Wireless Netw (2010) 16:955–967
DOI 10.1007/s11276-009-0181-2

Location-centric storage and query in wireless sensor networks

Kai Xing · Xiuzhen Cheng · Jiang Li · Min Song

Published online: 20 May 2009

Abstract Location-centric storage (LCS) is envisioned as a promising scheme for robust and user-friendly on-demand data storage in networking environments such as the roadway sensor networks [Xing K et al. J Parallel Distributed Comput 67:336–345, 2007; Xing K et al. in: IEEE wireless communication and networking conference (WCNC), 2005]. In this paper, we analyze the performance of LCS in terms of storage and query overheads. This study indicates that LCS utilizes network resource efficiently and achieves good scalability. In particular, the storage overhead of sensors is independent of the network size, and is evenly distributed across the network. We also propose two algorithms for data retrieval in LCS-enabled one-dimensional and two-dimensional sensor networks. Our algorithms guarantee that acquiring the stored data of any event only takes a small number of communication hops to query a small number of sensors.

The research of Dr. Xiuzhen Cheng is supported by NSF CAREER Award CNS-0347674. The research of Dr. Min Song is supported by NSF CAREER Award CNS-0644247.

K. Xing (✉) · X. Cheng
Computer Science, The George Washington University, Washington, DC 20052, USA
e-mail: kaix@gwu.edu

X. Cheng
e-mail: cheng@gwu.edu

J. Li
Systems & Computer Science, Howard University, Washington, DC 20059, USA
e-mail: lij@scs.howard.edu

M. Song
Electrical & Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA
e-mail: msong@odu.edu

Keywords Storage in sensor networks · Location-centric storage · LCS · Query

1 Introduction

So far, four data storage methods have been proposed for sensor networks. In *Local Storage* (LS), data is stored where it is generated (and therefore can only be short-lived), i.e., locally at the home sensor. In *External Storage* (ES), data is sent to an outside access point for storage as well as processing. In *Data-Centric Storage* (DCS), data is stored according to name/location. For example, a data centric storage scheme [10] based on geographic hash tables [8] 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 [4, 8–11]. The study indicates that no one outperforms the other two in all application scenarios. The fourth storage method is termed *Location-Centric Storage* (LCS), which complements DCS, LS, and ES.

In LCS, event data is 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 has, and the farther away from the home location the event is stored. The intensity value of an event depends on the application scenarios and the users' demands.[1] For example, a collision accident on highway

[1] We are not going to discuss how to determine the intensity value of an event as it is beyond the scope of this paper. The impacts of intensity values under different application scenarios will be investigated in our future research.

may deserve the highest intensity and this event information should be propagated miles away in order to keep drivers nearby being alerted, while a collision accident on a local road is usually as severe, and this event is not necessarily sent far away because drivers at miles away may not need this information. The basic concept of LCS has been applied to the one-dimension sensor network mimicking a unidirectional highway for safety warning [15]. The generalization to roadways with intersections has been reported in [14]. Two-dimension LCS has been proposed in [13]. Compared to LS, ES, and DCS, LCS is based on a completely different concept, and is more context-aware.

In this paper, we analyze the performance of LCS in terms of storage and query overhead through both theoretical analysis and simulations. Our major contributions are summarized as follows.

- We analyze the performance of LCS on storage overhead. The results indicate the fairness of LCS on storage utilization, therefore no storage hot spots will be generated. In addition, the storage consumption at each sensor is independent of the network size, thereby LCS scales well to large networks. Such locality property testifies the efficiency of LCS in resource (power, bandwidth, memory, etc.) consumption, which is further verified by our simulation study.
- We theoretically study the query overhead for a user to obtain the relevant information in his/her close proximity. The study shows that the overhead is low. Indeed, with known location information, the queries can be sent to the right sensor hosting the required information deterministically, thus significantly conserves network resources such as bandwidth and battery power.
- We also propose algorithms to effectively and efficiently query a small subset of sensors to retrieve all the information stored in an one-dimension or two-dimension sensor network, which is important for surveillance sensor networks. To obtain the set of sensors to query, we derive a hypercube from the original network graph, and then apply the single-error-correcting code (Hamming-code) to compute a small dominating set, which contains all the stored information. Our analysis indicates that the computed dominating set is at most twice of the optimal subset that should be queried in order to retrieve all the information.

The rest of the paper is organized as follows. First we briefly overview the location-centric storage protocol for sensor networks in Sect. 2. We then conduct theoretical performance analysis on storage and query overheads in Sect. 3. We also propose two algorithms for one-dimension and two-dimension sensor network data retrieval and study their performance in Sect. 4. Simulation results are reported in Sect. 5. The paper is concluded in Sect. 6 with a discussion of the four storage methods.

2 Overview of location-centric storage

In this section, we briefly overview the basic concept of LCS. One-dimension LCS was first introduced in [15] for roadway safety warning. LCS for general sensor networks was proposed in [13].

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, 6, 12]. We further assume that a robust broadcasting protocol is in place such that information can be properly disseminated.

When detecting an event, the home sensor[2] 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 with its home sensor. Note that the *time* and *location* fields together uniquely identifies an event record. Here we assume that there is at most one event occurs at a specific location at any instant of time.
- An integral *intensity* value (σ) that characterizes the event. Intensity values are application-specific. For example, if the event is a car crash [15] in roadway safety warning, the intensity value could characterize the time needed to clear the road. In context-aware facility query [13], the intensity value indicating the availability of a gas station may be proportional to the price the owner would like to pay for this service. Generally speaking, the higher the intensity, the wider area the record should be dispatched to; 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"[3] to the event location. Thus each sensor is able to locally and independently determine whether it should drop or store the received event record.

[2] 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.

[3] 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.

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 $\in \{x + 2^0, x + 2^1, x + 2^2, \ldots, x + 2^{\sigma-1}\}$, and (b) its Y coordinate $\in \{y + 2^0, y + 2^1, y + 2^2, \ldots, y + 2^{\sigma-1}\}$, 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.

It should be noticed that for a particular pair of (i, j), where $i, j \in \{0, 1, 2, \ldots, \sigma - 1\}$, 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, fewer sensors in a smaller area store the copies of the record.

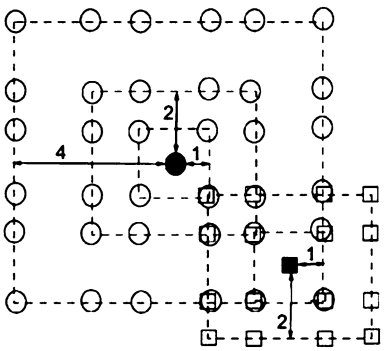


Fig. 1 An example of location-centric storage scenario. All circles store the event record for the event at the solid dot, whose intensity value is 3; And all squares keep a copy of the event record for the event at the solid square, whose intensity value is 2

3 Storage and communication overheads analysis

Due to its simple structure, the LCS 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.*
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 $\{(A_x \pm 2^i, A_y \pm 2^j) | i, j \in \{0, 1, \ldots, \sigma_A - 1\}\}$. 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, ∞). 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

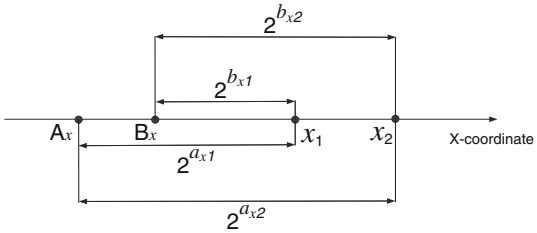


Fig. 2 Two nodes at $(x1, y1)$ and $(x2, y2), x1, x2 \in (B_x, \infty)$

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

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

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

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

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

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

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

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

Equation 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, Eqs. 8 and 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.

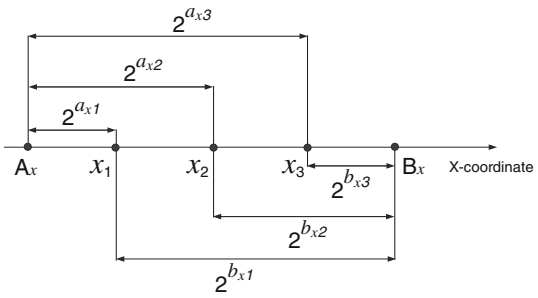


Fig. 3 Three nodes at $(x1, y1)$, $(x2, y2)$ and $(x3, y3), x1, x2, x3 \in [A_x, B_x]$

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 four 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 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 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 λ $(2\sigma + 1)^2$ $T = \lambda(4\sigma^2 + 4\sigma + 1)T$. □

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, the LCS 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 an 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 $i = 0, 1, \ldots, 2^{\sigma-1}$, needs to broadcast once; and each of the sensors at $(x \pm 2^i, y \pm j)$, where $i = 0, 1, \ldots, \sigma - 1$ and $j = 0, 1, \ldots, 2^{\sigma-1} - 1$, 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 [8, 11], 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 $y \in [S_y - 2^{\sigma-1}, S_y + 2^{\sigma-1}]$, the average query distance d_q is:*

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

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

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

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 $i = 0, 1, \ldots, \sigma - 1$, and $b = 0$ indicates that $y = S_y \pm 2^j$ for $j = 0, 1, \ldots, \sigma - 1$. 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

$$d_q = \frac{\int_0^a \int_0^b \sqrt{x^2 + y^2} dx dy}{ab} = \frac{\sqrt{a^2+b^2}}{3} + \frac{a^2 \ln\left(\frac{\sqrt{a^2+b^2}+b}{a}\right)}{6b} + \frac{b^2 \ln\left(\frac{\sqrt{a^2+b^2}+a}{b}\right)}{6a}$$

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

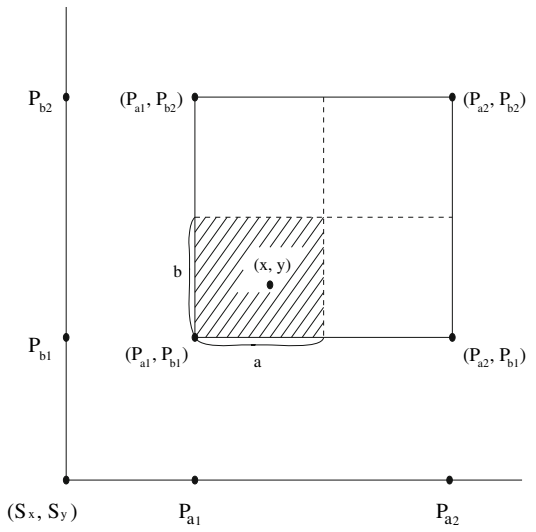


Fig. 4 For a user at (x, y) in the shaded area, the record of the event at (S_x, S_y) will be provided by the node at (P_{a1}, P_{b1}). The query distance is thus the distance from (x, y) to (P_{a1}, P_{b1}). When the user is closer to (P_{a1}, P_{b2}), (P_{a2}, P_{b1}) or (P_{a2}, P_{b2}), the calculation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

(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 the LCS 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. This is characterized by Theorem 3.5.

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,*

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

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 LCS query performance analysis

In this section, we study the query performance of LCS. 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. In the following, we first propose a method for data retrieval in a one-dimension network, and then extend the basic idea to the two-dimension 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 NP-complete [3]. 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 $u \in \{0, 1, \ldots, N-1\}$, 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 [7]:

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 Query performance of LCS in one-dimension sensor networks

Assume a one-dimension network containing 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. For simplicity we focus on the special case when $N = 2^k$ and $min_{i=0}^{N-1}\{\sigma_{S_i}\} \geq k$. For the general case, the one-dimension network can be partitioned into multiple subnetworks satisfying the above conditions, and the performance can be analyzed accordingly.

A graph $G(V, E)$ can be constructed by setting $V = \{S_0, S_1, S_2, S_3,\ldots, S_{N-1}\}$ 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 $i \in \{0, 1, \ldots, k-1\}$. 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. Since computing a minimum dominating set is NP-complete [3], a computationally efficient algorithm in this section is to be sought to find out a good approximation. In the following, we first identify a hypercube H_k as a subgraph 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. Since $V' = V$ and $E' \subset E$, a dominating set of H_k is also a dominating set of 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-correcting code. Let's consider the cases of $k = 2^t - 1$ and $k \neq 2^t - 1$ separately, where t is a non-negative integer.

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

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 $t \in \{0, 1, \ldots\}$, H_k has a perfect dominating set containing exactly $\frac{2^k}{k+1}$ nodes.*

Proof For a (k, t)-Hamming codeword $b_1\, b_2 \ldots b_k$, the bit positions b_i satisfying $i = 2^j$ with $j = 0, 1, \ldots, t-1$ 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-correcting 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 $\frac{N}{2k} = \frac{N}{2\log_2 N}$, which means that $OPT \geq \frac{N}{2\log_2 N}$. According to Theorem 4.3, the size of the perfect dominating set computed based on the construction of a (k, t)-Hamming code is $\frac{2^k}{k+1} = \frac{N}{k+1} = \frac{N}{\log_2 N+1} \leq \frac{N}{\log_2 N} \leq 2 \cdot OPT$. □

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

When $k \neq 2^t - 1$, we can extend the binary representation of each node in H_k from k bits to k' bits, where $k' = 2^{t'} - 1$ and $t' = \lceil \log_2(k+1) \rceil$. Therefore, we can compute the (k', t')-Hamming code and find the corresponding dominating set in the network. The following theorem maps the (k', t')-Hamming code to a dominating set D of H_k when $k \neq 2^t - 1$.

Lemma 4.5 *Given a k-dimension hypercube H_k, where $k \neq 2^t - 1$. Let $t' = \lceil \log_2(k+1) \rceil$ and $k' = 2^{t'} - 1$. Let $B_u = b_1^u b_2^u \cdots b_k^u 0 \cdots 0$ denote the k'-bit binary representation of vertex u in H_k, where the bit positions $b_{k+1}^u, b_{k+2}^u, \ldots, b_{k'}^u$ are set to 0. In other words, B_u is obtained*

by zero-extending the k-bit binary representation of u in H_k to k′-bits. Let C be the (k′, t′)-Hamming code. Let $D' \subset C$ be the set of Hamming codewords $b'_1 b'_2 \cdots b'_k b'_{k+1} \cdots b'_{k'}$ whose binary segment $b'_{k+1} b'_{k+2} \cdots b'_{k'}$ contains at most one bit 1. *We claim that*

- *for ∀ vertex u in H_k, there exists one codeword in D′ such that the Hamming distance between this codeword and B_u is at most* 1;
- *the size of D′ is at most $2^{k-t'}$.*

Proof For contradiction we assume that there exists a vertex v such that the Hamming distance between B_v and any codeword in D' is larger than 1. Since C is the (k', t')-Hamming code, there exists a codeword $w \in C$ such that the Hamming distance between w and B_v is at most 1. Therefore $w \notin D'$. Thus w must have at least two 1's in its binary segment $b^w_{k+1} b^w_{k+2} \cdots b^w_{k'}$. Note that the binary segment $b^v_{k+1} b^v_{k+2} \cdots b^v_{k'}$ of B_v are all zeros, therefore the Hamming distance between B_v and w is at least 2, which contradicts the previous derivation that the Hamming distance between B_v and w is at most 1.

Note that a Hamming codeword in C contains t' check bits. From the definitions of k' and t', these t' check bits reside in the least significant k-bits of the codeword. Since $D' \subset C$, no two codewords in D' have the same least significant k-bits. Therefore $|D'| \leq 2^{k-t'}$. □

Note that for each codeword $w' \in D'$, there exists a vertex u in H_k such that the least significant k bits of B_u is the same as those of w'. Let D be the set of vertices satisfying this condition.

Theorem 4.6 *D is a dominating set of H_k.*

Proof Let u be an arbitrary vertex in H_k. Based on Lemma 4.5, there exists w in D' such that the Hamming distance between w and B_u is at most 1. If the most significant $k' - k$ bits of w has the bit 1, then the least significant k bits of B_u and w must be the same; therefore $u \in D$. If the most significant $k' - k$ bits of w contain bit 0 only, B_u either equals w, or differ by 1 bit position with w. In the first case, $u \in D$; in the second case, u is adjacent to some node in D. Therefore D is a dominating set of H_k. □

Theorem 4.7 *The size of D is at most* 2·*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 From the construction of D, we have $|D| \leq |D'| \leq 2^{k-t'} = \frac{N}{2^{\lceil \log_2(k+1) \rceil}} \leq \frac{N}{k+1} \leq \frac{N}{k} = \frac{N}{\log_2 N}$. By the same argument as that in Theorem 4.4, we have $OPT \geq \frac{N}{2 \log_2 N}$. Therefore $|D| \leq 2 \cdot OPT$. □

4.2.3 Algorithm

Algorithm 1 summarizes the procedure of finding a small subset of nodes to query in the one-dimension LCS network when $N = 2^k$ 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,

Algorithm 1

Input: Node set $S_1, S_2, \ldots, S_N$.

Output: D, the dominating set in the network.

1: **function** D=**nodeSelect1D**$(S_1, S_2, \ldots, S_N)$
2: $G(V, E) \leftarrow S_1, S_2, \ldots, S_N$ ▷ Construct $G(V, E)$ to model the one-dimension LCS network.
3: $H_k \leftarrow G(V, E)$ ▷ Compute the Hypercube H_k as a subgraph of G based on Lemma 4.2.
4: **if** $k = 2^t - 1$ **then**
5: $HCode(k, t)$ =(k,t)-Hamming code ▷ Compute the (k, t)-Hamming code.
6: $D \leftarrow H_k(HCode(k, t))$ ▷ Directly map the Hamming codewords to the corresponding nodes in the hypercube H_k.
7: **else**
8: $t' = \lceil log_2(k + 1) \rceil$
9: $k' = 2^{t'} - 1$
10: $HCode(k', t')$ =(k',t')-Hamming code ▷ Compute the (k', t')-Hamming code.
11: $D' \leftarrow H_k(HCode(k', t'))$ ▷ Find the Hamming codewords that have the Hamming distance 1 to the binary representation of the vertices of the hypercube H_k.
12: $D \leftarrow D'$ ▷ $b_1 b_2 \cdots b_k 0 \cdots 0 \leftarrow b_1 b_2 \cdots b_k b_{k+1} \cdots b_{k'}$, map the Hamming codewords in D' to the corresponding nodes in the hypercube H_k.
13: **end if**
14: **end function**

0101010, 0101101, 0110011, 0110100, 1001011, 1001100, 1010010, 1010101, 1100001, 1100110, 1111000, 1111 111}. Therefore, we need to query only 16 sensors (S_0, S_7, S_{25}, S_{30},…) in order to retrieve all event information within the one-dimension LCS network of 128 nodes.

4.3 Query performance of LCS in two-dimension sensor networks

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

Theorem 4.8 *All the information in the user-queried area $N \times N$ is covered by the queried sensors in the set D.*

Proof It is obvious that the information stored by sensor $S_{a,b}$ at (a, b) is covered by the sensors in the set D if $a \in D_{X,j}$ and $b \in D_{i,Y}$, namely $S_{a,b} \in D$.

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

Based on Theorem 4.8, the size of the dominating set found via Algorithm 2 is at most $\frac{N^2}{(\log_2 N+1)^2}$. Table 1 reports several example results for the case of two-dimension sensor networks. It indicates that Algorithm 2 is able to greatly reduce the number of nodes to be queried.

5 Simulation

To evaluate the LCS protocol, we have conducted simulations. In the simulations, we assume that there is a robust routing protocol for message delivery. Besides, whenever a node generates a record, the record is propagated in the network immediately.

A two-dimension grid topology and a topology with nodes uniformly distributed (referred to as random topology henceforth)[4] were considered. For the former, we used a 64×64 grid. One node is placed at the center of each grid cell. For the latter, nodes are uniformly randomly deployed in a 64×64 area. In either case, the total number of nodes is 4096. We assume that the number of records generated by each node within one second (i.e., the record generating rate) follows the Poisson distribution with the mean λ. Other settings are as follows:

- The total simulation time is 200 s.
- $\lambda = 2^i \times 10^{-3}$, where i is one of 0, 1,…, 8 for various simulations.
- The intensity value σ is randomly chosen from [0, 6].
- The TTL value is randomly chosen from [1, 100] in seconds.
- The TTL value decreases by 1 at every second after the record is inserted into the database.
- A record is removed from the database immediately when its TTL value reaches zero.

During the simulation, the number of records stored in each node is checked at every second. All the simulation results shown in the following are averaged over 5 runs.

To measure the performance, we use the max-vs.-average storage ratio $\rho(t)$. Let $N_i(t)$ be the number of records stored on node i ($i = 1, 2,\ldots, 4096$) at time t. For $t = 1, 2,\ldots, 200$, we define

$$M(t) = \max_i \{N_i(t)\}$$

$$A(t) = \frac{\sum_i^i N_i(t)}{4096}$$

$$\rho(t) = \frac{M(t)}{A(t)}$$

Among them, for a particular time t, $M(t)$ indicates the worst node storage consumption in the network, while $A(t)$ corresponds to the best case when all records are perfectly evenly distributed across the network. Therefore, the ratio measures the fairness of node storage of the LCS protocol.

Figure 5 shows the ratio at different simulation times for $\lambda = 0.016$, 0.032 and 0.064. The simulations for both topologies have very similar results:

- The ratio drops quickly after the simulation starts and soon becomes stable.
- The smaller the λ, the slower the ratio becomes stable.
- The smaller the λ, the larger the ratio.
- For different λ's, the ratios are not the same but fairly close to each other.

Table 1 The network size and the corresponding number of sensors 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

[4] Although we have only tested on two types of networks, based on the properties aforementioned, we believe that the simulation results can be extended to more general topologies where nodes are deployed at random with arbitrary distributions.

Algorithm 2

Input: Node set $S_{1,1}, \ldots, S_{1,N}, S_{2,1}, \ldots, S_{2,N}, S_{N,1}, \ldots, S_{N,N}$.

Output: D, the dominating set in the network.

1: **function** S=**nodeSelect2D**($S_{1,1}, \ldots, S_{1,N}, S_{2,1}, \ldots, S_{2,N}, S_{N,1}, \ldots, S_{N,N}$)
2: **for** $j = 1$ to N **do**
3: $D_{X,j} \leftarrow nodeSelect1D(S_{1,j}, \ldots, S_{N,j})$ ▷ Apply Algorithm 1 to the jth column.
4: **end for**
5: **for** $i = 1$ to N **do**
6: $D_{i,Y} \leftarrow nodeSelect1D(S_{i,1}, \ldots, S_{i,N})$ ▷ Apply Algorithm 1 to the ith row.
7: **end for**
8: $D \leftarrow \emptyset$
9: **if** $x \in D_{X,j}$ & $y \in D_{i,Y}$ **then**
10: $D = D \cup \{S_{x,y}\}$ ▷ Select sensors into the query set D.
11: **end if**
12: **end function**

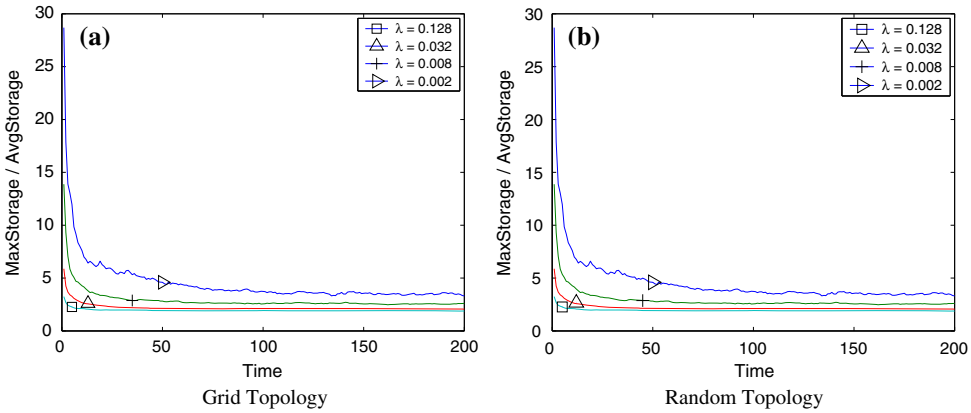


Fig. 5 The max versus average storage ratio versus simulation time for $\lambda = 0.002, 0.008, 0.032, 0.128$. **a** Grid topology, **b** Random topology

In the grid topology, the ratio becomes stable after $t = 40$ s. In the random topology, the ratio takes a little bit more time to stabilize. It owes to the randomness that makes the worst case of storage consumption (i.e., $M(t)$) volatile.

Figure 6 shows the max-vs.-average ratio versus λ, the event generating rate for the two network topologies. It is interesting to observe that when λ increases, the ratio drops below 2 quickly. It indicates that the worst case is closer to the best case with higher λ. Figure 7 shows how $M(t)$ and $A(t)$ change with time for three different λ values. We observe that both $M(t)$ and $A(t)$ become stable after $t = 100$ s. We also notice that larger λ results in larger $M(t)$ and $A(t)$. This is consistent with the expectation because larger λ means more records generated per unit time.

6 LCS, DCS, LS, and ES: a discussion

Note that our LCS protocol is a complement to ES, LS, and DCS. Comparison study of ES, LS, and DCS on the overhead of storage, query, and update[5] has been conducted in [8, 10]. In this section, we will give a brief discussion on the performance of these methods.

In ES, all detected events are directed to a central processing server such as a base station. Therefore the user query overhead is negligible while the information update overhead is a function of the network diameter. In LS, events are stored locally. User queries must be flooded to the whole network but the information update overhead is negligible. In DCS, data is stored by name/location. A geographic hash table [8] based data centric storage [10] maps the data of the same type (name) to a fixed location in the sensor network. DCS induces moderate amount of query and update overhead. In the worst case, both kinds of overhead of DCS are a function of the network diameter. For a sensor network with significant number of queries and relatively infrequent updates, ES is preferred; on the

[5] When we talk about the "overhead of query and update", we actually refer to the communication overhead induced by user query and information update. This is reasonable since in a resource-constrained sensor network, communication is the most aggressive energy consumer, which strongly impacts the network lifetime.

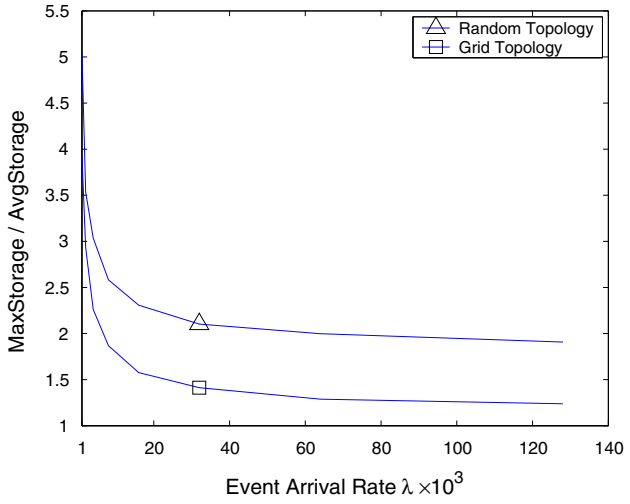


Fig. 6 The max versus average storage ratio versus $\lambda \times 10^3$

other hand, if the number of updates is far more than that of queries, LS is a better choice. DCS is applicable when the amounts of queries and updates are comparable [10].

Among LS, ES and DCS, the last one has many interesting features and deserves more discussion. A great amount of research effort [4, 8–11] has been put into it since its first introduction in the year of 2002 [10]. Compared to LCS, DCS is based on a different view of routing data, in which a type or a name instead of an IP address should be attached as data identifier. This observation holds true for many sensor network applications. For example, the measurement summary or the occurrences of some abnormal events, collaboratively computed by many sensors, should be reported to certain nodes in a sensor network deployed for monitoring and control. The concept of data-centric binds the type with the data, since in reality users care about "what has happened" instead of "which sensor observes the occurrence". In DCS, a named data is directed to a fixed location determined by a Geographic Hash Table (GHT) [8]. All data with the same names are hashed to the same position (actually to the sensors close to that position). GHT relies on a geographic routing protocol such as GPSR [5] for data dissemination, and exploits two operations, **Put**(k, v) and **Get**(k, v), for data update and query, where k is the name of the data whose value is v.

DCS suffers from the problem of single-point-of-failure, since each type of data is mapped (hashed) to exactly one location, and stored by the sensor that is geographically the nearest to the hashed location. To improve the resilience to node failure, Ratnasamy et al. [8] proposed to replicate the stored data locally through the periodic *refresh* messages; They also extended DCS to obtain Structured Replication in DCS (SR-DCS) for load-balancing and better scalability. In the Resilient DCS (R-DCS) [4], the coordinate space is partitioned into Z zones, with each containing sensors operating at different modes. An event is stored in its home zone if there exists a sensor working at the Replica Mode for that event type; Otherwise, the data is forwarded to the closest replica node for that event type in nearby zones. It is obvious that this two-level replication strategy improves both the resilience of DCS against node failure and its scalability. A different hierarchical architecture for DCS based on *Rendezvous regions* (RR) is reported in [9], in which all data with the same name are mapped to a RR region instead of a single point. The storage of the data within a region is controlled by a few elected nodes within the region. This approach can tolerate dynamics such as node mobility and has better scalability compared with the basic DCS, but it suffers from clustered node failures. To further improve resilience, Tamishetty et al. [11] proposed to employ multiple hashing such that one type of event can be replicated at multiple locations computed by different hash functions.

Compared to DCS, LCS utilizes a completely different concept. While DCS basically does m(data)-to-1(location) mapping, LCS does 1(data)-to-m(locations), i.e., an event record is replicated at multiple positions in the neighborhood of the event location based on the intensity of the event. Consequently, LCS has better scalability and stronger resilience against node failures compared with DCS.

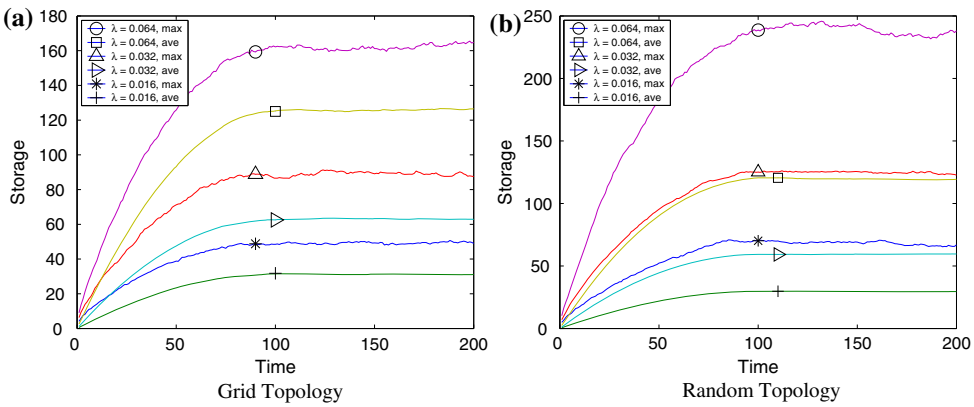


Fig. 7 *MaxStorage* and *AvgStorage* versus simulation time for $\lambda = 0.016, 0.032, 0.064$. **a** Grid topology, **b** Random topology

Note that LCS and DCS can coexist in a sensor network since they target different application scenarios. DCS is designed for large-area data dissemination. The designated storage location for a type of events 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 the user is approaching the event location. Querying a specific event in LCS requires global flooding.

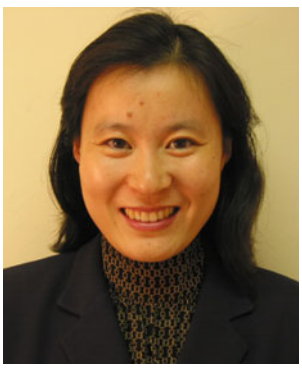
LCS is different from LS too. In LCS, a group of sensors store an event generated roughly at the geometric center of the group, while in LS a sensor stores its local observations. LCS has better resilience against node failures, and is suitable for summary data dissemination. Overall, LCS is a novel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

References

1. Cheng, X., Thaeler, A., Xue, G., & Chen, D. (2004). Tps: A time-based positioning scheme for outdoor sensor networks. In *IEEE international conference on computer communications (INFOCOM)*, HongKong, China (pp. 2685–2696).
2. Ding, M., Chen, D., Xing, K., & Cheng, X. (2005). Localized fault-tolerant event boundary detection in sensor networks. In *IEEE international conference on computer communications (INFOCOM)*, Miami, Florida (pp. 902–913).
3. Garey, M. R., & Johnson, D. S. (1978). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: Freeman.
4. Ghose, A., Grossklags, J., & Chuang, J. (2003). Resilient data-centric storage in wireless ad-hoc sensor networks. In *MDM '03: Proceedings of the 4th international conference on mobile data management* (pp. 45–62).
5. Karp, B., & Kung, H. T. (2000). Gpsr: Greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on mobile computing and networking* (pp. 243–254).
6. Liu, F., Cheng, X., Hua, D., & Chen, D. (2005). Tpss: A time-based positioning scheme for sensor networks with short range beacons. In *International conference on computer networks and mobile computing (ICCNMC'05)* (pp. 33–42).
7. Livingston, M., & Stout, Q. (1990). Perfect dominating sets. In *Congressus numerantium* (Vol. 79, pp. 187–203).
8. Ratnasamy, S., Karp, B., Yin, L., & Yu, F. (2003). Data-centric storage in sensornets with ght, a geographic hash table. *Journal of Mobile Networks and Applications, 8*, 427–442.
9. Seada, K., & Helmy, A. (2004). 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* (pp. 218–225).
10. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., & Estrin, D. (2003). *Data-centric storage in sensornets* (Vol. 33, pp. 137–142).
11. Tamishetty, R., Ngoh, L. H., & Keng, P. H. (2004). An efficient resiliency scheme for data centric storage in wireless sensor networks. In *Vehicular technology conference VTC 2004* (Vol. 4, pp. 2936–2940).
12. Thaeler, A., Ding, M., & Cheng, X. (2005). iTPS: An improved location discovery scheme for sensor networks with long range beacons. *Journal of Parallel and Distributed Computing, 65*, 98–106.
13. Xing, K., Cheng, X., & Li, J. (2005). Lcs: Location-centric storage in sensor networks. In *The 2nd IEEE international conference on mobile ad-hoc and sensor systems (MASS)*, Washington, DC (pp. 492–501).
14. Xing, K., Cheng, X., Liu, F., & Rotenstreich, S. (2007). Location-centric storage for safety warning based on roadway sensor networks. *Journal of Parallel and Distributed Computing, 67*, 336–345.
15. Xing, K., Ding, M., Cheng, X., & Rotenstreich, S. (2005). Safety warning based on highway sensor networks. In *IEEE wireless communication and networking conference (WCNC)* (pp. 2355–2361).

Author Biographies

Kai Xing is a Ph.D. Candidate in the Department of Computer Science, The George Washington University. He received the B.S. degree from University of Science and Technology of China in 2003 and the M.S. degree from The George Washington University in 2006, both in computer science. His current research interests include mesh, ad hoc and sensor networking, in-network information processing, wireless and mobile security, and algorithm design and analysis. He is a student member of the IEEE and the ACM.



Xiuzhen Cheng is an Associate Professor at the Department of Computer Science, The George Washington University. She received her M.S. and Ph.D. degrees in Computer Science from the University of Minnesota—Twin Cities in 2000 and 2002, respectively. Her current research interests include Cyber-Physical Systems, Wireless and Mobile Computing, Sensor Networking, Wireless and Mobile Security, and Algorithm Design and Analysis. Dr. Cheng has served in the editorial boards of several technical journals and in the technical program committees of various professional Conferences/workshops. She is a program co-chair or vice co-chair for several conferences such as WASA'06, MSN'09, ICPP'09, and MASS'09. Dr. Cheng worked as a program director in the National Science Foundation (NSF) for six months in 2006 and joined NSF again as a part-time program director in April 2008. She received the NSF CAREER Award in 2004.

Jiang Li received his B.S. (1995) and M.S. (1998) degrees in Computer Science from University of Science and Technology of China, and Ph.D. degree (2003) in Computer Science from Rensselaer Polytechnic Institute. He is currently an assistant professor in the Department of Systems and Computer Science at Howard University, Washington, DC. His research interests include computer networking, network security and network simulations. He has published in such areas as congestion control, multicast, sensor networks. His research now focuses on mobile ad hoc networks and disruption tolerant networks.

Min Song is an Associate Professor in the Department of Electrical and Computer Engineering at Old Dominion University. He received his Ph.D. in Computer Science from the University of Toledo in 2001. His research interests include protocols design and performance analysis of mobile ad hoc networks and wireless sensor networks, computer networks security, wireless communications, and packet switch architectures. Dr. Song is the recipient of NSF CAREER award. He received early tenure and promotion in June of 2007. He is the founder and director of the Wireless Communications and Networking Laboratory. Dr. Song is an IEEE Senior member.