Self-Repairing Codes for Distributed Storage — A Projective Geometric Construction

Frédérique Oggier
Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
Email: frederique@ntu.edu.sg

Anwitaman Datta
Division of Computer Science
School of Computer Engineering
Nanyang Technological University, Singapore
Email: anwitaman@ntu.edu.sg

Abstract—**Self-Repairing Codes (SRC) are codes designed to suit the need of coding for distributed networked storage: they not only allow stored data to be recovered even in the presence of node failures, they also provide a repair mechanism where as little as two live nodes can be contacted to regenerate the data of a failed node. In this paper, we propose a new instance of self-repairing codes, based on constructions of spreads coming from projective geometry. We study some of their properties to demonstrate the suitability of these codes for distributed networked storage.**

Index Terms—**self-repair, projective geometry, coding, distributed storage**

I. Introduction

Storing digital data is a basic necessity of modern societies. The volume of data to be stored is tremendous, and is rapidly increasing. The kinds of data vary widely - from corporate and financial data repositories, archive of electronic communications to personal pictures, videos and work documents stored and shared in Web 2.0 and cloud based services, and much more. Distribution of such huge amount of data over multiple networked storage devices is thus the only practical and scalable solution.

All across the wide gamut of networked distributed storage systems design space, eventual failure of any and all individual storage devices is a given. Consequently, storing data redundantly is essential for fault tolerance. Furthermore, over a period of time, due to failures or departure of storage devices from the system, the redundancy will gradually decrease - risking the loss of the stored data, unless the redundancy is recreated. The possible ways for recreating redundancy depends on, to start with, the kind of redundancy being used.

Data redundancy can be achieved using replication - however that entails a very large storage overhead. Erasure coding based strategies in contrast can provide very good amount of redundancy for a very low storage overhead. However, when an encoded data block is lost and needs to be recreated, for traditional erasure codes, one would first need data equivalent in amount to recreate the whole object in one place (either by storing a full copy of the data, or else by downloading adequate encoded blocks), even in order to recreate a single encoded block. Such drawback of traditional erasure codes has in recent years given rise to a new flavor of coding research: designing erasure codes which need much less information to carry out the recreation of a lost encoded block.

A. Related Work

More precisely, consider a network of n storage nodes, each with storage capacity α, where an object $\mathbf{o}$ of size B has to be stored. A source encodes the object $\mathbf{o}$, splits it into n blocks, each of size at most α, and stores such blocks at n storage nodes. When a data collector wants to retrieve the object, he should be able to do so by contacting a subset of live nodes. We define k as the minimum number of nodes that need to be contacted to retrieve the object, where the data collector may download upto $k\alpha$ amount of data, and possibly process (decode) the downloaded data. For maximal distance separable (MDS) erasure codes, any arbitrary subset of k nodes allow data retrievability. New nodes joining the network are assumed to perform the repair by contacting d live nodes, from each of which they download β amount of data.

There are arguably two extreme points possible in the design-space of codes for distributed networked storage:

(i) Minimize the absolute amount of data transfer $d\beta$ needed to recreate the lost data from one node. Network-coding inspired analysis determines the storage-bandwidth (per repair) trade-offs, and codes achieving this trade-off have been called *regenerating codes* (RGC) [2], [5]. Explicit RGCs are already known for certain points on the trade-off curve. Regenerating codes, like MDS erasure codes, allow data retrievability from any arbitrary set of k nodes.

(ii) Minimize the number of nodes to be contacted for repairing one node failure. Recently proposed *self-repairing codes* [4] achieve this optimal, by allowing one repair while contacting only two nodes, i.e. $d = 2$.[1] More specifically, self-repairing codes satisfy two cardinal properties, namely: (a) repairs can be performed directly through other subsets of nodes, without having to download data equivalent to that needed to reconstruct first the original object, ensuring that (b) a block is repaired from a fixed number of blocks, the number depending only on how many blocks are missing and independent of which specific blocks are missing. Note that

[1]Replication (or repetition code) would achieve $d = 1$, however it leads to very high storage overhead for equivalent fault-tolerance [6].

minimization of the number of contacted nodes for a repair is achieved when the fixed number in clause (b) is in fact two.

Homomorphic self-repairing codes (HSRC) were proposed in [4], which, besides satisfying the cardinal properties elaborated above, were shown to (i) have $(n-1)/2$ distinct pairs with which the data for a single missing node could be regenerated, and consequently, (ii) self-repair for up to $(n-1)/2$ node failures could be carried out simultaneously using two nodes for each self-repair, from the pool of the remaining $(n+1)/2$ live nodes.

B. Contributions

This paper proposes a new family of self-repairing codes (PSRC) derived from a projective geometric construction. Apart satisfying the basic properties of self-repairing codes, PSRC has several other salient features, summarized next:

(i) Both the encoding and self-repair processes for PSRC involve only XOR operations, unlike HSRC encoding which involved the relatively more expensive task of evaluating a polynomial.

(ii) Similar to regenerating codes [2], in PSRC, each encoded block (i.e., data stored by a node) comprise of several (α) pieces. Regeneration of the whole encoded block thus can likewise be done by regenerating the individual constituent pieces. This is in contrast to HSRC, where the encoded blocks were 'atomic', and hence repair of the whole encoded block had to be carried out atomically. This gives PSRC some of the advantages of regenerating codes, while also naturally retaining the advantages of self-repairing codes, and provides several additional desirable properties, as elaborated next.

(iii) For self-repair of a specific node, if one live node is chosen arbitrarily, then there are several other nodes with which the first chosen node can be paired to regenerate the lost encoded block. This is in contrast to HSRC, where there is a unique pairing for one lost node, once one live node is chosen.

(iv) While the resulting code is strictly speaking not systematic in terms of what is stored at each node, if the constituent pieces stored over the nodes are considered, then systematic reconstruction of the object is possible, though this will need communication with αk instead of k specific nodes.

II. Background from Projective Geometry

The proposed construction as described in next section relies on the notion of spread coming from projective geometry. We thus start by providing the required background.

Consider the finite field $\mathbb{F}_q$, where q is a power of a prime p, and a vector space of dimension m over $\mathbb{F}_q$, namely, a projective space denoted $PG(m-1,q)$. Note that we will adopt a row vector convention for the rest of the paper.

Definition 1: Let $\mathcal{P}$ be a projective space. A *t-spread* of $\mathcal{P}$ is a set $\mathcal{S}$ of t-dimensional subspaces of $\mathcal{P}$ which partitions $\mathcal{P}$, i.e., every point of $\mathcal{P}$ is contained in exactly one t-space of $\mathcal{S}$.

If $\mathcal{P}$=PG$(m-1,q)$ is a finite projective space, then a t-spread can only exist if the number of points of a t-space divides the number of points of the whole space, i.e., if $\frac{q^{t+1}-1}{q-1}|\ \frac{q^m-1}{q-1}$ and hence $(q^{t+1}-1)|(q^m-1)$, which holds if and only if $(t+1)|\ m$. André [1] showed that this necessary condition is also sufficient.

Theorem 1: [3] In PG$(m-1,q)$, a t-spread exists if and only if $t+1|\ m$.

A systematic construction of spreads can be obtained through field extensions as follows. Suppose that $t+1|\ m$. Consider the finite fields $F_0=\mathbb{F}_q$, $F_1=\mathbb{F}_{q^{t+1}}$ and $F_2=\mathbb{F}_{q^m}$. Then $F_0\subseteq F_1\subseteq F_2$. The field F_2 is an m-dimensional vector space V over F_0. The subspaces of V form the projective space $\mathcal{P}$=PG(m,q). The field F_1 is a $(t+1)$-dimensional subspace of V and hence a t-dimensional (projective) subspace of $\mathcal{P}$. The same holds for all cosets aF_1, $(a\in F_2)$. These cosets partition the multiplicative group of F_2. Hence they form a t-spread of $\mathcal{P}$.

Example 1: Take as base field $F_0=\mathbb{F}_2$, i.e., the alphabet is $\{0,1\}$. In order to obtain planes, we consider 1-spread, i.e., $t=1$ and hence $F_1=\mathbb{F}_4$. Finally, assume $m=4$, that is $F_2=\mathbb{F}_{16}$:

$$\begin{tikzcd} F_2=\mathbb{F}_{16} \arrow[d, dash, "\frac{m}{2}"] \\ F_1=\mathbb{F}_4 \arrow[d, dash, "2"] \\ F_0=\mathbb{F}_2 \end{tikzcd}$$

Denote by $\mathbb{F}_q^*$ the multiplicative group of $\mathbb{F}_q$. Recall that $\mathbb{F}_q^*$ is a cyclic group. Let ω and ν be the respective generators of F_2^* and F_1^*. We have that ν is an element of order 3 contained in F_2, so $\nu=\omega^5$. Thus F_1^* can be written $F_1^*=\{1,\omega^5,\omega^{10}\}$. As F_2^* can be written

$$F_2^*=\{\omega^i\}_{i=1}^{15}=\{\omega^i,\omega^{5+i},\omega^{10+i}\}_{i=1}^{5}=\bigsqcup_{i=1}^{5}\omega^i\mathbb{F}_4^*,$$

we have a partition (denoted by $\bigsqcup$) of $\mathbb{F}_{16}$ into cosets of the form $\omega^i\mathbb{F}_4^*$, $i=1,\dots,5$. These five cosets define five disjoint planes. More precisely, $\mathbb{F}_{16}$ can be decomposed into direct sums of $\mathbb{F}_2$:

$$\mathbb{F}_{16}=\mathbb{F}_4\oplus\nu\mathbb{F}_4=\mathbb{F}_2\oplus\nu\mathbb{F}_2\oplus\omega\mathbb{F}_2\omega\nu\mathbb{F}_2,$$

so that each element of $\mathbb{F}_{16}$ can be written as a 4-tuple. For example, the coset $\omega\mathbb{F}_4^*$ contains the elements $\omega,\omega\nu,\omega\nu^2$. As $\nu^2=\nu+1$, $\omega\nu^2$ is the sum of the two other points. Thus writing $\omega=(0,0,1,0)$ and $\omega\nu=(0,0,0,1)$, we finally get that the plane defined by the coset $\omega\mathbb{F}_4^*$ is $\{(0010),(0001),(0011)\}$.

III. Code Construction

Recall that our goal is to encode an object of size B to be stored over n nodes, each of storage capacity α, such that each failure can be repaired by contacting any d live nodes ($d\geq 2$) and downloading β amount of data from each of them. We denote by $PSRC(n,k)$ the self-repairing code with parameters n and k obtained from a spread construction.

N_1 to N_7	N_8 to N_{14}	N_{15} to N_{21}
(100000),(110111)	(011000),(001110)	(001010),(110100)
(010000),(101011)	(001100),(000111)	(000101),(011010)
(001000),(100101)	(000110),(110011)	(110010),(001101)
(000100),(100010)	(000011),(101001)	(011001),(110110)
(000010),(010001)	(110001),(100100)	(111100),(011011)
(000001),(111000)	(101000),(010010)	(011110),(111101)
(110000),(011100)	(010100),(001001)	(001111),(101110)

TABLE I
BASIS VECTORS FOR THE SCENARIO WHERE WE HAVE $B = 6$, $\alpha = 2$, $n = 1 + 2^2 + (2^2)^2 = 21$ NODES $N_1, \ldots, N_{21}$.

$B = k\alpha$	α	$n = 1 + 2^\alpha + \ldots (2^\alpha)^{k-1}$
4	2	5
6	2	21
6	3	9
8	2	85
8	4	17

TABLE II
SET OF SOME SMALL AVAILABLE PARAMETERS FOR $PSRC(n, k)$.

We will assume for simplicity that we work over the base field $\mathbb{F}_2$, though spreads can be constructed over larger alphabets.

A. Setting the Parameters and Encoding

1) We first set $m = B$, so that we are working with elements in $F_2 = \mathbb{F}_{q^B}$, that is B-dimensional vectors over $\mathbb{F}_2$.

2) Consider a t-spread $\mathcal{S}$ formed of t-dimensional subspaces of $\mathcal{P}$ such that $t+1|B$. In particular, take $F_1 = \mathbb{F}_{q^{t+1}}$. Since every subspace is a $(t+1)$-dimensional vector space over $\mathbb{F}_2$, it is described by a $\mathbb{F}_2$-basis containing $(t+1)$ vectors. We thus set $t+1 = \alpha$, and assign to each node an $\mathbb{F}_2$-basis containing α vectors. The number of nodes that will store the object is consequently (at most)

$$n = \frac{2^B - 1}{2^\alpha - 1}.$$

Since we must take $\alpha|B$, that is $B = b\alpha$, we can further write

$$n = \frac{2^{b\alpha} - 1}{2^\alpha - 1} = 1 + 2^\alpha + (2^\alpha)^2 + \ldots + (2^\alpha)^{b-1}. \quad (1)$$

For a data collector to be able to retrieve the object from some k nodes, we need $b \leq k$. In the following, we will discuss specifically the case when $b = k$, which corresponds to the minimum storage at each node.

3) Let us denote by v_i the collection of all $n\alpha$ vectors, ordered such that $v_1, \ldots, v_\alpha$ correspond to the first node, $v_{\alpha+1}, \ldots, v_{2\alpha}$ to the second node, etc. The ith node will then store $\{Bv^T_{(i-1)\alpha+1}, \ldots, Bv^T_{i\alpha}\}$ for a total storage of α.

Example 2: Consider the partition described in Example 1, where we recall that $\nu^4 = \nu + 1$, $|\mathbb{F}^*_{16}| = 15$, $\nu^{15} = 1$ $\omega^2 = \omega + 1$, $|\mathbb{F}^*_4| = 3$, $\omega^3 = 1$, $\omega = \nu^5 = \nu^2 + \nu$.

The final partition of the space is thus:

$$\begin{aligned}
\mathbb{F}^*_4 &= \{(1000), (0110), (1110)\} \\
\nu\mathbb{F}^*_4 &= \{(0100), (0011), (0111)\} \\
\nu^2\mathbb{F}^*_4 &= \{(0010), (1101), (1111)\} \\
\nu^3\mathbb{F}^*_4 &= \{(0001), (1010), (1011)\} \\
\nu^4\mathbb{F}^*_4 &= \{(1100), (0101), (1001)\}
\end{aligned}$$

This corresponds to the code parameters $B = 4$, $\alpha = 2$, $n = 1+2^2 = 5$ from (1). Let us denote by N_i, $i = 1, \ldots, 5$ the 5 storing nodes, with storage capacity $\alpha = 2$, and by $\mathbf{o} = (o_1, o_2, o_3, o_4)$ the object to be stored. For example, we can use the basis vectors as follows:

node	basis vectors	data stored
N_1	$v_1 = (1000),\ v_2 = (0110)$	$\{o_1, o_2 + o_3\}$
N_2	$v_3 = (0100),\ v_4 = (0011)$	$\{o_2, o_3 + o_4\}$
N_3	$v_5 = (0010),\ v_6 = (1101)$	$\{o_3, o_1 + o_2 + o_4\}$
N_4	$v_7 = (0001),\ v_8 = (1010)$	$\{o_4, o_1 + o_3\}$
N_5	$v_9 = (1100),\ v_{10} = (0101)$	$\{o_1 + o_2, o_2 + o_4\}$

Furthermore, the first available parameters are summarized in Table III-A.

B. Repair

We now need to make sure that the above coding strategy allows for object retrieval and repair. Let us discuss the repair of data stored in one storage node. It was shown in [4] for HSRC that it is possible to repair data for one node by contacting $d = 2$ nodes, and there are $(n-1)/2$ such choices of 2 nodes that allow repair. This holds also for PSRC.

Lemma 1: Suppose we have n nodes, each storing α pieces of data encoding an object using $PSRC(n, k)$. Then if one node N_l fails, it is possible to repair it by contacting $d = 2$ nodes. More precisely, for any choice of node N_i among the remaining $n-1$ live nodes, there exists at least one node N_j such that N_l can be repaired by downloading the data stored at nodes N_i and N_j.

Proof: The lth node N_l stores a subspace of the form $\nu^l\mathbb{F}^*_{2^\alpha}$, $l = 1, \ldots, n$. Let us assume this lth node fails, and a new comer joins. It contacts any node, say N_i. Since N_i stores $\nu^i\mathbb{F}^*_{2^\alpha}$, we need to show that there exists a node N_j such that

$$\nu^i\mathbb{F}^*_{2^\alpha} \bigsqcup \nu^j\mathbb{F}^*_{2^\alpha}$$

repairs N_l. Now

$$(\nu^i + \nu^l)\mathbb{F}^*_{2^\alpha} \subset \nu^i\mathbb{F}^*_{2^\alpha} \bigsqcup \nu^l\mathbb{F}^*_{2^\alpha}$$

so we can take j such that $\nu^j = \nu^i + \nu^l$. By combining the data stored at node N_i and N_j, we thus get

$$\nu^i\mathbb{F}^*_{2^\alpha} \bigsqcup (\nu^i + \nu^l)\mathbb{F}^*_{2^\alpha}$$

which contains $\nu^l\mathbb{F}^*_{2^\alpha}$. ■

Example 3: Let us continue with Example 2. If say N_1 fails, the data pieces o_1 (corresponding to the basis vector (1000)) and o_2+o_2 (corresponding to the basis vector (0110)) are lost. A new node joining the network can contact nodes N_3 and N_4, from which it gets respectively $v_5 = (0010)$, $v_6 = (1101)$ and $v_7 = (0001)$, $v_8 = (1010)$. Now $v_8 + v_5$ gives (1000) while $v_8 + (v_6 + v_7)$ gives (0110).

Actually, in general, the redundancy for self-repair provided by $PSRC$ is even stronger than that of $HSRC$, as we now illustrate.

Lemma 2: Suppose we have $n = 21$ nodes, each storing $\alpha = 2$ pieces of data, encoding an object of size $B = 6$ using $PSRC(21, 3)$, as summarized in Table 2. Then if one node N_l fails, for any choice of node N_i among the remaining 20 live nodes, there exists three nodes N_{j_1}, N_{j_2}, N_{j_3} such that N_l can be repaired by downloading the data stored at either nodes N_i and N_{j_1}, or N_i and N_{j_2}, or even N_i and N_{j_3}.

Proof: Recall that ω is the generator of the cyclic group $\mathbb{F}_4^*$. We have that node N_l stores $\nu^l\mathbb{F}_4^*$, and N_i similarly stores $\nu^i\mathbb{F}_4^*$. Now

$$\begin{gathered}\nu^l\mathbb{F}_4^* \bigsqcup \nu^i\mathbb{F}_4^* = \\ \{\nu^i + \nu^l, \nu^i\omega + \nu^l\omega, \nu^i + \nu^i\omega + \nu^j + \nu^j\omega \\ \nu^i, \nu^i\omega, \nu^i + \nu^i\omega, \\ \nu^l, \nu^l\omega, \nu^l + \nu^l\omega, \\ \nu^i + \nu^l\omega, \nu^i + \nu^i\omega + \nu^l, \nu^i\omega + \nu^l + \nu^l\omega \\ \nu^i\omega + \nu^l, \nu^i + \nu^i\omega + \nu^l\omega, \nu^i + \nu^l + \nu^l\omega\} = \\ (\nu^i + \nu^l)\mathbb{F}_4^* \bigsqcup \nu^i\mathbb{F}_4^* \bigsqcup \nu^l\mathbb{F}_4^* \bigsqcup (\nu^i + \nu^l\omega)\mathbb{F}_4^* \bigsqcup (\nu^l + \nu^i\omega)\mathbb{F}_4^*.\end{gathered}$$

Take j_1, j_2, j_3 such that

$$\nu^{j_1} = \nu^i + \nu^l, \ \nu^{j_2} = \nu^i + \nu^l\omega, \ \nu^{j_3} = \nu^l + \nu^i\omega.$$

We have then

$$\begin{aligned}(N_i, N_{j_1}) &\Rightarrow \nu^i\mathbb{F}_4^* \bigsqcup (\nu^i + \nu^l)\mathbb{F}_4^* \supset \nu^l\mathbb{F}_4^*, \\ (N_i, N_{j_2}) &\Rightarrow \nu^i\mathbb{F}_4^* \bigsqcup (\nu^i + \nu^l\omega)\mathbb{F}_4^* \supset \nu^l\mathbb{F}_4^*, \\ (N_i, N_{j_3}) &\Rightarrow \nu^i\mathbb{F}_4^* \bigsqcup (\nu^l + \nu^i\omega)\mathbb{F}_4^* \supset \nu^l\mathbb{F}_4^*.\end{aligned}$$

■

This proof actually gives an algorithm to find the different pairs that repair a given failed node.

Example 4: Consider the code described in Table 2, and suppose that the node N_1 fails, and a new comer contacts node N_4 which stores $\nu^3\mathbb{F}_4^*$. We have

$$\begin{aligned}\nu^l + \nu^i &= 1 + \nu^3 = \nu^{21}\nu^{11} &&\Rightarrow N_{12} \\ \nu^l\omega + \nu^i &= \omega + \nu^3 = \nu^{21}\nu^9 &&\Rightarrow N_{10} \\ \nu^l + \nu^i\omega &= 1 + \nu^3\omega = \nu^4 &&\Rightarrow N_5.\end{aligned}$$

Thus the node N_1 can be repaired by contacting the following three pairs all involving N_4: (N_4, N_{12}), (N_4, N_{10}), (N_4, N_5).

C. Object Retrieval

If a data collector connects to any choice of k nodes, then he can access upto $k\alpha$ blocks, while trying to reconstruct an object of size B. Thus, $k \geq B/\alpha$. Note that in the examples considered in this paper, $k = B/\alpha$.

Lemma 3: If $k = 2$, then the object can be retrieved from any choice of $k = 2$ nodes, in which case, we may see $PSRC(n, k)$ as a MDS code.

Proof: If $k = 2$, then each node stores $\alpha = B/2$ linearly independent vectors. Pick any two nodes say N (containing $v_1, \ldots, v_\alpha$) and N' (similarly storing $u_1, \ldots, u_\alpha$). Suppose that there exists a vector v in N which is linearly dependent of some vectors in N':

$$v = \sum_{i=1}^{\alpha} a_i v_i + \sum_{j=1}^{\alpha} b_j u_j.$$

Since $v \in N$ and $\sum_{i=1}^{n} \alpha a_i v_i \in N$, it must be that $\sum_{j=1}^{\alpha} b_j u_j \in N$, a contradiction since N and N' are non-intersecting by the definition of spread. ■

Note that any MDS code with $k = 2$ also trivially repairs a lost node by contacting $d = 2$ nodes.

To recover the object, the data collector just solves the system of linear equations in $\mathbf{o}$.

In general, when $k \geq 3$, SRC codes are not maximum distance separable (MDS). A static resilience analysis provides an estimate of how much deterioration the system may suffer due to the lack of the maximum distance separability.

Static resilience of a distributed storage system is defined as the probability that an object, once stored in the system, will continue to stay available without any further maintenance, even when a certain fraction of individual member nodes of the distributed system become unavailable. Let p_{node} be the probability that any specific node is available. Then, under the assumptions that node availability is $i.i.d$, and no two fragments of the same object are placed on any same node, we can consider that the availability of any fragment is also $i.i.d$ with probability p_{node}. The probability p_{obj} of recovering the object is then

$$p_{obj} = \sum_{x=k}^{n} \rho_x C_x^n p_{node}^x (1 - p_{node})^{n-x},$$

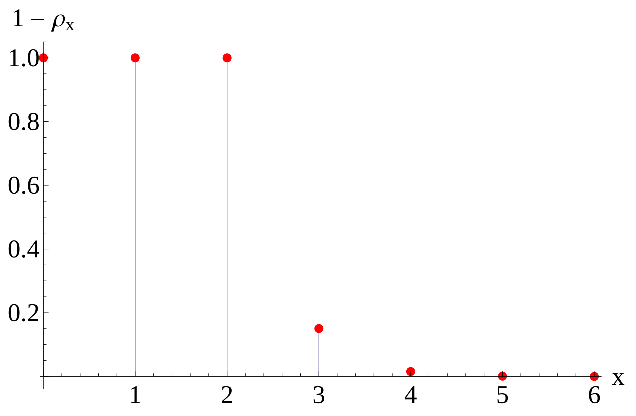
where ρ_x is the conditional probability that the stored object can be retrieved by contacting an arbitrary x out of the n storage nodes.

For (n, k) MDS erasure codes, ρ_x is a deterministic and binary value equal to one for $x \geq k$, and zero for smaller x. For self-repairing codes, the value is probabilistic. In Fig. 1(a) we show for our toy example $PSRC(21, 3)$ the probability that the object cannot be retrieved, i.e., $1 - \rho_x$, where the values of ρ_x for $x \geq k$ were determined by exhaustive search.[2]

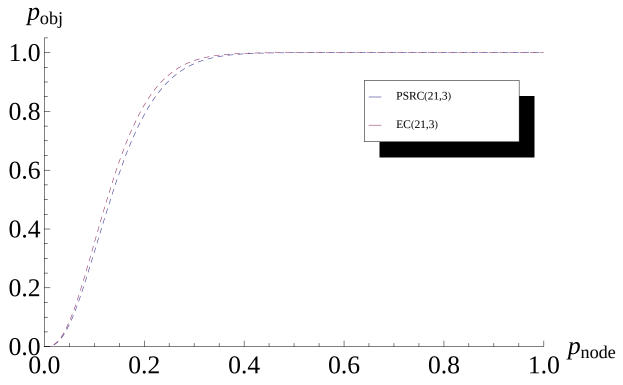
In particular, one can list 17 unique groups of 5 nodes, whose 10 basis vectors generate a matrix with rank less than 6, out of the $\binom{21}{5} = 20349$ unique groups of 5. This means that if we choose any 5 arbitrary nodes, the object still cannot be retrieved with a probability of 0.00083. Similarly, if we chose any arbitrary 3 nodes, the probability of unretrievability is 0.150375. In contrast, for MDS codes, the object will be retrievable from the data available at any arbitrary three nodes. Of-course, this rather marginal sacrifice provides PSRC an incredible amount of self-repairing capability. For any one node lost, as shown earlier in Lemma 2, one can choose any of the twenty remaining live nodes, and pair it with three other nodes, and regenerate the lost data.

In Fig. 1(b) we compare the static resilience p_{obj} for $PSRC(21, 3)$ with respect to what could be achieved using

[2] ρ_x is zero for $x < k$ for PSRC also.



(a) $1-\rho_x$ (determined using exhaustive enumeration)



(b) Static resilience (determined numerically)

Fig. 1. Results for $PSRC(21,3)$

a MDS $EC(21,3)$. The values were determined numerically, using the ρ_x values evaluated as mentioned above. We note that in practice a MDS erasure code may or not exist for any arbitrary (n,k) parameters. More importantly, we notice that the degradation of static resilience of $PSRC(21,3)$ to achieve the self-repairing property is marginal with respect to that of a MDS erasure code, if such a code were to/does exist.

IV. Further discussions

We point out a few more properties of the proposed codes.

Systematic Like Code: It is usually appreciated from an implementation perspective to use a systematic code, since it makes the object retrieval immediate. We notice that though our code is not systematic, we can however contact $B = \alpha k$ specific nodes (instead of k), namely those storing as pieces each of the canonical basis vectors of $\mathbb{F}_{2^B}$ to reconstruct the object in a systematic manner.

Bandwidth cost for regeneration: Unlike HSRC, the PSRC encoded blocks are not atomic, and instead comprise of α pieces. Thus, similar to regenerating codes, one could also expect to regenerate an encoded block piece-by-piece, by contacting more (larger d) number of nodes. For example, when using $PSRC(21,3)$, if the data for node N_1 needs to be regenerated, one could do so by contacting two nodes and downloading four pieces (units) of data, as we have already seen. One could instead also contact $d = 3$ nodes, and regenerate the two lost pieces by downloading only three units of data. For instance, by downloading (010000) from N_2, (110000) from N_7 and (000111) from N_9.

As noted previously, for our examples, $\alpha = B/k$, corresponding to what is known as the Minimum Storage Regeneration (MSR) point for regenerating codes. At MSR point, a node needs to contact $d \geq k$ nodes, and download $\frac{B}{k(d-k+1)}$ data from each, resulting in a total download of $\frac{Bd}{k(d-k+1)}$ data. Thus, for the same choices of α, B, k and with $d = 3$, one would need to download 6 units of data, and for $d = 4$, one would need to download 4 units of data, while $d = 2$ is not allowed. Thus, for the regeneration of one lost node, PSRC can outperform regenerating codes both in terms of absolute bandwidth needed, as well as the number of nodes needed to carry out such regeneration, moreover, for upto $(n-1)/2$ failures, the regeneration overhead per node's data stays constant for PSRC. It of-course needs to be noted that, in order to achieve these very interesting performance, we sacrificed the MDS property. In practice, this sacrifice however has marginal impact, as can be observed from the resulting codes' static resilience.

V. Concluding remarks

In this work, we showed the existence of another instance of self-repairing codes, which are codes tailor made to meet the peculiarities of distributed networked storage. The proposed code family in this paper is based on constructions of spreads from projective geometry. We provided a preliminary study of the properties of this new family, demonstrating that they outperform existing code families both in several quantitative as well as qualitative metrics. Further analysis to comprehend and harness these codes in practical settings are currently under investigation.

Acknowledgement

F. Oggier's research for this work has been supported by the Singapore National Research Foundation grant NRF-CRP2-2007-03. A. Datta's research for this work has been supported by AcRF Tier-1 grant number RG 29/09.

References

[1] J. André,"Uber nicht-desarguessche Ebenen mit transitiver Translationsgruppe," Math. Z., p. 156-186, 1954.

[2] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright and K. Ramchandran, "Network Coding for Distributed Storage Systems" *IEEE Transactions on Information Theory*, Vol. 56, Issue 9, Sept. 2010.

[3] J. Eisfeld and L. Storme, "Partial t-spreads and minimal t-covers in finite projective spaces", unpublished.

[4] F. Oggier and A. Datta, "Self-repairing Homomorphic Codes for Distributed Storage Systems", *INFOCOM 2011*.

[5] K. V. Rashmi, N. B. Shah, P. Vijay Kumar, K. Ramchandran, "Explicit Construction of Optimal Exact Regenerating Codes for Distributed Storage", *Allerton 2009*.

[6] H. Weatherspoon, J. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison", *IPTPS 2002*.