Differentiated Data Persistence with Priority Random Linear Codes

Yunfeng Lin, Baochun Li, and Ben Liang *

Abstract

Both peer-to-peer and sensor networks have the fundamental characteristics of node churn and failures. 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.

1 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 unreliability and energy-conservation protocols (to periodically go to energy-conserving hibernation modes).

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

*The authors are affiliated with the Department of Electrical and Computer Engineering, University of Toronto. Their email addresses are {*ylin,bli*}*@eecg.toronto.edu* and *liang@comm.utoronto.ca*.

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 are encoded from a subset of source blocks, our pre-distribution mechanism ensures that only source blocks in such a subset 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 [7] (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. 2, we describe the network model. In Sec. 3, we introduce priority random linear codes and partial decoding algorithms, with extensive analysis of their properties. In Sec. 4, we describe the pre-distribution protocol. Performance evaluation of priority random linear codes is in Sec. 5. We compare our approach with related work in Sec. 6. Finally, Sec. 7 concludes the paper.

2 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.

The measured data (possibly by multiple nodes) are segmented into source blocks. We assume that N source blocks are produced, 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 $b_{i-1} + 1 \leq j \leq b_i$.

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 [20], 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 [11] 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 [22], layered data compression [19], and any other application which 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.

3 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).

3.1 Stacked and Progressive Linear Codes

Both SLC and PLC are based on Random Linear Codes (RLC) [8]. 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 essen-

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

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

Figure 1. Example of three coding schemes.

tially 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 $M \geq N$ in order to decode N source blocks.

The priority coding schemes deviate from RLC in that most coded blocks are not linear combinations of *all* source blocks, 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 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 forms of the coding coefficients are shown. Each row in the figure shows the coding coefficients of a coded block, and the ith column of coding coefficients is multiplied by the ith source block. The example in Fig. 1 represents that three source blocks belong to two 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 allow 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 is 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 in [14], which is omitted here due to space constraint.

3.2 Partial Decoding Algorithms

Next, we describe decoding algorithms that can be used to partially decode source blocks from a set of coded blocks

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

(a)

$$\begin{bmatrix} 71 & 178 & 0 & 0 & 0 & 0 \\ 141 & 8 & 17 & 0 & 0 & 0 \\ 81 & 59 & 193 & 0 & 0 & 0 \\ 51 & 62 & 88 & 124 & 3 & 0 \\ 12 & 91 & 26 & 47 & 35 & 159 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 239 \\ 0 & 0 & 0 & 0 & 1 & 5 \end{bmatrix}$$

(b) (c)

Figure 2. (a) The decoding matrix. (b) The decoding matrix with sorted rows. (c) RREF.

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 propose to use *Gauss-Jordan elimination* rather than usual Gaussian elimination since it is unable to partially solve a underdetermined linear system. Gauss-Jordan elimination is a variant of Gaussian elimination. It transforms a matrix to its *reduced row-echelon form* (RREF) [9], in which each row contains only zeros until the first nonzero element, which must be 1. 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 elimination, the decoding process can be *progressive*. The decoding process starts as soon as the first coded block has arrived. Thereafter, the decoding process 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. More precisely, the decoding process proceeds as follows. As each new coded block is accumulated, the coding coefficients of the coded block are appended to the current decoding matrix. A pass of Gauss-Jordan elimination is performed on the existing decoding matrix — with identical operations performed on the data blocks as well — such that the matrix is reduced to RREF.

In the following, we illustrate how Gauss-Jordan elimination is used in the partial decoding for PLC. To facilitate the presentation, we first sort the rows of the decoding matrix according to the number of nonzero coefficients in each row such that the decoding matrix is approximately a lower triangular matrix, *e.g.*, Fig. 2(b). In Fig. 2(b), the first three

source blocks can be decoded since the 3×3 submatrix at the top left corner can be inverted and the elements of the submatrix at the right of it are all zero. This is further confirmed by the identity submatrix in the first three rows in the RREF of the decoding matrix, Fig. 2(c), which is the result of Gauss-Jordan elimination on five rows. Since the RREFs of two matrices are identical, if they differ only in row orders [9], we conclude Gauss-Jordan elimination can partially decode the first three source blocks after processing five coded blocks, even without row pre-sorting.

3.3 Decoding Performance

We study the decoding performance of SLC and PLC characterized by m' *decoding constraints*, in the form (M_i, k_i), where $1 \leq i \leq m'$, 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. Clearly, the smaller M_i is, the more severe node failures that the data in the first k_i levels can survive.

We further define the percentage of the coded blocks of each level among all coded blocks as *priority distribution*, which can be achieved in a decentralized way by the protocols presented in Sec. 4. 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, 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.

3.3.1 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. 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 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:

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

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:

$$D_{k+1} \geq 0, \qquad D_{k+2,n} \geq 0. \tag{4}$$

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 $\mathbf{P} = [p_1, \ldots, p_k, P_{k+2,n}]$:

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

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

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

Then, we can compute the expected number of decoded levels in (1). We use an efficient algorithm in [13] to compute (6) with a complexity $O(M \log M(k+2) \log(k+2))$ by dynamic programming and FFT, instead of simply enumerating the vectors in B, which has complexity $O(M^{k+1})$.

3.3.2 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 invertible $b_k \times b_k$ 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. 3.2. We then have

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

Theorem 1 *PLC decodes the source blocks in the first k levels if and only if events $A_1, \ldots, A_m$ all happen, where*

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

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 [14].

Lemma 2 *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 3 *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 [14], where approximations are used to reduce computation complexity.

3.4 Designing Priority Distribution

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. 3.3. 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 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:

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

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 diverse goals.

4 Distributed Encoding Algorithms

In this section, we describe a protocol to distribute the source blocks in the network and the distributed encoding algorithm. There are three requirements for such protocol. 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 random set of *caching* nodes for encoding and storage. Second, the dissemination protocol needs to ensure the designed priority distribution for the coded blocks in the network. Third, the dissemination protocol should be bandwidth efficient. 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. Our protocol achieves all three requirements by utilizing characteristic of geometric networks, which are described in Sec. 2.

In our protocol, 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. All source blocks will be disseminated to the nodes that are closest to a subset of the M random locations in the network by a geometric routing protocol, depending on they priorities. We enforce each random location stores one coded block. Therefore, M is a parameter upper-bounded by the total storage space in the network. If there are W nodes in the network, and each node can store d coded blocks, M should be smaller than Wd. 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 the randomness. We can utilize "the power of two choices" to balance the load on nodes [4], where the maximal load on all nodes is $\Theta(\ln \ln M / \ln 2)$.

After all source blocks are disseminated, 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,

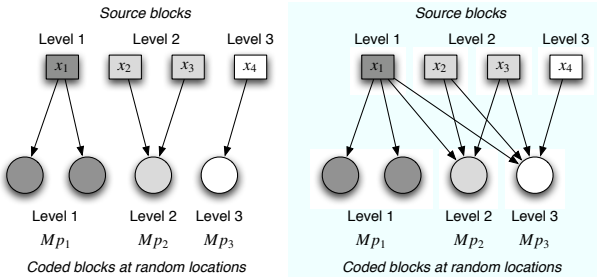


Figure 3. The destinations of source blocks in (a) SLC (b) PLC, according to their priorities.

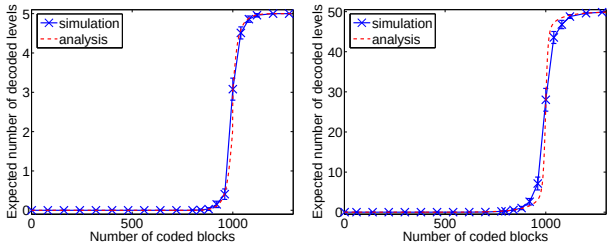
with $c = c + \beta x$, where β is a coding coefficient randomly chosen from a Galois field. Fig. 3 illustrates the destinations of source blocks according to their priorities. Let p_i denote the percentage of coded blocks in level i. For SLC, the coded blocks in a particular level are encoded from the 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.

In the above protocols, each source block is disseminated to all locations in its corresponding subset of the M random locations. Dimakis *el al.* [7] 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 results since it is essentially composed of n RLC. It is easy to see PLC also benefits from such result, which is further confirmed by simulations [14].

5 Performance Evaluation

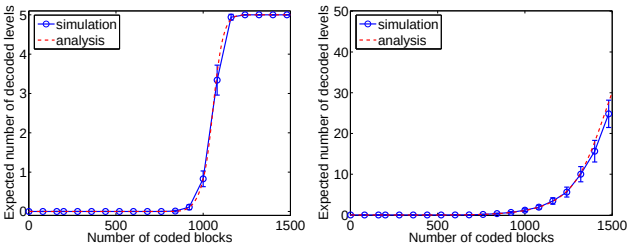
In this section, we validate our numerical analysis and study the decoding performance of SLC and PLC. In all experiments and numerical results, we measure the differentiated performance of our priority coding schemes in the *decoding curves* where the expected number of decoded priority levels are shown against the number of processed coded blocks. With an example feasibility problem, we demonstrate the effectiveness of our priority coding schemes.

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



(a) Number of priority levels is 5. (b) Number of priority levels is 50.

Figure 4. Analysis vs. simulations for PLC.



(a) Number of priority levels is 5. (b) Number of priority levels is 50.

Figure 5. Analysis vs. simulations for SLC.

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.

5.1 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. 4(a) shows that our analysis for PLC agrees with the experiments when the number of levels is 5. On the other hand, Fig. 4(b) shows that our analysis deviates slightly from experiments when the number of priority is 50. The reason is that our approximation in Sec. 3.3 for PLC is related to the number of levels. In particular, the more priority levels, the less accurate the approximation is. Fig. 5 shows the analysis agrees with experiments very well for SLC.

5.2 PLC Outperforms SLC

As we have shown in Theorem 1 of [14], PLC outperforms SLC under the strict priority model in terms of 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 are 10 and 50 and each level contains 100 and 20 source blocks, respectively. Fig. 6 show that

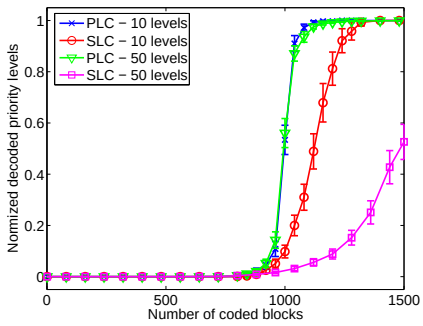


Figure 6. SLC vs. PLC.

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 less 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 is 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 [18], where recovering all N source blocks require $O(N \ln N)$ coded blocks. On the other hand, even if each level contains one source block, PLC do still mix source blocks together and enjoy the coding advantage. In the following, we only show the results for PLC.

5.3 Differentiated Decoding

We proceed to show examples using the constrained feasibility framework introduced in Sec. 3.4 to find a priority distribution satisfying a given set of decoding constraints.

Our experimental settings are as follows. 500 source blocks are divided to three levels with 50, 100, and 350 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 1. For example, $(130, 1)$ in the first row of Table 1 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 1.

Fig. 7 shows the decoding curve for three priority distributions with the following observations. First, in comparison with RLC, which requires at least 500 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 287 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.

	Decoding Constraints	p_1	p_2	p_3
Case 1	(130, 1) (950, 2)	0.5138	0.0768	0.4094
Case 2	(265, 1) (287, 2)	0	0.6149	0.3851
Case 3	(240, 1) (450, 2)	0.2894	0.3246	0.3860

Table 1. The priority distribution solved from the optimization problem.

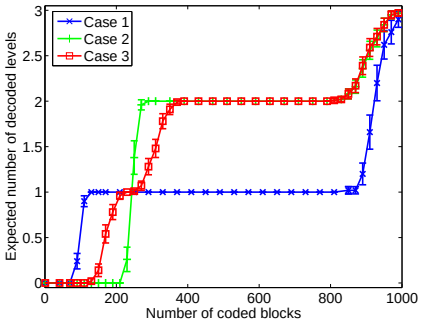


Figure 7. The decoding curves from the priority distribution of Table 1.

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 less than 400 coded blocks, whereas the decoding constraint is to decode level 2 with 450 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.

6 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 [16]. In contrast, our work along with recent research work in sensor networks [10, 7, 21, 15] and distributed storage systems [1, 23, 6] 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 [10] 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*. [22] 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 [2, 12] and its distributed implementations utilizing random linear codes [8, 5] allow coding operations besides replication and forwarding on the intermediate nodes and achieve the maximal multicast capacity of a network. Chou *el al*. [5] consider priority encoding in network coding to achieve network multicast capacity, which is different from our problem. Chunked Codes [17] 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.*, in [3]. To the best of our knowledge, there is no known way to implement such priority encoding schemes in a distributed way.

7 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 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.

8 Acknowledgment

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

References

[1] S. Acedanski, S. Deb, M. Medard, and R. Koetter. How Good is Random Linear Coding Based Distributed Networked Storage? In *NetCod*, 2005.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[3] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan. Priority Encoding Transmission. *IEEE Transactions on Information Theory*, 42(6):1737–1744, Nov. 1996.

[4] J. Byers, J. Considine, and M. Mitzenmacher. Geometric Generalizations of the Power of Two Choices. In *ACM SPAA*, 2004.

[5] P. A. Chou, Y. Wu, and K. Jain. Practical Network Coding. In *Allerton*, 2003.

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

[7] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized Erasure Codes for Distributed Networked Storage. *IEEE Transactions on Information Theory*, 2006.

[8] 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.

[9] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice-Hall, second edition, 1971.

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

[11] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *ACM MOBICOM*, 2000.

[12] R. Koetter and M. Medard. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, October 2003.

[13] P. Kontkanen and P. Myllymaki. Computing the Regret Table for Multinomial Data. Technical report, Helsinki Institute for Information Technology, 2005.

[14] Y. Lin, B. Li, and B. Liang. Differentiated Data Persistence with Priority Random Linear Codes. Technical report, http://iqua.ece.toronto.edu/papers/prioritycoding.pdf, ECE, University of Toronto, Dec. 2006.

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

[16] 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.

[17] P. Maymounkov, N. J. A. Harvey, and D. S. Lun. Methods for Efficient Network Coding. In *Allerton*, 2006.

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

[19] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, third edition, 2006.

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

[21] D. Wang, Q. Zhang, and J. Liu. Partial Network Coding for Continuous Data Collection in Sensor Networks. In *IWQoS*, 2006.

[22] W. Wang and K. Ramchandran. Random Distributed Multiresolution Representations with Significance Querying. In *IPSN*, 2006.

[23] C. Wu and B. Li. Echelon: Peer-to-Peer Network Diagnosis with Network Coding. In *Proc. of IWQoS*, 2006.