# P2P Distributed Data Replenishment

Kien Nguyen, Thinh Nguyen, Viet Le
School of EECS, Oregon State University
Email: {nguyenki,thinhq,lev}@eecs.oregonstate.edu)

Yevgeniy Kovchegov
Department of Mathematics, Oregon State University
Email: kovchegy@math.oregonstate.edu

***Abstract*—We investigate a class of randomized peer-to-peer (P2P) approach to Internet-wide distributed data storage systems that promises to reduce the coordination complexity and increases performance scalability. The core of these randomized P2P data storage systems is the data replenishment mechanism. The data replenishment automates the process of maintaining a sufficient level of data redundancy to ensure the availability of data in presence of peer departures and failures. The dynamics of peers entering and leaving the network is modeled as a stochastic process. A novel analytical time-backward technique is proposed to bound the expected time for a piece of data to remain in P2P systems. Both theoretical and simulation results are in agreement, indicating that a proposed data replenishment via random linear network coding (RLNC) outperforms other strategies that employ popular repetition and channel coding techniques. Specifically, we show that the expected time for a piece of data to remain in a P2P system, the longer the better, is exponential in the redundancy amount for the RLNC-based strategy, while they are quadratic for other strategies.**

## I. Introduction

Recent development of Peer-to-Peer (P2P) networks opens a new possibility for building large scale distributed systems over the Internet. Typically in such systems, data are replicated across multiple nodes (peers) at different network locations such that network failures in some parts of the Internet will not prevent a user from accessing the data stored in other parts of the Internet. To that end, many recent research efforts have been focused on using P2P platforms to build reliable, large scale distributed systems for Internet services [1], [2]. In this paper, we investigate the theoretical underpinnings and examine the simulated performance for a class of large scale distributed systems based on a randomized P2P approach via coding techniques.

In a nutshell, a distributed system over the Internet is an overlay network of storage and computing nodes, linked together in such a way to allow computational, storage, and bandwidth resources to be shared. Popular P2P network such as BitTorrent is a distributed systems that enable their users to share data and bandwidth. Since the overlay nodes are located geographically apart, each node has a different network access, and data are replicated across multiple nodes, these systems are less susceptible to the bottleneck failures. However, if not properly designed, they will incur substantial communication/coordination overheads among nodes. For a distributed storage system, one of the main challenges is to design efficient coordination mechanisms among nodes in order to maintain the correct data in the system while minimizing the communication overheads.

**Data replenishment.** Distributed storage system research have been focused on indexing, maintaining, and retrieving data correctly and efficiently. In this paper, we will not discuss these aspects as they have been well investigated in [3], [1]. Rather, we will focus on scalable methods for maintaining data in a highly volatile environment such as P2P networks. Specifically, in a P2P network, the data is stored on a peer's hard drive. Consequently, when a peer departs the network, so does the data it carries. Therefore, it is preferable to employ some form of data replenishment mechanism which ensures that at any time, the requested data is available at one or multiple peers collectively. Furthermore, the data replenishment mechanism should be simple for it to be effective in highly dynamic and distributed environments. Data replenishment mechanism is the focus of this paper.

**Approach Overview.** Traditionally in a distributed storage system, a file is replicated in its entirety at one or multiple locations. However, for the same overall storage redundancy, a more robust approach is to break up a single file into many pieces, code these pieces properly, then disperse them to multiple nodes in a network [4], [5], [6], [7]. A user recovers the file by downloading its many pieces simultaneously from different locations. In this setup, a file to be stored, is first broken up into multiple pieces or packets, coded using either Reed-Solomon, repetition, or random linear network codes (RLNC), then dispersed to a number of peers in the network. Now, any peer can depart the network along with its data. If a new peer joins, it can be recruited to help replenish the missing data. There are many ways to replenish the missing data. One way is better than the others. We will show that the data replenishment via RLNC is much more efficient than the strategies using repetition and traditional channel code such as Reed-Solomon code. We note that the concept of data replenishment is similar to data repair as termed in [7]. On the other hand, in [6], [7], the authors examined the fundamental tradeoff between the replenishment bandwidth and storage capacity in a static setting, while our work considers the dynamics of data replenishment of different techniques and their effects on data recoverability over time.

## II. Related Work

In [8], Dimakis et al. provided a survey on recent network coding techniques for distributed storage. Majority of the works in this area have been focused on (1) the fundamental trade-off between the *repair* bandwidth and the storage capacity of nodes and (2) techniques for constructing capacityachieving network codes. In a distributed storage system, if a node fails, a new node is recruited and attempts to reconstruct the missing data from the failed node by downloading the data from other nodes. The codes for this type of setting is called regenerating codes, and the amount of downloaded data required for reconstructing the missing data is called the repair bandwidth. It has been shown that the repair bandwidth using network coding can be substantially smaller than that of using traditional MDS codes provided that the storage capacities of nodes in the systems are sufficiently large [7].

Li et al. also considered joint design of regenerating codes and network topology for efficiently utilizing network links and reducing repair bandwidth [9]. Their design called RC-TREE, is able to produce efficient code regeneration, even in dynamic environments where nodes enter and depart the network frequently. In [10], Duminuco and Biersack studied computational, communication, and storage costs of a real implementation of random linear regenerating codes in peer-to-peer systems. They concluded that with a small increase in storage cost and computation, a significant reduction of the communication cost can be achieved.

## III. Network Model and Data Replenishment Strategies

When a peer leaves the network, so does its data. This effectively reduces the robustness of the system temporarily or permanently if the peer never rejoins or rejoins without its data. Theoretically, the collective data in the network will remain the same if one is to replace the exact missing data due to a departed peer with a new peer. However, this approach requires the system to know which peer leaves the network and which pieces of data that it has. Then, a precise coordination and communication mechanism is needed to reproduce the equivalent state of the network. This could potentially create significant communication and coordination overheads. In this paper we study a more scalable, randomized approach that aims to approximately reproduce the state of the network prior to a peer's departures, i.e. *data replenishment.*

We model the network dynamics as follows. For every peer that leaves the network, the system can find another peer to take over the responsibility of the departed peer. This model approximates the dynamics of a network with constant number of peers since the departures and arrivals are synchronized. An analysis of a more general model with Poisson arrival and departures is included in the extended version of this paper submitted to a journal. We will describe three replenishment strategies in this paper. Each strategy has to follow the basic rules which model the limited communication and storage capacities of the peers. We abstract the replenishment process as the following game:

The game involves $N$ peers. The objective of the game is for the $N$ peers to collectively maintain some given data for as long as possible, subject to the following rules:

1) Each peer is allowed to carry a maximum of $T$ bits.
2) At every time step, a peer is selected uniformly at random to leave the game. Thus the $T$ bits that it carries will also be deleted.
3) A new peer is recruited to replace the departed peer. It is allowed to communicate with a maximum of $M$ peers in an attempt to replenish the data.
4) Peers can modify the data in any way, as long as they do not exceed their storage capacity of $T$ bits.

*Given these rules, what is the optimal strategy for the system to maintain a piece of data for as long as possible?* We consider three replenishment strategies below:

**Repetition Code Based Strategy.** Suppose a file to be stored is $C$ bits long, and there are $N$ peers, each can store up to $C/2$ bits. The repetition strategy divides the peers into two groups. Peers in one group are assigned to store the first half of the file, while peers in the other group store the remaining half. Note that the redundancy ratio is the total storage divided by the file size. In this particular case, the redundancy is $\frac{NC/2}{C} = N/2$. Whenever a peer departs, a new peer joins, and communicates with $M = 2$ other peers selected uniformly at random. Since the new peer's capacity is only $C/2$ bits, even it contacts two peers, it will only copy the data from one of these peers, or effectively, $M = 1$. The game is played repeatedly until all the peers have the same piece of data which is either the first half or the second half of the file.

**Reed-Solomon Code Based Strategy.** Intuitively, a better strategy is to employ the standard channel coding techniques such as the Reed-Solomon code. Using this strategy, a file of $C$ bits is first divided into three equal parts, which are then channel coded to produce $N$ codewords of length $C/3$ bits. Each peer then keeps a codeword. The redundancy in this case is $N/3$. The property of $RS(N, 3)$ code ensures that a file can be recovered using any of three distinct codewords. Now, the game is played in exactly the same way as before. When a peer departs, the new peer joins, and is allowed to communicate with $M = 2$ peers in an attempt to replenish the missing data.

**Random Linear Network Code Based Strategy.** A yet intuitively better strategy is to employ Random Linear Network Coding technique [11][12]. Using this strategy, a file of $C$ bits is first divided into three equal parts. $N$ codewords are produced, each is a random linear combination of the three original parts of the file. Each peer then keeps a codeword. The redundancy is $N/3$, identical to that of RS code strategy. Mathematically, an $n-bit$ pattern can be viewed as an element from a finite field. Thus, a codeword can be viewed as a vector of elements from a finite field. A codeword $A$ is a random linear combination of codewords $B$ and $C$, then

$$A = c_1 B + c_2 C, \tag{1}$$

where $c_i$'s are elements drawn uniformly at random from a finite field. Assuming that coefficients $c_i$'s are known, it is clear that if all peers have at least three independent codewords (i.e., three linear independent equations), then the file can be recovered. Now, the game for RLNC is played a bit different from the previous two. When a peer departs, the new peer randomly chooses $M = 2$ peers, then copies their data. However, since the new peer's storage capacity is only $C/3$

bits, it generates and stores only one new codeword as a random linear combination of the two codewords it just copied.

In all of these strategies, the game ends at the moment when all the peers collectively cannot recover the original file. We will show theoretically that the RLNC based strategy is much better than the others, i.e., it will take longer to play the game.

## IV. Discrete Stochastic Model for Random Linear Network Coding Based Replenishment Strategy

In this section, we describe a discrete stochastic model for the RLNC based replenishment strategy. To help with the modeling process, we start with the following claim:

*Given $N$ codewords, each is a vector of $L$ elements in a finite field $\mathbb{F}$. A new codeword of the same length is generated with the elements drawn uniformly at random from the same field. The probability that this new codeword is linearly independent from any combination $M$ codewords from the given $N$ codewords is almost unity if $L$ and $|\mathbb{F}|$ are sufficiently large.*

It is straightforward to show that the lower bound for this probability is $1 - \frac{\binom{N}{M}}{|\mathbb{F}|^{L-1}}$. As an example, suppose a file is broken up into $K = 3$ parts, then $N$ codewords are generated by linearly combining these three parts (codewords) at random. Now, at every time step, a new peer is chosen uniformly at random to depart. A new peer joins. Two distinct remaining peers ($M = 2$) are then uniformly chosen at random to have their codewords copied to the new peer. The new peer then generates its new codeword by linearly combining these two codewords with random coefficients. The file is no longer recoverable if all the codewords possessed by the peers collectively come from fewer than $K(= 3)$ independent codewords.

The diagram in Figure 1 visually depicts how the dependencies among the codewords progress in discrete time steps. The meanings of the solid and non-solid circles will become clear shortly when we discuss the time-backward process. For now, at time step $t = 0$, there are 7 codewords. Any of these codewords can be represented as a linearly combination of any three other codewords due to the initial mixing. At time step $t = 1$, the codeword 6 is replaced by a random linear combination of codewords 5 and 7. At this stage, the file can be recovered using any triplet of codewords except (5,6,7) since these three codewords are not linearly independent. At $t = 2$, codeword 2 is replaced by a random linear combination of codewords 1 and 3. As such, one cannot use triplets (5,6,7) or (1,2,3) to recover the file at this time. The process repeats, and eventually, all codewords will be some linear combinations of some two codewords, and the file will no longer be recoverable. A discrete time Markov chain representation, specifically a transition probability matrix can be used describe this replenishment process. However, a direct application of this method requires an exponentially large number of states ($N$) where a state denotes a configuration in the diagram. For example, at any time step, there are approximately $N \times \binom{N}{2}$ states that the chain can transition to, making this approach analytically intractable. Our contributions is a

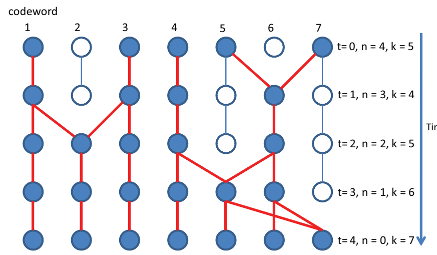


Fig. 1. Progression of codewords over time, $N = 7$, $M = 2$.

modeling technique that produces an approximate but closed form solution for the expected number of time steps to get from any state to any other, including the state in which the file is no longer recoverable. Furthermore, we can bound the error on this approximate time by a factor of 2 via the time-backward model.

## V. Time-Backward Model

We consider the time-backward process where the time is indexed as $n$ instead of $t$ as shown in Fig. 1. The solid circles denotes the parent nodes which are the codewords involving in the linear combination at different time steps. The non-solid circle represent codewords that are not parents nodes. Now, let $X_n$ denote the number of parent nodes at (backward) time $n$. For example, in Figure 1, $X_0 = 7$ and $X_4 = 5$. Clearly, all the codewords at time $n = 0$ are linearly dependent on the codewords 1,2,3,4,5,6 at time $n = 1$. All the codewords at time $n = 1$, are linearly dependent on the codewords 1, 2, 3, 4, 6 at time $n = 2$, and so on. With this setup, one can view the time-backward process as a one dimensional random walk $X_n$ on $2, 3, \ldots, N$. For $M = 2$, one can write down the following transition probabilities:

$$P(X_{n+1} = k - 1 | X_n = k) = \frac{k}{N}(\frac{k-1}{N-1})(\frac{k-2}{N-2})$$

$$P(X_{n+1} = k + 1 | X_n = k) = \frac{k}{N}(\frac{N-k}{N-1})(\frac{N-1-k}{N-2})$$

$$P(X_{n+1} = k | X_n = k) = 1 - \frac{k}{N}(\frac{k-1}{N-1})(\frac{k-2}{N-2}) - \frac{k}{N}(\frac{N-k}{N-1})(\frac{N-1-k}{N-2})$$

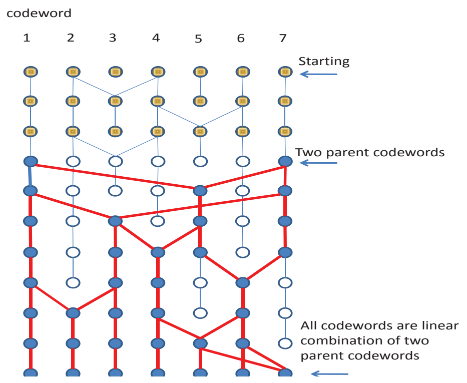
Suppose the number of original information packets is $K$, then when $X_n = K - 1$, we can artificially stop the process. At this stage, the file is no longer recoverable since the number of independent codewords is less than the number of information codewords. *With this backward time model, $X_n$ can only take on values between $K - 1$ and $N$, so the size of the transition matrix is $(N - K + 1) \times (N - K + 1)$, thus is much more manageable as compared to modeling the time-forward process.* We now show that the expected absorption time of the time-forward process can be approximated well by that of the corresponding time-backward walk with the following Proposition:

*Proposition 5.1:* Let $F$ and $B$ be the random variables denoting the absorption times of the time-forward process and time-backward walk with the parameters $K$, $N$, and $M = 2$, starting in state $N$, respectively, then

$$\mathbb{E}B \leq \mathbb{E}F < 2\mathbb{E}B \tag{2}$$

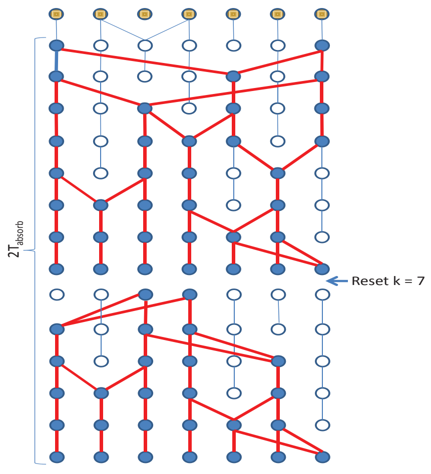
*Proof:* We first prove the lower bound. Without loss of generality, we assume $K = 3$ for the purpose of illustration. As shown in Fig. 2(a), a sequence of forward walk that results in all the codewords being the children of only two codewords must contain a sequence of backward walk that reaches these two codewords. Thus, $\mathbb{E}B \leq \mathbb{E}F$.

We now prove the upper bound. Suppose we walk backward from the state $X_0 = k = N$ until there are only two parents nodes at $n$ steps later ($X_n = 2$). We then reset $X_n = N$, then walk backward one more time as shown in Figure 2(b). The total expected number of time steps to reach the second merge, counting from the beginning, is $2\mathbb{E}B$. Now if the forward walk starts earlier than $2\mathbb{E}B$ time steps, then it must have encountered at least two instances where every node is a descendant of only two parents node. But this contradict the definition of $\mathbb{E}F_i$ as the number of time steps for the chain to reach $X_n = 2$ for the first time. Thus, $\mathbb{E}F < 2\mathbb{E}B$. ■



codeword
1
2
3
4
5
6
7
Starting
Two parent codewords
All codewords are linear combination of two parent codewords

(a)



(b)

Fig. 2. Illustrating diagrams for proofs of a) lowerbound; (b) upperbound.

## VI. Exponential Rate for Data Replenishment via Random Linear Network Coding

### A. Analysis of Exponential Absorption Time

The time-backward model allows one to compute a closed-form solution in matrix notations for the expected absorption time starting from any state using standard matrix techniques. The matrix notations however often cannot be used to examine the asymptotic behavior of the absorption time as a function of $N$, $K$, and $M$. To this end, we present a lower bound on the absorption time in terms of $N$, $K$, and $M$ with the following proposition:

*Proposition 6.1:* Given $N$, $K$, and $M = 2$, the mean absorption time $B$ for the time backward walk, starting in the state $X_0 = N$ and ending in state $X_n = K-1$, is at least:

$$B > \frac{\binom{2N-4}{N-2} - 1 - \sum_{i=1}^{K-1} \binom{N-2}{i}^2}{\binom{N-2}{K-1}^2} + \frac{2N-3}{N-3}. \tag{3}$$

For a large $N$, $B$ is bounded below by a simpler exponential in $N$ (assuming fixed $K$)

$$B > \frac{2^{2N-4} - (K-1)(N-2)^{2(K-1)}}{(N-2)^{2(K-1)}}. \tag{4}$$

For $K = 3$, even a more simplified lower bound is:

$$B > \frac{4^N}{16N^4} \tag{5}$$

*Proof:* We present a proof based on the classical method for computing hitting time of a discrete Markov chain. Denote $h_k$ as the mean absorption time starting in the state $X_0 = k$, for $k = K, K+1, \ldots, N-2$, and ending in state $X_n = K-1$ for some $n$. Then, we can write down the following recursion:

$$\begin{aligned} h_k = 1 \quad &+ \quad h_{k-1}\frac{k}{N}\left(\frac{k-1}{N-1}\right)\left(\frac{k-2}{N-2}\right) \\ &+ \quad h_{k+1}\frac{k}{N}\left(\frac{N-k}{N-1}\right)\left(\frac{N-1-k}{N-2}\right) \\ &+ \quad h_k\left[1-\frac{k}{N}\left(\frac{k-1}{N-1}\right)\left(\frac{k-2}{N-2}\right)\right] \\ &- \quad h_k\frac{k}{N}\left(\frac{N-k}{N-1}\right)\left(\frac{N-1-k}{N-2}\right) \end{aligned}$$

Letting $y_k = h_{k+1} - h_k$, and after some term re-arrangements, we have

$$y_{k-1} = y_k\frac{(N-k)(N-1-k)}{(k-1)(k-2)} + \frac{N(N-1)(N-2)}{k(k-1)(k-2)}$$

Now, this is a difference equation with the following initial conditions:

$$y_{N-1} = h_N - h_{N-1} = 1, y_{N-2} = h_{N-1} - h_{N-2} = \frac{N}{N-3}.$$

The first initial condition is true because the first step always reduces the number of parents nodes by 1. The second condition is true because in the special state $X_n = N-1$, the chain can either go to $(N-2)$ or stay at $(N-1)$. It will never go back to $N$. The probability that the chain will go to $N-2$ given that it is currently in $N-1$ is the probability that all three selected nodes are the parent nodes in the current time step. This probability is $\frac{(N-1)(N-2)(N-3)}{N(N-1)(N-2)} = \frac{N-3}{N}$. Thus the expected number of trials before moving to $N-2$ is $\frac{N}{N-3}$. The difference equation is of the form:

$$y_{k-1} = a_k y_k + b_k, \tag{6}$$

where

$$a_k = \frac{(N-k)(N-1-k)}{(k-1)(k-2)}, b_k = \frac{N(N-1)(N-2)}{k(k-1)(k-2)},$$

for $k = K, K+1, \ldots, N-2$.

By performing a few recursions, starting at $N-2$, we have

$$\begin{aligned} y_{N-2-k} &= a_{N-2}a_{N-3}\ldots a_{N-1-k}y_{N-2} \\ &+ a_{N-3}a_{N-4}\ldots a_{N-1-k}b_{N-2} + \ldots \\ &+ a_{N-1-k}b_{N-k} + b_{N-1-k}, \end{aligned} \tag{7}$$

Now, we have

$$\sum_{j=K-1}^{N-3} y_j = h_{N-2} - h_{K-1} = h_{N-2}, \tag{8}$$

since $h_{K-1} = 0$. The expected number of time steps before the file is no longer recoverable starting from $X_0 = N$ is therefore:

$$h_N = h_{N-2} + 1 + \frac{N}{N-3} = \sum_{j=K-1}^{N-3} y_j + 1 + \frac{N}{N-3} \tag{9}$$

We now consider

$$\begin{aligned} z_{N-2-k} &= a_{N-2}a_{N-3}\ldots a_{N-1-k} \\ &+ a_{N-3}a_{N-4}\ldots a_{N-1-k} + \cdots + a_{N-1-k} \end{aligned}$$

for $k = K-1, K, K+1, \ldots, N-2$.

Note that $z_k$ is a modified version of $y_k$ as defined in Equation (7). It is easy to see that

$$y_k > z_k, \tag{10}$$

since $y_{N-2} > 1$ and $b_k > 1$. By (9) and (10), we have

$$h_N > z_{K-1}, \tag{11}$$

and we will show that $z_{K-1}$ has a lower bound shown in Proposition 6.1.

First we note that

$$a_k = \frac{(N-k)(N-1-k)}{(k-1)(k-2)} > \left(\frac{N-1-k}{k}\right)^2$$

Now let $a'_k = \left(\frac{N-1-k}{k}\right)^2$, we have

$$\begin{aligned} z_{K-1} &= a_K + a_K a_{K+1} + \cdots + a_K a_{K+1} \ldots a_{N-2} \\ &> a'_K + a'_K a'_{K+1} + \cdots + a'_K a'_{K+1} \ldots a'_{N-2} \\ &= \frac{\sum_{i=1}^{N-2} \prod_k = 1^i a'_k - \sum_{i=1}^{K-1} \prod_{k=1}^{i} a'_k}{\prod_{k=1}^{K-1} a'_k} \\ &= \frac{\sum_{i=1}^{N-2} \prod_{k=1}^{i} (\frac{N-1-k}{k})^2 - \sum_{i=1}^{K-1} \prod_{k=1}^{i} (\frac{N-1-k}{k})^2}{\prod_{k=1}^{K-1} (\frac{N-1-k}{k})^2} \\ &= \frac{\binom{2N-4}{N-2} - 1 - \sum_{i=1}^{K-1} \binom{N-2}{i}^2}{\binom{N-2}{K-1}^2} \end{aligned} \tag{12}$$

Adding additional time steps from $X_0 = N$ to $X_t = N-2$, to (12), we obtain the absorption time shown in Proposition 6.1. Subsequent simpler bounds can be obtained using the Stirling's formula for large $N$ and noting that $(N-2)(N-3)\ldots(N-2-K+1) < (N-2)^{K-1}$. ■

By Proposition 5.1, the absorption time for the time-forward model must be as large as the absorption time for the time-backward walk $B$. For a fixed $K$, it is exponential in $N$.

## *B. Simulation results for RLNC based data replenishment*

In this section, we show the simulation results that demonstrate the theoretical exponential absorption time using the RLNC-based data replenishment. Furthermore, our simulation results show that the absorption time of the time-forward model is very close to that of time-backward model, i.e. much smaller closer to the lower bound than the upper bound given in Proposition 5.1.

First we show that the absorption time is indeed exponential in $N$ for fixed $K$ and $M$. Figure 3 shows the log of mean absorption time as a function of $N$, the number of nodes for both the time-forward and time-backward models. Recall that $N$ is the number of nodes used to store the data. In this simulation, an arrival node always connects to $M = 2$ existing nodes to download and mix their data. Originally, there is a total of three pieces of independent information $(K = 3)$.

The graphs in Figure 3 shows two relatively straight line segments. Since the $y$-axis is in log scale, this indicates that both the absorption times of the time-forward and time-backward models are exponential in the number of nodes used to store the data. Furthermore, both absorption times are almost identical in the log scale. This closeness also exhibits in the linear scale graphs (not shown). It is noted that he absorption time for the time-forward model is always larger than that of the time-backward model, following the lower bound in Proposition 5.1. Next, we investigate the absorption

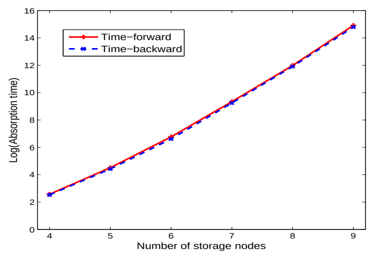


Fig. 3. Log of mean absorption time vs. the number of nodes for $N = 4,..,9$; $M = 2, K = 3$ for the RLNC strategy.

time as a function of redundancy given a constant number of nodes $N = 9$. Specifically, if at the start, there are $k < N$ pieces of independent information, then the absorption time is the time that the number of parents nodes reduces to $k-1$. Every newly arrival peer still connects to $M = 2$ peers for data replenishment. Figure 4 shows log of mean absorption time versus $k$, the number of parents nodes. Again, the absorption time for the time-forward model is higher than that of the time-backward model as predicted, but they are close to each other. Finally, we also investigate the performance of the RLNC-based data replenishment scheme when $M \geq 2$ connections are used. Specifically, a newly arrival peer chooses two, three, or four peers uniformly at random to download the data and perform replenishment. Figure 5 shows the log of the absorption time time-backward vs. $k$, the number of parents nodes, for the case of $N = 8$ nodes. The simulation results show that for larger values of $M$, one can expect a longer

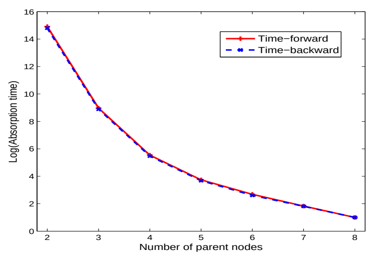
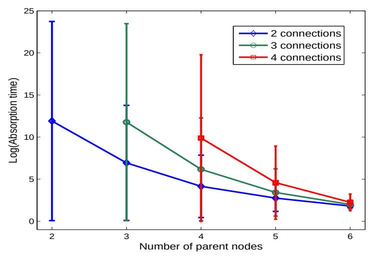
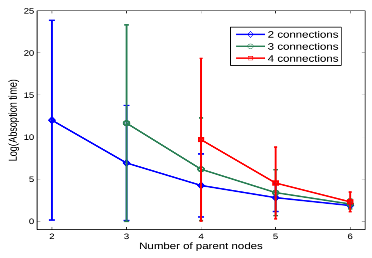


Fig. 4. Log of mean absorption time vs. the number of parent nodes for $N = 9; M = 2, K = 3$ for the RLNC strategy.



(a)



(b)

Fig. 5. Log of mean absorption time vs. number of parent nodes for $N = 8; M = 2, 3, 4$ for the RLNC strategy with (a) Time-forward model and (b) Time-backward model.

absorption time.

## VII. Quadratic Rate for Data Replenishment via RS and Repetition Codes

We show that using replenishment based on the RS and repetition codes, the number of time steps before a file is no longer recoverable is of $O(N^2)$, and thus is less effective than that of the RLNC based strategy.

**Absorption Time for RS-based Strategy.** A file is divided into three parts, coded using $RS(N, 3)$. Each peer keeps a codeword, resulting in a redundancy level of $N/3$. A new peer is allowed to contact with $M = 2$ peers. Since with $M = 2$, the new peer cannot recover the file, thus it will randomly copy the codeword from one of the two peers. Using the time-backward model, it is straightforward to obtain the following Proposition:

*Proposition 7.1:* The expected absorption time using RS code with $M = 2, K = 3$ is approximately $\frac{(N-1)(N-2)}{2}$, and thus is quadratic in $N$.

*Proof:* We omitted the proof due to lack of space, but simply note that we applied the same method used in the RLNC case for this proof. ■

**Absorption Time for Repetition Code Strategy.** Suppose a file is split into two parts and there are $N$ peers, each containing either parts of the file. For this strategy, whenever a peer leaves and a new peer enters, a peer is picked uniformly at random out of $N - 1$ existing peers, and its data is copied to the new peer.

*Proposition 7.2:* The expected absorption time using repetition code is approximately $\ln 2 \cdot N^2$.

*Proof:* We omit the proof due to lack of space, but merely mention that our proof uses the birth-and-death process $\{0, 1, \ldots, N\}$ with two absorbing states, $0$ and $N$. ■

## VIII. Concluding Remarks

In conclusion, we suggest that, to maintain data for as long as possible in a distributed setting with limited peer communication and storage, it is better to mix the data as proposed in the RLNC strategy. We show that the average number of replenishments before a file is no longer recoverable is exponential in the number of peers used store the data distributedly for RLNC-based strategy and quadratic for other traditional strategies. We also propose a novel time-backward technique to approximate the mean absorption time.

## References

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of ACM SIGCOMM*, August 2001.

[2] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IIEEE Journal on Selected Areas in Communications*, January 2004.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.

[4] Kien Nguyen, Thinh Nguyen, and Sen-Ching Cheung, "Video streaming with network coding," *Journal of Signal Processing Systems*, vol. 59, pp. 319–333, 2010, 10.1007/s11265-009-0342-7.

[5] Kien Nguyen, Thinh Nguyen, and Y. Kovchegov, "A p2p video delivery network (p2p-vdn)," in *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, 2009, pp. 1 –7.

[6] s. Acendanski, S. Deb, M. Medard, and R. Koetter, "How good is random linear coding based distributed networked storage?," in *NetCod*, 2005.

[7] A.G. Dimakis, P.B. Godfrey, Yunnan Wu, M.J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4539 –4551, Sept 2010.

[8] Alexandros G. Dimakis, Kannan Ramchandran, Yunnan Wu, and Changho Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, March 2011.

[9] Jun Li, Shuang Yang, Xin Wang, and Baochun Li, "Tree-structured data regeneration in distributed storage systems with regenerating codes," in *Proceedings of the 29th conference on Information communications*, Piscataway, NJ, USA, 2010, INFOCOM'10, pp. 2892–2900, IEEE Press.

[10] A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, 2009, pp. 376 –384.

[11] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, October 2006.

[12] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, July 2000.