Priority Random Linear Codes in Distributed Storage Systems

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*—Node churn and failures exist as fundamental characteristics in both peer-to-peer (P2P) and sensor networks. Peers in P2P networks are highly dynamic, whereas sensors are not dependable. As such, maintaining the persistence of periodically measured data in a scalable fashion has become a critical challenge in such systems, without the use of centralized servers. To better cope with node dynamics and failures, we propose *priority random linear codes*, as well as their affiliated pre-distribution protocols, to maintain measurement data in different priorities, such that critical data have a higher opportunity to survive node failures than data of less importance. A salient feature of priority random linear codes is the ability to *partially* recover more important subsets of the original data with higher priorities, when it is not feasible to recover all of them due to node dynamics. We present extensive analytical and experimental results to show the effectiveness of priority random linear codes.**

***Index Terms*—Distributed networks, distributed applications, distributed priority coding, random linear codes**

I. Introduction

One of the most important challenges in fully autonomous networks, including peer-to-peer (P2P) networks and wireless sensor networks, has been the dynamic behavior of peer nodes and sensors. Peers in P2P architectures tend to participate in and depart from ongoing sessions in a highly dynamic fashion, and sensors are widely acknowledged to show strikingly similar dynamics, due to their lack of reliability, or the existence of energy-conserving protocols to periodically put sensors to the standby mode.

Nevertheless, in both P2P and sensor networks, periodically measured data are generated on an ongoing basis, which should be preserved for subsequent analysis at a later time. In P2P networks, it is critical for operators to monitor the performance and "health" of live peer-to-peer sessions. For example, in live media streaming applications, it is essential to monitor the achieved streaming rate, the number of upstream and downstream peers, the latency to neighboring peers, and resource usage such as bandwidth and CPU load. Similarly, in sensor networks, the task of each sensor is to monitor the environment, with periodic measurements collected for later retrieval.

How do we collect such periodically measured data, which may grow to substantial volumes over time? There are reasons to believe that centralized servers may not be the appropriate answer. In P2P networks, periodic reporting to central logging servers does not scale well to a large number of peers, and may morph into a *de facto* distributed denial-of-service attack at the logging server. 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.

In this paper, we study the challenges involved when no centralized servers exist in autonomous networks, and 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.

Random linear codes, traditionally used in network coding, achieves an *"all or nothing"* paradigm of decoding. When measured data are segmented as *original source blocks*, with random linear codes, we need as many coded blocks as the original source blocks to decode *any* useful data. We argue that such a paradigm is not appropriate for either P2P or sensor networks, since node departures and failures may easily render the remainder of coded blocks useless! Having many more coded blocks than source blocks certainly helps, but we would prefer to progress beyond simple over-provisioning of cache storage, especially when cache spaces on nodes are limited.

In this paper, we propose *priority random linear codes* in a generic network model that encompasses both P2P and sensor networks. A salient feature of priority random linear codes is the ability to *partially recover* more important subsets of the original data with higher priorities, when it is not feasible to recover all of them due to node dynamics. In a nutshell, we achieve this by making sure that coded blocks for important data are linear combinations of *fewer* source blocks, as compared to those for non-critical data. In essence, we assign lower coding rates for important data, such that they can be recovered with fewer coded blocks, and may survive higher percentages of node departures and failures.

In addition to our extensive theoretical analysis of priority random linear codes, we also present their affiliated pre-distribution protocols. Utilizing the fact that each coded block is encoded from a subset of source blocks, our pre-distribution mechanism ensures that only source blocks in such a subset

A preliminary version of this paper was presented in IEEE ICDCS 2007 [1].

are delivered to their designated receivers for storage, rather than all source blocks. Furthermore, utilizing the previously known result that each coded block only needs to be the linear combination of $O(\ln N)$ random chosen source blocks for successful decoding [2] (where N is the total number of source blocks), our pre-distribution protocol is very efficient.

The remainder of this paper is organized as follows. In Sec. II, we describe the network model. In Sec. III, we introduce priority random linear codes and partial decoding algorithms, with extensive analysis of their properties. In Sec. IV, we describe the pre-distribution protocol. Performance evaluation of priority random linear codes is in Sec. V. We compare our approach with related work in Sec. VI. Finally, Sec. VII concludes the paper.

II. Network Model

In this paper, we consider a generic network model of autonomous networks with unreliable nodes, which encompasses commonly accepted models of both P2P and sensor networks. We consider such a general model to show that priority random linear codes can be applied to a wide range of autonomous networks, rather than specific to any particular type.

Our model of an autonomous network consists of a set of nodes and the communication links among them. Each node produces measurement data over time. There does not exist centralized servers at our disposal; instead, all measured data from a particular node must be distributed to other nodes in the network for peer-to-peer collaborative storage. We assume that each node only has a limited amount of storage space, and can only store a small fraction of the data generated in the network. At a later time, measured data stored at a random subset of existing nodes will be retrieved for analysis. All nodes in the network may depart or fail unpredictably. We partition the continuously generated measurement data by time slots, where a source block refers to the amount of the data generated in one time slot on a node. Clearly, how many time slots of data can be cached depends on the size of the node cache storage. In particular, the larger the cache storage is, the more time slots of data can be cached, assuming new data replace the oldest data when the cache is full. Without loss of generality, we focus on caching the source blocks produced in one time slot on all nodes.

We assume that N source blocks are produced in one time slot, which are classified to n different *priority levels*, in descending levels of importance — source blocks in priority level i are more important than those in level j, if $i < j$. The number of source blocks in priority level i is denoted by a_i, where $1 \leq i \leq n$. To facilitate later derivation, we introduce $b_1, b_2, \ldots, b_n$, where $b_i = \sum_{j=1}^{i} a_j$, *i.e.*, b_i represents the total number of source blocks from priority level 1 to i. In this case, the source blocks in priority level i are indexed as blocks $\{x_j\}$, where $b_{i-1} + 1 \leq j \leq b_i$. In general, the methods to segment measurement data to different priority classes are application dependent. Data segmentation may be implemented in a distributed way assuming that nodes can distinguish important observations from unimportant ones in a real time fashion. On the other hand, data segmentation may be determined offline if nodes can judge the importance of data based on locations or historical observations as in some sensor network applications.

To disseminate source blocks and perform decentralized encoding in the network, our protocol uses the characteristic of *geometric networks*, where each node is identified with a point in a geometric space. Such networks include instances of sensor networks and P2P networks. In particular, the sensors usually know their locations since the collected data are more useful if their generation locations are known. In P2P networks, Distributed Hash Tables (DHT), *e.g.*, Chord [3], are widely used to improve the network scalability, where each node has a unique ID in a one-dimensional geometric space. We further assume a *geometric routing protocol* can route source blocks to a random point in the geometric network such as GPSR [4] in sensor networks, and DHT routing protocols in P2P networks.

In this work, we assume a *strict* priority model for decoding, such that the data at higher priority levels are strictly more preferable and are decoded before those at lower priority levels. This model describes a wide range of scenarios in practical applications, including multi-resolution sensor image dissemination [5], layered data compression [6], and any other application that requires sequential decoding based on priority. It is also possible to consider a less stringent priority model, where obtaining a large amount of low priority data may be preferable to obtaining a small amount of high priority data. However, such a model requires the specification of an application-specific utility function over the priority levels. This is outside the scope of this paper and remains an open problem for future research.

III. Priority Random Linear Codes

We introduce the design and performance analysis framework for two distributed priority random linear coding schemes, termed *Stacked Linear Codes* (SLC) and *Progressive Linear Codes* (PLC). An important outcome of the proposed analysis is to derive feasibility regions for designing the distribution of coded blocks over different priority levels given certain decoding constraints.

A. Stacked and Progressive Linear Codes

Both SLC and PLC are based on Random Linear Codes (RLC), which were used as a distributed implementation for network coding [7]. Given N source blocks $x_1, x_2, \ldots, x_N$, RLC generates each coded block c_i as a linear combinations of *all* N source blocks in the following form: $c_i = \sum_{j=1}^{N} \beta_{i,j} x_j$, where the *coding coefficients* $\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,N}$ are randomly chosen from a Galois field. Such an encoding process for a coded block essentially constructs a linear equation where the unknown variables are the source blocks, given the coding coefficients $\beta_{i,j}$ and the coded block c_i are known. The decoding process of RLC on M coded blocks solves the M linear equations constructed by the encoding process, where $M \geq N$.

The priority coding schemes deviate from RLC in that most coded blocks are not linear combinations of *all* source blocks,

$$\begin{bmatrix}\beta_{1,1} & \beta_{1,2} & \beta_{1,3}\\ \beta_{2,1} & \beta_{2,2} & \beta_{2,3}\\ \beta_{3,1} & \beta_{3,2} & \beta_{3,3}\end{bmatrix} \begin{bmatrix}\beta_{1,1} & 0 & 0\\ 0 & \beta_{2,2} & \beta_{2,3}\\ 0 & \beta_{3,2} & \beta_{3,3}\end{bmatrix} \begin{bmatrix}\beta_{1,1} & 0 & 0\\ \beta_{2,1} & \beta_{2,2} & \beta_{2,3}\\ \beta_{3,1} & \beta_{3,2} & \beta_{3,3}\end{bmatrix}$$

(a) RLC (b) SLC (c) PLC

Fig. 1. A comparison among three coding schemes. $\beta_{i,j}$ is a nonzero coding coefficient. Three source blocks belong to two priority levels, where the first one is in level 1 and the second and the third source block are in level 2.

$$\begin{bmatrix}4 & 0 & 0\\ 0 & 12 & 9\\ 0 & 37 & 6\end{bmatrix} \qquad \begin{bmatrix}4 & 0 & 0\\ 5 & 12 & 9\\ 7 & 37 & 6\end{bmatrix}$$

(a) SLC (b) PLC

Fig. 2. Three pairs of corresponding coded blocks in SLC and PLC, where the first pair is in level 1 and the remaining two are in level 2.

but a *subset* of source blocks. In SLC, the source blocks are encoded in different levels separately. Thus, the kth set of coded blocks are created by encoding all the source blocks in the kth level, *i.e.*, $c_i = \sum_{j=b_{k-1}+1}^{b_k} \beta_{i,j} x_j$, where $\beta_{i,j}$ is a nonzero random number uniformly chosen from a Galois field and c_i denotes the coded block. In PLC, the source blocks are encoded progressively in descending priority. In particular, the kth level coded blocks are encoded from source blocks between levels 1 and k, *i.e.*, $c_i = \sum_{j=1}^{b_k} \beta_{i,j} x_j$. Fig. 1 illustrates these two coding schemes and RLC by simple examples, where the matrix form of the coding coefficients of three coded blocks is shown. Specifically, the figure illustrates the setting where three source blocks belong to two priority levels, where the first one is in level 1 and the second and the third source block are in level 2.

Both SLC and PLC enjoy the advantage of allowing partial recovery of a subset of the source blocks, even when the number of accumulated coded blocks is less than the total number of source blocks. In the examples of Fig. 1, RLC requires at least three coded blocks to decode any useful information. However, for both PLC and SLC, as long as the first coded block is received, the first source block can be decoded.

Furthermore, with SLC, because the source blocks in each level are coded separately, the decoding results of different levels in SLC are independent. With PLC, to decode the source blocks in level k, all the source blocks between levels 1 and $k-1$ must be already decoded, or be decoded at the same time. However, we can show that PLC outperforms SLC in terms of the number of required coded blocks to recover the same set of source blocks, as stated in Theorem 1 below.

We first define the *corresponding coded blocks* at level k of SLC and PLC as the coded blocks that share the same coding coefficients for the source blocks at level k. For example, in Fig. 2 (a) and (b), the 1×1 submatrix at the top left corner and the 2×2 submatrix at the bottom right corner are identical, leading to corresponding coded blocks. Furthermore, we say that a set of coded blocks C_1 in SLC is *corresponding* to another set of coded blocks C_2 in PLC, if each coded block in C_1 is corresponding to one coded block in C_2 and vice versa. Then, we have the following theorem.

Theorem 1: If SLC can decode the first k priority levels following the strict priority model from a set of coded blocks C_1, then PLC can at least decode the first k priority levels from C_2, the corresponding set of C_1. The reverse is not true.

Proof: We prove the theorem by induction. First, we show the basis is true. When SLC can decode level 1, PLC can also decode level 1, since the corresponding coded blocks in level 1 are identical. Furthermore, the decoding algorithms are also identical for both coding schemes in level 1 as they reduce to RLC.

Second, we show the induction step is true. Assume that the statement is true for k, we need to show it is also true for $k+1$. If SLC can decode the first $k+1$ levels, PLC can decode the first k levels by the induction hypothesis. Given the first k levels are both known in SLC and PLC, and the corresponding coded blocks at level $k+1$ have the same coding coefficients for the source blocks in the $k+1$th level, the linear system induced by the coded blocks in level $k+1$ of SLC and PLC are identical. Therefore, since SLC can decode the source blocks in level $k+1$, PLC can also decode the source blocks in level $k+1$. Hence, PLC can decode the source blocks in the first $k+1$ levels.

We show the reverse is not true by a counter example. Assume there are two source blocks such that the first source block belongs to level 1 and the second belongs to level 2. The coded blocks in level 2 of PLC are the linear combination of both source blocks. Hence, PLC can decode all source blocks with high probability, when receiving two linearly independent coded blocks in level 2. However, SLC cannot decode the first source block with two corresponding coded blocks in level 2. $\square$

The implication of Theorem 1 is clear when noting that, given the same set of source blocks and a set of coded blocks C_1 encoded by SLC, PLC can encode the corresponding coded blocks C_2 of C_1 by choosing the same coding coefficients for the kth level source blocks inside the kth level coded blocks. This can be achieved by using the same random seed. Similarly, SLC can also encode C_1 corresponding to C_2 encoded by PLC.

B. Partial Decoding Algorithms

Next, we describe decoding algorithms that can be used to partially decode source blocks from a set of coded blocks accumulated in a data collecting server. For SLC, the partial decoding algorithm is essentially the decoding algorithm of RLC for the coded blocks in each level. Once the accumulated coded blocks in a level are sufficient to decode all the source blocks in this level, they are decoded despite the source blocks in other levels may not be decoded.

For PLC, we use *Gauss-Jordan elimination* rather than usual Gaussian elimination since it is unable to partially solve a underdetermined linear system. Gauss-Jordan elimination transforms a matrix to its *reduced row-echelon form* (RREF) [8] (*e.g.*, Fig. 3(3)). The benefit of the RREF is that, given the first k unknown variables can be solved with the first k rows, once these k rows have been processed, the first k elements of the resulting vector on the right-hand-side of the equations constitute the partial solution. Therefore, with Gauss-Jordan

$$\begin{bmatrix} 12 & 91 & 26 & 47 & 35 & 159 \\ 141 & 8 & 17 & 0 & 0 & 0 \\ 71 & 178 & 0 & 0 & 0 & 0 \\ 51 & 62 & 88 & 124 & 3 & 0 \\ 81 & 59 & 193 & 0 & 0 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 239 \\ 0 & 0 & 0 & 0 & 1 & 5 \end{bmatrix}$$

(b)

Fig. 3. (a) The decoding matrix. (b) The RREF of the decoding matrix after Gauss-Jordan elimination.

elimination, the decoding process can be *progressive*. The decoding process starts as soon as the first coded block has arrived, and decodes coded blocks as soon as they are decodable, when new coded blocks are accumulated. Thus, the data collecting server can stop collecting coded data once the partial decoded data fulfill the application requirement. For example, Fig. 3 shows the decoding matrix and its RREF after Gauss-Jordan elimination. From the RREF, we observe that the first 3 source blocks are partially decoded from the 5 coded blocks.

C. Decoding Performance

Under the strict priority model, it is natural to characterize the decoding performance of SLC and PLC by m' *decoding constraints*, in the form (M_i, k_i), where $1 \le i \le m'$, and the ith tuple refers to the constraint that given M_i randomly accumulated coded blocks, on average, the first k_i levels of source blocks can be decoded. It is apparent that the smaller M_i is, the more severe node failures that the data in the first k_i levels can survive.

We introduce the protocol parameters that can be controlled to achieve different decoding performance: *priority distribution*, which is defined as the percentage of the coded blocks of each level among all coded blocks. The priority distribution can be attained in a decentralized way by the protocols presented in Sec. IV. By adjusting the priority distribution, the coding schemes can achieve different decoding constraints. For example, if we increase the percentage of coded blocks in the first k_i levels, the probability to accumulate such coded blocks is increased. Hence, we can fulfill more stringent decoding constraint (M_i, k_i) with a smaller M_i. However, given that the total storage space in all nodes is fixed, the consequence is that the percentage of coded blocks from level k_i+1 to n decreases such that the number of required randomly accumulated coded blocks to decode the source blocks in these levels will increase. Hence, the priority distribution must be carefully chosen in order to meet all decoding constraints.

We then derive the numerical relation between the priority distribution and the decoding constraints for SLC and PLC. With such numerical analysis, we can formulate different optimization problems to search for the feasible priority distribution for a particular set of decoding constraints. The notations used throughout this section are summarized in Table I.

1) Decoding Performance of SLC: We introduce the random variable X to denote the number of priority levels that can be decoded from M randomly accumulated coded blocks.

N:	total number of source blocks
n:	number of priority levels
M:	number of coded blocks
m:	maximal number of coded blocks that can be decoded from M coded blocks, *i.e.*, $\arg\max_i\{b_i \le M\}$
x_i:	the ith source block
a_i:	number of source blocks in level i
b_i:	number of source blocks in the first i levels
p_i:	priority distribution, *i.e.*, the probability that a coded block belongs to level i
D_i:	number of coded blocks in level i
$D_{i,j}$:	number of coded blocks between level i and level j, *i.e.*, $\sum_{k=i}^{j} D_k$
$B(n,k,p)$:	binomial term $\binom{n}{k}p^k(1-p)^{n-k}$
$P_{i,j}$:	sum of priority probabilities from level i to level j, *i.e.*, $\sum_{k=i}^{j} p_k$

TABLE I
TABLE OF NOMENCLATURE

The expected value of X is then

$$\mathrm{E}(X) = \sum_{k=1}^{n} k\Pr(X=k). \tag{1}$$

To compute (1), we derive $\Pr(X=k)$. In SLC, each level corresponds to a RLC and is independent of other levels. That is, a_i source blocks in level i can be decoded with high probability as long as the number of accumulated coded blocks in level i is larger than or equal to a_i[1]. To decode exactly k levels of source blocks, we need two sets of conditions. First, the source blocks of the first k levels can be decoded. Second, the source blocks in level $k+1$ cannot be decoded. These conditions are summarized as the following events:

$$\begin{aligned} A_i &= \{D_i \ge a_i\} \quad \text{for} \quad i = 1, 2, \ldots, k \\ A_{k+1} &= \{D_{k+1} \le a_{k+1} - 1\}. \end{aligned} \tag{2}$$

where D_i is the number of coded blocks in level i. Therefore, we have $\Pr(X=k) = \Pr(A_1 \cap A_2 \cap \cdots \cap A_{k+1})$.

Let $\mathbf{D}$ denote the vector of $[D_1, \ldots, D_{k+1}, D_{k+2,n}]$, where $D_{i,j}$ is the number of coded blocks between level i and level j, *i.e.*, $\sum_{k=i}^{j} D_k$. The sum of the elements in $\mathbf{D}$ should be the total number of the coded blocks M,

$$M = D_1 + \ldots + D_{k+1} + D_{k+2,n}. \tag{3}$$

Moreover, D_{k+1} and $D_{k+2,n}$ should meet the constraints:

$$\begin{aligned} D_{k+1} &\ge 0, \\ D_{k+2,n} &\ge 0. \end{aligned} \tag{4}$$

Since $\mathbf{D}$ is a partition of M, the probability that a given vector $\mathbf{D}$ appears is a function of $\mathbf{D}$ and the priority distribution $\mathbf{P} = [p_1, \ldots, p_{k+1}, P_{k+2,n}]$:

$$f(\mathbf{D}, \mathbf{P}) = \binom{M}{D_1, \ldots, D_{k+1}, D_{k+2,n}} p_1^{D_1} \cdots p_{k+1}^{D_{k+1}} P_{k+2,n}^{D_{k+2,n}}. \tag{5}$$

Let B denote the set of vectors satisfying the constraints (2), (3), and (4). The probability to decode k levels is

$$\Pr(X=k) = \sum_{\mathbf{D} \in B} f(\mathbf{D}, \mathbf{P}). \tag{6}$$

[1]We assume a sufficiently large Galois field such as GF(2^8) is used to generate coding coefficients.

Then, we can compute the expected number of decoded levels in (1). We use a similar efficient algorithm in [9] to compute (6) with a complexity of $O(M^2(k+2)\log(k+2))$ by dynamic programming instead of simply enumerating the vectors in B, which has complexity $O(M^{k+1})$. We present the details of the efficient algorithm in Appendix I.

2) Decoding Performance of PLC: We again use X to denote the number of levels that can be decoded from M random coded blocks. Hence, the expected number of decoded levels $\mathrm{E}(X)$ can be computed by (1), by first deriving the probability to decode k levels of source blocks $\Pr(X = k)$, which is the probability that there is an invertable $b_k \times b_k$ submatrix W at the left of the decoding matrix and the elements in the submatrix at the right of W are all zero after row sorting on the decoding matrix as illustrated in Sec. III-B. We then have

Theorem 2: PLC decodes the source blocks in the first k levels if and only if events $A_1, \ldots, A_m$ all happen, where

$$\begin{aligned} A_i &= \{D_{i,k} \geq b_k - b_{i-1}\} \quad \text{for } i = 1, \ldots, k, \\ A_j &= \{D_{k+1,j} \leq b_j - b_k - 1\} \text{ for } j = k+1, \ldots, m, \end{aligned} \quad (7)$$

where m is the maximal number of coded blocks that can be decoded from M coded blocks, *i.e.*, $\arg\max_i\{b_i \leq M\}$, and $b_0 = 0$. The proof of this theorem follows immediately from the following lemmas, whose proofs are given in Appendix II and III, respectively.

Lemma 3: The source blocks in the first k levels can be decoded from the coded blocks between level 1 and level k if and only if events $A_1, A_2, \ldots, A_k$ all happen.

Lemma 4: Given the source blocks in the first k levels are decoded, none of the source blocks between level $k+1$ and level m can be decoded if and only if events $A_{k+1}, A_{k+2}, \ldots, A_m$ all happen.

Thus, the probability that PLC decodes k levels is

$$\Pr(X = k) = \Pr(\cap_{i=1}^{m} A_i). \quad (8)$$

The detailed derivations of (8) is shown in Appendix IV, where approximations are used to reduce computational complexity.

D. Designing Priority Distribution under Decoding Constraints

With the analytical result presented above, we formulate a numerical feasibility problem to design the priority distribution, $p_1, p_2, \ldots, p_n$, under a given set of decoding constraints, defined in Sec. III-C. The obtained feasibility region can be used to optimize the design of priority coding. Since the optimization objectives are application dependent, instead of limiting our analysis on any such particular objective, here we demonstrate the effectiveness of our general approach by the following feasibility formulation.

Let X_{M_i} denote the random variable representing the number of levels that can be decoded from M_i coded blocks. The priority distribution must satisfy the constraints:

$$\mathrm{E}(X_{M_i}) \geq k_i, \qquad \text{for} \quad i = 1, 2, \ldots, m', \quad (9)$$

where $\mathrm{E}(X_{M_i})$, derived in (1), is a function of the priority distribution. In addition, we may impose a special constraint to ensure that the number of coded blocks to recover all source blocks is controlled within a reasonable range:

$$\Pr(X_{\alpha N} = n) \geq 1 - \epsilon, \quad (10)$$

where N is the total number of source blocks, α is a number greater than 1, and ϵ is a small positive number close to 0. This constraint guarantees that the number of coded blocks to recover all source blocks is smaller than αN with high probability. Finally, the priority distribution must satisfy the following constraints according to the definition of probability:

$$\begin{aligned} & p_i \geq 0, \\ & \sum_{j=1}^{n} p_j = 1, \text{ for } i = 1, \ldots, n \end{aligned} \quad (11)$$

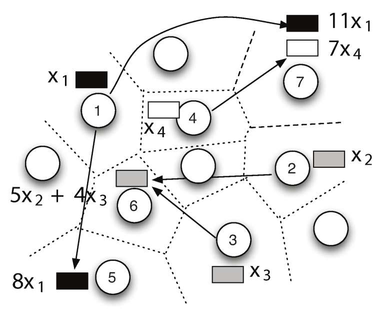
We emphasize that the constraints defined by in (9), (10), and (11) are fairly general. They can be a building block to combine with other constraints and optimization objectives to determine the priority distribution with a wide range of diverse goals.

IV. Distributed Encoding Algorithms

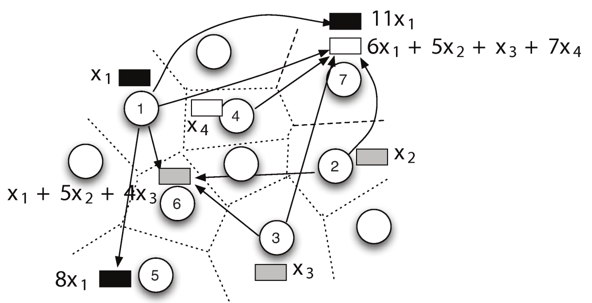
In this section, we describe a protocol to disseminate the source blocks and the distributed encoding algorithm to encode them in the network. Fig. 4(b) shows an example. Node 1 disseminates its source block x_1 to node 5, 6, and 7. Similarly, node 2, 3 and 4 disseminate their source blocks x_2, x_3, x_4 to a subset of nodes in the network. A node encodes all received source blocks to one or more coded blocks by random linear combination. For example, node 7 produces two coded block, $11x_1$ and $6x_1 + 5x_2 + x_3 + 7x_4$.

In a nutshell, the protocol should encode source blocks in a way such that the coded blocks constitute an erasure code with the priority coding distribution described in Sec. III-D. We summarize the protocol requirements in more details. First, the protocol must satisfy the *coding requirements* imposed by SLC and PLC. For example, for SLC, the protocol must deliver different source blocks in the same level to the same set of *caching* nodes for encoding and storage. As an illustration, Fig. 4(a) shows that node 2 and 3 transmit x_2 and x_3, the two source blocks in level 2, to the same node, node 6, to encode them together. Furthermore, the dissemination protocol needs to ensure the designed priority distribution for the coded blocks in the network. Second, the dissemination protocol should be efficient, due to the energy constraints in wireless sensor networks, or the bandwidth conservation requirements in P2P networks. The ideal protocol will disseminate a source block to a node only if the source block will be encoded with the coded blocks on that node. For example, x_2 should be sent to only node 6 and 7 for encoding in Fig. 4(b). Finally, the protocol should be implemented in a fully distributed way. Our protocol achieves these requirements by utilizing the characteristic of geometric networks (described in Sec. II), and the sparse coding result from [2].

To meet the coding requirement to encode source blocks generated from different locations together, all nodes should be aware of the same subset of nodes to cache coded blocks of the same priority level. Furthermore, this subset of nodes should



(a)



(b)

Fig. 4. Distributed encoding with (a) SLC (b) PLC. The rectangles and circles denote data blocks and nodes, respectively. Black, gray, and white rectangles represent the data blocks in priority level 1, 2, and 3, respectively. The dotted lines define the Vononoi diagram of the nodes.

be *randomly* distributed among all nodes to tolerate different failure patterns. To achieve these goals, in our protocol, all nodes memorize the same set of caching nodes. Then the protocol selects the same random subsets from them to cache coded blocks in different priority levels. To memorize the same set of caching nodes without actually storing the addresses of all of them, all nodes are assigned with a common random seed such that each node can use this random seed to generate the same set of M random points in the geometric space. These random points are used to identify the nodes to cache the coded blocks. In particular, a random point corresponds to a coded block, and the node closest to the random point is used to cache the coded block corresponding to the random point. Hence, the source blocks produced in one time slot are encoded into M coded blocks in the network, and M is upper-bounded by the average total storage space allocated for the source blocks produced in one time slot in the network. In the following, we use random point and coded block interchangeably. In Fig. 4(a) and (b), the dotted lines define the Vononoi diagram [10] of all nodes such that all points (coded blocks) in a polygon are cached on the node belong to that polygon. For example, the random points corresponding to coded blocks $11x_1$ and $7x_4$, are cached on node 4 in Fig. 4(a). Finally, the protocol disseminates all source blocks the nodes closest to the M random locations in the network by a geometric routing protocol (described in Sec. II).

Upon receiving a new source block x, the node in charge of the random location will encode it with the coded block c in that location, with $c = c + \beta x$, where β is a coding coefficient randomly chosen from a Galois field. Fig. 4 illustrates the destinations of source blocks according to their priorities. Let p_i denote the percentage of coded blocks in level i. For SLC, the coded blocks in a particular level are encoded from the source blocks in the same level. Hence, we divide the M random locations to n parts, where the ith part has Mp_i locations and is used to store the coded blocks for the ith level. The source blocks in level i are *only* disseminated to the ith part of random locations. For PLC, the coded blocks in level i are encoded from the source blocks from level 1 to level i. Therefore, the source blocks in level i are *only* disseminated to the set of $M(\sum_{j=i}^{n} p_j)$ locations from the ith to the nth part of random locations. As an example, Fig. 4(b) shows that the source blocks in level 1 are disseminated to coded blocks in all priority levels, whereas the source blocks in level 2 are disseminated to the coded blocks in level 2 and level 3.

Since each node is in charge of a small area in the geometric space, multiple random locations may fall on the same node such that each node stores multiple coded blocks, and the number of coded blocks on each node is generally not equivalent because of different area sizes and randomness. It is well known that "the power of two choices" can be used to achieve load balance with the maximal load $\Theta(\ln \ln M / \ln 2)$ [11]. The basic idea of "the power of two choices" is as follows. To find the node to cache a coded block, two uniformly random locations are generated first. Afterwards, with a geometric routing protocol, the two nodes closest to the two random locations are located and the number of coded blocks already cached on them are obtained. The node with fewer coded blocks is then chosen to cache this coded block. To integrate this idea into our system, rather than using the original set of M random locations, we seek a different set of *load-balanced* M random locations. Without loss of generality, we consider the process to generate the i-th random location in the load-balanced set. Two random locations are generated as the candidates to be the i-th random location. After locating the two nodes nearest to these two random locations, the protocol uses the random location corresponding to the node with the least load as the i-th random location in the load-balanced set.

In the above protocol, each source block is disseminated to all locations in its corresponding subset of the M random locations. Dimakis *el al.* [2] have shown that for RLC, with $O(\ln N)$ nonzero coding coefficients on each row, the decoding matrix can be inverted with high probability. This reduces the number of source blocks need to be disseminated from N locations to $O(\ln N)$ locations. Clearly, SLC enjoys such a result since it is essentially composed of n RLC. It is easy to see PLC also benefits from such a result, which is further confirmed by simulations in Sec. V-D.

The asymptotic result in [2] serves as a guideline. However, we would like to know the exact number of locations needed to disseminate a source block to, when implementing real systems. In the following, we compute γ_i, referred to as *density* hereafter, the lower bound of the fraction of locations that a source block needs to be disseminated to among its corresponding random locations. Our computation is based on the following observation. In order to decode a source block x_j in level i from any M_i coded blocks, the data collecting server should accumulate at least a coded block encoded from x_j.

Let q_i denote the probability that the data collecting server collects a coded block encoded from source block x_j when visiting a coded block. We first calculate q_i in order to obtain γ_i. It is easy to see that the server does not obtain a coded block encoded from x_j with probability $(1-q_i)^{M_i}$ after visiting M_i random locations. Hence, the server decodes the source block x_j with a success probability $1-(1-q_i)^{M_i}$. Finally, since we have N_i source blocks in level i, the probability $(1-(1-q_i)^{M_i})^{N_i}$ to decode all of them should be close to 1, given the fact that the dissemination of all N_i source blocks are independent. Therefore, we have

$$(1-(1-q_i)^{M_i})^{N_i} \geq 1-\epsilon, \tag{12}$$

where ϵ is a small positive number close to zero. By solving (12), we have the lower bound of q_i.

Next, we describe the relation between q_i and γ_i. We first notice that the M_i coded blocks that the data collecting server collects may be any M_i out of the M total number of coded blocks. However, a node may disseminate a source block to all M coded blocks. Hence, each copy of this source block has probability $\frac{M_i}{M}$ to be encoded into any coded block among the M_i coded blocks the server visits. Furthermore, as discussed previously in this section, a source block in level i is only disseminated to $\sum_{j=i}^{n} p_i$ fraction of random locations for PLC[2], where n is the total number of priority levels. Therefore, we have $q_i = \gamma_i \frac{M_i}{M}(\sum_{j=i}^{n} p_i)$. Hence, we have

$$\gamma_i = \frac{q_i}{\frac{M_i}{M}(\sum_{j=i}^{n} p_i)} \tag{13}$$

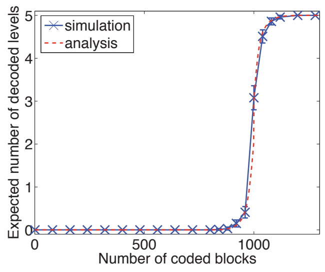
where q_i is derived from (12).

V. Performance Evaluation

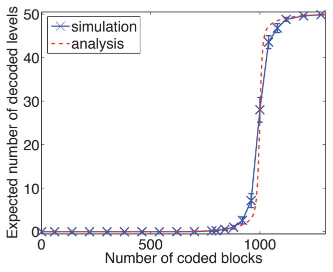
In this section, we validate our numerical analysis and study the decoding performance of SLC and PLC. In all experiments and numerical results, we measure the differentiated performance of our priority coding schemes in the *decoding curves* where the expected number of decoded priority levels are shown against the number of processed coded blocks. With an example feasibility problem, we demonstrate the effectiveness of our priority coding schemes. We also explore how much transmission cost can be saved by using a sparse decoding matrix and study the performance of our priority coding schemes under realistic settings with imperfect priority information.

To illustrate the communicational encoding cost of our coding schemes, we simulate them in a wireless network with 1024 nodes in a square with size 17×17 units. The radio range of all nodes is 1. The average number of neighbors of a node is 9.53516. Nodes are placed in a grid network with random

[2]This number is p_i for the case of SLC.

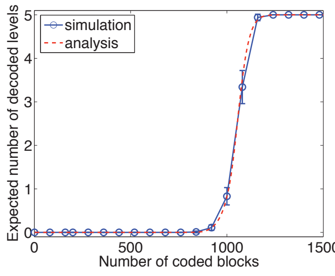


(a) Number of priority levels is 5.

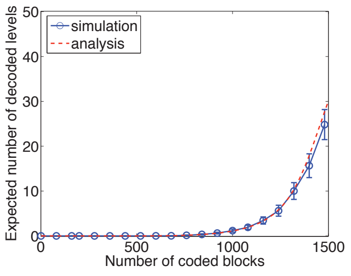


(b) Number of priority levels is 50.

Fig. 5. The analysis of PLC agrees with experiments.



(a) Number of priority levels is 5.



(b) Number of priority levels is 50.

Fig. 6. The analysis of SLC agrees with experiments.

perturbation. In particular, each node deviates from its grid location with a random distance of 0.3 along each axis. We implement greedy geographical routing algorithm to deliver a source block from the node generating this data fragment to a random location in network.

In all simulations, where GF(2^8) is used, we randomly generate a set of coded blocks according to the priority distribution and the encoding algorithms, and use the partial decoding algorithms to recover the maximal number of source blocks from the coded blocks. The number of coded blocks is varied in each experiment to observe the decoding curve. To mitigate randomness in simulations, we show, for each data point in all figures, the average and the 95% confidence intervals from 100 independent experiments.

A. Validating Numerical Analysis

For both SLC and PLC, we set the number of source blocks to 1000 and the priority distribution to uniform. Two sets of experiments are executed with 5 and 50 levels and 200 and 20 source blocks in each level, respectively. Fig. 5(a) shows that our analysis for PLC agrees with the experiments when the number of levels is 5. On the other hand, Fig. 5(b) shows that our analysis deviates slightly from experiments when the number of priority is 50. The reason is that our approximation in Sec. III-C for PLC is related to the number of levels. In particular, the more priority levels, the less accurate the approximation is. Fig. 6 shows the analysis agrees with experiments very well for SLC.

B. PLC Outperforms SLC

As we have shown in Theorem 1 of Sec. III-A, PLC outperforms SLC under the strict priority model in terms of

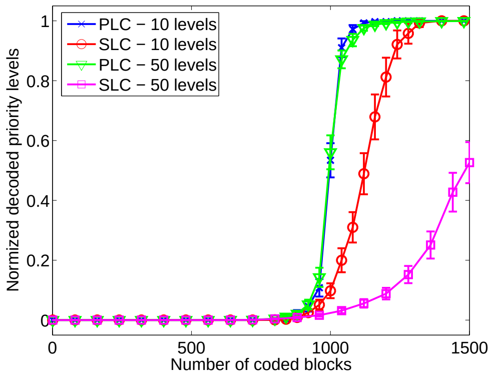


Fig. 7. PLC outperforms SLC.

the number of coded blocks to recover the same set of source blocks. In this section, we run experiments to explore the performance gap between them with the following experimental parameters. The number of source blocks is 1000. The number of levels is 10 and 50, and each level contains 100 and 20 source blocks, respectively. Fig. 7 shows that when the number of levels is 10, the decoding performance gap between SLC and PLC is modest. However, when the number of levels is 50, the performance gap between SLC and PLC is significant. Furthermore, the number of levels do not have much impact on the decoding performance of PLC, but do have significant impact on SLC. In particular, the more priority levels, the smaller amount of source blocks can be recovered by SLC with the same number of coded blocks. This is because if the number of levels is large, the source blocks in SLC are less mixed. In the extreme case where each level contains one source block, SLC degrades to the scheme of no coding. Hence, the "coupon collector" effect comes into play [12], where recovering all N source blocks requires $O(N \ln N)$ coded blocks. On the other hand, even if each level contains one source block, PLC does still mix source blocks together and enjoy the coding advantage. In the following, we only show the results for PLC.

C. *Differentiated Decoding*

We proceed to show examples using the constrained feasibility framework introduced in Sec. III-D to find a priority distribution satisfying a given set of decoding constraints. Our experimental settings are as follows. 512 source blocks are divided to three levels with 50, 100, and 362 source blocks in each level. We perform the experiments for three different sets of decoding constraints, in the form of (M_i, k_i) in (9), and are shown in the first column of Table II. For example, $(130, 1)$ in the first row of Table II requires that the expected number of priority levels decoded from 130 coded blocks is 1. We further enforce the constraint (10) with $\alpha = 2$ and $\epsilon = 0.01$ and (11) in all three sets of experiments. We solve the three numerical feasibility problems with MATLAB, using uniform distribution as the initial searching point. MATLAB terminates and produces a feasible solution which is the first solution it finds such that all constraints are satisfied. The priority distributions produced by the feasibility problem are shown in the last three columns in Table II.

	Decoding Constraints	p_1	p_2	p_3
Case 1	(130, 1) (980, 2)	0.5130	0.0791	0.4079
Case 2	(270, 1) (385, 2)	0.0739	0.5141	0.4120
Case 3	(240, 1) (500, 2)	0.3304	0.2813	0.3883

TABLE II
THE PRIORITY DISTRIBUTION SOLVED FROM THE OPTIMIZATION PROBLEM.

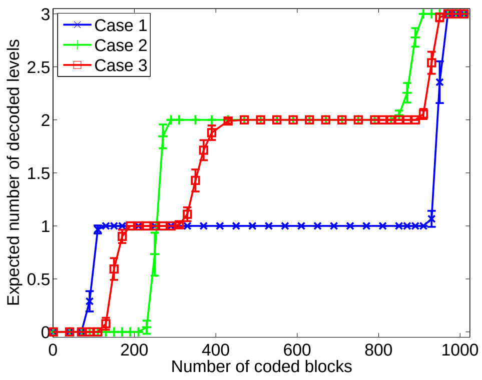


Fig. 8. The decoding curves from the priority distribution of Table II.

Fig. 8 shows the decoding curve for three priority distributions with the following observations. First, in comparison with RLC, which requires at least 512 coded blocks to decode any source block, PLC can decode the first level with only 130 coded blocks in "Case 1" and the second level with only 385 coded blocks in "Case 2". Second, all decoding curves satisfy their decoding constraints and the decoding of higher priority levels precedes lower priority levels. Finally, different decoding constraints produce significantly different decoding curves, which demonstrates the flexibility of our approach towards a diverse set of differentiated decoding requirements.

Since we are searching for one of the feasible solutions, the produced decoding curve may not exactly match the decoding constraints. For example, the decoding curve of "Case 3" climbs to level 2 with slightly more than 400 coded blocks, whereas the decoding constraint is to decode level 2 with 500 coded blocks. Moreover, it is possible that no feasible solutions are found given a set of decoding constraints. This implies the decoding constraints cannot be fulfilled.

D. *Reducing Transmission Cost with Sparse Decoding Matrices*

In this section, we investigate the communication cost of PLC. We use the same source blocks distribution as in Sec. V-C, and the decoding constraints in the third row of Table II. By (13) with $\epsilon = 0.01$, we have the densities for the three priority levels: 0.1487, 0.0558, and 0.0263. We refer to the PLC with such densities as $(0.1487, 0.0558, 0.0263)$-*sparse codes*, and the original PLC as *dense codes*. We conduct the experiments with the densities $(0.0743, 0.0279, 0.0132)$ as well for comparison. Fig. 9 shows the results. In particular, the decoding curve of $(0.1487, 0.0558, 0.0263)$-sparse codes is almost the same as the dense codes. Hence, reducing code

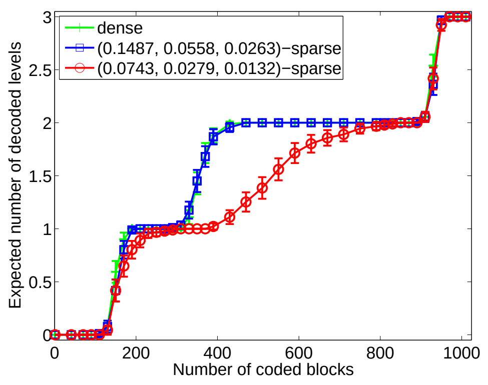


Fig. 9. The decoding curves of the dense codes and sparse codes.

TABLE III

THE DISSEMINATION COST IN TRANSMITTING A SOURCE BLOCK.

code density	average	level 1	level 2	level 3
dense	72536	14702	12655	4733
sparse	403.4	2196	520.5	123.5

densities appropriately does not degrade the coding performance. On the other hand, $(0.0743, 0.0279, 0.0132)$-sparse codes deviate from the dense codes significantly and does not meet the decoding constraints. Therefore, the code densities computed from (13) are relatively tight. In the following, we omit the result of $(0.0743, 0.0279, 0.0132)$-sparse codes since it does not satisfy the coding requirement, and we simply refer to $(0.1487, 0.0558, 0.0263)$-sparse codes as sparse codes.

We next investigate the communication cost in constructing PLC using greedy geographic routing in wireless sensor networks. The network setup is described in details at the beginning of Sec. V. We define the number of hop transmissions involved in disseminating one source block as the communication cost. Table III compares the communication cost between dense codes and sparse codes. As a benchmark, the communication cost to transmit a packet to all nodes by flooding is at least 1024 since the total number of nodes in the network is 1024. We observe that the average communication cost to deliver one source block in dense codes is much higher than flooding since the asymptotic cost is $\Theta(N\sqrt{N})$ [2]. On the other hand, the average communication cost for the sparse codes is significantly lower than flooding as the cost is $\Theta(\log N\sqrt{N})$ [2]. Finally, we notice that the cost to disseminate source blocks in higher priority levels is higher than the cost in lower priority levels. This is because the source blocks in the higher levels need to be disseminated to more random locations. Furthermore, the code densities of higher priority levels are also higher.

E. Impact of Imprecise Priority Information

Up until now, all analysis and experiments assume precise knowledge of priority information. In this section, we explore how imprecise priority information affects the performance of our algorithms with two sets of experiments. We use the same experimental settings as the dense codes in Sec. V-D except for assuming that the network nodes use different sets of imprecise priority information to derive different priority distributions. The correct priority information is always 70, 140, and 490 source blocks for the three priority levels. However, the priority distribution is computed from different sets of priority information with an error of 10 source blocks as shown in Table IV. We observe that the priority distributions derived from slightly imprecise priority information is similar to the priority distribution derived from precise information. This fact is further confirmed by Fig. 10, where the decoding curves of the experiments with precise and imprecise information are shown to be similar.

TABLE IV

THE PRIORITY DISTRIBUTION COMPUTED WITH IMPRECISE PRIORITY INFORMATION. SET 1 AND SET 2 HAVE IMPRECISE PRIORITY INFORMATION ON THE FIRST TWO AND LAST TWO PRIORITY LEVELS, RESPECTIVELY.

Type	Priority Information	p_1	p_2	p_3
Precise	(70,140,490)	0.2691	0.3380	0.3929
Set 1	(60,150,490)	0.2396	0.3807	0.3797
	(80,130,490)	0.2817	0.3203	0.3980
Set 2	(70,130,500)	0.2906	0.3225	0.3869
	(70,150,480)	0.2634	0.3642	0.3724

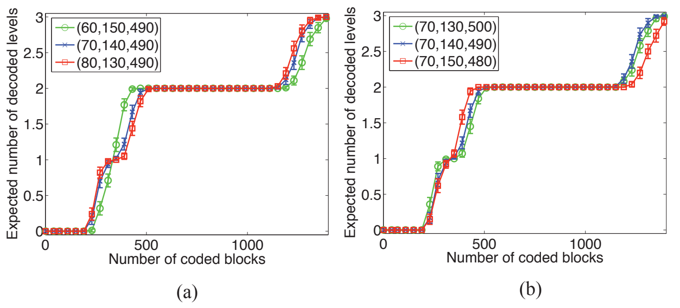


Fig. 10. (a) Decoding curves of set 1. (b) Decoding curves of set 2. The 3-tuples represent priority information used to compute the priority distribution.

VI. Related Work

In sensor networks, extensive research efforts have studied various distributed source coding schemes to save data transmissions by exploring the spatial and temporal data correlations such as in [13]. In contrast, our work along with recent research work in sensor networks [14], [2], [15], [16] and distributed storage systems [17], [18], [19] belongs to *distributed channel coding*, which provides data redundancy such that original data can be efficiently recovered when data loss are common due to node failures. However, most existing distributed channel coding schemes either recover all data or nothing. To cope with such coding disadvantage, Growth codes [14] have been proposed to maximize partially recovered data on the sink in case not all data can be recovered in sensor networks. Growth Codes treat all data equivalently despite data may have different importance in many applications. Therefore, if it is used, unimportant data may be recovered at the expense of failing to recover important data. In contrast to Growth Codes, we encode data in different priorities such that important data always have higher opportunity to be recovered.

Wang *el al*. [5] introduce a *distributed source coding* scheme to support partial decoding where partially decoded data from incomplete coded data are an approximation of the true data. The more coded data are collected and processed by the data collecting server, the closer is the decoded data to the true data.

Network coding [20], [21] and its distributed implementations utilizing random linear codes [7], [22] allow coding operations besides replication and forwarding on the intermediate nodes and achieve the maximal multicast capacity of a network. Chou *el al*. [22] consider priority encoding in network coding to achieve network multicast capacity, which is different from our problem. Chunked Codes [23] reduce the complexity of random linear codes by partitioning message to "chunks" and utilizing pre-coding. Although SLC uses similar partitioning, we focus on partial decoding whereas they concentrate on reducing complexity. Furthermore, in Chunked Codes, all data have to be pre-encoded in the source node before dissemination, whereas in our work, data are encoded in different nodes in a decentralized way.

The research work on priority encoding of data has been considered for multimedia system, *e.g.*, Priority Encoding Transmission (PET) in [24]. To the best of our knowledge, there is no known way to implement PET in a distributed way. Furthermore, we believe a naive implementation of PET under a distributed setting may incur much higher communication cost than our proposal due to the following reason. Each coded block in PET consists of the information of all priority levels. Hence, the data from any priority level are required to be disseminated to all nodes caching coded blocks. In contract, in our schemes, a coded block in most priority levels is not encoded from all priority levels but a part of them. Hence, a source block from one priority level need to be delivered to only the nodes with the *subset* of coded blocks where it is required for encoding. Therefore, our schemes incur much lower communication cost than a naively implemented PET system.

VII. Conclusion

In this paper, we introduce priority encoding under a distributed setting, where data are generated in different nodes and encoding operations are executed in a decentralized manner. The proposed priority random linear codes can be applied to a wide range of autonomous networks, including P2P and sensor networks with node churn and failure, to partially recover data cached on the network nodes. Our study is based on extensive mathematical analysis and simulation experiments. We show that with our priority coding, important data can be recovered with much fewer coded blocks as compared with random linear codes, hence they are more likely to survive under severe network instability. Furthermore, the proposed theoretical analysis provides insights into the fundamental tradeoffs in priority coding, leading to a flexible framework for optimal coding design based on application requirements.

VIII. Acknowledgment

The authors would like to thank Guanfeng Liang for his insights in our extensive discussions.

Appendix I
Efficient Computation of Eq. (6)

We compute

$$R_K^N(q_1,\ldots,q_K) = \sum_{\substack{h_1+\ldots+h_K=N \\ h_i\ge a_i, h_{K-1}\le a_{K-1}-1 \\ \text{for } i=1,\ldots,K-2}} \frac{N!}{h_1!\cdots h_K!} q_1^{h_1}\cdots q_K^{h_K}, \tag{14}$$

where $\sum_{i=1}^K q_i = 1$ and $h_i \ge 0$. Clearly, Pr$(X=k)$ in Eq. (6) is $R_K^N(q_1,\ldots,q_K)$ if $N=M$, $K=k+2$, $D_i=h_i$, $D_{k+2,n}=h_K$, $p_i=q_i$, $P_{k+2,n}=q_K$.

Let $Q_{i,j}=\sum_{l=i}^j q_l$. Then we define

$$T_{K,N,i} = R_K^N\left(\frac{q_i}{Q_{i,i+K-1}},\ldots,\frac{q_{i+K}}{Q_{i,i+K-1}}\right). \tag{15}$$

With such a definition, it is apparent that $R_K^N(q_1,\ldots,q_K) = T_{K,N,1}$, and (6) is equivalent to $T_{K,M,1}$.

Then it is easy to verify that we have the following recursive form to compute $T_{K,N,i}$:

$$T_{K,N,i} = \sum_{N_1=0}^N \frac{N!}{N_1!N_2!}\left(\frac{Q_{i,i+K_1-1}}{Q_{i,i+K-1}}\right)^{N_1}\left(\frac{Q_{i+K_1,i+K-1}}{Q_{i,i+K-1}}\right)^{N_2} \cdot T_{K_1,N_1,i}T_{K_2,N_2,i+K_1}, \tag{16}$$

where $K_2=K-K_1$ and $N_2=N-N_1$, and the constraints of $h_i\ge a_i$ and $h_{K-1}\le a_{K-1}-1$ in (14) will be integrated later when computing the initial values.

Next, we present the dynamic-programming algorithm to efficiently compute $T_{K,M,1}$, utilizing (16). The initial values are computed from (15) with the constraints $h_i \ge a_i$ and $h_{K-1}\le a_{K-1}$. For $N=0,\ldots,M$, we have

$$T_{1,N,i} = \begin{cases} 0 & \text{if } N < a_i, \\ 1 & \text{if } N \ge a_i, \end{cases} \tag{17}$$

for $i=1,\ldots,K-2$. When $i=K-1$,

$$T_{1,N,K-1} = \begin{cases} 1 & \text{if } N \le a_{K-1}-1, \\ 0 & \text{if } N > a_{K-1}-1. \end{cases} \tag{18}$$

Furthermore, we have $T_{1,N,K}=1$ for all N.

Afterwards, we build a table for dynamic programming with a standard doubling trick, utilizing (16). We have

$$T_{2^j,N,i} = \sum_{N_1=0}^N \frac{N!}{N_1!N_2!}\left(\frac{Q_{i,i+2^{j-1}-1}}{Q_{i,i+2^j-1}}\right)^{N_1}\left(\frac{Q_{i+2^{j-1},i+2^j-1}}{Q_{i,i+2^j-1}}\right)^{N_2} \cdot T_{2^{j-1},N_1,i}T_{2^{j-1},N_2,i+2^{j-1}}, \tag{19}$$

for $j=1,\ldots,\lfloor\log(K)\rfloor$, $N=0,\ldots,M$, and $i=1,1+2^j,\ldots,1+2^j(l-1)$, where $l=\lfloor K/2^j\rfloor$.

Finally, we compute $T_{K,M,1}$ based on the table built by (17), (18), and (19). We decompose K as the sum of the powers of 2.

$$K = x_{\lfloor\log(K)\rfloor}2^{\lfloor\log(K)\rfloor}+\ldots+x_1 2+x_0, \tag{20}$$

where x_i is either 0 or 1. Let X^j represent the sum of the first j items in (20). The initial values $T_{X_1,N,1}$ are

$$T_{X_1,N,1} = T_{2^{\lfloor \log(K) \rfloor},N,1}, \quad \text{for } N = 0, \dots, M, \tag{21}$$

and are given in the table built by (17), (18), and (19).

Then $T_{X_j,N,1}$ are computed as follows:

$$T_{X_j,N,1} = T_{X_{j-1},N,1}, \tag{22}$$

if $x_j = 0$. Otherwise,

$$T_{X_j,N,1} = \sum_{N_1=0}^{N} \frac{N!}{N_1! N_2!} \left(\frac{Q_{1,X_{j-1}}}{Q_{1,X_j}} \right)^{N_1} \left(\frac{Q_{X_{j-1}+1,X_j}}{Q_{1,X_j}} \right)^{N_2} \cdot T_{X_{j-1},N_1,1} T_{2^{X_j - X_{j-1}},N_2,X_{j-1}+1}, \tag{23}$$

where $N_2 = N - N_1$, and for $j = 2, \dots, \lfloor \log(K) \rfloor$. $T_{2^{X_j - X_{j-1}},N_2,X_{j-1}+1}$ are given in table built by (19).

It is easy to see the complexity of the above algorithm is determined by the four levels of loops (on variables N_1, j, N, and i_1) in building the table for dynamic programming in (19). Therefore, the computational complexity of $T_{K,M,1}$ (*i.e.*, (6)) is $O(M^2 K \log(K))$.

Appendix II
Proof of Lemma 3

We introduce two notations to facilitate the proofs of Lemma 3 here and Lemma 4 in Appendix III. First, let $M_{i,j}$ denote the submatrix of the decoding matrix with rows corresponding to the coded blocks between level i and level j and columns corresponding to the source blocks between level i and level j. For example, $M_{2,3}$ in Fig. 3(b) is $\left[\begin{smallmatrix} 17 & 0 & 0 \\ 193 & 0 & 0 \\ 88 & 124 & 3 \end{smallmatrix}\right]$. Second, let $I_{i,j}$ denote a submatrix composed of a maximal set of linearly independent rows in $M_{i,j}$. For instance, one $I_{2,3}$ in Fig. 3(b) is $\left[\begin{smallmatrix} 17 & 0 & 0 \\ 88 & 124 & 3 \end{smallmatrix}\right]$ and another $I_{2,3}$ is $\left[\begin{smallmatrix} 193 & 0 & 0 \\ 88 & 124 & 3 \end{smallmatrix}\right]$. Hence, $I_{i,j}$ is a submatrix of $M_{i,j}$ and both $M_{i,j}$ and $I_{i,j}$ have width $b_j - b_{i-1}$, assuming $b_0 = 0$. Furthermore, it is easy to see that the elements of the submatrix at the right of $M_{i,j}$ is always zero. Therefore, as long as there are b_k linearly independent rows in $M_{1,k}$, we can decode the source blocks in the first k levels. We show the detailed proof of Lemma 3 in the following.

Proof of Lemma 3: We prove the following equivalent statement. There are b_k linearly independent rows in $M_{1,k}$ if and only if events $A_1, A_2, \dots, A_k$ all happen. This equivalent statement is proved by induction. The basis is the following statement. There exists $b_k - b_{k-1}$ independent rows in $M_{k,k}$ if and only if event A_k happen. This is true since any $b_k - b_{k-1}$ rows in $M_{k,k}$ are linearly independent with high probability given the coding coefficients are randomly chosen from GF(2^8).

Assume the statement is true for $k-1$, *i.e.*, there are $b_k - b_1$ linearly independent rows in $M_{2,k}$ if and only if event $A_2, A_3, \dots, A_k$ all happen, we prove the statement is true for k. Suppose events $A_1, A_2, \dots, A_k$ all happen, there are $b_k - b_1$ linearly independent rows in $M_{2,k}$ by the induction hypothesis. We need to prove there are b_k linearly independent rows in $M_{1,k}$.

There are two types of linearly independent rows in $M_{1,k}$ besides the $b_k - b_1$ linearly independent rows expanded from $I_{2,k}$ with the first b_1 columns. First, the additional linearly independent rows may come from the coded blocks in level 1. Note that any b_1 rows from the coded blocks in level 1 can be the linearly independent rows in $M_{1,k}$. Second, the additional linearly independent rows may be expanded from the rows in $M_{2,k}$ but not in $I_{2,k}$. The number of the ith level rows in $I_{2,k}$ is smaller than $b_i - b_1$, since otherwise these rows are linearly dependent in $I_{2,k}$, contradicting with the assumption that any subset of rows in $I_{2,k}$ are linearly independent. Hence, any b_1 rows in level i expanded from the rows in $M_{2,k}$ but not in $I_{2,k}$ can be linearly independent rows in $M_{1,k}$. Therefore, it is easy to find the additional b_1 linearly independent rows in $M_{1,k}$, given $A_1 = \{D_{1,k} \geq b_k\}$ happens. Hence, there are b_k linearly independent rows in $M_{1,k}$.

Conversely, suppose there are b_k linearly independent rows (columns) in $M_{1,k}$. $M_{2,k}$ is transformed from $M_{1,k}$ in two steps. First, the first b_1 columns of $M_{1,k}$ are removed. Second, any rows with all zero in the submatrix from the previous step are also removed. Since the first b_1 columns of $M_{1,k}$ can contribute at most b_1 linearly independent columns, $M_{2,k}$ hence contains at least $b_k - b_1$ linearly independent columns. Furthermore, there are only $b_k - b_1$ columns in $M_{2,k}$. Therefore, $M_{2,k}$ consists of $b_k - b_1$ linearly independent rows. By the induction hypothesis, $A_2, A_3, \dots, A_k$ all happen. Furthermore, A_1 should happen, since the fact that b_k linearly independent rows exist in $M_{1,k}$ implies that there are more than b_k coded blocks from level 1 to level k, *i.e.*, $A_1 = \{D_{1,k} \geq b_k\}$ happens. □

Appendix III
Proof of Lemma 4

Proof: Since the source blocks in the first k levels are decoded, we focus on $M_{k+1,m}$. We use induction to prove this with the following basis statement. None of the source blocks in level $k+1$ can be decoded if and only if event A_{k+1} happens. Since $M_{k+1,k+1}$ is in the form of a RLC decoding matrix, none of the $b_{k+1} - b_k$ source blocks in level $k+1$ can be decoded if and only if the number of coded blocks $D_{k+1,k+1} \leq b_{k+1} - b_k - 1$ with high probability. Hence, the basis is true.

Assume the statement is true for $m-1$, *i.e.*, none of the source blocks in level $k+1$ to level $m-1$ can be decoded if and only if event $A_{k+1}, A_{k+2}, \dots, A_{m-1}$ happen, we need to prove the statement is true for m. Suppose $A_{k+1}, A_{k+2}, \dots, A_m$ happen, then none of the source blocks in level $k+1$ to level $m-1$ can be decoded by the induction hypothesis. Since $A_m = \{D_{k+1,m} \leq b_m - b_k - 1\}$ happens, the source blocks in level m cannot be decoded either, because it requires at least $b_m - b_k$ coded blocks in order to decode the $b_m - b_k$ source blocks from level $k+1$ to level m.

Conversely, since none of the source blocks from level $k+1$ to level $m-1$ are decoded, events $A_{k+1}, A_{k+2}, \dots, A_{m-1}$ happen by the induction hypothesis. We need to show that A_m happens. We claim the rows in the submatrix $M_{k+1,m}$ are linearly independent, given none of source blocks from level

$k+1$ to level m can be decoded. This claim can be proved by induction as well. If the coded blocks in level $k+1$ cannot be decoded, the number of rows in $M_{k+1,k+1}$ is less than the number of columns, and the rows are linearly independent with high probability, since $M_{k+1,k+1}$ is in the form of a RLC decoding matrix (all coding efficnets are nonzero). Assume that the rows in $M_{k+1,m-1}$ are linearly independent, we need to prove that the rows in $M_{k+1,m}$ are also linearly independent. Following the above hypothesis, any rows in $M_{k+1,m}$ expanded from $M_{k+1,m-1}$ are linearly independent, given all rows in $M_{k+1,m-1}$ are linearly independent. Furthermore, they are independent from all rows in level m because they have less nonzero elements than any rows in level m. Finally, any $b_m - b_k$ (or less than $b_m - b_k$) rows in level m are also linearly independent. Hence, all rows in $M_{k+1,m}$ are linearly independent as long as $D_{k+1,m} \leq b_m - b_k$. Therefore, the event $A_m = \{D_{k+1,m} \leq b_m - b_k - 1\}$ should occur; otherwise, all source blocks in level $k+1$ to m can be decoded, contradicted with the hypothesis. □

Appendix IV
Deriving Pr($X = k$) in PLC

In this appendix, we show the detailed derivation of $\text{Pr}(X = k)$ in (8). This probability can be expanded as follows by the chain rule:

$$\begin{aligned}\text{Pr}(X = k) &= \text{Pr}(\cap_{i=1}^{m} A_i)\\ &= \text{Pr}(\cap_{i=1}^{k} A_i)\text{Pr}(\cap_{i=k+1}^{m} A_i | \cap_{i=1}^{k} A_i). \qquad (24)\end{aligned}$$

Hence, we derive $\text{Pr}(X = k)$ in (24) with the following two steps: deriving $\text{Pr}(\cap_{i=1}^{k} A_i)$ and deriving $\text{Pr}(\cap_{i=k+1}^{m} A_i | \cap_{i=1}^{k} A_i)$.

A. Deriving Pr($\cap_{i=1}^{k} A_i$)

By the chain rule, $\text{Pr}(\cap_{i=1}^{k} A_i)$ is

$$\text{Pr}(\cap_{i=1}^{k} A_i) = \text{Pr}(A_k)\text{Pr}(A_{k-1}|A_k)\cdots\text{Pr}(A_1|\cap_{i=2}^{k} A_i). \quad (25)$$

To simply notations, we use $B(n, k, p)$ to denote the binomial term $\binom{n}{k}p^k(1-p)^{n-k}$. Since the number of level-k coded blocks in the M coded blocks conforms to the binomial distribution, we have

$$\begin{aligned}\text{Pr}(A_k) &= \text{Pr}(D_{k,k} \geq b_k - b_{k-1})\\ &= \sum_{z=b_k-b_{k-1}}^{M} B(M, z, p_k). \qquad (26)\end{aligned}$$

Next, we derive $\text{Pr}(A_{i-1}| \cap_{j=i}^{k} A_j)$. Define $\overline{C_i} = \{D_{i,k} \geq b_k - b_{i-2}\}$ and $C_{i,z} = \{D_{i,k} = z\}$, where $b_k - b_{i-1} \leq z \leq b_k - b_{i-2} - 1$. $\overline{C_i}$ and $C_{i,z}$ partition A_i as follows:

$$\begin{aligned}A_i &= \{D_{i,k} \geq b_k - b_{i-1}\}\\ &= \{D_{i,k} \geq b_k - b_{i-2}\} \cup (\cup_{z=b_k-b_{i-1}}^{b_k-b_{i-2}-1}\{D_{i,k} = z\})\\ &= \overline{C_i} \cup (\cup_{z=b_k-b_{i-1}}^{b_k-b_{i-2}-1} C_{i,z}). \qquad (27)\end{aligned}$$

Let W_i denote $\cap_{j=i+1}^{k} A_j$. Hence, $\text{Pr}(A_{i-1}| \cap_{j=i}^{k} A_j) = \text{Pr}(A_{i-1}|A_i \cap W_i)$. By the law of total probability on the conditional space of event $A_i \cap W_i$, we have

$$\begin{aligned}&\text{Pr}(A_{i-1}|A_i \cap W_i) = \text{Pr}(A_{i-1}|\overline{C_i} \cap W_i)\text{Pr}(\overline{C_i} \cap W_i|A_i \cap W_i)+\\ &\sum_{z=b_k-b_{i-2}}^{b_k-b_{i-1}} \text{Pr}(A_{i-1}|C_{i,z} \cap W_i)\text{Pr}(C_{i,z} \cap W_i|A_i \cap W_i), \quad (28)\end{aligned}$$

since $\overline{C_i}$ and $C_{i,z}$ are subsets of A_i. We derive the unknown elements in (28). First, we have

$$\text{Pr}(A_{i-1}|\overline{C_i} \cap W_i) = 1. \qquad (29)$$

This follows because $\overline{C_i} = \{D_{i,k} \geq b_k - b_{i-2}\}$ and $D_{i-1,k} = D_{i-1,i-1} + D_{i,k} \geq D_{i,k}$ imply $A_{i-1} = \{D_{i-1,k} \geq b_k - b_{i-2}\}$.

Second, A_{i-1} and W_i are conditionally independent given $C_{i,z}$. This is because given the number of coded blocks from level i to level k, the number of coded blocks from level $i-1$ to level k is independent of the number of coded blocks from level j to level k, where $j > i$. Hence, we have

$$\begin{aligned}\text{Pr}(A_{i-1}|C_{i,z} \cap W_i) &= \text{Pr}(A_{i-1}|C_{i,z})\\ &= \text{Pr}(D_{i-1,k} \geq b_k - b_{i-2}|D_{i,k} = z)\\ &= \text{Pr}(D_{i-1,i-1} \geq b_k - b_{i-2} - z|D_{i,k} = z)\\ &= \sum_{l=b_k-b_{i-2}-z}^{M} B(M - z, l, \frac{p_{i-1}}{1 - P_{i,k}}),\end{aligned} \qquad (30)$$

where the second and the third equality follow from the definition of A_{i-1}, $C_{i,z}$, and $D_{i,j}$. The fourth equality holds because given the number of coded blocks from level i to level k is z, the number of coded blocks in level $i-1$ in the remaining $M - z$ coded blocks is a binomial distribution where the probability to choose a coded block in level $i-1$ is $\frac{p_{i-1}}{1-P_{i,k}}$.

Third, we derive $\text{Pr}(C_{i,z} \cap W_i|A_i \cap W_i)$. We show that the exact derivation of this probability involves recursion and is computationally extremely complex. Hence, we give an approximated derivation. We start with the exact derivation as follows:

$$\text{Pr}(C_{i,z} \cap W_i|A_i \cap W_i) = \frac{\text{Pr}(C_{i,z} \cap W_i)}{\text{Pr}(A_i \cap W_i)}, \qquad (31)$$

since $C_{i,z} \subset A_i$. $\text{Pr}(A_i \cap W_i)$ can be calculated using intermediately computed result as follows:

$$\begin{aligned}\text{Pr}(A_i \cap W_i) &= \text{Pr}(\cap_{j=i}^{k} A_j)\\ &= \text{Pr}(A_k)\text{Pr}(A_{k-1}|A_k)\cdots\text{Pr}(A_i| \cap_{j=i+1}^{k} A_j),\end{aligned} \qquad (32)$$

where $\text{Pr}(A_k)$ and $\text{Pr}(A_l| \cap_{j=l+1}^{k} A_j)$, $l = i, \ldots, k-1$, are derived prior to $\text{Pr}(A_i \cap W_i)$ in (26) and (28). $\text{Pr}(C_{i,z} \cap W_i)$ in (31) can be computed as follows:

$$\begin{aligned}\text{Pr}(C_{i,z} \cap W_i) &= \text{Pr}(C_{i,z})\text{Pr}(W_i|C_{i,z})\\ &= \text{Pr}(D_{i,k} = z)\text{Pr}(\cap_{j=i+1}^{k} A_j|D_{i,k} = z)\\ &= B(M, z, P_{i,k})\text{Pr}(\cap_{j=i+1}^{k} A_j|D_{i,k} = z),\end{aligned} \qquad (33)$$

where computing $\Pr(\cap_{j=i+1}^{k} A_j | D_{i,k} = z)$ is the same problem as computing (25), although the former is defined on the conditional space on event $\{D_{i,k} = z\}$. Hence, we can use the same method in deriving (25) to derive $\Pr(\cap_{j=i+1}^{k} A_j | D_{i,k} = z)$. However, since the computation of this probability is encountered for each z and recursively, the computational cost of (25) is too high by this method. Hence, we give the approximated derivation for $\Pr(C_{i,z} \cap W_i | A_i \cap W_i)$ as follows:

$$\begin{aligned}\Pr(C_{i,z} \cap W_i | A_i \cap W_i) &\approx \Pr(C_{i,z}|A_i) = \frac{\Pr(C_{i,z})}{\Pr(A_i)} \\ &= \frac{\Pr(D_{i,k} = z)}{\Pr(D_{i,k} \geq b_k - b_{i-1})} \\ &= \frac{B(M, z, P_{i,k})}{\sum_{l=b_k-b_{i-1}}^{M} B(M, l, P_{i,k})}, \quad (34)\end{aligned}$$

where the first equality follows since $C_{i,z} \subset A_i$ and the second and third equality follow from the definition of $D_{i,j}$. The approximation in (34) is reasonable since $W_i = \cap_{j=i+1}^{k} A_j$ has much smaller impact to $\Pr(C_{i,z} \cap W_i | A_i \cap W_i)$ than $C_{i,z}$ and A_i. However, if k grows larger, the approximation becomes less accurate. We verify that such approximation is sufficiently accurate with simulations in Section V-A.

Finally, we have

$$\Pr(\overline{C_i} \cap W_i | A_i \cap W_i) = 1 - \sum_{z=b_k-b_{i-1}}^{b_k-b_{i-2}-1} \Pr(C_{i,z} \cap W_i | A_i \cap W_i), \quad (35)$$

since $\overline{C_i}$ and $C_{i,z}$ partition A_i.

By substituting (29), (30), (34), and (35) to (28), and substituting (26) and (28) into (25), we derive the formula to compute $\Pr(\cap_{i=1}^{k} A_i)$.

B. Deriving $\Pr(\cap_{i=k+1}^{m} A_i | \cap_{i=1}^{k} A_i)$

Similarly, we derive $\Pr(\cap_{i=k+1}^{m} A_i | \cap_{i=1}^{k} A_i)$. Let H denote $\cap_{i=1}^{k} A_i$. By the chain rule, we have

$$\Pr(\cap_{i=k+1}^{m} A_i | H) = \Pr(A_m|H) \cdots \Pr(A_{k+1} | \cap_{i=k+2}^{m} A_i \cap H). \quad (36)$$

First, we derive $\Pr(A_m|H)$. We define $\overline{C_1} = \{D_{1,k} \geq M - b_m + b_k + 1\}$, $C_{1,z} = \{D_{1,k} = z\}$, where $b_k \leq z \leq M - b_m + b_k$. C_1 and $C_{1,z}$ partition A_1 as follows:

$$\begin{aligned}A_1 &= \{D_{1,k} \geq b_k\} \\ &= \{D_{1,k} \geq M - b_m + b_k + 1\} \cup \cup_{z=b_k}^{M-b_m+b_k} \{D_{1,k} = z\} \\ &= C_1 \cup \cup_{z=b_k}^{M-b_m+b_k} C_{1,z}. \quad (37)\end{aligned}$$

Let W_1 denote $\cap_{i=2}^{k} A_i$, by the law of total probability on the conditional space of event $A_1 \cap W_1$,

$$\begin{aligned}&\Pr(A_m|A_1 \cap W_1) = \Pr(A_m|\overline{C_1} \cap W_1)\Pr(\overline{C_1} \cap W_1 | A_1 \cap W_1) + \\ &\sum_{z=b_k}^{M-b_m+b_k} \Pr(A_m|C_{1,z} \cap W_1)\Pr(C_{1,z} \cap W_1 | A_1 \cap W_1), \quad (38)\end{aligned}$$

since $\overline{C_1}$ and $C_{1,z}$ are subsets of A_1.

We proceed to derive the unknown elements in (38). First, because $\overline{C_1} = \{D_{1,k} \geq M - b_m + b_k + 1\}$, we have $D_{k+1,m} = M - D_{1,k} \leq b_m - b_k - 1$, *i.e.*, A_m happens. Hence, we have

$$\Pr(A_m|\overline{C_1} \cap W_1) = 1. \quad (39)$$

Second, A_m and W_1 are conditionally independent given $C_{1,z}$. This is because given the number of coded blocks from level 1 to level k, the number of coded blocks from $k+1$ to m is independent of the number of coded blocks from level i to level k, where $i > 1$. Hence, we have

$$\begin{aligned}\Pr(A_m|C_{1,z} \cap W_1) &= \Pr(A_m|C_{1,z}) \\ &= \Pr(D_{k+1,m} \leq b_m - b_k - 1 | D_{1,k} = z) \\ &= \sum_{l=0}^{b_m-b_k-1} B(M-z, l, \frac{P_{k+1,m}}{1-P_{1,k}}), \quad (40)\end{aligned}$$

where the second equality follows from the definition of A_m and $C_{1,z}$, the third equality holds because given the number of coded blocks from level 1 to level k is z, the number of coded blocks from level $k+1$ to level m in the remaining $M - z$ coded blocks is a binomial distribution where the probability to choose a coded block in level $k+1$ to level m is $\frac{P_{k+1,m}}{1-P_{1,k}}$.

Third, similar to (34) we have the approximated derivation of $\Pr(C_{1,z} \cap W_1 | A_1 \cap W_1)$ in order to avoid the infeasible computation of exact derivation:

$$\begin{aligned}\Pr(C_{1,z} \cap W_1 | A_1 \cap W_1) &\approx \Pr(C_{1,z}|A_1) = \frac{\Pr(C_{1,z})}{\Pr(A_1)} \\ &= \frac{\Pr(D_{1,k} = z)}{\Pr(D_{1,k} \geq b_k)} \\ &= \frac{B(M, z, P_{1,k})}{\sum_{l=b_k}^{M} B(M, l, P_{1,k})}, \quad (41)\end{aligned}$$

where the first equality holds since $C_{1,z} \subset A_1$ and the second and third equality follow from the definition of $D_{i,j}$. Finally, we have

$$\Pr(\overline{C_1} \cap W_1 | A_1 \cap W_1) = 1 - \sum_{z=b_k}^{M-b_m+b_k} \Pr(C_{1,z} \cap W_1 | A_1 \cap W_1), \quad (42)$$

since $\overline{C_1}$ and $C_{1,z}$ partition A_1.

By substituting (39), (40), (41), and (42) to (38), we derive the probability $\Pr(A_m|H)$.

We proceed to derive the probability $\Pr(A_{i-1} | \cap_{j=i}^{m} A_j \cap H)$ for $i = k+2$ to m. Define $\overline{C_i} = \{D_{k+1,i} \leq b_{i-1} - b_k - 1\}$ and $C_{i,z} = \{D_{k+1,i} = z\}$, where $b_{i-1} - b_k \leq z \leq b_i - b_k - 1$. $\overline{C_i}$ and $C_{i,z}$ partition A_i as follows:

$$\begin{aligned}A_i &= \{D_{k+1,i} \leq b_i - b_k - 1\} \\ &= \{D_{k+1,i} \leq b_{i-1} - b_k - 1\} \cup (\cup_{z=b_{i-1}-b_k}^{b_i-b_k-1} \{D_{k+1,i} = z\}) \\ &= \overline{C_i} \cup (\cup_{z=b_{i-1}-b_k}^{b_i-b_k-1} C_{i,z}). \quad (43)\end{aligned}$$

Let W_i denote $\cap_{j=i+1}^{m} A_j \cap H$. By the law of total probability on the conditional space of event $A_i \cap W_i$, we have

$$\begin{aligned}&\Pr(A_{i-1}|A_i \cap W_i) = \Pr(A_{i-1}|\overline{C_i} \cap W_i)\Pr(\overline{C_i} \cap W_i | A_i \cap W_i) + \\ &\sum_{z=b_{i-1}-b_k}^{b_i-b_k-1} \Pr(A_{i-1}|C_{i,z} \cap W_i)\Pr(C_{i,z} \cap W_i | A_i \cap W_i), \quad (44)\end{aligned}$$

since $\overline{C_i}$ and $C_{i,z}$ are subsets of A_i.

We derive the unknown elements in (44). First, because $\overline{C_i} = \{D_{k+1,i} \leq b_{i-1} - b_k - 1\}$ and $D_{k+1,i-1} \leq D_{k+1,i} \leq b_{i-1} - b_k - 1$, A_{i-1} happens. Hence, we have

$$\Pr(A_{i-1}|\overline{C_i} \cap W_i) = 1. \tag{45}$$

Second, A_{i-1} and W_i are conditionally independent given $C_{i,z}$. This is because given the number of coded blocks from level $k+1$ to level i, the number of coded blocks from $k+1$ to $i-1$ is independent of the number of coded blocks from level 1 to level k and from level $k+1$ to level j, where $j > i$. Hence, we have

$$\begin{aligned}\Pr(A_{i-1}|C_{i,z} \cap W_i) &= \Pr(A_{i-1}|C_{i,z}) \\ &= \Pr(D_{k+1,i-1} \leq b_{i-1} - b_k - 1 | D_{k+1,i} = z) \\ &= \Pr(z - b_{i-1} + b_k + 1 \leq D_{i,i} \leq z | D_{k+1,i} = z) \\ &= \sum_{l=z-b_{i-1}+b_k+1}^{z} B(z, l, \frac{P_{i,i}}{P_{k+1,i}}), \end{aligned} \tag{46}$$

where the second and third equality follow from the definition of A_{i-1} and $C_{1,z}$, the fourth equality holds because given the number of coded blocks from level $k+1$ to level i is z, the number of coded blocks in level i in these z coded blocks is a binomial distribution where the probability to choose a coded block in level i is $\frac{P_{i,i}}{P_{k+1,i}}$.

Third, similar to (34), we have the approximated derivation of $\Pr(C_{i,z} \cap W_i | A_i \cap W_i)$ in order to avoid the infeasible computation of exact derivation:

$$\begin{aligned}\Pr(C_{i,z} \cap W_i | A_i \cap W_i) &\approx \Pr(C_{i,z}|A_i) = \frac{\Pr(C_{i,z})}{\Pr(A_i)} \\ &= \frac{\Pr(D_{k+1,i} = z)}{\Pr(D_{k+1,i} \leq b_i - b_k - 1)} \\ &= \frac{B(M, z, P_{k+1,i})}{\sum_{l=0}^{b_i - b_k - 1} B(M, l, P_{k+1,i})}, \end{aligned} \tag{47}$$

where the first equality holds since $C_{i,z} \subset A_i$ and the second and third equality follow from the definition of $D_{i,j}$. Finally, we have

$$\Pr(\overline{C_i} \cap W_i | A_i \cap W_i) = 1 - \sum_{z=b_{i-1}-b_k}^{b_i - b_k - 1} \Pr(C_{i,z} \cap W_i | A_i \cap W_i), \tag{48}$$

since $\overline{C_i}$ and $C_{i,z}$ partition A_i.

By substituting (45), (46), (47), and (48) to (44), we derive $\Pr(A_{i-1}|H \cap \cap_{j=i}^{m} A_j)$. By substituting (38) and (44) to (36), we derive $\Pr(\cap_{i=k+1}^{m} A_i | \cap_{i=1}^{k} A_i)$. With (25) and (36), we have derived $\Pr(X = k)$ in (24).

Due to the approximations in (34), (41), and (47), the sum of $\Pr(X = k)$, where $k = 0, 1, \ldots, m$, derived in this appendix is less than 1. Therefore, we normalize these probabilities such that their sum equals 1.

References

[1] 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.

[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] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proc. of ACM SIGCOMM*, 2001.

[4] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2000.

[5] W. Wang and K. Ramchandran, "Random Distributed Multiresolution Representations with Significance Querying," in *Proc. of the Fifth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2006.

[6] K. Sayood, *Introduction to Data Compression*, 3rd ed. Morgan Kaufmann, 2006.

[7] 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.

[8] K. Hoffman and R. Kunze, *Linear Algebra*, 2nd ed. Prentice-Hall, 1971.

[9] P. Kontkanen and P. Myllymaki, "Computing the Regret Table for Multinomial Data," Helsinki Institute for Information Technology, Tech. Rep., 2005.

[10] M. de Berg, M. van Krefeld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[11] J. Byers, J. Considine, and M. Mitzenmacher, "Geometric Generalizations of the Power of Two Choices," in *Proc. of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*, 2004.

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

[13] J. Liu, M. Adler, D. Towsley, and C. Zhang, "On Optimal Communication Cost for Gathering Correlated Data through Wireless Sensor Networks," in *ACM MOBICOM*, 2006.

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

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

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

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

[18] 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.

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

[20] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[21] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

[22] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *Proc. of 41th Annual Allerton Conference on Communication, Control and Computing*, October 2003.

[23] P. Maymounkov, N. J. A. Harvey, and D. S. Lun, "Methods for Efficient Network Coding," in *Proc. of 44th Annual Allerton Conference on Communication, Control and Computing*, 2006.

[24] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority Encoding Transmission," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1737–1744, Nov. 1996.

Yunfeng Lin. Yunfeng Lin received his B.Engr. degree from Department of Computer Science and Technology, Tsinghua University, China, in 2000, and his M.Math. degree from School of Computer Science, University of Waterloo, Canada, in 2005. He is currently a Ph.D. candidate at the Department of Electrical and Computer Engineering, University of Toronto, Canada. His current research interests include performance modeling and distributed algorithm design in wireless networks.

Ben Liang. Ben Liang received honors simultaneous B.Sc. (valedictorian) and M.Sc. degrees in electrical engineering from Polytechnic University in Brooklyn, New York, in 1997 and the Ph.D. degree in electrical engineering with computer science minor from Cornell University in Ithaca, New York, in 2001. In the 2001 - 2002 academic year, he was a visiting lecturer and post-doctoral research associate at Cornell University. He joined the Department of Electrical and Computer Engineering at the University of Toronto in 2002, where he is now an Associate Professor. His current research interests are in mobile networking and multimedia systems. He won an Intel Foundation Graduate Fellowship in 2000 toward the completion of his Ph.D. dissertation and an Early Researcher Award (ERA) given by the Ontario Ministry of Research and Innovation in 2007. He was a co-author of the Best Paper Award at the IFIP Networking conference in 2005 and the Runner-up Best Paper Award at the International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks in 2006. He is an editor for the IEEE Transactions on Wireless Communications and an associate editor for the Wiley Security and Communication Networks journal. He serves on the organizational and technical committees of a number of conferences each year. He is a senior member of IEEE and a member of ACM and Tau Beta Pi.

Baochun Li. Baochun Li received his B.Engr. degree in 1995 from Department of Computer Science and Technology, Tsinghua University, Beijing, China, and his M.S. and Ph.D. degrees in 1997 and 2000 from the Department of Computer Science, University of Illinois at Urbana-Champaign. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a Full Professor. He holds the Nortel Networks Junior Chair in Network Architecture and Services since October 2003, and the Bell University Laboratories Endowed Chair in Computer Engineering since August 2005. In 2000, he was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems. In 2005, he won the Best Paper Award at the Thirteenth IEEE International Workshop on Quality of Service (IWQoS). His research interests include large-scale multimedia systems, peer-to-peer networks, applications of network coding, and wireless networks.