Periodica Polytechnica
Electrical Engineering
and Computer Science

61(1), pp. 12-21, 2017
DOI: 10.3311/PPee.9377
Creative Commons Attribution

RESEARCH ARTICLE

On the Effectiveness of Recoding-based Repair in Network Coded Distributed Storage

Márton Sipos[1,2*], Patrik János Braun[1], Daniel Enrique Lucani[2,4], Frank H. P. Fitzek[3], Hassan Charaf[1]

Received 25 April 2016; accepted 06 September 2016

Abstract
High capacity storage systems distribute files across several storage devices (nodes) and apply an erasure code to meet availability and reliability requirements. Since devices can lose network connectivity or fail permanently, a dynamic repair mechanism must be put in place. In such cases a new recovery node gets connected to a given subset of the operating nodes and receives a part of the stored data.

The objective of this paper is to investigate data survival for Random Linear Network Coding (RLNC) as a function of topology and communication overhead, defined by the number of connections and the number of transmitted packets to the recovery node, respectively. The paper includes two main contributions. First, a sufficient set of conditions for quasi-infinite longevity of the stored data is derived. Second, a comparison using experimental results shows that RLNC can be up to 50% more effective than traditional erasure codes like Reed-Solomon.

Keywords
distributed storage, erasure coding, network coding

[1] Department of Automation and Applied Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics

[2] Department of Electronic Systems, Faculty of Engineering and Science, Aalborg University

[3] Faculty of Electrical Engineering and Information Technology, Dresden University of Technology

[4] Chocolate Cloud ApS

[*] Corresponding author, e-mail: siposm@aut.bme.hu

1 Introduction and related work

Reliable distributed storage has been one of the driving forces behind most online services in the last decade. It has also played a key role in the creation of entire new fields such as cloud computing and big data. Many traditional distributed storage systems that are employed in controlled, observable and predictable scenarios, use replication. For example, the widely used Apache Hadoop File System (HDFS) uses 3-way replication by default. However, there has been a long-running trend towards using erasure codes to reduce the storage cost in exchange for computational overhead. Windows Azure Storage was one of the first large services to make use of erasure codes, namely a (6,2,2) Local Reconstruction Code [1] that employed 2 local parity fragments and 2 global parity fragments for every 6 data fragments. Facebook employed an extended version of HDFS-RAID [2] that introduces Locally Repairable Codes for storing rarely accessed cold data. Depending on file size, this meant either a Reed-Solomon(10,4) code, or a simpler XOR-based parity code. An evolution of this called HDFS-Xorbas [3] was also considered. Google has stated [4] that Colossus, the successor to the Google File System [5] will also make use of a Reed-Solomon code.

On the other hand, in distributed storage systems that lack a central entity to direct the repair process and for which the exact system state is hard to observe and predict, these traditional codes have proved less effective [6]. These include general P2P storage such as mobile and vehicular storage clouds and sensor networks. These systems also behave in a more dynamic way, nodes leave and join the system regularly, therefore it is crucial to limit the transmission cost associated with maintaining data integrity. In this paper we advocate the use of random linear network coding, which is better suited for this dynamic scenario.

Network coding was first introduced in 2000 by Ahlswede et al. [7] as a way of improving the throughput of packet switched networks. Random Linear Network Coding (RLNC) has been shown to be very effective as an erasure code for distributed storage [8]. Furthermore, our previous work [6, 9] has shown using simulation data, that it can outperform replication-based storage and Reed-Solomon codes even in traditional centrally controlled systems if the amount of storage and repair traffic is limited. RLNC is effective in maintaining data survival because of recoding-based repair, a form of functional repair. There is no need for the recovery node to gather the entire surviving data and there is no signaling taking place among nodes to describe the state of the system or the distribution of data or to coordinate the repair process. This paper sets out to understand and formalize the effectiveness of the recoding process and show the values for which recoding provides data survival for a large number of subsequent non-concurrent failures.

It is motivated in part by the seminal work of Dimakis et al. [10] which shows that there is an inherent trade-off between the amount of data stored on each node and the amount that needs to be transmitted to a new node during reconstruction. The paper defines bounds on erasure codes using a curve to express this trade-off and discusses the two extremal points on it. Minimum Storage Regenerating (MSR) codes need the least amount of storage to ensure a given level of reliability and are therefore equivalent with Maximum Distance Separable (MDS) codes. Conversely, Minimum Bandwidth Regenerating (MBR) codes store more information in order to decrease the amount of information transmitted during reconstruction to a minimum. [10] also introduces the concept of regenerating codes, which are optimal in terms of this trade-off and proves the existence of such codes based on the existence of a minimum cut in the information flow graph for every point on the curve. Both extremal points have seen great interest since. [11] studies MBR exact repair codes with the repair-by-transfer property and assumes that all surviving nodes partake in the reconstruction of the lost data. It establishes the non-achievability of most of the interior points on the curve for exact repair. [12] proposes a scheme to create MSR codes which are also optimal in terms of the number of I/O operations performed on each of the nodes participating in the reconstruction.

One of the assumptions [10] makes is that both the amount of stored information (α) and transmitted (β and γ respectively) information can have non-negative, real values. In real systems erasure codes are only able to store and transmit an integer number of bits and, most likely, are bound by other system requirements to manage and store larger data portions. A probabilistic approach could be introduced to model the storage and transmission of real number of bits. For example, $\alpha = 2.3$ could be seen as each node storing 2 with 0.7 probability and 3 with 0.3 probability. However, it is unclear how such an approach would affect the aforementioned bounds. Furthermore, increasing the granularity with which data is viewed in a system is generally not feasible in practice due to a cubic increase in the complexity of encoding and decoding operations. Therefore, we have chosen to introduce a more strict condition that these values must instead be positive integers. Thus, the bounds that we introduce are at least as constraining as those in the original work.

Beyond this practical consideration, our experimental results go a step further by also restricting the number of nodes that are available to use during the reconstruction, but without explicitly designing the system, as done in [10], to compensate for it. This means that we do not consider the same guarantees of reliability on each recovery, but can still provide reliability over an arbitrarily large number of loss/recovery operations. This additional constraint can be used to model storage nodes that are unavailable temporarily due to network connectivity or are unable to respond in due time due to uneven, dynamically changing loads in the system. This is also a novel constraint absent in [10].

The paper is organized as follows. First, we propose a model in Section 2 and show in Section 3 that if the values of certain parameters that define the storage system are chosen correctly, data integrity can be guaranteed for a large number of rounds. We define the constraints to choosing the appropriate values in Section 4. Finally, we show the most cost-effective sets of values and compare RLNC with other erasure codes using results from simulations in Section 5.

2 System model

This section introduces the elements of a distributed storage system and proposes a simple and effective way to model the reconstruction process for RLNC encoded files based on an information flow graph [7]. Each file is divided into g packets of identical size and distributed evenly to n storage devices using a rateless RLNC code. Devices are prone to failure and any data that is stored on a failed device is lost and must be reconstructed from the surviving ones onto a new recovery device. Recoding is performed on both surviving devices taking part in the recovery process and the recovery device itself using all available packets. We assume that the reconstruction process always completes successfully before another failure occurs. We have previously looked into how this assumption affects the system in [13] using real-world traces, as well as how effective erasure codes are when dealing with concurrent device failures [9]. We assume that all devices have the same probability to fail, regardless of their age in the system. We evaluate the ability of a system to store data reliably by considering k rounds of failure and reconstruction pairs representing the transitions between the different states of the system. We wish to establish the sets of parameters for which a storage system that uses recoding is able to maintain data integrity after a large number of failure and recovery rounds.

2.1 Recoding

Recoding is the central part of the repair process for RLNC. It is the creation of packets by linearly combining existing encoded ones. If a storage device has stored q linearly independent packets, it can create q linearly independent recoded packets from these. RLNC is a rateless code, therefore, the number of packets that can be generated using recoding is only limited in theory by the field size used for calculation. However, from any set of generated packets for a given file, at most g will be linearly independent.

$$R_j = \sum_{i=1}^{g} \alpha_{ij} P_i \tag{1}$$

R_j is a recoded packet, P_i is a previously encoded/recoded packet and α_{ij} is a randomly selected coefficient. When working over a small finite field, there is a possibility that the values for the α_{ij} are selected in such a way that the q recoded packets are not linearly independent. The probability for this to happen is reduced significantly by selecting a larger field [14]. Our model assumes the use of a high field, where recoding does not introduce significant linear dependence. Experimental results presented in Subsection 5.2 use the relatively small GF(2^8) field and suggest that the probability of generating linearly dependent coefficients is low enough to not impact the effectiveness of RLNC compared to other erasure codes. The use of a larger field such as GF(2^{16}) should make this issue negligible in practice. To verify this assumption, we have performed and described further experiments in Subsection 5.3. Recently, Abdrashitov et al. [15] investigated the long-term behavior of a distributed storage system using a model very similar to the one used in this paper. One of their main results was to define approximations on data survival based on Markov chains. These theoretic results are consistent with our simulation-based findings.

A different approach can be used if a guarantee is necessary that a distributed storage system employing recoding ensures data recoverability indefinitely with no constraint on the field size used. By checking the invertability of several possible coefficient matrices used to encode the data prior to performing a repair, it is possible to select a combination of data and coefficients that is able to maintain the amount of linear independence in certain parts of the system. However, the practical applicability of this approach may be limited for highly dynamic, decentralized systems due to its high computational cost as well as the need for a mechanism to communicate the coefficients to the node that performs the checks and repair selection.

2.2 Modeling distributed storage using network flows

To describe and analyze the way packets are stored and used for repair, we employ a model based on network flows, similar to that proposed in [10]. In this interpretation, the storage system behaves similarly to a multi-hop unicast lossy wireless network. The maximum number of packets that can be transmitted between the source and the sink in such setups has been proven to be the network's maximum flow, achievable using network coding [7]. The nodes of the network are the states of the storage devices in each round.

Node N_i^j denotes storage device i in round j (where $i = 1,2,\ldots n$ and $j = 1,2,\ldots k$, where $n,k \in \mathbb{N}^+ = \{1,2,\cdots\}$). Each node stores q packets and is able to perform recoding on these to generate new packets. We denote the set of nodes belonging to round j as $\Gamma^j = \left\{ N_i^j \middle| 1 \le i \le n \right\}$. We also introduce two special nodes: the data source N_s and the data collector or sink N_t. These appear before the first round and after the last round respectively. The data source splits a piece of data that needs to be stored into g pieces (packets) and encodes them before distributing them to the storage devices. A special N_{ENC} encoder and N_{DEC} decoder node is introduced to model the encoding and decoding process. Figure 1 shows an overview of the network.

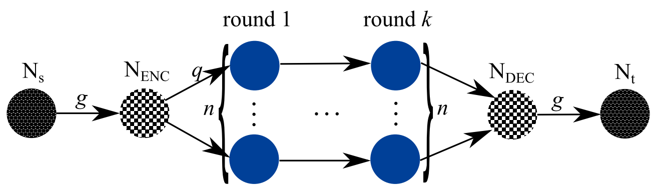


Fig. 1 Overview of the directed acyclic graph representing the storage system

An edge symbolizes the route taken by a packet between two states of the network. We distinguish between two cases: first, the storage node i that is present in the system in both round l and $l+1$ is denoted by two nodes N_i^l and N_i^{l+1} with q directed edges connecting them signifying the number of packets that are stored on it. Second, each new recovery node is filled with data by randomly selecting p parent nodes from the surviving ones and transferring c recoded packets from each. The recovery node will recode over the received $p \cdot c$ packets and store q out of them. The most significant difference between this model and the one introduced in [10] is that $q, c \in \mathbb{N}^+$ as opposed to $q, c \in \mathbb{R}_{\ge 0}$. We assume $p \cdot c \ge q$, as otherwise packets on a recovery storage device would become linearly dependent. In this case, edges between parents and a recovery node denote the transfer of packets used for the reconstruction of data. Formally, nodes N_a^l and N_b^{l+1}, where $a \ne b$, are connected by c directed edges if N_b^{l+1} is a new recovery node that has been reconstructed using c packets from N_a^l. No edges leave a node N_i^l that failed in round l, as we consider all packets it stored lost. Each edge has unit capacity as they denote the storage/transmission of a single packet in the network. In figures, we represent parallel edges with a single edge and a number to show their multiplicity. Due to the way the graph is constructed, edges only run between nodes of consecutive rounds. This structure gives a topological ordering for the underlying directed acyclic graph, where each level is comprised of nodes for a given loss-recovery round.

Edges in the network show the path packets take from the source to the sink. Each node N_i^l (not counting the source and the sink) receive $deg^-\left(N_i^l\right)$ packets on its incoming edges and recodes over them to send $deg^+\left(N_i^l\right)$ on its outgoing edges. Recovery nodes receive $deg^-\left(N_i^l\right) = p \cdot c$ packets. This is potentially more than the storage capacity q. To correctly reflect this constraint in the model, all reconstructed recovery devices need to be represented by two separate nodes N^l_{i-} and N^l_{i+} connected with q edges, Fig. 2 illustrates this. We have chosen against showing this detail in figures to make them easier to read. Note also that not just the recovery, but all devices may be represented in this manner.

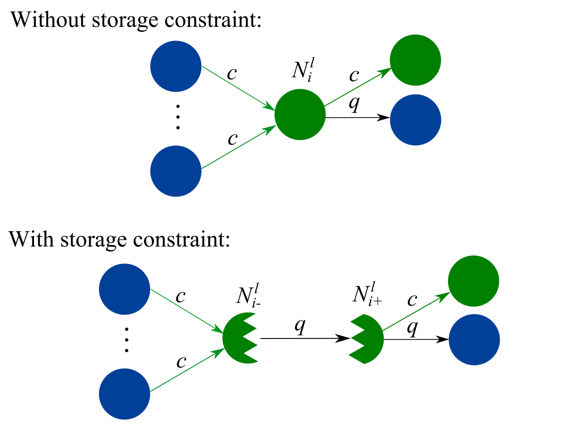


Fig. 2 Model for data flow with and without storage constraints

Remark 1. The edges used by a flow of capacity f between the source and the sink on a network of the aforementioned type can be used to transfer f linearly independent packets between the source and the sink.

Proof. The edges portray how linear combinations of packets are generated using recoding. In this sense, edges denote linear dependence between packets of two nodes. Therefore, if we find f edge-disjoint directed paths between the source and the sink, then these will enable the transmission of f linearly independent packets. As all edges have unit capacity by definition, f edge-disjoint paths can support a flow of capacity f (0-1 flow due to the integrality theorem) and as such, the transmission of f linearly independent packets between the source and the sink. □

Please note, that for all flows, $f \le g$ because of the minimum cut separating the N_s and N_{ENC} as visible on Fig. 1. To be able to perform an evaluation based on network flows for RLNC, we assume an infinitely large field in Section 3 and 4. We then evaluate how much the choice of field size affects data survival in practice in Section 5 to validate this approximation.

The choice to represent the distributed storage system as a network comes with additional benefits for the future. Concurrent storage device failures can be modeled with few changes. Furthermore, devices that are only temporarily unavailable between rounds a and b can also be portrayed by introducing edges between N^a_i and N^b_i. However, a small part of the results that are presented in this paper would need to be reevaluated, as this would mean that it would no longer be possible to state trivially that the structure of the underlying graph is topologically sorted.

3 System convergence

We wish to show that a storage system using RLNC can ensure data survival with a high probability after a large number of failures given a judicious selection of values for some key parameters. Using the previously introduced model, the criterion for this is to have a maximum flow with value $f = g$ (or g edge-disjoint paths) between the source and the sink. In other words, the data collector must be able to gather as many linearly independent packets from the surviving storage nodes as the number of packets that the source introduced into the network (g). First, the initial data distribution must be performed such that this property is satisfied. Second, subsequent node loss and recovery rounds must ensure that the property is kept for a large number of rounds. We denote $M^j \subseteq \Gamma^j$ as a randomly selected set of non-failing nodes in round j. It is the smallest possible set that stores g packets (to make possible a flow with value g between the source and the sink without the traversal of other nodes of round j).

$$m = \left|M^j\right| = \frac{g}{q} \tag{2}$$

Definition 2. Robust Data Recoverability (RDR) property: a system has the RDR property if and only if data can be recovered from any set M^j of non-failing nodes in any round j, where $\left|M^j\right| = \frac{g}{q}$.

For RLNC this is analogous to having a submatrix of rank g of the matrix composed of the coefficients used to (re)encode the data for all selections of nodes of size m in each round. The RDR property is a generalization of the Maximum Distance Separable (MDS) property. It illustrates the efficiency of a code in terms of the required storage for a given level of redundancy. We introduced RDR to handle cases where $q \neq 1$, common for RLNC. For $g \mid q$ the two concepts coincide.

To have this property, the initial distribution of data must meet some conditions. More importantly, the storage system must have a robust reconstruction transition between rounds, as there is no way to rebuild lost data paths. Newly introduced repair edges to the recovery device have the goal of increasing the interconnectedness of nodes. They build redundant paths to be used in case another node fails in the future.

The following proposition states that any given non-adaptive recovery mechanism either maintains the RDR property indefinitely or loses it after some rounds and never recovers it.

Proposition 3. Considering a storage network with a fixed set of values for g, n, q and a reconstruction transition with fixed values for c and p that maintains the RDR property between at least one pair of consequent rounds l and $l+1$, then this transition will also maintain the RDR property for any round $j \in N^+$.

Proof. We divide the proof into several parts:

- For $j < l$: Based on the network construction that provides a topologically sorted form, it is trivial that if layer l exhibits the RDR property then all layers $j < l$ must also have this property as the premise of the proposition can be applied recursively until round 1.
- For $j > l + 1$:
  Let us assume that round j is the first round after $l + 1$, for which there exists a selection of nodes M^j that do not have g edge-disjoint paths pass through them. We will show by contradiction that such a selection cannot be made and thus round j also has the RDR property.
  M^j can be selected in $\binom{n}{m}$ ways. Fortunately, it is enough to consider two distinct cases.
  - M^j does not include the newly recovered node.
    In this case, all nodes in M^j were already present in the previous round, therefore we can easily find the corresponding set of nodes M^{j-1} that includes the same storage nodes. These can support g edge-disjoint paths because round j is the first to not have the RDR property. The q paths between the m pairs of nodes in round M^{j-1} and M^j will ensure g edge-disjoint paths pass through M^j. We have arrived to a contradiction.
  - M^j includes the newly recovered node.
    Surviving nodes already ensure at least $g - q$ edge-disjoint paths pass through nodes in the corresponding M^{j-1} set in round $j - 1$. Let us assume that the recovered node does not provide the minimal number of q additional paths necessary for the system to keep the RDR property in round *j*. This implies that there are less then q edges between the recovery node and nodes from round $j - 1$ outside of M^{j-1}, i.e. $(p - (m - 1))c < q$. The original assumption of the proposition is that p, g, q, c and thus m have a fixed value. Therefore, this bound must also have had to have been in place in rounds l and $l + 1$. This would mean that the transition to round $l + 1$ would have lost the system the RDR property because the M^{l+1} set that included the node recovered in round $l + 1$ would not have had g edge-disjoint paths pass through it either. Again, we have arrived to a contradiction.

Because we arrived at contradictions for both categories of cases, we can conclude that the system must be able to support robust data recoverability in rounds following round $l + 1$ as well. □

For RLNC, the assumption of a large enough field size must be made so that g edge-disjoint paths correspond to the storage and transmission of g linearly independent packets with a high probability. Even so, the likelihood of maintaining the RDR property decreases as j increases.

Another way of phrasing the contents of this section is that the parameters of the system define the lowest value for the minimum cut in the network. After a sufficiently large number of recovery rounds, the system will converge to this value. This is the amount of data that can be stored safely in the distributed storage system. By maintaining the RDR property in all rounds using a given reconstruction transition, the system ensures that the minimum cut is above g. An important consequence of Proposition 3 is that the system can be considered memoryless if the RDR property is maintained because the ability of a reconstruction transition to maintain the RDR property is only influenced by the state of the system in the pre-transition round.

4 Criteria for maintaining the Robust Data Recoverability property

In the previous section, we have shown that given a correct set of values for the parameters of the system, the RDR property can be maintained for a large number of rounds with high probability. Here we give the criteria for the parameters as a set of inequalities.

Our model can be defined using the previously introduced parameters: $n, q, g, p, c \in \mathbb{N}^+$. We derive the key constraints for these by examining each state and state change of the system. From the initial state it is possible to conclude that to be able to store the data, we must have at least $n \geq \lceil \frac{g}{q} \rceil$ nodes. However, an extra node is required to be able to handle a loss.

$$n \geq \left\lceil \frac{g}{q} \right\rceil + 1. \tag{3}$$

Because the original data is divided into g pieces, there is no reason to store more than $q \leq g$ packets on a single node (there is a g sized cut between the source and the rest of the network).

The defining state changes is the transition between any two consecutive rounds l and $l + 1$. A node fails in round l and a recovery node is filled with data in round $l + 1$ to functionally repair the lost data. To be able to contact enough parent nodes, the recovery node must have access to at least these p nodes. Therefore,

$$n \geq p + 1, \tag{4}$$

where the $+1$ is the recovery node in round $l + 1$. Furthermore, each node stores q packets, therefore it should receive at least that many to make the recoding of q linearly independent packets possible

$$q \leq pc. \tag{5}$$

Let us look at how to ensure that the system maintains the RDR property in round $l + 1$, i.e. there exists a network flow with value $f = g$ between the source and the sink that only traverses nodes from a selected M^{l+1}. As detailed in Subsection 2.2, this is the same as having g linearly independent packets transferred on g edge-disjoint paths. We assume that this property holds for round l and let us examine the transition to $l+1$. We denote the set of nodes representing the surviving devices from round l which are also elements of M^{l+1} with $M^{l'}$ as illustrated on Fig. 3.

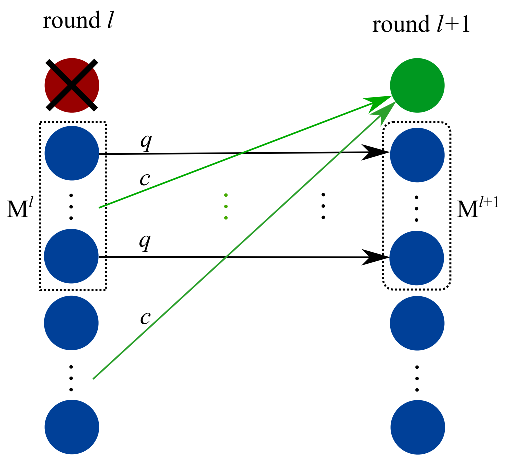


Fig. 3 Case I. M^{l+1} does not include the newly joined recovery node.

We can choose m nodes out of n in $\binom{n}{m}$ ways and the parent nodes for the recovery node can be chosen in $\binom{n-1}{p}$ ways. However, there are only three distinct cases to analyze (case III. is only achievable for storage systems with large numbers of nodes):

- Case I. M^{l+1} does not include the newly joined recovery node.
  In this case, there are no additional constraints on any of the parameters, as the property will hold true regardless of which nodes were used to fill the recovery node. This is because M^{l+1} includes the same devices as $M^{l'}$ for which the property was true.
- Case II. M^{l+1} includes the newly joined node, which was repaired using all $m-1$ nodes in $M^{l'}$ except itself, i.e. $p \geq m-1$.
  Each of these parent nodes will supply q edge-disjoint paths. To be able to have a total of g edge-disjoint paths crossing M^{l+1}, additional paths must traverse the recovery node. However, these should not be the same paths as supplied by the parents, as those would not necessarily be edge-disjoint. This can be achieved using additional parents (p_{add}), which results in

$$p_{add} = \left\lceil \frac{g-(m-1)q}{c} \right\rceil = \left\lceil \frac{g-\left(\left\lceil \frac{g}{q} \right\rceil -1\right)q}{c} \right\rceil \qquad (6)$$

  Considering the additional parents from inequality (6) results in

$$p \geq m-1+p_{add} = \left\lceil \frac{g}{q} \right\rceil -1+\left\lceil \frac{g-\left(\left\lceil \frac{g}{q} \right\rceil -1\right)q}{c} \right\rceil \qquad (7)$$

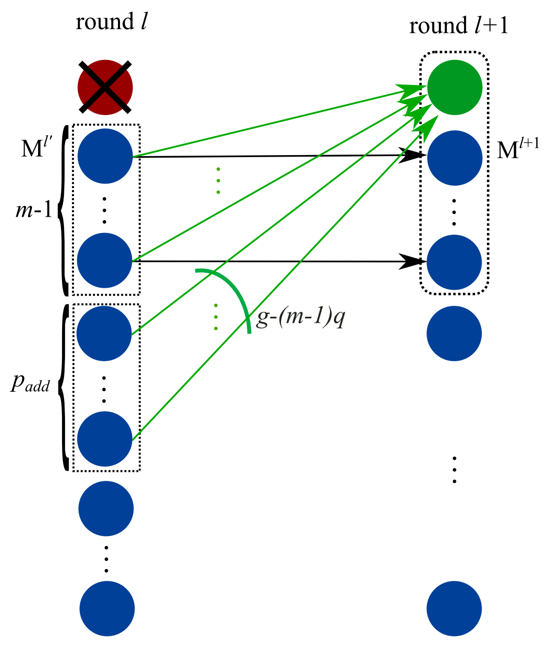


Fig. 4 Case II. M^{l+1} includes the newly joined node, which was repaired using all $m-1$ nodes in $M^{l'}$ except itself: $p \geq m-1$.

- Case III. M^{l+1} includes the newly joined node, which was repaired using $j = 0, \cdots, m-2$ nodes from M^{l}. A total of jc paths are created between $M^{l'}$ and the newly joined node. With reasoning similar to II., the number of additional paths that must be created between parents not in $M^{l'}$ and the recovery node is the same as described in Eq. (6).
  This gives the following lower bound for the number of parents:

$$p \geq j+p_{add} = j+\left\lceil \frac{g-\left(\left\lceil \frac{g}{q} \right\rceil -1\right)q}{c} \right\rceil \qquad (8)$$

  This is a less strict condition for p than Inequality (7) because $j < m-1 = \left\lceil \frac{g}{q} \right\rceil -1$ by definition. Furthermore, this case is only possible if: $n \geq j+p+1$, which is a more strict condition for n then the one in Inequality (4). It would also be possible to extend III. to include $j = m-1$ and thus be a generalization of II.

Having studied all types of transitions from round l to $l+1$, we have identified the sufficient conditions to ensure that any randomly chosen M^{l+1} surviving nodes can be used to create a flow with value g in round $l+1$ if the property was true for round l. A constraint on the number of nodes can also be expressed using Inequalities (4) and (7).

$$n \geq \left\lceil \frac{g}{q} \right\rceil + \left\lceil \frac{g-\left(\left\lceil \frac{g}{q} \right\rceil -1\right)q}{c} \right\rceil \qquad \forall g,q,c \in \mathbb{N}^{+} \qquad (9)$$

Our initial expectation is that these or a subset of these also defines the necessary conditions. We plan to investigate this in the future.

Finally, the last state change is the data retrieval itself. However, having already established the constraints to allow the recovery of the data using any m nodes for the repair transition, no new constraints need to be added.

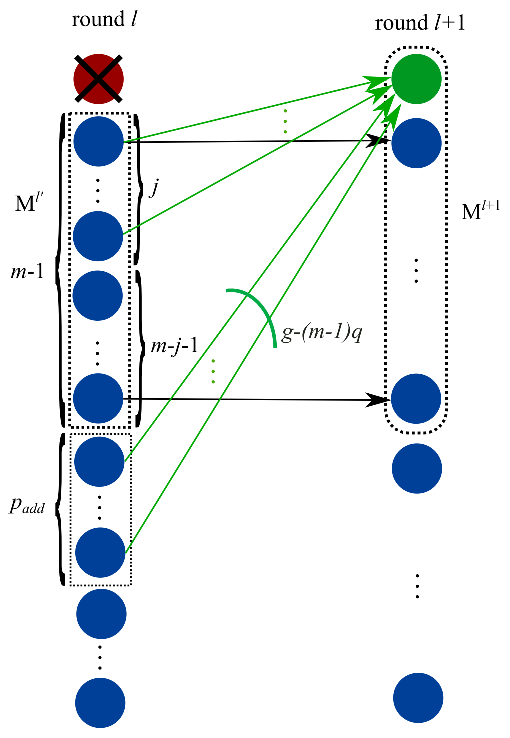


Fig. 5 Case III. M^{l+1} includes the newly joined node, which was repaired using $j = 0, \cdots, m-2$ nodes from $M^{l'}$.

5 Discussion

5.1 Constraints

In this section we evaluate a storage system with parameters that satisfy all previously presented constraints. First, we examine the number of storage devices n_{suf} that are sufficient to maintain the RDR property between subsequent rounds. We derive these results from Eqs. (3) and (9). Figure 6 shows that by increasing the storage space q on each device, the amount of devices needed initially declines. The reason behind this reduction is that the required edge-disjoint paths used by the repair process can be provided by fewer parent nodes as expressed in Inequality (7). However, n_{suf} increases after a point, as the amount of parent nodes required to fill the recovery node increases with q as expressed by Inequality (5). The relationship between the parameters of the system shows similar trends for both $g = 10$ and $g = 20$, as well as other values we have examined but not included in the paper.

Figure 7 shows the same relationship for a wider range of values for repair traffic (c). A similar trend can be observed as in Fig. 6 for values to the right of the $c = q$ plane. Values to the left of the $c = q$ plane are for cases where $c > q$. Clearly, such systems have no advantage compared to systems where $c = q$, as recoding cannot produce more linearly independent packets to send to the recovery device than was stored on the parent device.

The set of constraints offers a wide range of values for the parameters for which data integrity is guaranteed. This is due to the effectiveness of recoding in the reconstruction process and makes RLNC-based systems cost-effective.

5.2 Comparison with other erasure codes

We have also included a short comparison with a repetition-based code and Reed-Solomon with parameters nq and $nq - g$ to showcase how much more effective RLNC is in maintaining data retrievable for a wide range of parameters. Results are presented for both a centrally controlled system and a fully decentralized one that has no cooperation among nodes during the repair process. Figure 8 [9] shows the system after 1000 rounds of simulation. Operations were performed over GF(2^8), the number of nodes was fixed at $n = 15$, $g = 15$, $c = q$. The vertical axis shows the statistically established probability that data availability is maintained. The horizontal axes show the constraints in terms of the amount of parent nodes can connect to and the amount of storage available per node.

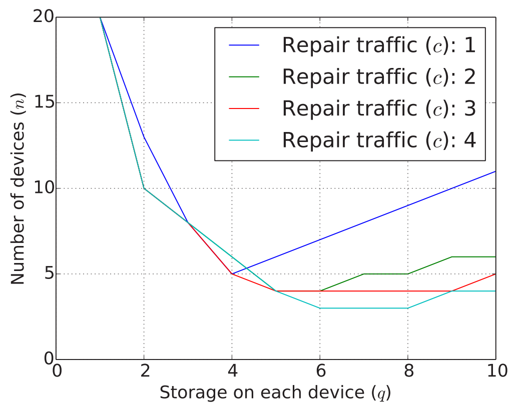
Simple replication-based storage has the benefit of fast retrieval performance. However, it is difficult to guarantee data survival if limits are placed on the amount of storage on each node and the number of nodes involved in the repair process. Data is lost as soon as every replica of any single piece is lost. Data survival can therefore only be guaranteed for very constrained cases. We have only included the results for the centrally controlled version, as the decentralized one basically guarantees that data is lost regardless of the parameters of the system.

Reed-Solomon codes are Maximum Distance Separable, making them a more effective solution in this case. Data is lost as soon as the number of distinct pieces falls below g. The main shortcoming of Reed-Solomon in traffic-bound scenarios is the need to gather enough packets to decode (g) before creating new repair packets. If this cannot be ensured, the best repair policy is to fall back to replicating the least common packets. The centrally controlled version is significantly more effective as it is able to reconstruct the rarest pieces in the system.

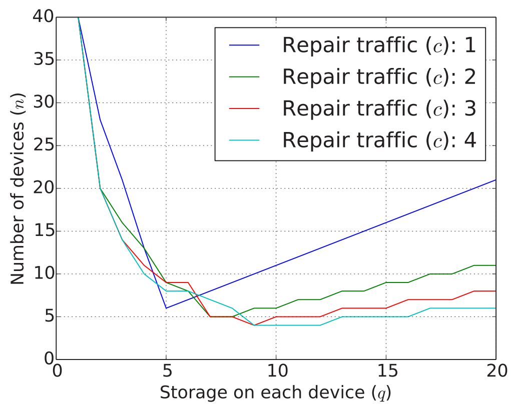
RLNC is able to maintain data integrity for a very wide range of parameters, making it a cost-effective solution in traffic-bound dynamic storage systems. Values of $p > 3$ and $q > 2$ practically guarantee that data remains recoverable. This means as much as 50% less storage and 50% less network traffic compared to the second best centrally controlled Reed-Solomon, as it requires $q > 4$ and $p > 6$.

5.3 The impact of field size on the effectiveness of recoding

To verify the assumption we made based on prior work [14] that recoding does not introduce a significant amount of linear dependence for RLNC given a large enough field for calculations, we have conducted further simulations with different fields. We chose four fields of practical interest. GF(2) allows

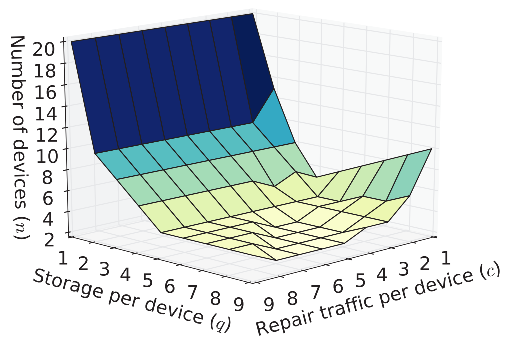


(a) $g = 10$

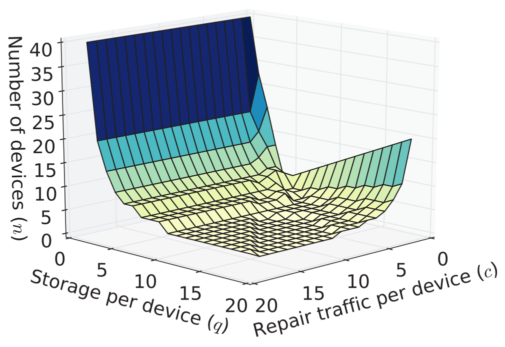


(b) $g = 20$

Fig. 6 The minimum number of storage devices needed to safely store data, with constraints on per device storage (q) and repair trac provided (c) for selected values.

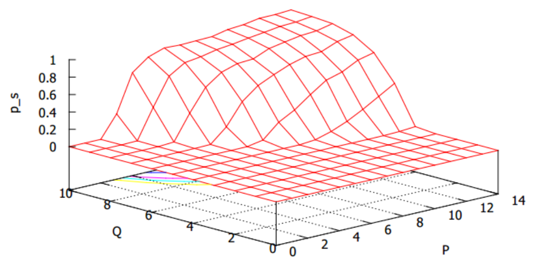


(a) $g = 10$

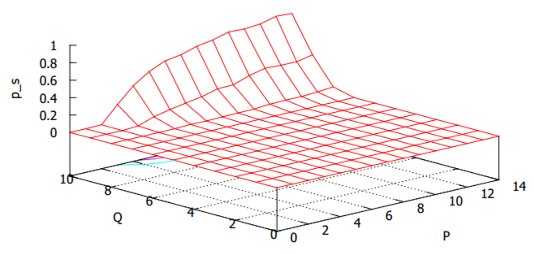


(b) $g = 20$

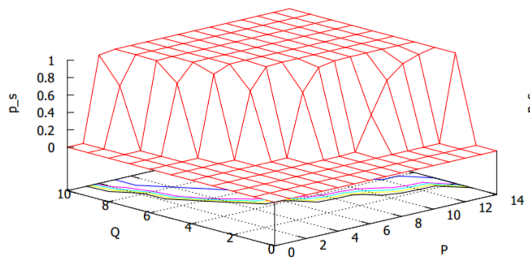
Fig. 7 The minimum number of storage devices needed to safely store data, with constraints on per device storage (q) and repair traffic provided (c) for a wider range of values



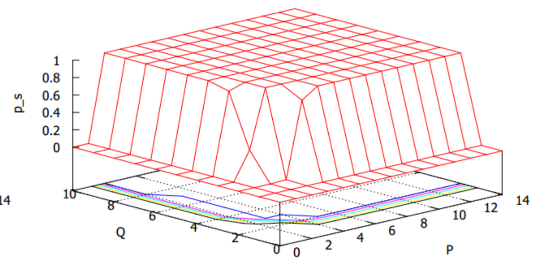
(a) Centrally controlled replication



(b) Decentralized Reed-Solomon

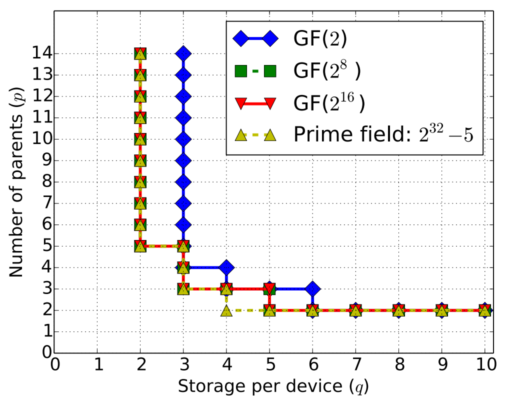


(c) Centrally controlled Reed-Solomon

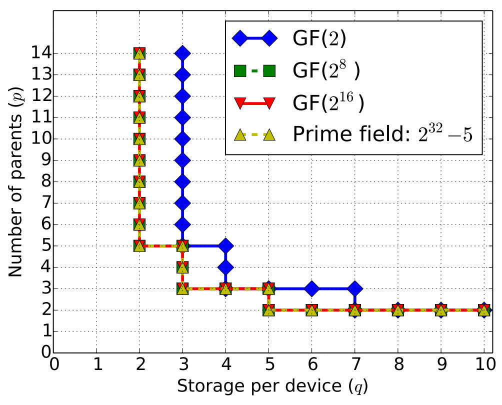


(d) RLNC ($c = q$)

Fig. 8 Success probability p_s to retrieve the stored data successfully in relation to the number of parents (P) and the amount of storage (Q) [9]



(a) $p_s = 0.95$



(b) $p_s = 0.99$

Fig. 9 Impact of field size on RLNC: the minimal values for p and q for which data survival is ensured with a given probability p_s. Multiple fields are illustrated over which operations are performed.

for very efficient computations due to the addition operation corresponding to the logical XOR operation and multiplication to logical AND respectively. Both use a small number of cycles on modern CPUs. GF(2^8) is a natural choice for use in erasure codes as its elements can be represented in a byte on most modern hardware architectures. The relative large size of the field is sufficient for use with non-deterministic coding schemes yet small enough that a multiplication and addition table can fit in memory for many systems. GF(2^{16}) has significantly more elements but is slower in terms of encoding and decoding performance. Finally, we have included the field based on the prime number $2^{32}-5$. It has the benefit of having a very large number of elements as well as relatively low encoding and decoding complexity because addition and multiplication can be performed using efficient integer addition, multiplication and the modulo operation. However, it requires some additional steps to map data prior to encoding, decoding [16, 17].

As in the previous subsection, we fixed the values of the following parameters $n = 15$, $g = 15$, $c = q$ and explored different values for q and p. As before, we looked at the success probability p_s of being able to recover the data after 1000 rounds of simulation based on 100 samples.

Figure 9 shows a top-down view of the results for RLNC from Fig. 8. It illustrates the smallest values of q and p for which data survival is ensured with probability p_s. GF(2) performs worse then the larger fields, however the difference is relatively small. The larger fields perform almost identically, with the prime field only surpassing GF(2^8) and GF(2^{16}) in a single case for $p_s \geq 0.95$. There is no noticeable difference between the three larger fields for $p_s = 0.95$. Our previous work [6] also looks at this for a fixed value of q and observes the same trend. These results confirm that GF(2^8) is sufficient to ensure data survival for RLNC with high probability over a large number of failure and recovery rounds.

6 Conclusion

Previous experimental results have suggested that RLNC is an effective solution, particularly suited to application in fully decentralized systems. Data integrity can be maintained with up to 50% less storage and 50% less repair traffic compared to a centrally controlled Reed-Solomon code. We have shown in this paper the reason for RLNC's effectiveness and given a set of conditions, that if met, ensure data survival with high probability for a large number of subsequent storage device failures. The main contribution of this paper is that the analytical results it presents can be used almost directly to create cost-effective, reliable storage systems.

Acknowledgment

The authors would like to thank Professor János Levendovszky for his valuable help in formalizing the concepts presented in the paper as well as proof-reading.

This work was partly supported by desk.io GmbH and partially financed by the Green Mobile Cloud project granted by the Danish Council for Independent Research (Grant No. DFF – 0602–01372B), the European Union and the European Social Fund through FuturICT.hu (Grant no. TAMOP–4.2.2.C–11/1/KONV–2012–0013) organized by VIKING Zrt. Balatonfüred, and the Hungarian Government, managed by the National Development Agency, financed by the Research and Technology Innovation Fund (grant no.: KMR_12-1-2012-0441).

Márton Sipos gratefully acknowledges the financial support he received as a recipient of the Csibi Sándor scholarship.

References

[1] Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J., Yekhanin, S. "Erasure Coding in Windows Azure Storage." In:Proceedings of the 2012 USENIX Conference on Annual Technical Conference, USENIX ATC'12, 2012, Boston, MA p. 2 URL: http://dl.acm.org/citation.cfm?id=2342821.2342823; USENIX Association, Berkeley, CA, USA.

[2] Weiyan Wang, H. K. "Saving capacity with HDFS RAID." 2014. [Online]. Available from: https://code.facebook.com/posts/536638663113101/saving-capacity-with-hdfs-raid. [Accessed: 21st November 2016]

[3] Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A. G., Vadali, R., Chen, S., Borthakur, D. "XORing elephants: novel erasure codes for big data." *Proceedings of the VLDB Endowment*. 6(5), pp. 325-336. 2013. https://doi.org/10.14778/2535573.2488339

[4] Google Faculty Summit AF. Storage Architecture and Challenges, 2015. [Online]. Available from: http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.reverse-proxy.org/en/us/university/relations/facultysummit2010/storage_architecture_and_challenges.pdf [Accessed: 21st November 2016]

[5] Ghemawat, S., Gobioff, H., Leung, S. T. "The Google File System." In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. SOSP '03. New York, NY, USA, ACM, Oct 19-22, 2003, pp. 29-43. https://doi.org/10.1145/945445.945450

[6] Fitzek, F., Toth, T., Szabados, A., Pedersen, M., Roetter, D., Sipos, M., Charaf, H., Medard, M. Implementation and Performance Evaluation of Distributed Cloud Storage Solutions using Random Linear Network Coding. In: 2014 IEEE International Conference on Communications Workshops (ICC), Sydney, NSW, June 10-14, 2014, pp. 249-254. https://doi.org/10.1109/ICCW.2014.6881204

[7] Ahlswede, R., Cai, N., Li, S.-Y. R., Yeung, R. W. "Network Information Flow." *IEEE Transactions on Information Theory*. 46(4), pp. 1204-1216. 2000. https://doi.org/10.1109/18.850663

[8] Acedaski, S., Deb, S., Medard, M., Koetter, R. "How good is random linear coding based distributed networked storage." In: 1st Workshop on Network Coding, Theory and Applications (NetCod), April 2005

[9] Sipos, M., Fitzek, F. H. P., Lucani, D. E. "Random Linear Network Coding is Key to Data Survival in Highly Dynamic Distributed Storage." In: 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, May 11-14, 2015, pp. 1-6. https://doi.org/10.1109/VTCSpring.2015.7146040

[10] Dimakis, A. G., Godfrey, P. B., Wainwright, M. J., Ramchandran, K. "Network Coding for Distributed Storage Systems." In: IEEE International Conference on Computer Communications (INFOCOM), Anchorage, AK, May 2007.

[11] Shah, N. B., Rashmi, K. V., Kumar, P. V., Ramchandran, K. "Distributed Storage Codes with Repair-by-Transfer and Non-achievability of Interior Points on the Storage-Bandwidth Tradeoff." *IEEE Transactions on Information Theory*. 58(3), pp. 1837-1852. 2012. https://doi.org/10.1109/TIT.2011.2173792

[12] Rashmi, K. V., Nakkiran, P., Wang, J., Shah, N. B., Ramchandran, K. "Having Your Cake and Eating It Too: Jointly Optimal Erasure Codes for I/O, Storage, and Network-bandwidth." In: 13th USENIX Conference on File and Storage Technologies (FAST 15), Santa Clara, CA, Feb. 2015. pp. 81-94. URL: https://www.usenix.org/conference/fast15/technical-sessions/presentation/rashmi

[13] Sipos, M., Fitzek, F., Lucani, D. "On the Feasibility of a Network Coded Mobile Storage Cloud." In: IEEE International Conference on Communications, London, June 8-12, 2015, pp. 466-471. https://doi.org/10.1109/ICC.2015.7248365

[14] Heide, J., Pedersen, M. V., Fitzek, F. H. P., Medard, M. "On Code Parameters and Coding Vector Representation for Practical RLNC." In: 2011 IEEE International Conference on Communications (ICC), Kyoto, July 29, 2011, pp. 1-5. https://doi.org/10.1109/icc.2011.5963013

[15] Abdrashitov, V., Médard, M. "Durable Network Coded Distributed Storage." In: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, 29 Sept.-2 Oct. 2015, pp. 851-856. https://doi.org/10.1109/ALLERTON.2015.7447095

[16] Crowley. P. "GF(232-5)." 2006. [Online]. Available from: http://www.lshift.net/blog/2006/11/29/gf232-5/. [Accessed: 21st November 2016]

[17] Pedersen, M. V., Heide, J., Vingelmann, P., Fitzek, F. H. P. "Network coding over the $2^{32} - 5$ prime field. In: 2013 IEEE International Conference on Communications (ICC), Budapest, 9-13 June, 2013, pp. 2922-2927. https://doi.org/10.1109/ICC.2013.6654986