Decentralized Compression and Predistribution via Randomized Gossiping

Michael Rabbat, Jarvis Haupt, Aarti Singh, and Robert Nowak [*]
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
{rabbat, jhaupt, singh}@cae.wisc.edu, nowak@engr.wisc.edu

ABSTRACT

Developing energy efficient strategies for the extraction, transmission, and dissemination of information is a core theme in wireless sensor network research. In this paper we present a novel system for decentralized data compression and predistribution. The system simultaneously computes random projections of the sensor data and disseminates them throughout the network using a simple gossiping algorithm. These summary statistics are stored in an efficient manner and can be extracted from a small subset of nodes anywhere in the network. From these measurements one can reconstruct an accurate approximation of the data at all nodes in the network, provided the original data is compressible in a certain sense which need not be known to the nodes ahead of time. The system provides a practical and universal approach to decentralized compression and content distribution in wireless sensor networks. Two example applications, network health monitoring and field estimation, demonstrate the utility of our method.

Categories and Subject Descriptors

E.4 [**Coding and Information Theory**]: Data compaction and compression; I.5.4 [**Pattern Recognition**]: Signal Processing

General Terms

Algorithms, Theory

Keywords

Random projections, decentralized compression, predistribution in sensor networks

[*]This work was supported by the National Science Foundation under Grants CCR-0310889 and CCF-0325390.

1. INTRODUCTION

Sensor networking is an emerging technology that promises an unprecedented ability to monitor and manipulate the physical world via a spatially distributed network of small, inexpensive wireless sensor nodes that have the ability to self-organize into a well-connected network. While the practically unlimited range of sensor network applications is quite evident, our current understanding of their design and management is far from complete. The decentralized nature of data collection necessitates sharing information via wireless communications, a costly operation due to limited energy resources. Consequently, a major challenge in sensor networking is the development of energy efficient methods for processing and communicating information.

Previous approaches to energy-efficient extraction and storage have been driven by the notion that measurements at nearby sensors are correlated, predict one another well, or otherwise contain roughly the same information. Sampling techniques reduce the number of transmissions by only collecting measurements from a subset of nodes. Information theoretic methods typically make assumptions about the correlation structure governing measurements at nearby nodes and use this structure to encode each measurement in fewer bits.

This paper introduces a novel approach to distributed compression and predistribution in wireless sensor networks (WSNs), based in part on very recent results in compression theory, distributed computing, and network coding. Our approach is based on the notion that the sensor data viewed across the network will be compressible under some linear transformation. Suppose we have a network of n nodes indexed $1, 2, \dots,$ where each node makes a single measurement, x_i. Then the data in the entire network can be represented as a vector $\mathbf{x} \in \mathbb{R}^n$, obtained by stacking each node's measurement. Transform coding techniques are used in many successful compression schemes such as JPEG, MPEG, and MP3. The idea of transform coding is to find a basis[1], $\{\boldsymbol{\psi}_i\}_{i=1}^n$, for $\mathbb{R}^n$ which admits a sparse representation of $\mathbf{x}$; i.e., we want to find a basis so that for $k \ll n$,

$$\mathbf{x} \approx \sum_{i=1}^{k} (\mathbf{x}^T \boldsymbol{\psi}_i)\boldsymbol{\psi}_i + \sum_{i=k+1}^{n} 0\boldsymbol{\psi}_i.$$

Then most of the energy or information in the signal, $\mathbf{x}$,

[1]Recall that a basis for $\mathbb{R}^n$ is a collection of vectors, $\boldsymbol{\psi}_i \in \mathbb{R}^n$, $i = 1, \dots, n$, such that any $\mathbf{x} \in \mathbb{R}^n$ can be represented as $\mathbf{x} = \sum_{i=1}^n \theta_i \boldsymbol{\psi}_i$, where $\theta_i = \mathbf{x}^T \boldsymbol{\psi}_i$ are called the *coefficients* of $\mathbf{x}$ in the basis $\{\boldsymbol{\psi}_i\}$.

is captured in just a few ($k \ll n$) of the coefficients. A major challenge is that one generally does not know ahead of time which coefficients, $\mathbf{x}^T \boldsymbol{\psi}_i$, are significant. Moreover, in exploratory data analysis, the system designer may not even know what basis the data is compressible in.

To relate this setup to a specific WSN application, suppose that most nodes are functioning suitably well, but some small number detect that their sensors have become corrupted and need to be re-calibrated or replaced. The corrupted sensors set a flag $x_i = 1$, and the other functioning sensors set $x_i = 0$. Assuming that only $k \ll n$ nodes have malfunctioned, the signal, $\mathbf{x}$, is sparse or compressible in the usual "coordinate" basis for $\mathbb{R}^n$ where $\boldsymbol{\psi}_i$ is a vector of all zeros except $\psi_{i,j} = 1$. It is impossible to know which nodes will need repairs ahead of time. One solution would be to have damaged nodes transmit a special purpose message to the fusion center using a specialized routing protocol, however there is a large amount of overhead involved in maintaining routes in unreliable networking conditions. In addition, it may be desirable for the repairman, scientist, or soldier servicing the network to be notified of malfunctioning nodes at an arbitrary location within the network rather than having to return to a fusion center.

More generally, it may be that $\mathbf{x}$ is compressible in another basis where computing each coefficient, $\mathbf{x}^T \boldsymbol{\psi}_i = \sum_{j=1}^{n} x_j \psi_{i,j}$, involves the values from all nodes. This situation arises in field estimation, where an "image" of sensor readings could be compressed (e.g., using JPEG 2000) if the data were centrally located. An interesting question is then: *Can we obtain the significant coefficients in some transform without extracting the complete data from all sensors if we don't know which coefficients are significant ahead of time?* The answer is yes and this paper explains how.

Our approach to efficiently extracting information in a WSN employs state-of-the art compression techniques based on random projections of the data to efficiently summarize the information in $\mathbf{x}$. Exciting recent advances in compressive sensing demonstrate that a compressible signal can be well approximated using projections onto random vectors, alleviating the need for prior knowledge of which coefficients are the significant ones in the compressible basis [6, 12, 15]. We propose a scheme where the same set of random vectors are used to encode multiple vectors of sensor data. We derive bounds on the reconstruction error as a function of the compressibility (roughly speaking, the number of significant coefficients), the number of random projections collected, and the number of data vectors compressed.

Computing each projection amounts to a simple aggregation. Although there are many ways this computation can be performed, *gossip* or *consensus* algorithms are a simple and robust scalable means of evaluating one projection with the added bonus that every node has an approximation to the projection coefficient when the algorithm terminates. Based on our performance bounds and recent convergence results for gossip algorithms we determine the trade-off between communication complexity and reconstruction error. Using an idealized communication scheme to compute k projection coefficients requires $O(nk)$ transmissions. This is order-wise the same communication complexity required by a clairvoyant pull-based scheme knowing exactly which k coefficients are significant, tasked with extracting an approximation of equivalent accuracy to a single location in the network. Using a more practical geographic gossip algorithm to compute k projection coefficients requires $O\big(kn^{3/2}/\sqrt{\log n}\big)$ transmissions. We also describe an efficient storage scheme whereby a user can obtain the k random projection coefficients by querying any (convenient) subset of $O(k \log n)$ nodes anywhere in the network.

1.1 Related Work

Distributed Source Coding. The first proposed approaches to reducing the amount of communication (and thus energy [21]) required to extract information from a wireless sensor network were founded on the principle that in a dense sensor network the data gathered at nodes will be highly correlated [13, 22, 26]. These schemes take an information-theoretic approach to reducing the number of bits needed to encode all of the network data. However they make assumptions on the underlying correlation structure of the sensor data which are difficult to verify or involve parameters which may be difficult to estimate in practice.

Decentralized Error Codes. In [11], Dimakis et al. describe a system which is similar in spirit to ours. We note that [11] inspired quite a bit of the work presented here. Their setup involves a small number, k, of the nodes in the network which have data that needs to be predistributed to other nodes in a network. There are a few major differences between their system and ours. First, the assumption is made in [11] that the k data nodes are known ahead of time and that the data at these nodes is already in a compressed form. The error codes they propose ensure that a user can query any k nodes in the network and recover the data with high probability. In terms of communication complexity, their scheme requires $O(kn^{1/2}\log(n))$ single-hop transmissions. However a significant difference is that they assume $k = O(n)$. Thus, the effective communication complexity of their scheme is $O\big(n^{3/2}\log n\big)$.

Assuming the signal is compressible in some basis (as formalized below), by using $k \geq n^{1/(2\alpha)}$ projections the total error for our scheme tends to zero as $n \to \infty$. In the worst case $\alpha = 1$, and we can take $k = O\big(n^{1/2}\log n\big)$. At this extreme, the communication complexity of our system is $O\big(n^2\sqrt{\log n}\big)$, a factor of $\sqrt{n/\log n}$ larger than that of [11]. However, our system is more flexible for a number of reasons. First, we do not require that only a few sensors have meaningful data and even if this is the case, we do not need to know which sensors have meaningful data ahead of time in order to reap the benefits. Moreover, the scheme proposed in this paper accomplishes data compression and denoising in addition to efficient distribution and storage.

Compressed Sensing. Finally, we note that the idea of obtaining efficient signal representations via random projections has very recently received a great deal of attention in the signal processing community, beginning with the ground breaking papers [6,8,12]. Initial work in the area focused on noise-free random projections. More recent developments have considered noisy random projections [7, 15, 16]. In this paper we extend these results to the case where multiple noisy signals are compressed using the same random projection vectors. To our knowledge, this paper is the first to describe a practical implementation of random projections for compression and distribution in a multi-hop wireless sensor network. In particular, in our previous work [15, 16] we suggested the possibility of using random projections in wireless sensing, but in concert with non-collaborative analog communication schemes for transmitting the data to a predefined destination; a completely different scenario compared to the multi-hop system under consideration here. Theoretical issues related to joint sparsity of signals at a single sensor and across the network are investigated in [2].

2. COMPRESSION & PREDISTRIBUTION IN WIRELESS SENSOR NETWORKS

To motivate the proposed system, suppose our goal is to reconstruct the sensor data $\mathbf{x}$ at a specific point in the network. In a "pull-based" approach all nodes funnel their data to the location of the querying user. If nodes are arranged in a planar grid then the network radius is on the order of $\sqrt{n}$ hops. To gather all of the data at a single point in the network, each node's data value must be transmitted over a distance roughly as long as the network radius. Thus, the total communication complexity is $O(n\sqrt{n}) = O(n^{3/2})$ single-hop transmissions using an idealized scheme with no overhead for routing.

In many WSN applications it may not be necessary to collect the raw data from each node to obtain all of the relevant information. Data collected at nearby nodes in a dense sensor network is expected to be highly correlated [22] and we can take advantage of this compressibility to reduce the amount of communication. As an extreme example, consider the network health application described in the introduction where only $k \ll n$ sensors have "meaningful" measurements (the flag that they need repair) and that the remaining $n-k$ sensors measure zero. In this case the vector $\mathbf{x}$ itself is sparse (*i.e.*, no compressing transform is required). Only a few nonzero terms are needed to accurately reconstruct $\mathbf{x}$, and the communication complexity is reduced from $O(n^{3/2})$ single-hop transmissions (collect data from all nodes) to $O(k\sqrt{n})$ single-hop transmissions (collect only the relevant information). Clearly, when $k \ll n$, this scheme is much more energy-efficient than collecting all the sensor data.

More generally, the actual network data $\mathbf{x}$ might not be sparse, but it may still be compressible in a certain transform basis. We formalize this notion as follows.

Definition 1 (Best m-term Approximation). *Let $\mathbf{x} \in \mathbb{R}^n$ and let $\Psi = \{\boldsymbol{\psi}_i\}_{i=1}^n$ be an orthonormal basis for $\mathbb{R}^n$. Denote by $\theta_i = \boldsymbol{\psi}_i^T\mathbf{x}$ the coefficients of $\mathbf{x}$ in this new basis. Reordering the coefficients in decreasing magnitude so that*

$$|\theta_{(1)}| \geq |\theta_{(2)}| \geq \cdots \geq |\theta_{(n)}|,$$

the best m-term approximation of $\mathbf{x}$ in Ψ is given by $\mathbf{x}^{(m)} = \sum_{i=1}^m \theta_{(i)}\psi_{(i)}$. We say that $\mathbf{x}$ is compressible in Ψ when the mean squared approximation error behaves like

$$\frac{\|\mathbf{x} - \mathbf{x}^{(m)}\|^2}{n} = O(m^{-2\alpha}),$$

for some $\alpha \geq 1$. The parameter, α quantifies the compressibility of $\mathbf{x}$ in Ψ.

Suppose the network clairvoyantly knows that $\mathbf{x}$ is compressible in a basis Ψ and knows which m coefficients, $\theta_{(1)}, \ldots, \theta_{(m)}$, give the best m-term approximation to $\mathbf{x}$ in Ψ. Each coefficient is computed as an inner product of the form $\boldsymbol{\psi}_{(i)}^T\mathbf{x} = \sum_{j=1}^n \psi_{(i),j}\ x_j$. Suppose, in addition, that nodes have already constructed routes which form a spanning tree through the network, rooted at the querying user. To deliver one coefficient to a user, the nodes first locally compute the product $\psi_{(i),j}\ x_j$ and then aggregate these values up the tree. Each node must transmit a single value to its parent, thus there are n single-hop transmissions per coefficient, or mn total single-hop transmissions to collect the m best coefficients.

Remark 1. *Computing a general inner product of the form*

$$\mathbf{y}^T\mathbf{x} = \sum_{j=1}^n y_j x_j = \sum_{j=1}^n r_j, \tag{1}$$

where the components x_j correspond to values at different nodes in the network requires at least *n transmissions. To see this, observe that there are n terms in* (1)*, each located at a different node. The most efficient way to compute* (1) *in a network is as a nested sum $r_1+(r_2 + (\cdots + (r_{n-1} + r_n))\ldots)$, where each pair of terms is aggregated at the receiving node after a transmission, in which case there must be n single-hop transmissions. Thus, the optimal communication complexity for computing k transform coefficients is kn single-hop transmissions. Spanning trees, as described above, are one efficient means of routing for such a computation. Hamilton cycles (a cycle through the network passing through each node once) are another routing construction which admit optimal performance.*

Generally speaking, it is highly unlikely that the user will know ahead of time which m of the coefficients give the best m-term approximation (*i.e.*, the ordering $\theta_{(1)}, \ldots,$ will not be known), and the compressing basis, Ψ, may not be known either. In order to avoid this problem we compute projections of the data onto appropriately designed random vectors and then show that the random projection coefficients suffice to accurately reconstruct the original signal as long as it is compressible in some basis known to the user performing the reconstruction. In this sense our scheme is universal since the basis used for reconstruction does not have to be specified ahead of time.

If sensor data is noise-free and is compressible in some basis with parameter α then the reconstruction error behaves like $O\big((k/\log n)^{-2\alpha}\big)$. This is within a logarithmic factor of the error rate for the *best* k-term approximation. In fact, in the extreme case that the data is noise-free and $\mathbf{x}$ has no more than k non-zero coefficients in a compressing basis (*i.e.*, x is *sparse* in the transform domain), then one can obtain an *exact reconstruction* of x by calculating only $O(k \log n)$ summary values, without knowing which nodes hold the meaningful data ahead of time [8].

If sensors instead make noisy observations of a compressible vector $\mathbf{x}$, then the expected reconstruction error behaves like $O\big((k/\log n)^{\frac{-2\alpha}{2\alpha+1}}\big)$, within a logarithmic factor of the minimax estimation rate one could achieve with all of the data collected at a central location. If $\mathbf{x}$ is truly sparse with no more than m non-zero transform coefficients, then the reconstruction error behaves like $O\big((\frac{k}{m \log n})^{-1}\big)$. These results assume that the user knows the appropriate compressing basis to use for reconstruction (although the network itself does not need this information in the compression and distribution process)[2].

[2]Straightforward extensions of our approach are possible to handle situations in which the user can search over a library of bases to select the one which gives the best reconstruction, but we will not pursue this here.

3. SIGNAL RECONSTRUCTION FROM RANDOM PROJECTIONS

The previous sections examined a scenario where each sensor had a scalar measurement and these measurements were stacked in a vector $\mathbf{x}$. More generally, suppose each sensor has a block of L values so that the network data can be represented as L vectors, $\mathbf{x}_1^*, \ldots, \mathbf{x}_L^*$, with each $\mathbf{x}_i^* \in \mathbb{R}^n$. We focus on the situation where the individual vectors are each compressible in some basis, but two or more vectors may not be jointly compressible. For example, the different measurements at a node may be of different data types which are uncorrelated, the measurements may have been taken at different times spaced large enough to be effectively independent, or the local data may have already been compressed down to L values.

In what follows, we assume that the sensors do not have direct access to the signals of interest, $\mathbf{x}_1^*, \ldots, \mathbf{x}_L^*$. Rather, they make noisy measurements of the form (in vector notation), $\mathbf{z}_i = \mathbf{x}_i^* + \boldsymbol{\eta}_i$, where the components $\eta_{i,j}$ are i.i.d. zero-mean Gaussian random variables with variance σ^2. We assume that the $\eta_{i,j}$ are Gaussian to simplify the discussion. Our results can easily be extended for other distributions on the components of $\eta_{i,j}$. In particular, the same results hold if the components of $\boldsymbol{\eta}_i$ are bounded random variables. This is relevant in the context of wireless sensor networks since real sensors only take measurements in a bounded range. Note that we are assuming that noise terms $\eta_{i,j}$ and $\eta_{i,j'}$ entering at differing sensors $j \neq j'$ are independent. We justify this assumption by observing that noise will generally enter the system *at* each individual node, e.g., in the form of thermal sensor noise. Our setup is not unreasonable. Our results also have straightforward extensions to the noise-free setting, and can be derived in a similar fashion using existing results on compressed sensing [6, 8, 12], but since the noise-free case is unrealistic in most sensing applications we will not delve into the theory for this situation.

In the predistribution phase, nodes compute and distribute random projections of their data using this routing structure. Specifically, let $\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k\}$ be a collection of k length-n random vectors. The components $\phi_{i,j}$, $j = 1, \ldots, n$, of each $\boldsymbol{\phi}_i$ are i.i.d. random variables, independent of the $\{\boldsymbol{\eta}_i\}$, which take the values $\pm 1/\sqrt{n}$ with equal probability. Thus $\mathbb{E}[\phi_{i,j}] = 0$ and $\mathbb{E}[\phi_{i,j}^2] = 1/n$. The network computes noisy random projections of the form

$$y_{i,j} \;=\; \boldsymbol{\phi}_j^T \mathbf{z}_i \;=\; \boldsymbol{\phi}_j^T \mathbf{x}_i^* + w_{i,j},$$

for $i = 1, \ldots, L$, and $j = 1, \ldots, k$, where $\{w_{i,j}\}$ are i.i.d. zero-mean Gaussian random variables with variance $\mathbb{E}[w_{i,j}^2] = \sigma^2$, statistically independent of $\{\boldsymbol{\phi}_j\}$ [15].

For the time being, assume that there is either spanning tree or Hamilton cycle routing available to the network. We will relax all assumptions on available routes in the next section. Computing one projection requires n single-hop transmissions using the pre-established routes. For the cost of an additional n transmissions the newly computed projection value can be distributed to all of the nodes in the network, either by broadcasting it back down the spanning tree or by passing it around the cycle. This procedure is repeated $k \times L$ times to compute and distribute $\{y_{i,j}\}$, so the total communication complexity is $2kLn$ single-hop transmissions.

When the process just described is completed, each node has access to the entire collection $\{y_{i,j}\}$ of random projection values. If nodes are only equipped with a limited amount of storage then each node only need store a subset $y_i = \{y_{i,1}, \ldots, y_{i,L}\}$ of the values, for some i. Such a subset is of the same size as the original data at each node. Suppose the nodes are arranged in a Hamilton cycle as depicted in Figure 1. If node j stores the subset $j \mod k$ then the user can query any group of k nodes which are adjacent in the cycle to obtain the complete set of summary values, as illustrated in Fig. 1. We will revisit the issue of determining which nodes should store which subset of the values so that all subsets can be easily accessed from any location in a more practical setting later.

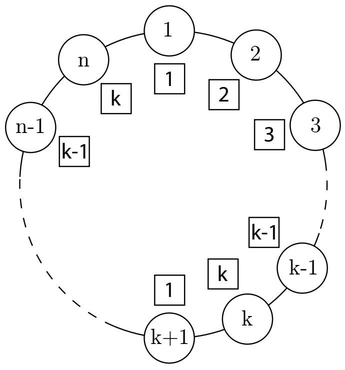

Figure 1: An example of n nodes arranged in a Hamilton cycle. The boxed value next to each node indicates which set of projection values $y_i = \{y_{i,j}\}_{j=1}^L$ stored at that node. Then a user can obtain a complete set of projection values from any group of k consecutive nodes in the cycle.

Now, suppose that the random projection values have been computed, distributed, and acquired by a user. We would like to choose reconstruction vectors $\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_L$, which have small expected squared error. Note that for a reconstruction $\widehat{\mathbf{x}}_i$,

$$\mathbb{E}\left[\frac{1}{k}\sum_{j=1}^{k}\left(y_{i,j} - \boldsymbol{\phi}_j^T \widehat{\mathbf{x}}_i\right)^2\right] = \frac{\|\mathbf{x}_i^* - \widehat{\mathbf{x}}_i\|^2}{n} + \sigma^2,$$

since the $\phi_{i,j}$ are i.i.d., with variance $\mathbb{E}[\phi_{i,j}^2] = \frac{1}{n}$. However, if we try to directly minimize the empirical squared error, $\frac{1}{k}\sum_{j=1}^{k}\left(y_{i,j} - \boldsymbol{\phi}_j^T \widehat{\mathbf{x}}_i\right)^2$, we run the risk of overfitting. To avoid this we use the complexity-regularization method introduced in [3].

In the complexity-regularization approach to non-parametric function estimation we focus on finding the best reconstruction from a certain class $\mathcal{X} \subseteq \mathbb{R}^n$ of candidate reconstructions. Assume that the original data vectors have bounded energy, $\|\mathbf{x}_i^*\|^2 \leq nB^2$. In addition, assume that $\mathbf{x}_i^*$ is compressible with respect to the basis Ψ. Any candidate vector $\mathbf{x} \in \mathcal{X}$ can be expanded in terms of Ψ by writing $\mathbf{x} = \sum_{i=1}^n \theta_i \boldsymbol{\psi}_i$ where the coefficients $\theta_i = \boldsymbol{\psi}_i^T \mathbf{x}$. Let $\Psi = [\boldsymbol{\psi}_1 \; \ldots \; \boldsymbol{\psi}_n]$ denote a matrix whose columns are $\boldsymbol{\psi}_i$. Then we have $x = \Psi\boldsymbol{\theta}$, where $\boldsymbol{\theta} = [\theta_1 \; \theta_2 \; \ldots \; \theta_n]^T$ is a columnized version of the coefficients. Let Θ denote the set of all coefficient vectors satisfying $\|\Psi\boldsymbol{\theta}\|^2 \leq nB^2$ and whose components are uniformly quantized in magnitude to n levels. We take our set of candidate reconstruction functions to be $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n \;:\; \mathbf{x} = \Psi\boldsymbol{\theta}, \boldsymbol{\theta} \in \Theta\}$. Moreover, for each

$\mathbf{x} = \Psi\boldsymbol{\theta} \in \mathcal{X}$, define the quantity

$$c(\mathbf{x}) \equiv c(\boldsymbol{\theta}) = 2\log(n)\|\boldsymbol{\theta}\|_0, \tag{2}$$

where $\|\boldsymbol{\theta}\|_0$ denotes ℓ_0 norm which is equal to the number of non-zero entries in $\boldsymbol{\theta}$. The value $c(\mathbf{x})$ can be thought of as a cost or penalty associated with choosing $\mathbf{x}$ as a reconstruction. With more non-zero coefficients, we have more degrees of freedom and are always able to improve the empirical squared error by using a more complex reconstruction. However, since we are assuming that $\mathbf{x}_i^*$ is compressible in Ψ we should only need a few coefficients to obtain an accurate approximation to $\mathbf{x}_i^*$. We will use the penalty terms $c(\mathbf{x})$ to encourage sparse reconstructions, hence the name "complexity-regularization". Specifically, given the measurements $\{y_{i,j}\}$ and the random projection vectors $\{\boldsymbol{\phi}_j\}$ the user chooses reconstructions by solving the optimization

$$\widehat{\boldsymbol{\theta}}_i^{(k)} = \arg\min_{\theta\in\Theta}\left\{\sum_{j=1}^{k}\left(y_{i,j} - \boldsymbol{\phi}_j^T\Psi\boldsymbol{\theta}\right)^2 + \lambda\,\|\boldsymbol{\theta}\|_0\right\}, \tag{3}$$

where $\lambda \equiv 2\log(2)\log(n)/\epsilon > 0$, and the reconstructed signal is given by $\widehat{\mathbf{x}}_i^{(k)} = \Psi\widehat{\boldsymbol{\theta}}_i^{(k)}$. The optimization above can be tackled with a variety of optimization methods, including those described in our previous work [15,16]. Also, one can convexify the optimization by replacing the ℓ_0 norm with the ℓ_1 norm, and simple projected gradient descent methods may be applied. A precise derivation of this reconstruction is given in the appendix where we also describe how to choose ϵ. In order to compute $\widehat{\mathbf{x}}_i^{(k)}$ the user needs the realizations of the random vectors $\boldsymbol{\phi}_j$. We describe a practical scheme for encoding and decoding this information in the following section.

We have the following bound on the reconstruction error given in terms of the number of random projections calculated and the compressibility of $\mathbf{x}_i^*$.

Theorem 1. *Assume that each $\mathbf{x}_i^*$ is compressible in the basis Ψ_i with parameter α_i. Let $\widehat{\mathbf{x}}_1^{(k)}, \ldots, \widehat{\mathbf{x}}_L^{(k)}$ be chosen according to* (3). *Then there exists a constant $C_1 > 0$ such that*

$$\max_{i=1,\ldots,L} \mathbb{E}\left[\frac{\|\mathbf{x}_i^* - \widehat{\mathbf{x}}_i^{(k)}\|^2}{n}\right] \le C_1 \max_{i=1,\ldots,L}\left(\frac{k}{\log(n)}\right)^{\frac{-2\alpha_i}{(2\alpha_i+1)}}.$$

Just as n^{-1} is the "usual" error rate for parametric estimation, $-2\alpha/(2\alpha+1)$ is the "usual" exponent governing the rate of convergence in nonparametric function estimation. Thus, we are within a logarithmic factor of the usual nonparametric rate. If the signals $\mathbf{x}_i^*$ are sparse in some basis (*i.e.*, $\theta_{(j)} = 0$ for $m < j \le n$) then we obtain stronger results.

Corollary 1. *Suppose that no $\mathbf{x}_i^*$ has more than m non-zero entries in some basis expansion. Let $\widehat{\mathbf{x}}_1^{(k)}, \ldots, \widehat{\mathbf{x}}_L^{(k)}$ be chosen according to* (3). *Then there exists a constant $C_2 > 0$ such that for all $i = 1, \ldots, L$*

$$\mathbb{E}\left[\frac{\|\mathbf{x}_i^* - \widehat{\mathbf{x}}_i^{(k)}\|^2}{n}\right] \le C_2\left(\frac{k}{m\log(n)}\right)^{-1}.$$

Proofs of both results appear in the appendix.

4. PRACTICAL DECENTRALIZED IMPLEMENTATION

4.1 Gossip Algorithms for Computation and Information Dissemination

In the previous section we assumed that there was a spanning tree or Hamilton cycle routing structure available in the network for computing and disseminating information. Although there are polynomial time decentralized algorithms for forming spanning trees and Hamilton cycles, maintaining either of these structures in unreliable or time-varying networks is a challenging task.

Completely decentralized algorithms for "consensus" or "agreement" computation in networked systems have recently received a great deal of attention [4,9,17,19,24,25]. This body of work includes Gossip algorithms and other variants. Algorithms of this nature are attractive for a number of reasons. Simple communication protocols are used to exchange information so there is no need to store or maintain complicated routing information. For the same reason, these algorithms scale well. They are resilient to changing network topology and unreliable transmission links. Also, since all nodes asymptotically compute the consensus value there is an added layer of robustness. The network cannot be compromised just by eliminating a fusion center. Though there is no overhead for complex routing schemes, the trade-off is that gossip algorithms may require more computational iterations than an approach using more complex routing.

A straightforward algorithm for gossip computation of the "average consensus" (average of the initial values at each node) is described in [4]. Let $u_l(t)$ denote the gossip value at node l after t iterations. At each iteration of this algorithm a node l_1 is activated uniformly at random. The active node chooses one of its neighbors l_2 uniformly at random. These two nodes exchange values and update $u_{l_1}(t+1) = u_{l_2}(t+1) = (u_{l_1}(t) + u_{l_2}(t))/2$. One can show that as $t \to \infty$, the values $u_l(t) \to \frac{1}{n}\sum_{j=1}^{n} u_j(0)$ for all nodes l. That is, all nodes arrive at a consensus value which is the average of the initial values throughout the network.

Using the gossip algorithm just described (or any other average consensus algorithm), we can simultaneously compute and distribute random projections, $\boldsymbol{\phi}_j^T\mathbf{z}_i$. Observe that, using the variable l to index nodes, we have $\boldsymbol{\phi}_j^T\mathbf{z}_i = \sum_{l=1}^{n}\phi_{j,l}\,z_{i,l}$. Suppose that node l has access to $\phi_{j,l}$ and the measurement value $z_{i,l}$. Then by initializing a consensus algorithm with $u_l(0) = n\phi_{j,l}z_{i,l}$ we have that $u_l(t) \to \boldsymbol{\phi}_j^T\mathbf{z}_i$ for all l as $t \to \infty$. Thus, all nodes in the network compute the value of the jth projection for data vector $\mathbf{z}_i$. By repeating this procedure for every projection vector $\boldsymbol{\phi}_j$ and data vector $\mathbf{z}_i$, each node obtains all kL projection values.

The iterates $u_l(t)$ only converge to the consensus value asymptotically as $t \to \infty$. Let $\bar{u}$ denote a vector with all entries equal to $\frac{1}{n}\sum_{l=1}^{n} u_l(0)$. In a practical system one will stop the algorithm after a finite number τ of cycles and incur an error $\|u(\tau) - \bar{u}\| = \varepsilon$. To keep this error from impairing the performance of our system we would like $\varepsilon^2 \approx \sigma^2$. This is equivalent to saying we would like the error to remain constant as we increase the network size; *i.e.*, we would like $\varepsilon = O(1)$.

To get a hold of the communication complexity associated with this algorithm we need to know how many iterations are required so that $\varepsilon = O(1)$. The rate of convergence of the gossip algorithm described above is related to the underlying network connectivity. Random geometric graphs have been widely adopted as a model for deployment and connectivity in wireless sensor networks [14, 20]. For this setup, Boyd et al. show in [4, 5] that the number of iterations which guarantees $\varepsilon = O(1)$ is $\tau = \Theta(n^2)$. To compute all kL random projection values requires $\Theta(kLn^2)$ single-hop communications since two nodes exchange values at each iteration. For this simple gossiping algorithm information is only exchanged between neighboring nodes, but the rate of convergence is slower than desired. In [10], Dimakis et al. describe and analyze *geographic gossip*, a variation on the algorithm described above where nodes additionally exchange gossip values with distant nodes in the network via greedy geographic routing. The only added requirement is that nodes know their geographic location. They prove that with high probability, geographic gossip requires $O\big(n^{3/2}/\sqrt{\log n}\big)$ single-hop transmissions to compute one random projection with $\varepsilon = O(1)$, yielding an overall communication complexity of $O\big(kLn^{3/2}/\sqrt{\log n}\big)$. Thus, geographic gossip provides a practical means of computing projection coefficients at roughly a factor of $n^{1/2}$ additional communication complexity over a scheme based on specialized routes.

We have only described gossip and consensus algorithms in a very idealized setting. A variety of other algorithms for computing the average consensus have been proposed which account for changing topologies, unreliable links, and other network uncertainties [19, 23, 28].

4.2 Generating Random Projection Vectors $\boldsymbol{\phi}_j$

In order to calculate reconstructions $\widehat{\mathbf{x}}_i^{(k)}$ according to (3) the user needs to know the realizations of the random vectors $\{\boldsymbol{\phi}_j\}$ used to compute the projection values in addition to the projection values $\{y_{i,j}\}$ themselves. This information can be efficiently generated using the seed of a pseudo-random number generator and the addresses of the nodes in order to draw the values $\{\phi_{i,j}\}$. Assume that each node has been assigned a unique address $i = 1, \ldots, n$. Denote by $R(s,i) : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$ a pseudo-random number generating function which takes as arguments a seed, s, and the index of a pseudo-random number in sequence. We assume that all nodes implement the same pseudo-random number generator, which is also available to the user.

Each random projection vector $\boldsymbol{\phi}_j$ is associated with a different seed s_j, and we set $\phi_{j,i} = R(s_j, i)$. Then, in order to have vectors $\{\boldsymbol{\phi}_j\}$ which are statistically independent, the seeds s_j just need to be chosen so that there is no overlap between the sequences $\{R(s_j,i)\}_{i=1}^n$ and $\{R(s_{j'},i)\}_{i=1}^n$. Enforcing this condition is specific to the pseudo-random number generator implementation. In the computation and predistribution phase, each node simply needs the seed values $\{s_j\}$ and its own address to generate the values $\phi_{j,i}$ required to compute the random projections. Similarly, the user can easily reconstruct the vectors $\{\boldsymbol{\phi}_j\}$ given the seed values and the number of nodes in the network.

4.3 Distributed Storage of Random Projection Values

When the predistribution phase (gossip computation) is completed, every node has access to all kL random projection values. If the nodes have sufficient storage they can store each of the values and supply them directly to a querying user. More likely, however, nodes will have a limited amount of storage and it may not be possible for each node to store all kL random projection values. In this case we would like to employ a storage scheme so that a user can acquire the entire collection of projection values by querying as few nodes as possible.

A simple way to deal with this issue is for each node to store one set of projection values $y_j = \{y_{i,j}\}_{i=1}^L$ which is the same size as its original set of data. Suppose each node chooses one of the y_j at random (independently and uniformly). This approach corresponds to the "uncoded random storage" scheme described in [1] where they show that a user need only to query on the order of no fewer than $k \log k$ nodes in order to obtain the entire collection of projection values with high probability.

One can do better than this by using random linear coding techniques from the network coding literature. Rather than having nodes choose one of the y_j to store at random, we will have each node store a linear combination of the projection values. First, note that in any practical implementation the nodes must store quantized representations of each $y_{i,j}$ rather than actual real-valued numbers. For concreteness, assume that our implementation uses 8-bit fixed point structures to represent each $y_{i,j}$ (the exact same idea works for 16 or 32-bit fixed or floating point as well). Denoting by $\mathbb{F}_{256}$ the finite field of bytes, $y_{i,j} \in \mathbb{F}_{256}$. Rather than storing the entire collection or one of the y_j, each node $i = 1, \ldots, n$ stores values $\gamma_{i,l} = \sum_{j=1}^k \alpha_{i,j} y_{l,j}$, for $l = 1, \ldots, L$, where $\alpha_{i,j}$ are i.i.d. random variables drawn uniformly from $\mathbb{F}_{256}$ and all arithmetic is with respect to $\mathbb{F}_{256}$. Similar to the approach described in the previous section for generating random vectors $\boldsymbol{\phi}_j$, the weights $\alpha_{i,j}$ can easily be generated using a unique seed and the index i of the node. In this setup, each node stores the L values $\gamma_i = \{\gamma_{i,l}\}_{l=1}^L$ and the seed which was used to generate the weights $\alpha_{i,j}$. The nodes also return these values when queried. Then, it can be shown that with high probability (at least $1 - 1/k$) the user can perfectly recover the entire collection $\{y_{i,j}\}$ after querying $O(k)$ nodes. See, e.g., Proposition 5.2 in [1]. Recovering the $y_{i,j}$ from k sets of values $\{\gamma_{i_l}\}_{l=1}^L$ and the seeds used to generate the $\alpha_{i,j}$ effectively amounts to inverting a matrix. Thus, for a small additional overhead (storing and transmitting one more seed per node) we have a practical system where, with high probability, the user can reconstruct an accurate approximation of the entire network data. Of course, if there are available storage resources then nodes can calculate and store more sets of combinations $\gamma_{i,l,1}, \gamma_{i,l,2}, \ldots$ by using two or more statistically independent sets $\{\alpha_{i,j,1}\}, \{\alpha_{i,j,2}\}, \ldots$. This will further reduce the number of nodes a user must query to obtain the entire collection of projection values.

5. EXAMPLE APPLICATIONS

To illustrate the theory and method developed above, we consider two potential applications, network diagnostics and field reconstruction. In both experiments we solve the convexified version of (3), with the ℓ_0 norm replaced by the ℓ_1 norm. The resulting objective function can be expressed as a quadratic program with linear inequality constraints, for which the gradient projection strategy is known to be quite effective (see Chapter 16.6 in [18]) and we use it here for our purposes.

5.1 Network Diagnostics

Experience with real life sensor networks suggests that one should be prepared for unreliable behavior from sen-

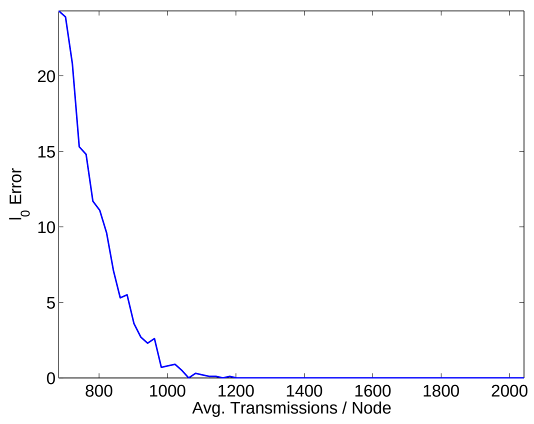

Figure 2: Error in network health monitoring as a function of the number of Gossip algorithm transmissions per node.

sor nodes [27]. This section illustrates how our compression/presdistribution system can be used to solve the problem of identifying a set of nodes which have failed. We assume that only a small number, $m \ll n$, of nodes have failed, and that the rest are functioning properly. Nodes compute and store random projections of a signal x^* where $x_i^* = 0$ if node i it is functioning, and $x_i^* = 1$ if it has failed. Such a signal can easily be constructed locally and since this data does not involve measurements of the environment there is no additive noise.

We have simulated this situation on networks of $n = 4096$ nodes placed uniformly on a square grid (so that each node has at most four neighbors), where $m = 41$ nodes at random locations have failed. Our theory suggests that $O(m \log n)$ random projections should suffice to recover the locations of the failed nodes, and in our simulation we set $k = 682 \approx 2m \log n$. Random projection coefficients are computed using a gossip algorithm where we vary the average number of transmissions per node to assess the effect of this parameter on performance. Figure 2 displays the ℓ_0 error (the number of nodes mis-identified as either needing or not needing repair) plotted against the average number of transmissions per node. Each data point corresponds to the average over 10 different network realizations.

5.2 Reconstructing a Piece-wise Smooth Field

Field estimation is another envisioned application of wireless sensor networks. Consider a network sensing a field composed of two regions of distinct behavior. An example of such a field is depicted in Fig. 3. In this application, we assume that the sensors make noisy observations of this field and the goal is to distribute information throughout the network so that a user can query an arbitrary subset of convenient nodes and reconstruct an accurate estimate of the field. We simulate this situation by adding a small amount of Gaussian noise to the field shown in Fig. 3(a), resulting in our sensor measurements. A typical realization of the sensor measurements is depicted in Fig. 3(b). There are $n = 128 \times 128$ sensors in this simulation. A typical reconstruction in the wavelet basis using $k = 1000$ random projections is shown in Fig. 3(c). Fig. 4 depicts the mean squared reconstruction error as a function of the number of random projections collected using 1200 gossip transmissions per node. Each data point corresponds to the average over 10 data realizations.

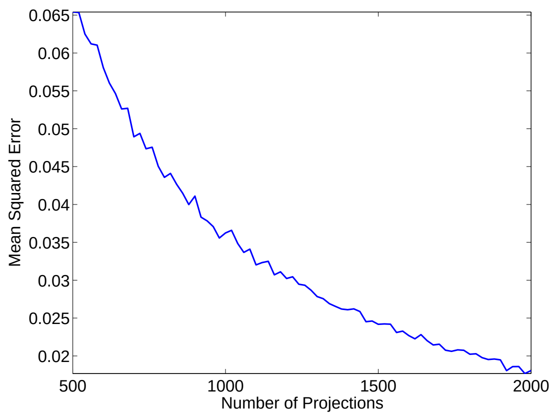

Figure 4: Mean squared reconstruction error versus k, the number of random projections collected for estimating a 128×128 piece-wise smooth field.

6. DISCUSSION AND EXTENSIONS

We have described a practical decentralized approach to simultaneously distilling the data collected by all sensors into a small number of summary values and distributing the summary values throughout the network. The end result is a distributed compression and storage system for sensor networks whereby a user can walk up to any point in the network and obtain sufficient information to reconstruct an accurate approximation of the entire network data after querying a small number of nodes. Each summary value – the projection of the network data onto a random vector – can be computed efficiently in a completely decentralized fashion using gossip algorithms.

If the original data is compressible in some basis such that the error of the best m-term approximation in that basis behaves like $O\big(m^{-2\alpha}\big)$ for some $\alpha \geq 1$, then the squared error of a reconstruction using k summary values behaves like $O\big((k/\log n)^{-2\alpha/(2\alpha+1)}\big)$. If the data is sparse so that only m sensors have significant data values then the squared error of a reconstruction using k summary values behaves like $O\big((\frac{k}{m \log n})^{-1}\big)$.

An interesting extension of our system involves progressive refinement of the approximation. Suppose the user begins with a reconstruction using k summary values. If the accuracy of this reconstruction is not satisfactory the user could send a query out into the network requesting that additional summary values be computed so as to improve the system performance.

APPENDIX

The results derived in this section are extensions of theory developed by Haupt and Nowak in [15] where the case of $L = 1$ is studied. Rather than reworking the proofs in detail we reference [15] where appropriate and emphasize the steps which need to be taken to extend those results to the ones presented in this paper.

Our goal is to obtain accurate reconstructions $\widehat{\mathbf{x}}_i$ of the functions $\mathbf{x}_i^*$ using the random projection values $\{y_{i,j}\}$ for $i = 1, \ldots, L$ and $j = 1, \ldots, k$, as defined in Section 3. Define the expected mean squared error, or *risk*, of a candidate

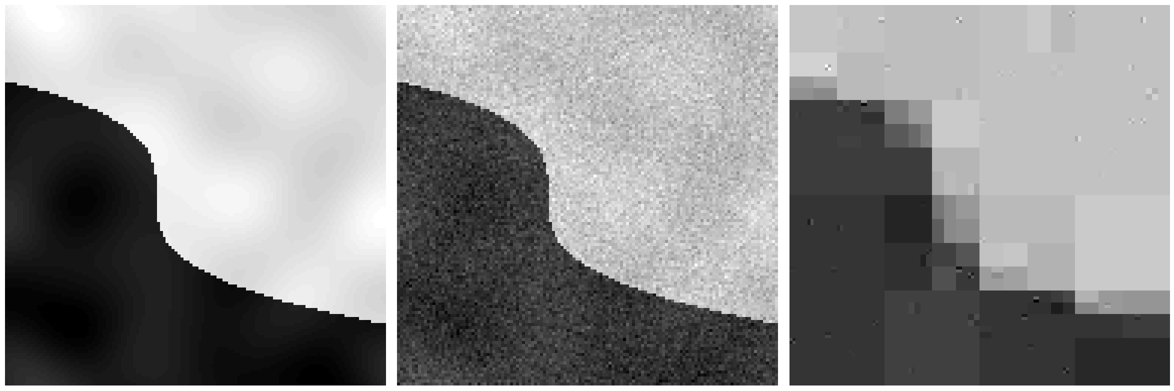

Figure 3: Reconstruction of a 128×128 piecewise smooth field. (a) Noise-free field. (b) Noisy sensor readings. (c) Typical reconstruction based on 1000 random projections (1/16 of the original data size).

reconstruction $\mathbf{x}$ for the original signal $\mathbf{x}_i^*$ to be

$$R_i(\mathbf{x}) = \mathbb{E}\left[\frac{1}{k}\sum_{j=1}^{k}\left(y_{i,j} - \boldsymbol{\phi}_j^T\mathbf{x}\right)^2\right].$$

Note that $R_i(\mathbf{x}) \equiv \|\mathbf{x}_i^* - \mathbf{x}\|^2/n + \sigma^2$ since $\{\phi_{i,j}\}$ and $\{w_{i,j}\}$ are independent and $\mathbb{E}[\phi_{i,j}^2] = 1/n$. Without knowing $\mathbf{x}_i^*$ there is no way to compute $R_i(\mathbf{x})$. However, given the vectors $\{\boldsymbol{\phi}_j\}$ and random projection values $\{y_{i,j}\}$ we can compute the empirical risk associated with a candidate reconstruction, $\widehat{R}_i(\mathbf{x}) = \frac{1}{k}\sum_{j=1}^{k}\left(y_{i,j} - \boldsymbol{\phi}_j^T\mathbf{x}\right)^2$.

The proofs of Theorem 1 and Corollary 1 from Section 3 rely on the following error bound which holds for all data vectors $\mathbf{x}_i^*$ regardless of whether they are compressible or sparse.

Theorem 2. *Let $\mathcal{X}$ be a countable set of candidate reconstruction functions such that $\|\mathbf{x}\|^2 \leq nB^2$ for every $\mathbf{x} \in \mathcal{X}$. Assign to each $\mathbf{x} \in \mathcal{X}$ a penalty $c(\mathbf{x})$ such that*

$$\sum_{\mathbf{x}\in\mathcal{X}} 2^{-c(\mathbf{x})} \leq 1, \tag{4}$$

and choose $\widehat{\mathbf{x}}_1^{(k)}, \ldots, \widehat{\mathbf{x}}_L^{(k)}$ by solving

$$\widehat{\mathbf{x}}_i^{(k)} = \arg\min_{\mathbf{x}\in\mathcal{X}}\left\{\widehat{R}_i(\mathbf{x}) + \frac{c(\mathbf{x})\log(2)}{k\epsilon}\right\}, \tag{5}$$

where $\epsilon > 0$ is specified below. Then there exists a constant $C > 0$ such that

$$\max_{i\in\{1,\ldots,L\}} \mathbb{E}\left[\frac{\|\mathbf{x}_i^* - \widehat{\mathbf{x}}_i^{(k)}\|^2}{n}\right] \leq$$

$$C\max_{i\in\{1,\ldots,L\}}\min_{\mathbf{x}\in\mathcal{X}}\left\{\frac{\|\mathbf{x}_i^* - \mathbf{x}\|^2}{n} + \frac{c(\mathbf{x})\log(2) + \log(L) + 4}{k\epsilon}\right\}.$$

Proof: Define the "excess risk" between a candidate reconstruction $\mathbf{x}$ and the actual data $\mathbf{x}_i^*$ to be $r_i(\mathbf{x}) = R_i(\mathbf{x}) - R_i(\mathbf{x}^*)$. To prove Theorem 2 and justify the reconstruction procedure (5) we utilize a slight modification of a bound on the excess risk for individual vectors $\mathbf{x}_i^*$ derived in [15]. Then we will uniformly bound the excess risks over all $\mathbf{x}_1^*, \ldots, \mathbf{x}_L^*$ to obtain the desired result.

Similar to above, define the "excess empirical risk" to be $\widehat{r}_i(\mathbf{x}) = \widehat{R}_i(\mathbf{x}) - \widehat{R}_i(\mathbf{x}_i^*)$. Using the techniques developed in Section III of [15] (c.f., equation (23)) one can show that for a particular vector $\mathbf{x}_i^*$ and a particular $\mathbf{x} \in \mathcal{X}$, for all $\delta_{\mathbf{x}} > 0$, with probability at most $\delta_{\mathbf{x}}$,

$$r_i(\mathbf{x}) - \widehat{r}_i(\mathbf{x}) \geq \frac{\log(1/\delta_{\mathbf{x}})}{k\epsilon} + \frac{\epsilon\ (8B^2 + 4\sigma^2)\ r_i(\mathbf{x})}{2(1-\zeta)},$$

where ϵ and ζ must be chosen such that $0 < \epsilon h \leq \zeta < 1$ and $h = 64\sqrt{2}B^2 + 64B\sigma$. For each $\mathbf{x} \in \mathcal{X}$ we have a penalty $c(\mathbf{x})$ satisfying (4). Let $\delta > 0$ and set $\delta_{\mathbf{x}} = 2^{-c(\mathbf{x})}\delta L^{-1}$. Applying the union of events bound over all $\mathbf{x} \in \mathcal{X}$ we have that for a particular $i \in \{1, \ldots, L\}$ and for all $\delta > 0$ the event that

$$\begin{aligned} r_i(\mathbf{x}) - \widehat{r}_i(\mathbf{x}) \quad \geq \quad & \frac{c(\mathbf{x})\log(2) + \log(L) + \log(1/\delta)}{k\epsilon} \\ & + \frac{\epsilon(8B^2 + 4\sigma^2)r_i(\mathbf{x})}{2(1-\zeta)} \end{aligned} \tag{6}$$

for any $\mathbf{x} \in \mathcal{X}$ occurs with probability at most δL^{-1}. Set $\zeta = \epsilon h = \epsilon(64\sqrt{2}B^2 + 64B\sigma)$ and choose

$$\epsilon < \frac{1}{(64\sqrt{2}+4)B^2 + 64\sigma B + 2\sigma^2},$$

which guarantees that $\zeta < 1$. Define $a = \frac{\epsilon(8B^2+4\sigma^2)}{2(1-\zeta)}$. Observe that for a random variable X, if $P(X \geq t) \leq p$ then $P(X < t) > 1 - p$. Rearranging the terms in (6), we now have that for a particular $i \in \{1, \ldots, L\}$, for all $\delta > 0$ and for any $\mathbf{x} \in \mathcal{X}$, with probability at least $1 - \delta L^{-1}$,

$$\begin{aligned} (1-a)r_i(\mathbf{x}) \quad \leq \quad & \widehat{R}_i(\mathbf{x}) - \widehat{R}_i(\mathbf{x}_i^*) \\ & + \frac{c(\mathbf{x})\log(2) + \log(L) + \log(1/\delta)}{k\epsilon}. \end{aligned}$$

To minimize the upper bound we take

$$\widehat{\mathbf{x}}_i^{(k)} = \arg\min_{\mathbf{x}\in\mathcal{X}}\left\{\widehat{R}_i(\mathbf{x}) + \frac{c(\mathbf{x})\log(2)}{k\epsilon}\right\}.$$

Replacing the empirical risk in the above expression with the risk, define

$$\widetilde{\mathbf{x}}_i^{(k)} = \arg\min_{\mathbf{x}\in\mathcal{X}}\left\{R_i(\mathbf{x}) + \frac{c(\mathbf{x})\log(2)}{k\epsilon}\right\}.$$

Then, from the definition of $\widehat{\mathbf{x}}_i^{(k)}$, we have for a particular $i \in \{1, \ldots, L\}$ and for all $\delta > 0$, with probability at most

δL^{-1}

$$(1-a)r_i(\widehat{\mathbf{x}}_i^{(k)}) \geq \widehat{r}_i(\widetilde{\mathbf{x}}_i^{(k)}) + \frac{c(\widetilde{\mathbf{x}}_i^{(k)})\log(2) + \log(L/\delta)}{k\epsilon}. \quad (7)$$

It is possible to obtain a similar bound on $\widehat{r}_i(\widetilde{\mathbf{x}}_i^{(k)}) - r_i(\widetilde{\mathbf{x}}_i^{(k)})$ so that for all $\delta > 0$, with probability at most δL^{-1}

$$\widehat{r}_i(\widetilde{\mathbf{x}}_i^{(k)}) - r_i(\widetilde{\mathbf{x}}_i^{(k)}) \geq a r_i(\widetilde{\mathbf{x}}_i^{(k)}) + \frac{\log(L) + \log(1/\delta)}{k\epsilon}. \quad (8)$$

See equation (33) in [15].

In order to have (7) and (8) hold simultaneously we apply the union bound. Then, for a particular $i \in \{1, \ldots, L\}$, the event that

$$r_i(\widehat{\mathbf{x}}_i^{(k)}) \quad \geq \quad \left(\frac{1+a}{1-a}\right) r_i(\widetilde{\mathbf{x}}_i^{(k)}) + \frac{c(\widetilde{\mathbf{x}}_i^{(k)})\log(2) + 2\log(L/\delta)}{k\epsilon(1-a)}$$

occurs with probability at most $2\delta L^{-1}$. Set $\delta = e^{-k\epsilon t(1-a)/2}$, with $t \in (0, \infty)$. Applying the union of events bound once more over all $i = 1, \ldots, L$ we have that

$$P\left(\max_{i\in\{1,\ldots,L\}} \left\{ r_i(\widehat{\mathbf{x}}_i^{(k)}) - \left(\frac{1+a}{1-a}\right) r_i(\widetilde{\mathbf{x}}_i^{(k)}) - \frac{c(\widetilde{\mathbf{x}}_i^{(k)})\log(2) + 2\log(L)}{k\epsilon(1-a)} \right\} \geq t \right) \leq 2e^{\frac{-k\epsilon t(1-a)}{2}}. \quad (9)$$

Note that for any real-valued random variable X, $\mathbb{E}[X] \leq \int_0^\infty P(X \geq t)dt$. Integrating (9) yields

$$\mathbb{E}\left[\max_{i\in\{1,\ldots,L\}} \left\{ r_i(\widehat{\mathbf{x}}_i^{(k)}) - \left(\frac{1+a}{1-a}\right) r_i(\tilde{\mathbf{x}}_i^{(k)}) - \frac{c(\tilde{\mathbf{x}}_i^{(k)})\log(2) + 2\log(L)}{k\epsilon(1-a)} \right\}\right] \leq \frac{4}{k\epsilon(1-a)}.$$

For scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$, the relation $\max_i(\alpha_i - \beta_i) \geq \max_i \alpha_i - \max_i \beta_i$ holds. Then, taking the max outside the expectation, we can write

$$\max_{i\in\{1,\ldots,L\}} \mathbb{E}\left[r_i(\widehat{\mathbf{x}}_i^{(k)})\right] \quad \leq \quad \max_{i\in\{1,\ldots,L\}} \left\{ \left(\frac{1+a}{1-a}\right) r_i(\tilde{\mathbf{x}}_i^{(k)}) + \frac{c(\tilde{\mathbf{x}}_i^{(k)})\log(2) + 2\log(L) + 4}{k\epsilon(1-a)} \right\}.$$

It is easily verified (see (13) and (26) in [15]) that $r_i(\mathbf{x}) = \|\mathbf{x}_i^* - \mathbf{x}\|^2/n$ and $a > 0$. Then after a few manipulations we finally obtain

$$\begin{aligned} &\max_{i\in\{1,\ldots,L\}} \mathbb{E}\left[\frac{\|\mathbf{x}_i^* - \widehat{\mathbf{x}}_i^{(k)}\|^2}{n}\right] \\ &\leq \left(\tfrac{1+a}{1-a}\right) \max_{i\in\{1,\ldots,L\}} \left\{ r_i(\tilde{\mathbf{x}}_i^{(k)}) + \tfrac{c(\tilde{\mathbf{x}}_i^{(k)})\log(2) + 2\log(L) + 4}{k\epsilon} \right\} \\ &\leq C \max_{i\in\{1,\ldots,L\}} \min_{\mathbf{x}\in\mathcal{X}} \left\{ \tfrac{\|\mathbf{x}_i^* - \mathbf{x}\|^2}{n} + \tfrac{c(\mathbf{x})\log(2) + 2\log(L) + 4}{k\epsilon} \right\}, \end{aligned}$$

with $C = (1+a)/(1-a)$. $\square$

The proofs of Theorem 1 and Corollary 1 now follow identically to the proofs of Theorem 2 and Corollary 1 in [15], using the result just derived in place of the oracle inequality derived in [15].

A. REFERENCES

[1] S. Acedański, S. Deb, M. Médard, and R. Koetter. How good is random linear coding based distributed networked storage? In *Proc. Netcod*, Italy, April 2005.

[2] D. Baron, M. Duarte, S. Sarvotham, M. Wakin, and R. Baraniuk. An information-theoretic approach to distributed compressed sensing. In *Allerton Conf. on Comm., Control, and Computing*, Sept. 2005.

[3] A. R. Barron. Complexity regularization with application to artificial neural networks. *Nonparametric Functional Estimation and Related Topics*, pages 561–576, 1991.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proc. IEEE Infocom*, Miami, FL, March 2005.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Mixing times for random walks on geometric random graphs. In *Proc. SIAM Workshop on Analytic Algorithms and Combinatorics*, Vancouver, B.C., Canada, January 2005.

[6] E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. Preprint, June, 2004.

[7] E. Candès and T. Tao. The dantzig selector: statistical estimation when p is much larger than n. Preprint. May, 2005.

[8] E. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? Preprint, October, 2004.

[9] J. Chen, G. Pandurangan, and D. Xu. Robust aggregates computation in wireless sensor networks: Distributed algorithms and analysis. In *Proc. IPSN*, Los Angeles, CA, April 2005.

[10] A. Dimakis, A. Sarwate, and M. Wainwright. Geographic gossip: Efficient aggregation for sensor networks. In *Proc. IPSN*, Nashville, TN, April 2006.

[11] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In *Proc. IPSN*, Los Angeles, CA, April 2005.

[12] D. L. Donoho. Compressed sensing. Preprint, September, 2004.

[13] M. Gastpar, P. L. Dragotti, and M. Vetterli. The distributed Karhunen-Loeve transform. Submitted to IEEE Trans. on Info. Theory, Nov., 2004.

[14] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46:388–404, 2000.

[15] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. Submitted to IEEE Trans. on Info. Theory, March, 2005.

[16] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections with applications to wireless sensing. In *Proc. IEEE Wkshp. on Stat. Sig. Proc.*, Bordeaux, France, July 2005.

[17] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. IEEE Conf. on FOCS*, 2003.

[18] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999.

[19] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Auto. Control*, 49(9):1520–1533, Sept. 2004.

[20] M. D. Penrose. *Rand. Geometric Graphs.* Oxford Press, 2003.

[21] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[22] S. S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, 19(2):51–60, March 2002.

[23] M. G. Rabbat, R. D. Nowak, and J. A. Bucklew. Generalized consensus algorithms in networked systems with erasure links. In *Proc. IEEE Wkshp. on Signal Proc. Advances in Wireless Comm.*, New York, NY, June 2005.

[24] V. Saligrama, M. Alanyali, and O. Savas. Distributed detection in sensor networks with packet losses and finite capacity links. *IEEE Trans. on Sig. Proc.*, to appear, 2006.

[25] D. Scherber and B. Papadopoulos. Locally constructed algorithms for distributed computations in ad-hoc networks. In *Proc. IPSN*, Berkeley, CA, April 2004.

[26] S. D. Servetto. On the feasibility of large-scale wireless sensor networks. *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[27] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proc. First European Workshop on Wireless Sensor Networks*, Berlin, Germany, January 2004.

[28] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proc. IPSN*, Los Angeles, CA, April 2005.