Networked Computing in Wireless Sensor Networks for Structural Health Monitoring

Apoorva Jindal, *Member, IEEE,* Mingyan Liu, *Senior Member, IEEE,*

***Abstract*—This paper studies the problem of distributed computation over a network of wireless sensors. While this problem applies to many emerging applications, to keep our discussion concrete, we will focus on sensor networks used for structural health monitoring. Within this context, the heaviest computation is to determine the singular value decomposition (SVD) to extract mode shapes (eigenvectors) of a structure. Compared to collecting raw vibration data and performing SVD at a central location, computing SVD within the network can result in significantly lower energy consumption and delay. Using recent results on decomposing SVD, a well-known centralized operation, we seek to determine a near-optimal communication structure that enables the distribution of this computation and the reassembly of the final results, with the objective of minimizing energy consumption subject to a computational delay constraint. We show that this reduces to a generalized clustering problem, and establish that it is NP-hard. By relaxing the delay constraint, we derive a lower bound. We then propose an integer linear program (ILP) to solve the constrained problem exactly as well as an approximate algorithm with a proven approximation ratio. We further present a distributed version of the approximate algorithm. We present both simulation and experimentation results to demonstrate the effectiveness of these algorithms.**

***Index Terms*—Networked Computing, Wireless Sensor Networks, Structural Health Monitoring, Clustering, Degree-Constrained Data Collection Tree, Singular Value Decomposition.**

I. Introduction

Over the past decade, tremendous progress has been made in understanding and using wireless sensor networks. Of particular relevance to this paper are extensive studies on in-network processing, e.g., finding efficient routing strategies when data compression and aggregation are involved. However, many emerging applications, e.g., body area sensing, structural health monitoring, and various other cyber-physical systems, require far more sophisticated data processing in order to enable real-time diagnosis and control.

This leads to the question of how to perform arbitrary (and likely complex) computational tasks using a distributed network of wireless sensors, each with limited resources both in energy and in processing capability. Previous results on establishing the communication structure for in-network computation mostly consider relatively simple functions like max, min, averages and sums, see e.g., [1]–[7] that cannot fully represent the complex computational requirements demanded by many practical engineering applications. Results on more general functions, e.g., the family of *symmetric functions* studied in [8]–[10], are often in the form of delivery equivalent representation of the data rather than a communication structure that would allow the in-network computation of the function. (A more extensive discussion on related work is presented in Section VII.)

This study is motivated by critical in-network computational needs that arise in structural health monitoring (SHM). SHM is a rapidly growing application area for wireless sensor network technologies [11]–[13] and cyber-physical system approaches [13] because of (i) the increasing need to provide low-cost and timely monitoring and inspection of the deteriorating national infrastructure, and (ii) the advances in integrated wireless sensing technologies.

Within this context, the most common approach to monitor fatigue or detect damage is to collect vibration data using a set of wireless sensors in response to white/free input to the structure and then compute the FFT of each individual sensed stream; this is then followed by the procedure of singular value decomposition (SVD) on these FFTs to determine a set of modes [14]–[16]. A *mode* is a combination of a frequency and a shape (in the form of a vector); the mode shape describes the expected curvature (or displacement) of a surface vibrating at the corresponding modal frequency. The mode shapes convey useful information as to whether the structure is behaving normally, and can thus be used to detect damage.

To obtain these modes, a straightforward way is to have all sensor nodes transmit raw vibration data in the form of time series of certain size, or the Fast Fourier Transform (FFT) of the raw data, to a central controller or the base station. This collection can be done either via single hop communication if the sensor network has a star topology surrounding the base station, or via multi-hop communication if the network spans a large area (e.g., along a highway bridge). This form of data collection can be very expensive: a single 4000-point FFT can translate into 8K bytes of data [17]. With 10s or 100s of sensors monitoring a large civil infrastructure, this data collection methodology can be a huge burden on the battery power of the wireless sensors. Instead, if we perform the SVD computation within the network, then we can potentially achieve a significant reduction in the amount of data that needs to be transmitted because the output of the SVD computation is a set of vectors much smaller in size compared to the FFTs (on the order of 10s of bytes [17]). It is therefore highly desirable to be able to perform the SVD computation within the network.

A. Jindal is a Member of Technical Staff at Juniper Networks and M. Liu is an Associate Professor with the Department of Electrical Engineering and Computer Science at University of Michigan, Ann Arbor. E-mail: ajindal@ujuniper.net,mingyan@eecs.umich.edu.

The work is partially supported by the U.S. Department of Commerce, National Institute of Standards and Technology (NIST) Technology Innovation Program (TIP) under Cooperative Agreement Number 70NANB9H9008.

There are a number of challenges in performing a complex matrix operation like SVD, a classical centralized procedure, in a distributed way over a network of resource-constrained sensors. A single sensor may be limited in its processing capability which in turn limits the size of the matrix operation it can handle, either due to lack of memory or excessive delay. In addition, the input FFTs originate from sensors at different locations, which makes it critical to find the right combination of a computation mechanism and a communication structure.

In this paper we present a method of obtaining the optimal communication structure for the distributed computation of SVD, by utilizing a functional decomposition of SVD recently developed by Zimmerman *et al.* [17]. Here optimality is defined with respect to minimizing energy consumption subject to a computational delay constraint. We show that this method seeks to construct an optimal data forwarding and computing structure, which turns out to be a certain type of tree. The construction of this tree structure may also be viewed as a generalized form of clustering as we detail in Section III. It is worth mentioning that SVD is an essential computational need in a very broad class of signal processing algorithms, including classification, identification and detection [17]–[22]. So the results obtained here can potentially be applied to many other signal processing applications beyond structural health monitoring. In this paper, to make our discussion concrete, we will focus on the SHM application. This allows us to select system parameters, constraints and network topologies based on real systems.

While the analysis and results presented here are framed within the context of SVD, they illustrate a general approach to the distributed in-network computation of complex objectives. Under this approach, a centralized operation is first decomposed into a number of computational elements (or operators) each operating on a set of inputs. This is then followed by an optimization procedure to determine on which nodes these elements should be placed (computed), which in turn determines to whom a node should send its data input. This is the main novelty of our approach. By using this method and the unique features of SVD computation we can dramatically reduce the amount of data transmission required. The main results and contributions of this paper are summarized as follows.

1) We formally define the above networked computing problem for SVD for the objective of minimizing energy consumption subject to a delay constraint, and establish that it is NP-hard.
2) We derive a lower bound on the energy consumption by relaxing the delay constraint and show that the optimal communication and computation for the unconstrained problem has a simple structure: the communication occurs along a shortest-path tree, where each non-leaf node performs a local SVD operation.
3) We develop an integer linear program (ILP) to solve the constrained problem, and introduce an approximate algorithm, along with its distributed implementation, with a proven approximation ratio. We also discuss modifications to our algorithms in the presence of noisy sensor measurements.
4) We use both simulations and testbed experiments to evaluate our algorithms and compare our results to a routing algorithm which does not use in-network computation as well as to a randomly generated communication structure.

It is also worth mentioning that in the process of deriving the approximation algorithm, we also propose a solution to the degree-constrained shortest path routing problem which is an important problem in its own right and has other applications in wireless sensor systems, like building a data collection tree to minimize interference when multiple channels are available for scheduling [23]. Results for this problem are known only for complete graphs whose weights satisfy the triangle inequality [24], [25]. Hence, our result here is the first to propose approximation algorithms and analytically derive their approximation factors for this problem in graphs induced by a communication network.

We end this introduction with a simple example to illustrate that different computational objectives will have different optimal communication structures. We compare the optimal routing for data compression, and for computing SVD. Assume that compression converts 2 input streams of size R bits each to an output stream of $R+r$ where $r<R$ [26]. The SVD operator, as discussed in detail in Section IV, converts k input streams of size R bits each, into k eigenvectors of size r bits each with $r<R$. Consider the simple 4-node topology of Figure 1 and the two possible communication structures, with node 0 being the base station and assume all links are of unit length/cost. As derived in [26], data compression requires an exchange of $3R+3r$ (using successive encoding) and $4R+r$ bits respectively for the communication structures (a) and (b). Hence, if $R>2r$, then (a) is better. On the other hand, in the case of SVD if we do not perform in-network computation, then sending all raw data to node 0 results in a cost of $6R$ and $5R$ over the two structures, respectively. If we perform in-network computation, then as detailed in Section IV, the resulting costs are $3R+6r$ and $3R+3r$ for the two structures, respectively. Hence (b) is always better for the SVD computation.

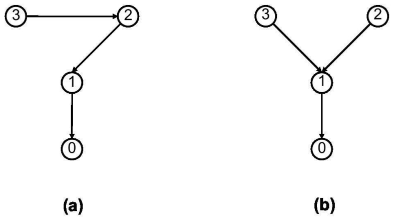


Fig. 1. In-network computation and compressed sensing can have a different optimal communication structure. (a) and (b) represent the two possible communication structures for a simple 4-node topology.

The remainder of this paper is organized as follows. In Section II, we formally present the distributed SVD problem. A lower bound and the optimal solution to the unconstrained problem are provided in Section III. Section IV presents an exact ILP solution for the constrained problem, a number of approximate algorithms with proven approximation ratios, and a distributed implementation of one of these approximate algorithms. In Section V, we discuss various relaxations and extensions, including alternative energy models and noise in sensor measurement. In Section VI we evaluate our algorithm through both numerical simulation and testbed experiments. Related work is discussed in Section VII, and we conclude the paper in Section VIII.

II. Problem Formulation and Preliminaries

In this section, we first give the relevant background on structural health monitoring, then present the network model and formally introduce the problem.

A. Background on Structural Health Monitoring

During the past two decades, the SHM community has become increasingly focused on the use of the structural vibration data to identify degradation or damage within structural systems. The first step in determining if the vibration data collected by a set of sensors represents a healthy or an unhealthy structure is to decompose the spectral density matrix into a set of single-degree-of-freedom systems. Assuming a broadband white input to the system, this can be accomplished by first obtaining an estimate of the output power spectral density (PSD) matrix for each discrete frequency by creating an array of frequency response functions using the Fast Fourier Transform (FFT) from each degree of freedom. Early studies in this field focused on identifying changes in modal frequencies or the eigenvalues of the PSD matrix using the peak picking method [27] to detect damage in large structural systems [28]. More recent studies have observed that viewing changes in modal frequencies in combination with changes in mode shape information (eigenvector of the PSD matrix) makes it increasingly possible to both detect and locate damage within a variety of structural types and configurations [14]–[16]. One of the most widely used method for mode shape estimation is the frequency domain decomposition (FDD) method proposed by Brincker *et al.* [29]. This method involves computing the SVD of the PSD matrix to extract the eigenvectors/mode shapes.

The most common implementation of the FDD method over a wireless sensor network is to have each sensor send its vibration data to a central sensor node which computes the SVD of the PSD matrix. This method requires significant computational power and memory at the central node as well as significant energy consumption in the network to communicate all this data to the central node. For example, if there are 100 sensor nodes in the network, this implementation requires the central sensor node to compute the SVD of a 100×100 PSD matrix as well as having each of the 100 sensor nodes send all their vibration data to one central node.

Zimmerman *et al.* [17] proposed an alternative implementation by decomposing the computation of SVD (graphically represented in Figure 2). Each sensor node is assumed to be aware of the eigenvalues of the PSD matrix (which have been determined using the peak-picking method[1]) and the FFT of its own sensed data stream. Denoting the entire set of nodes as V, if a sensor has the FFT of $N \subset V, |N| > 1$ sensors and all the eigenvalues, then it can compute the SVD of the PSD matrix using $|N|$ sets of FFT results and determine $|N|$ eigenvectors. Let another sensor node be in possession of the FFT of $N' \subset V, |N'| > 1$ sensors. It can perform a similar computation to determine $|N'|$ eigenvectors. To be able to combine results from these two computations to construct the $|N \cup N'|$ eigenvectors, one needs to be able to determine the appropriate scaling factors so that the common eigenvectors has the same constant in both N and N'. This notion is precisely given in the following.

Definition 1: Two computations are called *combinable* if one can determine the appropriate scaling factors to combine them. A computation on N nodes and another computation on N' nodes is combinable if and only if either $N \cap N' \neq \phi$ (that is, there is at least one common sensor in N and N'), or there exists another computation on a set of N'' nodes which is combinable with both N and N'.

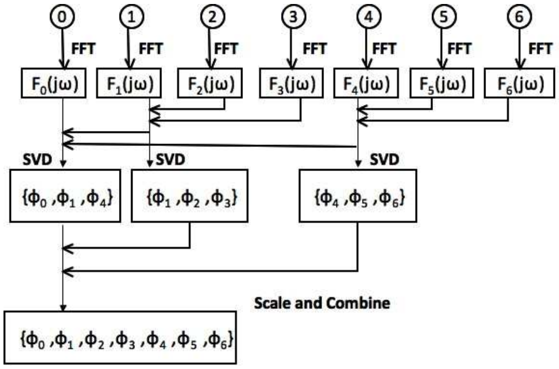

0
1
2
3
4
5
6
FFT
$F_0(j\omega)$
$F_1(j\omega)$
$F_2(j\omega)$
$F_3(j\omega)$
$F_4(j\omega)$
$F_5(j\omega)$
$F_6(j\omega)$
SVD
SVD
SVD
$\{\phi_0, \phi_1, \phi_4\}$
$\{\phi_1, \phi_2, \phi_3\}$
$\{\phi_4, \phi_5, \phi_6\}$
Scale and Combine
$\{\phi_0, \phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6\}$

Fig. 2. Decomposing the computation of SVD using in-network computation. $F_i(j\omega)$ and $\phi_i, 0 \leq i \leq 6$ denote the FFTs and the eigenvectors respectively.

If R denotes the size in bits required to represent the FFT of a sensor stream and r denotes the size in bits to represent a eigenvector, each SVD computation which combines the FFT of k sensor streams reduces the number of bits from kR to kr. Note that the size of the output stream does not depend on R but only on r, which depends on the size of the network.

B. Network Model and Problem Definition

With the above decomposition, the associated communication problem may also be viewed as a *generalized clustering problem*: the solution lies in determining which subset of sensors (*cluster*) should send their FFTs to which common node (*cluster head*), who then computes the SVD for this subset, such that these subsets have the proper overlap to allow individual SVDs to be scaled and combined. This specific overlapping requirement makes it different from most clustering studies in the literature; see a detailed discussion in Section VII.

We proceed to assume a network model of an undirected graph denoted as $G(V,E)$. Each (sensor) node in V acts as both a sensor and a relay. If two nodes can successfully exchange messages directly with each other, there exists a weighted edge $e \in E$ between them, with weight denoted $w_e \geq 0$. Without loss of generality, we take node 0 to be the central node or the base station. We also assume that all sensors (including the base station) are identical in their radio capability (and hence

[1] Peak picking can be done in a separate process preceding the SVD calculation, and does not require transmissions of FFTs; it involves only sending peaks which can be represented in very few bytes.

have the same energy consumption per bit). This is done to keep the presentation simple and can be easily relaxed.

Each node has a local input vibration stream. The goal is to evaluate the SVD of the PSD matrix formed by the input vibration streams of all the sensors. We define a *sensing cycle* to be the time duration in which each sensor performs the sensing task to generate a vibration stream, the SVD is then computed and the mode shapes are made known at the base station. The length of this cycle as well as how this procedure may be used in practice are discussed in Section V. Our objective is to determine the optimal communication structure to minimize the energy consumption in a sensing cycle under a constraint on the maximum duration of a sensing cycle.

The two metrics of interest, namely energy consumption and computational delay are precisely defined as follows.

Energy consumption is defined as the total communication energy consumed in the network in one sensing cycle. Let E_{Tx} and E_{Rx} denote the energy consumed in transmitting and receiving a bit of data, and denote by $E_b = E_{Tx} + E_{Rx}$. We will assume that the energy consumed in transmitting a packet of B bits over an edge e is given by $w_e B E_b$. The edge weight w_e allows us to take into account physical layer factors like link quality, e.g., by using as weight the expected number of retransmissions.

Computational delay is defined as the time it takes to compute a designated function at a sensor node. As observed in [17], [30], the computational time is the chief contributor to delay as packet sizes in sensor systems tend to be very small. Thus, the duration of a sensing cycle depends primarily on the maximum computational delay amongst all sensor nodes. A constraint on the computational delay essentially translates into a constraint on the maximum number of FFT's which can be combined at a node.

Our algorithms and analysis do not depend on the exact model used for energy consumption, provided that it remains a function of the number of bits transmitted. On the other hand, the above choice of energy model does not take into account energy consumed in idle listening, which is often observed to be on the same order as active receiving. The difficulty in incorporating idle listening lies in the fact that it is heavily dependent on the MAC layer used; a MAC protocol with good sleep scheduling mechanisms may consume significantly less energy in idle listening mode than another. The above energy model thus may be interpreted as capturing an ideal MAC with zero idle listening. Alternatively it may also be interpreted as capturing a MAC that never puts nodes to sleep; this is because with active reception and idle listening consuming energy on the same order, the difference between one scheme and another becomes dominated by energy consumed in transmission, which is a function of the total number of bits transmitted. To summarize, the intention behind this energy model is so that we can analyze the optimal communication/routing structure at the network layer without having to make specific assumptions on the underlying MAC layer. With this model, the total energy consumption is effectively translated into the total number of bits communicated.

Also note that the above energy model only concerns communication but not processing. This is because in general processing can occur without turning on the radio, and consumes much less energy by comparison. In addition, operations like FFT is required by each node no matter what communication schemes we use. Therefore not including it in the model will not affect the resulting optimal solution.

We now formally introduce the problem.

Problem P1: Find (1) the set S of sensor nodes on which the SVD computation will take place, (2) for each $s \in S$, their corresponding set N_s of sensor nodes whose FFT will be made available at s, and (3) a routing structure, so as to minimize E, the total energy consumed, subject to the constraint that $|N_s| \leq n_s$, $\forall s \in S$, where n_s denotes the maximum cluster size allowed at node s that corresponds to its computational delay constraint, and that the computations on all pairs $s_1, s_2 \in S$ are combinable.

The set S will also be referred to as the set of cluster heads, and set N_s the cluster associated with head node s. In the above description we have imposed individual delay constraints. Note that the computational delay of one round of SVD is dominated by the largest delay among all nodes if the computations of successive rounds are pipelined. One could also try to minimize the maximum computational delay with a constraint on the energy consumption. Indeed, it can be shown that the dual of the linear programs we propose will optimally solve this alternative formulation.

Theorem 1: There is no polynomial time algorithm that solves P1, unless $P = NP$.

This can be shown through a reduction from set cover. The proof is omitted due to space constraints. The interested reader is referred to [31].

III. A Lower Bound on the Value of P1

To simplify presentation, in this section we will assume that the weights of all edges are equal. This is not restrictive as all bounds derived in this section can be easily modified to incorporate different weights. With this assumption, the energy consumed in sending data from one node to another merely depends on the number of hops on the path between them.

Definition 2: A *data collection tree (DCT)* for $G(V,E)$ is the spanning tree such that the path from each node $v \in V$ to the base station has the minimum weight.

Compared to a minimum spanning tree (MST), a DCT is a shortest-path tree that offers minimum weight on each path to the root rather than over the entire tree. Since all weights are equal, a path of minimum weight is equivalent to that of minimum hop count. Let $d_0(v)$ denote the hop count of node $v \in V$ in the DCT.

The following lemma provides a lower bound on the minimum energy consumption for P1 given any choice of S.

Lemma 1: Consider P1 defined on graph $G(V,E)$, and a set of cluster heads $S \neq \phi$, then a lower bound on the optimal energy consumption, denoted by $E(S)$, is given by

$$E(S) \geq \left((|V|-1)R + \sum_{v \in V} (d_0(v) - 1)r + |S|r \right) E_b. \qquad (1)$$

Proof: For all nodes $v \in V \backslash S$, a message of size R (containing the FFT) needs to be transmitted from v to some node in S. This message goes over at least one hop to reach this node, after which the size reduces to r (the eigenvector). Since the minimum hop count from v to the base station is $d_0(v)$, if the message of size R goes over one hop, the message of size r will go over at least $d_0(v)-1$ hops. As $R > r$, the amount of transmission required includes $|V|-|S|$ transmissions of size R and $\sum_{v\in V\setminus S}(d_0(v)-1)$ transmissions of size r.

In addition to the above, each of the $|S|$ computations needs to be combinable. This means that $\forall s_1, s_2 \in S$, either $N_{s_1} \cap N_{s_2} \neq \phi$ or there exists another node $s_3 \in S$ such that the computations at nodes s_1 and s_3, as well as that at nodes s_2 and s_3 are combinable, respectively. To understand how many extra messages are needed to satisfy this constraint, we construct the following graph $G^S(S, E^S)$: an edge is added to E^S between nodes s_1 and s_2, $s_1, s_2 \in S$, only if $N_{s_1} \cap N_{s_2} \neq \phi$. Each edge in this graph thus represents at least one common node between N_{s_1} and N_{s_2}; each common node needs to transmit a message of size R to both s_1 and s_2.

It follows that each edge implies at least one extra transmission of size R in addition to the $|V|-|S|$ transmissions of size R computed earlier. Two nodes in $s_1, s_2 \in S$ are combinable if and only if there exists a path between s_1 and s_2 in $G^S(S, E^S)$. For a path to exist between every pair of nodes, $G^S(S, E^S)$ needs to have at least $|S|-1$ edges. This means at least $|S|-1$ extra transmissions of size R are required for all pairs $s_1, s_2 \in S$ to be combinable. Taking this into account, at least $|V|-1$ transmissions of size R and $\sum_{v\in V\setminus S}(d_0(v)-1)$ transmissions of size r have to take place.

Finally, computed eigenvectors from nodes $v \in S$ each goes through at least $d_0(v)$ hops. Combining all of the above yields

$$\begin{aligned} E(S) \quad &\geq \quad \left((|V|-1)R + \sum_{v\in V\setminus S}(d_0(v)-1)r + \sum_{v\in S} d_0(v)r\right)E_b \\ &= \quad \left((|V|-1)R + \sum_{v\in V}(d_0(v)-1)r + |S|r\right)E_b \ . \end{aligned} \tag{2}$$

■

An interesting observation is that the lower bound only depends on the size of S and not its membership. One way to get close to this bound is to limit the delivery of any FFT to a single hop and route the FFT and the subsequent eigenvector along shortest paths. This motivates a particular solution for any given tree structure.

Definition 3: Consider a graph $G(V,E)$ and a routing tree T. Define a communication structure $A_{P2}(T)$ as follows: (1) All non-leaf nodes in T constitute the set S, (2) cluster $N_s, s \in S$ consists of all immediate children of $s \in S$, and (3) each node sends its own FFT to its parent node on T, and a node $s \in S$ sends eigenvectors for itself and its children along T to the base station. This will be referred to as *tree solution T*.

We next consider an unconstrained version of P1, i.e., by removing the computational delay constraint. We refer to this unconstrained problem as **P2**.

Lemma 2: Consider P2 defined on a graph $G(V,E)$, and a routing tree denoted by T defined on the same graph. Let $d_T(v)$ denote the hop count of node $v \in V$ in T. Then tree solution $A_{P2}(T)$ is feasible and has an energy consumption

$$E_{A_{P2}}(T) = \left((|V|-1)R + \sum_{v\in V}(d_T(v)-1)r + |S|r\right)E_b \ . \tag{3}$$

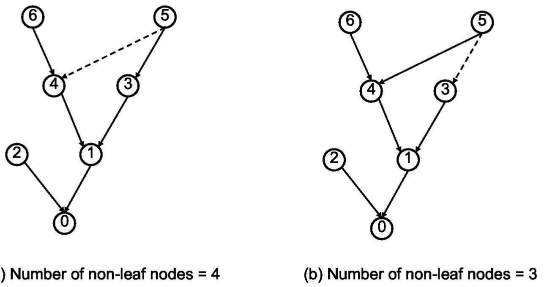
Proof: We first show feasibility, i.e., each pair of nodes $s_1, s_2 \in S$ are combinable. Since S consists of all non-leaf nodes on a tree, there exists a path between any pair of these nodes. Thus $A_{P2}(T)$ is feasible.

Next since each node (except for the base station) sends its FFT to its parent, this results in a cost of $(|V|-1)RE_b$; each non-leaf node computes the SVD from its children's FFT and its own, and then sends the eigenvectors to the base station, resulting in $\sum_{v\in V}(d_T(v)-1)r + |S|r$ bits. Putting everything together yields the lemma. ■

This lemma suggests that of all solutions given by a tree structure, the one that minimizes both $d_T(v)$ and $|S|$ will result in the smallest energy consumption. This motivates the construction of a DCT (which minimizes $d_T(v)$) that has a minimum number of non-leaf nodes (which minimizes $|S|$).

Definition 4: A *minimum non-leaf node data collection tree*, or MDCT, defined on graph $G(V,E)$ is a DCT that has the smallest number of non-leaf nodes among all DCTs defined on $G(V,E)$. We will denote this tree as T_M.

A key property of an MDCT is that it is impossible to move all the children of non-leaf node $v \in V$ on T_M to other *non-leaf nodes* of height $\leq d_T(v)$. This is because if this could be done then we can effectively reduce the number of non-leaf nodes on T_M, which is a contradiction. Figure 3 gives an example: both (a) and (b) are DCTs on the same graph, but the former is not a MDCT while the latter is.



(a) Number of non-leaf nodes = 4

(b) Number of non-leaf nodes = 3

Fig. 3. Two data collection trees for the same network. The solid lines represent the edges of the tree. T_M is the tree in (b).

Theorem 2: Consider P2 defined on $G(V,E)$, and an associated MDCT T_M with cluster head set S. Under the condition $R > 2r$, an optimal solution to P2 is given by $A_{P2}(T_M)$.[2]

Proof: From Lemma 2 we know that $E_{A_{P2}}(T_M)$ matches exactly the lower bound given in (2). Consider any other solution with a cluster head set S' such that $|S'| \geq |S|$. By Lemma 1 $E(S') \geq E_{A_{P2}}(T)$ so any solution with a larger set S' is no better.

Consider next any solution with a set S'' such that $|S''| < |S|$. By Lemma 1, using S'' instead of S reduces the energy by no more than $(|S|-|S''|)rE_b$. On the other hand, consider a node $v \in S$ and $v \notin S''$. By the property of the MDCT T_M, there are only three possibilities in how the children of v can send their

[2] The condition $R > 2r$ is easily satisfied in SVD computation for SHM.

FFT under the new solution S'': (1) each child of v sends its FFT to some node $v'' \in S''$ with $d_0(v'') = d_0(v)$ via a single hop; (2) at least one child of v sends its FFT to a $v'' \in S''$ with $d_0(v'') > d_0(v)$ via a single hop; (3) at least one child of v sends its FFT over at least $d \geq 2$ hops to reach $v'' \in S''$ with $d_0(v'') + d \geq d_0(v) + 1$. Denote these sets as V_1, V_2 and V_3, respectively. Note that $|S| = |S \cap S''| + |V_1| + |V_2 \cup V_3|$.

In case (1), at least one such $v'' \in S''$ cannot be in S, and these v'' nodes will be distinct for different $v \in S, \notin S''$ nodes, for otherwise it contradicts the definition of an MDCT. Thus for each such $v \in S, \notin S''$ there corresponds a $v'' \in S'', \notin S$. Therefore case (1) does not contribute to any reduction in energy consumption compare solution S'' to S. Thus, $|S''| < |S|$ can only be true if either (2) or (3) is true for some $v \in S, \notin S''$; in other words, $|S| - |S''| = |V_2 \cup V_3|$. For each such v, if it falls under case (2) then there is an energy increase (from S to S'') of at least rE_b due to the height increase of v'' over v; if it falls under case (3), the energy increase is at least $(R-r)E_b > rE_b$ by the condition stated in the theorem. Thus the total energy increase is at least rE_b for each $v \in V_2 \cup V_3$; therefore the increase is at least $(|S| - |S''|)r$. Hence any solution with a smaller set S'' is no better, completing the proof. ■

To summarize, an MDCT yields the optimal solution for P2, which also serves as a lower bound to the value of P1. Note that in this solution the overlap between clusters is through cluster heads; all cluster heads (except for the base station) is a member of another cluster.

IV. Exact and Approximate Algorithms

We next present an integer linear program (ILP) to solve P1 exactly and a $O(log(|V|))$ approximation algorithm for P1.

A. An Exact ILP for P1

We first introduce optimization variables used in the ILP.

The set of variables $x_{ij}, i, j \in V$ define both the sets S and $N_s, \forall s \in S$ as follows. $x_{ij} := 1$ if the FFT of node i is evaluated at node j (i.e. $i \in N_j$), and 0 otherwise. $x_{ii} := 1$ if $i \in S$ and 0 otherwise.

$p_{ijk} := 1$ if the FFT of node k is evaluated at both nodes i and j. This notation is used for convenience of presentation only as it is completely determined by $x_{ij}, i, j \in V$.

Finally, the variables c_{ijn} recursively verifies the combinability relationship between two nodes $i, j \in S$ as follows:

$$c_{ijn} = \begin{cases} 1 & \text{if } n = 0 \text{ and } \sum_{k \in V} p_{ijk} \geq 1, \\ 1 & \text{if } 0 < n < |V| \text{ and} \\ & \sum_{k \in V} c_{ik(n-1)} . c_{jk(n-1)} + c_{ij(n-1)} \geq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Thus $c_{ij0} = 1$ if the pair $i, j \in S$ share common nodes in their respective clusters, $c_{ij1} = 1$ if the pair i, j either share common nodes directly or each shares common nodes with a common third cluster, and so on. If the pair $i, j \in S$ are combinable, we will have $c_{ij(|V|-1)} = 1$.

Finally, W_{ij} denotes the sum weight of all edges along the shortest path from node i to node j.

The ILP below solves P1 exactly, where the minimization is over the choice of $x_{ij}, \forall i, j \in V$.

$$\textbf{(ILP\_P1)} \quad \min \ \sum_{i \in V, j \in V} x_{ij} E_b \left(RW_{ij} + rW_{j0} \right) \tag{5}$$

$$\text{s.t.}$$

$$\sum_{j \in V, j \neq i} \frac{x_{ji}}{V} \leq x_{ii} \leq \sum_{j \in V, j \neq i} x_{ji}, \forall i \in V \tag{6}$$

$$\sum_{j \in V} x_{ij} \geq 1, \forall i \in V \tag{7}$$

$$p_{ijk} \leq \frac{x_{ki} + x_{kj}}{2}, \forall i, j, k \in V \tag{8}$$

$$c_{ij0} \leq \sum_{k \in V} p_{ijk}, \forall i, j \in V \tag{9}$$

$$c_{ij(|V|-1))} \geq x_{ii} + x_{jj} - 1, \forall i, j \in V \tag{10}$$

$$t_{ijkn} \leq \frac{c_{ik(n-1)} + c_{jk(n-1)}}{2}, \forall i, j, k \in V, 0 < n < |V| \tag{11}$$

$$c_{ijn} \leq c_{ij(n-1)} + \sum_{k \in V} t_{ijk(n-1)}, \forall i, j \in V, 0 < n < |V| \tag{12}$$

$$c_{iin} = 0, \forall i \in V, 0 \leq n < |V| \tag{13}$$

$$\sum_{i \in V} x_{ij} \leq n_j, \forall j \in V \tag{14}$$

$$x_{ij}, c_{ijk}, p_{ijk}, t_{ijkn} \in \{0,1\} \forall i, j, k \in V, 0 \leq n < |V| \tag{15}$$

The objective (Eqn (5)) is fairly straightforward: if the FFT of node i is sent to node j, it costs $RW_{ij}E_b$. The FFT from i produces a unique eigenvector of size r at node j as a result of this SVD computation, which costs $rW_{j0}E_b$ to send to the base station.

The first constraint (Eqn (6)) sets the value of x_{ii} to 1 if $N_i \neq \phi$, and 0 otherwise (note that if $N_i \neq \phi$, then $1 \leq \sum_{j \in V, j \neq i} x_{ji} \leq |V|$). Eqn (7) ensures that the FFT of every sensor node is sent to at least one node. Eqn (8) ensures that $p_{ijk} = 1$ if the FFT from node k is sent to both nodes i and j.

The next five constraints ensure the combinability of the solution by limiting the value of c_{ijn}. Eqn (9) ensures that $c_{ij0} = 1$ if there is at least one node common to N_i and N_j. Eqn (10) states that if both $i, j \in S$, the computations at i and j should be combinable. Eqns (11) and (12) populate the value of c_{ijn}. Note that t_{ijkn} is a temporary variable introduced to express the quadratic condition in Equation (4) as a linear function. Note that the presence of Eqn (10) forces Eqns (8), (9), (11), and (12) to assign the maximum possible value to the LHS; similarly, the presence of the latter forces (10) to assign the minimum possible value to the LHS.

Eqn (13) sets the value of c_{iin} to zero for every $i \in V, 0 \leq n < |V|$. This prohibits a corner case where c_{ijn} is set to 1 by setting $c_{ii(n-1)}$ to 1 without ensuring that the computation at i and j are combinable. Finally, Eqn (14) imposes the computational delay constraint at each sensor node.

B. Degree-Constrained DCT: Problem P3

In this and the next two subsections we will develop a $O(log(|V|))$ approximation to the optimal solution of P1. To simplify the presentation, we will again assume that all edge weights are equal, an assumption easily relaxed without affecting the approximation ratios.

The basic idea is to first use a DCT to find a feasible solution to P1. A feasible solution requires that each cluster is size-limited due to the computational delay constraint: a node v cannot have more than $n_v - 1$ immediate children. This leads to the following definition.

Definition 5: A *degree-constrained data collection tree*, or DDCT, is a tree T which minimizes $\sum_{v\in V} d_T(v)$ under the constraint that a node $v \in V$ has no more than $n_v - 1$ immediate children, where $n_v, \forall v \in V$ are given constants.

Problem P3: Find a DDCT for $G(V,E)$, which in turn determines the set S, clusters $N_s, \forall s \in S$, and the routing structure.

That a solution to P3 is feasible for P1 is obvious, but it may not be optimal for P1, even if it has the fewest non-leaf nodes among all DDCTs because a node may no longer be on its shortest path.

It is worth noting that P3 is also NP-hard; it is APX-hard even when weights on edges satisfy the triangle inequality [24]. Results on P3 are known only for complete graphs whose weights satisfy the triangle inequality [24], [25]. To the best of our knowledge our work here is the first to propose algorithms with proven approximation ratios for P3 in graphs induced by a communication network.

We proceed as follows. We first present an ILP (ILP_P3) to solve P3 exactly. This ILP has much fewer variables and constraints than ILP_P1, and hence takes less time to solve. We then relax ILP_P3 to an LP and solve it via appropriate rounding of fractional values. This rounding algorithm, referred to as algorithm **LPR**, is thus an approximation algorithm for P3, and therefore also an approximation algorithm for P1. We derive the approximation factor for LPR with respect to problem P1 in Theorem 3. Finally, based on the intuition derived while analyzing LPR, we present a simpler, distributed approximation algorithm with the same asymptotic approximation factor.

C. An ILP for Problem P3

We define the following variables used in the ILP in finding a DDCT. For a given $G(V,E)$, define a graph $\bar{G}(V,\bar{E})$ with directed edges, by replacing each undirected edge in E with two directed edges, one in each direction. Let $O_v, v \in V$ denote the set of outgoing edges from node v in $\bar{E}$. Similarly, let $I_v, v \in V$ denote the set of incoming edges into node v in $\bar{E}$.

The set of variables $x_e, e \in \bar{E}$ define whether an edge is on the DCT as follows. $x_e := 1$ if edge e is on the DCT, and 0 otherwise. The variable $f_e, e \in \bar{E}$ will be referred to as the *flow value* over the edge e; it denotes the number of nodes using edge e to reach the base station on the DCT. $f_e = 0$ if edge e is not on the DCT.

The following ILP solves P3 exactly, where the minimization is over the choice of $x_e, \forall e \in \bar{E}$.

$$\textbf{(ILP\_P3)} \quad \min \textstyle\sum_{e\in\bar{E}} f_e \tag{16}$$

$$\textstyle\sum_{e\in I_0} f_e - \sum_{e\in O_0} f_e = |V| - 1 \tag{17}$$

$$\textstyle\sum_{e\in I_v} f_e - \sum_{e\in O_v} f_e = -1, \forall v \in V\backslash\{0\} \tag{18}$$

$$f_e \le (|V|-1)x_e, \forall e \in \bar{E} \tag{19}$$

$$\textstyle\sum_{e\in\bar{E}} x_e = |V| - 1 \tag{20}$$

$$\textstyle\sum_{e\in O_v} x_e = 1, \forall v \in V\backslash\{0\} \tag{21}$$

$$\textstyle\sum_{e\in I_v} x_e \le n_v - 1, \forall v \in V \tag{22}$$

$$x_e \in \{0,1\}, \forall e \in \bar{E} \tag{23}$$

$$f_e \in \{0,1,\ldots,|V|-1\}, \forall e \in \bar{E} \tag{24}$$

The objective function minimizes the total flow, which essentially minimizes $\sum_{v\in V} d_T(v)$.

The first two constraints ensure that each node sends a unit flow towards the base station. The third constraint forces f_e to be 0 if x_e is 0, otherwise, it is redundant. Eqn (20) ensures that the output has exactly $|V|-1$ edges. Eqns (21) and (22)) ensure that there is no more than one outgoing edge per vertex (other than the base station) and no more than $n_v - 1$ incoming edges into vertex v. Eqns (20) and (21) together ensure that the output is a tree and Eqn (22) ensures that a node v has no more than $n_v - 1$ immediate children.

D. Algorithm LPR: an LP Rounding Approximation

We next present a polynomial-time approximation algorithm which relaxes ILP_P3 to a linear program (LP), by allowing $0 \le x_e \le 1$ and $f_e \ge 0$ to be fractional and appropriately rounding the fractional values. This algorithm is referred to as LPR and shown in Figure 4.

LPR is an iterative algorithm. At each iteration, we solve the fractional ILP along with the additional constraints that the edges for whom x_e was set to 1 in the previous iteration, remains set to 1. At iteration h, we add the nodes which will be h hops away from the root in the final DCT. For every node at height $h-1$, if there are more than $n_v - 1$ incoming edges with non-zero x_e, choose the largest $n_v - 1$ values and set them to 1 and the remaining to zero; and if there are less than $n_v - 1$ such edges, set all of them to 1. Also, add the nodes from which these edges (whose x_e's were set to 1) emanate, to the DCT at height h.

```
$NV = \{0\}$, $NE = \phi$, $h = 0$, assign $h_v = -1,\ \forall v \in V\backslash\{0\}$ and
$h_0 = 0$
while $(NV != V)$ do
    $h = h + 1$
    Solve fractional ILP_P3 + constraint
    $x_e = 1, \forall e \in NE$
    For $\forall v \in NV$, $h_v = h - 1$
        /* For $v \in V$, define $S_v$ to be the set of
        incoming edges at $v$ with $x_e > 0$ */
        If $|S_v| > n_v - 1$
            $S'_v$ = Sort$(S_v)$
            /* Sort() will sort according to $x_e$ in a
            descending order */
            $S_v$ = Extract$(S'_v, n_v - 1)$
            /* Extract$(S,n)$ will extract the first $n$
            edges from $S$ */
        $\forall e \in S_v$
            $x_e = 1$
            Add$(NE,e)$ /* Add(X,y) adds y to X */
            $v$ = Outgoing$(e)$
            /* Outgoing(e) returns the vertex from
            which edge $e$ emanates */
            Add$(NV,v)$
            $h_v = h$
```

Fig. 4. Algorithm LPR: The LP rounding approximation algorithm for P3.

E. The Approximation Factor of LPR

Even though LPR makes no assumptions on the network, our derivation of the approximation factor assumes the following: (1) $n_{min} \ge 3$, where $n_{min} = \min_{v\in V} n_v$, (2) the unconstrained MDCT constructed over $G(V,E)$ has a height of $O(\log(|V|))$, and (3) nodes can transmit to each other if the distance between them is less than a transmission range R_{tx}.

To understand (1), note that if $n_v = 2, \forall v \in V$, then each node has at most 1 child and the constructed tree is thus linear (a chain). The problem subsequently reduces to the traveling salesman problem. Similarly, if most nodes disallow more than 1 child, the resulting tree will be close to linear, which is not a very interesting routing structure to study. Finally, and more importantly, most existing sensor platforms have sufficient computational power to quickly combine FFT's from at least 3 nodes, easily satisfying this assumption. Assumption (2) is also easily satisfied as sensor networks used for SHM are in general not very sparse. Assumption (3) is very commonly adopted for analytical tractability. However, our analysis is not heavily dependent on this assumption (more is discussed in the footnote in the proof of Lemma 4) and the same approximation factor also holds under more realistic physical layer assumptions.

We next derive the approximation factor of LPR with respect to P1. The analysis is based on the observation that the approximation factor is essentially the ratio between the height of the DDCT constructed using LPR and the height of the MDCT (discussed in more detail in the proof of Theorem 3).

Denote the height of the MDCT by h_{orig} and the height of the DDCT generated by algorithm LPR h_{ddct}. Define a *non-full node* to be a node v at height $h < h_{ddct}$ which has less than $n_v - 1$ children. A height $1 \le h < h_{ddct}$ is defined to be a *non-full height* if there exists at least one non-full node at height h. We then have the following lemma.

Lemma 3: Consider running algorithm LPR on a set of m nodes with a randomly selected base station and a topology such that the maximum set of nodes that cannot transmit to each other has a size p. Then the resulting DDCT cannot have more than p non-full heights.

Proof: We prove this by contradiction. Let there be $p+1$ non-full heights: $h_1 < \ldots < h_{p+1}$. Let v_i be a non-full node at height $h_i, 1 \le i \le p+1$. Then, v_i, v_j, $1 \le i < j \le h_{p+1}$ cannot transmit to each other, for otherwise LPR would have labeled v_j as the child of v_i. Thus none of the nodes $v_1, \ldots, v_{p+1}$ can transmit to each other. However, by assumption we cannot have more than p nodes which cannot transmit to each other, thus a contradiction. ■

Lemma 4: Under the assumption that the height of the MDCT $h_{orig} = O(\log(|V|))$, the height of the DDCT constructed by LPR is $h_{ddct} = \Theta(\log(|V|))$.

Proof: By the construction of the MDCT, the maximum distance of a node from the base station is $h_{orig} R_{tx}$[3]. Using geometric arguments similar to the ones used in [34], it's easy to show that the set of nodes none of which can transmit to each other has a size of no more than $\frac{2\pi}{\cos^{-1}\left(1 - \frac{1}{2h_{orig}^2}\right)} \le \frac{2\pi}{\cos^{-1}\left(1 - \frac{1}{2c^2 \log^2(|V|)}\right)} \approx 2\pi c \log(|V|)$, for some constant c, where the equality follows from the small angle approximation $\cos(x) \approx 1 - \frac{x^2}{2}$.

Thus by Lemma 3, there are no more than $2\pi c \log(|V|)$ non-full heights. At the same time, the number of full heights is $\Theta(\log(|V|))$ by definition. Hence $h_{ddct} = \Theta(\log(|V|))$. ■

Theorem 3: The approximation factor of LPR is $O(\log(|V|))$.

Proof: To derive the approximation factor, we compare the energy consumed in the DDCT constructed using LPR (given by Lemma 2) to the lower bound on the optimal solution of P1 (given in Lemma 1). First, we note that $|S| \ge \left(\frac{V}{n_{max}}\right)$ in the optimal solution and $|S| = c_1 \left(\frac{V}{n_{min}}\right)$ in the DDCT (as $h_{ddct} = \Theta(\log(|V|))$) where c_1 is a positive constant, $n_{max} = \max_{v \in V} n_v$ and $n_{min} = \min_{v \in V} n_v$. Thus, the approximation factor is $\le \frac{\sum_{v \in V}(d_{ddct}(v) - 1) + c_1\left(\frac{V}{n_{min}}\right)}{\sum_{v \in V}(d_0(v) - 1) + \left(\frac{V}{n_{max}}\right)} \le \log(|V|)$, where $d_{ddct(v)}$ denotes the hop count of node v in the DDCT. The final inequality holds because $h_{org} \le c_2 \log(|V|)$ and $h_{ddct} = c_3 \log(|V|)$, for some positive constants c_2 and c_3. Hence the approximation factor is $O(\log(|V|))$. ■

F. A Distributed Approximation Algorithm (DAA)

The approximation algorithm LPR is centralized as it requires solving an LP globally. We now present a simpler, distributed algorithm with the same asymptotic approximation factor.

The proof of Lemma 3 uses the following observation from LPR: at height h, if there exists a node v with more than $n_v - 1$ neighbors which are not yet a part of the tree, the algorithm will add $n_v - 1$ children to it. Otherwise, all its neighbors not yet a part of the tree will be added as its children.

Using this intuition, we propose a modified Dijkstra's shortest path algorithm DAA in Figure 5. This algorithm satisfies the observation made in the previous paragraph, hence Lemma 3 holds, and so do Lemma 4 and Theorem 3. Thus, the approximation factor for DAA is also $O(\log(|V|))$. The tree is built top down from the root with each node v choosing its $n_v - 1$ children arbitrarily. Hence, like any shortest path algorithm [35] it can be built by message exchanges only between neighboring nodes. We will compare this modified Dijkstra's algorithm with LPR through simulation in Section VI.

```
$NV = \{0\}$, $h_v = \infty,\ \forall v \in V \backslash \{0\}$, $h_0 = 0$, $C_v = 0,\ \forall v \in V$.
/* $C_v$ denotes the number of children of node $v$ */
while $(NV != V)$ do
    /* Define $E'$ to be the set of edges which
connect nodes
    in $v \in NV$ and $v' \in V \backslash NV$ and $C_v < n_v - 1$ */
    $\forall e : v \rightarrow v' \in E'$
        $h'_v = \mathtt{min}(h'_v, h_v + 1)$
    $v_{min} = \mathtt{argmin}_v \{h_v \mid \forall v \in V \backslash NV\}$
    Add$(NV, v_{min})$.
    $v_{parent}$ = Parent$(v_{min})$
    /* Parent$(v)$ returns the parent of $v$ in $NV$ */
    $C_{v_{parent}} = C_{v_{parent}} + 1$
    $h_v = \infty,\ \forall v \in V \backslash NV$
```

Fig. 5. Algorithm DAA: Modified Dijkstra's algorithm for P3.

V. Discussion

In this section we discuss ways to include additional constraints like accuracy if sensor measurements are noisy, as

[3] Note that due to fading effects, the transmission range may not be a constant. However, there will always exist distances R_0 and R_1 such that if two nodes are within R_0 of each other, they can transmit to each other with negligible loss, and if they are more than R_1 apart, they cannot exchange packets with each other [32], [33]. R_0 and R_1 may be much smaller and larger respectively than the actual transmission range; replacing R_{tx} by these constants appropriately allows the same argument to go through for a more general physical layer model.

well as the applicability of distributed SVD computation in practice.

The model and algorithms presented here can be easily extended to include additional constraints, including accuracy and storage. In our decentralized SVD computation, the eigenvectors are determined by linearly combining those computed locally at different sensor nodes. If the sensors are noiseless, then the eigenvectors computed using this decomposition will exactly match the actual eigenvectors. However, the presence of noise in the sensed values can lead to errors in the computation [36]. This is because in a centralized implementation, a least-squares effect minimizes the error due to noise across all eigenvectors, whereas the decentralized implementation allows this error to accumulate through each combination of locally computed eigenvectors.

The larger the number of FFT's being combined at each sensor node, the smaller this error. Hence a desired accuracy will impose a constraint on the minimum cluster size $|N_s|, s \in S$. This is the opposite of the delay constraint, and incorporating it in our models is quite straightforward. Denote this constraint by n_a, i.e., $|N_s| \geq n_a, \forall s \in S$. Then in ILP_P1, the following constraint is added: $\sum_{i \in V} x_{ij} \geq n_a x_{jj}, \forall j \in V$. Similarly, in ILP_P3, we add (1) $\sum_{e \in I_v} x_e \geq (n_a - 1) l_v, \forall v \in V$, where variable $l_v \in \{0,1\}$ is set to 1 if v is a non-leaf node, and (2) $\sum_{e \in I_v} x_e / |V| \leq l_v \leq \sum_{e \in I_v} x_e, \forall v \in V$, to ensure that l_v is set 1 only if v is a non-leaf node. Finally, the two approximation algorithms, LPR and DAA, can both be easily modified to maintain the number of children of each node in the data collection to be greater than $n_a - 1$. The effect of an added accuracy constraint will be examined in numerical studies presented in Section VI.

As the number of FFT's being computed at a node increases, not only the delay but also the storage required increases [36]. A storage constraint acts in a way very similar to the delay constraint: it essentially bounds the maximum number of FFT's that can be combined at a sensor. Therefore to incorporate this constraint we simply need to upper bound the value of $|N_s|$ to be the lesser of the two, which results in an identical problem.

While the proposed SVD computation can run continuously as a stream process, in practice it suffices to schedule it several times a day, each lasting on the order of minutes (the actual duration of the sensing cycle depends on the size of the FFT and the computation capacity of the sensors), as one does not in general expect mode shapes of a structure to change rapidly over time. Even though the task is performed infrequently, the saving in each operation is indeed significant (see results in Section VI), and the accumulated effect are undoubtedly beneficial for a sensor network to have a lifetime on the order of months or years.

One weakness common to many in-network processing methods is that they typically deliver *summaries* or *features* of data rather than raw data itself; thus we potentially lose the ability to store and post-analyze the data (e.g., for an entirely different purpose than originally intended). In this sense, this type of operation is most advantageous when used in a real-time setting concerning instantaneous detection and diagnosis. For instance, a human inspector can use this approach (i.e., activate this SVD operation) to quickly check the mode shapes of a structure before deciding whether and what more (manual) inspection is needed.

VI. Simulation and Experimentation

We use both simulation and experimentation on a real sensor platform to evaluate the performance of the proposed algorithms.

A. Simulations

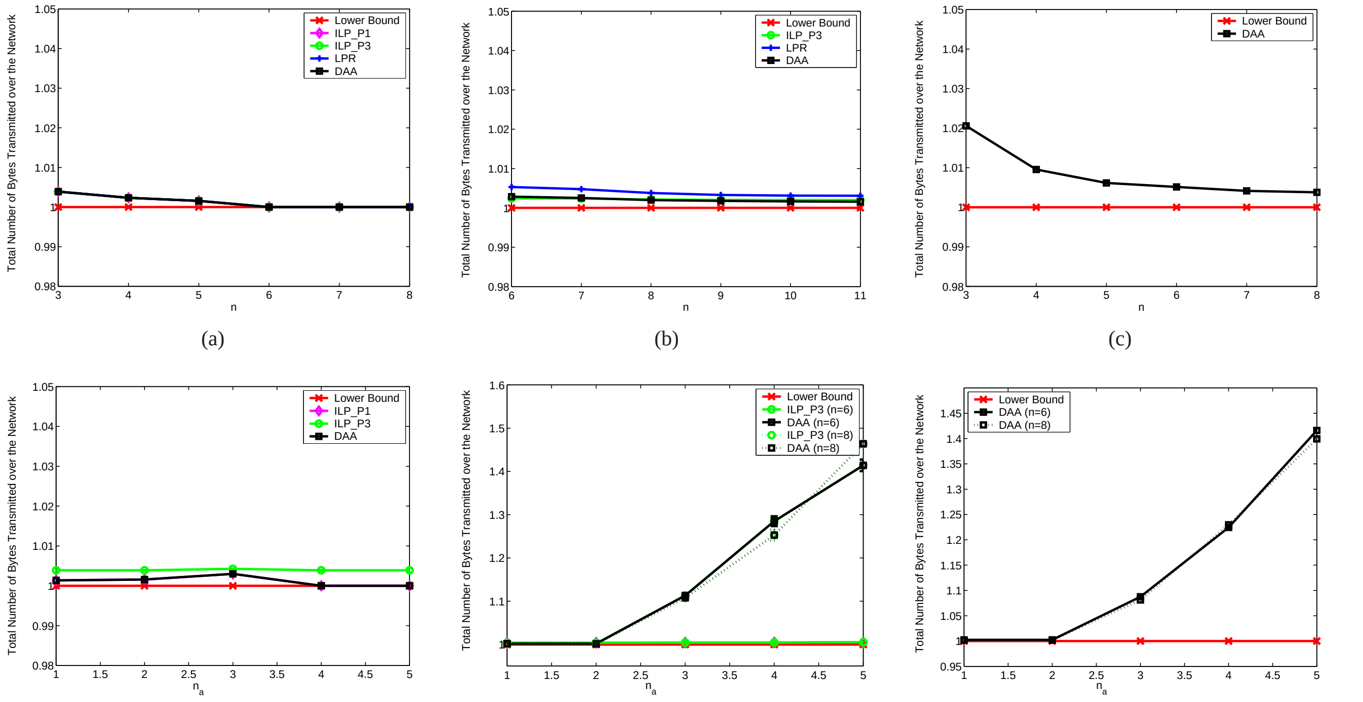
For simulation we use CPLEX [37] to solve the ILPs, and all simulations are done on topologies generated by randomly distributing nodes assuming a density of 8 nodes per 100 square meters. The transmission range is assumed to be 10m. For each simulation parameter, we generate 100 random topologies and plot the average as well as error-bars showing 2× variance. (Note that the variance after averaging over 100 runs becomes negligible leading to very small error bars in all figures.) For comparison, we also report (i) the performance without in-network computation, and (ii) the performance with in-network computation over randomly generated clusters where the cluster-head for each node is chosen uniformly at random, and once clusters have been thus formed, FFT's are exchanged between cluster-heads to ensure combinability. A performance comparison with the method proposed in [8]–[10] is discussed in Section VII. For the SVD computation, we use $R = 8192$ bytes and $r = 32$ bytes [17]. We also assume that the computational delay constraint is the same for all nodes: $n_v = n, \forall v \in V$.

We first examine the effect of delay constraint n on energy consumption by solving the optimization problem proposed in Section II-B (without incorporating the accuracy constraint in the formulation). Figure 6(a) compares the number of bytes transmitted under the lower bound (Lemma 1), using the optimal communication structures derived by solving ILP_P1 (Section IV-A), and using the three approximation algorithms ILP_P3 (Section IV-C), LPR (Figure 4), and DAA (Figure 5), for different values of n[4], with $|V| = 6$.

We observe that the approximation algorithms perform very close to the optimal. It takes more than one hour of computation to solve the ILP_P1 for $|V| > 6$ on a 2.99 GHz machine with 4 GB of RAM. Hence for larger values of $|V|$ we only compare the three approximation algorithms against the lower bound, shown in Figure 6(b). We note that (i) all approximation algorithms are within 1% of the optimal, and (ii) DAA outperforms LPR. These results also demonstrate the advantage of using the ILP_P3 over ILP_P1; it runs much faster and converges within an hour up to $|V| = 40$.

For even larger value of $|V|$, we compare the performance of DAA (as it consistently outperforms LPR) against the lower bound in Figure 6(c). We observe that it is always within 2% of the optimal. These results clearly demonstrate the advantage of in-network computation as the number of bytes transmitted over the network are reduced by more than a third. Finally,

[4] The values of n chosen are typical for the Narada sensor platform and the delay constraints associated with the SHM application.



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 6. Simulation Results. Ratio of the number of bits transmitted with different algorithms and the lower bound. (a) $|V| = 6$ $(1.0, 1.2)$. (b) $|V| = 30$ $(1.5, 1.5)$. (c) $|V| = 200$ $(3.1, 3.1)$. The numbers in brackets denote the following ratios: (the ratio of number of bytes transmitted in the network without in-network computation and the lower bound, the ratio of number of bytes transmitted in the network with random clustering and the lower bound). Simulation Results with an accuracy constraint. (g) $|V| = 5, n = 5$. (h) $|V| = 30$. (i) $|V| = 200$.

Figure 6(c) also shows the trade-off between communication energy and computational delay. The more delay allowed per node (larger the value of n), the smaller the energy consumed in the network.

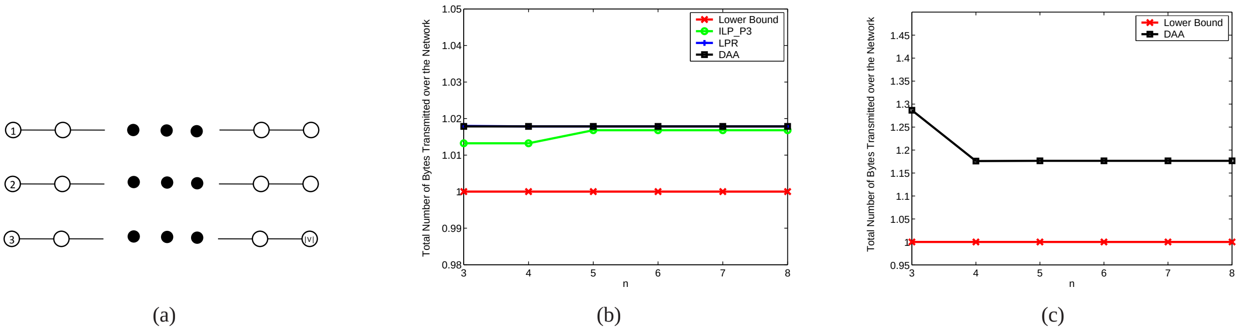
In Figures 6(d)-6(f) we compare the performance of different approximation schemes after incorporating an accuracy constraint in the formulation for different values of $|V|$, n and n_a[5]. In this scenario, we observe that ILP_P3 yields results within 2% of the optimal while DAA yields values within 45% of the optimal. And the advantage of using a better centralized algorithm becomes more pronounced as the value of n_a increases as any sub-optimal local decision in this scenario leads to an extra transmission of a FFT (R bits) and not just an eigenvector (r bits).

We next evaluate the performance of the proposed algorithms in a tri-linear topology of Figure 7(a). This topology is representative of what is deployed over a highway bridge. For such structures, the sensors are attached to the underside of the bridge and typically the topology is such that sensors are placed at regular spacings along the length of the bridge, and a few such parallel lines span the width of the bridge (typically about 2-4 traffic lanes). Such a topology would also be applicable in the monitoring of tall buildings: in experiment with model buildings, sensors are typically placed at the same lateral positions on each floor, resulting in a few parallel lines (vertically in a 3-D space).

Figure 7(b) compares the performance of ILP_P3, LPR and DAA for this tri-linear topology with $|V| = 30$. We again observe that these approximation schemes are within 2% of the optimal. In this topology, the ratio of the number of bits transmitted without in-network computation and the lower bound is 5.6, while the same ratio with random clustering and the lower bound is 3.3. Thus, the improvements become even more significant in this more realistic topology.

Our simulations have so far assumed a binary physical layer without considering the effects of shadowing and random fading which can cause packet losses. Incorporating these effects in our model is quite straightforward as the proposed algorithms allows each edge to have a different weight. Choosing this weight to increase with the loss rate will ensure that good links (with lower loss rates) are preferred over bad ones for routing data. Figure 7(c) compares the performance of DAA against the lower bound for a randomly generated 30 node topology assuming log-normal shadowing and Rayleigh fading at the physical layer. The weight of an edge e is set to $w_e = \frac{1}{1-l_e}$ where l_e is the loss rate on edge e and is empirically measured. The choice of this weight, which represents the expected amount of transmissions required per packet, ensures that edges with low loss rate are preferred by our routing algorithms. Again, we observe that the approximation algorithm is within 30% of the optimal. In this topology, the ratio of the number of bits transmitted without in-network computation and the lower bound is 7.4, while the same ratio with random clustering and the lower bound is 10.1. Since physical layer losses cause extra transmissions of size

[5] The value of n_a depends on the noise level in the sensors and one needs to calibrate the sensors to determine how large the cluster size needs to be.



(a)

(b)

(c)

Fig. 7. (a) The tri-linear topology. (b) Ratio of the number of bits transmitted with different algorithms and the lower bound in the tri-linear topology; $|V| = 30$. (c) Ratio of the number of bits transmitted with different algorithms and the lower bound with a physical layer shadowing and fading; $|V| = 30$.

R, the improvement of using networked computation becomes more significant in this scenario.

B. Experiments

We next evaluate the performance of DAA on a real sensor platform, the Narada sensing unit developed at the University of Michigan [11]. This wireless device is powered by an Atmel ATmega 128 microprocessor. It is supplemented by 128 KB of external SRAM and utilizes the 4-channel, 16-bit ADS8341 ADC for data acquisition. Narada's wireless communication interface consists of Chipcon CC2420 IEEE 802.15.4 compliant transceiver, which makes it an extremely versatile unit for developing large-scale WSNs. This prototype is powered by a constant DC supply voltage between 7 and 9 volts, and has an operational life expectancy of approximately 48 hours with 6 AA batteries, given constant communication and data analysis demands.

We use a testbed of 12 Narada sensor nodes deployed in a corridor in the Electrical Engineering and Computer Science building at the University of Michigan. Each Narada wireless sensor is programmed with DAA algorithm, and asked to autonomously form computational clusters with varying values of n. The root of the tree is randomly selected in each experiment. In a manner similar to [36], the weight of an edge e is set to $w_e = \frac{1-p_{CF}}{1+e^{-0.4(40+RSSI)}}$[6], where $RSSI$ is the radio signal strength indicator reported by the radio and p_{CF} is the probability that a communication link with perfect RSSI fails due to unforeseen circumstances and is empirically measured to be equal to 0.1 for the Narada platform.

The objective of our experiment is to study the time it takes to construct the data collection tree using DAA, as well as the cluster sizes and the corresponding sensing cycles as a function of n in a real-world setting. Figures 8(a) and 8(b) plot the time it takes to construct the tree as a function of n and $|V|$ respectively[7]. We see that this time only depends on the size of the network. Figure 8(c) shows the data collection trees constructed for $n = 3$ and $n = 5$, respectively. To summarize, the implementation and the experimental results verify the feasibility of DAA on a real SHM sensor platform.

VII. Related Work

In this section, we review prior works most relevant to this paper, in the areas of in-network computation, clustering, and functional decomposition for SHM sensor networks.

A. In-network Computation

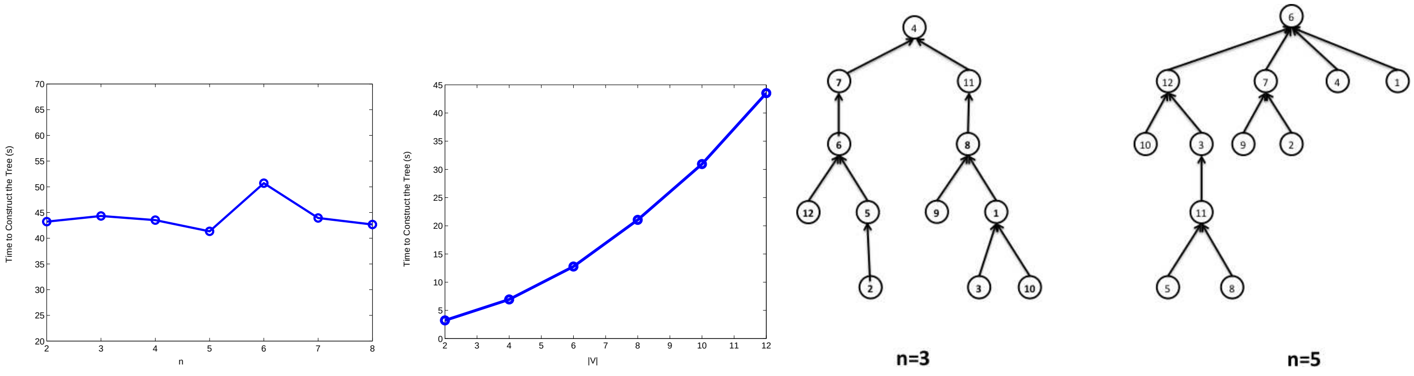
Prior works that study the communication structure for the purpose of in-network computation generally fall into one of three categories.

1) Delivering Equivalent Representation of Sensor Data: Within the first category, there is a body of literature that focuses on how to represent and deliver sensor data (in its original or an equivalent form) in as few bits as possible through well designed communication structures, so that a certain function $f(x_1, x_2, \cdots, x_n)$, where x_i originates from sensor i, may be computed (near) error-free at a central location, see e.g., [8]–[10]. The emphasis here is primarily on the presentation and delivery of the set of data $\{x_i\}$, rather than the actual computation of $f()$ within the network. For instance, in [8], [9] the class of *symmetric functions* is considered, i.e., $f()$ is invariant to any permutation of $\{x_i\}$. It is observed that due to the symmetry, for the purpose of computing $f()$ an equivalent representation of the data is its histogram, provided that x_i takes values from a finite set. Specifically, if x_i can take on D different values, then representing the histogram over n sensor inputs/measurements $x_1, x_2, \cdots, x_n$ takes a total of $D \log n$ bits (as opposed to directly representing the data which takes a total of $n \log D$ bits), which becomes a very attractive way of delivering equivalent input to $f()$ when the network (i.e., n) is large. As a result, it is shown in [8], [9] that the histograms can be collected at a rate of $\frac{1}{\log n}$ along a suitably constructed tree. Similar results are also available for sub-classes of symmetric functions, including the class of type-sensitive functions and type-threshold functions, as well as when the channels are noisy [8]–[10].

An obvious advantage of the model used in the above studies lies in its generality; it applies to any symmetric function $f()$ (interestingly the centralized SVD computation is a symmetric operation by the above definition). On the

[6] This weight is used as Narada provides access to the RSSI information. If the RSSI information were not available, we would have to collect and maintain loss rates and either use $w_e = \frac{1}{1-l_e}$ as before or some variant.

[7] Since each instance of the experiment is conducted at a different time and the environment is dynamic, the weights are recomputed for each instance.



(a)

(b)

(c)

Fig. 8. (a) Time to construct the tree vs n; $|V| = 12$. (b) Time to construct the tree vs $|V|$; $n = 4$. (c) Data collection trees for $n = 3$ and $n = 5$.

other hand, this approach does not actually compute $f()$ within the network (to do so one will need to exploit additional features of $f()$). This is the main difference between our approach and those cited above. As a result of this difference, in our case the data that ultimately reaches the base station is no longer an equivalent representation of the original data but rather the computed output of $f()$. In particular, in the case of SVD computation, a single measurement x_i is of size R bits (the FFT), which can take on $D = 2^R$ possible values. If we use the histogram approach above, then the total number of bits reaching the base station would be $2^R \log n$ (or $n \log 2^R = nR$ if we encode directly the values and not the histogram). In contrast, under our distributed computation the total number of bits reaching the base station is nr, $r \ll R$, which is significantly less, unless n is very large compared to R (note that this is only calculating the amount delivered to the base station; there are also savings in the amount of data transmitted within the network). Specifically, for networks like the ones we study in Section VI, the ratio of the number of bits transmitted over the network by the histogram approach and the lower bound is more than 10^{2400}.

2) Information Theoretic Formulations: Within the second category, there is a rich literature on information theoretic formations of in-network data processing. In many cases, the goal is to deliver an (near) equivalent representation of the original data set $\{x_i\}$ to a central location by using a combination of data compression and communication structure (data collection) design, see e.g., the classical distributed source coding [38], [39] and applied to sensor networks [26], [40]–[45]. A typical application of these methods is the reconstruction of a field image. In other cases, the goal is to compute a function $f()$ over correlated measurements $\{x_i\}$, see e.g., [46], [47]. In both cases, the emphasis is on the exploitation of the correlation structure among measurements to design good encoders and decoders at successive sensor nodes as data is being collected, and/or design good communication and data collection structures that work well with the encoding.

It is possible for us to also exploit correlation to further reduce the amount of data transmitted in the SVD computation. In particular, one could encode the FFTs and the computed mode shapes (eigenvectors) using fewer number of bits. This can be applied directly on top of the communication structure derived in the paper; thus the two are orthogonal. A more sophisticated approach can take correlation into account in constructing the data collection tree, e.g., it may be beneficial to send highly correlated FFTs to the same node for SVD computation, in the hope that the computed eigenvectors may also be correlated so fewer number of bits are needed to encode them for further transmission. Both approaches require the knowledge of the underlying correlation structure and is out of the scope of the present paper; they are however interesting directions of future research.

3) Distributed Computation: A third class of studies focuses on deriving explicit, distributed computational procedures to be performed over a network; this category is the closest in spirit to what we presented in this paper. A prime example is the family of random gossip algorithms, see e.g., [3]–[5], [48] and the references therein. There are also various deterministic algorithms, see e.g., [1], [2] for query aggregation and processing, [6] for complexity analysis, and [7] for the computation of certain Boolean functions. A common feature of these studies and the approaches developed therein is that in most cases the computation is limited to relatively simple functions like max, min, averages, and sums. These obviously are not sufficient to represent the many complex computational requirements demanded by practical engineering applications like the one studied in this paper.

B. Clustering

In addition to the literature on in-network data processing, another area very relevant to the work presented here is clustering. Clustering algorithms have been developed for a variety of purposes, see e.g., [49], [50] for routing in ad hoc networks, [51]–[53] for the energy efficient operation and data collection in wireless sensor networks, among others; see also a survey [54] and the references therein. Driven by different goals, different clustering algorithms vary in their constructions, and we do not know of any existing clustering algorithm that applies to the problem considered in this paper. This is primarily due to the combination of the delay constraint (which limits the size of a cluster) and the unique combinability constraint (which requires the clusters to overlap in a specific way). Most prior work on clustering does not explicitly require overlapping between clusters, with the notable exception of [55]. In [55], a clustering algorithm is presented to construct overlapping clusters that cover all nodes and where all cluster heads are connected. However, the overlapping condition in this case only requires that one cluster must share a certain number of nodes with *at least* another cluster; as a result, there need not be any overlap between two sets of clusters that overlap within their respective sets. This is quite different from the overlapping condition required in our SVD computation, where one cluster is required to either directly or indirectly (through a common, overlapping third cluster) overlap with *all* other clusters. The resulting clustering structures are also quite different.

C. Functional Decomposition for SHM

Finally, the studies on distributed computation of different SHM algorithms [12], [17], [56] focus chiefly on the correctness of the functional decomposition and do not study the routing problem. They assume a fully connected mesh of sensor nodes and impose an arbitrary communication structure to demonstrate the advantages of distributed computation. To our knowledge, only one, [36], studies the determination of a routing structure for a given topology; however, the formulation in this paper does not have a clearly defined optimization objective and the resulting algorithm has no performance guarantee. In contrast, our work is more general and systematic with a clearly defined performance objective, and all the approximation algorithms proposed have proven approximation factors.

VIII. Conclusion

This paper studies the problem of networked computation within the context of wireless sensor networks used for structural health monitoring. It presents centralized ILPs and distributed approximation algorithms to derive optimal communication structures for the distributed computation of SVD. Both simulations and implementations are used to evaluate their performance. Our results demonstrate the advantage of in-network computation as it significantly reduces the amount of data transmitted over the network.

There are a number of open problems we are interested in pursuing. One concerns combining this distributed computation approach with the compression of data, as discussed in Section VII. A second problem is to seek automated procedures to decompose a given computational task into elements/operators at the right level granularity. Our present approach relies on such a decomposition being available. Such an automated decomposition procedure would make the overall distribution computation framework much more general and application-independent.

References

[1] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, “TAG: a Tiny AGgregation service for ad-hoc sensor networks,” in *Proc. of OSDI*, 2002.

[2] B. Patt-Shamir, “A note on efficient aggregate queries in sensor networks,” in *Proc. of ACM PODC*, 2004.

[3] S. Kashyap, S. Deb, K. Naidu, and R. Rastogi, “Efficient gossip-based aggregate computation,” in *Proc. of ACM PODS*, 2006.

[4] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in *Proc. IEEE Symposium of Foundations of Computer Science*, 2003.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Gossip algorithms: design, analysis and applications,” in *Proc. of IEEE INFOCOM*, 2005.

[6] X. Li, Y. Wang, and Y. Wang, “Complexity of convergecast and data selection for wireless sensor networks,” Illinois Institute of Technology, Tech. Rep., 2009.

[7] H. Kowshik and P. Kumar, “Optimal strategies for computing symmetric Boolean functions in collocated networks,” in *Proc. of IEEE Information Theory Workshop*, 2010.

[8] A. Giridhar and P. R. Kumar, “Towards a Theory of In-Network Computation in Wireless Sensor Networks,” *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, 2006.

[9] ——, “Computing and Communication Functions Over Sensor Networks,” *IEEE J. Selected Areas in Communications (JSAC)*, vol. 23, no. 4, pp. 755–764, 2005.

[10] L. Ying, R. Srikant, and G. Dullerud, “Distributed Symmetric Function Computation in Noisy Wireless Sensor Networks,” *IEEE Trans. on Information Theory*, vol. 53, no. 12, 2007.

[11] A. Swartz, D. Jung, J. Lynch, Y. Wang, D. Shi, and M. Flynn, “Design of a Wireless Sensor for Scalable Distributed In-Network Computation in a Structural Health Monitoring System,” in *Proc. of the 5th International Workshop on Structural Health Monitoring*, 2005.

[12] A. T. Zimmerman and J. P. Lynch, “Automated Damage Estimation in Wireless Sensing Networks using Parallelized Model Updating,” in *Proc. of International Workshop on Structural Health Monitoring*, 2007.

[13] G. Hackmann, W. Guo, G. Yan, C. Lu, and S. Dyke, “Cyber-Physical Codesign of Distributed Structural Health Monitoring With Wireless Sensor Networks,” in *Proc. of International Conference on Cyber-Physical Systems (ICCPS)*, 2010.

[14] S. Fang and R. Perera, “Power mode shapes for early damage detection in linear structures,” *Journal of Sound and Vibration*, vol. 324, no. 1-2, pp. 40–56, 2009.

[15] Z. Ismail, H. Razak, and A. Rahman, “Determination of damage location in RC beams using mode shape derivatives,” *Engineering Structures*, vol. 28, no. 11, pp. 1566–1573, 2006.

[16] E. Clayton, B. Koh, G. Xing, C. Fok, S. Dyke, and C. Lu, “Damage detection and correlation-based localization using wireless mote sensors,” in *IEEE International Symposium on Intelligent Control*, 2005.

[17] A. T. Zimmerman, M. Shiraishi, R. A. Swartz, and J. P. Lynch, “Automated Modal Parameter Estimation by Parallel Processing within Wireless Monitoring Systems,” *Journal of Infrastructure Systems*, vol. 14, no. 1, pp. 102–113, 2008.

[18] D. Friedlander, C. Griffin, N. Jacobson, S. Phoha, and R. Brooks, “Dynamic agent classification and tracking using an ad hoc mobile acoustic sensor network,” *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 4, pp. 371–377, 2003.

[19] J. Gupchup, R. Burns, A. Terzis, and A. Szalay, “Model-Based Event Detection in Wireless Sensor Networks,” in *Proc. of Workshop on Data Sharing and Interoperability on the World-Wide Sensor Web*, 2007.

[20] L. Balzano and R. Nowak, “Blind Calibration of Sensor Networks,” in *Proc. of IPSN*, 2007.

[21] G. Derveaux, G. Papanicolaou, and C. Tsogka, “Time reversal imaging for sensor networks with optimal compensation in time,” *The Journal of the Acoustical Society of America*, vol. 121, no. 4, pp. 2071–2085, 2007.

[22] V. Raykar, I. Kozintsev, and R. Lienhart, “Position Calibration of Microphones and Loudspeakers in Distributed Computing Platforms,” *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, 2005.

[23] O. Durmaz-Incel and B. Krishnamachari, “Enhancing the Data Collection Rate of Tree-Based Aggregation in Wireless Sensor Networks,” in *Proc. of IEEE SECON*, 2008.

[24] B. Brinkman and M. Helmick, “Degree-constrained minimum latency trees are apx-hard,” Miami University, Tech. Rep. 2008-03-25, 2008.

[25] M. Helmick and F. Annexstein, “Depth-Latency Tradeoffs in Multicast Tree Algorithms,” in *Proc. of The IEEE 21st International Conference on Advanced Information Networking and Applications*, 2007.

[26] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer, “Network correlated data gathering with explicit communication: NP-completeness and algorithms,” *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 41–54, 2006.

[27] D. Ewins, *Modal testing: Theory and practice*, 2nd ed. Research Studies Press Ltd.

[28] O. Salawu, “Detection of structural damage through changes in frequency: A review,” *Engineering Structures*, vol. 19, no. 9, pp. 718–723, 1997.

[29] R. Brincker, L. Zhang, and P. Andersen, “Modal identification of output-only systems using frequence domain decomposition,” *Smart Materials and Structures*, vol. 10, no. 3, pp. 441–445, 2001.

[30] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, “Wishbone: Profile-based Partitioning for Sensornet Applications,” in *Proc. of NSDI*, 2009.

[31] A. Jindal and M. Liu, “Networked computing for structural health monitoring,” University of Michigan, Tech. Rep. CSPL-2010-01, 2010.

[32] D. Aguuayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, “Link-Level Measurements from an 802.11b Mesh Network,” in *Proc. of ACM SIGCOMM*, 2004.

[33] J. Zhao and R. Govindan, “Understanding packet delivery performance in dense wireless sensor networks,” in *Proc. of ACM SenSys*, 2003.

[34] W. Wu, H. Du, X. Jia, Y. Li, and S.-H. Huang, “Minimum Connected Dominating Sets and Maximal Independent Sets in Unit Disk Graphs,” *Theoretical Computer Science*, vol. 352, no. 1, pp. 1–7, 2006.

[35] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection Tree Protocol,” in *Proc. of ACM SenSys*, 2009.

[36] A. Zimmerman and J. Lynch, “Market-based frequency domain decomposition for automated mode shape estimation in wireless sensor networks,” *Journal of Structural Control and Health Monitoring*, 2010.

[37] “ILOG CPLEX 11.0,” http://www.ilog.com/products/cplex.

[38] D. Slepian and J. Wolf, “Noiseless Coding of Correlated Information Sources,” *IEEE Trans. on Information Theory*, vol. 19, no. 4, pp. 471–480, July 1973.

[39] S. S. Pradhan and K. Ramchandran, “Distributed source coding: Symmetric rates and applications to sensor networks,” in *Proc. IEEE Data Compression Conference (DCC)*, March 2000.

[40] Y. Zhu, K. Sundaresan, and R. Sivakumar, “Practical Limits on Achievable Energy Improvements and Useable Delay Tolerance in Correlation Aware Data Gathering in Wireless Sensor Networks,” in *Proc. of IEEE SECON*, 2005.

[41] S. Pattem, B. Krishnmachari, and R. Govindan, “The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks,” in *Proc. of IPSN*, 2004.

[42] E. J. Duarte-Melo, M. Liu, and A. Misra, “A computational approach to the joint design of distributed data compression and data dissemination in a data-gathering wireless sensor network,” in *Proc. of the Allerton Conference on Control, Communications and Computing*, 2003.

[43] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *ACM/IEEE Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2002.

[44] A. Boulis, S. Ganeriwal, and M. Srivastava, “Aggregation in sensor networks: An energy-accuracy trade-off,” in *Proc. of the First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[45] L. Galluccio, S. Palazzo, and A. Campbell, “Efficient data aggregation in wireless sensor networks: An entropy-driven analysis,” in *Proc. of the IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2008.

[46] H. Kowshik and P. Kumar, “Zero-error function computation in sensor networks,” in *Proc. of CDC*, 2009.

[47] S. Feizi and M. Medard, “When do only sources need to compute? On functional compression in tree networks,” in *Proc. of the Allerton Conference on Control, Communications and Computing*, 2009.

[48] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized Gossip Algorithms,” *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[49] C. Lin and M. Gerla, “Adaptive Clustering for Mobile Wireless Networks,” *IEEE J. Selected Areas in Communications*, vol. 15, pp. 1265–1275, September 1997.

[50] A. McDonald and T. Znati, “A Mobility Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks,” *IEEE J. Selected Areas in Communications*, vol. 17, no. 8, pp. 1466–1487, August 1999.

[51] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the Hawaii International Conference on System Sciences*, 2000.

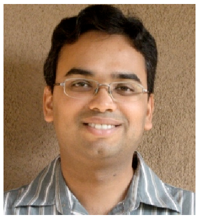
[52] F. Ye, H. Luo, J. Chung, S. Lu, and L. Zhang, “A Two-Tier Data Dissemination Protocol for Large-Scale Wireless Sensor Networks,” in *Proc. ACM MobiCom*, 2002.

[53] E. J. Duarte-Melo and M. Liu, “Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks,” in *Proc. IEEE Globecom*, 2002.

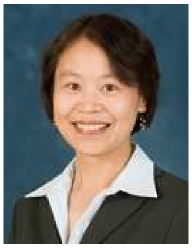
[54] A. Abbasi and M. Younis, “A Survey on Clustering Algorithms for Wireless Sensor Networks,” *J. Computer Communication, special issue on network coverage and routing schemes for wireless sensor networks*, vol. 30, pp. 2826–2841, 2007.

[55] M. Youssef, A. Youssef, and M. Younis, “Overlapping Multi-hop Clustering for Wireless Sensor Networks,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, no. 12, December 2009.

[56] A. Zimmerman and J. Lynch, “A Parallel Simulated Annealing Architecture for Model Updating in Wireless Sensor Networks,” *IEEE Sensors Journal*, vol. 9, no. 11, pp. 1503–1510, 2009.



Apoorva Jindal is a Member of Technical Staff at Juniper Networks. He received his B.Tech. degree from Indian Institute of Technology at Kanpur and his MS and PhD degrees from the University of Southern California. He completed his post-doctoral research at the University of Michigan, Ann Arbor. His research interests include performance analysis and design of protocols for multi-hop wireless networks.



Mingyan Liu Mingyan Liu (M’00 / ACM’01) received her B.Sc. degree in electrical engineering in 1995 from the Nanjing University of Aero. and Astro., Nanjing, China, M.Sc. degree in systems engineering and Ph.D. Degree in electrical engineering from the University of Maryland, College Park, in 1997 and 2000, respectively. She joined the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, in September 2000, where she is currently an Associate Professor. Her research interests are in optimal resource allocation, performance modeling and analysis, and energy efficient design of wireless, mobile ad hoc, and sensor networks. She is the recipient of the 2002 NSF CAREER Award, and the University of Michigan Elizabeth C. Crosby Research Award in 2003. She serves on the editorial board of IEEE/ACM Trans. Networking, IEEE Trans. Mobile Computing, and ACM Trans. Sensor Networks.