Location-Centric Storage for Sensor Networks

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

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

***Abstract*— This paper presents Location-Centric Storage (LCS), a novel distributed data storage protocol for sensor networks. In the protocol, each event detected by sensors is associated with an intensity value (σ)(by sensors), where σ is a parameter that depends on the characteristics of the event and the application context. When event information is broadcast, a sensor decides whether to store the record of an event by checking its distance to the event location and the σ. In general, the higher the intensity of an event, the further its information can propagate geographically in the sensor network. Besides, the closer to the event location, the denser the sensors are that store the event information, and thus the quicker and better a user can know about the event (by reading from surrounding sensors). The protocol 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. We provide detailed theoretical analysis and simulation study to support the claims. We also ran simulations to show the advantage of our protocol over some previous work. LCS can be used for applications such as context-dependent information mining in pervasive computing and on-demand warning in surveillance sensor networks.**

Keywords: sensor networks, location-centric storage, on-demand warning, context-dependent information mining

I. Introduction

Sensor networks consist of micro sensors for monitoring and interacting with the physical world. There exist many exciting sensor network applications, including smart learning [27], battlefields [8], [9], environmental monitoring and control [3], [13], [17], [19], target detection and tracking [4], [18], [20], roadway safety warning [25], [26], etc. These applications are expected to have strong economic influence in the near future.

Nevertheless, sensor networks pose many new challenges [28], [29]. One of the challenges is how to store data efficiently to facilitate user query and on-demand warning[1] across the entire sensor network. The challenge is significant due to the per sensor resource constraints (limited battery, memory, bandwidth, etc). It is made harder by multi-hop communication and the overwhelming amount of sensor readings at any time over a wide geographic area. In this paper, we will propose a novel data storage method, termed as *location-centric storage (LCS)*, to efficiently disseminate aggregated data based on the intensity (explained later in this section) of the data. We will show that LCS can be easily applied to pervasive computing for context-dependent information query and to surveillance networks for on-demand warning.

There exists three canonical 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 [15] based on geographic hash tables [16] 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 [5], [15], [16], [21], [22]. These studies indicate that no one outperforms the other two in all situations. In fact, none of these methods targets the application scenarios considered in our LCS design. For example, on-demand warning requires zero delay and high reliability. However, the query-response delay involved in LS, ES, and DCS may not be tolerable, and the possibility of single point of failure renders them unreliable. Further, ES and DCS incur bottlenecks at certain storage locations. In contrast, in LCS, the sensor sending out warning signals are always local to users, resulting in negligible delay. Like LS, LCS well balances sensor storage load in the network, as long as the event occurrence rate in the network is uniformly distributed.

The basic idea of LCS is sketched as follows: A record is generated by a sensor (the home sensor) when detecting the occurrence of some event (e.g. adversaries, chemical spills, etc.). The record is stored in the database of the home sensor and sensors that are some distance away. Sensors decide whether to store an event record based on the distance to the home sensor and the *intensity* of the event. The intensity of an event is a function of the significance of the event, the application scenario, the physical environment, etc. For example, the intensity of a motor vehicle crash event is related to the time needed to clear the road. The intensity of the event indicating the availability of a printer in a building can be a function of the printer's capability and the owner's intention. Generally speaking, the closer to the event location, the higher the density of the sensors storing the corresponding record.

The paper has the following major contributions:

1) We propose a novel information dissemination method called LCS for sensor networks. LCS is purely localized, thus scales well to large networks. Furthermore, LCS can complement existing data storage methods. It has abundant applications such as context-dependent information mining in pervasive computing and on-demand warning

[1]It is called "on-demand warning" since these warnings will be generated only when some mobile objects are approaching.

0-7803-9466-6/05/$20.00 ©2005 IEEE

in surveillance sensor networks.
2) We identify example applications of LCS in pervasive computing and surveillance sensor networks.
3) We study the performance of LCS through detailed theoretic analysis, which indicates that LCS is efficient in communication and storage overheads.
4) We conduct extensive simulations to verify the efficiency and effectiveness of LCS. To be specific, we assess the storage overhead caused by LCS at the worst and average cases.
5) We compare the message overheads of LS, ES, and LCS through simulations. The results indicate that LCS outperforms LS and ES given varying query number, fixed event number and growing network size.

Our simulation results are consistent with what we obtain from the theoretical performance analysis. Both studies indicate that LCS can effectively disseminate event records. If events occur uniformly randomly in the whole network, the storage spaces needed are fair to all sensors.

In the rest of the paper, we will first briefly summarize related research. After that, we will present our location-centric storage protocol for sensor networks in Section III. Possible example applications of LCS are identified in Section IV. The theoretical performance analysis is given in Section V, followed by the simulation report in Section VI. We conclude this paper by Section VII.

II. Related Work

There exist three canonical data dissemination methods in sensor networks: External Storage (ES), Local Storage (LS), and Data-Centric Storage (DCS). Comparison studies of ES, LS, and DCS in the overheads of storage, query, and update[2] have been conducted in [15], [16]. In this section, we will give a brief literature survey on these methods. It is interesting to observe that our LCS protocol is a complement to ES, LS, and DCS. There also exist methods [11], [12] for reliably disseminating data from the source to the destination in harsh environments. This topic is out of the scope of our study.

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 [16] based data centric storage [15] maps the data of the same type (name) to a fixed location in the sensor network. DCS induces moderate amount of query and update overheads. In the worst case, both overheads of DCS are functions of the network diameter. For a sensor network with significant number of queries and relatively infrequent updates, ES is preferred; on the other hand, if the number of updates is far more than that of queries, LS is a better choice. DCS is applicable when the numbers of queries and updates are comparative [15].

A great amount of research effort [5], [15], [16], [21], [22] has been put to DCS since it was first introduced in the year 2002 [15]. 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 to identify the data. 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 the users 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 this occurrence". In DCS, a named data is directed to a fixed location determined by the Geographic Hash Table (GHT) [16]. 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 [7] 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 failures, Ratnasamy *et al.* [16] propose to replicate the stored data locally through the periodic *refresh* messages; They also extend DCS to obtain Structured Replication in DCS (SR-DCS) for load-balancing and better scalability. In the Resilient DCS (R-DCS) [5], 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 failures and its scalability. A different hierarchical architecture for DCS based on *Rendezvous regions (RR)* is reported in [21], 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 one 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, Ngoh, and Keng [22] propose to employ multiple hashing such that one type of event can be replicated at multiple locations computed by different hash functions.

The Location-Centric Storage (LCS) scheme proposed in this paper utilizes a completely different concept. LCS is relatively more context-aware. In LCS, events are replicated at multiple positions 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. Thus LCS has

[2]When we talk about the "overheads of query and update", we actually refer to the communication overheads 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.

better scalability and stronger resilience against node failures compared with DCS. 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. Query 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 the various sensor network applications.

The basic concept of location centric storage has been applied to an one-dimensional sensor network mimicking a unidirectional highway for safety warning [25]. The generalization of this model to roadways with intersections has been reported in [26]. Note that these works focus on the philosophy of roadway safety warning based on sensor networks. In this paper, we will formally propose LCS in the context of typical sensor networks and conduct extensive analysis to verify its performance.

III. Location-Centric Storage

As a prerequisite of our protocol, we assume that sensors can obtain their own geometric coordinates (S_x, S_y) using GPS or other techniques, such as those proposed in [2], [10], [23]. We also assume that a robust broadcasting protocol is in place such that event records can be properly disseminated.

When detecting an event, the home sensor[3] 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.
- An integral *intensity* value (σ) that characterizes the event. Intensity values are application-specific. If the event is a car crash in roadway sensor networks [25], the intensity value characterizes the time needed to clear the road in highway safety warning. In pervasive computing, the intensity value of the event 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 storing the record within the sensor.
- A *Time-To-Live* (TTL) as the expiration time (relative to the current moment) of the record. The TTL value tells the sensor storing this record when to purge the corresponding entry from its database.
- The *event type* bearing other information of the event.

Event records will be distributed and stored following our LCS protocol. In LCS, a sensor when receiving an event record computes its distance Δ to the event location and check whether Δ is "close enough"[4]. Thus each sensor is able to locally and independently determine whether it should drop or store the received event record. To free space for new events, records are purged from the database when their TTL values reach 0. When a user query is received, a response can be generated based on the information stored in its database. This procedure can be summarized as follows.

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 $\in \{x + 2^1 - 1, x + 2^2 - 1, \cdots, x + 2^\sigma - 1\}$, and (b) its Y coordinate $\in \{y + 2^1 - 1, y + 2^2 - 1, \cdots, y + 2^\sigma - 1\}$, where σ and (x, y) are the intensity value and the event location drawn from the received record. Otherwise, the record is dropped. In both cases, the sensor broadcasts the record if its distance to the event location is less than 2^σ in both X dimension and in Y dimension.
3) After a record is inserted into the database of some sensor, its TTL value starts to decrease as its local clock ticks, and 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 database will be generated.

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

From the algorithm above, it is 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 the intensity of 3. Therefore, its record is stored in a sensor whose horizontal and vertical distances to the dark dot are members of the set $\{1, 3, 7\}$ (corresponding to $2^1 - 1, 2^2 - 1$ and $2^3 - 1$ respectively). Similarly, the other event detected at the solid square has the intensity of 2. In this case, fewer sensors in a smaller area store the record.

IV. Example Applications

The design of our LCS location-centric storage protocol is triggered by real-world applications. In this section, we study several example scenarios to which LCS can be applied.

[3] An event is usually detected by multiple sensors simultaneously but one sensor will be designated for reporting the event [1], [3]. We term this sensor the "home sensor" of the event.

[4] 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.

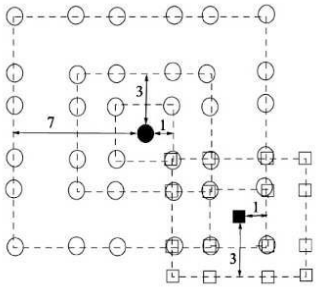


Fig. 1. An Example Location-Centric Storage Scenario. All circles store the event record for the event at the solid dot, whose intensity value is 3; And all squares keep a copy of the event record for the event at the solid square, whose intensity value is 2.

A. Context-Dependent Information Dissemination for Pervasive Computing

Mark Weiser devised a background computing paradigm in his 1991 seminar paper "The Computer for the 21st Century" [24], in which technologies "weave themselves into the fabric of everyday life until they are indistinguishable from it". Ever since then, the research on pervasive (ubiquitous) computing has been flourishing. A very important topic in this area is data dissemination [6], [14], which should be able to facilitate the query of timely and context-aware data. LCS as a data dissemination method can do the job well. In this context, sensors with moderate amount of storage supplies can be deployed along roadways, within buildings, campuses, etc. Each sensor works as a mini-sized database. Therefore the availability and location information of a gas station can be disseminated based on the LCS protocol, where the intensity can be determined by jointly considering the distance to other gas stations, the amount of money the manager would like to pay, and the environment (food stores, lodging facilities, etc., in the neighborhood). Thus a driver in a foreign area can easily identify the nearest gas station that is in good condition and best fits her need. Similarly the availability of public printers in a building can be announced based on LCS. Thus a conference attendant can modify and then print out her handouts from the nearest printer just before her talk starts.

B. On-Demand Warning in Surveillance Sensor Networks

If sensors have the ability to generate warning messages for the stored events, LCS can be exploited for on-demand warning[5] in surveillance networks. To realize this, users must be equipped with a device, which could be a very simple sensor, that is able to communicate with the sensors in the network. In this scenario, sensors can detect the approaching of the mobile objects (people or enemy trucks, etc.). In the following, we identify several simple applications of LCS for on-demand warning.

In military, it is not hard to sprinkle a large amount of micro sensors on the battle field during a war. These sensors can be used to detect land mines, enemy tanks [4], etc. LCS can be employed for on-demand warning. A scout can be alerted early when he is walking toward a land mine. With very low probability he will miss the second or the third warning signal for the same land mine if the first one has been missed. And he can choose a better route toward his target based on the number of received warnings. Similarly, if a sensor network is deployed for protecting an ammunition depot, LCS can help to detect enemies that are approaching. The closer the enemy, the more number of warning messages the depot will receive, based on LCS.

Even for sensor networks installed for civilian usage, LCS can get a foot in. For example, in a sensor network deployed for monitoring the activities of an active volcano, LCS can be used to warn pedestrians, travellers, and scientists that are doing in-situ research. The transport of chemical spills [3], pollutants, and other harmful materials can be observed through a sensor network. With LCS, human beings when approaching these poisoned areas can be alerted. LCS guarantees that the closer to the event location, the more number of warning messages, and the less chance that a victim misses all warnings.

C. Roadway Safety Warning

The "Zero Fatality, Zero Delay"highway safety philosophy has been proposed at the World Congress on ITS (Intelligent Transportation Systems and Services) in Madrid, Spain in the year 2003. This indicates that future transportation systems should be able to minimize avoidable delays, injuries and fatalities by integrating various advanced technologies. We have proposed our exploratory work toward safety warning based on roadway sensor networks in [25]. In this research, we consider an one-dimensional network that mimics a single direction of a highway with sensors uniformly deployed. Event records for car crashes, fogs, slippery roads, etc., are constructed and propagated toward the against traffic direction. Warning messages are generated by a device inside the vehicle that can obtain information from roadway sensors when drivers passing by. This idea is similar to the placement of exit signs along the highway, in which a driver can observe more number of signs when he is approaching the exit. In this work [25], we have employed a simplified version of LCS, in which the sensor network is modelled as an one-dimensional line graph.

V. Performance Analysis

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

Theorem 5.1: Given two records *A* and *B* produced by two nodes at different locations (A_x, A_y) and (B_x, B_y) respectively,

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 σ is the smaller intensity value among the two.

[5]It is called "on-demand warning" since these warnings will be generated only when some mobile objects are approaching.

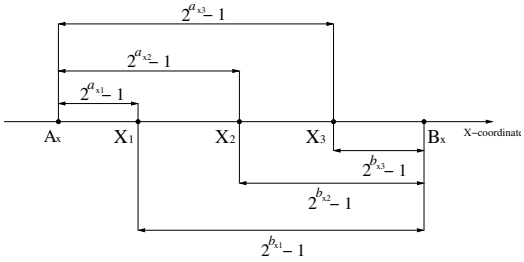


Fig. 2. Two nodes at $(x1, y1)$ and $(x2, y2)$, $x1, x2 \in [B_x, \infty)$.

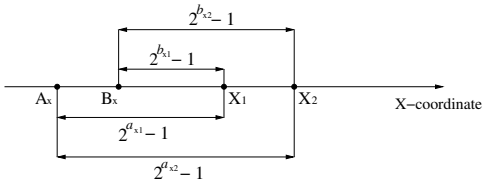


Fig. 3. Two nodes at $(x1, y1)$ and $(x2, y2)$, $x1, x2 \in [A_x, B_x]$.

Proof: We assume that both records are alive at the same time (since otherwise no nodes will store both of them). Let the intensities of the corresponding events be σ_1 and σ_2 respectively. The storage locations for record *A* are then $\{(A_x \pm (2^i - 1), A_y \pm (2^j - 1)) | i, j \in \{1, 2, \ldots, \sigma_1\}\}$. 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

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

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 Eq. (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 $[B_x, \infty)$ 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

$$x1 = A_x + 2^{a_{x1}} - 1 = B_x - 2^{b_{x1}} + 1 \quad (4)$$
$$x2 = A_x + 2^{a_{x2}} - 1 = B_x - 2^{b_{x2}} + 1 \quad (5)$$
$$x3 = A_x + 2^{a_{x3}} - 1 = B_x - 2^{b_{x3}} + 1 \quad (6)$$

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 pairs of coordinates 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 σ is the smaller intensity value in the event records. 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 $A_y = B_y$.

The proof completes. ■

Corollary 5.1: 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 algorithm, the node can only store event records generated by nodes at $(S_x \pm (2^i - 1), S_y \pm (2^j - 1))(i, j \in \{0, 1, 2, \ldots, \sigma\})$. Therefore, we can reverse the roles of records and nodes in the proof of the Theorem 5.1, which leads to the corollary. ■

Remark: Theorem 5.1 indicate 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. 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 5.2. Corollary 5.1 shows that records are distributed among all nodes instead of converging onto some of them. Therefore, no hot spots will be created.

Theorem 5.2: Assume broadcast is instantaneous. Denote the average intensity value of records as σ, the average TTL value as T (assumed as an integer). Also assume that at any node, the number of events detected during the unit time, N, follows a Poisson distribution with the 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 $x = S_x \pm (2^i - 1)$ and $y = S_y \pm (2^j - 1)$ for $i, j \in \{0, 1, \ldots, \sigma\}$) during the time interval $[t-T, t]$. Let $N_k^{x,y}$ be the number of events for which the node at (x, y) generate 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 as $\lambda(2\sigma+1)^2 T = \lambda(4\sigma^2+4\sigma+1)T$. ■

Remark: Note from Theorem 5.2 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, and is thus highly scalable.

Theorem 5.3: 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 $(2^{\sigma+1}-3)^2$ times. With a careful broadcast scheduling, this upper bound can be reduced to $(2\sigma+2)\times(2^{\sigma+1}-4)+1$.

Proof: According to our protocol, an 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 $(2^{\sigma+1}-3)^2$.

This upper bound can be improved if the record is propagated horizontally first. $2\sigma+1$ nodes on the horizontal line then store the record. After that, each of these nodes broadcasts the record vertically. The number of horizontal broadcasts is $2^{\sigma+1}-3$, and that of vertical broadcasts is $(2\sigma+1)\times(2^{\sigma+1}-3)$. In fact, $2\sigma+1$ horizontal broadcasts (from those nodes storing records) can also serve as vertical broadcasts. The total number of broadcasts is thus

$$\begin{aligned} & 2^{\sigma+1}-3+(2\sigma+1)\times(2^{\sigma+1}-3)-(2\sigma+1) \\ = \;& (2\sigma+1)\times(2^{\sigma+1}-4)+2^{\sigma+1}-3 \\ = \;& (2\sigma+2)\times(2^{\sigma+1}-4)+1 \end{aligned}$$

■

Remark: From Theorems 5.2 and 5.3, 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 [16], [22], which creates storage hot spot even when the number of events in the network is low.

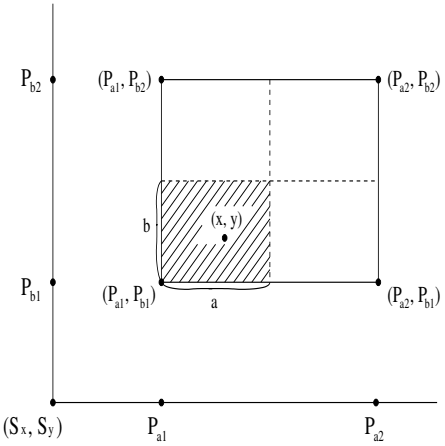


Fig. 4. For a user at (x, y) in the shaded area, the record of the event at (S_x, S_y) will be provided by the node at (P_{a1}, P_{b1}). The query distance is thus the distance from (x, y) to (P_{a1}, P_{b1}). When the user is closer to (P_{a1}, P_{b2}), (P_{a2}, P_{b1}) or (P_{a2}, P_{b2}), the calculation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

Theorem 5.4: 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:

$$\begin{cases} \frac{\sqrt{a^2+b^2}}{3} + \frac{a^2 \ln(\frac{\sqrt{a^2+b^2}+b}{a})}{6b} \\ \quad + \frac{b^2 \ln(\frac{\sqrt{a^2+b^2}+a}{b})}{6a} & \text{if } x \neq S_x \text{ and } y \neq S_y \\ \frac{\sqrt{a^2+b^2}}{2} & \text{otherwise} \end{cases}$$

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

$$P_{a1} = S_x + 2^{\lfloor \log_2 d_x \rfloor} - 1,\ P_{a2} = S_x + 2^{\lceil \log_2 d_x \rceil} - 1$$
$$P_{b1} = S_y + 2^{\lfloor \log_2 d_y \rfloor} - 1,\ P_{b2} = S_y + 2^{\lceil \log_2 d_y \rceil} - 1$$

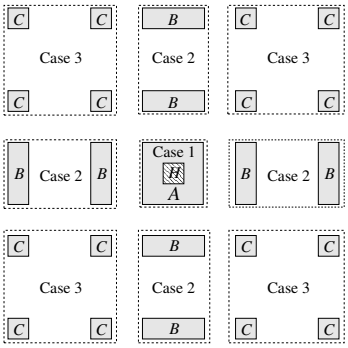


Fig. 5. Areas in Which Event Information Can Reach The User

Therefore, $a = (P_{a2} - P_{a1})/2$, $b = (P_{b2} - P_{b1})/2$. 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 send the query to the node at that point. Whichever point he 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

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

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 $0 = \frac{\sqrt{a^2+b^2}}{2}$.

Proof completes. ■

Remark: Theorems 5.4 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 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 nodes within the neighborhood area, called by us 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 5.5: Assume all events have the same intensity value σ, and the sensor density is very high (such that any event can be recorded by a sensor at the same location). Suppose a user at (x, y) can only communicate with any node within the area ($\mathcal{H}$) of $l \times l$ centered at (x, y). Let $\beta = \min(\sigma, \lfloor \log_2(l+1) \rfloor)$. If an event occurs at (S_x, S_y),

1) If $|S_x - x| \leq 2^\sigma - 1 + l/2$ and $|S_y - y| \leq 2^\sigma - 1 + l/2$, the probability that the user is notified of the event is
$$\begin{cases} 1 & \text{if } \beta = \sigma \\ \left(\frac{2^{\beta+1}+2l(\sigma-\beta)+l-2}{2^{\sigma+1}+l-2}\right)^2 & \text{if } \beta < \sigma \end{cases}$$
2) Otherwise, the user cannot be notified of the event.

Proof:

Let $d_x = |S_x - x|$, and $d_y = |S_y - y|$. There are four cases:

Case 1: $d_x \leq 2^\beta - 1 + l/2$ and $d_y \leq 2^\beta - 1 + l/2$.
The area that consists of all (S_x, S_y) satisfying this condition is a square centered at (x, y) (Fig. 5 case 1). The side of the square is $2 \times (2^\beta - 1 + l/2) = 2^{\beta+1} + l - 2$, and the size is thus $(2^{\beta+1} + l - 2)^2$. Denote the square as $\mathcal{A}$. From any point in $\mathcal{A}$ to $\mathcal{H}$, the shortest distance (either horizontal or vertical) is no more than $2^\sigma - 1$. Since the user can communicate with any node in $\mathcal{H}$, he can get the information of any event in $\mathcal{A}$.

Note that when $\beta = \sigma$, the area of $\mathcal{A}$ contains all events of which the user can be notified. Therefore, in this case, the probability of notification is 1.

Case 2: $d_x \leq 2^\beta - 1 + l/2$ and $2^\beta - 1 + l/2 < d_y \leq 2^\sigma - 1 + l/2$, or $2^\beta - 1 + l/2 < d_x \leq 2^\sigma - 1 + l/2$ and $d_y \leq 2^\beta - 1 + l/2$. Not all information of events at such (S_x, S_y) are covered. For example, if an event occurs at $x + l/2 + 2^\beta - 1 + \epsilon, y$ (where ϵ is a small fractional value), the user won't receive the information about the event. The covered area $\mathcal{B}$ under this condition is composed of the following non-contiguous rectangles, as shown in Fig. 5 case 2:

- $S_x \in [x - l/2 \pm 2^{\beta+i} \mp 1, x + l/2 \pm 2^{\beta+i} \mp 1]$,
  $S_y \in [y - l/2 - 2^\beta + 1, y + l/2 + 2^\beta - 1]$;
  or,
- $S_x \in [x - l/2 - 2^\beta + 1, x + l/2 + 2^\beta - 1]$,
  $S_y \in [y - l/2 \pm 2^{\beta+i} \mp 1, y + l/2 \pm 2^{\beta+i} \mp 1]$.

where $i = 1, 2, \ldots, \sigma - \beta$. Each rectangle is of the size $l \times (2^{\beta+1} + l - 2)$, and there are totally $4(\sigma - \beta)$ such rectangles. The totally size of $\mathcal{B}$ is thus $4l(\sigma - \beta)(2^{\beta+1} + l - 2)$.

Case 3: $2^\beta - 1 + l/2 < d_x, d_y \leq 2^\sigma - 1 + l/2$. Similar to case 2, the covered area $\mathcal{C}$ under this condition is composed of the following non-contiguous squares, as shown in Fig. 5 case 3:

- $S_x \in [x - l/2 \pm 2^{\beta+i} \mp 1, x + l/2 \pm 2^{\beta+i} \mp 1]$, $S_x \in [y - l/2 + 2^{\beta+j} - 1, y + l/2 + 2^{\beta+j} - 1]$; or,
- $S_y \in [x - l/2 \pm 2^{\beta+i} \mp 1, x + l/2 \pm 2^{\beta+i} \mp 1]$, $S_y \in [y - l/2 - 2^{\beta+j} + 1, y + l/2 - 2^{\beta+j} + 1]$.

where $i, j = 1, 2, \ldots, \sigma - \beta$. Each square is of the size l^2, and there are totally $4(\sigma - \beta)^2$ of them. The total size of $\mathcal{C}$ is thus $4l^2(\sigma - \beta)^2$

Case 4: For all other values of d_x and d_y, the user won't be able to receiver the event information, since no sensor in the user communication area will have the information. In particular, if $d_x, d_y > 2^\sigma - 1 + l/2$, even the sensor farthest from the event location cannot reach the user communication area.

In summary, if $d_x, d_y \leq 2^\sigma - 1 + l/2$, from the discussion above, we know that the areas covered by the user communication area $\mathcal{H}$ include $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. The total size of them is $(2^{\beta+1} + l - 2)^2 + 4l(\sigma - \beta)(2^{\beta+1} + l - 2) + 4l^2(\sigma - \beta)^2 = (2^{\beta+1} + 2l(\sigma - \beta) + l - 2)^2$. On the other hand, the area corresponding to $d_x, d_y \leq 2^\sigma - 1 + l/2$ has the size of $(2^{\sigma+1} + l - 2)^2$. Therefore, given an event in this area, this probability that the user can be notified of it is

$$\frac{2^{\beta+1} + 2l(\sigma - \beta) + l - 2)^2}{(2^{\sigma+1} + l - 2)^2} = \left(\frac{2^{\beta+1} + 2l(\sigma - \beta) + l - 2}{2^{\sigma+1} + l - 2}\right)^2$$

■

VI. Simulation

We have run simulations to evaluate our protocol. In these simulations we have considered the two-dimensional grid topology and the topology with nodes uniformly distributed (referred to as random topology henceforth)[6]. These simulations assume that message delivery is instantaneous and reliable. Besides, whenever a node generates a record, the record is propagated in the network immediately.

A. LCS Performance Evaluation

For the grid topology, we used a 64×64 grid. One node is placed in the middle of each grid cell. For the random topology, 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 during one second (i.e. the record generating rate) follows the Poisson distribution with the mean as λ. The simulation setup is as follows:

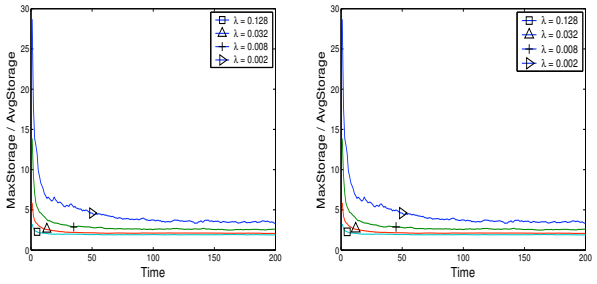
- The total simulation time is 200 seconds.
- $\lambda = 2^i \times 10^{-3}$, where i is one of $0, 1, \cdots, 8$ for various simulations.
- The intensity σ is randomly chosen from $[0, 6]$.
- The TTL value is randomly chosen from $[1, 100]$ in seconds.
- The TTL value decreases by 1 every second after 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 are tracked 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 by node i ($i = 1, 2, \cdots, 4096$) at time t. For $t = 1, 2, \cdots, 200$, we define

$$\begin{aligned} M(t) &= \max_i\{N_i(t)\} \\ A(t) &= \frac{\sum_i N_i(t)}{4096} \\ \rho(t) &= \frac{M(t)}{A(t)} \end{aligned}$$

Among them, for a particular time t, $M(t)$ indicates the worst node storage consumption in the network, while $A(t)$ indicates the best case when all records are prefectly evenly distributed among all nodes in the network. Therefore, the ratio measures the fairness of node storage of our location-centric storage protocol.



(a) Grid Topology (b) Random Topology

Fig. 6. The Max-vs-average Storage Ratio *vs.* Simulation Time for $\lambda = 0.002, 0.008, 0.032, 0.128$.

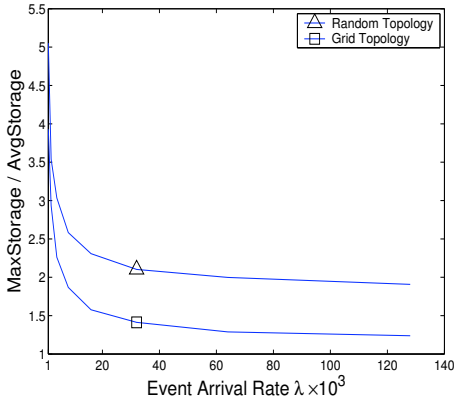
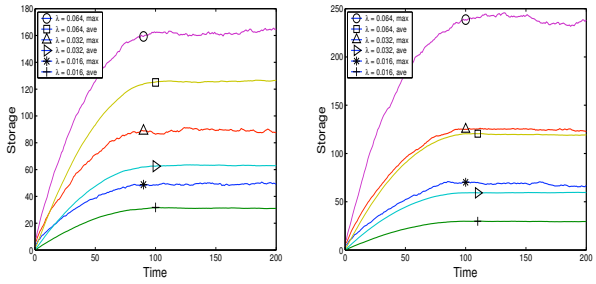


Fig. 7. The Max-vs-average Storage Ratio *vs.* $\lambda \times 10^3$

[6]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.



(a) Grid Topology (b) Random Topology

Fig. 8. $MaxStorage$ and $AvgStorage$ *vs.* Simulation Time for $\lambda = 0.016$, 0.032, 0.064

Fig. 6 shows the relationship between the ratio and simulation time 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.

In the grid topology, the ratio becomes stable after $t = 40s$. 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.

Fig. 7 shows the max-vs-average ratio versus λ, the event generating rate for the two network topologies. It is interesting to observer that when λ increases, the ratio drops below 2 quickly. It indicates that the worst case is close to the best case with higher λ. Fig. 8 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 $t = 100s$. We also notice that larger λ results in larger $M(t)$ and $A(t)$. It is expected because larger λ means more records generated per unit time.

B. Comparative Study

In this section, we compare the performance of LCS with that of LS and ES in message overhead. Note that we did not compare LCS with DCS because these two storage methods target different application scenarios and they employ a totally different set of input parameters. For example, the message overhead in DCS depends on the number of event types, the harsh function exploited, etc. But in LCS, events are stored and disseminated based on its home location and its characteristics (seriousness, price, intention, etc.). (For more information, we refer the readers to the related work section (Section II).) Therefore, we found that it is almost impossible to design a simulation study for fairly comparing LCS and DCS.

For readers' convenience, we elaborate the ES and LS protocols in detail as follows:

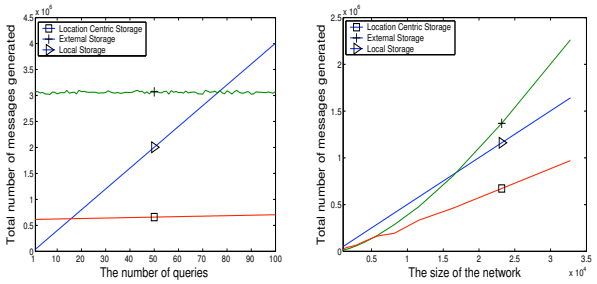
- External Storage (ES): Upon events occur, the relevant data are sent to the base station. This incurs communication cost to report the data. The query cost is only the communication between the user and the base station, which is negligible.
- Local Storage (LS): Upon events occur, the relevant data are stored locally at the detected nodes. This incurs no communication cost. Queries are flooded to the whole network at a cost of $O(N)$, where N is the network size. Then responses including the data are sent back to the user, which incurs a unicast communication cost between the user and the nodes that store the data.

The packet generation and forward behavior of ES and LS are faithfully followed. In the simulation, we assume nodes are uniformly distributed in a rectangle area, the packet delivery is instantaneous and error-free, and all the nodes are static. The relevant parameters used in the simulation are as follows and all other factors are held fixed during simulation:

- N, the number of nodes in the network.
- Q, the number of queries during simulation.
- E, the number of records during simulation. It is obvious that the number of records E generated in the simulation is proportioned to the network size N when the record generation rate at each sensor follows the same Poisson distribution.

In this simulation, we put 4 base stations in the four corners of the simulated area when testing ES. Each node sends its data to the nearest base station. In LCS, σ is set to ∞ for fair comparison since otherwise, event records may not be able to reach the boundary nodes.

First, we hold N fixed ($N = 40000$) and vary Q. Then we set Q=50, hold the network density fixed and vary N. In each test, we report the total number of messages generated by ES, LS and LCS. In the following, we report the results that is first averaged over 5000 runs for each network topology and then averaged over 5 different network topologies.



(a) The total number of messages generated *vs.* the number of queries, the network size (N) = 40000.

(b) The total number of messages generated *vs.* the network size, the number of queries (Q) = 50.

Fig. 9. Performance Comparison of Location-Centric Storage, Local Storage and External Storage

The results from varying Q are shown in Fig. 9(a). We observe that the lower the value of Q, the smaller the number of messages generated in the LS mechanism, namely the lower communication overhead. However, the communication load of LS increases linearly with Q, which leads LS to a poor performance with larger quantity of queries. For larger Q, the

total number of messages generated in LCS is lower than that in EX, since the main overhead of LCS is produced by local communication, which is dependent on the intensity value σ and is fairly low. The communication load of ES and LCS are both relatively independent of Q.

The results from varying N are shown in Fig. 9(b). Obviously as the network size increases, the total number of records generated during simulation increases when nodes generating records follow the same Poisson distribution. As shown in Fig. 9(b), ES starts off (at low N) with the lowest value, and ends up (at high N) with the highest value. The total number of messages generated by ES is only related to the number of generated records and the distance to the base station, which is a function of the network size. In Fig. 9(b), we notice that the total number of messages generated by LS increases linearly as the network size increases, since the query flooding distance is a function of the network size. Also we observe that LCS performs better than ES and LS. This is because LCS restricts the local flooding, and the user only needs to sample certain areas to obtain the data.

VII. Conclusion

In this paper we present a novel distributed location-centric data storage protocol called LCS for sensor networks. This protocol has many nice features, as indicated by our theoretical performance analysis and simulation study. We have identified several simple application scenarios of LCS, including safety warning in highway sensor networks, on-demand warning in surveillance networks, and context-dependent information mining in pervasive computing. We believe that the application of LCS is unlimited, and target this as our future research.

VIII. Acknowledgment

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

References

[1] D. Chen, M. Ding, and X. Cheng, Localized Event Detection in Sensor Networks, *manuscript*, 2005.

[2] X. Cheng, A. Thaeler, G. Xue, and D. Chen, TPS: A Time-Based Positioning Scheme for Outdoor Sensor Networks, *IEEE INFOCOM 2004*, HongKong China, March 2004.

[3] M. Ding, D. Chen, K. Xing, and X. Cheng, Localized Fault-Tolerant Event Boundary Detection in Sensor Networks, *IEEE INFOCOM 2005*, Miami, Florida, March 2005.

[4] M. Ding, D. Chen, A. Thaeler, and X. Cheng, Fault-Tolerant Target Detection in Sensor Networks, *IEEE WCNC 2005*, New Orleans, Louisiana, March 2005.

[5] A. Ghose, J. Grossklags, and J. Chuang, Resilient Data-Centric Storage in Ad-Hoc Wireless Sensor Networks, *Proceedings of the 4th International Conference on Mobile Data Management (MDM2003)*, Melborune, Australia, January 2003.

[6] M. Grossmann, M. Bauer, and N. HŎnle, Efficiently Managing Context Information for Large-Scale Scenarios, *PerCom 2005*.

[7] B. Karp and H.T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*, Boston, Massachusetts, August 2000.

[8] Q. Li, M. De Rosa, and D. Rus, Distributed Algorithms for Guiding Navigation across a Sensor Network, *ACM MobiCom 2003*, pp. 313-325, San Diego, Sept. 2003.

[9] Q. Li, R. Peterson, M. De Rosa and D. Rus Reactive Behavior in Self-reconfiguring Sensor Networks, *ACM Mobicom 2002*, Atlanta, Sep. 25-27, 2002.

[10] F. Liu, X. Cheng, D. Hua, and D. Chen, TPSS: A Time-based Positioning Scheme for Sensor Networks with Short Range Beacons, *2005 International Conference on Computer Networks and Mobile Computing (ICCNMC'05)*, April 2 -4, 2005, ZhangJiaJie, Hunan, China.

[11] W. Liu, Y. Zhang, W. Lou, Y. Fang and T. Wong Scalable and robust data dissemination in wireless sensor networks, *IEEE Globecom 2004*, Dallas, Texas, USA, November 2004.

[12] W. Lou, W. Liu and Y. Fang, SPREAD: enhancing data confidentiality in mobile ad hoc networks, *The 2004 IEEE International Conference on Computer Communications (INFOCOM'04)*, Hong Kong, March 2004.

[13] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, Wireless Sensor Networks for Habitat Monitoring, *ACM WSNA02*, Atlanta GA, September 2002.

[14] F. Perich, A. Joshi, T. Finin, and Y. Yesha, On Data Management in Pervasive Computing Environments, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16 (5), May 2004.

[15] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp and S. Shenker, Data-Centric Storage in Sensornets, *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, September 28 2002, Atlanta, GA.

[16] S. Ratnasamy, B. Karp, L. Yin and F. Yu, Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table, *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*, Vol. 8, pp.427-442, 2003.

[17] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang, Analyzing Object Detection Quality under Probabilistic Coverage in Sensor Networks, *Thirteenth International Workshop on Quality of Service (IWQoS 2005)*, Passau, Germany, June 21-June 23 , 2005.

[18] S. Ren, Q. Li, H. Wang, and X. Zhang, Design and Analysis of Wave Sensing Schedules for Object-Tracking Applications, *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Marina del Rey, CA, June 30-July 1, 2005.

[19] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang, Analyzing Object Tracking Quality under Probabilistic Coverage in Sensor Networks, *ACM Mobile Computing and Communications Review*, 9(1), pages 73-76, Jan. 2005.

[20] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang, Probabilistic Coverage for Object Tracking in Sensor Networks, *ACM MobiCom 2004 poster*, Philadelphia, Pennsylvania, Sep. 26-Oct. 1, 2004.

[21] K. Seada and A. Helmy, Rendezvous regions: a scalable architecture for service location and data-centric storage in large-scale wireless networks, *18th International Parallel and Distributed Processing Symposium, 2004*, pp.218-225, 26-30 April, 2004.

[22] R. Tamishetty, L. H. Ngoh, and P. H. Keng, An efficient resiliency scheme for data centric storage in wireless sensor networks, *Vehicular Technology Conference*, 2004. VTC2004-Fall. 2004 IEEE 60th Volume 4, pp.2936 - 2940, 26-29 Sept. 2004.

[23] A. Thaeler, M. Ding, and X. Cheng, iTPS: An Improved Location Discovery Scheme for Sensor Networks with Long Range Beacons, to appear 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*, Fall 2004.

[24] M. Weiser, The Computer for the 21st Century, *Scientific American*, September 1991.

[25] K. Xing, M. Ding, X. Cheng and S. Rotenstreich, Safety Warning Based on Highway Sensor Networks, *IEEE WCNC 2005*, New Orleans, Louisiana, March 2005.

[26] K. Xing, M. Ding, X. Cheng and S. Rotenstreich, Safety Warning Based on Roadway Sensor Networks, *Manuscript*, 2005.

[27] S. S. Yau, S. K. S. Gupta, F. Karim, S. I. Ahamed, Y. Wang, and B. Wang, Smart Classroom: Enhancing Collaborative Learning Using Pervasive Computing Technology, *Proc. of 6th WFEO World Congress on Engineering Education and Second ASEE International Colloquium on Engineering Education (ASEE)*, Nashville, Tennessee, June 2003.

[28] Y. Zhang, W. Liu, W. Lou and Y. Fang, Securing sensor networks with location-based Keys, *IEEE Wireless Communications and Networking Conference (WCNC'05)*, New Orleans, LA, 13-17 March, 2005.

[29] Y. Zhou, Y. Zhang and Y. Fang, LLK: a link-layer key establishment scheme in wireless sensor networks, *IEEE Wireless Communications and Networking Conference (WCNC'05)*, New Orleans, LA, 13-17 March, 2005.