Data Persistence in Sensor Networks: Towards Optimal Encoding for Data Recovery in Partial Network Failures

Abhinav Kamra, Jon Feldman, Vishal Misra and Dan Rubenstein

Email: {kamra, jonfeld, misra, danr}@cs.columbia.edu

1 Introduction and Related Work

Sensors networks are capable of collecting an enormous amount of data over space and time. Often, the ultimate objective is to "sample, store and forward", that is to sense the data, store it locally and ultimately forward it to a central host (or "master node") where data from other sensor nodes is also collected and analyzed. A useful example is a traffic sensing network, there being traffic sensors at each intersection that estimate the traffic and relay it to a central processing station.

Typical sensor nodes are wireless nodes with limited storage and computational power. Furthermore, they are prone to "failure", by going out of wireless range, interference, running out of battery etc. When a sensor node fails, the data it was storing is lost. In a cooperative sensor network, it is a good idea to have nodes' data duplicated and spread around the network so it can be recovered from other nodes in case of failure. In particular, every node can store some of its own data as well as data from other nodes up to its storage capacity.

It has been shown [1, 2] that in a distributed network where the sensor nodes don't exactly know the data other nodes currently possess, it is beneficial to store encoded symbols of data units instead of the original data. The work of Yeung et. al. [3] also gives the idea that with limited storage, the coding may be required to maximize the information content. They further show [4] that Linear Coding suffices. That is, it is sufficient to store linear combinations of the data as the encoded symbols.

The benefits of storing combinations of data instead of original data has been studied in various works [5, 6]. Traditional error-correcting erasure codes can also be used to achieve the goal of encoding data such that if some of the encoding symbols are lost, data can still be recovered. Reed-Solomon codes [7] are block erasure codes that have been traditionally used for error correction. Tornado codes [8] and more recently LT codes [9] are erasure codes that require slightly more encoded symbols to recover the data but have much faster encoding/decoding. These codes focus on the problem of choosing an encoding such that all the data can be recovered using a minimum number of encoded symbols.

If part of the sensor network fails, the data stored in the failed sensor nodes is lost. The only data which remain is the data remaining at the surviving nodes. The surviving nodes have symbols encoded from the data produced by all the sensor nodes, so there is still a chance of recovering data produced by failed nodes using the surviving encoding symbols. We focus on the problem of choosing an encoding to maximize the data that can be recovered from the surviving encoded symbols as opposed to finding the minimum encoding symbols required to recover "all" data as studied in Tornado and LT codes.

2 Problem Formulation

Our network model consists of a sensor network with N nodes, each having a limited storage capacity. Each node has some data x_i that it has generated (or sensed). We call the x_i's data units. Since every node wants to spread its data throughout the network and since storing encoded symbols is better than storing unencoded data, there is a data mixing protocol according to which the sensor nodes form encoded symbols, exchange them with other nodes and store them.

As in the case of LT codes, we assume that encoded symbols are XOR-encodings. That is each symbol is formed by taking the bitwise XOR of a subset of data units $\bar{x}$. The number of data units XOR-ed to form a symbol is called the degree of the symbol. Typically, encoded symbols are produced according to a probability distribution $\bar{\pi}$ such that an encoded symbol is degree i with probability π_i. The i data units that form the symbol are chosen uniformly randomly.

In practice, the content and spread of encoded symbols will of course depend on the particular data mixing protocol between the nodes and the topology of the network so that the actual encoded symbols of a particular degree may not be uniformly chosen from the set of all symbols of that degree. But for the purpose of analysis, we assume that the encoded symbols have been mixed together and that they are generated according to some degree distribution $\bar{\pi}$ with symbols of a particular degree chosen uniformly randomly. Using these abstractions and assumptions, we state the following computational problem as follows:

Definition 2.1. *Given a set S of data units and an encoded symbol s of degree d, the distance of s from set S, $dist(s, S)$, is the number of data units that form the symbol s and are not present in S.*

It is easy to see that if S is the set of recovered data units, then a symbol s recovers a new data unit if and only if $dist(s, S)$ is 1.

Definition 2.2. *The iterative decoder* D *works as follows.*

1. *Decode all degree* 1 *symbols and add them to set S. So, initially S is the set of distinct data units contained in all degree* 1 *symbols.*
2. *From the remaining symbols, choose a symbol s such that $dist(s, S)$ is the minimum.*
3. *If $dist(s, S) = 0$, throw away this symbol as a redundant or duplicate symbol.*
4. *If $dist(s, S) = 1$, decode a new data unit and add it to S. Goto Step 2.*
5. *If $dist(s, S) > 1$, stop. The remaining symbols have distance greater than* 1 *from S and are useless.*

This is the decoder used by Tornado and LT codes. It may not decode all possible data units from the given encoded symbols. We can recover more using a Gaussian elimination method. But we will use this decoder since its much faster compared to Gaussian elimination.

Consider another decoder *D*' which given a fixed sequence of encoded symbols $s'_1, s'_2, \ldots, s'_k$, works as follows. Initially the set S' is empty.

1. At the i^{th} step, choose symbol s_i. If it has a distance 1 from current set S', then decode a new data unit and add to set S'.
2. If s_i has distance 0 or more than 1, throw it away.

The following result holds:

Lemma 2.3. *Given a sequence of symbols $s_1, s_2, \ldots, s_k$, the number of data units decoder* D *can recover from this set of symbols is at least as much as that recovered by decoder* D*' given any fixed sequence $s'_1, s'_2, \ldots, s'_k$ which is a permutation of the symbols $s_1, s_2, \ldots, s_k$.*

The proof is omitted due to space constraints.

Problem 2.4. *There are N data units $\bar{x}$. Given k encoded symbols are recovered, what is the degree distribution $\bar{\pi}$ employed for encoding so that the expected number of x_i's that can be decoded from the k encoded symbols is maximized if we use the iterative decoder D described above.*

Definition 2.5. *If k encoded symbols are generated such that their degrees are chosen according to a probability distribution $\bar{\pi}$ and symbols of a particular degree are chosen uniformly randomly, then $\bar{\pi^*}$ is optimal for k if the expected number of data units recovered by decoder* D *is at least as much if the k symbols had been generated according to some other probability distribution*

3 Designing Optimal Degree Distributions for Iterative Decoding

3.1 Main Results

Let

$$k_1 = \sum_{i=0}^{\frac{N}{2}-1} \frac{N}{N-i}, r_i = \frac{iN}{i+1} \forall_{i \in [1, N-1]} \tag{1}$$

If $s_1, s_2, \ldots, s_k$ is a set of k encoded symbols, then let $f(s_1, \ldots, s_k)$ be the number of data units that can be recovered from these symbols using decoder *D*.

Let $C(i,j)$ be the expected number of data units that can be recovered from a set of $i+j$ encoded symbols, i of which are of degree 1 and the rest of degrees 2 or more. Specifically, if $a_1,\ldots,a_i$ are i degree 1 symbols and $b_1,\ldots,b_j$ are j symbols of degree 2 or more, then $C(i,j) = f(a_1,\ldots,a_i,b_1,\ldots,b_j)$.

Lemma 3.1. *Given $k = i+j$ encoded symbols of which i symbols $a_1,\ldots,a_i$ are of degree 1 and j symbols $b_1,\ldots,b_j$ are of degree 2 or more, then $\forall_{j>0} C(k-j,j) \leq C(k,0)$ if $C(k-j,j) \leq \frac{N}{2} (= r_1)$*

The proof is omitted due to space constraints.

Theorem 3.2. *To recover r data units, such that $r \leq r_1 = \dfrac{N}{2}$, the optimal degree distribution has only degree 1 symbols*

Proof. We will prove this by contradiction. Suppose there is an optimal degree distribution $\bar{\pi}^{opt}$ such that $\pi_1^{opt} < 1$. Consider a set of k encoded symbols according to this degree distribution such that there are $k-j$ degree 1 symbols and j symbols of degree 2 or more. The expected number of data units recovered is $r = C(k-j,j) \leq \frac{N}{2}$.

Using Lemma 3.1, we have $C(k,0) \geq C(k-j,j)$ and hence $\bar{\pi}^{opt}$ is not optimal for the given k according to Definition 2.5.

Hence, the optimal degree distribution is given by $\bar{\pi} : \pi_1 = 1, \forall_{i>1} \pi_i = 0$.

□

Theorem 3.3. *To recover $r_1 = \dfrac{N}{2}$ symbols, the expected number of encoded symbols required is $E[k_1] = \displaystyle\sum_{i=0}^{\frac{N}{2}-1} \frac{N}{N-i}$ which approaches $\dfrac{3N}{4}$ from below as N goes to infinity*

Proof. From Theorem 3.2, the optimal degree distribution will contain only degree 1 symbols to recover the first $\frac{N}{2}$ data units. Since all the encoded symbols have to be degree 1, this is the well studied *Coupon Collector's Problem* where to get the first r distinct coupons, one needs to collect $\displaystyle\sum_{i=0}^{r-1} \frac{N}{N-i}$ coupons.

We omit the second part of the proof due to space constraints. □

Theorems 3.2 and 3.3 show that if most of the network nodes fail and less than three-fourth of the data survives, then not using any coding is the best way to recover maximum number of symbols.

Theorem 3.4. *To recover r data units such that $r \leq r_j = \dfrac{jN}{j+1}$, the optimal degree distribution has symbols of degree j or less only*

The proof is along the same lines as that for Theorem 3.2 and we omit it due to space constraints.

Theorem 3.5. *To recover $r_j = \dfrac{jN}{j+1}$ symbols, the expected number of encoded symbols required is at most $E[k_{j+1}] \leq E[k_j] + \displaystyle\sum_{i=r_j}^{r_{j+1}-1} \frac{N^{j+1}}{(j+1)i^j(N-i)}$*

The proof is along the same lines as that for Theorem 3.3 and we omit it due to space constraints.

3.2 A Close to Optimal Degree Distribution

According to the analysis in section 3.1, it is best to use only degree 1 symbols to recover the first r_1 data units, only degree 2 symbols to recover the next $r_2 - r_1$ data units and so on. This defines a natural probability distribution on the degrees of the encoded symbols. In particular, if we have a total of k encoded symbols, we should have k_1 degree 1 symbols, $k_2 - k_1$ degree 2 symbols and so on as long as some of k symbols are remaining. The degree distribution can thus be defined as

$$\bar{\pi}^* : \pi_i^* = max(0, min(\frac{k_i - k_{i-1}}{k}, \frac{k - k_{i-1}}{k})) \tag{2}$$

Note that the values of $E[k_1]$ in Theorem 3.3 is exact whereas those of $E[k_{i:i>1}]$ are only upper bounds. Hence the degree distribution $\bar{\pi}^*$ is an approximation to the optimal distribution.

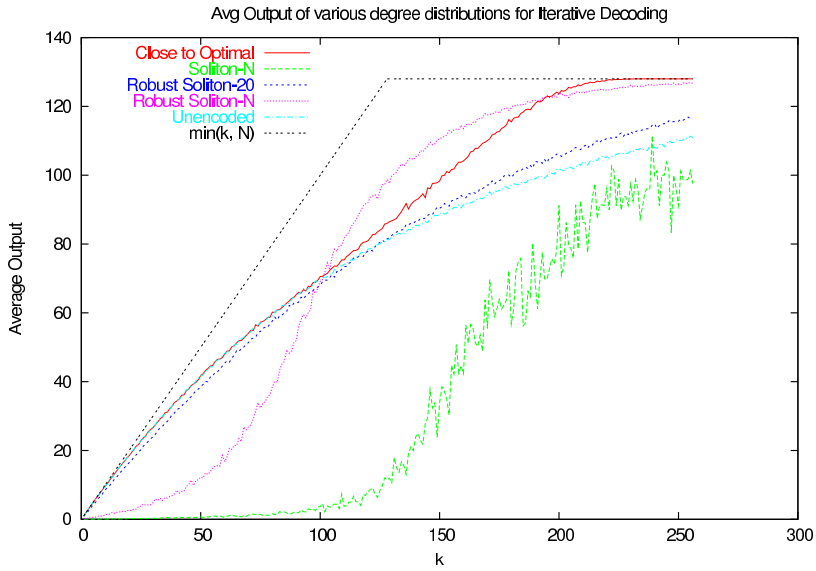

Figure 1: Comparing the performance of various degree distributions. N is chosen to be 128

4 Simulations

We compare the performance of the degree distribution given by Equation 2 to some well know distributions used in related works using simulations.

Soliton-α is defined as $\bar{\pi}^{S-\alpha} : \pi_1^{S-\alpha} = \frac{1}{\alpha}, \pi_{i\in[2,\alpha]}^{S-\alpha} = \frac{1}{i(i-1)}$. This distribution and a slightly modified form called the Robust Soliton-α are discussed in [9].

Figure 1 shows the average number of data units recovered using decoder *D* for varying k values and different degree distributions. N is chosen as 128. For nearly all values of k, $\bar{\pi}^*$ performs very well especially for low values of k which means that it is a good distribution to use if the network failures tend to be large.

References

[1] S. Acedanski, S. Deb, M. Medard and R. Koetter, "How Good is Random Linear Coding Based Distributed Networked Storage," in *Workshop on Network Coding, Theory and Applications*, 2005.

[2] A. G. Dimakis, V. Prabhakaran and K. Ramchandran, "Ubiquitous Acess to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," in *Symposium on Information Processing in Sensor Networks*, 2005.

[3] R. Ahlswede, N. Cai, S. Y. R. Li and R. W. Yeung, "Network Information Flow," in *IEEE Transactions on Information Theory*, 2000, vol. 46, pp. 1004–1016.

[4] N. Cai, S. Y. R. Li and R. W. Yeung, "Linear Network Coding," in *IEEE Transactions on Information Theory*, 2003, vol. 49, pp. 371–381.

[5] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," in *ACM/IEEE Transactions on Networking*, 2003, vol. 11, pp. 782–795.

[6] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," in *Proceedings of INFOCOM*, 2005.

[7] Lin and Costello, *Error Control Coding: Fundamentals and Applications*, 1983.

[8] M. Luby, M. Mitzenmacher, M. A. Shokrollahi and D. Spielman, "Efficient Erasure Correcting Codes," in *IEEE Transactions on Information Theory*, 2001, vol. 47, pp. 569–584.

[9] M. Luby, "LT Codes," in *Symposium on Foundations of Computer Science*, 2002.