LDPC Code Design for Distributed Storage: Balancing Repair Bandwidth, Reliability, and Storage Overhead

Hyegyeong Park, *Student Member, IEEE*, Dongwon Lee, *Student Member, IEEE*, and Jaekyun Moon, *Fellow, IEEE*

***Abstract*— Distributed storage systems suffer from significant repair traffic generated due to the frequent storage node failures. This paper shows that properly designed low-density parity-check (LDPC) codes can substantially reduce the amount of required block downloads for repair thanks to the sparse nature of their factor graph representation. In particular, with a careful construction of the factor graph, both low repair-bandwidth and high reliability can be achieved for a given code rate. First, a formula for the average repair bandwidth of LDPC codes is developed. This formula is then used to establish that the minimum repair bandwidth can be achieved by forcing a regular check node degree in the factor graph. Moreover, it is shown that given a fixed code rate, the variable node degree should also be regular to yield minimum repair bandwidth, under some reasonable minimum variable node degree constraint. It is also shown that for a given repair-bandwidth requirement, LDPC codes can yield substantially higher reliability than the currently utilized Reed–Solomon codes. Our reliability analysis is based on a formulation of the general equation for the mean-time-to-data-loss (MTTDL) associated with LDPC codes. The formulation reveals that the stopping number is closely related to the MTTDL. It is further shown that LDPC codes can be designed such that a small loss of repair-bandwidth optimality may be traded for a large improvement in erasure-correction capability and thus the MTTDL.**

***Index Terms*— Distributed storage, repair bandwidth, mean-time-to-data-loss (MTTDL), low-density parity-check (LDPC) codes, factor graph.**

I. Introduction

DISTRIBUTED storage has been deployed as a solution to storing and retrieving massive amounts of data. By using the MapReduce architecture [2], the distributed feature of recent storage systems enables data centers to store big data sets reliably while allowing scalability and offering high bandwidth efficiency. However, since distributed storage systems consist of commodity disks, failure events occur frequently. As a case in point, in the Google File System (GFS) "component failures are the norm rather than the exception" [3]. Simply replicating data multiple times prevent data loss against the node failure events in GFS [3] and Hadoop Distributed File System (HDFS) [4], but the associated costs in terms of storage overhead are rather high.

In order to reduce the large storage overhead of replication schemes, erasure codes have been introduced as alternatives [5]. Reed-Solomon (RS) codes [6] are typical erasure codes having the maximum distance separable (MDS) property that can tolerate a certain maximum number of erasures given a number of parity blocks. Typically, an (n, k) RS code splits a file to be stored into k blocks and encodes them into $n = k + m$ blocks including m parity blocks [7]. These n blocks of a code are referred to as a *stripe* in distributed storage. Any k out of n blocks can be used to reconstruct the original file, which is exactly how the MDS property is defined. In practice, a (14, 10) RS code is implemented on the Facebook clusters [7] whereas a (9, 6) RS code is used in the GFS [8]. Both of these codes have high storage efficiency as well as orders of magnitude higher reliability compared to 3-replication [5], [9]. Hence, erasure coding schemes based on RS codes have become popular choices especially for archival storage systems where maintaining optimal tradeoff between data reliability and storage overhead is priority.

However, the point at issue is that MDS codes such as RS codes require high bandwidth overhead for the repair process. If a node failure event happens, the erased blocks need to be reconstructed in order to retain the same level of reliability; the amount of blocks to be downloaded for this repair task is defined as *repair bandwidth*. Since the repair bandwidth represents a limited and expensive resource for data centers, bandwidth overhead associated with the repair job should be carefully managed. For a typical (n, k) RS code, k blocks are required to reconstruct a failed block whereas replication schemes need only one block. For instance, the (14, 10) RS code has a $10\times$ repair bandwidth overhead relative to a replication scheme, consuming a significant amount of bandwidth during repair as confirmed by real measurements in the Facebook's clusters [7].

A number of recent publications have dealt with the repair bandwidth issues. Dimakis *et al.* [10] showed repair models

Manuscript received February 9, 2017; revised August 8, 2017 and October 4, 2017; accepted October 24, 2017. Date of publication November 2, 2017; date of current version February 14, 2018. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2016R1A2B4011298), and in part by the BK21 Plus. This paper was presented in part at the IEEE International Conference on Communications (ICC), 2016 [1]. The associate editor coordinating the review of this paper and approving it for publication was L. Dolecek. *(Corresponding author: Hyegyeong Park.)*

The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: parkh@kaist.ac.kr; leedw1020@kaist.ac.kr; jmoon@kaist.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2017.2769116

of MDS codes for functional repair and exact repair. Whereas exact repair restores the failed blocks by generating blocks having exact copies of the data, functional repair generates blocks that can be different from the failed blocks as long as the MDS property is maintained. They established optimal storage-bandwidth tradeoff for functional repair and coined the term *regenerating codes* for the codes that achieve optimality in this sense. Many researchers have since designed regenerating codes for exact repair that operate in some specific environments [11], [12].

In contrast to existing works, this paper focuses on coding schemes that offer significant reliability advantages, while achieving highly competitive repair bandwidth and storage overhead tradeoff. Local reconstruction codes/locally repairable codes (LRCs) and piggybacked RS codes are known methods aiming at reducing repair bandwidth sharing the same key idea. LRCs are non-MDS codes that add local parity symbols to existing RS codes to reduce repair bandwidth at the expense of an increased parity overhead. Windows Azure Storage (WAS) by Microsoft [8] and HDFS-Xorbas by Facebook [13], [14] are practical applications for LRCs.[1] Rashmi *et al.* [7] suggested piggybacked RS codes which can reduce the repair bandwidth of the RS codes without using extra storage but at the expense of code complexity.

This paper specifically explores design of low-density parity-check (LDPC) codes [15] for distributed storage applications exploiting tradeoffs of the key performance metrics such as repair bandwidth, reliability and storage overhead. LDPC codes have been considered as an alternative for conventional distributed storage coding schemes. However, most known works in this area have been about reducing the coding overhead factor of LDPC codes rather than the repair bandwidth [16], [17]. Whereas Wei *et al.* [18]–[20] showed a low latency of LDPC codes based on simulation and suggested that LDPC codes may have advantages in repair bandwidth and reliability, there has been no rigorous analysis for repair bandwidth and reliability except in [1].

In [1], the present authors have demonstrated that LDPC codes provide benefits in terms of both repair bandwidth and reliability given the same storage overhead. Since a variable node of LDPC codes is connected to a relatively small number of nodes, LDPC codes have inherent local repair property as LRCs. The repair bandwidth of an LDPC code does not depend on the length of the code. The reliability typically gets better with increasing code length. Thus, in the case of LDPC codes, the code length can be allowed to grow to achieve excellent reliability without worrying about expanding repair bandwidth as in RS codes. The only limiting factor in growing the code length in the LDPC codes is the computation and buffer requirements, but compared to the RS codes, the implementation complexity/buffer requirements of the LDPC codes grow considerably slower with code length.

This paper refines and adds to the analysis of [1]. Optimality associated with the regularity of the LDPC codes and dependency of LDPC codes' reliability on the stopping set are made precise in the form of theorems with proofs. In addition, this paper also finds LDPC codes that allow a control of the repair bandwidth while targeting high reliability. It is shown that properly designed LDPC codes can achieve very high mean-time-to-data-loss (MTTDL) at the slight expense of the repair bandwidth overhead.

Overall, the key contributions of this paper are as follows. The average repair bandwidth of the LDPC codes is formulated which leads to the observation that a regular check node degree achieves the minimum repair bandwidth given a fixed total number of edges in the factor graph. Moreover, given a fixed code rate, the variable node degree is also forced to be regular for the repair bandwidth minimality, under some reasonable minimum variable node degree constraint. For reliability analysis, a general formula for the MTTDL of LDPC codes is derived. The formula shows how the stopping number of a code directly affects reliability. It is confirmed that increasing the stopping number of the factor graph greatly enhances reliability. Regular quasi-cyclic (QC) progressive-edge-growth (PEG) LDPC codes with different code rates have been designed and compared against representative RS codes and their variants. The results show that with LDPC codes a slight relaxation of the repair bandwidth minimality may allow meaningful improvement in reliability. In summary, LDPC codes could be a powerful choice for distributed storage systems enjoying both reasonably low repair-bandwidth and very high MTTDL.

The rest of this paper is organized as follows. In Section II, we give preliminary information on LDPC codes. Section III provides repair bandwidth analysis of LDPC codes. In Section IV, a design of LDPC codes for high reliability and reasonable repair bandwidth efficiency is discussed. In Section V, reliability analysis of LDPC codes is given. Approaches to increase reliability are introduced as well. In Section VI, some specific examples of LDPC codes are discussed which show great performance on distributed storage. Simulation results that compare LDPC codes with other schemes are also given in this section. Finally, the paper draws conclusions in Section VII.

II. Preliminaries

A. LDPC Codes

An LDPC code is a class of linear block codes defined as the null space of an $m \times n$ sparse parity check matrix $\mathbf{H}$ (i.e., $\mathbf{cH}^T = 0$ if and only if $\mathbf{c}$ is a codeword), where m is the number of parity blocks and n is the length of the codeword. The LDPC codes we are concerned with in this paper are binary, which are defined over GF(2). $\mathbf{H}$ can be illustrated by a factor graph in Fig. 1, which consists of the check nodes (squares), variable nodes[2] (circles), and edges (lines between squares and circles) [21]. The factor graph describing the LDPC code is called a bipartite graph since it consists of two kinds of nodes: variable node (VN) and check node (CN). In a bipartite graph of an LDPC code,

[1]The Azure-LRC and the Xorbas-LRC are represented by (k, l, r_1) and $(k, n-k, r_2)$, respectively, where l denotes the number of local groups, r_1 represents the number of global parities and r_2 indicates the block locality.

[2]Since each variable node stores a coded data block, data node and the variable node are used synonymously in this paper.

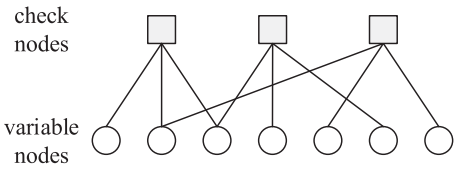

Fig. 1. A graphical representation of an LDPC code.

there are m CNs representing m parity check equations and n VNs indicating n coded blocks. CN r is connected to VN u (i.e., CN r involves VN u), if h_{ru}, the element of $\mathbf{H}$, is 1.

The ensemble of LDPC codes is specified with a variable degree distribution polynomial $\lambda(x)$ and a check degree distribution polynomial $\rho(x)$ [21],

$$\lambda(x) = \sum_{d\geq 2} \lambda_d x^{d-1} \text{ and } \rho(x) = \sum_{d\geq 2} \rho_d x^{d-1},$$

where λ_d (resp. ρ_d) is the fraction of edges connected to VNs (resp. CNs) with degree d and $\lambda(1) = \rho(1) = 1$ (i.e., the sum of coefficients is equal to one). This definition of the degree distribution pair (λ, ρ) is based on the "edge perspective". If the number of edges connected to each VN/CN is all identical, which means that the number of nonzero elements in each row and column in $\mathbf{H}$ are both constant, the corresponding LDPC code is termed a regular LDPC code. Otherwise, it is designated an irregular LDPC code.

Assuming that the parity check matrix $\mathbf{H}$ is full rank, the code rate R of an LDPC code can be represented by its degree distribution pair (λ, ρ) [22],

$$R = 1 - \frac{m}{n} = 1 - \frac{\int_0^1 \rho(x)\,\mathrm{d}x}{\int_0^1 \lambda(x)\,\mathrm{d}x} = 1 - \frac{\sum_{d\geq 2} \rho_d/d}{\sum_{d\geq 2} \lambda_d/d}. \quad (1)$$

The degree distributions from a node perspective can be expressed from an edge perspective description:

$$\Lambda_d = \frac{\lambda_d/d}{\int_0^1 \lambda(x)\,\mathrm{d}x} \text{ and } \mathrm{P}_d = \frac{\rho_d/d}{\int_0^1 \rho(x)\,\mathrm{d}x}, \quad (2)$$

where Λ_d and P_d are the fractions of VNs and CNs, respectively, with degree d.

B. Density Evolution

Density evolution is a deterministic numerical tool which tracks the fraction of erased variable nodes as iterative decoding proceeds. For the binary LDPC codes we are concerned with, the failure of a data block can be translated into an erased variable node over binary erasure channel (BEC). In this case, the expected fraction of erased data nodes at the l-th iteration, P_l, as the block length goes to infinity is presented as the recursion [21]:

$$P_l = P_0 \lambda(1 - \rho(1 - P_{l-1})) \text{ for } l \geq 1.$$

Here, for the channel erasure probability of BEC ϵ, $P_0 = \epsilon$. Decoding with an LDPC code constructed by a degree distribution pair (λ, ρ) and an initial erasure probability ϵ is successful if and only if $\lim_{l\to\infty} P_l = 0$. This condition for successful decoding is equivalent to

$$\epsilon \lambda(1 - \rho(1 - x)) < x \text{ for } x \in (0, \epsilon].$$

As the block length grows to infinity, every code in an ensemble tends to behave in the same way. Assuming that the code is sufficiently large, the performance of an individual code thus can be captured in the performance of the ensemble average. The decoding threshold ϵ^* of an LDPC code is the largest ϵ value for which the above inequality condition is satisfied. We can evaluate the decoding threshold of LDPC codes with the density evolution technique as the block length tends to infinity. The decoding threshold divides the channel into areas where data can be reliably stored and areas that are not. Density evolution therefore provides information on the maximum channel erasure probability that can be corrected by message-passing decoding averaged over all LDPC codes configured by a particular ensemble. This maximum channel erasure probability is called the decoding threshold for the ensemble.

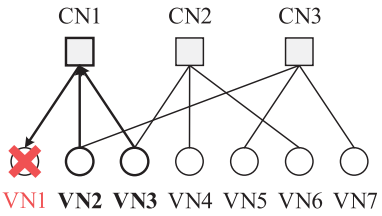

Fig. 2. A block erasure can be represented as a variable node erasure in factor graph. If a block is erased, the erased block can be recovered by downloading other blocks connected to the same check node. VN2 and VN3 are the blocks to be downloaded when VN1 fails.

III. Repair Bandwidth Analysis of LDPC Codes

In this section, the repair bandwidth of LDPC codes is described in the average sense for all nodes. LDPC codes are similar to LRCs regarding the repair process since the parity blocks of both codes are made locally from a small portion of data blocks. As shown in Fig. 2, if a block represented by node VN1 is erased, repair job can be done by downloading adjacent blocks VN2 and VN3 connected to the same check node CN1. This simple example demonstrates that LDPC codes can reconstruct erased data by using a relatively small number of blocks.

When an erased block is connected to multiple CNs, as is usually the case, we can choose a specific CN for repair. If the LDPC code is regular, any choice is equally good statistically. For an irregular LDPC code, however, the choice of a CN affects the amount of repair bandwidth since each check may have different degree. We thus define the bandwidth in the average sense. If a VN is erased, the repair bandwidth for that VN is the number of blocks downloaded averaged over all choices of CNs the VN is connected to. Note that all other VNs are assumed intact in this definition. This value is then averaged over all VN erasure positions. This final average repair bandwidth is obtained by first considering all VNs connected to each CN. For CN r with degree $d_{c,r}$, there are $d_{c,r}$ VNs attached to it, each of which will have a repair

bandwidth of $d_{c,r}-1$, assuming the other VNs attached to CN r are downloaded for repair. The total repair bandwidth associated with CN r can be said to be equal to $d_{c,r}(d_{c,r}-1)$. Summing over all m CNs, we get $\sum_{r=1}^{m} d_{c,r}(d_{c,r}-1)$. To get to the per-VN repair bandwidth, we recognize that each VN is counted as many times as its node degree in the computation of $\sum_{r=1}^{m} d_{c,r}(d_{c,r}-1)$ since each VN is connected to multiple CNs in general. Thus, this sum should be divided by nd_v, where d_v is the average VN degree, to arrive at the per-VN average repair bandwidth we are looking for. But $nd_v = md_c = \sum_{r=1}^{m} d_{c,r}$, where d_c is the average CN degree. Note that nd_v also represents the total number of edges, E, in the factor graph. We establish a definition:

Definition 1: The average repair bandwidth or simply repair bandwidth γ of an LDPC code is defined as

$$\gamma = \frac{\sum_{r=1}^{m} d_{c,r}(d_{c,r}-1)}{E}, \tag{3}$$

where m represents the number of parity blocks of an LDPC code, $d_{c,r}$ denotes the degree of check node r and E indicates the total number of edges in the factor graph.

The following lemma subsequently tells us how the CN degrees should be distributed to minimize the average repair bandwidth of (3).

Lemma 1: Given a fixed number E of edges on the factor graph, a regular check node degree minimizes the repair bandwidth of LDPC codes to $d_c - 1$, where d_c denotes the check node degree of the corresponding LDPC codes.

Proof: The repair bandwidth in (3) can be rewritten as

$$\gamma = \frac{\sum_{r=1}^{m}(d_{c,r}-\frac{1}{2})^2 - \sum_{r=1}^{m}(\frac{1}{2})^2}{E}.$$

By using the Cauchy-Schwarz inequality and the constraint $\sum_{r=1}^{m} d_{c,r} = E$, the choice

$$d_{c,1} = d_{c,2} = \cdots = d_{c,m} = E/m$$

minimizes the average bandwidth. Thus, a regular CN degree minimizes the repair bandwidth and the corresponding minimum value is $\gamma_{\min} = d_c - 1$, one less than the CN degree. ∎

Lemma 1 indicates that an LDPC code must be CN-regular in order to minimize the repair bandwidth. How about the VN degrees? Before discussing desirable VN degree characteristics in light of the repair bandwidth issue, it is natural to impose a minimum VN degree constraint such that any VN in a factor graph has a degree at least equal to some positive integer $d_{v_{\min}}$. This is due to practical reasons having to do with decodability. For example, we obviously need $d_{v_{\min}} = 1$ so that each VN is attached to at least one CN, in order to reconstruct any single VN erasure. In practical applications where a node may fail before the current failure can be repaired, we actually need a more stringent condition: LDPC codes with even degree-1 VNs have been deemed impractical [23]–[25], suggesting that we should set $d_{v_{\min}} = 2$. This type of minimum VN degree requirement calls for the VN-regularity as well. We summarize the desired CN and VN degree characteristics in the following combined statement.

Theorem 1: Among the factor graphs having no VNs with degree less than $d_{v_{min}}$, a chosen graph yields an LDPC code of rate R with minimum repair bandwidth if and only if it is both CN- and VN-regular with $d_c = d_{v_{min}}/(1-R)$ and $d_v = d_{v_{min}}$.

Proof: Lemma 1 states that a minimum-repair-bandwidth LDPC code is CN-regular with the uniform CN degree of d_c. The proof follows directly from this lemma combined with the fact that the ratio of d_v, the average VN degree, to d_c gets fixed once the code rate is given as seen in the relation: $d_v/d_c = 1-R$, where $R = k/n = (n-m)/n$. For a given R value, the average VN degree d_v must be made as small as possible to minimize d_c so that

$$\gamma_{\min} = d_c - 1 = \frac{d_v}{1-R} - 1 \tag{4}$$

is in turn minimized. But since each VN degree is greater than or equal to $d_{v_{\min}}$ by assumption, so is the average VN degree d_v. Apparently, the minimum average VN value of $d_v = d_{v_{\min}}$ is achieved when all VNs have a fixed degree of $d_{v_{\min}}$, i.e., when the factor graph is VN-regular. As for the CN degree, we obviously need $d_c = d_{v_{\min}}/(1-R)$ for minimum repair bandwidth. ∎

It is clear that the repair bandwidth of an LDPC code does not depend on the code length, but on d_c. This property makes the LDPC codes a powerful option for distributed storage. Moreover, for a fixed d_v, (4) also reveals an interesting relationship that $\gamma_{\min}$ increases with increasing R, which is due to the fact that for a fixed d_v, increasing R must also mean increasing d_c.

The regularity of minimum repair bandwidth LDPC codes automatically results in a condition on the code rate, as stated in the following corollary.

Corollary 1: An LDPC code of rate R allows minimum repair bandwidth only if $d_{v_{min}}/(1-R)$ is an integer greater than $d_{v_{min}}$.

Theorem 1 indicates that under the practical constraint of $d_{v_{\min}} = 2$, regular LDPC codes with $d_v = 2$ give the best repair bandwidth efficiency. At this point, a useful question arises: if we are allowed to increase d_v beyond 2, in hopes of improving reliability for certain applications, how rapidly do we lose repair-bandwidth efficiency? In other words, we are interested in investigating the possibility of relaxing the repair bandwidth minimality in an effort to improve reliability. Specifically, we shall compare the reliability-bandwidth trade-offs of the regular LDPC codes having a fixed $d_v = 2$ with VN-irregular LDPC codes with average VN degree beyond 2. In the process, we provide a new VN-irregular LDPC code design that allows a good erasure correction capability at the slight expense of the efficiency of the repair bandwidth. In comparing different coding schemes we consider three code rates: 1/2, 2/3 and 3/4. Theorem 1 provides the reference point for minimal repair bandwidth.

Remark 1 (Non-binary LDPC Codes): As can be seen in Fig. 3, suppose we have an $(n,k) = (5,3)$ coded system consisting of words (or symbols), each of which has q data bits in it (e.g., non-binary LDPC codes of GF(2^q) or RS codes of GF(2^q)). As an example of non-binary LDPC codes consisting of q-bit symbols, the repair bandwidth is given by $B/q \cdot (d_c-1) \cdot q = B(d_c-1)$, where B is the block size and assume for simplicity that B is a multiple of q. Likewise, for

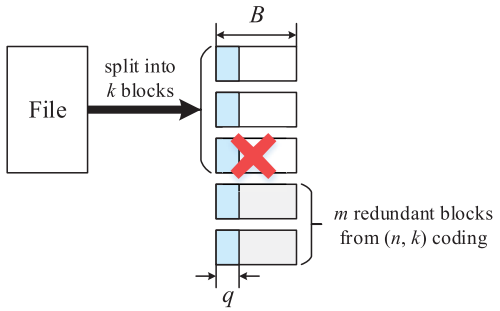

Fig. 3. An example of coded storage using multiple bits per symbol. B/q codes each of which is made up of q-bit symbols are employed across blocks of size B. This example is in systematic form (i.e., the original data is a part of the coded blocks) for illustrative purpose.

RS codes made up of q-bit words, $B/q \cdot k \cdot q = Bk$ bits are required for repairing a failed block. From the two examples above, the repair bandwidth of codes consisting of multiple bits does not change with the symbol or word size q. Hence we shall consider only binary LDPC codes in this paper without loss of generality. We will stick to the normalized value $d_c - 1$ for the repair bandwidth instead of the more general expression $B(d_c - 1)$ for simplicity.

IV. A Design for the Efficiency in Both Repair and Protection

In this section, we suggest a degree distribution design criterion to guarantee high erasure-correction-capability while enjoying reasonable efficiency in repair-bandwidth. Recall that a regular CN degree minimizes the repair bandwidth for a given number of edges in the factor graph and the minimum repair bandwidth is given by $\gamma_{\min} = d_c - 1$ in Lemma 1. While maintaining the CN-regularity, we will relax the regularity condition on VN to find the appropriate VN degree distribution. The average VN degree d_v can be computed as

$$d_v = \sum_d d\Lambda_d = \sum_d d\frac{\lambda_d/d}{\int_0^1 \lambda(x)\,\mathrm{d}x} = \frac{1}{\int_0^1 \lambda(x)\,\mathrm{d}x} = \frac{1}{\sum_d \frac{\lambda_d}{d}}, \quad (5)$$

where $\Lambda_d = \frac{\lambda_d/d}{\int_0^1 \lambda(x)\,\mathrm{d}x}$ from (2) and $\sum_d \lambda_d = 1$.

Using (5), $\gamma_{\min}$ can be rewritten as

$$\gamma_{\min} = d_c - 1 = \frac{d_v}{1-R} - 1 = \frac{1}{1-R}\cdot\frac{1}{\sum_d \frac{\lambda_d}{d}} - 1.$$

Therefore, we need to maximize $\sum_d \frac{\lambda_d}{d}$ in order to minimize $\gamma_{\min}$ for a fixed code rate constraint of R.

We propose a design of LDPC codes that balances the repair bandwidth overhead and the system reliability. To do this, we consider the following optimization problem with the VN degree distribution parameters λ_d as optimization variables.

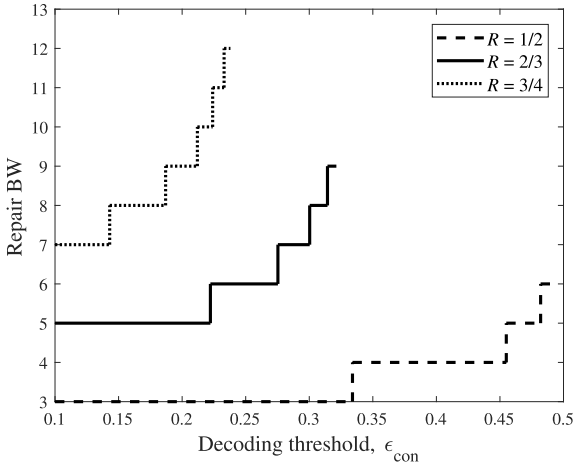

Fig. 4. Repair bandwidth and decoding threshold tradeoff curves for different code rates.

$$\text{maximize} \quad \sum_d \frac{\lambda_d}{d} \quad (6)$$

$$\text{subject to} \quad \epsilon_{\text{con}}\lambda(1-\rho(1-x)) < x, \quad x \in (0, \epsilon_{\text{con}}] \quad (7)$$

$$\sum_d \frac{\lambda_d}{d} = \frac{1}{1-R}\sum_d \frac{\rho_d}{d} \quad (8)$$

where ϵ_{con} represents the minimum level of reliability imposed. The constraint (7) ensures successful decoding as discussed in Section II.B, and (8) is the rate constraint from (1). In addition, obvious extra constraints exist on any VN degree distribution polynomial: $\sum_d \lambda_d = 1$ and $0 \le \lambda_d \le 1$.

We employ a CN degree distribution $\rho(x) = x^{d_c-1}$, forcing the CN-regularity. Hence, (7) and (8) reduce to the following constraints:

$$\epsilon_{\text{con}}\lambda(1-(1-x)^{d_c-1}) < x, \quad x \in (0, \epsilon_{\text{con}}] \quad (9)$$

$$\sum_d \frac{\lambda_d}{d} = \frac{1}{1-R}\cdot\frac{1}{d_c}. \quad (10)$$

Our optimization problem can then be stated as: for a given value of ϵ_{con}, find the distribution λ_d that will minimize d_c while satisfying (9) and (10). A small d_c value would be great for maintaining repair efficiency but to tolerate a higher value of ϵ_{con} in ensuring reliability, a compromise would have to be made on how small d_c could get. Noticing that d_c are integer values forming a relatively small search space, a clear picture on this tradeoff can be obtained conveniently by fixing d_c and then iteratively finding the maximum value of ϵ_{con} and the corresponding λ_d that satisfy (9) and (10) for each fixed value of d_c. Fig. 4 shows the relationships obtained for the minimum repair bandwidth $d_c - 1$ versus the decoding threshold ϵ_{con} for different code rates. Fig. 4 clearly reveals the maximum level of reliability that can be achieved for a given repair bandwidth or, equivalently, the minimum repair bandwidth attainable for a given level of reliability, for some fixed code rate.

We also present several examples of designed degree distributions for different target code rates of R = 1/2, 2/3 and 3/4 in Table I. The scaled maximum decoding threshold $\epsilon^*_{\text{con}}/(1-R)$

TABLE I
EXAMPLES OF DESIGNED VN DEGREE DISTRIBUTIONS

R	$\gamma = d_c - 1$	$\epsilon^*_{\text{con}}/(1-R)$	d_v	$\lambda(x)$
1/2	3	0.6680	2	x
1/2	4	0.9100	2.4997	$0.5496x + 0.1549x^2 + 0.2956x^3$
1/2	5	0.9640	2.9990	$0.4128x + 0.1789x^2 + 0.1128x^3 + 0.1371x^6 + 0.1584x^7$
1/2	6	0.9840	3.4987	$0.3394x + 0.1403x^2 + 0.1036x^3 + 0.0940x^5 + 0.0963x^6 + 0.0378x^{14} + 0.1886x^{15}$
2/3	5	0.6667	2	x
2/3	6	0.8260	2.3328	$0.5716x + 0.4284x^2$
2/3	7	0.9010	2.6662	$0.4775x + 0.0880x^2 + 0.4098x^3 + 0.0247x^4$
2/3	8	0.9430	2.9977	$0.3927x + 0.2279x^2 + 0.2907x^5 + 0.0887x^6$
2/3	9	0.9640	3.3321	$0.3469x + 0.1440x^2 + 0.1331x^3 + 0.0708x^4 + 0.1001x^8 + 0.2051x^9$
3/4	7	0.5720	2	x
3/4	8	0.7480	2.2500	$0.6704x + 0.3296x^2$
3/4	9	0.8480	2.4997	$0.4548x + 0.4462x^2 + 0.0991x^3$
3/4	10	0.8960	2.7500	$0.4486x + 0.1325x^2 + 0.2488x^3 + 0.1702x^4$
3/4	11	0.9320	2.9973	$0.3867x + 0.2270x^2 + 0.3863x^5$
3/4	12	0.9520	3.2468	$0.3495x + 0.1640x^2 + 0.1432x^3 + 0.3306x^7 + 0.0127x^8$

indicates how close the designed decoding threshold is to the BEC capacity. The repair bandwidth $\gamma = d_c - 1$ and the average VN degree d_v are also shown.

V. RELIABILITY ANALYSIS OF LDPC CODES

A. The Mean Time to Data Loss

We now provide reliability analysis for LDPC codes. In particular, we show that increasing the stopping number of the factor graph directly influences reliability. A Markov model is introduced to estimate system reliability of coding schemes. Continuous-time Markov models have been commonly used to compare reliability of storage systems in terms of the MTTDL (e.g., see [8], [9], [13], [26], [27]). Unlike the bit-error-rate (BER) or the word-error-rate (WER) performance metric, the MTTDL metric based on the Markov model considers the repair speed, which is our main interest in this paper.

Fig. 5 shows a Markov model example of the (14, 10) RS code [13]. The MTTDL is mainly influenced by the number of failures which can be tolerated before data loss as well as by the repair rate. Here, λ indicates the failure rate of a node and μ represents the repair rate of the nodes. Typically, $\mu \gg \lambda$ for storage applications. We can assume that each node fails independently at rate λ if the blocks are stored in different racks (physically separated storage units in data centers). Then, it is reasonable to ignore the possibility of burst failures. Also, the adoption of a continuous-time Markov model presupposes that only a single node failure is allowed at a given instance. Each state of the Markov model represents the number of erased blocks in a stripe. For the (14, 10) RS code, state 5 is the data loss (DL) state since five erasures in a stripe cannot be decoded. Whereas the failure rates depend on the state, the repair rates are all the same since the number of blocks to be downloaded for repair is always 10. The MTTDL can be obtained from this Markov model by calculating the mean arrival time to the DL state. The MTTDL of the MDS codes are well-established [9], [28]. The MTTDL analysis for MDS codes can be modified and extended for the LDPC codes, as discussed next.

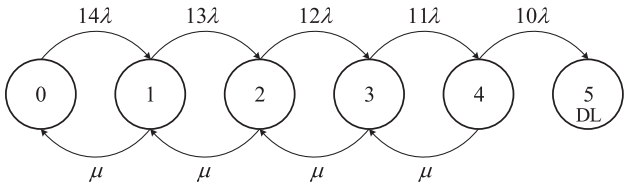

Fig. 5. Markov model of the (14, 10) RS code.

B. MTTDL of LDPC Codes

In this section, details in calculating the MTTDL for non-MDS codes are described. While the Markov model is already discussed for the LDPC codes in [29], the general formula for the MTTDL of the LDPC codes has not been given. We provide such a formula here. We also develop insights into how the MTTDL of the LDPC codes is affected by the stopping number. Before presenting the Markov model of LDPC codes, some key terms are clarified. On factor graphs, the girth indicates the shortest cycle. A stopping set [30] is a subset of variable nodes such that all check nodes connected to it are connected by at least two edges, and the stopping number is the size of the smallest stopping set.

The derivation process is similar to that for MDS codes. However, as shown in Fig. 6, LDPC codes can directly go to the data loss state with only a small number of erasures. For instance in Fig. 2, if VN5 and VN7 fail, it is impossible to repair those nodes unlike in MDS codes. To model this behavior, probability parameters are introduced to the Markov model. Probability p_i is the conditional probability that a stripe of a given code can tolerate an additional node failure given state i. This means that the code has already survived from i failures and can tolerate one more failure with probability p_i. In general, LDPC codes are designed to guarantee $p_0 = 1$ and $p_1 = 1$ since length-4 cycles are prohibited; however,

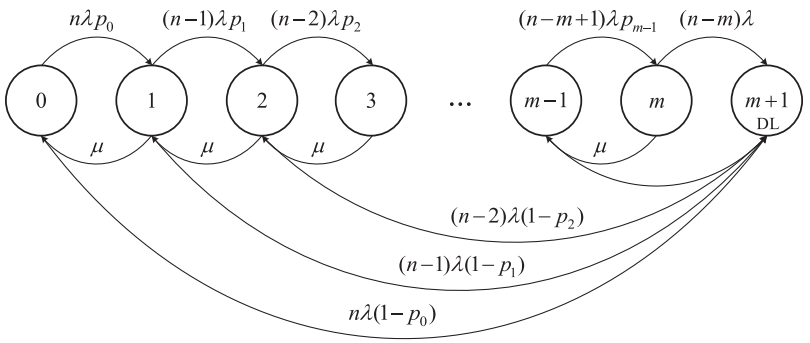

Fig. 6. Markov model of LDPC codes with m parity blocks.

other probabilities depend on the parity-check matrix of the code. If the parity-check matrix of the LDPC code is given, the p_i values can be obtained by the relationship, $p_i = q_{i+1}/q_i$, where q_i denotes the unconditional probability that a given code can tolerate i failures [29]. Such unconditional probabilities can be estimated by decoding simulation of LDPC codes on the erasure channel. Exploiting the estimators of q_i and q_{i+1}, we can obtain an asymptotically unbiased estimator of p_i given a large number of samples. This can be justified as follows.

Note: Given two random variables Y and Z, assume that we cannot directly measure the ratio Y/Z. From the measured realizations y and z, we wish to estimate Y/Z. Suppose that $\tilde{y}$ and $\tilde{z}$ are samples means over n_s samples. Given the n_{s} samples of y and z, a possible estimator for the ratio Y/Z is a sample ratio $\tilde{y}/\tilde{z}$. The bias of this estimator goes as $O(1/n_{\mathrm{s}})$:

$$\mathbb{E}\left[\frac{Y}{Z}\right] = \frac{\tilde{y}}{\tilde{z}} + O\left(\frac{1}{n_{\mathrm{s}}}\right), \tag{11}$$

which indicates that $\tilde{y}/\tilde{z}$ is an unbiased estimator as n_{s} tends to infinity.

For m parity blocks (see Fig. 6), the MTTDL equation is given by the following lemma.

Lemma 2: For an arbitrary number m of the parity blocks, the MTTDL of (n, k) LDPC codes can be represented by

$$\mathrm{MTTDL} \to \frac{(m+1)\mu^m}{f(n, m, \lambda, \mu, p_0, \cdots, p_{m-1})} \text{ as } \frac{\lambda}{\mu} \to 0, \tag{12}$$

where $f(n, m, \lambda, \mu, p_0, \cdots, p_{m-1})$ is defined by

$$\begin{aligned} &f(n, m, \lambda, \mu, p_0, \cdots, p_{m-1}) \\ &= n\lambda(1-p_0)\cdot\mu^m \\ &\quad + \sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot\mu^{m-j}\right\}\right] \\ &\quad + \left\{\prod_{i=0}^{m}(n-i)\cdot\lambda^{m+1}\right\}\cdot\prod_{i=0}^{m-1}p_i\,. \end{aligned} \tag{13}$$

Proof: See Appendix A. ∎

From Lemma 2 it is seen that with all other parameters fixed, making the p_i values large increases the MTTDL by examining what each term in the MTTDL is doing in the limit. In order to see the behavior in the limit, divide the numerator and denominator of the MTTDL by μ^m and write:

$$\begin{aligned} &\frac{f(n, m, \lambda, \mu, p_0, \cdots, p_{m-1})}{\mu^m} \\ &= \left(\frac{\lambda}{\mu}\right)^0 n\lambda(1-p_0) + \left(\frac{\lambda}{\mu}\right)^1 \lambda(1-p_1)p_0\cdot n(n-1) \\ &\quad + \left(\frac{\lambda}{\mu}\right)^2 \lambda(1-p_2)p_0p_1\cdot n(n-1)(n-2) + \cdots \\ &\quad + \left(\frac{\lambda}{\mu}\right)^{m-1} \lambda(1-p_{m-1})p_0p_1\cdots p_{m-2} \\ &\qquad \cdot n(n-1)\cdots(n-m+1) \\ &\quad + \left(\frac{\lambda}{\mu}\right)^m \lambda p_0p_1\cdots p_{m-1}\cdot n(n-1)\cdots(n-m). \end{aligned} \tag{14}$$

Decreasing (14) increases the MTTDL for a given m. In the right hand side of (14), the value of $\left(\frac{\lambda}{\mu}\right)^i$ in the i^{th} term drops quickly with increasing i for a small value of $\frac{\lambda}{\mu}$. Note that the 0^{th} term disappears as p_0 is forced to 1 in any practical LDPC code. It is easy to see that if p_i in the i^{th} term is set to 1, then this term reduces to zero. Since $\left(\frac{\lambda}{\mu}\right)^i$ is larger for a smaller value of i, forcing as many p_i's for small i as possible to 1 is crucial to minimize (14) or, equivalently, maximize the MTTDL. This property is the key to designing factor graphs that enhance reliability. Since the stopping number is the smallest number of erasures that cannot be corrected, it is clear that increasing the stopping number is equivalent to driving more p_i's to 1. Therefore, a large stopping number of the factor graph would mean an enhanced MTTDL. Theorem 2 makes this relationship between the MTTDL and the stopping number more precise.

Theorem 2: The MTTDL for LDPC codes is a monotonically increasing function of the stopping number s^* of the given factor graph as $\frac{\lambda}{\mu} \to 0$ and assuming $\frac{\lambda}{\mu} < \frac{1}{n}$.

Proof: See Appendix B. ∎

Especially for the VN degree of 2, the stopping number s^* is equal to $g/2$, where g is the girth of the graph [30]. As a result, to increase reliability of the regular LDPC codes with $d_v = 2$, the girth should be increased. This observation motivates LDPC code design by PEG, which is an effective search method for factor graphs with good girth properties.

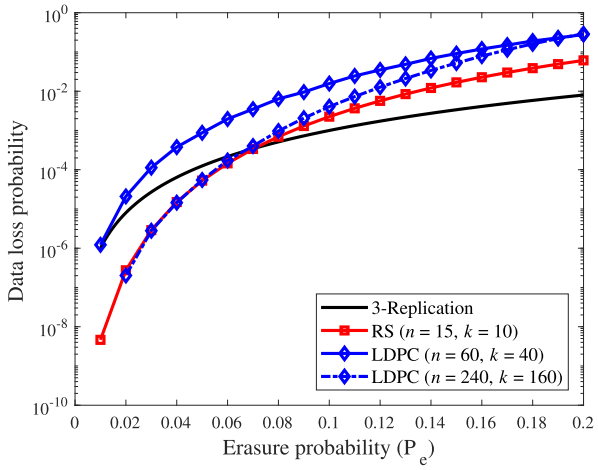

Fig. 7. Data loss probability of the (60, 40) and (240, 160) LDPC codes with $d_v = 2$ compared to the 3-replication and (15, 10) RS codes.

Remark 2: As can be seen in (B.3) derived in the proof of Theorem 2, only the single probability p_{s^*-1} really matters in computing the MTTDL. Empirical results also show that the simplified expression (B.3), which is reproduced below, yields virtually identical MTTDL values as the full expression (12).

$$\text{MTTDL} \rightarrow \frac{(m+1)}{\left(\frac{\lambda}{\mu}\right)^{s^*-1} \lambda(1-p_{s^*-1}) \prod_{i=0}^{s^*-1}(n-i)} \text{ as } \frac{\lambda}{\mu} \rightarrow 0. \tag{15}$$

Since the MTTDL is governed essentially by a single probability p_{s^*-1}, computing the MTTDL of an LDPC code now does not require estimating all p_i probabilities through very extensive error pattern search.

VI. Quantitative Results

From the repair bandwidth analysis in Section III, it is shown that a regular CN degree minimizes the average repair bandwidth of LDPC codes. It is also shown that regular LDPC codes with $d_v = 2$ can minimize repair bandwidth for a given code rate, provided degree-1 VNs are prohibited. In addition, from the MTTDL analysis in Section V, it is verified that LDPC codes should have a large stopping number which helps to improve reliability. With regards to regular LDPC codes with $d_v = 2$, the size of the girth plays the same role as the stopping number. We shall focus on PEG-LDPC codes in this section. PEG is a well-known algorithm which can construct factor graphs having a large girth [31]. However, a concern that may arise for setting $d_v = 2$ is a potentially poor decoding capability in practical scenarios where multiple erasures may occasionally occur within a single codeword, since each VN is protected by only two sets of checks with $d_v = 2$. We plot the data (codeword) loss probability of two $d_v = 2$ regular LDPC codes in Fig. 7 in environments where each symbol erasure occurs independently within each codeword. The results indicate that even a short LDPC code with $d_v = 2$ shows erasure correction behavior similar to 3-replication at low erasure probabilities. Note that decoding capability improves when a larger LDPC code is adopted, showing data loss probability comparable to the (15,10) RS code. In the case of irregular LDPC codes, even though the direct correlation between the girth and the stopping number is unknown, PEG is still a reasonable approach.

TABLE II
Parameters Used for MTTDL Simulation

Parameter	Value	Description
C	40 PB	Total amounts of data
B	256 MB	Block size
N_{disk}	2000	Number of disk nodes
S	20 TB	Storage capacity of a disk
r_{node}	1 Gbps	Network bandwidth on each node
$1/\lambda$	1 year	MTTF (mean-time-to-failure) of a node
μ	$\frac{1}{T_t+T_r}$	Repair rate
T_t	15 min	Detection and triggering time for repair
T_r	$\frac{S \cdot \text{BW}_{\text{cost}}}{r_{\text{node}} \cdot (N_{\text{disk}}-1)}$	Downloading time of blocks
BW_{cost}		Repair BW overhead of the given code
n		Number of total coded blocks in a stripe
k		Number of data blocks in a stripe
m		Number of parity blocks in a stripe

TABLE III
Performance of QC-PEG LDPC Codes With $d_v = 2$, $R = 2/3$

Coding scheme	Storage overhead	Repair BW overhead	MTTDL (days)
3-replication	3x	1x	1.20E+3
(15, 10) RS	1.5x	10x	2.13E+10
(10, 6, 5) Xorbas LRC	1.6x	5x	7.38E+7
(15, 10, 6) Binary LRC	1.5x	6x	3.00E+4
(60, 40) LDPC	1.5x	5x	1.40E+7
(150, 100) LDPC	1.5x	5x	1.42E+8
(210, 140) LDPC	1.5x	5x	2.91E+11

Having ensured a good decoding capability, the metrics considered for comparison are storage overhead (code rate inverse), repair bandwidth and MTTDL. For the MTTDL simulation, the following normalized equation is used for fair comparison among codes having different lengths:

$$\text{MTTDL} = \frac{\text{MTTDL}_{\text{stripe}}}{C/nB},$$

where $\text{MTTDL}_{\text{stripe}}$ is the MTTDL given in Section V for a stripe. Here, the MTTDL for a stripe is normalized by the number of stripes, C/nB, in storage system. The parameters used for MTTDL simulation are given in Table II. These values are chosen consistent with the existing literature [8], [13]. Note that for the repair rate, both the triggering time and the downloading time are included; the downloading time depends on the repair bandwidth (BW) overhead of the coding scheme.

For LDPC code simulations, using specific QC-PEG parity-check matrices, p_i's are first obtained from decoding simulation and the MTTDL values are calculated from (12) or (15).[3] Table III shows performance of the QC-PEG LDPC codes with $d_v = 2$ for $R = 2/3$. Here, the (15, 10) RS code is chosen for comparison as well as simple replication and existing LRC methods.

[3]Note that the MTTDL value shown here for 3-replication is different from that in [8], [13] due to the fact that the definition of the repair rate is different (in [8], $\mu = 1/T_r$ for repair from a single failure and $\mu = 1/T_t$ from multiple failures, and in [13], $\mu = r_{\text{node}}/B$).

For a given storage overhead, LDPC codes in Table III have a 5× repair bandwidth overhead, relative to replication, whereas the RS code has a 10× overhead. Thus, compared to the RS code, these LDPC codes require only one half of the repair bandwidth given the same storage overhead. Moreover, LDPC codes maintain the same repair bandwidth even as the code length is increased. This indicates that LDPC codes can get better MTTDLs than the (15, 10) RS code and the (10, 6, 5) Xorbas LRC [13] when longer codes are used. The table shows specifically that the (150, 100) and (210, 140) LDPC code has better performance in terms of both repair bandwidth and MTTDL compared to the (15, 10) RS code. Relative to the (10, 6, 5) Xorbas LRC, we observed that the (150, 100) and (210, 140) LDPC codes provide higher MTTDL. This is at the expense of a longer code length. In general, it is expected that the price of increasing the code length will be complexity. However, the complexity of encoding/decoding of LDPC codes in erasure channels is quite reasonable for the code lengths discussed here. The computational complexity issue of the LDPC code is discussed below.

Remark 3 (Computational Complexity): The computational complexity that need be considered in the context of distributed storage includes the encoding and decoding complexity. Note that LDPC encoding/decoding is based on simple XOR operations, while RS code and LRC require expensive Galois field operations. The encoding complexity of RS codes and LRCs both increases quadratically with respect to n; on the other hand, encoding of the LDPC code requires a linear (or near-linear) complexity. Decoding complexity is directly related to the computational burden required for reading data or repairing the failed block, which are the most frequent events in operating distributed storage. From this point of view, decoding complexity is also referred to as repairing complexity in distributed storage. The decoding/repairing traffic per one node of the LDPC code depends on the check node degree. Since d_c is independent of n as presented in Section III, overall decoding complexity of the LDPC code is only linear with n, whereas decoding the LRC and RS code requires complexity quadratic in n. Specifically, the required numbers of additions and multiplications on average to decode/repair an LDPC code of rate $2/3$ that we employed are four and zero, respectively, regardless of the code length. For decoding of the $(14, 10)$ RS code, nine additions and ten multiplications are required, which can increase tremendously with increasing code length. The binary LRC [27], [32] is a modification of the Xorbas LRC to reduce computational complexity at the expense of repair bandwidth and MTTDL. For example, considering the failure of single nodes, decoding/repairing of a $(k, n-k, r_2) = (10, 5, 6)$ binary LRC (see Table III for its repair bandwidth overhead and MTTDL) which is constructed based on a (10, 6, 5) Xorbas LRC requires five additions and zero multiplications. For the corresponding Xorbas LRC, four additions and 4.75 multiplications in binary extension field are needed on average. We thus observed that the LDPC code is competitive in terms of the decoding complexity as well thanks to its low-repair-bandwidth and the XOR-only feature. Note also that the difference in decoding complexity will increase further as the code length becomes longer.

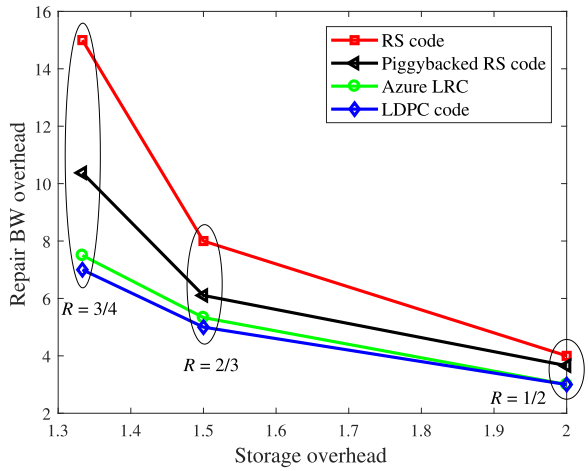

Fig. 8. Tradeoffs between repair bandwidth overhead and storage overhead for different codes. Coding schemes having higher reliability than the (14, 10) RS code are considered.

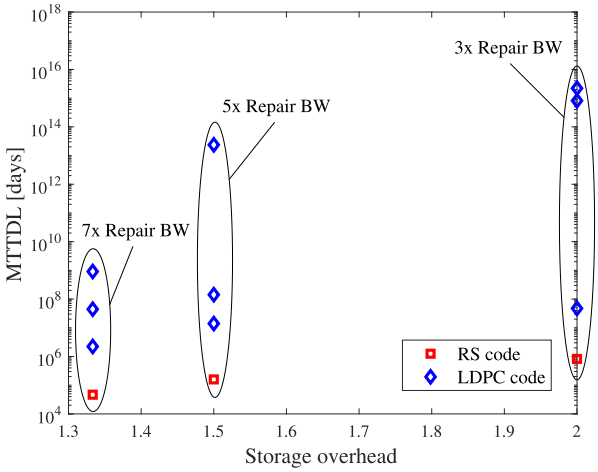

Fig. 9. MTTDL comparison of regular LDPC and RS codes under different storage overhead and repair bandwidth constraints.

For rates 3/4, 2/3 and 1/2, various coding schemes are compared in Fig. 8. The code parameters used in Fig. 8 are given in Table IV. Here we only consider codes that have higher MTTDLs than the (14, 10) RS code used in the Facebook cluster. The MTTDL of the (14, 10) RS codes is 1.61E+7. Note that our comparison with all other codes are done by averaging systematic and parity nodes. For the three storage overhead factors (code rate inverses), it is shown that LDPC codes have consistently better repair-bandwidth/storage-space tradeoffs compared to other codes. As the storage overhead is forced to decrease, LDPC codes enjoy a bigger performance gap relative to other codes with the exception of the LRC codes that perform similar to the LDPC codes.

For given storage and repair bandwidth overheads, LDPC codes can achieve better MTTDL by increasing the code length, compared to the LRC and other codes. Fig. 9 shows such MTTDL comparison between the RS and LDPC codes, where for a given storage overhead, the MTTDL advantage of the LDPC codes is evident. Since the MTTDL of the LRC is known to be similar to that of the RS codes [8], LDPC codes will have definite reliability advantages over the LRCs.

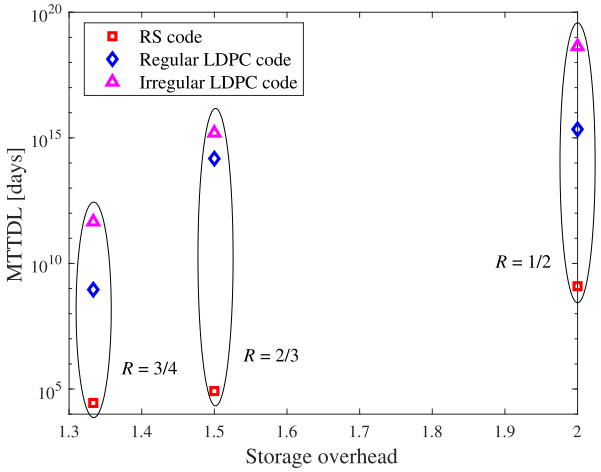

Fig. 10. MTTDL comparison of irregular LDPC, regular LDPC and RS codes under different storage overhead and repair bandwidth constraints.

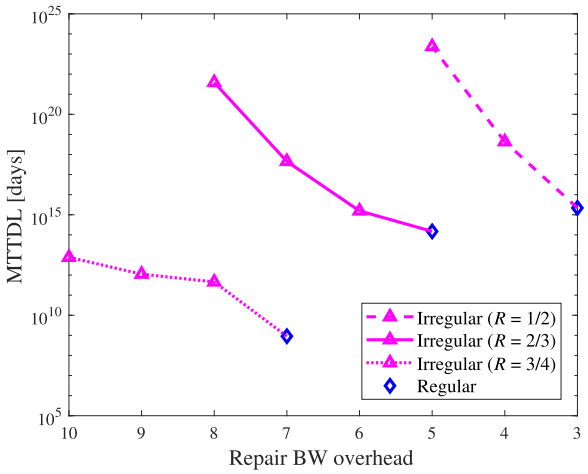

Fig. 11. Tradeoffs between MTTDL and repair bandwidth overhead of LDPC codes under different code rates.

TABLE IV

PARAMETERS OF CODES USED IN FIG. 8

Scheme	R = 3/4	R = 2/3	R = 1/2
RS	(20, 15)	(12, 8)	(8, 4)
Piggybacked RS	(20, 15)	(12, 8)	(8, 4)
Azure LRC	(18, 3, 3)	(12, 3, 3)	(6, 3, 3)
LDPC	(240, 180)	(120, 80)	(56, 28)

TABLE V

PARAMETERS OF CODES USED IN FIG. 9. LDPC1 REPRESENTS THE LDPC CODES WITH THE LOWEST MTTDLS

Scheme	R = 3/4	R = 2/3	R = 1/2
RS	(10, 7)	(8, 5)	(6, 3)
LDPC1	(80, 60)	(60, 40)	(44, 22)
LDPC2	(200, 150)	(150, 100)	(72, 36)
LDPC3	(320, 240)	(240, 160)	(100, 50)

TABLE VI

PARAMETERS OF CODES USED IN FIGS. 10 AND 11. BOTH REGULAR AND IRREGULAR LDPC CODES ARE BASED ON THE SAME CODE PARAMETERS

Scheme	R = 3/4	R = 2/3	R = 1/2
RS	(11, 8)	(9, 6)	(8, 4)
LDPC	(320, 240)	(240, 160)	(100, 50)

MTTDLs of irregular LDPC codes that are designed to enhance the system reliability are shown in Figs. 10 and 11. Irregular LDPC codes are designed by the VN degree distributions given in Table I. The code-lengths are set to be identical to those of LDPC3, and it is guaranteed that the global girth size is strictly larger than 4.

Fig. 10 shows the MTTDLs of the designed irregular LDPC codes with repair bandwidths increased by one relative to the regular LDPC codes also included in the figure. The MTTDLs of RS and regular LDPC codes are also shown for comparison. The parameters of the RS and LDPC codes are in Table VI. The LDPC codes have the same code parameters as LDPC3 in Table V, and the parameters of the RS codes are set to have the same repair bandwidth as the irregular LDPC codes being compared. As can be seen, the designed irregular LDPC codes outperform RS and regular LDPC codes in terms of the MTTDL, at the cost of increased repair bandwidth (by 1).

Fig. 11 represents the behavior of MTTDLs versus repair bandwidth for LDPC codes. The code parameters are the same as those used in Fig. 10. As mentioned above, regular LDPC codes with $d_v = 2$ have the minimum repair bandwidth for given code parameters. We see that MTTDLs improve substantially when the repair bandwidths are allowed to grow from the minimum value. The tradeoff effect is more dramatic for smaller code rates.

VII. CONCLUDING REMARKS

A. Conclusion

For distributed storage applications, this paper shows that LDPC codes could be a highly viable option in terms of storage overhead, repair bandwidth and reliability tradeoffs. Unlike the RS code, the repair bandwidth of the LDPC codes does not increase with the code length. As a result, the LDPC codes can be designed to enjoy both low repair bandwidth and high reliability compared to the RS code and its known variants. It has been specifically shown that for a given number of edges in the factor graph, CN-regular LDPC codes minimize the repair bandwidth. In addition to the requirement of the CN-regularity, VN-regularity with $d_v = 2$ minimizes the repair bandwidth for a given code rate, barring all VNs with degree 1. A code design that takes advantage of the improved reliability of LDPC codes has been given, yielding useful tradeoff options between MTTDL and repair bandwidth. The MTTDL analysis for LDPC codes has also been provided that relates the code's stopping set size with its MTTDL.

B. Future Work

Interesting future work includes LDPC code design aiming at reduction of both repair-bandwidth and latency. For the reliability analysis in this paper, we assumed that there occurred only one node failure at a time since it was the most frequent failure pattern. Considering multiple erasures it can be shown that the repair bandwidth of LDPC codes is still one

less than the CN degree. However, the number of decoding iterations required to repair multiple erasures may differ from one specific code design to next. Since the decoding latency of LDPC codes is proportional to the number of decoding iterations [33], we need to design LDPC code degree distributions to minimize the number of decoding iterations. It would be meaningful to study LDPC code structures that maximize the number of single-step recoverable nodes in combating latency.

The update complexity (defined as the maximum number of coded symbols updated for one changed symbol in the message [34]) is an important measure especially in applications to highly dynamic distributed storage in which data updates are frequent. The study on the existence and construction of update-efficient codes is an active area of research (e.g., see [34]–[37]). Investigating the relationships between update complexity and other performance metrics considered in this paper such as the MTTDL would be a good direction as well.

APPENDIX A
PROOF OF LEMMA 2

Proof: As can be seen in Fig. 6, the Markov model of LDPC codes with m parity blocks consists of $m+2$ states regarding the state $m+2$ as a DL state. Let $\pi_i(t)$ denote the state probability of the state i at time t and I the set of all states. Then we have a constraint $\sum_{i\in I}\pi_i(t)=1$, since the process must be in one of the states at any given time $t\geq 0$. For an arbitrary number m of parity blocks, we build the sets of equations describing the Markov model which are followed by the MTTDL equation of LDPC codes with m parity blocks.

Assume that state 0 is the initial state of the Markov chain, so that

$$\pi_i(0)=\begin{cases}1 & \text{if } i=0\\ 0 & \text{otherwise.}\end{cases}$$

First we construct a set of differential equations from the Markov model in Fig. 6.

- $i=0$

$$\frac{\mathrm{d}\pi_i(t)}{\mathrm{d}t}=-n\lambda\pi_i(t)+\mu\pi_{i+1}(t)\,, \tag{A.1}$$

- $1\leq i<m$

$$\frac{\mathrm{d}\pi_i(t)}{\mathrm{d}t}=-(n-i)\lambda\pi_i(t)-\mu\pi_i(t) +\mu\pi_{i+1}(t)+(n-i+1)\lambda p_{i-1}\pi_{i-1}(t), \tag{A.2}$$

- $i=m$

$$\frac{\mathrm{d}\pi_i(t)}{\mathrm{d}t}=-(n-i)\lambda\pi_i(t)-\mu\pi_i(t) +(n-i+1)\lambda p_{i-1}\pi_{i-1}(t), \tag{A.3}$$

- $i=m+1$ (Data loss state)

$$\frac{\mathrm{d}\pi_i(t)}{\mathrm{d}t}=(n-i+1)\lambda\pi_{i-1}(t) +(n-i+2)\lambda(1-p_{i-2})\pi_{i-2}(t)+\cdots +(n-1)\lambda(1-p_1)\pi_1(t)+n\lambda(1-p_0)\pi_0(t). \tag{A.4}$$

Taking the Laplace transforms of eqs.(A.1) to (A.4) yields the following string of equations, where $\bar{\pi}_i(s)$ denotes the Laplace transform of $\pi_i(t)$.

- $i=0$

$$\bar{\pi}_i(s)=-n\lambda\bar{\pi}_i(s)+\mu\bar{\pi}_{i+1}(s)+1\,, \tag{A.5}$$

- $1\leq i<m$

$$s\bar{\pi}_i(s)=-(n-i)\lambda\bar{\pi}_i(s)-\mu\bar{\pi}_i(s) +\mu\bar{\pi}_{i+1}(s)+(n-i+1)\lambda p_{i-1}\bar{\pi}_{i-1}(s), \tag{A.6}$$

- $i=m$

$$s\bar{\pi}_i(s)=-(n-i)\lambda\bar{\pi}_i(s)-\mu\bar{\pi}_i(s) +(n-i+1)\lambda p_{i-1}\bar{\pi}_{i-1}(s), \tag{A.7}$$

- $i=m+1$ (Data loss state)

$$s\bar{\pi}_i(s)=(n-i+1)\lambda\bar{\pi}_{i-1}(s) +(n-i+2)\lambda(1-p_{i-2})\bar{\pi}_{i-2}(s)+\cdots +(n-1)\lambda(1-p_1)\bar{\pi}_1(s)+n\lambda(1-p_0)\bar{\pi}_0(s). \tag{A.8}$$

Solving eqs. (A.5) to (A.8) for $s\bar{\pi}_{m+1}(s)$, $s\bar{\pi}_{m+1}(s)$ is presented as follows:

$$\begin{aligned}&s\bar{\pi}_{m+1}(s)\\ &=\frac{n\lambda(1-p_0)\cdot G_1(s)}{G_0(s)}\\ &\quad+\frac{\sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot G_{j+1}(s)\right\}\right]}{G_0(s)}\\ &\quad+\frac{\left\{\prod_{i=0}^{m}(n-i)\cdot\lambda^{m+1}\right\}\cdot\prod_{i=0}^{m-1}p_i}{G_0(s)},\end{aligned} \tag{A.9}$$

where $G_i(s)$ for $0\leq i\leq m$ is recursively defined by

$$\begin{aligned}G_m(s)&=s+(n-m)\lambda+\mu,\\ G_{m-1}(s)&=\{s+(n-m+1)\lambda+\mu\}G_m(s)\\ &\quad-\mu(n-m+1)\lambda p_{m-1},\\ G_{1\leq i<m-1}(s)&=\{s+(n-i)\lambda+\mu\}G_{i+1}(s)\\ &\quad-\mu(n-i)\lambda p_i G_{i+2}(s),\\ G_0(s)&=(s+n\lambda)G_1(s)-\mu n\lambda p_0 G_2(s).\end{aligned} \tag{A.10}$$

Equation (A.10) is determined by the following set of equations:

$$\begin{aligned}\bar{\pi}_m(s)&=\frac{(n-m+1)\lambda p_{m-1}}{s+(n-m)\lambda+\mu}\bar{\pi}_{m-1}(s)\\ &:=\frac{(n-m+1)\lambda p_{m-1}}{G_m(s)}\bar{\pi}_{m-1}(s)\,,\\ \bar{\pi}_{m-1}(s)&=\frac{(n-m+2)\lambda p_{m-2}}{s+(n-m+1)\lambda+\mu-\mu\cdot\frac{(n-m+1)\lambda p_{m-1}}{G_m(s)}}\bar{\pi}_{m-2}(s)\\ &:=\frac{(n-m+2)\lambda p_{m-2}\cdot G_m(s)}{G_{m-1}(s)}\bar{\pi}_{m-2}(s)\,,\end{aligned}$$

$$\begin{aligned}
\bar{\pi}_{m-2}(s) &= \frac{(n-m+3)\lambda p_{m-3}}{s+(n-m+2)\lambda+\mu-\mu\cdot\frac{(n-m+2)\lambda p_{m-2}G_m(s)}{G_{m-1}(s)}}\bar{\pi}_{m-3}(s) \\
&:= \frac{(n-m+3)\lambda p_{m-3}\cdot G_{m-1}(s)}{G_{m-2}(s)}\bar{\pi}_{m-3}(s)\,, \\
&\vdots \\
\bar{\pi}_1(s) &= \frac{n\lambda p_0}{s+(n-1)\lambda+\mu-\mu\cdot\frac{(n-1)\lambda p_1 G_3(s)}{G_2(s)}}\bar{\pi}_0(s) \\
&:= \frac{n\lambda p_0\cdot G_2(s)}{G_1(s)}\bar{\pi}_0(s)\,, \\
\bar{\pi}_0(s) &= \frac{1}{s+n\lambda-\mu\cdot\frac{n\lambda p_0 G_2(s)}{G_1(s)}} \\
&:= \frac{G_1(s)}{G_0(s)}\,.
\end{aligned}$$

From the moment generating property of Laplace transforms, the MTTDL is given by

$$\text{MTTDL} = -\frac{\mathrm{d}}{\mathrm{d}s}\Big(s\bar{\pi}_{m+1}(s)\Big)\Big|_{s=0}. \tag{A.11}$$

Combining (A.9) and (A.11), we arrive at the final result as shown in (A.12) through (A.14) at the bottom of this page. This completes the proof. Equation (A.12) follows from the final-value theorem associated with the Laplace transform: as $s\to 0$, (A.9) implies (A.15) given the final-value theorem as shown in (A.16). Equation (A.13) is due to (A.15) as well as the fact that $G_1(0)\geq\mu^m$, $G_2(0)\geq\mu^{m-1}$, $\cdots$, $G_{m-1}(0)\geq\mu^2$, and $G_m(0)\geq\mu$ in (A.15). Equation (A.14) is because the numerator of (A.13) approaches $(m+1)\mu^m$ as $\frac{\lambda}{\mu}\to 0$.

$$\begin{aligned}
&G_0(s) \\
&= n\lambda(1-p_0)\cdot G_1(s) \\
&\quad+\sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot G_{j+1}(s)\right\}\right] \\
&\quad+\left\{\prod_{i=0}^{m}(n-i)\cdot\lambda^{m+1}\right\}\cdot\prod_{i=0}^{m-1}p_i, \qquad\text{(A.15)}\\
&\lim_{t\to\infty}\pi_{m+1}(t)=\lim_{s\to 0}s\bar{\pi}_{m+1}(s)=1\,. \qquad\text{(A.16)}
\end{aligned}$$

■

Appendix B
Proof of Theorem 2

Proof: Consider the right hand side of (14). Recall that the stopping number s^* is the smallest number of erasures that cannot be corrected and that p_i is the conditional probability that an additional erasure will be tolerated given i erasures. This leads to the property that $p_i=1$ for any $i<s^*-1$, and the first probability that is not equal to 1 as i increases from 0 is p_{s^*-1}. Then, the first non-zero term in the right hand side of (14) is

$$\left(\frac{\lambda}{\mu}\right)^{s^*-1}\lambda(1-p_{s^*-1})\cdot n(n-1)\cdots(n-s^*+1). \tag{B.1}$$

We now show that this term dominates as $\frac{\lambda}{\mu}\to 0$. In fact, the ratio of the next term and this term is given by

$$\frac{\lambda}{\mu}\cdot\frac{(1-p_{s^*})p_{s^*-1}(n-s^*)}{1-p_{s^*-1}}, \tag{B.2}$$

$$\begin{aligned}
\text{MTTDL} &= \frac{G_0(0)\cdot G_0'(0)}{G_0^2(0)}-\frac{n\lambda(1-p_0)\cdot G_1'(0)\cdot G_0(0)}{G_0^2(0)} \\
&\quad-\frac{\sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot G_{j+1}'(0)\right\}\right]}{G_0(0)} \qquad\text{(A.12)}\\
&\leq \left\{G_0'(0)-n\lambda(1-p_0)\cdot G_1'(0)-\sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot G_{j+1}'(0)\right\}\right]\right\} \\
&\quad\Big/\left\{n\lambda(1-p_0)\cdot\mu^m+\sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot\mu^{m-j}\right\}\right]\right. \\
&\qquad\left.+\left\{\prod_{i=0}^{m}(n-i)\cdot\lambda^{m+1}\right\}\cdot\prod_{i=0}^{m-1}p_i\right\} \qquad\text{(A.13)}\\
&\to (m+1)\mu^m\Big/\left\{n\lambda(1-p_0)\cdot\mu^m+\sum_{j=1}^{m-1}\left[\left\{\prod_{i=0}^{j}(n-i)\cdot\lambda^{j+1}\right\}\cdot\left\{\prod_{i=0}^{j-1}p_i\cdot(1-p_j)\cdot\mu^{m-j}\right\}\right]\right. \\
&\qquad\left.+\left\{\prod_{i=0}^{m}(n-i)\cdot\lambda^{m+1}\right\}\cdot\prod_{i=0}^{m-1}p_i\right\}\ \text{as}\ \frac{\lambda}{\mu}\to 0\,. \qquad\text{(A.14)}
\end{aligned}$$

which approaches zero for any finite n as $\frac{\lambda}{\mu} \to 0$. Using the same argument the similar ratio of any two successive terms reduces to zero in the limit. This means that the MTTDL of an LDPC code simplifies to

$$\frac{(m+1)}{\left(\frac{\lambda}{\mu}\right)^{s^*-1} \lambda(1-p_{s^*-1}) \prod_{i=0}^{s^*-1}(n-i)} \tag{B.3}$$

for $\frac{\lambda}{\mu} \to 0$. Now, for any reasonably large n, we have $\prod_{i=0}^{s^*-1}(n-i) \approx n^{s^*}$. The MTTDL in the limit of $\frac{\lambda}{\mu} \to 0$ can now be rewritten as

$$\frac{(m+1)}{\left(\frac{\lambda}{\mu} \cdot n\right)^{s^*} \mu(1-p_{s^*-1})}, \tag{B.4}$$

which is a monotonically and very rapidly increasing function of s^*, as long as $\frac{\lambda}{\mu} < \frac{1}{n}$. This completes the proof. ■

REFERENCES

[1] D. Lee, H. Park, and J. Moon, "Reducing repair-bandwidth using codes based on factor graphs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–10.

[5] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. Int. Workshop Peer-Peer Syst.* 2002, pp. 328–337.

[6] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.

[7] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. USENIX Workshop Hot Topics Storage File Syst. (HotStorage)* 2013, pp. 1–5.

[8] C. Huang *et al.*, "Erasure coding in windows azure storage," in *Proc. USENIX Annu. Tech. Conf.*, Boston, MA, USA, 2012, pp. 2–12.

[9] D. Ford *et al.*, "Availability in globally distributed storage systems," in *Proc. USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2010, pp. 1–14.

[10] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[11] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[12] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[13] M. Sathiamoorthy *et al.*, "XORing elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 325–336, 2013.

[14] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[15] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[16] J. S. Plank and M. G. Thomason, "A practical analysis of low-density parity-check erasure codes for wide-area storage applications," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2004, pp. 115–124.

[17] J. S. Plank, A. L. Buchsbaum, R. L. Collins, and M. G. Thomason, "Small parity-check erasure codes—Exploration and observations," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2005, pp. 326–335.

[18] Y. Wei, Y. W. Foo, K. C. Lim, and F. Chen, "The auto-configurable LDPC codes for distributed storage," in *Proc. IEEE Int. Conf. Comput. Sci. Eng.*, Dec. 2014, pp. 1332–1338.

[19] Y. Wei, F. Chen, and K. C. Lim, "Large LDPC codes for big data storage," in *Proc. ASE BigData SocialInformatics*, 2015, pp. 1–6.

[20] Y. Wei and F. Chen, "expanCodes: Tailored LDPC codes for big data storage," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2016, pp. 620–625.

[21] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. ACM Symp. Theory Comput.*, May 1997, pp. 150–159.

[22] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[23] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[24] T. V. Nguyen, A. Nosratinia, and D. Divsalar, "The design of rate-compatible protograph LDPC codes," *IEEE Trans. Commun.*, vol. 60, no. 10, pp. 2841–2850, Oct. 2012.

[25] J. Garcia-Frias and W. Zhong, "Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix," *IEEE Commun. Lett.*, vol. 7, no. 6, pp. 266–268, Jun. 2003.

[26] S. Ramabhadran and J. Pasquale, "Analysis of long-running replicated systems," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2006, pp. 1–9.

[27] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "A class of binary locally repairable codes," *IEEE Trans. Commun.*, vol. 64, no. 8, pp. 3182–3193, Aug. 2016.

[28] K. S. Trivedi, *Probability and Statistics With Reliability, Queuing, and Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2008.

[29] J. L. Hafner and K. Rao, "Notes on reliability models for non-MDS erasure codes," IBM Res., San Jose, CA, USA, Tech. Rep. RJ10391, 2006.

[30] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang, "Stopping sets and the girth of tanner graphs," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2002, p. 2.

[31] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth tanner graphs," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 2. Nov. 2001, pp. 995–1001.

[32] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "An efficient binary locally repairable code for hadoop distributed file system," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1287–1290, Aug. 2014.

[33] B. Smith, M. Ardakani, W. Yu, and F. R. Kschischang, "Design of irregular LDPC codes with optimized performance-complexity tradeoff," *IEEE Trans. Commun.*, vol. 58, no. 2, pp. 489–499, Feb. 2010.

[34] N. P. Anthapadmanabhan, E. Soljanin, and S. Vishwanath, "Update-efficient codes for erasure correction," in *Proc. Annu. Allerton Conf. Commun. Control, Comput.*, Sep. 2010, pp. 376–382.

[35] A. Mazumdar, V. Chandar, and G. W. Wornell, "Update-efficiency and local repairability limits for capacity approaching codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 976–988, May 2014.

[36] A. Jule and I. Andriyanova, "Some results on update complexity of a linear code ensemble," in *Proc. Int. Symp. Netw. Coding (NetCod)*, Jul. 2011, pp. 1–5.

[37] K. Kralevska, D. Gligoroski, and H. Øverby, "Balanced locally repairable codes," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Sep. 2016, pp. 280–284.

Hyegyeong Park (S'12) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2012 and 2014, respectively, where she is currently pursuing the Ph.D. degree. Her research interests include coding and information theory with the current focus on distributed storage and computing.

Dongwon Lee (S'14) received the B.S. degree in electrical engineering from Yonsei University, Seoul, South Korea, and the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2013 and 2015, respectively.

Jaekyun Moon (F'05) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor of electrical engineering with KAIST. From 1990 to 2009, he was with the faculty of the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities. He consulted as Chief Scientist for DSPG, Inc. from 2004 to 2007. He was also a Chief Technology Officer with Link-A-Media Devices Corporation. His research interests are in the area of channel characterization, signal processing and coding for data storage and digital communication. He received the McKnight Land-Grant Professorship from the University of Minnesota. He received the IBM Faculty Development Awards and the IBM Partnership Awards. He was awarded the National Storage Industry Consortium Technical Achievement Award for the invention of the maximum transition run code, a widely used error-control/modulation code in commercial storage systems. He served as Program Chair for the 1997 IEEE Magnetic Recording Conference. He is also Past Chair of the Signal Processing for Storage Technical Committee of the IEEE Communications Society. He served as a Guest Editor for the 2001 IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Special Issue on Signal Processing for High Density Recording. He also served as an Editor for IEEE TRANSACTIONS ON MAGNETICS in the area of signal processing and coding from 2001 to 2006.