On Coding for Distributed Networked Storage Systems

Frédérique Oggier,
Joint work with Anwitaman Datta

Nanyang Technological University, Singapore

IMS-NTU Workshop on Coding and Cryptography,
Singapore, May 2011
Outline

1. Coding for Distributed Networked Storage

2. Self-Repairing Codes: Definition and Constructions

3. Self-Repairing Codes: Analysis and Properties
A data owner wants to store data over a network of nodes (e.g. data center, back-up or archival in peer-to-peer networks).
A data owner wants to store data over a network of nodes (e.g. data center, back-up or archival in peer-to-peer networks).

Redundancy is essential for resilience.
A data owner wants to store data over a network of nodes (e.g. data center, back-up or archival in peer-to-peer networks).

Redundancy is essential for resilience

- Replication: good availability and durability, but very costly.
A data owner wants to store data over a network of nodes (e.g. data center, back-up or archival in peer-to-peer networks).

Redundancy is essential for resilience

- Replication: good availability and durability, but very costly.
- Erasure codes: good trade-off of availability, durability and storage cost.
Erasure codes for storage systems

Data = Object

Encoding

Retrieval any $k'$ ($\geq k$) blocks

Decoding

Reconstruct Data

$k$ blocks

$n$ encoded blocks (stored in storage devices in a network)
Nodes may go offline, or may fail, so that the data they store becomes *unavailable*.
Nodes may go offline, or may fail, so that the data they store becomes *unavailable*.

Redundancy needs to be *replenished*, else data may be permanently lost over time (after multiple storage node failures).
Repair process using traditional Erasure Codes

Retrieve any $k'$ (≥ $k$) blocks

Decoding

Original $k$ blocks

$n$ encoded blocks
(stored in storage devices in a network)

Recreate lost blocks

Encoding

Re-insert

Reinsert in (new) storage devices, so that there is (again) $n$ encoded blocks.
Related work


Related work


Outline

1. Coding for Distributed Networked Storage

2. Self-Repairing Codes: Definition and Constructions

3. Self-Repairing Codes: Analysis and Properties
Self-Repairing Codes (SRC)

- Motivation: *minimize* the number of nodes necessary to repair a missing block.
Self-Repairing Codes (SRC)

- **Motivation:** *minimize* the number of nodes necessary to repair a missing block.
- **Gain:** lower *bandwidth* consumption, lower *computational complexity of repair*, possibility for *faster and parallel* replenishment of lost redundancy.
Self-Repairing Codes (SRC)

- Motivation: *minimize* the number of nodes necessary to repair a missing block.
- Gain: lower *bandwidth* consumption, lower *computational complexity of repair*, possibility for *faster and parallel* replenishment of lost redundancy.
- **Self-repairing codes** are \((n, k)\) codes such that

  - a fragment can be repaired from a fixed number of encoded fragments, independently of which specific blocks are missing (analogous to erasure codes supporting reconstruction using any \(n-k\) losses, independently of which).
Motivation: minimize the number of nodes necessary to repair a missing block.

Gain: lower bandwidth consumption, lower computational complexity of repair, possibility for faster and parallel replenishment of lost redundancy.

Self-repairing codes are \((n, k)\) codes such that encoded fragments can be repaired directly from other subsets of encoded fragments.
Self-Repairing Codes (SRC)

- Motivation: *minimize* the number of nodes necessary to repair a missing block.
- Gain: lower *bandwidth* consumption, lower *computational complexity of repair*, possibility for *faster and parallel* replenishment of lost redundancy.
- **Self-repairing codes** are \((n, k)\) codes such that
  - encoded fragments can be repaired *directly* from other subsets of encoded fragments.
  - a fragment can be repaired from a *fixed number* of encoded fragments, *independently* of which specific blocks are missing (analogous to erasure codes supporting reconstruction using any \(n - k\) losses, independently of which).
Self-Repairing Codes (a black-box view)

Retrieve some \( k' \) (< \( k \)) blocks (e.g. \( k' = 2 \)) to recreate a lost block.

Lost blocks

\( n \) encoded blocks (stored in storage devices in a network)

Reinsert in (new) storage devices, so that there is (again) \( n \) encoded blocks.
Homomorphic SRC (HSRC)

- A first instance of self-repairing code.

Self-repairing Homomorphic Codes for Distributed Storage Systems
F. Oggier, A. Datta, *INFOCOM 2011*
A weakly linearized polynomial $p(X)$ over $\mathbb{F}_q$, $q = 2^m$, has the form

$$p(X) = \sum_{i=0}^{k-1} p_i X^{2^i}, \ p_i \in \mathbb{F}_q.$$ 

Let $a, b \in \mathbb{F}_{2^m}$ and let $p(X)$ be a weakly linearized polynomial given by $p(X) = \sum_{i=0}^{k-1} p_i X^{2^i}$. We have

$$p(a + b) = p(a) + p(b).$$
HSRC: Encoding

1. Take an object $o$ of length $M$:

   $$o = (o_1, \ldots, o_k), \quad o_i \in \mathbb{F}_{2^{M/k}}.$$

2. Take a linearized polynomial with coefficients in $\mathbb{F}_{2^{M/k}}$

   $$p(X) = \sum_{i=0}^{k-1} p_i X^{2^i},$$

   and encode the $k$ fragments $p_i = o_{i+1}, \ i = 0, \ldots, k - 1$.

3. Evaluate $p(X)$ in $n$ non-zero values $\alpha_1, \ldots, \alpha_n$ of $\mathbb{F}_{2^{M/k}}$ to get a
   $n$-dimensional codeword

   $$(p(\alpha_1), \ldots, p(\alpha_n)),$$

   and each $p(\alpha_i)$ is given to node $i$ for storage.
HSRC: Encoding Illustration

Data = Object

\[ p(X) = \sum_{i=0}^{k-1} p_i X^{2^i} \]

Encoding

with \( p_i = O_{i+1} \)

k blocks
Each of size M/k

n encoded blocks
HSRC: Decoding and Repair

1. *Decoding* is ensured by Lagrange interpolation.

\[
\alpha_i = \frac{M}{k} \sum_{j=1}^{M/k} \alpha_{ij} b_j, \quad \alpha_{ij} \in \mathbb{F}_2 \Rightarrow p(\alpha_i) = \frac{M}{k} \sum_{j=1}^{M/k} \alpha_{ij} p(b_j).
\]

Computational cost of a repair: XORs.
HSRC: Decoding and Repair

1. **Decoding** is ensured by Lagrange interpolation.

2. **Repair**: Express $\alpha_i$ in a $\mathbb{F}_2$-basis $B = \{b_1, \ldots, b_{M/k}\}$ of $\mathbb{F}_2^{M/k}$, then

$$
\alpha_i = \sum_{j=1}^{M/k} \alpha_{ij} b_j, \quad \alpha_{ij} \in \mathbb{F}_2 \Rightarrow p(\alpha_i) = \sum_{j=1}^{M/k} \alpha_{ij} p(b_j).
$$
HSRC: Decoding and Repair

1. **Decoding** is ensured by Lagrange interpolation.

2. **Repair**: Express $\alpha_i$ in a $\mathbb{F}_2$-basis $B = \{b_1, \ldots, b_{M/k}\}$ of $\mathbb{F}_{2^{M/k}}$, then

$$\alpha_i = \sum_{j=1}^{M/k} \alpha_{ij} b_j, \quad \alpha_{ij} \in \mathbb{F}_2 \Rightarrow p(\alpha_i) = \sum_{j=1}^{M/k} \alpha_{ij} p(b_j).$$

3. Computational cost of a repair: XORs.
HSRC: A toy example (I)

A data file \( o = (o_1, \ldots, o_{12}) \) of \( M = 12 \) bits is cut into \( k = 3 \) fragments \( (M/k = 4) \)

\[ o_1 = (o_1, \ldots, o_4), \quad o_2 = (o_5, \ldots, o_8), \quad o_3 = (o_9, \ldots, o_{12}) \in \mathbb{F}_2^4. \]
HSRC: A toy example (I)

- A data file $\mathbf{o} = (o_1, \ldots, o_{12})$ of $M = 12$ bits is cut into $k = 3$ fragments ($M/k = 4$)

  \[ \mathbf{o}_1 = (o_1, \ldots, o_4), \quad \mathbf{o}_2 = (o_5, \ldots, o_8), \quad \mathbf{o}_3 = (o_9, \ldots, o_{12}) \in \mathbb{F}_2^4. \]

- $\mathbb{F}_2^* = \langle w \rangle$, with $w^4 = w + 1$. Encode with the polynomial

  \[ p(X) = \sum_{i=1}^{4} o_i w^i X + \sum_{i=1}^{4} o_{i+4} w^i X^2 + \sum_{i=1}^{4} o_{i+8} w^i X^4. \]
HSRC: A toy example (I)

- A data file \( \mathbf{o} = (o_1, \ldots, o_{12}) \) of \( M = 12 \) bits is cut into \( k = 3 \) fragments \((M/k = 4)\)

\[ \mathbf{o}_1 = (o_1, \ldots, o_4), \quad \mathbf{o}_2 = (o_5, \ldots, o_8), \quad \mathbf{o}_3 = (o_9, \ldots, o_{12}) \in F_{2^4}. \]

- \( F_{2^4}^* = \langle w \rangle \), with \( w^4 = w + 1 \). Encode with the polynomial

\[ p(X) = \sum_{i=1}^{4} o_i w^i X + \sum_{i=1}^{4} o_{i+4} w^i X^2 + \sum_{i=1}^{4} o_{i+8} w^i X^4. \]

- For \( n = 7 \), evaluate \( p(X) \) at say \( 1, w, w^2, w^4, w^5, w^8, w^{10} \). We get:

\[ (p(1), p(w), p(w^2), p(w^4), p(w^5), p(w^8), p(w^{10})) \]
### HSRC: A toy example (II)

<table>
<thead>
<tr>
<th>missing fragment(s)</th>
<th>pairs to reconstruct missing fragment(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(1) )</td>
<td>((p(w), p(w^4));(p(w^2), p(w^8));(p(w^5), p(w^{10})))</td>
</tr>
<tr>
<td>( p(w) )</td>
<td>((p(1), p(w^4));(p(w^2), p(w^5));(p(w^8), p(w^{10})))</td>
</tr>
<tr>
<td>( p(w^2) )</td>
<td>((p(1), p(w^8));(p(w), p(w^5));(p(w^4), p(w^{10})))</td>
</tr>
<tr>
<td>( p(1) ) and ( p(w) )</td>
<td>((p(w^2), p(w^8)) \text{ or } (p(w^5), p(w^{10}))) for ( p(1) ) ((p(w^8), p(w^{10})) \text{ or } (p(w^2), p(w^5))) for ( p(w) )</td>
</tr>
<tr>
<td>( p(1) ) and ( p(w) ) and ( p(w^2) )</td>
<td>((p(w^5), p(w^{10}))) for ( p(1) ) ((p(w^8), p(w^{10}))) for ( p(w) ) ((p(w^4), p(w^{10}))) for ( p(w^2) )</td>
</tr>
</tbody>
</table>
A second instance of self-repairing code.

Preliminaries: Spreads

- Consider a vector space of dimension $m$ over $\mathbb{F}_q$, namely, a projective space $PG(m-1, q)$.
- Let $\mathcal{P}$ be a projective space. A $t$-spread of $\mathcal{P}$ is a set $S$ of $t$-dimensional subspaces of $\mathcal{P}$ which partitions $\mathcal{P}$.
Consider a vector space of dimension $m$ over $\mathbb{F}_q$, namely, a projective space $PG(m-1, q)$.

Let $\mathcal{P}$ be a projective space. A $t$-spread of $\mathcal{P}$ is a set $S$ of $t$-dimensional subspaces of $\mathcal{P}$ which partitions $\mathcal{P}$.

Theorem (André)

In $PG(m-1, q)$, a $t$-spread exists if and only if $t + 1 | m$. 
Suppose that \( t + 1 \mid m \). Consider the finite fields \( F_0 = \mathbb{F}_q \), \( F_1 = \mathbb{F}_{q^{t+1}} \) and \( F_2 = \mathbb{F}_{q^m} \).
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 $P = PG(m, q)$. The field $F_1$ is a $(t + 1)$-dimensional subspace of $V$ and hence a $t$-dimensional (projective) subspace of $P$. 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 $P$. 
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 $\mathcal{P} = \text{PG}(m, q)$. The field $F_1$ is a $(t + 1)$-dimensional subspace of $V$ and hence a $t$-dimensional (projective) subspace of $\mathcal{P}$.
Suppose that \( t + 1 \mid m \). 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 \( P = \text{PG}(m, q) \). The field \( F_1 \) is a \( (t + 1) \)-dimensional subspace of \( V \) and hence a \( t \)-dimensional (projective) subspace of \( P \).

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 \( P \).
PSRC: Encoding

1. For an object \( o \) of size \( M \), consider the finite field \( \mathbb{F}_{2^M} \).

2. Consider a \( t \)-spread \( S \) formed of \( t \)-dimensional subspaces of \( \mathcal{P} \) such that \( t + 1 | M \). Set \( \alpha = t + 1 \). Assign to each node an \( \mathbb{F}_2 \)-basis containing \( \alpha \) vectors. The number of storage nodes is (at most)

\[
    n = \frac{2^M - 1}{2^\alpha - 1}.
\]

3. The \( i \)th node will actually store

\[
    \{ ov_{i\alpha + 1}^T, \ldots, ov_{(i+1)\alpha} \}
\]

for a total storage of \( \alpha \).
Decoding is solving a system of linear equations.
PSRC: Decoding and Repair

1. **Decoding** is solving a system of linear equations.

2. **Repair** The $l$th node $N_l$ stores $\nu^l \mathbb{F}_2^{2\alpha}$, $l = 1, \ldots, n$. Let us assume this $l$th node fails, and a new comer $N_i$ joins. Contact the $j$th node $N_j$ such that $\nu^j = \nu^i + \nu^l$. By combining the data stored at node $N_i$ and $N_j$, we get

   \[ \nu^i \mathbb{F}_2^{2\alpha} \bigoplus (\nu^i + \nu^l) \mathbb{F}_2^{2\alpha} \]

   which contains $\nu^l \mathbb{F}_2^{2\alpha}$.

Lemma

*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 $N_j$.**
**PSRC: A toy example**

<table>
<thead>
<tr>
<th>node</th>
<th>basis vectors</th>
<th>data stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>$v_1 = (1000)$, $v_2 = (0110)$</td>
<td>${ o_1, o_2 + o_3 }$</td>
</tr>
<tr>
<td>$N_2$</td>
<td>$v_3 = (0100)$, $v_4 = (0011)$</td>
<td>${ o_2, o_3 + o_4 }$</td>
</tr>
<tr>
<td>$N_3$</td>
<td>$v_5 = (0010)$, $v_6 = (1101)$</td>
<td>${ o_3, o_1 + o_2 + o_4 }$</td>
</tr>
<tr>
<td>$N_4$</td>
<td>$v_7 = (0001)$, $v_8 = (1010)$</td>
<td>${ o_4, o_1 + o_3 }$</td>
</tr>
<tr>
<td>$N_5$</td>
<td>$v_9 = (1100)$, $v_{10} = (0101)$</td>
<td>${ o_1 + o_2, o_2 + o_4 }$</td>
</tr>
</tbody>
</table>
Outline

1. Coding for Distributed Networked Storage
2. Self-Repairing Codes: Definition and Constructions
3. Self-Repairing Codes: Analysis and Properties
Static resilience

- There is at least one pair to repair a node, for up to \((n - 1)/2\) simultaneous failures.
- Static resilience of a distributed storage system is the probability that an object stored in the system stays available without any further maintenance, even when a fraction of nodes become unavailable.
Static resilience: HSRC versus EC

Figure: Static resilience of self-repairing codes (SRC): Validation of analysis, and comparison with erasure codes (EC)
Static resilience: PSRC versus EC

Figure: Static resilience of self-repairing codes (HSRC): Comparison with erasure codes (EC)
More on Resilience: HSRC versus EC

Figure: Static resilience of self-repairing codes (HSRC): Comparison with erasure codes (EC)
More on Resilience: PSRC versus EC

Figure: Static resilience of self-repairing codes (HSRC): Comparison with erasure codes (EC)
Fast & parallel repairs using HSRC: A toy example

Consider:

- $(15,4)$ code, nodes storing $p(w^i)$ for $i = 0, 1, 2, 3, 4, 5, 6$ are missing
- Nodes have upload/download bandwidth limit: one block per time unit
Self-Repairing Codes: Analysis and Properties

Fast & parallel repairs using HSRC: A toy example

- Consider:
  - (15,4) code, nodes storing \( p(w^i) \) for \( i = 0, 1, 2, 3, 4, 5, 6 \) are missing
  - Nodes have upload/download bandwidth limit: one block per time unit
  - Possible pairs to repair each missing block:

<table>
<thead>
<tr>
<th>fragment</th>
<th>suitable pairs to reconstruct</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(1) )</td>
<td>( (p(w^1), p(w^9));(p(w^{11}), p(w^{12})) )</td>
</tr>
<tr>
<td>( p(w) )</td>
<td>( (p(w^7), p(w^{14}));(p(w^8), p(w^{10})) )</td>
</tr>
<tr>
<td>( p(w^2) )</td>
<td>( (p(w^7), p(w^{12}));(p(w^9), p(w^{11}));(p(w^{12}), p(w^{10})) )</td>
</tr>
<tr>
<td>( p(w^3) )</td>
<td>( (p(w^8), p(w^{13}));(p(w^{10}), p(w^{12})) )</td>
</tr>
<tr>
<td>( p(w^4) )</td>
<td>( (p(w^9), p(w^{14}));(p(w^{11}), p(w^{13})) )</td>
</tr>
<tr>
<td>( p(w^5) )</td>
<td>( (p(w^7), p(w^{13}));(p(w^{12}), p(w^{14})) )</td>
</tr>
<tr>
<td>( p(w^6) )</td>
<td>( (p(w^7), p(w^{10}));(p(w^8), p(w^{14})) )</td>
</tr>
</tbody>
</table>
Fast & parallel repairs using HSRC: A toy example

Consider:

- (15,4) code, nodes storing $p(w^i)$ for $i = 0, 1, 2, 3, 4, 5, 6$ are missing
- Nodes have upload/download bandwidth limit: one block per time unit

Possible pairs to repair each missing block:

<table>
<thead>
<tr>
<th>fragment</th>
<th>suitable pairs to reconstruct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(1)$</td>
<td>$(p(w^1), p(w^9)); (p(w^{11}), p(w^{12}))$</td>
</tr>
<tr>
<td>$p(w)$</td>
<td>$(p(w^7), p(w^{14})); (p(w^8), p(w^{10}))$</td>
</tr>
<tr>
<td>$p(w^2)$</td>
<td>$(p(w^7), p(w^{12})); (p(w^9), p(w^{11})); (p(w^{12}), p(w^{10}))$</td>
</tr>
<tr>
<td>$p(w^3)$</td>
<td>$(p(w^8), p(w^{13})); (p(w^{10}), p(w^{12}))$</td>
</tr>
<tr>
<td>$p(w^4)$</td>
<td>$(p(w^9), p(w^{14})); (p(w^{11}), p(w^{13}))$</td>
</tr>
<tr>
<td>$p(w^5)$</td>
<td>$(p(w^7), p(w^{13})); (p(w^{12}), p(w^{14}))$</td>
</tr>
<tr>
<td>$p(w^6)$</td>
<td>$(p(w^7), p(w^{10})); (p(w^8), p(w^{14}))$</td>
</tr>
</tbody>
</table>

A parallelized schedule:

<table>
<thead>
<tr>
<th>node</th>
<th>$p(w^0)$</th>
<th>$p(w^1)$</th>
<th>$p(w^2)$</th>
<th>$p(w^3)$</th>
<th>$p(w^4)$</th>
<th>$p(w^5)$</th>
<th>$p(w^6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time 1</td>
<td>$p(w^7)$</td>
<td>$p(w^8)$</td>
<td>$p(w^9)$</td>
<td>$p(w^{13})$</td>
<td>$p(w^{11})$</td>
<td>$p(w^{12})$</td>
<td>$p(w^{10})$</td>
</tr>
<tr>
<td>Time 2</td>
<td>$p(w^9)$</td>
<td>$p(w^{10})$</td>
<td>$p(w^{11})$</td>
<td>$p(w^8)$</td>
<td>$p(w^{13})$</td>
<td>$p(w^{14})$</td>
<td>$p(w^7)$</td>
</tr>
</tbody>
</table>
### Systematic Object Retrieval using PSRC: A toy example

<table>
<thead>
<tr>
<th>node</th>
<th>basis vectors</th>
<th>data stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_1)</td>
<td>(v_1 = (1000), \ v_2 = (0110))</td>
<td>({o_1, o_2 + o_3})</td>
</tr>
<tr>
<td>(N_2)</td>
<td>(v_3 = (0100), \ v_4 = (0011))</td>
<td>({o_2, o_3 + o_4})</td>
</tr>
<tr>
<td>(N_3)</td>
<td>(v_5 = (0010), \ v_6 = (1101))</td>
<td>({o_3, o_1 + o_2 + o_4})</td>
</tr>
<tr>
<td>(N_4)</td>
<td>(v_7 = (0001), \ v_8 = (1010))</td>
<td>({o_4, o_1 + o_3})</td>
</tr>
<tr>
<td>(N_5)</td>
<td>(v_9 = (1100), \ v_{10} = (0101))</td>
<td>({o_1 + o_2, o_2 + o_4})</td>
</tr>
</tbody>
</table>
Future/ongoing work

- Efficient decoding, other instances of SRC
- Implementation & integration in a distributed storage system
- Various systems/algorithmic issues: Topology optimized placement, repair scheduling
Wrap Up

- Design of codes for distributed networked storage
Wrap Up

- Design of codes for distributed networked storage
- Self-Repairing Codes
Wrap Up

- Design of codes for distributed networked storage
- Self-Repairing Codes
- New research topic in coding theory!
Q&A

More information:
http://sands.sce.ntu.edu.sg/CodingForNetworkedStorage/
Q&A

- More information: http://sands.sce.ntu.edu.sg/CodingForNetworkedStorage/
- Contact: {frederique, anwitaman}@ntu.edu.sg