Geometric Random Linear Codes in Sensor Networks

Yunfeng Lin, Ben Liang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{ylin, bli}@eecg.toronto.edu, liang@comm.toronto.edu

***Abstract*—Wireless sensor networks consist of unreliable and energy-constrained sensors connecting to each other wirelessly. As measured data may be lost due to sensor failures, maintaining the persistence of periodically measured data in a scalable fashion has become a critical challenge in sensor networks, without the use of centralized servers. To cope with node failures, while providing convenient access to measured data, we propose *geometric random linear codes*, to encode data in a hierarchical fashion in geographic regions with different sizes, such that data are easy to access, if the original sensors producing the data are alive. Otherwise, data are persistently available elsewhere in the network. Although our coding scheme is simple, we have shown that it enjoys the same low encoding cost as sparse random linear codes, while dramatically decreasing the decoding cost. We present extensive analytical and experimental results to show the effectiveness of geometric random linear codes.**

I. Introduction

Sensor networks consist of sensors that fail dynamically because sensors are unreliable and energy-constrained. Furthermore, as a common strategy to prolong network life, sensors are put through sleep scheduling to conserve energy, *i.e.*, they are turned on and off in a periodical fashion.

Yet, as sensor networks are deployed to monitor environments, the measured data have to be preserved for later analysis. How do we collect such periodically measured data, which may grow to substantial volumes over time? There are two reasons to believe that centralized servers or sinks may not be the appropriate answer. First, in sensor networks, it may be too costly and unrealistic to periodically maintain routing structures (*e.g.*, aggregation trees) to centralized sinks, again due to frequent sensor failures and energy-conserving measures. Second, it may not be feasible to deploy powered sinks or storage server in inaccessible geographic regions such as isolated islands or battle fields.

In this paper, we study the challenges involved where no sinks exist in sensor networks, and where periodically measured data must be stored within the network itself in a collaborative fashion. This conforms to the peer-to-peer mentality, but could be a serious problem when nodes are inherently dynamic and failure-prone. The objective of this paper is to propose new coding techniques inside the network, inspired by traditional random linear codes commonly used in network coding, such that data stored in the network can be efficiently recovered.

This work was supported in part by Bell Canada through its Bell University Laboratories R&D program.

Random linear codes, traditionally used in network coding, is a practical implementation of erasure codes to provide data persistence in sensor networks in a decentralized fashion. We argue that it is not appropriate for many applications in sensor networks due to the following reason. It has been envisioned [1] that sensor networks are a database interface to the physical world. Under such a database abstraction, a data query may be injected anywhere in the sensor field, and may only be interested in data measured in a small geographic region. If using random linear codes, when measured data are segmented as original source blocks, we need as many coded blocks as the original source blocks to decode any useful data. Therefore, the communication cost in collecting coded blocks to decode any useful data is too high in these applications.

In this paper, we propose geometric random linear codes to provide both data fault tolerance and easy data access, by trading off a modest amount of storage space. In essence, we encode data hierarchically in geographic regions with different sizes. If sufficient sensors in neighborhood are alive when a data query arrives, these sensors can serve the request with a small amount of coded blocks. On the other hand, in the event of group sensor failures within local area, the data query can be satisfied by remote active sensors, albeit with a larger amount of coded blocks and a higher communication cost.

Through theoretical analysis, we show that although geometric random linear codes disseminate more data in the encoding phase, the encoding communication cost is asymptotically identical to sparse random linear codes [2], and hence is optimal. Despite the fact that our storage overhead is higher than random linear codes by $O(\log N)$, given there are N sensors in the network, we believe such a tradeoff is worthwhile, as our decoding communication cost is asymptotically smaller than random linear codes by $\Theta(N/\log N)$ under common sensor failure models and network parameters. More importantly, a small storage overhead can be justified by the significantly reduced communication cost, which increases the lifetime of sensor networks dramatically.

The remainder of the paper is organized as follows. In Sec. II, we compare our approach with related work. In Sec. III, we describe the network model. In Sec. IV, we introduce geometric random linear codes. We present extensive analysis of their properties in Sec. V. Simulation evaluation of geometric random linear codes is in Sec. VI. Sec. VII concludes the paper.

II. Related Work

Different variants of distributed erasure coding have been proposed in sensor networks [2]–[5], or distributed networked storage [6]–[9]. However, most of the existing work requires collecting a large amount of coded data that are at least the volume of the data generated in the network in order to recover any useful data. Similar to growth codes [3] and priority random linear codes [9], we support partially recovering of a subset of data in the network in this work. However, our coding scheme targets different applications from them. In our applications, data queries can be injected *anywhere* in a sensor field, and coded data are delivered to the requesting sensor via multi-hop wireless communication. We are interested in reducing the communication cost to recover data. In contrast, growth codes aim to recover as much as possible data on sinks in a catastrophic scenario. Hence, they exchange data among sensors with a gossiping style of protocol and have much higher communication cost than ours. Priority random linear codes assume that a mobile collector moves around within the sensor field to gather cached coded data directly. Hence, unlike in our applications, there is no multi-hop communication during the data gathering phase, and they do not consider the communication cost in decoding.

Huang *et al.* [10] propose pyramid codes to trade space for reading efficiency in traditional data storage systems. Their work has similar spirit as ours, but differs in three important aspects. First, pyramid codes are not applicable for sensor networks where data to be encoded are generated from different locations. Second, there are no concept of geographic distance and locality in pyramid codes. Third, pyramid codes assume that the decoder can access all coded data at the same time, whereas we assume that a sensor collects coded data from multiple sensors through wireless multi-hop communication.

III. Network Model

Our network model consists of N sensors deployed uniformly in a square sensor field with size $\sqrt{N} \times \sqrt{N}$. In addition, we assume that all sensors know their locations in the sensor field, and greedy geographic routing or GPSR [11] are implemented to route packets from one location to another, through the shortest geographic path in the network. Furthermore, we assume that K data sensors are distributed uniformly among all sensors, where $K \leq N$. Yet, we also assume that the K data sensors are deployed sufficiently dense such that they cover the entire sensor field, and $K = \Theta(N)$, *i.e.*, $K = \lambda N$, where λ is a positive number less than 1. Note that all N sensors, including the K data sensors, are used to store coded data.

We assume that each data sensor monitors the environment and compresses the obtained measurements to generate a segment of data, referred to as a *source block* hereafter, for each time interval periodically. Without loss of generality, we focus on the persistence of the source blocks generated in one time interval. These K source blocks are encoded and disseminated among all N sensors such that they can be retrieved elsewhere if some data sensors fail. Finally, we assume that data queries can be injected from any sensor in the network, and the destination of a data query can be anywhere in the sensor field as well.

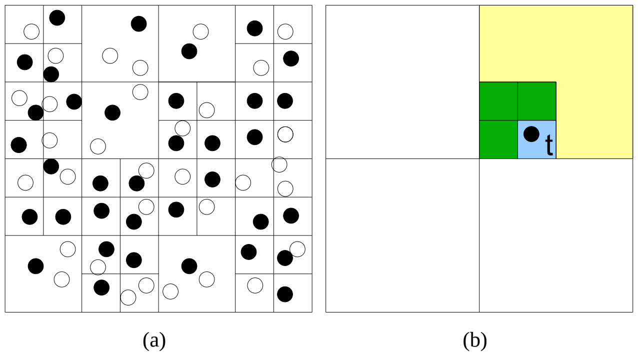

Fig. 1. (a) Leaves of the quad tree built in the sensor field. The solid and empty circles represent data sensors and non-data sensors, respectively. (b) Sensor t belongs to multiple levels of squares.

IV. Geometric Random Linear Codes

We describe our method for encoding and decoding data in sensor networks using *Geometric Random Linear Codes* (G-RLC).

A. Distributed Encoding Algorithm

We construct a standard *virtual* quad tree T in a sensor field. The root node of T corresponds to the bounding box of the sensor field covering all sensors. The nodes on quad tree T are assigned recursively such that each node on the tree are divided to four squares with equal sizes, *until at most one data sensor exists in the leaf nodes*. Fig. 1(a) shows an example of the leaves of the quad tree. In the following, we use squares or nodes on the quad tree interchangeably. Since we assume that each sensor is aware of its location in the sensor field, it is able to compute all levels of squares in the sensor field. For example, it knows all tiles that it sits in as illustrated in Fig. 1(b). Such information is used in the encoding algorithm described later in this section. Note that we do not need to store the tree nodes of quad tree on sensors in practice.

By dividing squares by 4 recursively, it is easy to see that there are $L = \log_4 K$ levels of squares with different sizes in a sensor field. Let the root node of quad tree be the 0-level square, and its ith level of descendants as the ith level squares. Hence, if we use N_i and K_i to denote the number of all sensors and data sensors in the ith level squares, we have

$$N_i = \frac{N}{4^i}, \qquad K_i = \frac{K}{4^i}. \tag{1}$$

Next, we present the decentralized encoding algorithms used in G-RLC. For each square in the sensor field[1], we use the sparse *Random Linear Codes* (RLC) [2] to encode data in a decentralized way. In particular, for each square S that a data

[1]Note that there are different levels of overlapping squares.

sensor sits in, it disseminates $\Theta(\log n)$ copies of its source block to $\Theta(\log n)$ uniform random locations in S by greedy geographic routing or GPSR [11], where n is the number of sensors in S. Each sensor, say sensor j, in S maintains a *coded block* c_j for S as a random linear combination of the source blocks produced in S. Initially, c_j is set to 0. Upon receiving a source block x_i from a data sensor i in S, sensor j will combine x_i with c_j with the following algorithm:

$$c_j = c_j + \beta_{i,j} x_i \tag{2}$$

where the coding coefficient $\beta_{i,j}$ is chosen, uniformly at random, from a Galois field. Beside the coded block c_j, sensor j stores the coding coefficient $\beta_{i,j}$ as well. In practice, the storage overhead for coding coefficients can be ignored since the size of coded blocks is usually significantly larger than the coding coefficients [2].

We defer the description of the algorithm to retrieve coded blocks in Sec. IV-B. Instead, we present the decoding algorithm when a sensor has obtained K_i coded blocks and their associated coding coefficients in an ith-level square. By the above encoding algorithm, each coded block represents a linear equation with the source blocks x_i as the unknown variables, because the coding coefficients and the coded block are known. Decoding the K_i source blocks is equivalent to solving the linear system composed of the K_i coded blocks. The decoding matrix represents the coefficient matrix of such a linear system. When the rank of the decoding matrix is K_i, the linear system can be solved and the K_i source blocks are decoded. Otherwise, there is linear dependence among the K_i coded blocks, and the sensor will continue to obtain more coded blocks until the K_i source blocks can be decoded.

In the above algorithm, we enforce $\Theta(\log n)$ copies of each source block is disseminated in all levels of squares. However, in practice there are only a small number of sensors located within the leaves or the several lowest levels of the descendants on the quad tree. However, the sparse decoding result (each source block needs to be disseminated to $\Theta(\log n)$ random locations) only hold under the condition that n is sufficiently large. Hence, in these small squares, each source block needs to be disseminated to more than $\Theta(\log n)$ random locations to guarantee that K_i coded blocks in the ith level square are sufficient to decode K_i source blocks with high probability. In the extreme case, a data sensor in a leaf on the quad tree replicates its data to all sensors in the leaf.

B. Coded Block Retrieval Algorithm

We then describe the algorithm to retrieve coded blocks. Clearly, if the network is disconnected due to node failures, decoding always fails. Hence, in the following, we assume that the network is always connected. Suppose that sensor r is interested in the data at location q. Note that q is covered by a data sensor by our assumption in Sec. III before node failures occur. The block retrieval algorithm seeks to reach the smallest square in the quad tree that cover location q, and has sufficient coded blocks to decode the data on location q.

In particular, r sends a data query to location q through greedy geographic routing or GPSR. Through these routing protocols, a data query always arrive at the closest sensor to location q. If such a sensor is one of the sensors inside the leaf of the quad tree that covers location q, because the source block of location q is replicated to every sensor in the leaf as described in Sec. IV-A, the data query is served and the original source block is transmitted back to sensor r, again with greedy geographic routing or GPSR. If all sensors in the leaf of the quad tree that covers location q fail, then the data query arrives at a sensor inside the parent (level $L-1$) of leaf on the quad tree. Clearly, the parent of leaf covers location q as well, since the leaf is geographically a part of its parent. The data query attempts to access sensors that are *closest* to sensor r in the parent of the leaf. If sufficient coded blocks are available on the active sensors to decode K_{L-1} source blocks, these coded blocks are sent back to sensor r for decoding. Otherwise, the data query will retrieve data on the level $L-2$ square that is the grandparent of the leaf. This process continues until either the decoding is successful in a particular level of square that covers location q, or decoding fails even on the root node of the quad tree because of too many sensor failures.

V. Analysis

In this section, we provide rigorous analysis to characterize the asymptotic storage and communication costs of G-RLC.

A. Sensor Storage Space Consumption

The following lemma characterizes the storage requirement for each sensor.

Lemma 1: A sensor stores $O(\log N)$ coded blocks.

Proof: A sensor belongs to at most $\log_4 K$ levels of squares as described in Sec. IV-A. It stores one coded block for each level of squares. Hence, it stores $O(\log K) = O(\log N)$ coded blocks due to the fact that $K = \Theta(N)$. ■

As compared to RLC where each sensor stores one coded block [2], G-RLC consumes more storage with a logarithmic factor. However, we will show that G-RLC significantly reduces the communication cost of decoding in Sec. V-C.

B. Communication Cost in Decentralized Encoding

We derive the asymptotic communication cost in the decentralized encoding of G-RLC, focusing on the dissemination cost of a source block.

Theorem 1: The dissemination cost for a source block is $\Theta(\sqrt{N} \log N)$.

Proof: By our encoding algorithm described in Sec. IV-A, a source block needs to be disseminated to $c_1 \log_2 n$ sensors, where n is the number of sensors in the square, and c_1 is a constant. Furthermore, it requires $c_2 \sqrt{n}$ to disseminate a source block to a random location in the square, where c_2 is a constant. Hence, the communication cost $d(n)$ to disseminate a source block is

$$\begin{aligned} d(n) &= c_1 c_2 \sqrt{n} \log_2 n \\ &= c \sqrt{n} \log_2 n, \end{aligned} \tag{3}$$

where $c = c_1 c_2$.

For G-RLC, the communication cost to disseminate a source block is the sum of the dissemination costs for all $\log_4 K$ squares that the data sensor belongs to. Hence, the total dissemination cost $D(N)$ of a source block is as follows:

$$\begin{aligned} D(N) &= \sum_{i=0}^{\log_4 K} d(\frac{N}{4^i}) \\ &= \sum_{i=0}^{\log_4 K} c\sqrt{\frac{N}{4^i}} \log_2 \frac{N}{4^i} \\ &= c\sqrt{N} \sum_{i=0}^{\log_4 K} \frac{1}{2^i}(\log_2 N - 2i) \\ &\le c\sqrt{N} \log_2 N \sum_{i=0}^{\log_4 K} \frac{1}{2^i} \\ &\le 2c\sqrt{N}\log_2 N, \end{aligned} \tag{4}$$

where the second equality is due to Eq. (3), and all other equalities are easy to see. We notice that in the lowest descendants of the quad tree, a data sensor disseminates more than $\Theta(\log n)$ copies of its source block. However, because these squares is very small, the dissemination cost remains very small, and does not change the conclusion. ∎

Because RLC has a dissemination cost of $\Theta(\sqrt{N}\log N)$ [2], we conclude that G-RLC has the same asymptotic dissemination cost. The underlying intuition is as follows. Although a data sensor disseminates more copies for each source block than RLC, the destinations of most copies are concentrated around the data sensor, and low communication costs are required. We further note that the constant in (4) is a small value 2, which implies in practice the dissemination cost increases at most twice as compared to RLC. However, such tradeoff is justified by the significantly decreased decoding communication cost as shown in Sec. V-C.

C. Communication Cost in Decoding

We assume the following sensor failure model. After the phase of decentralized encoding, each sensor is still alive with probability p independently. We first consider the following two extreme cases. First, when $p \to 1$, there are very few sensor failures in the network. Hence, with high probability, a data query is able to be served in the leaves of the quad tree with the original source block. Therefore, the communication cost for decoding the data is the diameter of the sensor field $\Theta(\sqrt{N})$. Note that we have significant improvement over RLC in such a case, as its decoding communication cost is $\Theta(N\sqrt{N})$ [2]. Second, if $p \to 0$, most sensors die such that both our coding scheme and RLC are unable to decode data because either the network is partitioned or insufficient coded data remain in the sensor field. In the following, we investigate how the decoding communication cost scales if p assumes a value between 0 and 1.

For analytical convenience while still capturing the problem essence, we ignore the effect of network partition due to node failures. Furthermore, by the law of large numbers [12], the number of surviving sensors in the sensor field is very close to pN when N is very large. Hence, if $p < \lambda$, where $\lambda = K/N$ is the coding rate, decoding almost always fails with RLC. Therefore, in our analysis, we concentrate on the case when $p > \lambda$.

Theorem 2: If $p > 1 - (\lambda/e)^{\lambda/(1-\lambda)}$, the decoding communication cost is $\Theta(\sqrt{N}\log N)$.

Proof: We define A_i as the event that the number of survived sensors M_i in an ith-level square is sufficient to decode the data with high probability. If random coefficients used in encoding are chosen from a sufficient large Galois field, event A_i is equivalent to the event $\{M_i \ge K_i\}$ [13], which is usually the case in reality. With the same argument of the law of large numbers [12], under the condition $p > \lambda$, we have

$$\Pr(A_0) = 1. \tag{5}$$

Let B_i denote the event that the decoding succeeds, and C_i represent the communication cost in the ith-level square. Then, we have the expected decoding communication cost $E[C|A_0]$ under the condition that decoding succeeds at least on the root node of the quad tree, *i.e.*, A_0 happens, as follows:

$$E[C|A_0] = \sum_{i=0}^{L} \Pr(B_i|A_0) C_i. \tag{6}$$

Furthermore, we have

$$\begin{aligned} \Pr(B_i|A_0) &= \frac{\Pr(B_i A_0)}{\Pr(A_0)} \\ &= \Pr(B_i A_0) \\ &\le \Pr(B_i). \end{aligned} \tag{7}$$

Substituting (7) into (6), we have

$$E[C|A_0] \le \sum_{i=0}^{L} \Pr(B_i) C_i. \tag{8}$$

Next, we derive the upper bounds of C_i and $\Pr(B_i)$ to bound the expected decoding communication cost $E[C]$. The decoding communication cost C_i is the product of the upper bound of the communication cost (*i.e.*, the diameter of the network $\Theta(\sqrt{N})$) to obtain a coded block and the number of coded blocks K_i that is sufficient to decode data with high probability. Hence, we have

$$C_i \le K_i c_1 \sqrt{N} = cN_i\sqrt{N}, \tag{9}$$

where c_1 and $c = \lambda c_1$ are constants, and K_i and N_i are from (1).

Next, we compute $\Pr(B_i)$. Because decoding occurs at the i level only if decoding fails from level $i+1$ to level L. Hence, we have

$$\begin{aligned} \Pr(B_i) &= \Pr(A_i \overline{A_{i+1}} \cdots \overline{A_L}) \\ &= \Pr(A_i | \overline{A_{i+1}} \cdots \overline{A_L}) \Pr(\overline{A_{i+1}} \cdots \overline{A_L}) \\ &\le \Pr(\overline{A_{i+1}} \cdots \overline{A_L}) \\ &\le \Pr(\overline{A_{i+1}}). \end{aligned} \tag{10}$$

The event A_{i+1} happens if less than K_{i+1} sensors survive in the $i+1$th level of square. Furthermore, given the independent sensor failure model, the number of sensors that are still alive conforms to a binomial distribution. Hence, we have

$$\begin{aligned}\Pr(\overline{A_{i+1}}) &= \sum_{j=0}^{K_{i+1}-1} \binom{N_{i+1}}{j} p^j (1-p)^{N_{i+1}-j} \\ &\le \sum_{j=0}^{K_{i+1}} \binom{N_{i+1}}{j} p^j (1-p)^{N_{i+1}-j} \\ &\le \binom{N_{i+1}}{K_{i+1}} (1-p)^{N_{i+1}-K_{i+1}} \\ &\le \left(\frac{eN_{i+1}}{K_{i+1}}\right)^{K_{i+1}} (1-p)^{(1-\lambda)N_{i+1}} \\ &= \left(\left(\frac{\lambda}{e}\right)^{\lambda}\left(\frac{1}{1-p}\right)^{1-\lambda}\right)^{-N_{i+1}} \\ &= \alpha^{-N_{i+1}}, \end{aligned} \tag{11}$$

where the second inequality is due to the tails of the binomial distribution [14], the third inequality is because of the binomial bounds [14], and the fact that $K_i = \lambda N_i$. $\alpha = (\lambda/e)^{\lambda}(1/(1-p))^{1-\lambda}$ is used to simplify the notations. Hence, $\Pr(\overline{A_{i+1}})$ decreases exponentially if $\alpha > 1$, *i.e.*, $p > 1-(\lambda/e)^{\lambda/(1-\lambda)}$.

For the special case where a source block is replicated to all N_L sensors in a square of level L. The event B_L denoting decoding success occurs if at least one sensor is alive in the square. Therefore, we have

$$\Pr(B_L) = 1-(1-p)^{N_L} \le 1 \tag{12}$$

where we use a trivial bound 1.

We are now ready to bound the conditional expected decoding communication cost. Substituting (11) to (10), then (9), (10), and (12) to (8), we have

$$\begin{aligned} E[C|A_0] &\le cN_L\sqrt{N} + \sum_{i=0}^{L-1} \alpha^{-N_{i+1}} cN_i\sqrt{N} \\ &= c\sqrt{N}(N_L + \sum_{i=0}^{L-1} N_i \alpha^{-N_{i+1}}). \end{aligned} \tag{13}$$

Let $f(i) = N_i\alpha^{-N_{i+1}}$. We will show that $f(i) \le f(i+1)$ when i is asymptotically less than $L = O(\log N)$ by the following argument:

$$\begin{aligned}\frac{f(i+1)}{f(i)} &= \frac{1}{4}\alpha^{\frac{3}{4^{i+2}}N} \\ &> 1. \end{aligned} \tag{14}$$

Henceforth, (13) becomes

$$\begin{aligned} E[C|A_0] &\le O(\sqrt{N}(N_L + LN_L\alpha^{-N_L})) \\ &= O(\sqrt{N}\log N), \end{aligned} \tag{15}$$

where the second equality utilizes the fact that $L = O(\log N)$, and $N_L = N/4^L$ is a constant. ■

Theorem 2 shows that our coding scheme has much smaller decoding communication cost than RLC as long as the surviving probability p is sufficiently large.

VI. Simulation Validation

In this section, we demonstrate the advantage of G-RLC with simulations. Our simulation results are consistent and in complement with the asymptotic theoretical results presented in Sec. V. We have implemented both RLC and G-RLC on a grid network with size $\sqrt{N} \times \sqrt{N}$, where a random subset of K sensors monitor the environment and generate data. We set $N = 4^7 = 16384$ and $K = N/2$. Furthermore, we use GF(2^8) as the Galois field where G-RLC and RLC are operated in all simulations. To focus on understanding the algorithmic performance difference between RLC and G-RLC, we ignore the effect of network partition due to node failures, because it has the same impact on both coding schemes.

The experiments are performed as follows. The K source blocks produced by data sensors are encoded with G-RLC and RLC, respectively. Each sensor then survives with probability p. Afterwards, the source and the destination of a data query are randomly chosen in the sensor field. The source then sends a data query to the destination, and we use the decoding algorithms of G-RLC and RLC to decode data.

A. Data Resilience

We first illustrate the data resilience performance of G-RLC and RLC in Fig. 2, where the fraction of data queries with decoding success among all 1000 independent data queries is reported. We observe that if the surviving probability p is greater than the percentage of data sensors (*i.e.*, $p > 0.5$, both G-RLC and RLC are able to serve almost all data queries successfully. However, if $p < 0.5$, almost all data queries of RLC fail. This is due to the fact that the number of survived sensors are very close to pN as justified in Sec. V-C by the law of large numbers. Hence, there are insufficient coded blocks remained in the network to decode all K source blocks with high probability, if $p < 0.5$. On the other hand, G-RLC can decode a faction of data even if $p < 0.5$. This is because G-RLC encodes data within small squares where the number of sensors is much smaller, and the law of large numbers does not apply. Therefore, in small squares of G-RLC, the probability that there are sufficient sensors alive to provide coded data is much higher than in RLC. In summary, Fig. 2 demonstrates that G-RLC provides significantly higher fault tolerance than RLC when the surviving probability p is smaller than the percentage of data sensors in the network.

B. Decoding Communication Cost

Next, we study the decoding communication cost of both coding schemes. To mitigate randomness in simulations, we show, for each data point in Fig. 3, the average and the 95% confidence intervals from 100 independent experiments. The experimental data of decoding communication costs consists of both the case of decoding success and decoding failure.

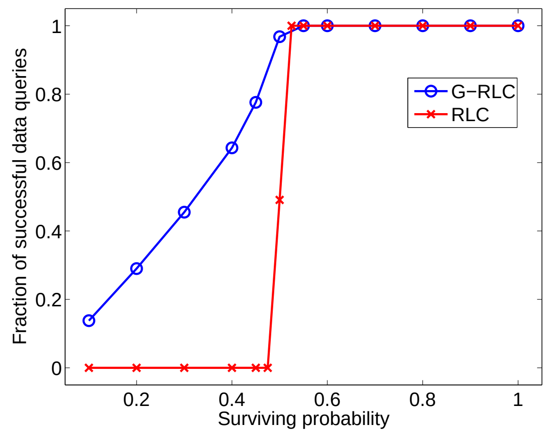

Fig. 2. The fraction of success data queries under different sensor surviving probabilities.

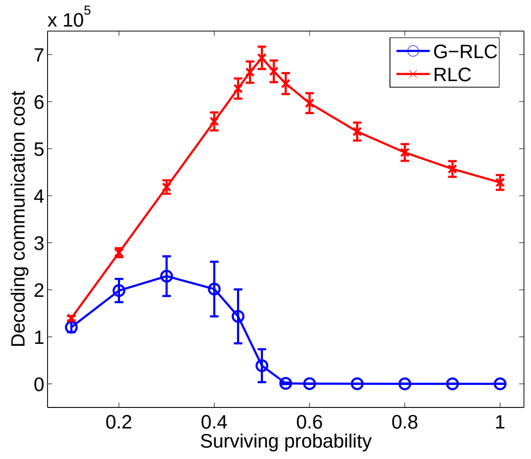

Fig. 3. The decoding communication cost under different sensor surviving probabilities.

If decoding succeeds, the decoding communication cost represents the amount of the transmissions in terms of hop counts in order to answer a data query. On the other hand, if decoding fails, the decoding communication cost is the total communication cost that the data query gathers all coded blocks on all active sensors.

We describe our findings from Fig. 3. First, we observe that the decoding communication cost of G-RLC is significantly smaller than RLC as expected, because of G-RLC can retrieve and decode from a small geographic region with significantly less coded data. Second, the decoding communication cost of RLC increases when the surviving probability increases from 0.1 to 0.5. This is because there are insufficient sensors for decoding success such that coded data on all active sensors are collected. Hence, the communication cost in proportion to the number of active sensors in the network, which increases with the surviving probability. If $p > 0.5$, the decoding communication cost of RLC decreases because the coded data can be gathered from more and more active sensors in local area. Third, similarly, we can explain the increasing trend for G-RLC when the surviving probability is very small with the same argument for RLC. However, G-RLC differs from RLC in that its decoding communication starts to decrease when p is around 0.3 and decreases very fast when p approaches 0.5 because the increasing density of active sensors.

VII. Conclusion

In this paper, we introduce geometric random linear codes to provide both data persistence and convenient data access in sensor networks. Our study is based on extensive theoretical analysis and experimental evaluations. We have shown that in comparison with random linear codes, although geometric random linear codes introduce a modest storage overhead and a small additional communication cost in decentralized encoding, it significantly reduces the communication cost to decode useful data, when a query is only interested in the data generated in a small geographic subregion, as compared to random linear codes. Hence, the proposed coding scheme represents a significant step toward applying practical usage of decentralized erasure coding in sensor networks.

References

[1] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker, “The Sensor Network As a Database,” University of Southern California, Tech. Rep., 2002.

[2] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, “Decentralized Erasure Codes for Distributed Networked Storage,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2809–2816, June 2006.

[3] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein, “Growth Codes: Maximizing Sensor Network Data Persistence,” in *Proc. of ACM SIGCOMM*, 2006.

[4] D. Wang, Q. Zhang, and J. Liu, “Partial Network Coding for Continuous Data Collection in Sensor Networks,” in *Proc. of the Fourteenth IEEE International Workshop on Quality of Service (IWQoS)*, 2006.

[5] Y. Lin, B. Liang, and B. Li, “Data Persistence in Large-scale Sensor Networks with Decentralized Fountain Codes,” in *Proc. of IEEE INFOCOM*, 2007.

[6] S. Acedanski, S. Deb, M. Medard, and R. Koetter, “How Good is Random Linear Coding Based Distributed Networked Storage?” in *Proc. of First Workshop on Network Coding, Theory, and Applications (NetCod)*, 2005.

[7] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, “Network Coding for Distributed Storage Systems,” in *Proc. of IEEE INFOCOM*, 2007.

[8] C. Wu and B. Li, “Echelon: Peer-to-Peer Network Diagnosis with Network Coding,” in *Proc. of the Fourteenth IEEE International Workshop on Quality of Service (IWQoS)*, 2006.

[9] Y. Lin, B. Li, and B. Liang, “Differentiated Data Persistence with Priority Random Linear Codes,” in *Proc. of 27th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2007.

[10] C. Huang, M. Chen, and J. Li, “Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems,” in *Proc. of IEEE International Symposium on Network Computing and Applications (NCA)*, 2007.

[11] B. Karp and H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” in *Proc. of the 6th ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[12] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[13] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, “The Benefits of Coding over Routing in a Randomized Setting,” in *Proc. of IEEE International Symposium on Information Theory*, 2003.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001.