Distributed Data Replenishment

Kien Nguyen, Thinh Nguyen, *Member, IEEE,* Yevgeniy Kovchegov, Viet Le

***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 that 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 are 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 the data replenishment via random linear network coding (RLNC) outperforms other popular strategies. 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 number of peers used to store the data for the RLNC-based strategy, while they are quadratic for other strategies.**

***Index Terms*—Stochastic Process, Absorption Time, Distributed Storage, Network Coding**

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], [3], [4]. In this paper, we investigate some 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 networks such as BitTorrent [5] and KaZaA [6] for example, are 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 integrity and availability of 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 various aspects of indexing and retrieving data. These topics have been well investigated in [1], [2], [7], [8], [9], [10], [11], [12], [13]. 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 and preferably distributed for it to be effective in highly dynamic 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 [14], [15], [16], [17]. A user recovers the file by downloading its many pieces simultaneously from different locations. In this paper, we consider the following variant of the setup described in [16], [18], [19]. In this setup, a file to be stored, is first broken up into multiple pieces or packets, coded using either Reed-Solomon, repetition, or

Kien Nguyen, Thinh Nguyen, and Viet Le are with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, 97331 USA (e-mail: {nguyenki,thinhq,lev}@eecs.oregonstate.edu).

Yevgeniy Kovchegov is with the Department of Mathematics, Oregon State University, Corvallis, OR 97331-4605, USA (e-mail: kovchegy@math.oregonstate.edu).

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. We note that the concept of data replenishment is very much similar to data repair as termed in [18]. Thus, our work is not new in the sense of using RLNC for distributed data storage. On the other hand, in [16], [17], [18], [19], 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. Specifically, we show that RLNC-based storage technique in P2P environments will, on average, result in higher data availability over time than other techniques.

The outline of our paper is as follows. In Section II, we provide a brief review on the recent advances in distributed storage with an emphasis on network coding. In Section III, we describe different replenishment strategies for a synchronous network model. In Section IV, we model the evolution of a piece of data through time for different replenishment schemes as discrete stochastic processes. These processes however are intractable, therefore we propose a *time-backward* model based on which, we are able determine an approximate closed form expression for the elapsed time that a piece of data is expected to remain in the system. We call this expected time the absorption time. Our analytical results show that using the data replenishment via RLNC, the absorption time is exponential in the number of peers used to store the data. This is much more robust than other data protection strategies based on repetition or channel coding techniques whose absorption times are quadratic in the number of peers. As an extension, in Section VIII, we present an analysis for an asynchronous model which describes peer arrivals and departures as Poisson processes. Our results indicate that the performance of the asynchronous model depends critically on the peer arrival and departure rates.

II. Related Work

In their seminal paper, Ahlswede et al. showed that it is possible to maximize the transmission rate from a source to multiple receivers (multicast capacity) by allowing the intermediate nodes to perform coding, i.e., mixing data from different flows [20]. Since then, network coding has found its way to many applications, ranging from efficient wireless communication and networking to distributed storage systems. Our work is motivated by the recent advances in network coding for distributed storage systems.

In [21], Dimakis et al. provided a survey on recent network coding techniques for distributed storage. Much research 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 capacity-achieving 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 (Maximum Distance Separable) codes provided that the storage capacities of nodes in the systems are sufficiently large [17]. In [22], Wu et al. showed techniques that can reduce the repair bandwidth in the case where there is only a single node failure. Wu et al. also proposed techniques for constructing MDS codes that achieve the fundamental bounds on the minimum repair bandwidth and storage capacity [23], [24].

Li et al. also considered joint design of regenerating codes and network topology for efficiently utilizing network links and reducing repair bandwidth [25]. Their design called RCTREE, is able to produce efficient code regeneration, even in dynamic environments where nodes enter and depart the network frequently. In [26], 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.

Regenerating codes also finds security applications in distributed storage systems. In [27], Pawar et al. derived the fundamental bounds on the storage capacity that guarantee against eavesdropping and adversarial attacks.

We want to point out a few differences between this work and the existing works. First, all the existing works focus on the constructing regenerating codes optimally. This requires detail information (which packets each node has) exchange between the newly joined node and the nodes it connects to for repairing the data. In contrast, our work is focused on a randomized approach in which a newly joined node connects to a few nodes at random and download data from these nodes. This method is suboptimal but is scalable since the information exchange between nodes is kept to minimal. Second, in many existing works, the authors examined the fundamental tradeoff between the repair 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. Specifically, we show that RLNC-based storage technique in P2P environments will, on average, result higher data availability over time than other techniques.

III. Synchronous Network Model and Data Replenishment Strategies

Our distributed storage systems of interest are the types whose files are not stored in their entirety at a specific location. Rather, they are broken up into many pieces, coded for redundancy, and dispersed to multiple locations in a P2P network. When a particular file is requested, its pieces are downloaded simultaneously from multiple locations. It can be shown that this method increases data availability and reduces congestion bottlenecks [14], [15].

When a peer leaves the network, so does its data. This effectively reduces the robustness of the system if the peer never rejoins or rejoins without its data. To avoid this, a peer can transfer its data to some other peers before its departure. However, if the data is large, a peer is less willing to wait until the transfer completes. Thus, without any proactive data replenishment, the redundancy level of a piece of data in the network is continuously reduced. After some period of time, the data of interest is not likely to be recoverable.

Theoretically, if one is to replace the exact missing data in the network, the redundancy level would remain the same. However, this requires global knowledge. Specifically the system needs to know the departed peer and its data. Then, a precise coordination and communication mechanism is needed to reproduce the equivalent state of the network prior to the peer's departure. This potentially creates significant communication and coordination overheads. Instead, 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.* In this paper, we explore techniques for this approach to maintain the data in the network for as long as possible while minimizing the coordination and communication overheads.

To capture how data redundancy in the network evolves over time, it is important to model the peer arrival and departure processes. For the majority of the paper, we will study a synchronous model for peer arrival and departure. In this synchronous model, for every peer that leaves the network, the system can find another peer to take over the responsibility of the departed peer. Under certain setting, this model approximates the dynamics of a network with constant number of peers since the departures and arrivals are synchronized. This is the most interesting model as the advantage of RLNC technique can be clearly demonstrated over other popular channel and repetition coding techniques. A brief analysis of a more general model with Poisson arrival and departures will also be discussed.

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, e.g., a file of C bits 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?

Note that, if $M = N - 1$, i.e., the new peer is able to communicate with every other peers, thus it will know exactly what the missing data is, and will be able to restore the missing data quite easily. When $M < N - 1$, it is impossible to know with certainty what data is missing. However, some form of data replenishment might be sufficient to maintain certain level of redundancy in the system. We now describe three replenishment strategies:

Repetition Code Based Strategy: To be specific, 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. It is not hard to see that this will happen with probability 1. When this happens, the file is no longer recoverable even with the help of all the peers. But before this happens, all the peers collectively will be able to provide 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 [28], [29]. 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. With $M = 2$, the new peer would not be able reconstruct the entire file. Therefore, its best strategy is to choose one of the codewords from the two contacted peers at random, and copies this codeword to itself to increase data redundancy in the system.

Random Linear Network Code Based Strategy: A yet intuitively better strategy is to employ Random Linear Network Coding (RLNC) technique [20][30]. 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. The N codewords are distributed to each peer, each keeps a codeword. This process is done at the beginning when a file is first stored in the network. Note that the redundancy is $N/3$, identical to that of RS code strategy. Mathematically, an $n-bit$ pattern ($n << C/3$) can be viewed as an element from a finite field. Thus, a codeword of $C/3$ bits which consists of a vector of $n - bit$ patterns, can be viewed as a vector of elements from a finite field. Most practical implementations of network coding use $n = 16$ or 32. A codeword A is a random linear combination of codewords B and C, then

$$A = c_1B + c_2C, \tag{1}$$

where c_i's are elements drawn uniformly at random from a finite field. All the operations in the equation above are finite field operations. Assuming that coefficients c_i's are known, it is clear that if all peers have at least three independent codewords (which are formed by three linear independent equations), then the file can be recovered. Note that the number of bits that represents the coefficients is included in the codewords, and is negligible for sufficiently long codewords. 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, following the Equation (1). Note that RLNC is performed at the beginning when the a file is stored, and also at each replenishment.

In all of these strategies, the game ends at the moment when all the peers together 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. In practice, one can view the distributed storage system based on the RLNC strategy as letting individual peers to perform a simple task in a random manner. However after some time, the data might no longer be recoverable. Thus, a practical system will allow simple replenishments to continue until the level of redundancy is deemed to fall below a certain threshold, then a more expensive, full-blown replenishment is performed to restore the full level of redundancy. This kind of systems is more scalable than ones that perform expensive replenishment at every single peer departure.

Remark: We note that the RLNC-based strategy preserves data in the network exponentially longer, it is the least efficient of the three in terms of replenishment (repair) bandwidth. This is because the amount of download data is more than a peer's storage capacity . To a certain extend, this is related to the unavoidable tradeoff between repair bandwidth and storage capacity as discussed in [21].

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. This model is analytically intractable due to a large number of states. However, it serves as a motivation for the proposed time-backward technique that approximates the expected time until the file is no longer recoverable.

As described in Section III, each replenished codeword can be viewed as a row in a matrix, and is linearly dependent on the other rows that were used to generate it. Therefore, over time one would expect the number of linearly independent rows decreases. Eventually, the rank of the matrix will reduce below the number of original data packets. At this point, even when all rows are used, the data cannot be recovered. Our objective is to determine the average time until this happens.

To help with the modeling process, we start with the following claim:

Claim 4.1: 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. [1]

Another way to view this is that if the elements are drawn from $\mathbb{R}$, then a randomly drawn row will definitely be independent from any other M equations because $|\mathbb{R}|$ is infinite. We will make this approximation to model the replenishment process as follows.

For simplicity, we will focus on the following simple scenario. 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. In general, if $M \geq K$, it will almost always be possible to recover the file, independent of the number of replenishments. This is because we can almost always get K linearly independent codewords, unless with a small probability, the generated codeword happens to be linearly dependent on some $M' < K$ codewords.

We use the diagram in Figure 1 to visually depict 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

[1]It is straightforward to show that the lower bound for this probability is $1 - \frac{\binom{N}{M}}{|\mathbb{F}|^{L-1}}$.

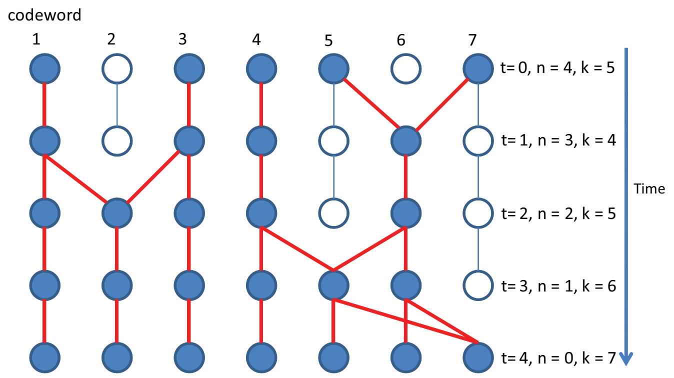


Fig. 1. Progression of codewords for seven peers, $N = 7$, $M = 2$ in discrete time steps. The index t represent time-forward process while the index n represent time-backward process. Codewords are represented by the circles. A circle in the current time step that is connected to two circles in the previous time step, represents a codeword that is a linear combination of two codewords. Solid circles are parent codewords that are part of the linear combinations in the current codewords, while non-solid circles are not part of the linear combinations of the current 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 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 contribution is a 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. The key to this modeling technique is to consider a more tractable time-backward model in which the replenishments are performed backward in time.

V. Time-Backward Model

To contrast the time-forward model, we use the index n to denote time step for the time-backward process. In Fig. 1 shows the time-backward walk begins with $n = 0$ and end with $n = 4$. The figure also shows two types of circles. 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 parent nodes. Now, let X_n denote the number of parent nodes at time n where n denotes the number of time steps from the initial state with the number of parent nodes $X_0 = N$. 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 the case where $M = 2$, one can write down the following transition probabilities:

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

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

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

$$-\frac{k}{N}(\frac{N-k}{N-1})(\frac{N-1-k}{N-2})$$

Furthermore, when $X_n = M$, we can artificially stop the process, i.e., setting the transition probabilities from this state to all other states to 0. At this stage, the file is no longer recoverable.

That said, X_n can only take on values between M and N, so the size of the transition matrix $\mathbf{P}$ is $(N - M + 1) \times (N - M + 1)$, thus is much more manageable as compared to modeling the time-forward process.

For example, with $M = 2$, $N = 7$, the corresponding transition probability matrix is:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1/35 & 4/5 & 6/35 & 0 & 0 & 0 \\ 0 & 4/35 & 27/35 & 4/35 & 0 & 0 \\ 0 & 0 & 2/7 & 2/3 & 1/21 & 0 \\ 0 & 0 & 0 & 4/7 & 3/7 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Note that this matrix has one recurrent state $X_n = 2$ (the first row in the matrix), the rest of the states are transient.

A. Mean Absorption Time

Based on $\mathbf{P}$, one can immediately compute the mean absorption time, i.e., the expected number of time steps, starting from an initial transient state to a recurrent state, using a standard technique. Specifically, let $\mathbf{Q}$ be the submatrix of $\mathbf{P}$ that includes only the rows and columns corresponding to the transient states, then by rearranging the order of the states, one can write the transition probability matrix $\mathbf{P}$ as:

$$\mathbf{P} = \begin{pmatrix} \tilde{\mathbf{P}} & \mathbf{0} \\ \mathbf{S} & \mathbf{Q} \end{pmatrix}.$$

Let

$$\mathbf{M} = (\mathbf{I} - \mathbf{Q})^{-1}, \tag{2}$$

Let B_i be the random variable denoting the absorption time starting in a transient state i, the the mean absorption time is:

$$\mathbb{E}B_i = \sum_j \mathbf{M}_{ij} \tag{3}$$

We also include a recursive algorithm for computing the variance of the absorption time in the appendix.

B. Bounding Absorption Time of Time-Forward Process with Time-Backward Walk

We have shown that, in contrast to the time-forward process, modeling the corresponding time-backward walk is quite tractable. 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. Specifically, we have the following Proposition:

Proposition 5.1: Let F_i and B_i be the random variables denoting the absorption times of the time-forward process and time-backward walk, starting in state i, respectively, then

$$\mathbb{E}B_i \leq \mathbb{E}F_i < 2\mathbb{E}B_i \tag{4}$$

Proof: We first prove the lower bound. 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, $\mathbb{E}B_i \leq \mathbb{E}F_i$.

Next, we prove the upper bound. First, we examine the procedure that is used to recognize whether we have reached the absorption state. The procedure starts with a time-forward process. At every new time step, we trace the parent nodes of the current nodes by tracing their ancestors back in time. During this backward trace, if at point in time, there are only two ancestors, then we declare that $F_i = n$. Note that since we perform this procedure at every new time step, n is the the first time that we encounter the absorption state. This trace must consist exactly one time-backward sequence that leads to only two parent nodes. Otherwise, n would not be the first time that the time-forward process encounters the absorption time. Consequently, $\mathbb{E}F_i < 2\mathbb{E}B_i$.

Alternatively, suppose given that the time-forward process is in the absorption state for the first time. We want to bound how long ago the time-forward process has started. We can do this by performing a backward walk starting in the absorption state. During this backward walk, at the first time that the number of parent nodes is two, we reset the number of parent nodes to N, and walk backward again until the number of parent nodes is two again. Now we argue that the starting time of the time-forward process cannot be larger than twice the absorption time for the backward walk since it if is (as shown in Figure 2(b)), the first time the time-forward process is in the absorption state, is at the "reset" state (in the middle of the Figure 2(b)), contradicting our assumption that it is at the bottom of the Figure 2(b). Thus, $\mathbb{E}F_i < 2\mathbb{E}B_i$. ■

Although the theoretical upper bound on $\mathbb{E}F_i$ is a bit loose, simulation results in the Section VI-B reveal that $\mathbb{E}F_i$ is quite close to $\mathbb{E}B_i$.

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:

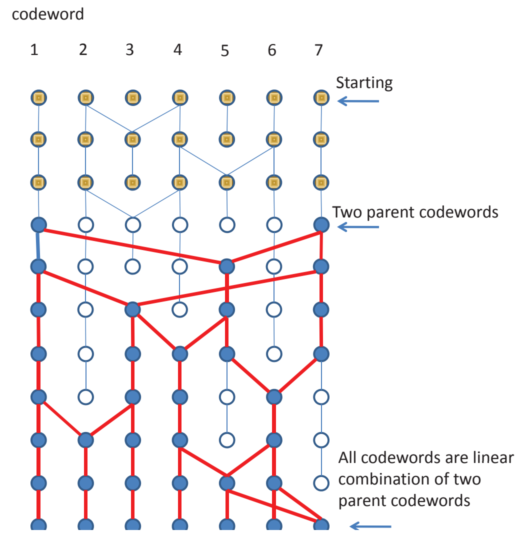
$$B > \frac{\binom{2N-4}{N-2} - 1 - \sum_{i=1}^{K-1} \binom{N-2}{i}^2}{\binom{N-2}{K-1}^2} + \frac{2N-3}{N-3}. \quad (5)$$

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

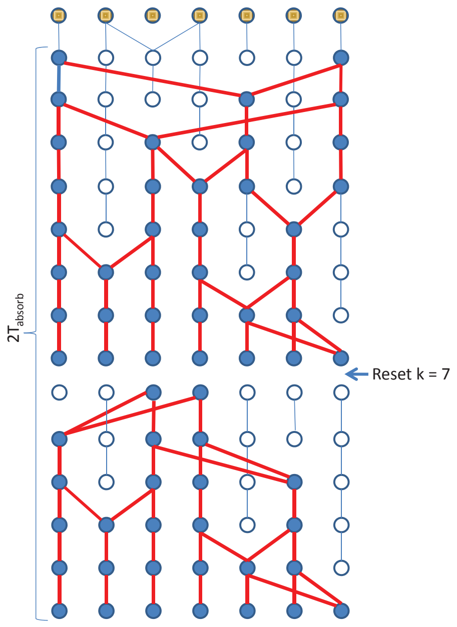
$$B > \frac{2^{2N-4} - (K-1)(N-2)^{2(K-1)}}{(N-2)^{2(K-1)}}. \quad (6)$$

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

$$B > \frac{4^N}{16N^4} \quad (7)$$


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 a) proof of lower bound; (b) proof of upper bound showing the impossible starting point for a forward walk .

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 rearrangements, 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:

$$y_{k-1} = a_k y_k + b_k, \tag{8}$$

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

$$\begin{aligned} y_{N-2-k} \quad &= \quad a_{N-2}a_{N-3}\ldots a_{N-1-k}y_{N-2} \qquad (9) \\ &+ \quad a_{N-3}a_{N-4}\ldots a_{N-1-k}b_{N-2} + \ldots \\ &+ \quad a_{N-1-k}b_{N-k} + b_{N-1-k}, \end{aligned}$$

Now, we have

$$\sum_{j=K-1}^{N-3} y_j = h_{N-2} - h_{K-1} = h_{N-2}, \tag{10}$$

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:

$$h_N = h_{N-2} + 1 + \frac{N}{N-3} = \sum_{j=K-1}^{N-3} y_j + 1 + \frac{N}{N-3} \tag{11}$$

We now consider

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

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 (9). It is not hard to see that

$$y_k > z_k, \tag{12}$$

since $y_{N-2} > 1$ and $b_k > 1$. By (11) and (12), we have

$$h_N > z_{K-1}, \tag{13}$$

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

$$\begin{aligned} z_{K-1} \quad &= \quad a_K + a_K a_{K+1} + \cdots + a_K a_{K+1}\ldots a_{N-2} \\ &> \quad a'_K + a'_K a'_{K+1} + \cdots + a'_K a'_{K+1}\ldots a'_{N-2} \\ &= \quad \frac{\sum_{i=1}^{N-2}\prod_k = 1^i a'_k - \sum_{i=1}^{K-1}\prod_{k=1}^{i} a'_k}{\prod_{k=1}^{K-1} a'_k} \\ &= \quad \frac{\sum_{i=1}^{N-2}\prod_{k=1}^{i}(\frac{N-1-k}{k})^2 - \sum_{i=1}^{K-1}\prod_{k=1}^{i}(\frac{N-1-k}{k})^2}{\prod_{k=1}^{K-1}(\frac{N-1-k}{k})^2} \\ &= \quad \frac{\binom{2N-4}{N-2} - 1 - \sum_{i=1}^{K-1}\binom{N-2}{i}^2}{\binom{N-2}{K-1}^2} \qquad (14) \end{aligned}$$

Adding additional time steps from $X_0 = N$ to $X_t = N-2$, to (14), 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.

An important result of Proposition 6.1 is that for fixed K and large N, the absorption time is at least exponential in N.

Remarks for the case $M > 2$: Computing the absorption time for the case $M > 2$ is related to a very hard problem in the study of Markov chains. Specifically, characterizing the asymptotic behavior of a Markov chain with a general transition probability P is an open problem. This involves the computation of the second largest eigenvalue of P which in general is hard to obtain analytically (though it can be obtained numerically via algorithms, e.g., eigendecomposition). For $M = 2$ and fixed small K, we already showed that the absorption time is exponential in N, then it is intuitive plausible that for $M > 2$, the absorption time would be even longer since there are more mixing involved. So the absorption time must be at least exponential as well (perhaps with higher exponent). While it is possible to obtain a tighter bound for the case $M > 2$, we would like to investigate this scenario in future work.

B. Simulation results for RLNC based data replenishment

In this section, we show the simulation results that demonstrate the theoretical exponential absorption time in N for large N and fixed K ($M = 2$) 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. Therefore in many settings, replacing the time-forward model with the time-forward model would not sacrifice much accuracy on how long a piece of data is to remain in the system. Our specific goal is to examine how long a piece of data is to remain in the system. Therefore, instead of using an accurate simulator that reflects time-varying bandwidth, packet loss, delay, etc. to examine other performance metrics of a realistic system, we choose to use a following simple simulation setting. A node is picked uniformly at random to leave the system. M out of $N - 1$ remaining nodes are picked uniformly at random for data replenishment, i.e., one incoming node will download data from these M nodes, mix (network coding) and store the mixed data. In other words, our simulator is simply an algorithm to simulate the random processes of picking which nodes to leave and which remaining nodes to use their data for mixing.

First, we want to show that the absorption time is indeed exponential in N for fixed M. Figure 3 shows the log of mean absorption time as a function of N for both the time-forward and time-backward models, with fixed $M = 2$ and $K = 3$. In this simulation, a newly arrival node always connects to $M = 2$ existing nodes chose uniformly at random (out of $N - 1$ nodes) to download and mix their data. Originally, there is a total of three pieces of independent information ($K = 3$), i.e., the redundancy ratio is $N/3$. The absorption time is the expected time (the number of pairs of departures and arrivals) until all the nodes collectively contain exactly two pieces of independent information. From the previous stochastic model, this happens when the number of parent nodes equal to two. 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 the 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.

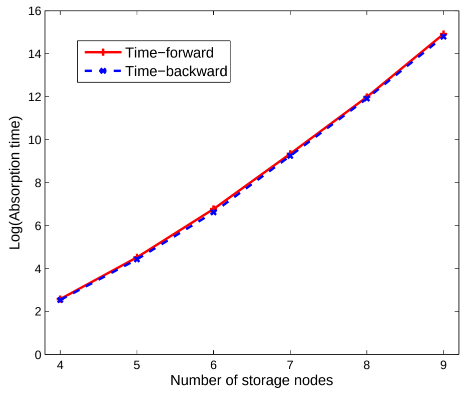


Fig. 3. Log of mean absorption time vs. the number of nodes for $N = 4, .., 9, M = 2$, and $K = 3$ for the RLNC strategy.

Next, we investigate the absorption 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 parent 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 parent 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.

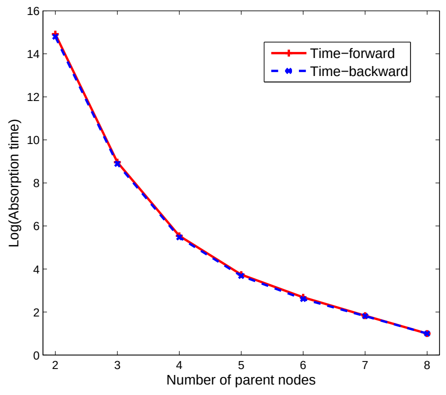


Fig. 4. Log of mean absorption time vs. the number of parent nodes for $N = 9, M = 2$, and $K = 2, \ldots, 8$, for the RLNC strategy.

Figure 5 shows the absorption times (minimum, maximum, and median) for the time-forward and time-backward models when the number of nodes N varies from 6 to 9, $M = 2$, and $K = 3$. Again, they are very much in agreement with 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 6 shows the log of the absorption time time-backward vs. k, the number of parent nodes, for the case of $N = 8$ nodes.

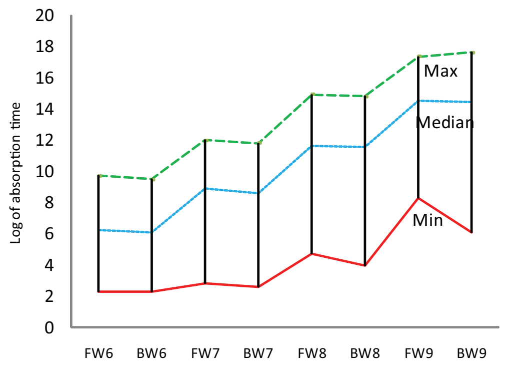
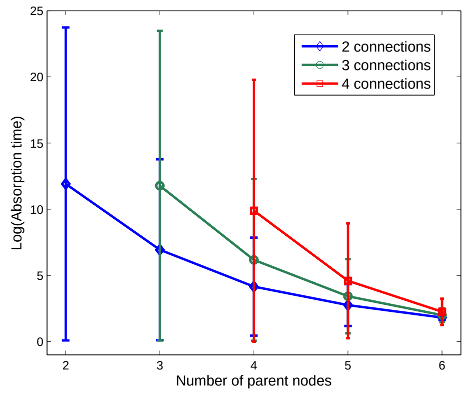
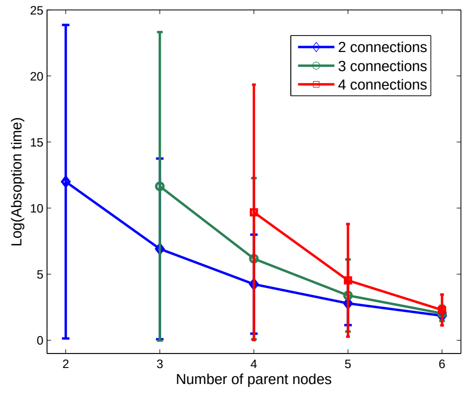


Fig. 5. Log of absorption time vs number of original nodes for $N = 6, .., 9, M = 2$, and $K = 3$ for the RLNC strategy.

The simulation results show that for larger values of M, one can expect a longer absorption time. This is intuitively plausible as a larger M would reduce the chance of creating dependency at each replenishment steps.



(a)



(b)

Fig. 6. Log of mean absorption time and its standard deviation vs number of parent nodes for $N = 8, M = 2, 3, 4$, and $K = M + 1$, for the RLNC strategy with (a) Time-forward model and (b) Time-backward model.

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

In this section, 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. We begin with the RS based strategy.

A. *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. In this special case, it is just as good as communicating with only $M = 1$ peer. Note that it is also straightforward to analyze the general case where $RS(N, K)$ with $(2 < M < K)$ is used. In this general case, the new peer still cannot reconstruct the file, however it is potentially better to copy the data that is least duplicated among the M peers, so as to increase the diversity. For simplicity, let us consider the case where $M = 2$, or effectively $M = 1$.

Using the time-backward model, it is straightforward to model the RS-based strategy as shown in Fig. 7.

Similar to the RLNC strategy, let k denote the number of parent nodes. A file is then irrecoverable when $k = 2$. We can write the mean absorption time using the recursion as

$$\begin{aligned} h_k &= \frac{k}{N}(\frac{k-1}{N-1})h_{k-1} + (1 - \frac{k}{N}\frac{k-1}{N-1})h_k + 1 \\ h_k &= h_{k-1} + \frac{N(N-1)}{k(k-1)} \end{aligned} \tag{15}$$

for $k = 3, 4, \ldots, N$.

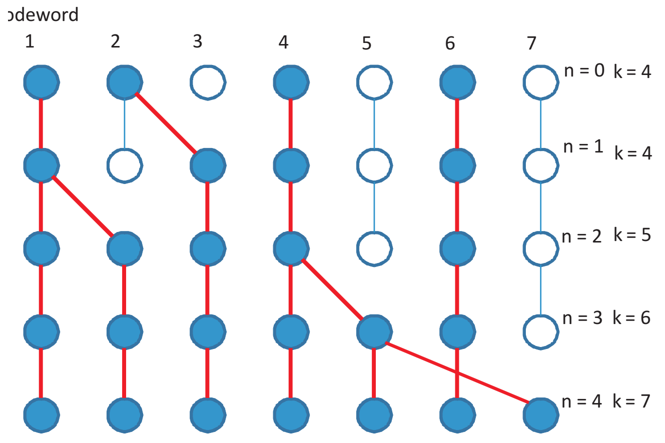


Fig. 7. Progression of codewords for seven peers, $N = 7$, $M = 2$ in discrete time step n. At each time step, a codeword is replaced by another.

Now $h_2 = 0$ hence,

$$\begin{aligned} h_N &= \sum_{k=2}^{N-1} \frac{N(N-1)}{k(k+1)} \\ &= N(N-1)\sum_{k=2}^{N-1}\left(\frac{1}{k} - \frac{1}{k+1}\right) \\ &= \frac{(N-1)(N-2)}{2} \end{aligned} \tag{16}$$

By Proposition 5.1, the absorption time of the time-forward process cannot be more than h_N.

Note that if $RS(N,2)$ is used, i.e., redundancy is increased to $N/2$. Then, using the same method, we have $h_N = (N-1)^2$. Thus, the absorption time for the time-forward walk cannot be more than $2h_N$. For large N, this is a small improvement over the repetition code strategy for the same redundancy as shown below.

B. *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. The process above is of birth-and-death type on $\{0,1,\ldots,N\}$ with two absorbing states, 0 and N. We would like to estimate the mean absorption time.

In the above birth-and-death process the forward probabilities are given by

$$p_k = \frac{k(N-k)}{N(N-1)} \quad \text{for } k = 1,2,\ldots N, \quad \text{and } p_0 = 0$$

and the backward probabilities are

$$q_k = \frac{k(N-k)}{N(N-1)} \quad \text{for } k = 0,1,\ldots N-1, \quad \text{and } q_N = 0.$$

The expected absorption time $h_k = E_k[T_0 \wedge T_n]$ solves the following recurrence equation: For $k = 1,2,\ldots,N-1$,

$$\begin{aligned} h_k &= 1 + \frac{k(N-k)}{N(N-1)}h_{k-1} + \frac{k(N-k)}{N(N-1)}h_{k+1} \\ &+ \left(1 - \frac{2k(N-k)}{N(N-1)}\right)h_k \\ h_0 &= h_N = 0 \end{aligned} \tag{17}$$

The above equation can be rewritten as follows

$$y_k = -(N-1)\left(\frac{1}{k} + \frac{1}{N-k}\right) + y_{k-1},$$

where $y_k = h_{k+1} - h_k$.

Thus

$$\begin{aligned} y_k &= -(N-1)\left(1 + \frac{1}{2} + \cdots + \frac{1}{k}\right) \\ &- (N-1)\left(\frac{1}{N-k} + \cdots + \frac{1}{N-1}\right) + y_0. \end{aligned} \tag{18}$$

Now, by symmetry,

$$-y_0 = y_{N-1} = -2(N-1)\left(1 + \frac{1}{2} + \cdots + \frac{1}{N-1}\right) + y_0$$

and therefore

$$y_0 = (N-1)\left(1 + \frac{1}{2} + \cdots + \frac{1}{N-1}\right). \tag{19}$$

Plugging y_0 into (18), and after some algebraic manipulations, we obtain:

$$\begin{aligned} h_{k+1} &= (N-1)k\left(1 + \frac{1}{2} + \cdots + \frac{1}{N-1}\right) \\ &-(N-1)(k+1)\left(1 + \frac{1}{2} + \cdots + \frac{1}{k}\right) \\ &+(N-1)(N-k-1)\left(\frac{1}{N-k} + \cdots + \frac{1}{N-1}\right). \end{aligned} \tag{20}$$

Taking $k+1 = \frac{N}{2}$, we obtain

$$h\left(\frac{N}{2}\right) \approx \ln 2 \cdot N^2 \tag{21}$$

Remarks on performance of different schemes as N grows:

To provide some quantitative performance gain of RLNC over the RS and repetition coding replenishments, Fig. 8 plots the analytical lower bound of the mean absorption times (Proposition 6.1) for the RLNC replenishment together with the lower bound of mean absorption times using the RS coding and the mean absorption time using the repetition coding replenishments (Eq. (16) and Eq. (21)).

As seen for large N, RLNC-based strategy outperforms the other two strategies significantly. However, one must caution again that RLNC uses more replenishment bandwidth than the other two.

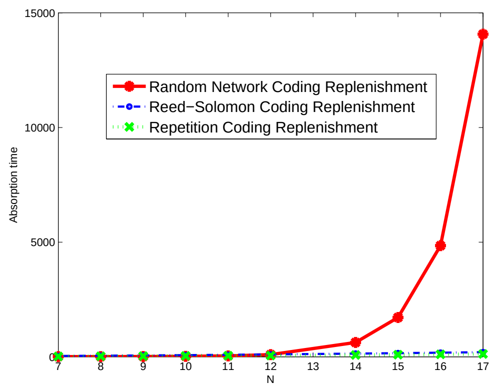


Fig. 8. Lower bounds of the mean absorption times vs. the number of nodes N for different schemes, $K = 3$.

VIII. Extension: Asynchronous Network Model

In Section III, we consider the network model in which, peer departures and arrivals are synchronized. In this section, this assumption is relaxed. Instead, we model the peer arrivals and departures as Poisson processes with rates $N\lambda$ and $N\mu$ [2] where N denotes the current number of peers in the network. It is not hard to show that when $\lambda < \mu$, the number of peers will reduce to zero quickly (linear with N) so that any form of replenishment will be ineffective in this scenario. On the other hand, if $\lambda \geq \mu$, a piece of data will be almost certain to remain in the network even when using the repetition coding technique, a weaker form of data replenishment. Intuitively, this is because the amount of storage increases with time outweighs the data dependency due to replenishment. Therefore, in this section, we will mainly address this model under the repetition coding technique for the case when $\lambda \geq \mu$, and show that there is a chance that the data will be disappeared , but this probability is exponentially small with N, the number of peers.

In particular, suppose the data is divided into r distinct equal-sized parts to be replicated at multiple nodes. We consider a system of $m \geq r$ peers, each peer stores exactly one part of the data. Redundancy is created by having multiple peers store the same part of the data. Peer arrivals and departures follows Poisson processes. Whenever a new peer arrives, it picks uniformly at random a peer from the existing peers and copies its data. Clearly, the data cannot be recovered if at least one part of the data is missing in system. *We are interested in computing this irrecoverable probability given that a distributed storage system begins with N peers.*

Remarks: It is important to note that in this setting ($\lambda \geq \mu$), the mean absorption time is infinite. Therefore, we do not consider the absorption time as a metric in this setting. To see why, we note that the absorption state is a transient state, i.e., there is a probability strictly less than 1 that the chain will reach to the absorption state (any state in which the number of peers is less than the number of distinct parts in the file), or equivalently there is a non-zero probability that the chain will never reach the absorption state. Clearly, when the chain never reaches the absorption state (this happens with non-zero probability), then the absorption time is infinite, so the mean absorption time is infinite. A simple classical example to illustrate this is to imagine a particle moving on a line with probability p to the left and $1 - p$ to the right. Suppose the particle starts 4, we ask does it ever reach 0? The answer depends on p. If $p > 1/2$, the answer is "yes", i.e. with probability 1 that the particle will eventually reach 0. On the other hand, if $p \geq 1/2$, then the particle might reach 0 with a positive probability but it is not 1. Thus, there is a positive probability that the particle will move to infinity without ever returning to 0. Therefore the absorption time is infinite.

One important observation is that this problem can be viewed as a classical parallel queuing system consisting of r statistically independent queues with different arrival and service rates. The number of packets in each distinct queue represent the number of peers storing the same parts of the data. Specifically, let n_1, n_2, $\ldots$, n_r be the queue sizes, i.e., the number of peers carrying the data parts 1 to r at any point in time. Then, it is sufficient to analyze the occupancy of any one queue. The departure rate of peers storing parts i is $k_i\mu/N$. The arrival rate of peer of type i is then $k_i\lambda/N$. We now analyze only one queue of interest, say queue i. To simplify the notation, let us denote $n = n_i$ as the number of packets in the queue i. Then, the number of packets in the queue (number of peers carrying the same parts of the data) evolves according to a birth-and-death process on $\{0, 1, \ldots\}$ with the recurrence state 0. It has the transition probabilities:

$$p(n, n-1) = \frac{n\mu}{n\lambda + n\mu} = \frac{\mu}{\lambda + \mu} \tag{22}$$

and

$$p(n, n+1) = \frac{n\lambda}{n\lambda + n\mu} = \frac{\lambda}{\lambda + \mu} \tag{23}$$

Let $a(n)$ be the probability that the system starting at state n ever reach state 0. Note that $a(0) = 1$ and the

[2]One can also make the peer arrival rate as λ instead of $N\lambda$ to reflect that the peer arrival rate is independent of the number of existing peers. However, this implies that no peer will survive after a short period of time (order of $\log N$), and thus this scenario is not interesting.

values of $a(n)$ is the same whether one considers the continuous-time or discrete-time system. It is satisfied the following recursive equation:

$$\begin{aligned} a(n) &= a(n-1)p(n,n-1) + a(n+1)p(n,n+1) \\ &= a(n-1)\frac{\mu}{\lambda+\mu} + a(n+1)\frac{\lambda}{\lambda+\mu},\ n \geq 1 \end{aligned} \quad (24)$$

Equation 24 can be rewritten as

$$a(n) - a(n+1) = \frac{\mu}{\lambda}[a(n-1) - a(n)], \quad n \geq 1 \quad (25)$$

We obtain

$$a(n) - a(n+1) = \frac{\mu^n}{\lambda^n}[a(0) - a(1)] \quad (26)$$

Hence,

$$\begin{aligned} a(n+1) &= a(n+1) - a(0) + a(0) \\ &= \sum_{j=0}^{n}[a(j+1) - a(j)] + a(0) \\ &= [a(1) - 1]\sum_{j=0}^{n}\left(\frac{\mu}{\lambda}\right)^j + 1 \end{aligned} \quad (27)$$

where the $j = 0$ term of the sum equals 1 since the chain starts in the ending state 0. We can find a non-trivial solution if the sum converges, i.e. the system is transient if and only if

$$\sum_{j=0}^{\infty}\left(\frac{\mu}{\lambda}\right)^j < \infty \quad (28)$$

The sum converges if and only if $\mu < \lambda$. For the case of $\mu \geq \lambda$, the system is recurrent with recurrent state 0, and the data will disappear quickly (linear with n). Thus we only consider the case $\lambda > \mu$.

We note that in this case the mean absorption time is infinite since the Markov chain is transient. So a better metric is the probability that the number of packets in the queue will reach zero, given that there are n packets initially.

Since $a(n)$ is the probability of absorption, we look for a solution of the form $a(n) = \theta^n$, where $0 \leq \theta \leq 1$. Substitute this into Equation 24 we have:

$$\theta^n = q\theta^{n-1} + p\theta^{n+1} \quad (29)$$

where

$$q = \frac{\mu}{\lambda+\mu}$$

and

$$p = \frac{\lambda}{\lambda+\mu}$$

Cancel out the power θ^{n-1}, the Equation 29 now becomes

$$\theta = q + p\theta^2$$

which has 2 roots

$$\theta_1 = 1$$

and

$$\theta_2 = \frac{q}{p} = \frac{\mu}{\lambda} < 1$$

Thus

$$a(n) = c_1\theta_1^n + c_2\theta_2^n = c_1 + c_2\theta_2^n = c_1 + c_2\left(\frac{\mu}{\lambda}\right)^n \quad (30)$$

We can find the value of c_1 and c_2 based on two initial conditions:

$$1 = a(0) = c_1 + c_2$$

and

$$0 = a(N) = c_1 + c_2\left(\frac{\mu}{\lambda}\right)^N$$

where N is very large. Then,

$$c_1 = -\frac{(\frac{\mu}{\lambda})^N}{1 - (\frac{\mu}{\lambda})^N} \approx 0$$

$$c_2 = \frac{1}{1 - (\frac{\mu}{\lambda})^N} \approx 1$$

Therefore,

$$a(n) = \left(\frac{\mu}{\lambda}\right)^n \quad (31)$$

Note that this probability is exponentially small with n, the initial number of peers.

Now, consider a system of r queues, each queue contains n_k pieces of data type k, where $N = \sum_{k=1}^{r} n_k$ is the total number of data pieces stored in the system. Since all the queues are independent, the probability that a piece of data is not recoverable is:

$$P = \sum_{i=1}^{r}\binom{r}{i}\prod_{k=1}^{i}a(n_k) = \sum_{i=1}^{r}\binom{r}{i}\prod_{k=1}^{i}\left(\frac{\mu_k}{\lambda_k}\right)^{n_k} \quad (32)$$

IX. 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. We believe this technique is general as such, it can be applied to other problems such as gene population in which, it is intractable to directly model an exponentially large number of states using Markov chain representations. For example, one simple direct application of our time-backward technique is to model the diversity of a gene pool after several generations of inbreeding and without mutation. Each gene then can be considered as a codeword. One would expect that after many generations of inbreeding, the gene population would then become homogeneous. Equivalently viewed, many codewords are linear combinations of a few codewords. One can also extend the model to include other factors, e.g, using other distributions other than the uniform distribution for selecting the departed nodes.

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] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM*, September 2001.

[3] 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.

[4] D.G. Andersen, *Resilient overlay networks, Master Thesis*, Massachusetts Institute of Technology, 2001.

[5] http://www.bittorrents.com.

[6] http://www.kazaa.com.

[7] 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.

[8] A. Rowstron, A-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in *NGC*, November 2001.

[9] A. Gupta, D. Agrawal, and A. Abbadi, "Approximate range selection queries in peer-to-peer systems," in *VLDB Conference on Innovative Data Research*, 2003.

[10] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne, "A peer-to-peer framework for caching range queries," in *IEEE International Symposium on High-Performance Distributed Computing*, June 2003.

[11] C. Schmidt and M. Parashar, "Flexible information discovery in decentralized distributed systems," in *IEEE International Symposium on High-Performance Distributed Computing*, June 2003.

[12] O. Sahin, A. Gupta, D. Agrawal, and A. Abbadi, "A peer-to-peer framework for caching range queries," in *IEEE International Conference on Data Engineering (ICDE)*, 2004.

[13] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weahterspoon, and J. Kubiatowicz, "Maintenance-free global data storage," *IEEE Internet Computing*, 2001.

[14] 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.

[15] Kien Nguyen, Thinh Nguyen, and Y. Kovchegov, "A p2p video delivery network (p2p-vdn)," in *International Conference on Computer and Communication Networks, 2009*, 2009, pp. 1 – 7.

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

[17] 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.

[18] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Transactions on Information Theory*, June 2006.

[19] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *submitted for publication*.

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

[21] 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.

[22] Yunnan Wu and Alexandros G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 4*, Piscataway, NJ, USA, 2009, ISIT'09, pp. 2276–2280, IEEE Press.

[23] Yunnan Wu, Ros Dimakis, and Kannan Ramch, "Deterministic regenerating codes for distributed storage," in *in Allerton Conference on Control, Computing, and Communication, (Urbana-Champaign, IL*, 2007.

[24] Yunnan Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J.Sel. A. Commun.*, vol. 28, pp. 277–288, February 2010.

[25] 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.

[26] 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.

[27] Sameer Pawar, Salim Y. El Rouayheb, and Kannan Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *CoRR*, vol. abs/1009.2556, 2010.

[28] S. Lin and D. Costello, *Error Control Coding (2nd Edition)*, Prentice Hall, 2004.

[29] S. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, 1994.

[30] 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.