Data Survivability for WSNs via Decentralized Erasure Codes

Louai Al-Awami
Department of Electrical
& Computer Engineering
Queen's University
Kingston, Ontario, Canada
Email: louai@cs.queensu.ca

Hossam Hassanein
School of Computing
Queen's University
Kingston, Ontario, Canada
Email: hossam@cs.queensu.ca

***Abstract*—We study the problem of data survivability in WSNs. We propose a coding scheme that achieves Data survivability through the use of Decentralized Erasure Codes (DEC) which provide many advantages including flexibility, scalability, and decentralized implementation. We quantify the redundancy required in terms of storage nodes and the duplicated source data to achieve data survivability under a given erasure rate. When applied to WSNs, the proposed solution is designed to take into consideration the specifics of WSNs such as energy and resource constraints. The performance of the proposed scheme is based on established theoretical results and is verified by means of extensive simulations.**

***Index Terms*—WSN, Decentralized Erasure Codes, Survivability, Energy Efficiency, Network Coding**

I. Introduction

Since their introduction in late 1990's [1], Fountain codes have been spiking an increasing interest in the research community. The main attracting attribute of this family of codes is that they are rateless, meaning they do not have a fixed rate associated with them a priori. Hence, compared to ordinary erasure codes such as Reed-Solomen [2], rateless codes can adapt to any given erasure channel with an associated erasure probability p_e.

Given a set of k *native* data blocks $B = \{b_1, b_2, ..., b_k\}$ and a probability distribution $\rho(k)$, the encoder of a Fountain code samples $\rho(k)$ for a value $1 \leq d_i \leq k$, at the ith step. Then, it uniformly selects d_i random data blocks from B and xor's the blocks together under the mathematics of $\mathbb{F}_2$ generating an *encoded* block e_i. d_i is referred to as the *degree* of the encoded block e_i. Similarly, $\rho(k)$ is called the *degree distribution*. In addition to the encoded block, a k-dimensional binary *encoding* vector $G_i = \{g_{i1}, g_{i2}, ..., g_{ik}\}$ is appended to e_i; where every entry g_{ij} is set to 1 if b_j was used to construct e_i and 0 otherwise. g_{ij} is referred to as the *encoding coefficient*. Let $E = \{e_1, e_2, ..., e_n\}$ and G be the set of encoded blocks and encoding vectors, respectively. The decoder on the receiving side, keeps receiving encoded blocks until solving the system of linear equations $E_{1\times n} = B_{1\times k}G_{k\times n}$, for B. The number of packets required for decoding beyond k is referred to as *code overhead*. Generally, the decoder requires $n = (1+\epsilon)k$ encoded blocks to recover all native data blocks, where $\epsilon > 0$.

The key design aspect of rateless codes is the degree distribution $\rho(k)$. In [3], M. Luby proposed LT codes using the *Robust Soliton Distribution*. For a probability of successful decoding $1-\delta$, LT codes require an overhead of only $\mathbf{O}(\sqrt{k}\log_2(k/\delta))$ with decoding complexity of $\mathbf{O}(k\log_2(k/\delta))$. Raptor codes [4], on the other hand, achieves a linear encoding and decoding at the expensive of extra overhead ($\mathbf{O}(k)$) using the idea of pre-coding.

Despite their advantages, Fountain codes are difficult to implement when the source data is decentralized. Therefore, A. G. Dimakis proposed Decentralized Erasure Codes (DEC) in [5]. DEC work as seen in Figure 1, given a network of k source nodes and n storage nodes, where $k < n$, each source node j generates a single data block b_j and forwards it to m storage nodes that are selected uniformly at random. Upon receiving the data blocks, the storage node i generates a random coefficient g_j drawn from a finite field $\mathbb{F}_q$ for every data block received and combines the received blocks as follows:

$$e_i = g_1.b_1 \oplus g_2.b_2 \oplus ... \oplus g_k.b_k$$

Note that every storage node will receive a different set of data packets, and also that $g_j = 1$ if b_j was included in the encoding at the current storage node and 0 otherwise. The storage node then stores the *encoded block* e_i in addition to the a k-dimensional vector of random coefficients $G_i = \{g_1, g_2, ..., g_k\}$ used for encoding. To retrieve the original blocks, the decoder needs to collect $(1+\epsilon)k$ encoded blocks to solve for B in the system of linear equations $E_{1\times n} = B_{1\times k}.G_{k\times n}$.

Despite their obvious similarities, Fountain codes and DEC are quite different. Whereas d_i, the degree of each coded block can be generated exactly to match $\rho(k)$ in Fountain codes, DEC have no means of controlling the distribution of the degree of the encoded packets since the generation of encoded blocks is distributed. In a distributed storage setup, DEC generates different overhead on the encoder side compared to the overhead needed by the decoder. Therefore, we define

978-1-4577-1379-8/12/$26.00

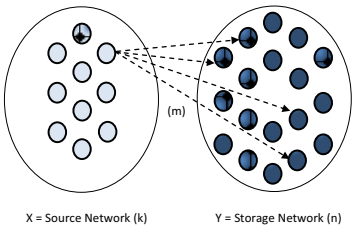


Fig. 1. Source and Storage Networks ($k = 10$, $n = 18$, $m = 4$, and $s = 0.8$).

α and β to be the Encoding Overhead (EO) and Decoding Overhead (DO) of DEC, respectively. Generally, $\alpha > \beta$. When $m_i = m$ is equal for all source nodes, $\alpha = k(m_i - 1)$. Likewise, we can express $\beta = \epsilon k$.

In [5],[6], and [7], Dimakis has introduced the idea of DEC and shown that $m \geq 5\frac{n}{k}\log(k)$ is sufficient to guarantee that collecting *any* $(1+\epsilon)k$ encoded blocks is enough to recover the original k blocks w.h.p, for some $\epsilon > 0$. The decoding is assumed to be using Gaussian Elimination requiring $\mathbf{O}(k^3)$ or $\mathbf{O}(k^2\log(k))$ when exploiting sparsity of the coefficient matrix. The construction of "Robust Soliton Distribution"-like decentralized code has been investigated in [8], [9], [10], [11], [12], [13], [14], [15], and [16]. In the *node-centric* approaches [8][9][10][11][12], the source data performs a *random walk* over the storage nodes, where at each step, the source data is *xor*-ed with the local data at the current storage node. On the other hand, *packet-centric* approaches [13][14][15][16], allow source packets to perform a random walk while encoding from the data on each newly visited storage node, until eventually residing at the walk-terminating node.

Even though many previous authors try to approximate Robust Soliton Distribution [8], [9], [10], [11], [12], [13], [14], [15], and [16], we argue that Robust Soliton Distribution is only optimized for a subset of requirements. More specifically, Robust Soliton Distribution ensures low complexity encoding and decoding. However, applications in WSNs where source data is decentralized, achieving such distribution is limited due to the initial dissemination of data. Besides, if decoding is performed off-line such as in Delay Tolerant Networks (DTN), low complexity decoding can be exploited for the sake of longer network life.

We also argue that achieving survivability requires less EO than that required to achieve low delay decoding as in [5]. The catch would however be at the cost of higher DO. Fortunately, motivated by the results in Section II, we know that the DO is upper bounded by $\beta = k + c$ where $c \simeq 8$.

In this paper, we present a decentralized survivability scheme (DS-DES) for WSNs based on DEC. Unlike previous work, our objective is to increase the immunity of data to failure rather than reducing number of packets required for decoding (β) as in [5],[6], and [7]. Energy in WSNs is crucial and using random walk as in [8][9][10][11][12] requires excessive energy to implement. In addition, random walk protocols are asymptotic in nature, requiring large number of nodes to converge to the required distribution. Our approach has been shown to be applicable to networks as small as 10 nodes. In addition, all previous studies assume n, the number of redundant nodes, to be given. We provide a way to calculate the number of storage nodes needed to achieve the required survivability. We also assume all encoding is performed over $\mathbb{F}_2$. Even though higher fields are seen in the literature, field choice has a tremendous impact on encoding/decoding performance in practice[17]. To our knowledge, such survivability scheme does not exist in the literature.

In Section II, we review some important results on the properties of random matrices over finite fields. The results will be used later to develop the proposed survivability scheme in Section III. Experimentation and results are then discussed in Section IV and a brief conclusion and future work is presented in Section V.

II. Rank Properties of Random Matrices over Finite Fields

In this section, we review some important results on the properties of random matrices over finite fields. These results will be used in the design and analysis of the proposed codes. We will restrict the discussion to $\mathbb{F}_2$. Interested readers are directed to [18] for a detailed discussion. Let G be a $k \times m$ random matrix where each element g_{ij} of G is drawn from F_2 according to a probability distribution $\rho(x)$. More specifically,

$$\rho(x) = P(g_{ij} = x) = \begin{cases} 1-p, & \text{for } x = 0 \\ p & \text{otherwise} \end{cases} \quad (1)$$

We are interested in the probability of G having a certain rank r, in terms of the column count m. It would be useful to note that the number of possible vectors in the k-dimensional space $\mathbb{F}_q^k$ is q^k. In the case where $q = 2$ and $p = 1/2$ (uniform distribution), the probability of $G_{k\times m}$ having a rank $r = m$, can be expressed as

$$\begin{aligned} P(rank(G) = m) =& (1-2^{-k})(1-2^{-(k-1)})...(1-2^{-(k-m+1)}) \\ =& \prod_{i=k-m+1}^{k}(1-2^{-i}) \end{aligned}$$

Specifically, when $m = k$, the probability of G being full rank is

$$P(rank(G) = k) = \prod_{i=1}^{k}(1-2^{-i}) \quad (2)$$

Where the first term in Eq. 2 represents the probability of choosing the first vector, namely, choosing any vector except the zero vector ($\overrightarrow{0}$). The second term represents the probability of choosing any vector except any linearly dependent vector of the vector chosen in the previous step and $\overrightarrow{0}$. The third term corresponds to choosing any vector other than the already chosen vector or any of their linear combinations. The rest of the formula can be deduced similarly.

Interestingly, Eq.(2) converges to a constant when $k \to \infty$. To see this, consider the following theorem from [19].

Theorem 1: Let G be a binary random $k \times n$, $n \geq 0$ matrix with entries chosen equally likely. Then for $k-s \leq min(k,n)$, $k \leq s \leq 0$ and $k \to \infty$ we have

$$P(rank(G) = k-s) \to 2^{-s(m+s)} \prod_{i=s+1}^{\infty}(1-\frac{1}{2^i}) \prod_{i=1}^{m+s}(1-\frac{1}{2^i})^{-1},$$

where the last product equals 1 for $m+s=0$ (i.e. full rank matrix).

Now, let Q_s denote the probability that a $k \times m$ matrix has a rank $r = min(k,m)$. Then

$$\begin{aligned} Q_s &= \prod_{i=s+1}^{\infty}(1-2^{-i}) \\ \log(Q_s) &= \log \prod_{i=s+1}^{\infty}(1-2^{-i}) = \sum_{i=s+1}^{\infty}\log(1-2^{-i}) \\ &= \sum_{i=1}^{\infty}\frac{-2^{si}}{i(2^i-1)} \end{aligned}$$

Q_0 can be seen as the probability that the matrix will have a full rank given that k columns have been generated. It is interesting to see that

$$Q_0 = \prod_{j=1}^{\infty}(1-\frac{1}{2^i}) = 0.2887880951... \quad (3)$$

Now, let P_m be the probability that exactly m extra packets beyond k are needed to achieve full rank. P_m can be expressed as

$$P_m = Q_m - Q_{m-1} \quad (4)$$

Therefore, the average number of extra packets $\overline{m}$ required to achieve full rank equals

$$\overline{m} = \sum_{m=0}^{\infty} mP_m = \sum_{i=0}^{\infty}(1-Q_i) = 1.6067 \quad (5)$$

There are two important results to note from Eq.(3) and Eq.(5). First, the probability of full rank of any uniformly distributed square matrix converges to a constant. This is in fact true to binary as well as non-binary matrices. In the case of binary matrices, this constant is 0.288. Table I shows Q_0 for different values of q. Second and more importantly, the number of extra vectors (beyond k) required to have a full rank with high probability is on average very low and is independent of k. For example, on average only two extra vectors are required to make G have a full rank. As shown on Table II, when the number of extra vectors is 8, the probability is $Q_0 = 0.996$. The latter is quite accurate for k as low as 10.

q	2	4	8	16	32	64	128	256
Q_0	0.288	0.689	0.859	0.934	0.968	0.984	0.992	0.996

TABLE I
INVERTIBILITY PROBABILITY VS. FINITE FIELD (F_q).

s	0	1	2	3	4	5	6	7	8
Q_s	0.288	0.577	0.770	0.880	0.938	0.969	0.985	0.992	0.996

TABLE II
INVERTIBILITY PROBABILITY VS. NUMBER OF EXTRA VECTORS ($\beta - k$).

The next important two theorems state that the full rank probability seen above is not specific to the uniform distribution ($p = \frac{1}{2}$). In fact, as long as the probability p is within a certain interval, the results from the uniform case still apply. To see this, consider Figure 2. The plot was generated for a randomly generated square matrix with $k = 20$ and it shows the average invertibility probability versus p for $\mathbb{F}_2$. The curve illustrates that the invertibility probability of 0.288 applies for a range of values of p and not only for $p = 1/2$.

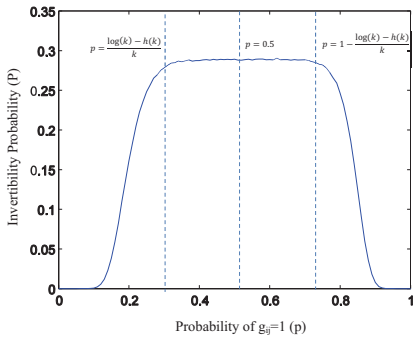


Fig. 2. Invertability Probability (P) vs. p for $\mathbb{F}_2$.

In [20] and [21], C. Cooper shows an expression of the probability of the rank of a random matrix over a finite field in terms of p. The expression applies to p over certain period according to the following theorem.

Theorem 2: ([20], Theorem 1)
Let $p = \frac{q-1}{q}$, and let G be a random $(k \times k)$-matrix with entries in $\mathbb{F}_q$. Let $p_n(s,q)$ be the probability that $rank(G) = k-s$, then

$$\lim_{k\to\infty} p_n(s,q) = \pi(s,q) \quad (6)$$

$$= \begin{cases} \prod_{j=1}^{\infty}(1-\frac{1}{q^j}), & s=0 \\ \frac{\prod_{j=s+1}^{\infty}(1-\frac{1}{q^j})}{\prod_{j=1}^{s}(1-\frac{1}{q^j})}(\frac{1}{q})^{s^2}, & s \geq 1 \end{cases} \quad (7)$$

Theorem 3: ([20], Theorem 2-i)
Let $G \in G(k,p,2)$ be a $k \times k$ random binary matrix over $\mathbb{F}_2$. Further, if $p(k) = \frac{\log k + h(k)}{k} \leq 1/2$: Then

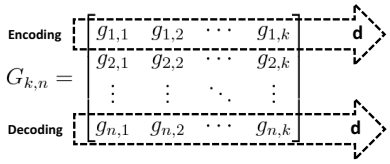
$$\lim_{k\to\infty} P(G \text{ is non-singular})$$

$$= \begin{cases} 0, & h(n) \to -\infty \\ c_2 e^{-2e^{-h}}, & \text{h constant} \\ c_2, & h(n) \to \infty \end{cases} \quad (8)$$

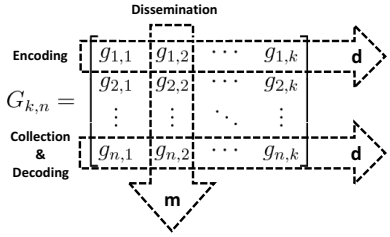
where $c_2 = \pi(0, 2)$.

III. Data Survivability via Decentralized Erasure Codes (DS-DES)

Before discussing the proposed scheme, it would be useful to outline some important observations. Figure 3 shows the difference in the construction of centralized versus decentralized erasure codes. As stated earlier, the degree of each **row** of the generator matrix (G) in the centralized case (Figure 3(a) can be produced exactly to match the value of d generated by the distribution $\rho(k)$. This is because all source data exists at the encoder. On the other hand, in the case of decentralized codes (Figure 3(b)), each source node chooses which storage nodes receive its data by disseminating m duplicate copies to m distinct storage nodes. Thus, setting exactly m entries per **column**. Therefore, instead of designing the code by controlling the degree distribution of the matrix, we try to control the over all distribution of the matrix.



(a) Centralized Codes



(b) Decentralized Codes

Fig. 3. Construction of Generating Matrix (G) of Erasure Codes.

One way to match the distribution of a centralized erasure codes is to send *all* source data to *all* storage nodes ($m = n$) and let each storage node choose a subset of the source data according to $\rho(k)$. Clearly, the communication requirements of such a solution is $\mathbf{O}(nk)$. The good news is that we can do better since we have control over the distribution of G. In what follows, we show how we can use the results from the previous section, to implement decentralized codes for data survivability applications.

We consider a network $N(k, s)$, similar to that shown in Figure 1, with a set X of k **source nodes**, $X = \{x_1, x_2, ..., x_k\}$ and a required survivability s. *Survivability* is defined as the maximum fraction of nodes to fail without compromising the recoverability of the native data. For example, $s = 0.8$ corresponds to a code that can tolerate $s \times 100 = 80\%$ failure ($0.8 \times k$ nodes). In other words, the code is decodable as there is at least $s_e = \frac{N}{K}$ fraction of nodes remaining. We are interested in designing a storage network with a code $C(n, s)$ and survivability s.

Let n be the number of **storage nodes**. n can be calculated as

$$n = k(s + 1) \tag{9}$$

Let $Y = \{y_1, y_2, ..., y_n\}$ be the set of storage nodes. In addition to storage, nodes in Y are also assumed to serve as relays. Without loss of generality, we assume that $X \cap Y = \oslash$. We also assume that a multi-hop routing mechanism is in place. Each source node x_i generates a data block b_i, selects a set of m storage nodes $Z = \{z_1, z_2, ..., z_m\}$ uniformly and randomly where $Z \subseteq Y$, and sends b_i to the set of selected nodes. We refer to m as the *Redundancy Factor* (RF). Let $B_S = \{b_1, b_2, ..., b_k\}$ represents the set of all native packets generated by the k source nodes. We refer to the total number of packets (α) sent out by all the source node as *Encoding Overhead (EO)*, where

$$\alpha = (k - 1)m \tag{10}$$

Upon receiving a set of data blocks B_R^j, each storage node Y_j combines the received blocks linearly to generate an encoded block e_j as

$$e_j = b_1 \oplus b_2 \oplus ... \ \forall \ b_i \in B_R^j$$

e_j represents a linear combination of a sum of a random subset of B_S. The number of packets $d_j = |B_R^j|$ used to construct an encoded packet e_j is called the *packet degree* of e_j . Along with e_j, a $|B_S|$-dimensional binary vector $G_j = \{g_{j1}, g_{j2}, ..., g_{jk}\}$ is generated with entries as

$$g_{ji} = \begin{cases} 0, & \text{if } b_i \notin B_R^j \\ 1, & \text{otherwise} \end{cases} \tag{11}$$

G_j is referred to as the *encoding vector*. Each node is assumed to have a storage space for only one encoded packet and its corresponding encoding vector. Let G be the *global generating matrix* as seen in Figure 3(b). When data is being collected, a subset of storage nodes are contacted to forward their encoded packets along with the corresponding encoded vectors. The data collector builds a *local generating matrix* $\overline{G} \in G$ using the received encoding vectors, an *encoded data matrix* E using the encoded blocks, and solves for the native data B in the system of equations $B = E\overline{G}^{-1}$.

Algebraically, to achieve a survivability s, we attempt to build G such that it is reversible with high probability even when $s \times k$ rows are deleted. According to Theorem 3, a random binary square matrix achieves it is highest probability of reversibility as long as

$$\frac{\log k + h(k)}{k} \geq p \geq 1 - \frac{\log k + h(k)}{k} \tag{12}$$

Further $h(k)$ can be a constant c resulting in an invertibility probability (P) as

$$P = c_2 e^{-2e^{-c}} \tag{13}$$

where $c_2 = \pi(0, 2)$ is given by Eq. 6. As shown in Figure 4, it is not difficult to see that $0 \leq P \leq c_2$. Further, $P = c_2 \forall\ x \geq 7$. The choice of a constant value for x is important since it affects the resulting overhead. Besides, the value of such a constant can be made absolutely small.

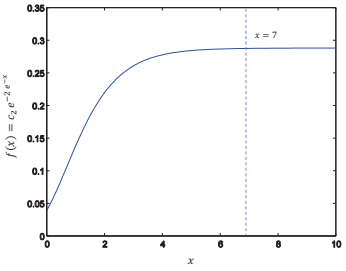


Fig. 4. $P = f(x) = c_2 e^{-2e^{-x}}$.

To maintain the same probability for the square $k \times n$ matrix, we need

$$p = \frac{m \times k}{n \times k}$$
$$m = np = \frac{n}{k}(\log k + c_1^\star) + c_2^\star$$
$$= (1 + s)(\log k + 7) + 8$$

The value of $c_1^\star$ and $c_2^\star$ corresponds to the constant in c in Eq. 13 and the number of extra vectors required for decoding as in Table II. We have chosen $c_1^\star = 7$ as discussed before. We have also chosen $c_2^\star = 8$ based on the experimental values shown in the table. When data collection takes place, the data collector retrieves e_j's and G_j's to build to matrices E and G, respectively, and decodes the native blocks as $B = EG^{-1}$. The only condition required for this is that G must be invertible (or full rank). As shown in Figure 3, the properties of the resulting code depend on the properties of the resulting coefficient matrix G. Note that unlike ordinary erasure codes, the packet degree distribution cannot be controlled in DEC. Therefore, the attention is shifted to the distribution (density) of matrix G. In addition, for ordinary erasure codes $EO = DO$, while for DEC $EO \neq DO$. To find the required redundancy $EO = \frac{\log k + 7}{k}$.

Clearly, there is a compromise between EO and the expected DO. Choosing EO to be small saves energy during encoding but results in higher DO than when chosen to be high. Remember that

$$(1 + s)(\log k + 6) + 10 \geq m \geq 5(1 + s)(\log k) \qquad (14)$$

IV. Experiment and Results

To verify the proposed survivability scheme and to evaluate its performance, a custom simulator was built. The simulator starts by creating a network of k source nodes and a storage network of n nodes. Based on the survivability required, the values for n and m are calculated as previously discussed. The simulator takes as input: k, s, and $\mathbb{F}_q$. Next, the dissemination phase begins where each source node randomly selects m distinct storage nodes and forwards the data to them. Unlike [5], we assume that the m random nodes chosen are all different. In the encoding phase, each storage node encodes the received packets over the finite field specified, and stores the encoded packets in addition to the corresponding encoding vector. To test the code survivability, we set f, the erasure rate, to values between 0 and 1. For each value of f, $f \times n$ storage nodes are deleted, and decoding is applied to test if naive data can still be decoded. To test decodability, storage nodes are selected randomly to build $\overline{G}$. If $rank(\overline{G}) = k$, decoding is successful and we record the number of packets used for decoding (β). Otherwise, decoding fails. To calculate the probability of successful decoding (P), large number of different test cases are generated and tested. Procedure 1 illustrates the general steps of the simulation.

Procedure 1 Simulation Pseudo-code

```
Input: k, s
  n = (s + 1)k;
  m = (s + 1)(log k + 7);
  for i = 1 to 10000 do
    Disseminate(n, m);
    for failure level (f) from 0 to 1 do
      CollectPackets(f);
    end for
  end for
```

The simulation used $k = 10 - 50$ and $s = 4$. The results are based on 10000 different runs with different initial choices of storage nodes by source nodes. We are interested in the probability of successful decoding (P) and the average number of packets required for decoding (β). We define P as

$$P = \frac{\text{Number of successful decodings}}{\text{Total number of trials}} \qquad (15)$$

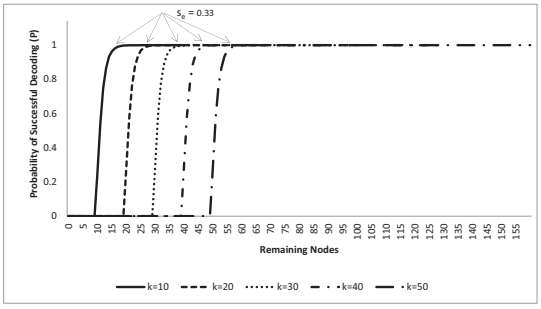


Fig. 5. Probability of Successful Decoding (P): $k = 10 - 50$, $s = 2$, $\mathbb{F}_2$.

Figure 5 shows the performance of the code in terms of the probability of successful decoding (P). The graph shows the value of P for different values of k and for $s = 2$. The The probability behaves as expected since $P = 1 \forall f > 0.2$. When $f < 0.2$, not enough data exists to recover all the native data. It is also interesting to note that the code works even for networks as small as $k = 10$. The decoding overhead required

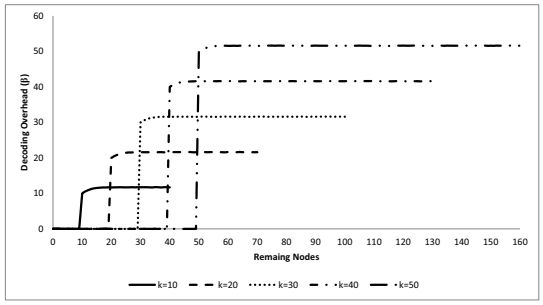


Fig. 6. Decoding Overhead (β): $k = 10 - 50$, $s = 2$, $\mathbb{F}_2$.

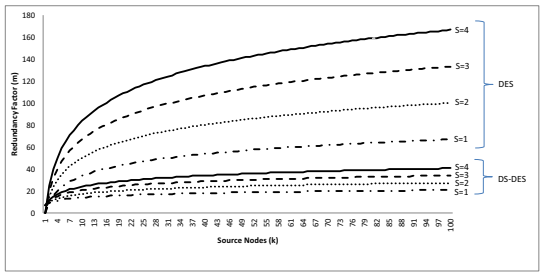


Fig. 7. Redundancy Factor (m) Vs. Source Nodes (k).

(β) represents the number of packets needed to successfully complete the decoding process. As shown in Figure 6, only one or two packets are required on average beyond k to successfully decode all packets. This is true for all values of k. This confirms with the theory discussed in Section II. We are also interested in the redundancy factor (m) to compare the proposed scheme with the scheme in [5]. As shown in Figure 7, DS-DES requires much less than the original DES to achieve data survivability. Furthermore, the low redundancy does not compromise the decodability of the code. The savings in redundancy translate to less energy requirements when applied to resource limited systems such as WSNs.

V. Conclusion

In this paper we have presented a survivability scheme for WSNs based on decentralized erasure codes. The scheme allows for maintaining a set of data in an encoded form in a hostile environment even when partial failure of nodes occurs. Further, the scheme allows designers to predict the number of redundant nodes in addition to the amount of redundancy in data to survive a given level of failure. A number of interesting directions can be pursed such as the impact of different routing techniques on the overall performance of the code. Moreover, we will study the benefits gained from applying network coding by exploiting coding opportunities at relay nodes. We leave these problems as future work.

Acknowledgment

This research is funded by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '98. New York, NY, USA: ACM, 1998, pp. 56–67.

[2] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[3] M. Luby, "LT Codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*, ser. FOCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 271–280.

[4] A. Shokrollahi, "Raptor Codes," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[5] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Distributed Data Storage in Sensor Networks using Decentralized Erasure Codes," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2, Nov. 2004, pp. 1387–1391 Vol.2.

[6] ——, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," in *IPSN '05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 15.

[7] ——, "Decentralized Erasure Codes for Distributed Networked Storage," *IEEE/ACM Trans. Netw.*, vol. 14, no. SI, pp. 2809–2816, 2006.

[8] ——, "Distributed Fountain Codes for Networked Storage," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 5, 2006, p. V.

[9] Y. Lin, B. Liang, and B. Li, "Data Persistence in Large-Scale Sensor Networks with Decentralized Fountain Codes," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1658–1666.

[10] S. A. Aly, Z. Kong, and E. Soljanin, "Fountain Codes Based Distributed Storage Algorithms for Large-Scale Wireless Sensor Networks," in *IPSN '08: Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 171–182.

[11] S. Aly, Z. Kong, and E. Soljanin, "Raptor Codes based Distributed Storage Algorithms for Wireless Sensor Networks," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, july 2008, pp. 2051 –2055.

[12] Z. Kong, S. A. Aly, and E. Soljanin, "Decentralized Coding Algorithms for Distributed Storage in Wireless Sensor Networks," *IEEE J.Sel. A. Commun.*, vol. 28, no. 2, pp. 261–267, 2010.

[13] C. Stefanovic, V. Stankovic, M. Stojakovic, and D. Vukobratovic, "Raptor Packets: A Packet-Centric Approach to Distributed Raptor Code Design," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, June 28-July 3 2009, pp. 2336 –2340.

[14] D. Vukobratovic, C. Stefanovic, V. Crnojevic, F. Chiti, and R. Fantacci, "A Packet-Centric Approach to Distributed Rateless Coding in Wireless Sensor Networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, june 2009, pp. 1 –8.

[15] D. Vukobratovic and, C. Stefanovic and, and V. Stankovic and, "Fireworks: A Random Linear Coding Scheme for Distributed Storage in Wireless Sensor Networks," in *Information Theory Workshop (ITW), 2010 IEEE*, 30 2010-sept. 3 2010, pp. 1–5.

[16] S. Kokalj-Filipovic, P. Spasojevic, and E. Soljanin, "Doped Fountain Coding for Minimum Delay Data Collection in Circular Networks," *IEEE J.Sel. A. Commun.*, vol. 27, no. 5, pp. 673–684, 2009.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the Air: Practical Wireless Network Coding," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 3, pp. 497–510, June 2008.

[18] C. Studholme and I. F. Blake, "Properties of Random Matrices and Applications," unpublished Report. [Online]. Available: http://www.cs.toronto.edu/ cvs/coding/

[19] V. F. Kolchin, *Random graphs*. New York, NY, USA: Cambridge University Press, 1999.

[20] C. Cooper, "On the Rank of Random Matrices," *Random Structures & Algorithms*, vol. 16, pp. 209–232, March 2000.

[21] ——, "On the Distribution of Rank of a Random Matrix over a Finite Field," *Random Structures & Algorithms*, vol. 17, no. 3-4, pp. 197–212, 2000.