Encoding for Persistent Sensor Networks

Abhinav Kamra, Vishal Misra[*], Dan Rubenstein[†]
Dept. of Computer Science
Columbia University
New York, NY
{kamra, misra, danr}@cs.columbia.edu

Jon Feldman
Dept. of IEOR
Columbia University
New York, NY
jonfeld@ieor.columbia.edu

Abstract

1 Introduction and Related Work

Sensor networks consist of a number of sensors spread across a geographical area. Each sensor has communication capability and some level of intelligence for signal processing and networking of data. Each sensor node in the network routinely 'senses' and stores data from its immediate environment. An important requirement of the sensor network is that the collected data be disseminated to the proper end users. In some cases, there are fairly strict requirements on this communication. For example, the detection of an intruder in a surveillance network should be immediately communicated to the police authorities. Each sensor node also has some storage capacity to store the collected data or to assemble the data prior to communicating it to another node.

Sensor networks are commonly deployed in environments where the various nodes collect data and the whole network acts as a distributed database. The popular approach to retrieving data in sensor networks is for a user or the owner of the sensor network to query an individual node or a group of nodes for information collected in the region. The desired data can then be routed from the source nodes to the querying node. Depending on the amount of data fusion to be performed, it may not be feasible to transmit a large amount of data across the network. Potentially, there can be a large latency in getting the required data out of a multitude of source nodes scattered randomly across the network due to the multi-hop routing phase to be performed after the query. In general, collecting data from multiple nodes in the sensor network at query time may introduce latency that may not be acceptable for certain applications. Additionally, it may also be unreliable since typical sensor nodes are wireless nodes with limited storage, bandwidth 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 and hence the required data may not even be available at a subset of the sensor nodes.

A neat solution is for various local sink nodes to collect the data from a given area. A query may be directed to the sink node nearest to the desired location which has all the

[*]Also with the Department of Electrical Engineering
[†]Also with the Department of Electrical Engineering

required data to compute the result of the query and respond directly to the querying node. Even if all the data stored at the sensor nodes in the area has not been collected at the sink node, it can compute the query results as best as it can from the data it has. Due to the unreliability of the sensor nodes and the communication links between them, the sink node needs to collect the data from the nodes in the area as fast as possible. Since the underlying topology of the sensor network may not allow every node to be connected to every other node, sensor nodes in the network are required to store data from other nodes to route data through the network towards the sink node. This may also be beneficial in a wireless sensor network where it is more energy efficient for a sensor node to forward data only to nearby nodes and have a multi-hop communication to the sink instead of forwarding directly to the destination using higher power. Since the sensor nodes or the entire network may fail at any time, it is important to have the sink collect as much data as it can at any given time.

This work is accordingly motivated at trying to maximize the information content that a sink can collect at any time. We show how a collection of sensor nodes connected in any topology, not knowing where the sink is located, can use their limited storage and bandwidth to exchange data such that the sink node, which is connected directly to 1 or more of the sensor nodes, can accumulate all the data collected by the sensor nodes as fast as possible. At any given time, if the sensor network fails completely or partially, the sink will have as much data as possible to respond to the queries.

We consider large sensor networks with individual nodes severely constrained by storage memory, computational power and network bandwidth available to communicate with neighbor nodes. In particular we assume a sensor network with the individual nodes connected in a certain underlying topology such that each node can only communicate directly with its neighbors. Each node 'senses' some data from its immediate environment and generates a data unit of a fixed size. Each node has a fixed storage size where it can store 1 or more data units. The idea is for the sensor nodes to exchange their data units at every time step such that the sink node can accumulate all of them eventually. The sink node is attached to and hence collects data from one or more of the sensor nodes. We assume a "Pull" based mechanism to exchange data. In particular, a node selects a sender node from one of its neighbors to receive data from. Both the sending and the receiving nodes choose one of the data units from their respective limited storage and exchange the selected data units.

It has been shown [1, 2] that in distributed storage networks, 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, 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.

The above mentioned encoding protocols, in most cases cannot directly be applied and in other cases are not particularly well suited to our distributed setting to achieve the goal of accumulating the maximum possible information at the sink at any given time. Accordingly, we design a distributed protocol of information encoding and communication which is well suited to enable substantial amount of original data to be recovered even from a small number of encoded symbols.

The rest of the paper is organized as follows: In Section 2, we mathematically formulate the problem and give some definitions. In Section 3 we prove some results and design an approximately optimal degree distribution.

2 Problem Formulation

Our network model consists of a sensor network with N nodes, connected in a certain topology, each with a fixed limited storage capacity. Each node n_i generates or senses some data from its immediate environment and generates a fixed size data x_i which we call a "data unit". Since the aim is to spread the data in the network so that the sink node is able to accumulate as much information as is possible at any given time, and since we have determined that data should be encoded for maximum information transfer to the sink, each node stores encoded symbols of data instead of the original data units. These encoded symbols are formed by bitwise XOR-ing a subset of the N data units $x_1, \ldots, x_N$ as is done in LT-codes. The number of data units which are XOR-ed to form a symbol is known as the degree of the symbol. For example, $x_1 \oplus x_3 \oplus x_6$ is an encoded symbol of degree 3. These encoded symbols are formed according to a degree probability distribution $\bar{\pi}$ such that an encoded symbol is of degree i with probability π_i. The i data units that form the symbol are chosen uniformly randomly from the N data units.

Each sensor node, at every time step, exchanges one of the stored symbols with one of its neighbors. Lets suppose the sink is attached to exactly one of the sensor nodes and hence observes the new symbol which this node receives at every time unit. The sink therefore, accumulates one encoded symbol per unit time. Using a decoding procedure the same as one used in LT-codes, the sink is able to "recover" some of the original data units from the accumulated encoded symbols. The symbols which cannot be decoded at any time due to insufficient information to decode them are stored for future decoding. The decoding algorithm is explained in Definition 2.4.

In practice, the content and spread of encoded symbols will depend on the particular data exchange protocol between the nodes and the topology of the network so that the sensor nodes are unable to construct encoded symbols conforming to the degree distribution $\bar{\pi}$ according to which they are supposed to be constructed. But for the purpose of analysis, we assume that the encoded symbols accumulated at the sink have been constructed according to some fixed degree distribution $\bar{\pi}$. Using these abstractions and assumptions, we state the following computational problem:

Problem 2.1 *There are N data units $x_1, \ldots, x_N$. Given that the sink receives one encoded symbol every time step, what is the degree distribution $\bar{\pi}$ according to which the symbols should be constructed such the maximum number of data units can be decoded at the sink at any time. The sink uses the decoder* D *as described in Definition 2.4.*

We want to receive symbols at the sink node such that at any time, the maximum possible number of data units can be recovered so that if the sensor network completely or partially fails at any time, we have recovered as much data as possible. It is quite possible that there is no degree distribution $\bar{\pi}$ according to which the symbols should be constructed such that it maximizes the number of data units can be decoded at the sink at "all" times. Hence, we give an alternate definition of the optimal degree distribution which can maximize the number of data units that can be recovered at a given time.

Definition 2.2 *If k encoded symbols are received at the sink node then the degree distribution $\bar{\pi}$ is optimal for the given k if it maximizes the number of data units that can be recovered from these k encoded symbols. The k symbols have been constructed according to the degree distribution $\bar{\pi}$.*

We now state some definitions and describe how the decoding algorithm at the sink node works.

Definition 2.3 *Given a set X of data units and an encoded symbol s of degree d, the distance of s from set X, $dist(s, X)$, is the number of data units that form the symbol s and are not present in X.*

It is easy to see that if X is the set of recovered data units, then a symbol s can be used to recover a new data unit if and only if $dist(s, X)$ is 1 since all but one of the data units which are XOR-ed to form the symbol s have already been recovered, so they can be "subtracted" from s to recover the additional data unit. The decoder we describe below uses this principal to recover data units from the set of received symbols $s_1, s_2, \ldots, s_k$.

Definition 2.4 *The iterative decoder* D *works as follows. Initially the set X of recovered data units is empty.*

1. *Decode all degree 1 symbols and add them to set X. This is trivial since the degree 1 symbols are the same as the original data units. So, initially X is the set of distinct data units contained in all degree 1 symbols.*

2. *From the remaining symbols, choose a symbol s such that $dist(s, X)$ is the minimum.*

3. *If $dist(s, X) = 0$, throw away this symbol as a redundant or duplicate symbol.*

4. *If $dist(s, X) = 1$, decode a new data unit and add it to X. Goto Step 2.*

5. *If $dist(s, X) > 1$, stop. The remaining symbols have distance greater than 1 from X and hence cannot be decoded without additional information.*

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 F which given a Fixed sequence of encoded symbols $s'_1, s'_2, \ldots, s'_k$, works as follows. Initially the set X' of recovered symbols is empty.

1. Let $i = 1$.

2. Choose symbol s_i. If it has a distance 1 from current set X', then decode a new data unit and add to set X'.

3. If s_i has distance 0 or more than 1, throw it away.

4. Increment i and go to step 2 until all symbols have been considered.

In a sense, the decoder F is more constrained than the decoder D since it has to decode the symbols in a fixed order only and it also throws away all symbols it considers which cannot be decoded at that point but might have been useful at a later point in the decoding process.

The following result holds:

Lemma 2.5 *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* F *given any fixed sequence $s'_1, s'_2, \ldots, s'_k$ which is a permutation of the symbols $s_1, s_2, \ldots, s_k$.*

Proof: Let $D(s_1, s_2, \ldots, s_k)$ be the number of symbols recovered by decoder D and $F(s'_1, s'_2, \ldots, s'_k)$ be the number of symbols recovered by decoder F. We will prove by induction on the number of symbols that $D(s_1, s_2, \ldots, s_k) \geq F(s'_1, s'_2, \ldots, s'_k)$.

- [Basis Step: k=1]: Clearly, $D(s_1) = F(s'_1)$. Specifically, if s_1 is of degree 1, then $D(s_1) = F(s'_1) = 1$ and $D(s_1) = F(s'_1) = 0$ otherwise. Hence for $k = 1$, $D(s_1, s_2, \ldots, s_k) \geq F(s'_1, s'_2, \ldots, s'_k)$.
- [Induction Hypothesis]: $D(s_1, s_2, \ldots, s_{k-1}) \geq F(s'_1, s'_2, \ldots, s'_{k-1})$.
- [Inductive Step]: There are two cases:
  1. [Degree of $s'_1 > 1$]: This means that F cannot decode s'_1. Hence, $F(s'_1, s'_2, \ldots, s'_k) = F(s'_2, s'_3, \ldots, s'_k)$. Using the induction step we have, $D(s_2, \ldots, s_k) \geq F(s'_2, \ldots, s'_k)$. Hence, $D(s_1, s_2, \ldots, s_k) \geq D(s_2, \ldots, s_k) \geq F(s'_1, s'_2, \ldots, s'_k)$.
  2. [Degree of $s'_1 = 1$]: In this case decoder F decodes s'_1. Hence, $F(s'_1, s'_2, \ldots, s'_k) = 1 + F(s'_2, s'_3, \ldots, s'_k)$.
     But since s_1 is of degree 1, decoder D also decodes s_1 in the first step when it decodes all degree 1 symbols. Hence, WLOG we can assume $s_1 = s'_1$. Therefore, $D(s_1, s_2, \ldots, s_k) = 1 + D(s_2, s_3, \ldots, s_k)$.
     Using the induction hypothesis, we have $D(s_1, s_2, \ldots, s_k) = F(s'_1, s'_2, \ldots, s'_k)$.

■

3 Designing Optimal Degree Distributions for Iterative Decoding

In Section 3.1, we prove some theorems which are critical in designing a close to optimal degree distribution in Section 3.2.

3.1 Main Results

Define decoder S as follows which given a set of encoded symbols $s_1, s_2, \ldots, s_k$, works as follows: Initially the set X of recovered symbols is empty.

1. Sort the symbols in non-decreasing order of their degrees to get the sequence $s'_1, s'_2, \ldots, s'_k$.

2. Run decoder F on the sequence $s'_1, s'_2, \ldots, s'_k$.

Decoder S is essentially a special case of decoder F which works on the sequence of symbols sorted in non-decreasing order of their degrees.

We now prove some properties of decoder S.

Lemma 3.1 *Let $\rho_{r,d}$ be the probability of successful decoding of a degree d symbol when r data units have already been recovered. This is the probability that the degree d symbols contains $d-1$ components which are contained in the r recovered data units and 1 component which has not been recovered. Then, $\rho_{r,d} = \dfrac{{}^{r}C_{d-1}(N-r)}{{}^{N}C_d}$.*

Proof: The d component data units of the degree d symbol are assumed to be distinct and uniformly chosen from the N data units. The number of ways of choosing a d degree symbol such that the component data units are distinct and are spread uniformly randomly is ${}^{N}C_d$. There are r recovered data units and $N-r$ unrecovered data units. For a degree d symbol, the number of ways of choosing 1 component from the $N-r$ unrecovered symbols is ${}^{N-r}C_1$. Similarly, the number of ways of choosing $d-1$ components from the set of r recovered symbols is ${}^{r}C_{d-1}$.

Hence the probability that for a degree d symbol, $d-1$ components are from the set of r recovered symbols and 1 from the set of $N-r$ unrecovered symbols is $\rho_{r,d} = \dfrac{{}^{r}C_{d-1}(N-r)}{{}^{N}C_d}$. ■

Lemma 3.2 *When using decoder* S*, to recover more than r data units, the optimal degree distribution contains symbols of degree j or more only if $\forall_{1\leq i<j}[\rho_{r,i} < \rho_{r,j}]$.*

Proof: Consider two sets of symbols sorted in non-decreasing order of their degrees such that the first symbol they differ in is the k^{th} symbol, i.e., $\forall_{1\leq<i<k} s_i = s'_i$ but $s_k \neq s'_k$.

Suppose s'_k is of degree j while s_k is of a smaller degree. Further suppose that after decoding the first $k-1$ symbols, r data units have been recovered when using decoder S. This the case for both sets of sequences since they are identical in the first $k-1$ symbols.

The probability of successful decoding of the k^{th} symbol for the first sequence is $\rho_{r,i}$ for some $i<j$, while that for the second sequence is $\rho_{r,j}$. It is easy to see that all the symbols $s_1, \ldots, s_{k-1}$ are of degrees smaller than j since s_k is of degree smaller than j and the symbols are sorted in non-decreasing order of degrees.

Hence, to recover more than r data units, the optimal degree distribution contains symbols of degree j only if $\forall_{1\leq i<j}[\rho_{r,i} < \rho_{r,j}]$. ■

Now we consider some properties of decoder D, which will be used in our sensor network protocols.

Let

$$R_1 = \frac{N-1}{2}, R_2 = \frac{2N-1}{3}, \ldots, R_i = \frac{iN-1}{i+1}\forall_{i\in[1,N-1]} \tag{1}$$

and

$$K_1 = \sum_{i=0}^{R_1-1} \frac{N}{N-i} \tag{2}$$

Theorem 3.3 *To recover r data units, such that $r \leq R_1 = \dfrac{N-1}{2}$, the optimal degree distribution has symbols of degree 1 only when using decoder* D

Proof: This follows from Lemma 3.2.

It is easy to see that in the beginning, decoder D behaves the same as decoder S since both decoders decode degree 1 symbols before decoding any other degree symbols. Also, since decoder S is a special case of decoder F, using Lemma 2.5, we see that decoder D is provably better than decoder S when given the same set of symbols.

According to Lemma 3.2, decoder S uses only degree 1 symbols to recover the first r data units as long as $\rho_{r,1} > \rho_{r,2}$. This is true for $r \leq \frac{N-1}{2}$.

Hence to recover the first $r = \frac{N-1}{2}$ data units, the optimal degree distribution of symbols when using decoder S has only degree 1 symbols.

Since decoder D works the same way as decoder S when the set of symbols has only degree 1 symbols coupled with the fact that decoder D is provably better than decoder S, it follows that the optimal degree distribution for decoder D to recover r ($r \leq \frac{N-1}{2}$), consists of degree 1 symbols only.

■

Theorem 3.4 *To recover* $R_1 = \dfrac{N-1}{2}$ *data units, the expected number of encoded symbols required is* $K_1 = \displaystyle\sum_{i=0}^{R_1-1} \frac{N}{N-i}$ *when using decoder* D.

Proof: From Theorem 3.3, the optimal degree distribution will contain only degree 1 symbols to recover the first $R_1 = \frac{N-1}{2}$ data units. Since all the encoded symbols have to be degree 1, this becomes 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.

Hence the expected number of symbols required is K_1 which is given by Equation 2.

■

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

Theorems 3.3 and 3.4 show that if most of the network nodes fail and a small amount of the data survives, then not using any coding is the best way to recover maximum number of data units. We generalize these proofs in theorems 3.6 and 3.7. But before that we prove a lemma which is useful for the theorems.

Lemma 3.5 *If X of size r is the set of symbols recovered by decoder* D *at any point during its execution and $a_1, \ldots, a_k$ are the remaining symbols to be considered then the output of decoder* D*, $D(X, a_1, \ldots, a_k)$ is at least as much as $D(X', a_1, \ldots, a_k)$ if X' is another set of recovered data units but of a smaller size*

Proof: We will prove this by induction on k.

- [Basis Step: k = 0]: Since $\|X\| > \|X'\|$, trivially $D(X) \geq D(X')$.
- [Induction Hypothesis]: $D(X, a_1, \ldots, a_{k-1}) \geq D(X', a_1, \ldots, a_{k-1})$ for $\|X\| > \|X'\|$
- [Inductive Step:] Let $X_{k-1} = r$ be the set of recovered data units after decoding $a_1, \ldots, a_{k-1}$ starting with X. Similarly, let $X'_{k-1} = r'$ be the set of recovered data units after decoding $a_1, \ldots, a_{k-1}$ starting with X'. According to the induction hypothesis, $\|X_{k-1}\| > \|X'_{k-1}\|$ which means $r \geq 1 + r'$ since both r and r' are integral.

$D(X, a_1, \ldots, a_k) = \|X_{k-1}\| + p_{a_k}$ where p_{a_k} is the probability that a_k has distance 1 from set X_{k-1}. Similarly, $D(X', a_1, \ldots, a_k) = \|X'_{k-1}\| + p'_{a_k}$ where p'_{a_k} is the probability that a_k has distance 1 from set X'_{k-1}.

Hence,

$$\begin{aligned} D(X, a_1, \ldots, a_k) - D(X', a_1, \ldots, a_k) &= r + p_{a_k} - r' - p'_{a_k} \\ &\geq 1 + p_{a_k} - p'_{a_k} \\ &\geq 0 \end{aligned}$$

■

Theorem 3.6 *To recover r data units such that $r \leq R_j = \dfrac{jN-1}{j+1}$, the optimal degree distribution has symbols of degree j or less only*

Proof: We will prove this by contradiction. Suppose there is an optimal degree distribution $\bar{\pi}^{opt}$ to recover r data units such that $\sum_{i=1}^{j} \pi_i^{opt} < 1$. Consider a set of k encoded symbols $s_1, s_2, \ldots, s_k$ according to this degree distribution such that there are 1 or more symbols of degrees greater than j. WLOG, lets assume $s_1, s_2, \ldots, s_k$ is the order in which decoder D considers the symbols while decoding.

The expected number of data units recovered using decoder D is given by $D(s_1, s_2, \ldots, s_k) \leq R_j$ according to the problem statement. Since there are symbols of degrees greater than j, let s_l be the first symbol in this sequence of degree greater than j, that is degree$(s_l) = d > j$.

Let X be the set of data units recovered after decoding symbols $s_1, \ldots, s_{l-1}$. Let $r = \|X\|$. The output of the first l symbols is given by $D(s_1, s_2, \ldots, s_l) = r + p_{s_l}$ where p_{s_l} is the probability that symbol s_l of degree $d > j$ has a distance 1 from the set X. Again, according to the problem statement $r \leq R_j$.

If we replace symbol s_l of degree $d > j$ with symbol s'_l of degree j and decode the symbols using the decoder F then the output of decoder F given this fixed sequence of symbols $s_1, s_2, \ldots, s_{l-1}, s'_l$ is given by $F(s_1, s_2, \ldots, s_{l-1}, s'_l) = r + p_{s'_l}$.

Since $r \leq R_j = \frac{jN-1}{j+1}$, $p_{s_l} = \frac{{}^rC_{d-1}(N-r)}{{}^NC_d}$ and $p_{s'_l} = \frac{{}^rC_{j-1}(N-r)}{{}^NC_j}$, it is easy to see that $p_{s'_l} > p_{s_l}$ because $d > j$ and $r \leq R_j$.

Hence, $F(s_1, s_2, \ldots, s_{l-1}, s'_l) > D(s_1, s_2, \ldots, s_l)$. Using Lemma 2.5, we get $D(s_1, s_2, \ldots, s_{l-1}, s'_l) > D(s_1, s_2, \ldots, s_l)$. Hence the output of decoder D increases by replacing a symbol of degree greater than j by a symbol of degree j.

Now, using this result and Lemma 3.5 we get $D(s_1, \ldots, s_{l-1}, s'_l, s_{l+1}, \ldots, s_k)$ is at least as much as $D(s_1, s_2, \ldots, s_l, s_{l+1}, \ldots, s_k)$.

Hence by replacing higher degree symbols with symbols of degree j, the output of decoder D increases. Therefore, the optimal degree distribution to recover $r \leq R_j$ data units has only symbols of degree j or less.

■

Theorem 3.7 *To recover $R_j = \dfrac{jN-1}{j+1}$ data units, the expected number of encoded symbols required is at most $K_j \leq K_{j-1} + \sum_{i=R_{j-1}}^{R_j - 1} \dfrac{{}^NC_j}{{}^iC_{j-1}(N-i)}$*

Proof: We will prove this by induction on j.

- [Basis Step: j = 1]: Using Theorem 3.4, to recover $R_1 = \frac{N-1}{2}$ data units, the expected number of encoded symbols required is $K_1 = \sum_{i=0}^{\frac{N}{2}-1} \frac{N}{N-i}$

- [Induction Hypothesis]: To recover $R_{j-1} = \frac{(j-1)N-1}{j}$ data units, the expected number of encoded symbols required is at most K_{j-1}.

- [Inductive Step:] Let X of size $\|X\| = R_{j-1}$ be the set of data units recovered at a point during the execution of decoder D. According to Theorem 3.6, the optimal degree distribution will have symbols of degrees $j-1$ or less. Furthermore, to recover the remaining $R_j - R_{j-1}$ symbols, the optimal degree distribution will have symbols of degrees j or less.

  A degree d symbol has a distance of 1 from the set X of size r with probability $\frac{{}^rC_{d-1}(N-r)}{{}^NC_d}$. For $R_{j-1} \leq r < R_j$, this probability is maximized if the symbols are of degree j. Hence, it is best to use symbols of degree j to recover the remaining $R_j - R_{j-1}$ symbols.

  When r data units, such that $R_{j-1} \leq r < R_j$ have been recovered, the next degree j symbol will have distance 1 from the set of recovered data units with probability $p = \frac{{}^rC_{j-1}(N-r)}{{}^NC_j}$. If in the worst case, we do not use the symbol later if it turns out to be of distance 2 or more, then the expected number of symbols required to recover the next data unit is $1.p + 2.p(1-p) + 3.p(1-p)^2 \ldots = \frac{1}{p}$.

  Hence, in the worst case, the expected number of degree j symbols required to recover $R_j - R_{j-1}$ data units is $\sum_{i=R_{j-1}}^{R_j-1} \frac{{}^rC_{j-1}(N-r)}{{}^NC_j}$ and hence

$$K_j \leq K_{j-1} + \sum_{i=R_{j-1}}^{R_j-1} \frac{{}^NC_j}{{}^iC_{j-1}(N-i)}$$

■

3.2 A Close to Optimal Degree Distribution

According to the analysis in section 3.1, we observe that 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. Furthermore, an expected number of K_1 encoded symbols are required to recover R_1 data units, an expected maximum K_2 symbols are required 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 so that we can recover an expected R_1 data units from them, $K_2 - K_1$ degree 2 symbols so that we can recover an expected $R_2 - R_1$ data units from them and so on as long as some of k symbols are remaining. A close to optimal degree distribution can thus be defined as

$$\bar{\pi}^*(k) : \pi_i^* = max(0, min(\frac{K_i - K_{i-1}}{k}, \frac{k - K_{i-1}}{k})) \qquad (3)$$

Note that the value of K_1 in Theorem 3.4 is exact whereas those of $K_{i:i>1}$ in Theorem 3.7 are only upper bounds. Hence the degree distribution $\bar{\pi}^*$ is an approximation to the optimal distribution.

4 At-Sink Simulation Model

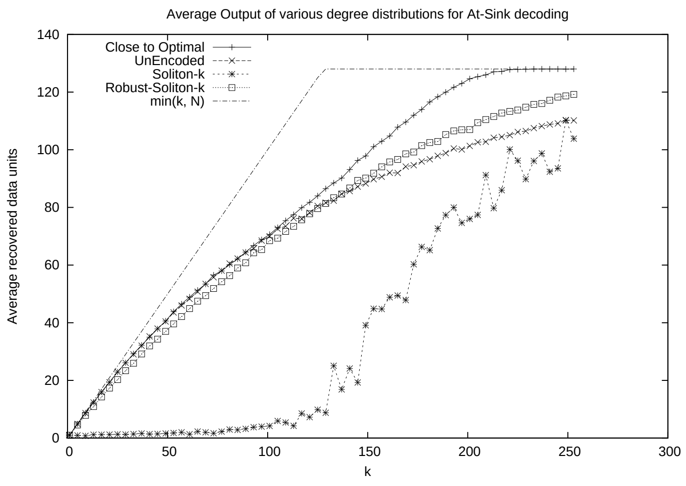

Figure 1: Comparing the performance of various degree distributions. N is chosen to be 128

Equation 3 gives an approximately optimal degree distribution to encode the symbols so as to recover as many data units as possible at the sink at any given time. Here we evaluate how a communication protocol based on this degree distribution performs in terms of number of data units that can be recovered at the sink at any given time.

To simulate the protocol, we generate k symbols independently according to the degree distribution $\bar{\pi}^*(k)$. We assume that these are the k symbols which are received at the sink node at time k (since the sink receives one encoded symbol at each time step). The sink can also be attached to more than one sensor nodes, and hence receiving more than one encoded symbol per time step. It is easy to model but for the sake of clarity, we stick with the one symbol per unit time protocol. We then use the decoder algorithm D and see how many data units can be recovered using these k encoded symbols. This is repeated for different k values to observe how the protocol performs.

We call this the At-Sink simulation model since we are generating the k symbols independently according to some degree distribution and assuming that these are the symbols received at the sink. In a more practical protocol, the symbols received at the sink may not exactly follow the distribution using which they are constructed at the nodes because of the underlying topology of the sensor network and other factors such as storage limitations at the nodes.

We compare the performance of the degree distribution $\bar{\pi}^*(k)$ with the un-encoded distribution where all the symbols are of degree 1 as well as some other well known degree distributions such as Soliton and Robust Soliton.

The degree distribution Soliton-α is defined as

$$\begin{aligned}\bar{\pi}^{S,\alpha} &: \\ \pi_1^{S,\alpha} &= \frac{1}{\alpha} \\ \pi_{i\in[2,\alpha]}^{S,\alpha} &= \frac{1}{i(i-1)} \\ \pi_{i\in[\alpha+1,N]}^{S,\alpha} &= 0 \end{aligned} \tag{4}$$

This distribution and a slightly modified form which performs better in practice and known as the Robust Soliton are discussed in [9]. The original Soliton and Robust Soliton distributions are fixed distributions which depend on N, the number of data units. We define Soliton-α and Robust-Soliton-α degree distributions as the original Soliton and Robust Soliton distributions but after substituting the parameter α for N in their specifications. In this manner these distributions vary with changing α. This is useful because Soliton and Robust Soliton distributions are essentially designed for recovering all N data units with the minimum symbols possible and not for partial recovery from the surviving symbols as is done by $\bar{\pi}^*(k)$. Hence to adapt these distributions for the task of recovering data units from a very small number of symbols, we construct modified forms of them which are Soliton-α and Robust-Soliton-α. The values of Robust Soliton parameters c and δ are chosen to be 0.9 and 0.1 respectively as suggested by the authors.

Figure 1 shows the average number of data units recovered using decoder D for varying k values and different degree distributions. We compare the approximately optimal degree distribution $\bar{\pi}^*(k)$ with Soliton-k and Robust-Soliton-k distributions. We also compare with the case when no coding is performed. N is chosen as 128. For nearly all values of k, $\bar{\pi}^*(k)$ 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. The good performance of the No-Coding protocol for small k values confirms the fact that if the number of symbols is small, they should be degree 1 to ensure recovery of maximal data units. Robust-Soliton-k distribution performs as well as the UnEncoded. Soliton-k does not perform very well for small k values since it assigns high probabilities to symbols of higher degrees.

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*, vol. 46, 2000, pp. 1004–1016.

[4] N. Cai, S. Y. R. Li and R. W. Yeung, "Linear Network Coding," in *IEEE Transactions on Information Theory*, vol. 49, no. 2, 2003, pp. 371–381.

[5] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," in *ACM/IEEE Transactions on Networking*, vol. 11, no. 5, 2003, 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*, vol. 47, no. 2, 2001, pp. 569–584.

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