Beyond Regenerating Codes

Anne-Marie Kermarrec, Nicolas Le Scouarnec, Gilles Straub

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Beyond Regenerating Codes

Anne-Marie Kermarrec — Nicolas Le Scouarnec — Gilles Straub

N° 7375

10 September 2010

Distributed Systems and Services

ISSN 0249-6399 ISRN INRIA/RR--7375--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Beyond Regenerating Codes

Anne-Marie Kermarrec*, Nicolas Le Scouarnec†, Gilles Straub†

Theme : Distributed Systems and Services
Networks, Systems and Services, Distributed Computing
Équipe-Projet Asap

Rapport de recherche n° 7375 — 10 September 2010 — 22 pages

Abstract: Erasure correcting codes are widely used to ensure data persistence in distributed storage systems. This paper addresses the repair of such codes in the presence of simultaneous failures. It is crucial to maintain the required redundancy over time to prevent permanent data losses. We go beyond existing work (i.e., regenerating codes by Dimakis *et al.*) and propose *coordinated regenerating codes* allowing devices to coordinate during simultaneous repairs thus reducing the costs further. We provide closed form expressions of the communication costs of our new codes depending on the number of live devices and the number of devices being repaired. We prove that deliberately delaying repairs does not bring additional gains in itself. This means that regenerating codes are optimal as long as each failure can be repaired before a second one occurs. Yet, when multiple failures are detected simultaneously, we prove that our coordinated regenerating codes are optimal and outperform uncoordinated repairs (with respect to communication and storage costs). Finally, we define *adaptive regenerating codes* that self-adapt to the system state and prove they are optimal.

Key-words: distributed storage, regenerating codes, multiple failures, self-healing, network coding

* INRIA Rennes - Bretagne Atlantique
† Technicolor

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Codes régénérants coordonnés et adaptatifs

Résumé : Les codes correcteurs d'effacements sont largement utilisés pour assurer la persistance des données dans les systèmes de stockage distribués. Ce rapport s'intéresse à la réparation de tels codes dans le cas de défaillances simultanées. Cette maintenance est cruciale afin de prévenir les pertes de données permanentes. Nous étendons les travaux existants (codes régénérants par Dimakis *et al.*) et proposons des *codes régénérants coordonnés* qui permettent aux éléments du systèmes de se coordonner durant les réparations de défaillances simultanées afin de réduire les coûts de réparation. Nous fournissons une forme close des coûts de communications de nos codes en fonction du nombre d'équipements vivants et du nombre d'équipements en cours de réparation. Nous prouvons, par ailleurs, que retarder les réparations de façon délibérée n'apporte pas de gains additionnels. Cela signifie que les codes régénérants sont optimaux tant qu'une première défaillance peut être réparée avant une seconde. Cependant, quand de multiples défaillances sont détectés simultanément, nous prouvons que nos codes régénérants coordonnés sont optimaux et dépasse les réparations non coordonnées (vis à vis des coûts de stockage et de réparation). Enfin, nous définissons des codes régénérants adaptatifs qui s'auto-adapte à l'état du système et prouvons qu'ils sont optimaux.

Mots-clés : stockage distribué, codes régénérants, défaillances multiples, auto-réparation, codage réseau

1 Introduction

Over the last decade, digital information to be stored, be it scientific data, photos, videos, etc, has grown exponentially. Meanwhile, the widespread access to the Internet has changed behaviors: users now expect reliable storage and seamless access to their data. The combination of these factors dramatically increases the demand for large-scale distributed storage systems. Such systems are used as back-ends by cloud service providers or as a basis for P2P systems to provide users with storage, backup or sharing capabilities. This is traditionally achieved by aggregating numerous physical devices to provide large and resilient storage [3, 5, 14, 23]. In such systems, which are prone to disk and network failures, redundancy is the natural solution to prevent permanent data losses. However, as failures occur, the level of redundancy decreases, potentially jeopardizing the ability to recover the original data. This requires the storage system to self-repair to go back to its healthy state (i.e., keep redundancy above a minimum level).

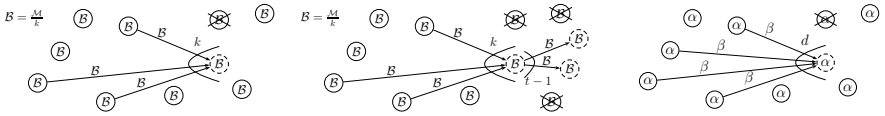
Repairing redundancy is of paramount importance for the design and implementation of distributed storage systems. A self-healing mechanism is usually composed of three phases. Firstly, the system must self-monitor to detect failures. Secondly, the system must trigger a repair on a set of spare devices. Finally, the system must regenerate the lost redundancy from the remaining one. In this paper, we focus on this last phase. Redundancy in storage systems has been extensively implemented using erasure correcting codes [19, 24, 27] because they enable tolerance to failures at a low storage overhead. In this context, the repair used to induce a large communication overhead, as it required to download and decode the whole file. Yet, Dimakis *et al.* recently showed [8, 9] that the repair cost can be significantly reduced by avoiding decoding in the so-called regenerating codes.

In this paper, we go beyond these works by considering simultaneous repairs in regenerating-like codes. We propose *coordinated regenerating codes* allowing devices to leverage simultaneous repairs: each of the t devices being repaired contacts d live (i.e., non-failed) devices and then coordinates with the $t-1$ others. Our contribution is threefold:

- As deliberately delaying repairs in erasure correcting codes leads to savings [3, 6, 7], it is natural to wonder if the same additional savings can be expected when delaying repairs for regenerating codes. By defining *coordinated regenerating codes*, we prove that, when relying on regenerating-like codes (MSR or MBR) [9], deliberately delaying repairs (so that $t > 1$) cannot lead to further savings.

- Yet in practical systems, it might be difficult to detect every single failure and fix it before a second one occurs (i.e., ensure $t = 1$). We establish the optimal quantities of information to be transferred when t devices must be repaired simultaneously from d live devices. Our *coordinated regenerating codes* consistently outperform existing approaches.

- In addition, most practical systems are highly dynamic. Therefore, assuming that t and d remain constant across repairs is unrealistic. To address this issue, we define *adaptive regenerating codes* achieving optimal repairs



(a) Erasure correcting codes (immediate repair) (b) Erasure correcting codes (delayed repair) (c) Regenerating codes

Figure 1: Repairing failures with codes. In a n device network, failed devices are replaced by new ones. The new devices get a given amount of data from live devices to repair the redundancy. In our examples, $k = 3$, $d = 4$, $\mathcal{B} = 1$, $\alpha = 1$, and $\beta = 1/2$. Some example of gains for $k = 32$ are given in Table 1

according to t and d. Under a constant system size $d + t$, their performance does not depend on t (i.e., the cost of a repair does not increase if the number of failed devices to repair increases).

Previous approaches are either known for not supporting simultaneous coordinated repairs [9] or known for assuming that repairing implies decoding which requires to download the whole file [3, 6, 7, 19, 27]. Hence, we define *coordinated regenerating codes* that fill the gap between the two aforementioned approaches by achieving simultaneous repairs without decoding (Figure 4). Since, for erasure correcting codes, simultaneous repairs reduce the communication overhead, we study the impact of deliberately delaying repairs with regenerating-like codes. Moreover, existing regenerating codes [9] assume a static setting: they require that d (the number of devices involved in the repairs) remains constant across repairs. Our *adaptive regenerating codes* consider a dynamic setting.

The paper is organized as follows. Section 2 describes the background on codes. Section 3 presents our *coordinated regenerating codes*. In this section, we prove the optimality of our code, we derive closed-form expressions for specific subsets of codes, and we show that deliberately delaying repairs does not reduce the costs further as long as each failure can be fixed before a second one occurs. In Section 4, we propose *adaptive regenerating codes* that self-adapt to the dynamic of the system and we prove their optimality.

2 Background

We consider a n device system storing a file of $\mathcal{M}$ bits split in k blocks of size $\mathcal{B} = \mathcal{M}/k$. To cope with device failures, blocks are stored with some redundancy so that a single failure cannot lead to permanent data losses. We consider code-based approaches to redundancy since they have been proved to be more efficient than replication with respect to both storage and repair costs [27]. We focus on self-healing systems (i.e., systems that automatically repair themselves upon failure). Distributed storage systems are required to be self-healing so as to ensure that the system does not gradually lose its ability to recover the initial file. Self-healing systems are equipped with a self-monitoring component that detects failed devices and triggers repairs [21] on new spare devices. To repair, the new spare devices regenerate the lost redundancy from data downloaded from live devices. The repair is constrained by the communication costs [4].

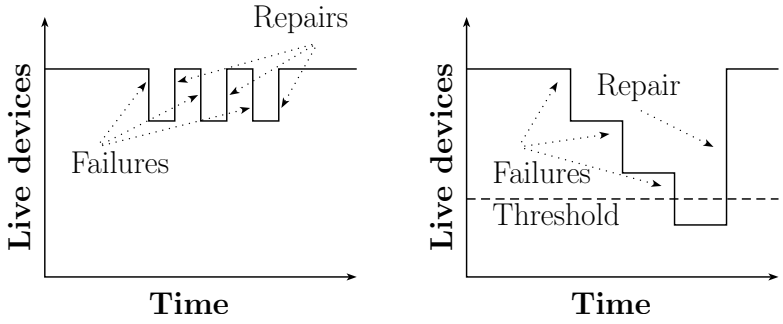
In the rest of this section, we describe the main code-based approaches for generating and repairing redundancy.

2.1 Erasure correcting codes (immediate/eager repairs)

Erasure correcting codes have been widely used to provide redundancy in distributed storage systems [19, 24, 27]. Devices store n encoded blocks of size $\mathcal{B}$, which are computed from the k original blocks. As erasure correcting codes are optimal with respect to recovery (i.e., they allow recovering the k original blocks from any subset of k encoded blocks), storing encoded blocks at $n = k + f$ devices is sufficient to tolerate f failures. This approach is efficient with respect to storage. Yet, repairing a lost encoded block is very expensive since the devices must fully decode the initial file. Hence, repairing a single lost block implies downloading k encoded blocks as shown on Figure 1a.

2.2 Erasure correcting codes (delayed/lazy repairs)

A first approach to limit the repair cost of erasure correcting codes is to delay repairs and factor downloading costs [3, 6, 7]. When a device has downloaded k blocks, it can produce as many new encoded blocks as wanted without any additional cost. Therefore, instead of performing a repair upon every single failure (Figure 2a), repairs are deliberately delayed until t (threshold) failures are detected (Figure 2b). This repair strategy is depicted on Figure 1b. One of the new devices downloads k blocks, regenerates t blocks and dispatches them to the $t - 1$ other spare new devices.



(a) Immediate repairs (b) Delayed repairs

Figure 2: Delaying repairs allows performing multiple repairs at once.

2.3 Network coding and regenerating codes

A second approach to increase the efficiency of repairs relies on network coding. Network coding differs from erasure correcting codes as it allows devices to generate new blocks with only partial knowledge (i.e., with less than $\mathcal{M}$ bits). Network coding was initially applied to multicast, for which it has been proved that linear codes achieve maxflow in a communication graph [1,17,18]. Network coding has latter been applied to distributed storage and data persistence [10,11, 16,20]. A key contribution in this area are regenerating codes [8,9] introduced by

Dimakis *et al.* They infer the minimum amount of information to be transferred to repair lost redundancy.

The idea behind regenerating codes [8] is that, when compared to erasure correcting codes, more devices are contacted upon repair but much less data is downloaded from them thus offering low repair cost for each single failure. Similarly to erasure correcting codes, n encoded blocks of size α bits are computed from the k original blocks. Each device stores $\alpha \approx \frac{\mathcal{M}}{k}$ bits. During a repair, the new device contacts $d > k$ other devices to get $\beta \ll \mathcal{M}/k$ bits from each and stores α bits as shown on Figure 1c.

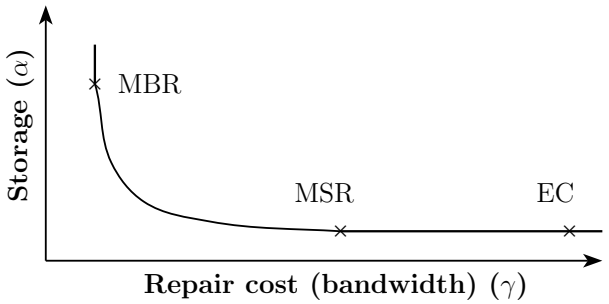


Figure 3: Regenerating codes (MSR or MBR) offer improved performances when compared to erasure correcting codes (EC)

Regenerating codes achieve an optimal trade-off between the storage α and the repair cost $\gamma = d\beta$. The graph on Figure 3 depicts the performance of the optimal regenerating codes. Points (α, γ) above the curve correspond to correct (but non-optimal) regenerating codes. Two specific regenerating codes are interesting: MSR (Minimum Storage Regenerating codes) offer optimal repair cost $\gamma = \frac{\mathcal{M}}{k}\frac{d}{d-k+1}$ for a minimum storage cost $\alpha = \frac{\mathcal{M}}{k}$, and MBR (Minimum Bandwidth Regenerating codes) offer optimal storage cost $\alpha = \frac{\mathcal{M}}{k}\frac{2}{2d-k+1}$ for a minimum repair cost $\gamma = \frac{\mathcal{M}}{k}\frac{2d}{2d-k+1}$. Regenerating codes can be implemented using linear codes [17, 18] be they random [13, 15] (i.e., random linear network codes), or deterministic [12, 22, 25, 26, 28]. Similarly to regenerating codes, our codes can be implemented using random linear network codes.

Table 1 gives some examples of storage cost α and repair cost γ for the codes we describe including the coordinated codes we propose. These costs depend on the number k of devices needed to recover the file, the number d of contacted live devices, and the number t of devices being repaired simultaneously. Regen-

Table 1: Some examples of repairs of codes for a file of 32 MB

	k	d	t	α	γ
Erasure codes	32	*NA*	*NA*	1 MB	32 MB
Erasure codes (delayed repair)	32	*NA*	4	1 MB	8.8 MB
Dimakis *et al.*'s MSR	32	36	*NA*	1 MB	7.2 MB
Dimakis *et al.*'s MBR	32	36	*NA*	1.6 MB	1.8 MB
Our MSCR (cf. Sec. 3.4.1)	32	36	4	1 MB	4.9 MB
Our MBCR (cf. Sec. 3.4.2)	32	36	4	1.5 MB	1.7 MB

erating codes by Dimakis *et al.* [9] represent a clear improvement over erasure correcting codes, and our coordinated regenerating codes allow reducing further the costs.

2.4 Design rationale

Regenerating codes transfer the minimal quantity of information needed to repair one device storing α bits. Dimakis *et al.* assume fully independent repairs: simultaneous failures are fixed independently. The cost of repairs increase linearly in t.

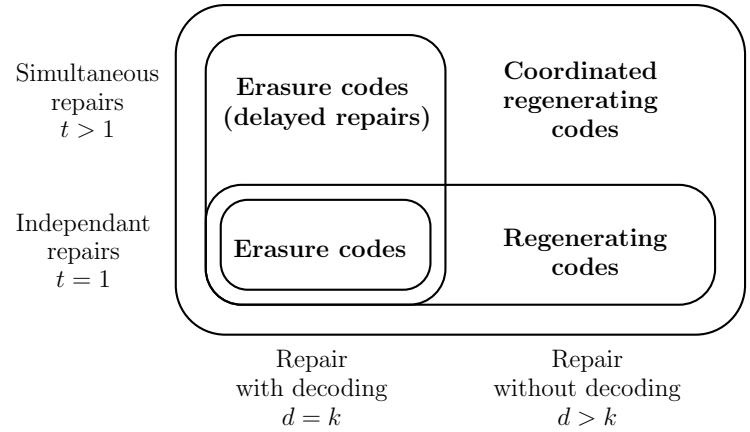


Figure 4: Our coordinated regenerating codes encompass all existing codes. Moreover, they allow the repair of multiple devices at once without decoding.

In this work, we investigate coordinately repairing simultaneous failures in an attempt to reduce the cost, along the lines of delayed erasure codes. Contrary to regenerating codes that repair only using data from live devices, our coordinated regenerating codes also allows the use of data from other devices being repaired. By coordinating the repair, we show that it is possible to repair multiple failures at once with an average cost of $\tilde{\gamma} < \gamma$. As depicted on Figure 4, our new codes encompass both erasure correcting codes ($d = k$) and regenerating codes ($t = 1$). In the next section, we detail our coordinated regenerating codes supporting coordinated repairs.

3 Coordinated regenerating codes

We consider a situation where t devices fail and, repairs are performed simultaneously. We assume that an underlying monitoring service triggers the repair and contacts all involved devices (i.e., the t spare devices that join the system to replace failed ones). Directly applying erasure correcting codes delayed repairs (Fig. 1b) to regenerating codes (Fig. 1c) is not appropriate. In short, the fact that all the data goes through the device that regenerates for others induce more network transfers than needed: as repairing does not require decoding, gathering all the information at a single device is not necessary.

3.1 Repair algorithm

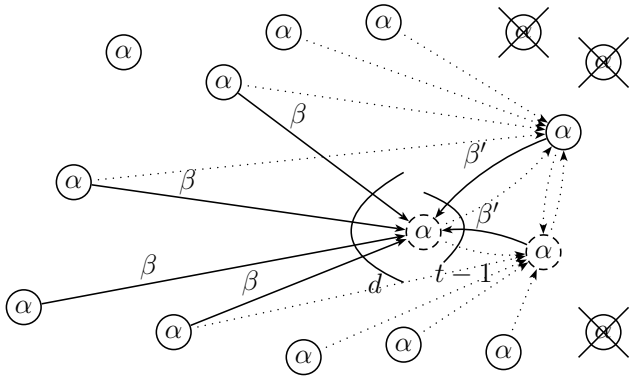


Figure 5: Repairing failures in coordinated regenerating codes. In a network of n devices storing α bits, when t devices have failed, t new devices collect β bits from d live devices. They coordinate by exchanging $(t-1)\beta'$ bits with other new devices and store α bits.

We introduce new *coordinated regenerating codes* allowing devices to repair simultaneously at the optimal cost (with respect to network communication). The repair is illustrated on Figures 5 and 7 showing the amounts of information transfered, and on Figure 6 showing the computations and exchanges of sub-blocks of data. A device being repaired performs the three following tasks, jointly with all other devices being repaired:

1. **Collect.** It downloads a set of sub-blocks (size β) from each of d live devices. The union of the sets is stored as W_1.

2. **Coordinate.** It uploads a set of sub-blocks (size β') to each of the $t-1$ other devices being repaired. These sets are computed from W_1. During this step, sub-blocks received from the $t-1$ other devices being repaired are stored as W_2. The data exchanged during this step can be considered as a digest of what each has received during the collecting step.

3. **Store.** It stores a set W_3 of sub-blocks (size α) computed from $W_1 \cup W_2$. W_1 and W_2 can be erased upon completion.

Interestingly, thanks to the explicit coordination step involving all devices, this approach evenly balances the load on all devices. Hence, coordinated regenerating codes avoid the bottleneck existing in erasure correcting codes delayed repairs (i.e., the device gathering all the information) (cf. Fig. 1b).

In the rest of this section, we present our main results: we investigate the optimal tradeoffs between storage and repairs costs (i.e., optimal values for α (data to store), β and β' (data to transfer)). As we consider the problem from an information theoretic point of view, we can ignore the nature of the information and only consider the amount of information that must be exchanged to repair the redundancy. Our results define the fundamental tradeoffs that can be achieved (i.e., lower bounds on amounts of information to be transfered to repair).

Our proof is inspired from the proof by Dimakis *et al.* found in [9]. We represent the system as an information flow graph. For we add a coordination

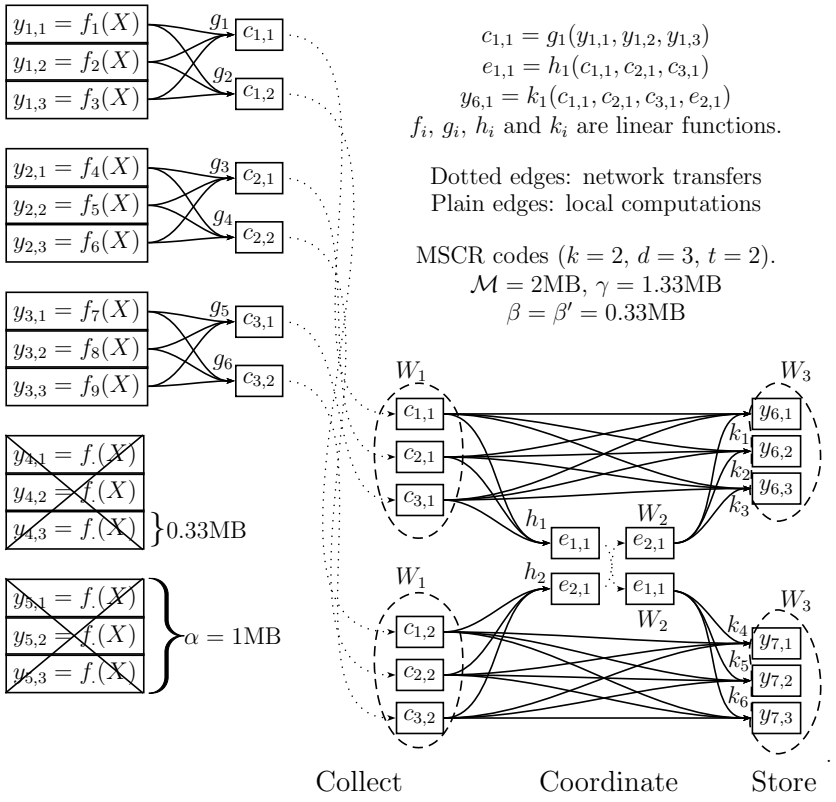


Figure 6: Coordinated regenerating codes based on linear codes. The system store a file X and is compound of 5 devices. Device i stores 3 sub-blocks $\{y_{i,1}, y_{i,3}, y_{i,3}\}$. Devices 4 and 5 fail and are replaced by devices 6 and 7.

Table 2: Notation used in Section 3

k	Constant (Integer)	Number of devices needed to recover
t	Constant (Integer)	Number of devices being repaired
d	Constant (Integer)	Number of live devices ($d \geq k$)
α	Variable (Real)	Quantity stored
β	Variable (Real)	Quantity transferred (collect)
β'	Variable (Real)	Quantity transferred (coordinate)
γ	Expression (Real)	Quantity transferred over the network

step, our graph differs from the one proposed in [9]. However, Lemmas 2 and 3 are identical to the ones proposed in [9] as they still apply to our information flow graph. When compared to [9], we allow the coordination of multiple repairs while they assume fully independant repairs.

We determine the optimal codes (i.e., we minimize the storage and the repair cost under some constraints obtained by studying information flow graphs). We give the expressions for optimal values of α (storage at each node), γ (repair cost), β (data transferred during collecting step) and β' (data transferred during coordinating step) as a function of d, k and t (parameters of the system). We also examine the influence of deliberately delaying repairs (i.e., increasing t while decreasing d). Our notations are summarized in Table 2.

3.2 Information flow graphs

Our study is based on information flow graphs similar to the ones defined in [9]. An information flow graph is a representation of a distributed storage system that describes how the information about the file stored is communicated through the network. The information flow graph $\mathcal{G}$ is a directed acyclic graph composed of a source S, intermediary nodes $x_{\text{in}}^{i,j}$, $x_{\text{coor}}^{i,j}$ and $x_{\text{out}}^{i,j}$, and data collectors DC_i (data collectors try to contact k devices to decode and recover the file). The capacities of the edges (α, β, β') correspond to the amount of information that can be stored or transferred.

In our approach, a real device $x^{i,j}$ is represented in the graph by 3 nodes ($x_{\text{in}}^{i,j}$, $x_{\text{coor}}^{i,j}$ and $x_{\text{out}}^{i,j}$) corresponding to its successive states. The graph of a repair of t devices is shown on Figure 7 (assume t divides k.). First, devices perform a collecting step represented by edges $x_{\text{out}}^{k,j} \to x_{\text{in}}^{i,j}$ ($k < i$) of capacity β (d such edges). Second, devices undergo a coordinating step represented by edges $x_{\text{in}}^{i,j} \to x_{\text{coor}}^{i,j'}$ of capacity β' for $j \neq j'$ ($t-1$ such edges). Devices keep everything they obtained during the first step justifying the infinite capacities of edges $x_{\text{in}}^{i,j} \to x_{\text{coor}}^{i,j}$. Third, they store α using edges $x_{\text{coor}}^{i,j} \to x_{\text{out}}^{i,j}$. Figure 8 gives other examples of information flow graphs.

The graph $\mathcal{G}$ evolves as repairs are performed. When a repair is performed, a set of nodes is added to the graph and the nodes corresponding to failed devices become inactive (i.e., subsequently added intermediary nodes or data collectors cannot be connected to these nodes).

The rest of the article relies on the concept of minimum cuts in information flow graphs. A cut $\mathcal{C}$ between S and DC_i is a subset of edges of $\mathcal{G}$ such that there is no path from S to DC_i that does not have at least one edge in $\mathcal{C}$. The minimum cut is the cut that has the smallest sum of edge capacities.

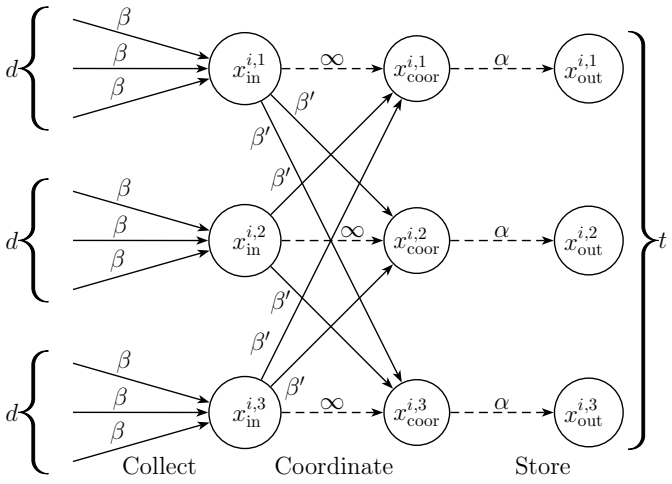


Figure 7: Information flow graph of a repair of $t = 3$ devices. The internal nodes of the graph represent intermediary steps in the repair. First, each device collects β from d live devices. Second, devices coordinate by exchanging β' with each other. Third, they store α. Plain edges correspond to network communication and dashed edges correspond to local communication.

3.3 Achievable codes

We define two important properties on codes:

Correctness A code $(n, k, d, t, \alpha, \gamma)$ is correct iff, for any system corresponding to an information flow graph which contains corresponding repairs, a data collector can recover the file by connecting to any k devices.

Optimality A code $(n, k, d, t, \alpha, \gamma)$ is optimal iff it is correct and any code $(n, k, d, t, \bar{\alpha}, \bar{\gamma})$ with $(\bar{\alpha}, \bar{\gamma}) < (\alpha, \gamma)$ is not correct.

The following theorem is an important result of our work.

Theorem 1. *A coordinated regenerating code $(n, k, d, t, \alpha, \gamma)$ is correct iff it is possible to find β and β' such that the Constraints* (1)*,* (2)*, and* (3) *are satisfied. A code matching the constraints with equality is optimal. If the code is correct, linear network codes suffice to build one instance. The repair cost of this code is γ as defined by Equation* (1)*.*

$$\gamma = d\beta + (t-1)\beta' \tag{1}$$

$$\sum_{i=0}^{k/t-1} t \min\{(d-it)\beta, \alpha\} \geq \mathcal{M} \tag{2}$$

$$\sum_{i=0}^{k-1} \min\{(d-i)\beta + (t-1)\beta', \alpha\} \geq \mathcal{M} \tag{3}$$

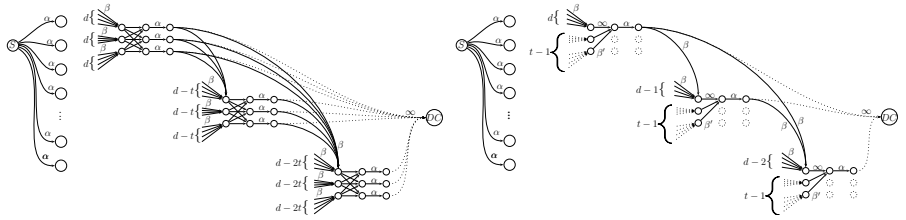
These constraints, proved in the rest of this subsection, mean that the sum of the amounts of information that can be downloaded from each of the k devices contacted by a data collector must be greater than the file size (amount of information needed to recover the original file). To this end, we consider, for a given coordinated regenerating code $(n, k, d, t, \alpha, \gamma)$, all (infinite) corresponding information flow graphs and evaluate the flow of information that can go from the source to any data collector in such graphs. We show that Equation (2) and Equation (3) must be satisfied to allow decoding at anytime (i.e., as long as the aforementioned constraints are satisfied, no data is lost).

Lemma 2. *For any information flow graph $\mathcal{G}$, no data collector* DC *can recover the initial file if the minimum cut in $\mathcal{G}$ between* S *and* DC *is smaller than the initial file size $\mathcal{M}$.*

Proof. Similarly to the proof in [9], since each link in the information flow graph can be used at most once, and since source to data collector capacity is less than the file size $\mathcal{M}$, the recovery of the file is impossible. □

Lemma 3. *For any finite information flow graph $\mathcal{G}$, if the minimum of the min-cuts separating the source and each data collector is larger than or equal to the file size $\mathcal{M}$, then there exists a linear network code such that all data collectors can recover the file. Furthermore, randomized network coding allows all collectors to recover the file with high probability.*

Proof. Similarly to the proof in [9], since the reconstruction problem reduces to multicasting on all possible data collectors, the result follows from the results in network coding theory which are briefly discussed in Section 2. □



(a) $\mathcal{G}_1^\star$ (b) $\mathcal{G}_2^\star$

Figure 8: Information flow graphs for which bounds in Equations (2) and (3) are matched with equality.

Lemma 4. *For any information flow graph $\mathcal{G}$ compound of initial devices that obtain α bits directly from the source* S *and of additional devices that join the graph in groups of t devices obtaining β from d existing devices and β' from each of the other $t-1$ joining devices, any data collector* DC *that connects to a subset of k out-nodes (g groups of u_i out-nodes) of $\mathcal{G}$ satisfies:*

$$\text{mincut}(\text{S},\text{DC}) \geq \sum_{i=0}^{g-1} u_i \min\{\alpha, (d-\sum_{j=0}^{i-1} u_j)\beta + (t-u_i)\beta'\} \tag{4}$$

where g is the number of different groups contacted, u_i is the size of each group, and $k = \sum_{i=0}^{g-1} u_i$ with $1 \leq u_i \leq t$.

Proof. We show that Equation (4) must be satisfied for any graph $\mathcal{G}$ (see examples in Figure 8) formed by adding devices according to the repair process described above. Consider a data collector DC that connects to a k-subset of "out-nodes", corresponding to a set of devices I, say $\{x_{\text{out}}^{i,j} : (i,j) \in I\}$. We show that any $\text{S}-\text{DC}$ cut in $\mathcal{G}$ has a capacity that satisfies Equation (4).

As all incoming edges of DC have infinite capacity, we only examine cuts $(U, \bar{U})$ with $\text{S} \in U$ and $\{x_{\text{out}}^{i,j} : (i,j) \in I\} \subset \bar{U}$.

Let $\mathcal{C}$ denote the edges in the cut (*i.e*, the set of edges going from U to $\bar{U}$). Every directed acyclic graph has a topological sorting [2], which is an ordering of its vertices such that the existence of an edge $x \to y$ implies $x < y$. In the rest of the analysis, we group nodes that were repaired simultaneously. Since nodes are sorted, nodes consider at one step cannot depend on nodes considered at the following steps.

First group. Let J be a set of indexes such that $\{x_{\text{out}}^{0,j} : j \in J\}$ are the topologically first output nodes in $\bar{U}$ corresponding to a first (same) repair. The set contains $\#\{x_{\text{out}}^{0,j} : j \in J\} = u_0$ nodes. Consider a subset $M \subset J$ of size m such that $\{x_{\text{in}}^{0,j} : j \in M\} \subset U$ and $\{x_{\text{in}}^{0,j} : j \in J - M\} \subset \bar{U}$. m can take any value between 0 and u_0.

First, consider the m nodes $\{x_{\text{in}}^{0,j} : j \in M\}$. For each node, $x_{\text{in}}^{0,j} \in U$. We consider two cases.

- If $x_{\text{coor}}^{0,j} \in U$, then $x_{\text{coor}}^{0,j} \to x_{\text{out}}^{0,j} \in \mathcal{C}$. The contribution to the cut is α.
- If $x_{\text{coor}}^{0,j} \in \bar{U}$, then $x_{\text{in}}^{0,j} \to x_{\text{coor}}^{0,j} \in \mathcal{C}$. The contribution to the cut is ∞.

Second, consider the $u_0 - m$ other nodes $\{x_{\text{in}}^{0,j} : j \in J - M\}$. For each node, the contribution comes from multiple sources.

- The cut contains d edges carrying β: since $x_{\text{out}}^{0,j}$ are the topologically first output nodes in $\bar{U}$, edges come from output nodes in U.
- The cut contains $t - u_0 + m$ edges carrying β' thanks to the coordination step. The node $x_{\text{coor}}^{0,j}$ has t incoming edges $x_{\text{in}}^{0,k} \to x_{\text{coor}}^{0,j}$. However, since $\#(\{x_{\text{in}}^{0,k}\} \cap \bar{U}) = u_0 - m$, the cut contains only $t - (u_0 - m)$ such edges.

Therefore, the total contribution of these nodes is

$$c_0(m) \geq m \min(\alpha, \infty) + (u_0 - m)(d\beta + (t - u_0 + m)\beta')$$

Since the function c_0 is concave on the interval $[0 : u_0]$, the contribution can be bounded thanks to Jensen's inequality.

$$c_0(m) \geq u_0 \min\{\alpha, d\beta + (t - u_0)\beta'\}$$

Second group. Let $\{x_{\text{out}}^{1,j} : j \in J\}$ be the topologically second output nodes in $\bar{U}$ corresponding to a second (same) repair. We follow a similar reasoning.

First, consider the m nodes $\{x_{\text{in}}^{1,j} : j \in M\} \subset U$. Similarly to the above, the contribution of each node is $\min(\alpha, \infty)$.

Second, consider the $u_0 - m$ nodes $\{x_{\text{in}}^{1,j} : j \in J - M\} \subset \bar{U}$. For each node, the contribution comes from multiple sources.

- The cut contains at least $d - u_0$ edges carrying β: since $x_{\text{out}}^{1,j}$ are the topologically second output nodes in $\bar{U}$, at most u_0 edges come from output nodes in U, and at least $d - u_0$ other edges come from output nodes in $\bar{U}$.
- Similarly to the above, the cut contains $t - u_1 + m$ edges carrying β' thanks to the coordination step.

Therefore, the total contribution of these nodes is

$$c_1(m) \geq u_1 \min\{\alpha, (d - u_0)\beta + (t - u_1)\beta'\}$$

***i*-th group.** Following the same reasoning, we find that the i-th group of nodes $(i = 0, \ldots, g - 1)$ in sorted set $\bar{U}$ contributes

$$c_i(m) \geq u_i \min\{\alpha, (d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_i)\beta'\}$$

Summing these contributions leads to Equation (4). □

Proof of Theorem 1. From Lemmas 2 and 3, we require $\text{mincut}(\text{S}, \text{DC}) \geq \mathcal{M}$ and, from Lemma 4, we know $\text{mincut}(\text{S}, \text{DC})$ satisfies Equation (4). By combining both, we know that a code satisfying the following equation is correct.

$$\sum_{i=0}^{g-1} u_i \min\{\alpha, (d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_i)\beta'\} \geq \mathcal{M}$$

The two extreme cases described hereafter can be identified.

First, the highest contribution of β is required when no contribution comes from β' (i.e., when $u_i = t$ for all i and $g = k/t$). In this case, Equation (4) boils down to Equation (2). This constraint is optimal as it is matched with

equality for graph $\mathcal{G}_1^\star$. $\mathcal{G}_1^\star$, depicted on Figure 8a, correspond to a data collector connecting to k devices that have joined successively by groups of t. Devices get their data in priority from the previously joined groups.

Second, the highest contribution of β' is required when the contribution from β is reduced (i.e., when $u_i = 1$ for all i and $g = k$). In this case, Equation (4) boils down to Equation (3). This constraint is optimal as it is matched with equality for graph $\mathcal{G}_2^\star$. $\mathcal{G}_2^\star$, depicted on Figure 8b, correspond to a data collector connecting to k devices that have joined in independent groups of size t. Devices get their data in priority from the previously joined devices. □

3.4 Optimal tradeoffs

Determining the optimal tradeoffs boils down to minimizing storage cost α and repair cost γ, as defined by Equation (1), under constraints of Equations (2) and (3). k, d and t are constants and α, β and β' are parameters to be optimized. In this subsection, we provide optimal tradeoffs (α, γ) between storage cost and repair cost (bandwidth).

3.4.1 MSCR codes

Minimum Storage Coordinated Regenerating Codes correspond to optimal codes that provide the lowest possible storage cost α and also minimize the repair cost γ. Figure 9 compares MSCR codes to both Dimakis *et al.*'s MSR [9] and erasure correcting codes with delayed repairs (ECC). Note that for $d = k$, we share the same repair cost as erasure correcting codes delayed repair. Yet, in this case, we still have the advantage that we balance the load evenly thus avoiding bottlenecks.

$$\alpha = \frac{\mathcal{M}}{k} \qquad \beta = \frac{\mathcal{M}}{k}\frac{1}{d-k+t} \qquad \beta' = \frac{\mathcal{M}}{k}\frac{1}{d-k+t}$$

Proof. The minimum storage is $\alpha = \frac{\mathcal{M}}{k}$ since k devices storing α must provide at least $\mathcal{M}$. The sum in Equation (2) (resp. Equation (3)) is a sum of k/t elements of size at most $t\mathcal{M}/k$ (resp. k elements of size $\mathcal{M}/k$). As the sum must be at least $\mathcal{M}$, each element must be equal to $t\mathcal{M}/k$ (resp. $\mathcal{M}/k$). Each part of the min must be greater or equal to $t\mathcal{M}/k$ (resp. $\mathcal{M}/k$). In particular the smallest, for $i = k/t - 1$ (resp. $i = k - 1$), must be equal (for our codes to be optimal). Therefore, from the Constraint (2), we obtain $(d-k+t)\beta = \mathcal{M}/k$. From the Constraint (3), we obtain $(d-k+1)\beta + (t-1)\beta' = \mathcal{M}/k$. □

3.4.2 MBCR codes

Minimum Bandwidth Coordinated Regenerating Codes correspond to optimal codes that provide the lowest possible repair cost (bandwidth consumption) γ and minimize the storage cost α. Figure 9 compares MBCR codes to both Dimakis *et al.* 's MBR [9] and erasure correcting codes with delayed repairs (ECC). Note that similarly to MBR, there is no expansion since every bit received is stored (i.e., $\alpha = \gamma$).

$$\alpha = \frac{\mathcal{M}}{k}\frac{2d+t-1}{2d-k+t}$$

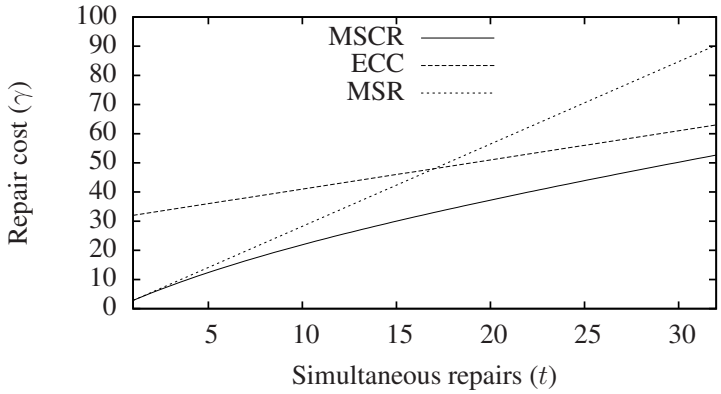


Figure 9: Total repair cost $t\gamma$ for $d = 48$ and $k = 32$. MSCR codes permanently outperform both erasure correcting codes and regenerating codes

$$\beta = \frac{\mathcal{M}}{k}\frac{2}{2d-k+t} \qquad \beta' = \frac{\mathcal{M}}{k}\frac{1}{2d-k+t}$$

Proof. First, we minimize γ alone by setting $\alpha = \infty$. We can evaluate the sums and obtain the values β and β'. Then, we know that α must be greater or equal to all second parts of min. Therefore, $\alpha = d\beta + (t-1)\beta'$. □

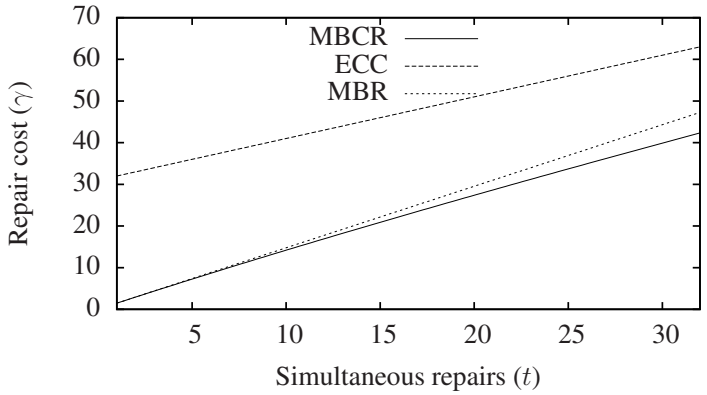


Figure 10: Total repair cost $t\gamma$ for $d = 48$ and $k = 32$. MBCR codes permanently outperform both erasure correcting codes and regenerating codes

3.4.3 CR codes

The general case corresponds to all possible trade-offs in between MSCR and MBCR. Valid points (α, β, β') can be determined by performing a numerical optimization of the objective function. Figure 11 shows the optimal tradeoffs (α, γ): coordinated regenerating codes ($t > 1$) can go beyond the optimal tradeoffs for independent repairs ($t = 1$) defined by regenerating codes by Dimakis *et al.* [9].

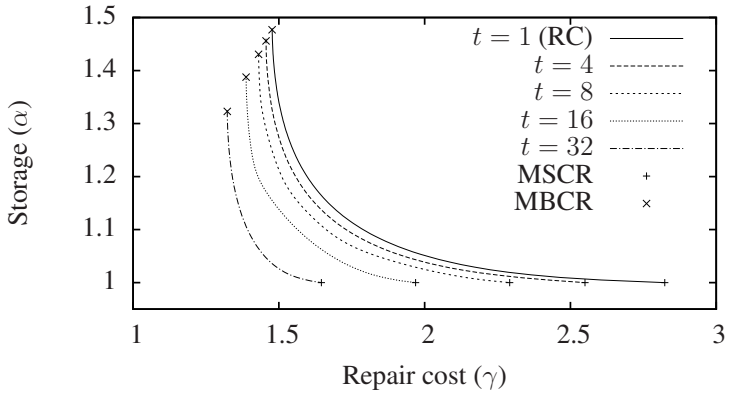


Figure 11: Optimal tradeoffs between storage and repair costs for $k = 32$ and $d = 48$. Regenerating codes (RC) [9] are depicted as $t = 1$. For each t, both MSCR and MBCR are shown. Costs are normalized by $\mathcal{M}/k$.

3.5 Optimal threshold

As previously explained, in regenerating codes, the higher d, the higher the savings on γ. Moreover, when repairs are delayed, higher values of t lead to higher savings on γ. If we consider a system of constant size $n = d + t$, these two objectives are contradictory: the longer the delay, the lower the number of live devices d. An interesting question is what is the optimal threshold t for triggering repairs considering $d+t$ to be constant (i.e., is it useful to deliberately delay repairs?).

Theorem 5. *If we consider a system of size $n = d + t$, for MBCR codes, the optimal value is $t = 1$ while for MSCR codes any value $t \in \{1 \dots n - k\}$ is optimal.*

Proof. Let us consider the repair cost assuming that $n = d + t$ is constant. For MBCR codes, the cost $\gamma = \frac{\mathcal{M}}{k} \frac{2n-t-1}{2n-k-t}$ increases when t increases. The optimal value of t for MBCR codes is the lowest possible value (i.e., $t = 1$). For MSCR codes, the cost $\gamma = \frac{\mathcal{M}}{k} \frac{n-1}{n-k}$ does not depend on t. The repair cost of MSCR remains constant, and t can be set to any value as there is no optimum. Neither MSCR nor MBCR allow additional gains when deliberately delaying repairs (i.e., deliberately setting $t > 1$). □

Corollary 6. *If we consider a system of size $n = d + t$ where t can be chosen freely (i.e., the value of t is not constrained by the system) both MSR and MBR regenerating codes [9] are optimal. Hence deliberately delaying repairs to force high values for t does not bring additional savings.*

However, if several failures are detected simultaneously, coordinated regenerating codes remain more efficient as they leverage simultaneous failures by coordinating during repairs.

4 Adaptive regenerating codes

In the previous section, we presented coordinated regenerating codes that assume t and d to remain constant across repairs. This is similar to regenerating codes [9], in which d remains constant across repairs. Yet, in real systems, the failure rate cannot be assumed to remain constant over time and, we cannot assume that every single failure can be repaired before a second one occurs.

In the particular case of Minimum Storage ($\alpha = \frac{\mathcal{M}}{k}$), such strong assumptions are not needed. Indeed, when minimizing the sum in Equations (2) and (3), we can minimize the different elements of the sum, which correspond to repairs, independently. Therefore, repairs are independent. We propose to adapt the quantities to transfer β and β' to the system state which is defined by the number t of devices being repaired and the number d of live devices.

4.1 Our approach

Theorem 7. *Adaptive regenerating codes (k, Γ) are both correct and optimal. Γ is a function $(t, d) \to (\beta_{t,d}, \beta'_{t,d})$ that maps a particular repair setting to the amounts of information to be transferred during a repair.*

$$\beta_{t,d} = \frac{\mathcal{M}}{k}\frac{1}{d-k+t} \qquad\qquad \beta'_{t,d} = \frac{\mathcal{M}}{k}\frac{1}{d-k+t} \tag{5}$$

In this subsection, we prove they are correct and optimal.

Lemma 8. *For any information flow graph $\mathcal{G}$ compound of initial devices that obtain α bits directly from the source* S *and of additional devices that join the graph in groups of t_i devices obtaining β_{t_i,d_i} from d_i existing devices and β'_{t_i,d_i} from each of the other $t_i - 1$ joining devices, any data collector* DC *that connects to a subset of k out-nodes (g groups of u_i out-nodes) of $\mathcal{G}$ has a* mincut(S, DC) *greater or equal to:*

$$\sum_{i=0}^{g-1} u_i \min\{(d_i - \sum_{j=0}^{i-1} u_j)\beta_{t_i,d_i} + (t_i - u_i)\beta'_{t_i,d_i}, \frac{\mathcal{M}}{k}\}$$

where g is the number of different groups contacted, u_i is the size of each group, and $k = \sum_{i=0}^{g-1} u_i$ with $1 \leq u_i \leq t_i$.

Proof. The proof is similar to the proof of Lemma 4. □

Proof of Theorem 7 (Correctness). Using Lemmas 2, 3 and 8, we can define the following sufficient condition for the code to be correct. The constraint is satisfied when β and β' take the values defined in Equations (5).

$$\sum_{i=0}^{g} u_i \min\{(d_i - \sum_{j=0}^{i-1} u_j)\beta_{t_i,d_i} + (t_i - u_i)\beta'_{t_i,d_i}, \frac{\mathcal{M}}{k}\} \geq \mathcal{M}$$

Since each element of the sum is at most $u_i\frac{\mathcal{M}}{k}$, each element of the sum must satisfy the following constraint.

$$\forall i < g, \min\{(d_i - \sum_{j=0}^{i-1} u_j)\beta_{t_i,d_i} + (t_i - u_i)\beta'_{t_i,d_i}, \frac{\mathcal{M}}{k}\} \geq \frac{\mathcal{M}}{k}$$

which simplifies to

$$\forall i < g,\ (d_i - \sum_{j=0}^{i-1} u_j)\beta_{t_i,d_i} + (t_i - u_i)\beta'_{t_i,d_i} \geq \frac{\mathcal{M}}{k}$$

Applying formulas of Equations (5),

$$\forall i < g,\ \frac{1}{d_i - k + t_i}(d_i - \sum_{j=0}^{i-1} u_j + (t_i - u_i)) \geq 1$$

which is satisfied if $\forall i \leq g-1,\ \sum_{j=0}^{i} u_j \leq k$, which is true since $\sum_{j=0}^{g-1} u_j = k$ and $u_j > 0$. Therefore, adaptive regenerating codes are correct. □

Proof of Theorem 7 (Optimality). We prove by contradiction that the proposed codes are optimal. Let us assume that there exists a correct code $(k, \bar{\Gamma})$ such that $\bar{\Gamma} < \Gamma$ (i.e., for some (t, d), $\bar{\Gamma}(t, d) < \Gamma(t, d)$). This is equivalent to (k, Γ) not being optimal.

Consider a set of failures such that all repairs are performed by groups of t devices downloading data from d devices. Consider the corresponding information flow graph. Assuming repairs are performed with a correct code $(k, \bar{\Gamma})$, the information flow graph also corresponds to a correct code $(d+t, k, t, d, \alpha, \bar{\beta}_{t,d}, \bar{\beta}'_{t,d})$.

Moreover, according to the previous section, these failures can be repaired optimally using the MSCR code $(d+t, k, t, d, \alpha, \beta_{t,d}, \beta'_{t,d})$. Therefore, there is a contradiction since the code $(d+t, k, t, d, \alpha, \bar{\beta}_{t,d}, \bar{\beta}'_{t,d})$ cannot be correct if the code $(d+t, k, t, d, \alpha, \beta_{t,d}, \beta'_{t,d})$ is optimal. A correct code $(k, \bar{\Gamma})$ cannot exist, and the adaptive regenerating code (k, Γ) defined in this section is optimal. □

Building on results from coordinated regenerating codes (especially MSCR), we have defined adaptive regenerating codes and proved that they are both correct and optimal. These codes are of particular interest for dynamic systems where failures may occur randomly and simultaneously.

4.2 Strawman approach

The strawman approach to the problem of adaptive repair would be to build, from MSR codes defined by Dimakis *et al.* in [9], and using a similar approach, adaptive regenerating codes without delayed repairs (k, Γ') where Γ' is a function $d \to \beta_d$. The t repairs needed are performed independently.

$$\beta_d = \frac{\mathcal{M}}{k}\frac{1}{d-k+1} \tag{6}$$

Proof. These codes are correct. A proof relies on observing that they are a particular sub-case of our approach described above ($t = 1$). □

4.3 Comparison

We can compare our approach to the strawman approach in the particular case where $d + t = n$. The average cost per repair of our codes remains constant $\gamma = \frac{\mathcal{M}}{k}\frac{n-1}{n-k}$. In the strawman approach, which requires repairs to be performed

independently, the average repair cost $\gamma' = \frac{\mathcal{M}}{k}\frac{n-t}{n-t-k+1}$ increases with t. Therefore, the performance of our adaptive regenerating codes does not degrade when the number of failures increases, as opposed to the simple construction upon Dimakis *et al.* 's codes. This is also shown on Figure 12.

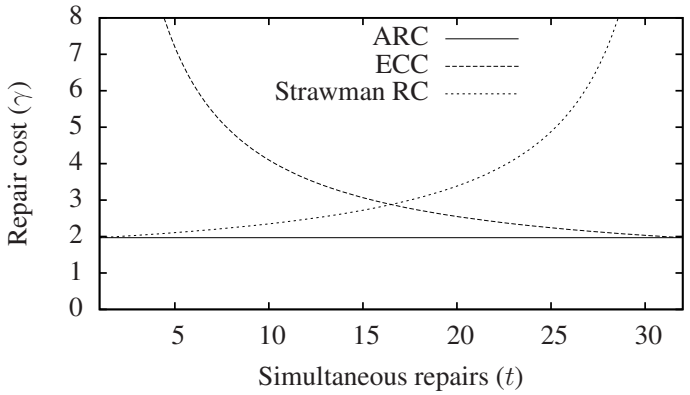


Figure 12: Average repair cost γ for $n = 64$ and $k = 32$. Adaptive Regenerating Codes (ARC) permanently outperform both erasure correcting codes (ECC) and the strawman approach based upon regenerating codes

5 Conclusion

In this paper we have proposed novel and optimal *coordinated regenerating codes* by considering simultaneous repairs in regenerating codes. This work has both a theoretical and practical impact. From a theoretical standpoint, we proved that deliberately delaying repairs cannot provide further gain in term of communication costs. However, in practical systems where several failures are detected simultaneously, our coordinated regenerating codes outperform regenerating codes [9] and are optimal. We also proposed *adaptive regenerating codes* that allow adapting the repair strategy to the current state of the system so that it always performs repairs optimally. For both codes, we provided and proved the minimum amounts of information that need to be transfered during the repair thus defining the fundamental tradeoffs between storage and repair costs for coordinately repairing multiple failures. Finally, our *coordinated regenerating codes* can also be viewed as a global class of codes that encompass erasure correcting codes with delayed repairs ($d = k$), regenerating codes ($t = 1$), erasure correcting codes ($t = 1$ and $d = k$), and new codes ($t > 1$ and $d > k$) that combine, previously incompatible, existing approaches of regenerating codes and delayed repairs (cf. Figure 4).

So far, our study has focused on the theoretical framework for performing multiple coordinated repairs in distributed storage. A straightforward implementation of such codes relies on random linear network codes (cf. Figure 6). Coordinated regenerating codes can therefore be used as they are for self-healing distributed storage. Yet, the seminal paper on regenerating codes [8] has been followed by studies of more constrained repairs (deterministic repairs known as exact repairs by opposition with functional repairs studied in this paper).

In short, in functional repairs, the regenerated redundancy is not necessarily the same as the lost one while, in exact repairs, the regenerated redundancy is exactly the same as the lost one. A recent survey [12] details the various works on exact repairs. Since our coordinated regenerating codes only consider the problem of functional repair (i.e., they leave open the problem of coordinated exact repair), an interesting extension of this work would be to study deterministic coordinated regenerating codes performing exact repair. Indeed, they would have appealing properties (simpler integrity checking, lower decoding complexity) when compared to coordinated regenerating codes based upon random linear network codes.

References

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Li, and Raymond Yeung. Network Information Flow. *IEEE Transactions On Information Theory*, 46(4):1204–1216, Jul. 2000.

[2] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag, 2001.

[3] Ranjita Bhagwan, Kiran Tati, Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total Recall: System Support for Automated Availability Management. In *NSDI*, 2004.

[4] Charles Blake and Rodrigo Rodrigues. High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two. In *HotOS*, 2003.

[5] Frank Dabek, Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area Cooperative Storage with CFS. In *SOSP*, 2001.

[6] Olivier Dalle, Frédéric Giroire, Julian Monteiro, and Stéphane Pérennes. Analysis of Failure Correlation Impact on Peer-to-Peer Storage Systems. In *P2P*, 2009.

[7] Anwitaman Datta and Karl Aberer. Internet-scale storage systems under churn – a study of steady-state using markov models. In *P2P*, 2006.

[8] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin O. Wainwright, and Kannan Ramchandran. Network Coding for Distributed Storage Systems. In *INFOCOM*, 2007.

[9] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin O. Wainwright, and Kannan Ramchandran. Network Coding for Distributed Storage Systems. *IEEE Transactions On Information Theory*, (to appear), 2010.

[10] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks Through Decentralized Erasure Codes. In *IPSN*, 2005.

[11] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. Decentralized erasure codes for distributed networked storage. In *Joint special issue, IEEE/ACM Transactions on Networking and IEEE Transactions on Information Theory*, 2006.

[12] Alexandros G. Dimakis, Kannan Ramchandran, Yunnan Wu, and Changho Suh. A survey on network codes for distributed storage. *CoRR*, abs/1004.4438, 2010.

[13] Alessandro Duminuco and Ernst Biersack. A Pratical Study of Regenerating Codes for Peer-to-Peer Backup Systems. In *ICDCS*, 2009.

[14] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In *SOSP*, 2003.

[15] Tracey Ho, Muriel Médard, Ralf Koetter, David Karger, Michelle Effros, Jun Shi, and Ben Leong. A Random Linear Network Coding Approach to Multicast. *IEEE Transaction on Information Theory*, 52(10):4413–4430, October 2006.

[16] Abhhinav Kamra, Vishal Misra, Jon Feldman, and Dan Rubenstein. Growth Codes: Maximizing Sensor Network Data Persistence. In *SIGCOMM*, 2006.

[17] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11:782–795, 2003.

[18] Shuo-Yen Li, Raymond Yeung, and Ning Cai. Linear Network Coding. *IEEE Transactions On Information Theory*, 49:371–381, 2003.

[19] W. K. Lin, D. M. Chiu, and Y. B. Lee. Erasure Code Replication Revisited. In *P2P*, 2004.

[20] Yunfeng Lin, Baochun Li, and Ben Liang. Differentiated Data Persistence with Priority Random Linear Codes. In *ICDCS*, 2007.

[21] Ramsés Morales and Idranil Gupta. Avmon: Optimal and scalable discovery of consistant availability monitoring overlays for distributed systems. *IEEE Transaction on Parallel and Distributed Systems*, 20:446–459, 2009.

[22] K. V. Rashmi, Nihar B. Shah, P. Vijay Kumar, and Kannan Ramchandran. Explicit Construction of Optimal Exact Regenerating Codes for Distributed Storage. In *Allerton Conference on Control, Computing, and Communication*, 2009.

[23] Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz. Pond: the OceanStore Prototype. In *FAST*, 2003.

[24] Rodrigo Rodrigues and Barbara Liskov. High Availability in DHTs: Erasure Coding vs. Replication. In *IPTPS*, 2005.

[25] Nihar B. Shah, K.V. Rashmi, P. Vijay Kumar, and Kannan Ramchandran. Explicit codes minimizing repair bandwidth for distributed storage. In *ITW*, 2010.

[26] Changho Suh and Kannan Ramchandran. Exact regeneration codes for distributed storage repair using interference alignment. In *ISIT*, 2010.

[27] Hakim Weatherspoon and John Kubiatowicz. Erasure Coding Vs. Replication: A Quantitative Comparison. In *IPTPS*, 2002.

[28] Yunnan Wu and Alexandros G. Dimakis. Reducing repair traffic for erasure coding-based storage via interference alignement. In *ISIT*, 2009.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399