This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

Survivable Distributed Storage with Progressive Decoding

Yunghsiang S. Han*, Soji Omiwade‡ and Rong Zheng‡
* Graduate Institute of Communication Engineering, National Taipei University, Taiwan, yshan@mail.ntpu.edu.tw
* Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan
‡ Department of Computer Science, University of Houston, Houston, TX 77204, {ooo00a,rzheng}@cs.uh.edu

***Abstract*—We propose a *storage-optimal and computation efficient primitive* to spread information from a single data source to a set of storage nodes, to allow recovery from both crash-stop and Byzantine failures. A progressive data retrieval scheme is employed, which retrieves minimal amount of data from live storage nodes. The scheme adapts the cost of successful data retrieval to the degree of errors in the system. Implementation and evaluation studies demonstrate comparable performance to that of a genie-aid decoding process.**

***Index Terms*—Network storage, Byzantine failures, Reed-Solomon code, Error-detection code**

I. Introduction

Many existing approaches to survivable storage assume crash-stop behaviors, i.e., a storage device becomes unavailable if failed (also called "erasure"). Solutions such as various RAID configurations [1] and their extensions are engineered for high read and write data throughput. Thus, typically low-complexity (replication or XOR-based) coding mechanisms are employed to recover from limited degree of erasure. We argue that Byzantine failures, where devices fail in arbitrary manner and cannot be trusted, are becoming more pertinent with the prevalence of cheap storage devices, software bugs and malicious attacks. Efficient encode and decode primitives that can detect data corruption and handle Byzantine failures serve as a fundamental building block to support higher level abstractions such as multi-reader multi-writer atomic register [2] in distributed systems.

In this paper, we provide a design and implementation of a novel progressive data retrieval mechanism that is *optimal* in storage and communication, and computationally efficient. It handles Byzantine failures in storage nodes gracefully as the probability of failures increases. The rest of the paper is organized as follows. Background and related work is given in Section II. Details of the incremental RS decoding algorithm are provided in Section III. An analysis of our coding and communication complexity is provided in Section IV. Evaluation results are presented in Section V. Finally, we conclude the paper in Section VI.

II. Background

Our redunancy scheme is based on (n, k) MDS codes, which can recover from any v errors if $v \leq t - \lceil \frac{s}{2} \rceil$, where s is the number of erasures (unretrievable symbols) and $t = (n-k)/2$ is the error-erasure correcting capability of the code. Without loss of generality, assume that a file is divided into k symbols, encoded into n symbols and stored at n storage nodes, one symbol per storage node.

Encoding: Let the sequence of k information symbols in $GF(2^m)$ be $\boldsymbol{u} = (u_0, u_1, \ldots, u_{k-1})$ and $u(x)$ be the information polynomial of $\boldsymbol{u}$ represented as $u(x) = \sum_{i=0}^{k-1} u_i x^i$. The codeword polynomial, $c(x)$, corresponding to $u(x)$ can be derived by multiplying $u(x)$ with $g(x)$, where $g(x)$ is a generator polynomial of an RS code [3].

Decoding: The decoding process of RS codes is more complex. A complete description of decoding of RS codes can be found in [4].

Let $r(x)$ be the received polynomial and $r(x) = c(x) + e(x) + \gamma(x) = c(x) + \lambda(x)$, where $e(x) = \sum_{j=0}^{n-1} e_j x^j$, $\gamma(x) = \sum_{j=0}^{n-1} \gamma_j x^j$ and $\lambda(x) = \sum_{j=0}^{n-1} \lambda_j x^j = e(x) + \gamma(x)$ are the error, erasure and errata polynomials, respectively. Note that $g(x)$ and (hence) $c(x)$ have $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+2t-1}$ as roots. This property is used to determine the error locations and recover the information symbols.

The basic procedure of RS decoding is shown in Figure 1. The last step of the decoding procedure involves solving a linear set of equations, and can be made efficient by the use of Vandermonde generator matrices [5]. Several XOR-based erasure codes (in a field of GF(2)) have been used in storage systems [6], [7]. But the gain in computation efficiency of these codes is achieved by trading off fault tolerance. Also, RAID-6 systems can recover from the loss of exactly two disks but cannot handle Byzantine failures.

III. Incremental RS Decoding

In this section, we present the incremental RS decoding algorithm. Compared to classic RS decoding, it utilizes intermediate computation results and decodes incrementally as more symbols become available.

A. The basic algorithm

Given the received coded symbols $(r_0, r_1, \ldots, r_{n-1})$ with erasures set to be zero, the generalized syndrome polynomial $S(x)$ can be calculated as [8],

$$S(x) = \sum_{j=0}^{n-1} r_j \alpha^{jb} \frac{T(x) - T(\alpha^j)}{x - \alpha^j} = \sum_{j=0}^{n-1} \lambda_j \alpha^{jb} \frac{T(x) - T(\alpha^j)}{x - \alpha^j}, \quad (1)$$

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

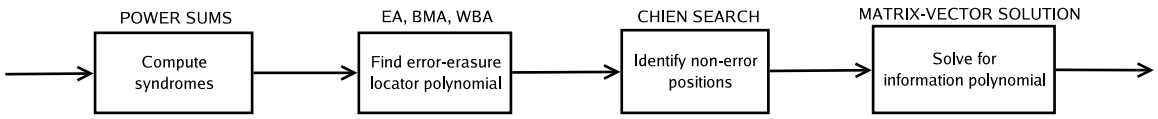


Fig. 1. Block diagram of RS decoding. The texts on top of the boxes correspond to existing algorithms.

where $T(x)$ is an arbitrary polynomial with degree $(n-k)$ and α is a primitive element in $GF(2^m)$. Assume that v errors occur in unknown locations $j_1, j_2, \ldots, j_v$ and s erasures in known locations $m_1, m_2, \ldots, m_s$ of the received polynomial. Then $e(x) = \sum_{i=1}^{v} e_{j_i} x^{j_i}$ and $\gamma(x) = \sum_{i=1}^{s} \gamma_{m_i} x^{m_i}$ where e_{j_ℓ} is the value of the ℓ-th error, $\ell = 1, \cdots, v$, and γ_{m_ℓ} is the value of the ℓ-th erasure, $\ell = 1, \cdots, s$.[1] The decoding process is to find all j_ℓ, e_{j_ℓ}, m_ℓ, and γ_{m_ℓ}. Let $\mathbf{E} = \{j_1, \cdots, j_v\}$, $\mathbf{M} = \{m_1, \cdots, m_s\}$, and $\mathbf{D} = \mathbf{E} \cup \mathbf{M}$. Clearly, $\mathbf{E} \cap \mathbf{M} = \emptyset$. It has been shown that a key equation for decoding is

$$\Lambda(x)S(x) = \Psi(x)T(x) + \Omega(x) \ , \tag{2}$$

where

$$\begin{aligned} \Lambda(x) &= \prod_{j \in \mathbf{D}} (x - \alpha^j) = \prod_{j \in \mathbf{E}} \left(x - \alpha^j\right) \prod_{j \in \mathbf{M}} \left(x - \alpha^j\right) \\ &= \Lambda_{\mathbf{E}}(x)\Lambda_{\mathbf{M}}(x) \qquad (3) \\ \Psi(x) &= \sum_{j \in \mathbf{D}} \lambda_j \alpha^{jb} \prod_{\substack{i \in \mathbf{D} \\ i \neq j}} \left(x - \alpha^i\right) \qquad (4) \\ \Omega(x) &= -\sum_{j \in \mathbf{D}} \lambda_j \alpha^{jb} T(\alpha^j) \prod_{\substack{i \in \mathbf{D} \\ i \neq j}} \left(x - \alpha^i\right) \ . \qquad (5) \end{aligned}$$

If $2v + s \leq n - k + 1$, then (2) has a unique solution $\{\Lambda(x), \Psi(x), \Omega(x)\}$. Instead of solving (2) by either the Euclidean or Berlekamp-Massey algorithm we introduce a reduced key equation [8] that can be solved by the Welch-Berlekamp (W-B) algorithm [4]. It will be demonstrated that by using W-B algorithm and the reduced key equation, the complexity of decoding can be reduced drastically. Let $\mathbf{T} = \{j | T(\alpha^j) = 0\}$. Let a set of coordinates $\mathbf{U} \subset \{0, 1, \ldots, n-1\}$ be defined by $\mathbf{U} = \mathbf{M} \cap \mathbf{T}$. A polynomial $\Lambda_{\mathbf{U}}(x)$ is then defined by $\Lambda_{\mathbf{U}}(x) = \prod_{j \in \mathbf{U}} \left(x - \alpha^j\right)$, which is known for the receiver since $T(x)$ and $\mathbf{M}$ are both known. Since $\Lambda_{\mathbf{U}}(x)$ divides both $\Lambda(x)$ and $T(x)$, according to (2), it also divides $\Omega(x)$. Hence, we have the following reduced key equation:

$$\tilde{\Lambda}(x)S(x) = \Psi(x)\tilde{T}(x) + \tilde{\Omega}(x) \ , \tag{6}$$

where $\Lambda(x) = \tilde{\Lambda}(x)\Lambda_{\mathbf{U}}(x)$, $T(x) = \tilde{T}(x)\Lambda_{\mathbf{U}}(x)$ and $\Omega(x) = \tilde{\Omega}(x)\Lambda_{\mathbf{U}}(x)$ Note that $\tilde{\Lambda}(x)$ is still a multiple of the error location polynomial $\Lambda_{\mathbf{E}}(x)$. The reduced key equation can have a unique solution if

$$\deg(\tilde{\Omega}(x)) < \deg(\tilde{\Lambda}(x)) < \frac{n - k + 1 + s}{2} - |\mathbf{U}| \ , \tag{7}$$

[1]Since the received values in the erased positions are zero, $\gamma_{m_\ell} = -c_{m_\ell}$ for $\ell = 1, \cdots, s$.

where $\deg(\cdot)$ is the degree of a polynomial and $|\mathbf{U}|$ is the number of elements in set $\mathbf{U}$.

For all $j \in \mathbf{T} \backslash \mathbf{U}$,[2] by (6), we have

$$\tilde{\Lambda}(\alpha^j)S(\alpha^j) = \tilde{\Omega}(\alpha^j) \tag{8}$$

since $\tilde{T}(\alpha^j) = 0$. Note that α^j is a sampling point and $S(\alpha^j)$ the sampled value for (8). The unique solution $\{\tilde{\Lambda}(x), \tilde{\Omega}(x)\}$ can then be found by the W-B algorithm with time complexity $O((n - k - |\mathbf{U}|)^2)$ [4]. Once all coefficients of the errata polynomial are found, the error locations j_ℓ can be determined by successive substitution through Chien search [9]. When the solution of (6) is obtained, the errata values can be calculated. Since there is no need to recover the errata values in our application we omit the calculations. In summary, there are three steps in the decoding of RS codes that must be implemented. First, the sampled values of $S(\alpha^j)$ for $j \in \mathbf{T} \backslash \mathbf{U}$ must be calculated. Second, the W-B algorithm is performed based on the pairs $\left(\alpha^j, S(\alpha^j)\right)$ in order to obtain a valid $\tilde{\Lambda}(x)$. If a valid $\tilde{\Lambda}(x)$ is obtained, then error locations are found by Chien search; otherwise, decoding failure is reported.

B. Incremental computation of $S(x)$, $\tilde{\Lambda}(x)$, $\tilde{\Omega}(x)$

Let us choose $T(x) = \prod_{i=0}^{v-k-1} (x - \alpha^{m_i})$, where m_i are those corresponding positions of missing data symbols after the data collector has retrieved encoded symbols from k storage nodes. In the decoding process, these are erased positions before the first iteration of error-erasure decoding. Let $\mathbf{U}_0 = \{m_0, \ldots, m_{n-k-1}\}$. It has been proven that the generator polynomial of a Vandermonde-matrix based RS code [3] has $\alpha^{n-k}, \alpha^{n-k-1}, \ldots, \alpha$ as roots. The error-erasure decoding algorithm is mainly based on W-B algorithm which is an iterative rational interpolation method.

In the ℓth iteration, ℓ errors are assumed in the data and the number of erasures is $n - k - 2\ell$. Let $(j_1^{(\ell)} + 1)$th and $(j_2^{(\ell)} + 1)$th nodes be the two storage nodes just accessed in the ℓth iteration. Let $\mathbf{U}_\ell = \mathbf{U}_{\ell-1} \backslash \{j_1^{(\ell)}, j_2^{(\ell)}\}$. Based on $\mathbf{U}_\ell$ the W-B algorithm will find $\tilde{\Lambda}^{(\ell)}(x)$ and $\tilde{\Omega}^{(\ell)}(x)$ which satisfy

$$\tilde{\Lambda}^{(\ell)}(\alpha^j)S^{(\ell)}(\alpha^j) = \tilde{\Omega}^{(\ell)}(\alpha^j) \text{ for all } j \in \mathbf{U}_0 \backslash \mathbf{U}_\ell \ ,$$

where $S^{(\ell)}(x)$ is the generalized syndrome with $r_i = 0$ for all $r_i \in \mathbf{U}_\ell$. It has been shown that $\deg(\tilde{\Lambda}^{(\ell)}(x)) > \deg(\tilde{\Omega}^{(\ell)}(x))$ for any ℓ by a property of W-B algorithm. Thus, if $\deg(\tilde{\Lambda}^{(\ell)}(x)) < \frac{n-k+1+|\mathbf{U}_\ell|}{2} - |\mathbf{U}_\ell| = \ell + 1/2$, then the unique solution will exist due to (7). By the definition of generalized syndrome polynomial in (1), for $i \in \mathbf{U}_0 \backslash \mathbf{U}_\ell$, we have

[2]$\backslash$ is the set difference.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

$$S^{(\ell)}(\alpha^i) \quad = \quad \sum_{\substack{j=0 \\ j \notin \mathbf{U}_0}}^{n-1} \frac{F_j}{\alpha^j - \alpha^i} + r_i \alpha^i T'(\alpha^i) \ , \tag{9}$$

where $T'(x)$ is the derivative of $T(x)$ and $F_j = r_j \alpha^j T(\alpha^j)$. Details of the derivation of (9) are available [3]. Note that $T'(\alpha^i) = \prod_{\substack{j \in \mathbf{U}_0 \\ m_j \neq i}} \left(\alpha^i - \alpha^{m_j}\right)$. It is easy to see that $S^{(\ell)}(\alpha^i)$ is not related to any r_j, where $j \in \mathbf{U}_0$ and $j \neq i$. Hence, $S^{(\ell-1)}(\alpha^i) = S^{(\ell)}(\alpha^i)$ for all $i \in \mathbf{U}_0 \backslash \mathbf{U}_{\ell-1}$. This fact implies that all sampled values in previous iterations can be directly used in current iteration of the W-B algorithm.

Define $\text{rank}[N(x), W(x)] = \max[2\deg(W(x)), 1 + 2\deg(N(x))]$. The incremental RS decoding algorithm is described in Algorithm 1. Upon success, the incremental RS decoding algorithm returns k non-error symbols. The procedure will report failure either as the result of mismatched degree of the error locator polynomial, or insufficient number of roots found by Chien search (Line 2). In both cases, no further erasure decoding is required. This reduces the decoding computation time.

Algorithm 1: Incremental RS Decoding $IncrRSDecode$

init : Calculate F_j given in (9) for all $j \notin \mathbf{U}_0$.
$\ell \leftarrow 0$; $\tilde{\Lambda}^{(0)}(x) \leftarrow 1$;
$\tilde{\Omega}^{(0)}(x) \leftarrow 0$; $\Phi^{(0)}(x) \leftarrow 0, \Theta^{(0)}(x) \leftarrow 1$.
input : stage l, two new symbols at the $(j_1^{(\ell)}+1)$th, and $(j_2^{(\ell)}+1)$th nodes
output: FAIL or non-error symbols $\boldsymbol{r}$
begin
  foreach $i = 1, 2$ **do**
1 $x_i^{(\ell)} \leftarrow \alpha^{j_i^{(\ell)}}$ and $y_i^{(\ell)} \leftarrow S^{(\ell)}(x_i^{(\ell)})$
  end
  for $i = 1$ **to** 2 **do**
    $b_i^{(\ell-1)} \leftarrow \tilde{\Omega}^{(\ell-1)}(x_i^{(\ell)}) - y_i^{(\ell)} \tilde{\Lambda}^{(\ell-1)}(x_i^{(\ell)})$;
    if $b_i^{(\ell-1)} = 0$ **then**
      $\tilde{\Lambda}^T(x) \leftarrow \tilde{\Lambda}^{(\ell-1)}(x)$; $\tilde{\Omega}^T(x) \leftarrow \tilde{\Omega}^{(\ell-1)}(x)$;
      $\Theta^T(x) \leftarrow (x - x_i^{(\ell)})\Theta^{(\ell-1)}(x)$;
      $\Phi^T(x) \leftarrow (x - x_i^{(\ell)})\Phi^{(\ell-1)}(x)$
    else
      $a_i^{(\ell-1)} \leftarrow \Theta^{(\ell-1)}(x_i^{(\ell)}) - y_i^{(\ell)} \Phi^{(\ell-1)}(x_i^{(\ell)})$;
      $\Theta^T(x) \leftarrow (x - x_i^{(\ell)})\tilde{\Omega}^{(\ell-1)}(x)$;
      $\Phi^T(x) \leftarrow (x - x_i^{(\ell)})\tilde{\Lambda}^{(\ell-1)}(x)$;
      $\tilde{\Omega}^T(x) \leftarrow b_i^{(\ell-1)}\Theta^{(\ell-1)}(x) - a_i^{(\ell-1)}\tilde{\Omega}^{(\ell-1)}(x)$;
      $\tilde{\Lambda}^T(x) \leftarrow b_i^{(\ell-1)}\Phi^{(\ell-1)}(x) - a_i^{(\ell-1)}\tilde{\Lambda}^{(\ell-1)}(x)$.
    end
    if *rank*$[\tilde{\Omega}^T(x), \tilde{\Lambda}^T(x)] >$ *rank*$[\Theta^T(x), \Phi^T(x)]$ **then**
      swap $[\tilde{\Omega}^T(x), \tilde{\Lambda}^T(x)] \leftrightarrow [\Theta^T(x), \Phi^T(x)]$.
    end
    if $i = 1$ **then**
      $\tilde{\Omega}^{(\ell-1)}(x) \leftarrow \tilde{\Omega}^T(x)$; $\tilde{\Lambda}^{(\ell-1)}(x) \leftarrow \tilde{\Omega}^T(x)$;
      $\Theta^{(\ell-1)}(x) \leftarrow \Theta^T(x)$, $\Phi^{(\ell-1)}(x) \leftarrow \Phi^T(x)$;
    else
      $\tilde{\Omega}^{(\ell)}(x) \leftarrow \tilde{\Omega}^T(x)$; $\tilde{\Lambda}^{(\ell)}(x) \leftarrow \tilde{\Omega}^T(x)$;
      $\Theta^{(\ell)}(x) \leftarrow \Theta^T(x)$; $\Phi^{(\ell)}(x) \leftarrow \Phi^T(x)$.
    end
  end
  if $\deg(\tilde{\Lambda}^{(\ell)}(x)) \neq \ell$ **then**
    return FAIL;
  end
  NumErrorLoc = ChienSearch($\tilde{\Lambda}^{(\ell)}(x)$).
2 **if** $NumErrorLoc > n - k \parallel NumErrorLoc \neq \deg(\tilde{\Lambda}^{(\ell)}(x))$ **then**
    return FAIL;
  end
  return k non-error symbols $\boldsymbol{r}$;
end

IV. Complexity Analysis

A. Computation complexity of decoding

In the subsequent complexity analysis, the worst case is assumed, namely, no failure on decoding is reported in Algorithm 1 (Line 2), and the algorithm runs to completion.

One polynomial division is performed in CRC checking. Since the dividend is of degree $T-1$ and the divider is of degree r, the computation complexity of CRC checking is $O(Tr)$.

Let v be the number of errors when the decoding procedure is completed. In the ℓth iteration, ℓ errors are assumed in the data and the number of erasures is $n-k-2\ell$. We first need to calculate two syndrome values. This can be obtained by the F_j calculated initially. For instance, in the first iteration, according to (9), the computation complexity is of $O(k(n-k))$ since there are k F_j's to be calculated and each is a product of $n-k$ terms. In the next iteration, two more symbols are added to (1). Hence, the updated syndrome values can be obtained by an extra $O(k)+O(n-k)$ computations. To find the error-locator polynomial, the W-B algorithm is performed two steps in each iteration with complexity $O(\ell)$. Since we only consider software implementation, the Chien search can be replaced by substituting a power of α into the error-locator polynomial. It needs to test for at most $k+\ell$ positions to locate k non-error positions such that it takes $O((k+\ell)\ell)$ computations. Finally, inversion of Vandermonde matrix $\hat{G}$ requires $O(k\log^2(k))$ time [10]. In summary, the computation in the ℓth iteration for $\ell > 1$ is $L_v(\ell) = O(k\log^2 k) + O(n-k) + O(k\ell + \ell^2)$. Counting for v iterations and the complexity of calculating F_j we have $O(vk\log^2 k)+O(k(n-k))+O(v^2k)+O(v(n-k))+O(v^3)$. Note the computation complexity is measured by finite field multiplications, which is equivalent to m^2 bit exclusive-ORs. Since the correctable number of errors v is at most $(n-k)/2$, the decoding complexity is at most $O(k(n-k)^2)$. For small v, The second term $O(k(n-k))$ dominates, which corresponds to syndrome computation.

B. Average communication cost of decoding

In this section, we provide a probabilistic analysis of the cost of communication by determining the number of stages the algorithm needs to take, and the probability of successful execution. Given n storage nodes and (n,k) RS codes, it is easy to see that the fewest number of storage nodes to be accessed in the proposed scheme is k and the most is n. We assume that the CRC checking can always detect an error if it occurs. Without loss of generality, we assume that all failures are Byzantine failures. s crash-stop failures can be easily modeled by replacing n with $n-s$. An important metric of the decoding efficiency is the average number of accessed storage

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

TABLE I
PERFORMANCE COMPARISON BETWEEN DEC AND INCRRSDECODE.

	DEC	IncrRSDecode
Storage	n	n
Code construction	$n \ln k$	n
Retrieving 1 symbol	k	1
Retrieving all symbols	k	k
Encoding	$T_0^2 n \ln k$	$kTmn$
Decoding[1]	$T_0^2 O(k^3)$	$f(v,n,k)$
Detect error	no	yes
Correct error	no	yes
Guarantee type	probabilistic	deterministic

[a] v errors assumed in the decoding procedure. If $v = 0$, then $f(v,n,k) = m^2 O(k \log^2 k)$; otherwise, $f(v,n,k) = m^2(O(vk \log^2 k) + O(k(n-k)) + O(v^2 k) + O(v(n-k)) + O(v^3))$.

nodes when the probability of compromising each storage node is p. Failure to recover data correctly may occur in two cases. First, $v > n - k$, i.e., there are insufficient number of healthy storage nodes. Second, $\lfloor \frac{n-k}{2} \rfloor < v < n - k$, in which the sequence of accessing determines the outcome (success of failure) of the decoding process. For example, if the first v nodes accessed are all compromised nodes, correct decoding is impossible. In both cases, the decoding algorithm stops after n accesses and declares a failure. The communication cost is n. The main result is summarized in the following theorem.

Theorem 1: With the progressive data retrieval scheme, the average number of accesses is given in Eq. (10). Eq. (11) gives the probability of successful decoding.
The proof [3] is omitted due to space limitations.

C. Comparing to decentralized erasure coding

Table I provides a quantitative comparison between the cost of decentralized erasure codes (DEC) [11] and our proposed scheme (IncrRSDecode). DEC uses randomized linear codes, and each data node routes its packet to $\frac{n}{k} \ln(k)$ storage nodes. In Table I, T_0 is the number of bits in a file. T is the total number of bits after a CRC is added. T is divided into groups of size k symbols in the proposed scheme. From Table I, we see that the proposed scheme outperforms DEC for most metrics. A detailed comparison can be found in [3].

V. EVALUATION

We have implemented the proposed and baseline algorithms in C. Evaluations are done on a desktop PC with a 2.66GHz Intel Xeon CPU, 4096 KB cache and available RAM of 2GB. Three algorithms are considered.

- *BMA* implements the Berlekamp-Massey (BMA) algorithm [4] for RS decoding. Similar to a progressive incremental algorithm [3], BMA *progressively* retrieves data from each storage node and performs decoding until the decoded symbols passes the CRC checks or failure is declared. However, decoding cannot be performed incrementally.
- *BMA-genie* knows *a priori* how many symbols are needed to successfully decode. BMA-genie decodes only once after retrieving sufficient number of symbols. Note that BMA-genie is impossible to implement in practice, and is included for comparison purposes only.
- *IncrRSDecode* implements the proposed progressive data retrieval scheme with the incremental decoding algorithm.

In all of our experiments illustrated in Figure 2, a randomly generated message of size $8k$ bytes is first partitioned into k information symbols and then encoded into $n = 1023$ coded symbols of length 10230 bits. Thus, the field size is $2^{10} = 1024$. A stored symbol is corrupted with an error probability, p, independently. Each point in the figure is an average of 50 runs.

Total computation time: Figure 2(a) illustrates the average computation time–in log scale–spent in decoding. Note that IncrRSDecode is quite efficient, since it utilizes intermediate results from previous iterations. Up to 20 times speed-up is attained, relative to classic RS decoding. Not only is our scheme efficient, but it is also optimal in storage and communication. By optimal, we refer to minimal amount of data retrieval. From Figure 2(a), we observe that the BMA and IncrRSDecode computation time increases as p increases. But the rate of increment in IncrRSDecode is much slower. When k is small or the redundancy is higher as in Figure 2(a), IncrRSDecode is faster than the genie-aided BMA. This is because in the genie-aided BMA, the computations of erasure polynomials (with $O((n-k)^2)$) dominate the decoding time in the case of small number of errors. In contrast, in IncrRSDecode, no erasure polynomials are computed.

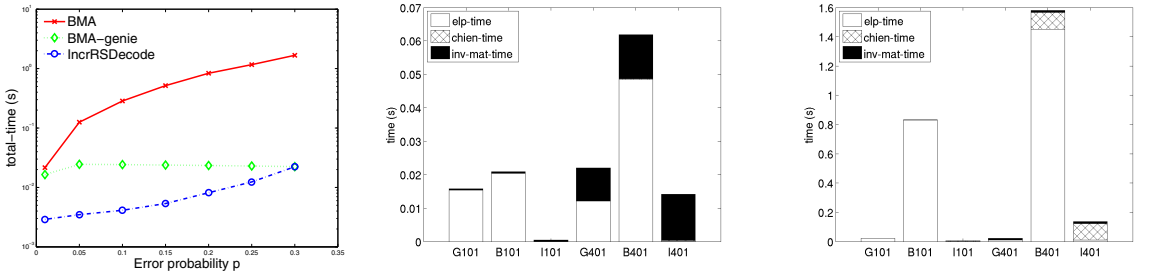
Breakdown of the decoding computation: We break down the decoding computation time to understand the dominant operations in the algorithms as well as how the time spent in each stage of the algorithms changes as the error probability increases. The break down includes the time to find the error-locator polynomial (*elp-time*), find the error locations (*chien-time*) and solve for the information polynomial (*inv-mat-time*). In Figure 1, the 1st and 2nd block shows the elp-time, and the 3rd and 4th blocks give chien-time and inv-mat-time, respectively.

When the error probability is low (Figure 2(b)), computation of error location polynomials appears to dominate for small k, while the matrix inversion time becomes significant when k is large. In our implementation, the cost of matrix inversion is quadratic in the number of symbols decoded. Also, Chien search is asymptotically the most time consuming procedure but is performed quite fast. When the error probability is high, computing error location polynomials dominates in IncrRSDecode.

Comparing Figure 2(b) and 2(c), we observe that the computation time in matrix inversion is almost negligible (on the order of tens of milliseconds) in BMA and IncrRSDecode, and is comparable to that in BMA-genie. This is because even though there are more errors with larger p, the decoding algorithm is likely to fail in or before Chien search (e.g., Algorithm 1 (Line 2)). Thus, in most cases, BMA and IncrRSDecode perform matrix inversion once.

$$\begin{aligned}\bar{N}(n,k) &= \sum_{v=0}^{n-k}\binom{n}{v}p^v(1-p)^{n-v}\sum_{i=0}^{\min(v,\lfloor\frac{n-k}{2}\rfloor,n-v-k)}(k+2i)\frac{\binom{n-v}{i+k-1}\binom{v}{i}}{\binom{n}{2i+k-1}}\times\frac{k}{i+k}\times\frac{n-v-(i+k-1)}{n-(2i+k-1)}\\ &+ \sum_{v=0}^{n-k} n\binom{n}{v}p^v(1-p)^{n-v}\left(1-\sum_{i=0}^{\min(v,\lfloor\frac{n-k}{2}\rfloor,n-v-k)}\frac{\binom{n-v}{i+k-1}\binom{v}{i}}{\binom{n}{2i+k-1}}\times\frac{k}{i+k}\times\frac{n-v-(i+k-1)}{n-(2i+k-1)}\right)\\ &+ \sum_{v=n-k+1}^{n} n\binom{n}{v}p^v(1-p)^{n-v}\ . \end{aligned} \tag{10}$$

$$Pr_{suc}(n,k) = \sum_{v=0}^{n-k}\binom{n}{v}p^v(1-p)^{n-v}\sum_{i=0}^{\min(v,\lfloor\frac{n-k}{2}\rfloor,n-v-k)}\frac{\binom{n-v}{i+k-1}\binom{e}{i}}{\binom{n}{2i+k-1}}\times\frac{k}{i+k}\times\frac{n-v-(i+k-1)}{n-(2i+k-1)}\ . \tag{11}$$



(a) Comparing to BMA; $k = 101$ (b) IncrRSDecode breakdown; $p = 1\%$ (c) IncrRSDecode breakdown; $p = 20\%$

Fig. 2. Decoding performance of IncrRSDecode

VI. Conclusions

In this paper, we developed a solution using RS codes to spread information distributedly and redundantly for the handling of Byzantine failures. The data retrieval procedure is carried out in a progressive manner such that the communication cost is minimized while intermediate computation results can be utilized to greatly reduce computation cost. In another work, we develop an analytical model to evaluate the communication cost of our incremental data retrieval scheme [3]. That analysis agrees with the experimental results in this paper.

Note that the proposed scheme guarantees latency of real time applications by collecting more data in each data retrieval, depending on the probability that a storage node is Byzantine. For many real time systems, energy minimization is critical, making our scheme suitable for energy constrained devices. Also, our scheme is desirable under high data rates as it minimizes communication costs by retrieving only necessary symbols.

Acknowledgment

Han's work was supported by the National Science Council of Taiwan, under grants NSC 96-2221-E-305-002-MY3 and his visit to LIVE lab at University of Texas at Austin. Omiwade and Zheng's work is supported in part by NSF CNS 0546391.

References

[1] "RAID, Redundant Array of Independent Disks," http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks.

[2] G. R. Goodson, J. J. Wylie, G. R. Ganger, and M. K. Reiter, "Efficient byzantine-tolerant erasure-coded storage," in *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*, 2004, p. 135.

[3] Y. S. Han, S. Omiwade, and R. Zheng, "Survivable distributed storage with progressive decoding," Technical Report UH-CS-09-17, Department of Computer Science, University of Houston, 2009.

[4] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Hoboken, NJ: John Wiley & Sons, Inc., 2005.

[5] H. William, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The art of scientific computing*, Cambridge university press New York, NY, USA, 1988.

[6] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of the 26th IEEE INFOCOM*, 2007, pp. 6–12.

[7] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," in *FAST '09: Proccedings of the 7th conference on File and storage technologies*, 2009, pp. 253–265.

[8] K. Araki, M. Takada, and M. Morii, "On the efficient decoding of Reed-Solomon codes based on GMD criterion," in *Proc. of the International Symposium on Multiple-Valued Logic*, Sendai, Japan, May 1992, pp. 138–145.

[9] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 2^{rd} edition, 2004.

[10] I. Gohberg and V. Olshevsky, "Fast algorithms with preprocessing for matrix-vector multiplication problem," *J. Complexity*, vol. 10, pp. 411–427, December 1994.

[11] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2809–2816, June 2006.