Energy Efficient 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*—Designing reliability solutions for WSNs poses intricate challenges due to limitations in processing power and available energy. Such networks are often deployed in harsh and inaccessible environments and are therefore required to be highly reliable. However, reliability normally translates to redundancy in hardware and other resources implying both complexity and higher costs. In this study, we consider data survivability in WSNs. We present a data-centric framework based on Decentralized Erasure Codes (DEC) to increase the likelihood of data survivability in case of sensor nodes failure. The proposed framework enables network engineers to estimate the redundancy in hardware and data to achieve a given data survivability level. We also show two approaches to reduce the energy requirements of the proposed coding scheme using Random Linear Network Coding (RLNC). In addition to being decentralized, the proposed schemes are low in complexity requiring only binary coding over $\mathbb{F}_2$. We evaluate the performance of the proposed schemes by simulations and compare them to schemes with no network coding.**

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

I. Introduction

Research in Wireless Sensor Networks (WSNs) has seen a growing interest in a wide range of applications such as manufacturing, health, security, and environment monitoring. In addition to being autonomous, WSNs are often required to be robust and reliable, especially when deployed under harsh working conditions. What adds to their complexity is that constraints both in resources and capabilities require that any WSN solution to be simple and efficient.

WSNs often exist in hostile and inaccessible environments characterized by rough conditions making sensor nodes susceptible to failures or destructions. While sensor nodes are often assumed to be inexpensive, the data they collect is usually valuable. In fact, for many applications, it is desirable to maintain the data safe even after the node has failed. This motivates us to introduce a new data-centric approach to survivability that is different from the commonly presented network-centric approach. We make the distinction between the network-centric "network survivability" and the data-centric "data survivability". *Network survivability* utilizes redundancy in both software and hardware to guarantee that network continues to operate even when failures occur. However, *data survivability* uses redundancy in data to allow sensed data to cope with partial loses. In both approaches, mathematical relations between the redundancy required and the maximum failure that can be tolerated are key aspects.

Common approaches to implement data survivability rely on either *replication* or *coding* of data over multiple nodes. This way, even if some nodes disappear, the data can still be recovered from other nodes that are available in the network. Unlike coding, replication often requires lots of storage space on every node. In other words, to attain the same level of reliability, replication based schemes require more redundancy than coding based schemes. In fact, for the same level of redundancy, coding can achieve an order of magnitude higher reliability than simple replication [1]. In addition, replication based approaches also need to keep track of where different data is, resulting in complicated data gathering protocols. Further, it has been shown analytically that on average the number of data blocks needed to retrieve a complete data set from a replication based distributed storage is more than that needed when using coding based distributed storage [2]. Coding has also been shown to greatly reduce storage requirements as well as simplify data gathering mechanisms.

In [3], we described the design and implementation of a data survivability coding scheme (DS-DEC) using Decentralized Erasure Codes (DEC)[4]. The proposed codes where studied under the assumption of a delay tolerant WSNs setup deployed in a remote location where sensed data need to be protected against network failures in the absence of a sink node.

Data survivability presents a novel approach to reliability compared to network-centric approaches. By disseminating and encoding data, we break the dependence between the location of the source and the data being sought, meaning any data packets can provide the information rather than a specific data from a certain node. In addition, the proposed scheme is independent in design of topology, which makes it applicable to virtually any network. Moreover, the overlay nature of the scheme, allows for easy integration into existing architectures. The scheme can easily be implemented as an application layer add-on.

Network Coding (NC) [5] evolved from information theory

978-1-4673-1564-7/12/$31.00 ©2012 IEEE

and has received a growing interest in many fields ever since. Random Linear Network Coding (RLNC) [6] was later proposed as a tool for applying NC in dynamic networks in a randomized fashion. In the context of our problem, we are proposing the use of RLNC to improve the performance of DS-DEC based distributed storage for the purpose of increasing data survivability while reducing the energy requirements needed for implementing the codes.

By utilizing coding opportunities arising at relay nodes during data dissemination phase, we lower the cost of the energy required for both dissemination and encoding. Multihop data dissemination is an inherent part of DS-DEC and RLNC fits well with the decentralized operation of the scheme. By taking advantage of RLNC, we design two energy efficient DS-DEC based coding schemes to increase data survivability while reducing energy requirements. The two schemes described in this paper are: Encode-and-Forward (DEC-EaF) and Encode-and-Disseminate (DEC-EaD).

The proposed framework enables network engineers to estimate the redundancy in hardware and data to achieve a given data survivability level. In addition to being decentralized, the proposed schemes are low in complexity requiring only $\mathbb{F}_2$ binary coding. We evaluate the performance of the proposed schemes through simulations and compare them to the schemes that do not employ network coding.

The remaining of the paper is organized as follows. Section II highlights relevant work in the area of coding based distributed storage. In Section III, we present an overview of the DS-DEC scheme. The proposed schemes are described in Section IV. Performance and simulation results are discussed in Section V. Finally, Section VI includes some conclusive remarks and future directions.

II. Related Work

In the context of our problem, we assume a sensor network composed of k source nodes and a storage network with n storage nodes. Each source node generates a single data block and sends it to m storage nodes. The storage nodes then combine the received blocks linearly and store the encoded block along with the corresponding coding vector. To retrieve the native blocks, a data collector contacts *enough* number of storage nodes to retrieve *enough* number of encoded blocks. Decoding of the blocks can then be carried using Gaussian Elimination.

In [7], Dimakis et al. introduce the idea of DEC and show 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], and [11]. In the *node-centric* approaches [8]-[10], 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 such as [11], 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 researchers tried to approximate Robust Soliton Distribution, 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 shown previously that DEC is enough to achieve data survivability [3].

III. Data Survivability Decentralized Erasure Codes

The main goal of data survivability is to ensure the recoverability of the native data stored in the network when a given maximum failure (erasure) occurs. The scheme can be described as in Figure 1. Given a **sensor network** $X = \{x_1, x_2, ..., x_k\}$ of k *source nodes* and a *survivability* s, a **storage network** $Y = \{y_1, y_2, ..., y_n\}$ of $n = k(s+1)$ *storage nodes* is deployed. Then, each source node x_i generates a data block b_i and forwards it to $m = (s+1)(\log(k)+7)+8$ distinct storage nodes $Z = \{z_1, z_2, ..., z_m\}$ which are chosen uniformly at random. Note that Figure 1 does not necessarily reflect the physical topology of the network. In fact, in our experiments we assume a single network with both types of sensor nodes uniformly distributed. We refer to m as the Redundancy Factor (RF). Moreover, the Encoding Overhead (EO) is defined as $\alpha = (k-1)m$. Let $B_S = \{b_1, b_2, ..., b_k\}$ represents the set of all *native blocks* generated by the k source nodes. We assume the existence of a multihop routing protocol responsible for delivery of data from source to destination.

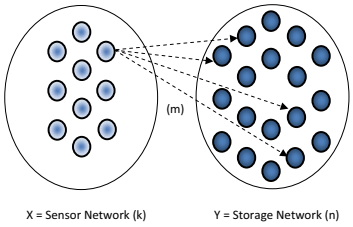


Fig. 1. Source and Storage Networks ($k = 10$, $n = 18$, $m = 4$, and $s = 0.8$).

Upon receiving a set of data blocks B_R^j, each storage node Y_j xor's the received blocks bitwise to generate an *encoded block* e_j as

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

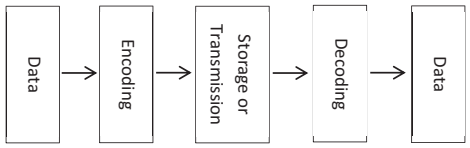
where e_j represents a linear combination of a sum of a random subset of B_S. The number of blocks $d_j = |B_R^j|$ used to construct an encoded block e_j is called the *block degree*

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

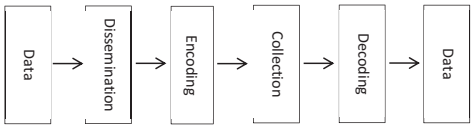
$$g_{ji} = \begin{cases} 0, & \text{if } b_i \notin B_R^j \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

G_j is referred to as the *encoding vector*. For simplicity, each storage node is assumed to have a capacity for only one encoded block and its corresponding encoding vector. Let G be the *Global Encoding Matrix* (GEM) as seen in Figure 3. When data is being collected, a subset of storage nodes are contacted to forward their encoded blocks along with the corresponding encoding vectors. The data collector builds a *Local Encoding Matrix* (LEM) $\overline{G} \in G$ using the received encoding vectors, an *encoded data matrix* E using the encoded blocks, and solves for the native data blocks B in the system of linear equations $B = E\overline{G}^{-1}$, where $\overline{G}^{-1}$ is the inverse of $\overline{G}$.

The problem of erasure code design in centralized setup is different than the decentralized case. In the centralized case (Figure 2(a)), all source data blocks B_S are available for a single encoder. Hence, the resulting degree distribution of the code matches exactly that of the required distribution $\rho(k)$. This can be illustrated by the construction of the GEM shown in Figure 3(a). In the decentralized case however, each distributed encoder have a subset of the data set $\in B_S$. This is mainly due to the dissemination phase preceding the encoding phase as shown Figure 2(b). This results in the GEM shown in Figure 3(b).



(a) Centralized Codes



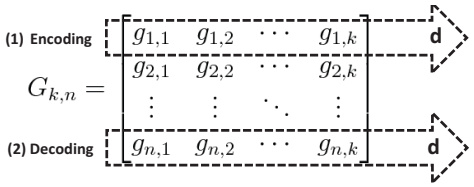
(b) Decentralized Codes

Fig. 2. Centralized vs. Decentralized Erasure Codes.

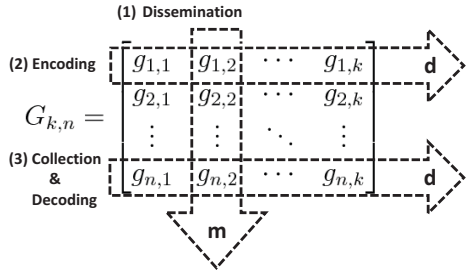
One strategy to achieve the required degree distribution in a decentralized setup similar to that of the centralized case is to send each source data block to all storage nodes. Then, we require the storage nodes to select a subset of data blocks according to a common degree distribution $\rho(k)$. This clearly would result in worst case communication cost of $\mathbf{O}(kn)$. Moreover, for survivability application, we have shown in [3] that it can be achieved at a much lower cost.

IV. Data Survivability via Decentralized Erasure Codes and Random Linear Network Coding

By restricting their role to pure routing/forwarding, relay nodes are not fully utilized by existing DEC schemes. So,



(a) Centralized GEM



(b) Decentralized GEM

Fig. 3. GEM for Centralized vs. Decentralized Erasure Codes.

after generating a source packet, choosing a set of candidate storage nodes, and forwarding the packet by source nodes, relay nodes help forward native packets to storage nodes without manipulating them. However, based on the argument that the cost of communication is generally much higher than processing in wireless nodes, we utilize the coding opportunities that arise during relaying packets using RLNC. The proposed modifications improve the efficiency of the dissemination phase by allowing relay nodes to participate in the encoding process during the dissemination phase. This can be done using one of two strategies: Encode-and-Forward (EaF) or Encode-and-Disseminate (EaD).

The two schemes that are presented here share the same fundamental model as the DS-DEC scheme we previously presented in [3]. DS-DEC provides quantitative measures to improve data survivability. Given a network of k source nodes and survivability s, DS-DEC works as follows:

1) Generate n storage nodes where
$$n = k(s+1). \quad (2)$$
2) Each source node i generates a source data block b_i.
3) Each source node i chooses
$$m = (s+1)(\log k + c_1^\star) + c_2^\star \quad (3)$$
distinct random storage nodes uniformly at random and forwards a copy of b_i to each of the selected nodes.

We also have shown that $c_1^\star \geq 7$ and $c_2^\star \geq 8$ should be sufficient to achieve the required survivability. Mathematically, since RLNC allows for a broader dissemination of data at a less energy cost, we can achieve the required density of the GEM using less energy. Remember that the GEM is required to have

$$\frac{\log k + h(k)}{k} \geq p \geq 1 - \frac{\log k + h(k)}{k} \quad (4)$$

to be invertible with high probability. Where p is defined as $p = P(g_{ji} = 1)$ and $h(k)$ is chosen to be a constant.

A. Encode-and-Forward (DEC-EaF)

In each step of the (DEC-EaF) coding algorithm, a target storage node is randomly chosen by the source node, and the native packet is forwarded accordingly in a multihop fashion. Then, for every relay node by which the packet passes, the relay node combines the packet it receives with the encoded packet it stores locally before forwarding it to the next hop. If no packet exists locally, the relay node saves a copy of the relayed packet. The source node is assumed to have multiple routes for every destination node. While the choice of the destination node is random, selecting the best route is not.

Let $R_i = \{r_1, r_2, ..., r_h\}$ be the set of possible routes to destination i. Further, let $r_j = \{s_1, s_2, ..., s_w\}$ be the set of nodes in route r_j. Let V be the set of nodes that have been either chosen as destination nodes or those which were in routes to previously selected destination nodes. In other words, V is the set of visited nodes. On the other hand, U is a set of nodes that have not been visited yet. Every time a destination s_{i1} and a route r_j pair is selected, V is updated as follows

$$V = V \cup s_{i1} \cup r_j \tag{5}$$

Accordingly, the redundancy factor RF which is first initialized to m, is updated as

$$RF = RF - |V| \tag{6}$$

In addition to subtracting the number of visited nodes from RF, the corresponding nodes that has been visited are also removed form candidate destination nodes as follows

$$U = U - (U \cap V) \tag{7}$$

We also define the depletion (σ) of a route r_1 as

$$\sigma = |r_1 \cap V| \tag{8}$$

Furthermore, we define the gain (ω) of a route r_1 as

$$\omega(r_1) = |r_1| - \sigma \tag{9}$$

Next, suppose s_{i2} is selected as a destination randomly. Amongst the possible routes (R_2) to s_{i2} we select r_i such that

$$r_i | \omega(r_i) > \omega(r) \forall r \in R_2 \tag{10}$$

Equivalently said, DEC-EaF chooses the the shortest path routes with the least number of visited nodes.

Algorithm 1 and 2 show pseudo codes describing the mechanism at the source node and the relay nodes, respectively.

It should be noted that the performance of the DEC-EaF depends on the topology of the network and the routing protocol in use. Let r_{ij} be the hop count between source node x_i and storage node y_j. Let $R = \{r_{ij}\} \forall i \in X, j \in Y$ be a $k \times n$ matrix containing the hop count between every pair of source and storage nodes. The average hop count $\overline{r}$ can be calculated as

$$\overline{r} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{n} r_{ij}}{k \times n} \tag{11}$$

The expected reduction in the required redundancy factor m should be roughly $\overline{r}$. However, due to the fact that the routes between nodes are not all disjoint, some relay nodes will be visited by the same packets more than once.

An example of how DEC-EaF works is shown in Figure 4. The nodes A, B, and C are assumed to be the source nodes while the numeric nodes are the storage nodes. First, node A chooses node 10 as a destination storage node. Clearly, the shortest paths between A and 10 are 3 hops long, which are $\{6, 7, 8, 10\}$, $\{6, 7, C, 10\}$ or $\{6, 9, C, 10\}$. We assume $\{6, 7, 8, 10\}$ is chosen randomly and the corresponding packet is forwarded along the chosen route. Since 5 copies of the source data have been disseminated during the last step (including the source node A), the new $RF = 7 - 5 = 2$. In the next step, assume node 5 has been selected as a destination storage node. Again, three candidate shortest path routes exist between A and 5; which are $\{6, 7, 8, 5\}$, $\{6, 7, B, 5\}$ and $\{6, 4, B, 5\}$. The number of unvisited nodes $(|U|)$ on each of the routes is 1, 2, and 3, respectively. Therefore, $\{6, 4, B, 5\}$ is chosen. If multiple routes have the same number of unvisited nodes, one is selected randomly. Since the value of RF becomes < 1, the dissemination process at Node A stops. The same logic can be applied to node B, using node 10 through $\{5, 8, 10\}$ and node A through $\{4, 6, A\}$. Finally, node C chooses node 11 through $\{13, 12, 11\}$ and node 3 through $\{7, B, 3\}$. The resulting code is shown in Figure 4-d).

Algorithm 1 Encode-and-Forward [DEC-EaF] (Source Node)

```
Generate a packet b_i;
RF ← m;
V ← φ;
for j = 1 → m AND RF > 0 do
    Selects target storage node y_j uniformly at random;
    Choose r according to Eq. 10;
    Route b_i to y_j through r;
    V = V ∪ r;
    RF = RF − |V|;
end for
```

Algorithm 2 Encode-and-Forward [DEC-EaF] (Relay Node)

```
Receive Packet x_j at relay node;
if g_i = 0 then
    If packet not already received
    Generate a new coefficient g_ji;
    c = c ⊕ (g_ji × b_i);
    g_i = g_ji;
end if
Forward b_i to next node in route;
```

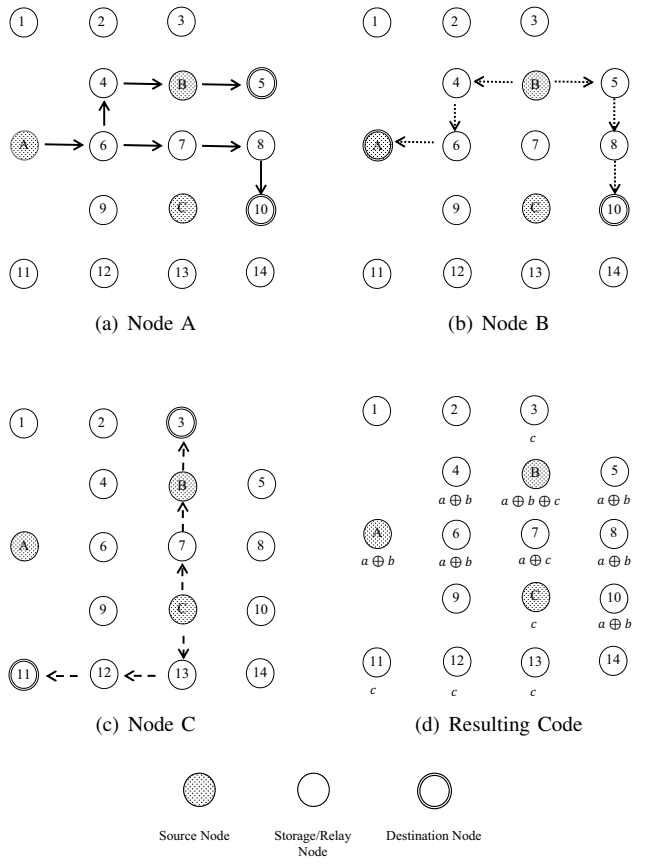


Fig. 4. Example of DEC-EaF: $m = 7$.

B. Encode-and-Disseminate (DEC-EaD)

In the second strategy, called "Encode-and-Disseminate (EaD)", source nodes disseminate the source packets using a node-centric random walk mechanism. Since the selection of target nodes is random in the original DS-DEC, it makes sense to use a random walk to eliminate the need for routing table construction and maintenance.

In random walk protocols, there is generally no guarantee that a certain packet will not visit the same node more than once. Therefore, adding s little intelligence at the relays can help. The relay forwarding strategy is based on the rotor-router model which is a quasirandom analog to the random walk process. The rotor-router model has been introduced in [12] and popularized by Jim Propp in 2001, and has found many applications. The main advantage of the model in our application is that it reduces chances of forwarding the packets to the same node while other neighbors have not been contacted. In addition, the model is simple and it eliminates the need for maintaining a forwarding table for each source packet.

In our algorithm, each node maintains a list of all single-hop neighbors. In addition, each relay node maintains an index for each packet. The index points to the next neighbor to whom each packet will be forwarded when the packet passes by the same relay. Whenever a packet is forwarded, the corresponding index is advanced to the following neighbor. This is required to reduces chances of some packets revisiting the same nodes. This mechanism also eliminates the need for a forwarding table. Note that the order in which neighboring nodes are selected is immaterial.

Likewise, when a relay node receives a packet, it encodes it locally. Then, it forwards the packet to one neighbor according to the corresponding index. To track the distribution of the coding process, each packet contains a counter (RF) of the number of copies of the packet that are left to be disseminated. The counter is decreased by one at each newly visited hop. When the counter reaches zero, the random walk terminates. The process is formally described in Algorithm 3 and 4.

Note that DEC-EaD does not require any routing. All that is required is knowing the set of neighbors which is not very difficult to discover. This is clearly an added advantage over DEC-EaF in addition to the reduction in the overall energy as will be shown in the next section.

Algorithm 3 Encode-and-Disseminate [DEC-EaD] (Source Node)

```
Generate a packet b_i;
RF ← m;
Route (b_i, RF) to w_0;
```

Algorithm 4 Encode-and-Disseminate [DEC-EaD] (Relay Node)

```
Receive Packet b_i;
if g_i = 0 then
    If packet not already received
    Generate a new coefficient g_i;
    c = c ⊕ (g_i × x);
end if
RF = RF − 1 ;
if RF > 0 then
    Route b_i to node w_(index);
    Advance index;
end if
```

An example of the operation of DEC-EaD is shown in Figure 5 for $m = 7$. Node A starts the following random walk $\{6, 7, C, 9, 6, 4, 2\}$. At each hop, the relay node chooses one neighbor (different from the last hop) uniformly at random. In addition, each relay node decrements the value of the RF field in the packet by 1. When the random walk reaches node 6 for the second time, the packet is forwarded to node 4, since all other neighbors have been visited. Moreover, RF remains unchanged. Hence, the random walk terminates at node 2. Similarly, node B and C perform their random walks as $\{7, C, 13, 12, 9, 6\}$ and $\{10, 8, 5, B, 4, 2\}$. The resulting code is shown in Figure 5-d).

V. Experiments and Results

To evaluate the performance of the proposed schemes, we built a custom simulator. The simulation takes as input the

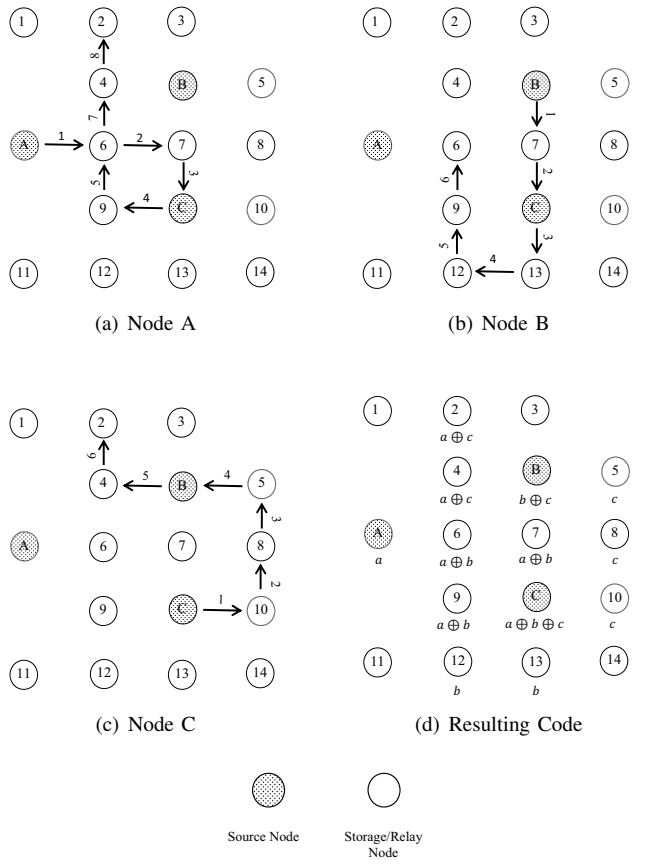


Fig. 5. An Example of DEC-EaD: $m = 7$.

number of source nodes k, the required survivability s and calculates both the number of storage nodes n and the value of the redundancy factor m required according to Eq. 2 and Eq. 3. The simulator is also supplied with the choice of encoding and routing strategy. Then, a random network topology is generated using the values of n and k. The location distribution of both source and storage nodes is uniform. The number of neighbors of each node is between 1 and 6. However, we ensure the network is connected with no isolated nodes. We assume all nodes can serve as storage and relay. However, only source nodes can generate source data.

In case of DS-DEC and DEC-EaF, a shortest path routing table is generated beforehand. In addition, both algorithms use a link-state routing strategy. Before, the dissemination phase starts, the routing table is built with multiple routes for each destination in case of DEC-EaF. Unlike DS-DEC and DEC-EaF, DEC-EaD does not require any routing table; and neighbor discovery takes place instead to operate the random walk. Although network may become disconnected during data collection phase due the failure of random nodes, we assume the data collector is powerful enough to reach all nodes. This assumption is made due to the focus on "data survivability" rather than "network survivability."

Errors due to channel or interference are not considered and sensor nodes are assumed to be stationary. However, failure of sensor nodes are simulated to examine the performance of the code. The main performance measures we are concerned with are: the survivability of the code, the energy required to construct the code, the number of packets required to decode (Decoding Overhead (DO)). All encoding and decoding are performed under $\mathbb{F}_2$. In addition, we evaluate the code degree distribution resulting form applying each scheme.

In order to quantify the energy requirements, we define the following energy model. We assume a network composed of wireless nodes powered by batteries. The model accounts for the energy consumed due to encoding, transmission, relaying, and reception. The focus of this work is on the number of operations executed during the dissemination and encoding process. Hence, we use an abstract energy model as described here.

The energy consumed by nodes can be due to either sensing (in the case of source nodes), communications (transmiting/receiving), or encoding. The corresponding energies consumed are therefore represented by ξ_s, ξ_t,ξ_r, and ξ_e, respectively, where $\xi_t, \xi_r >> \xi_e$. The energy of the data collector is assumed to be unbounded.

Accordingly, the cost of forwarding a packet without RLNC becomes $\xi_F = \xi_r + \xi_t$ for one reception and one transmission. On the other hand, the cost of relaying with RLNC becomes $\xi_{RLNC} = \xi_r + \xi_e + \xi_t$. The extra cost counts for generating random coefficient, multiplying the received packet by the random coefficient, and xor-ing the result with the local encoded packet. We also do not consider the energy consumed by sensing since it is assumed the same for all nodes.

For each test case, the simulator starts by disseminating and encoding the source packets based on the specified algorithm. For example, DEC-EaD starts a random walk at each source node which terminate when $RF = 0$. The energy consumed by the sensor nodes for data dissemination, relaying and encoding is recorded. After the code is generated, the data collection phase starts. The code decodability is evaluated under different values of the erasure factor $(f) \in [0, 1]$. For each value of the erasure factor, a number of storage nodes are deleted to simulate a failure and code decodability is attempted using the remaining nodes. For example, to simulate an erasure factor of $f = 0.1$ or 10%, $0.1 \times n$ randomly chosen nodes are deleted. The data collection is performed over the remaining set of nodes and results are recorded. Two aspects of decoding are of interest which are whether the code is decodable and the number of packets required for successful decoding. When performed over a large number of trials, those factors translate to probability of successful decoding and the average number of packets required to complete the decoding (DO). For each trial, a different set of storage nodes are chosen to simulate failure. We have tested the proposed algorithms using k=10,20,30,40, and 50. For each value of k, 1000 different test cases were generated.

Figure 6 demonstrates the survivability performance of DS-DEC and the proposed schemes. The performance shown corresponds to the DS-DEC. However, both of the proposed schemes, DEC-EaF and DEC-EaD, show the same perfor-

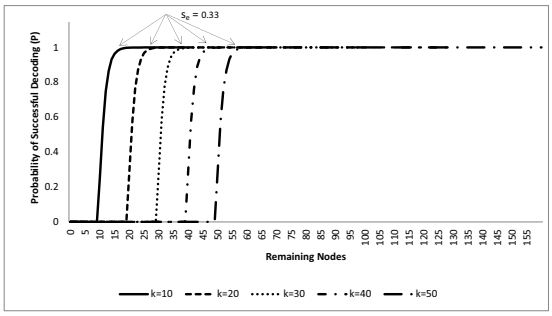


Fig. 6. Probability of Successful Decoding (P): $k = 10 - 50$, $s = 2$, $\mathbb{F}_2$.

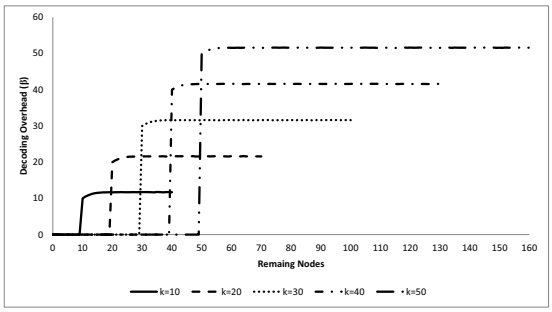


Fig. 7. Decoding Overhead (β): $k = 10 - 50$, $s = 2$, $\mathbb{F}_2$.

mance when tested under the same conditions, namely, survivability, network size, and network topology. However, similar graphs are omitted since they convey no new information. A similar statement can be said about DO, which is shown in Figure 7.

Since this study is more concerned with energy consumption, we discuss the performance in regard to energy required for achieving survivability. Table I shows the energy required to implement each scheme for different value of source nodes k. For each scheme, the number of required encode (COD), send (SND), and receive (REC) operation is tabulated. The total power is shown assuming CC1000 chip. Finally, the saving in energy compared to the basic DS-DEC scheme is also shown. We can see that DEC-EaF can implement the required code using less energy followed by DEC-EaD. The results however do not show the energy required to build and maintain routing functions. The number of send and receive operations are equal since each single "send" operation accompanies a corresponding "receive" operation by the receiving node. It is generally the case in wireless sensor networks that send and receive consume more energy than local processing. When taking that into consideration, we can see that the local encoding performed by RLNC although contributes to the total energy consumed by the scheme, it saves significant amount of energy over all. It can also be seen that whereas DS-DEC and DEC-EaD achieve the exact number of encodings, DEC-EaF does not. The reason is when the algorithm reaches the required RF, the number of unvisited nodes on the selected route may be more than what is required to reach $RF = 0$. This in turn results in extra encodings. This can be improved, especially when the number of nodes is large, the accumulative effect could be significant.

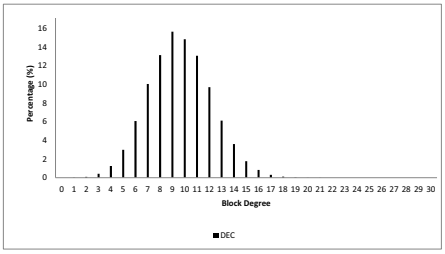
For example, on a CC1000 chip running at $868MHz$, processing requires $5mW$ while "send" and "receive" consumes $25.8mW$ $28.8mW$ [13]. Therefore, coding for a $k = 20$ network requires on the same chip needs $3W$ for the DS-DEC and DEC-EaD, while $3.11W$ for the DEC-EaF. On the other hand, communications require $126.7W$, $24.7W$, and $50.5W$ for DS-DEC, DEC-EaF, and DEC-EaD, respectively. Even though the performance of the DEC-EaF is superior to that of DEC-EaD, it comes at the cost of energy needed for building and maintaining routing tables. Given the difference between the two, DEC-EaF may be more feasible for more WSN where routing is not required.

TABLE I
NUMBER OF CODING, SEND, RECEIVE OPERATIONS REQUIRED TO IMPLEMENT DATA SURVIVABILITY SCHEMES. TOTAL POWER IS COMPUTED FOR CC1000.

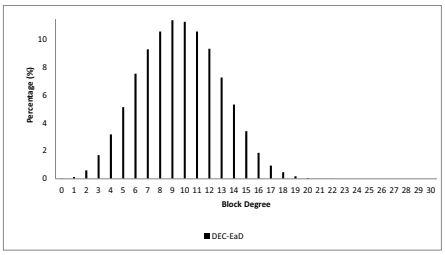
K	Operation	DS-DEC	DEC-EaD	DEC-EaF
10	COD	280	280	286.357
	SND	847.375	496.978	421.127
	REC	847.375	496.978	421.127
	Total power (W)	47.67	28.54	24.43
	Saving compared to DS-DEC		40.13%	48.75%
20	COD	600	600	622.24
	SND	2321.6	925.928	857.448
	REC	2321.6	925.928	857.448
	Total power (W)	129.76	53.56	49.93
	Saving compared to DS-DEC		58.72%	61.52%
30	COD	960	960	1003.38
	SND	4393.854	1441.42	1356.294
	REC	4393.854	1441.42	1356.294
	Total power (W)	244.70	83.50	79.07
	Saving compared to DS-DEC		65.88%	67.69%
40	COD	1320	1320	1393.7
	SND	6869.34	1926.99	1811.99
	REC	6869.34	1926.99	1811.99
	Total power (W)	381.67	111.81	105.90
	Saving compared to DS-DEC		70.71%	72.25%
50	COD	1650	1650	1759.714
	SND	9563.376	2354.948	2216.028
	REC	9563.376	2354.948	2216.028
	Total power (W)	530.41	136.83	129.79
	Saving compared to DS-DEC		74.20%	75.53%

We further evaluate the degree distribution of the constructed codes. For fast decoding, it is favorable to have the degree of resulting packets to follow Soliton Distribution or similar distribution. In this work however, we assume the decoding is performed offline and that data is not time sensitive. In other words, data can be collected from the site, and then taken to a stationary decoding station where energy is not a concern. In such a case, decoding can complexity can be $\mathbf{O}(k^3)$ is Gaussian Elimination is used. Despite the previous assumption, it is useful to consider the degree distribution produced by each coding scheme. DEC-EaF shows a more uniform degree distribution (Figure 8(c)) due to the fact that

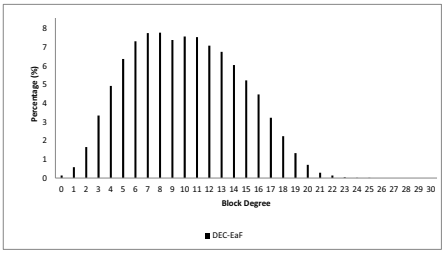
it chooses destinations evenly. DEC-EaD and DS-DEC on the other hand result in a more normal distribution ((Figure 8(a)) and (Figure 8(b))). This is expected since the behavior of the degree distribution follows the central limit theorem. It also can be seen that DS-DEC results in a higher average degree distribution. This can be explained by the complete independence between successively selected nodes. Unlike DS-DEC, DEC-EaF and DEC-EaD node selection depends on location, i.e., each data is disseminated over nodes that are close to each other. Remains to be said that due to DEC-EaD using random walk mechanism, it results in a more localized distribution of data with respect to location in the network. This can be seen as a shortcoming since it increases the likelihood of losing data in case of localized burst failures (e.g. flood or fire).



(a) DS-DEC



(b) DEC-EaD



(c) DEC-EaF

Fig. 8. Code Degree Distribution for DS-DEC, DEC-EaD, and DEC-EaF for $k = 30$.

VI. Conclusion

In this paper we have presented a data survivability scheme based on Decentralized Erasure Codes for WSN applications called DS-DEC. The scheme allows network engineer to calculate the redundancy required in data as well as in storage nodes to protect sensor data against a maximum expected erasure. We also presented two energy efficient coding schemes (DEC-EaF and DEC-EaD) that utilize Random Linear Network Coding (RLNC) to reduce the energy requirements of DS-DEC. While DEC-EaF depends on routing, DEC-EaD use random walk based on rotor router model. The proposed coding schemes have been evaluated by means of simulations and compared in terms of energy consumption, resulting code degree, and survivability.

The work presented in this paper can be extended to include more specific results for certain known topologies. The routing strategy can also be optimized against different objectives. Another dimension is the generation or approximation of a Soliton-like distribution. We leave such problems as future work.

Acknowledgment

This work was made possible by a National Priorities Research Program (NPRP) grant from the Qatar National Research Fund (Member of Qatar Foundation).

References

[1] H. Weatherspoon and J. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 328–338.

[2] S. Acedan'ski, S. Deb, M. Mdard, and R. Koetter, "How Good is Random Linear Coding Based Distributed Networked Storage," in *In NetCod*, 2005.

[3] L. Al-Awami and H. Hassanien, "Data Survivability in WSN via Decentralized Erasure Codes," *The 7th International Wireless Communications and Mobile Computing Conference*, Accepted for publication.

[4] 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.

[5] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[6] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," *the Proceedings of the 2003 IEEE International Symposium on Information Theory (ISIT'03)*, p. 442, June 2003.

[7] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "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.

[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] 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.

[11] 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.

[12] V. B. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy, "Eulerian walkers as a model of self-organized criticality," *Phys. Rev. Lett.*, vol. 77, pp. 5079–5082, Dec 1996.

[13] Q. Wang, M. Hempstead, and W. Yang, "A Realistic Power Consumption Model for Wireless Sensor Network Devices," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, vol. 1, sept. 2006, pp. 286 –295.