is suitable for summary data dissemination. Overall, LCS is a novel storage method that is a complement to DCS, LS, and ES, and fits in well with the various sensor network applications.

6 Acknowledgment

The research of Dr. Xiuzhen Cheng is supported by NSF CAREER Award CNS-0347674.

References

[1] X. Cheng, A. Thaeler, G. Xue, and D. Chen. Tps: A time-based positioning scheme for outdoor sensor networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, HongKong China, 2004.

[2] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, Miami, Florida, 2005.

[3] W. Duckworth, P. Dunne, A. Gibbons, and M. Zito. Leafy spanning trees in hypercubes. In *Technical Report CTAG-97008 (1997)*, University of Liverpool, 1997.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, San Frncisco, CA, 1978.

[5] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of npcompleteness. W.H.Freeman and Company, 1979.

[6] A. Ghose, J. Grossklags, and J. Chuang. Resilient data-centric storage in ad-hoc wireless sensor networks. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM2003)*, Melborune, Australia, January 2003.

[7] F. Liu, X. Cheng, D. Hua, , and D. Chen. Tpss: A time-based positioning scheme for sensor networks with short range beacons. In *International Conference on Computer Networks and Mobile Computing (ICCNMC'05)*, ZhangJiaJie, Hunan, China, 2005.

[8] M. Livingston and Q. Stout. Perfect dominating sets. In *Congressus Numerantium*, volume 79, pages 187–203, 1990.

[9] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, and S. Shenker. Data-centric storage in sensornets. In *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, 2002.

[10] S. Ratnasamy, B. Karp, L. Yin, and F. Yu. Data-centric storage in sensornets with ght, a geographic hash table. In *Mobile Networks and Applications (MONET), Journal of Special Issues on* Mobility of Systems, Users, Data, and Computing: Special Issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks, volume 8, 2003.

[11] K. Seada and A. Helmy. Rendezvous regions: a scalable architecture for service location and data-centric storage in large-scale wireless networks. In *18th International Parallel and Distributed Processing Symposium*, pages 218–225, 2004.

[12] R. Tamishetty, L. H. Ngoh, and P. H. Keng. An efficient resiliency scheme for data centric storage in wireless sensor networks. In *Vehicular Technology Conference VTC 2004*, volume 4, pages 2936 – 2940, 2004.

[13] A. Thaeler, M. Ding, and X. Cheng. itps: An improved location discovery scheme for sensor networks with long range beacons. In *Special Issue on* Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks *of* Journal of Parallel and Distributed Computing, 2004.

[14] K. Xing, X. Cheng, and J. Li. Lcs: Location-centric storage in sensor networks. In *The 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 492–501, Washington, DC, 2005.

[15] K. Xing, M. Ding, X. Cheng, and S. Rotenstreich. Safety warning based on highway sensor networks. In *IEEE Wireless Communication and Networking Conference (WCNC)*, New Orleans, Louisiana, 2005.