Purdue University
Purdue e-Pubs

Computer Science Technical Reports | Department of Computer Science

2007

Scalable Data Collection in Dense Wireless Sensor Networks

Asad Awan

Suresh Jagannathan
Purdue University, suresh@cs.purdue.edu

Ananth Y. Grama
Purdue University, ayg@cs.purdue.edu

Report Number:
07-018

Awan, Asad; Jagannathan, Suresh; and Grama, Ananth Y., "Scalable Data Collection in Dense Wireless Sensor Networks" (2007). *Computer Science Technical Reports.* Paper 1682.
http://docs.lib.purdue.edu/cstech/1682

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

SCALABLE DATA COLLECTION IN DENSE WIRELESS SENSOR NETWORKS

Asad Awan
Suresh Jagannathan
Ananth Grama

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD TR #07-018
July 2007

Scalable Data Collection in Dense Wireless Sensor Networks

Asad Awan Suresh Jagannathan Ananth Grama
Department of Computer Science, Purdue University, West Lafayette, IN 47906
awan,suresh,ayg@cs.purdue.edu

Abstract—
Applications of sensor networks often necessitate fine-grained data collection, requiring dense deployment of sensors, with associated high data rates. Such deployment and application scenarios impose significant constraints on aggregate network data rate, resource utilization, and robustness. Effective protocols for supporting such data transfers are critical for dense sensing applications. These protocols often rely on spatio-temporal correlations in sensor data to achieve in-network data compression. The message complexity of these schemes is generally lower bounded by n, for a network with n sensors, since correlation is not collocated with sensing. Consequently, as the number of nodes and network density increase, these protocols become increasingly inefficient. We present here a novel protocol, called SNP, for fine-grained data collection, which requires approximately $O(n-R)$ messages, where R, a measure of redundancy in sensed data generally increases with density. SNP uses spatio-temporal correlations to near-optimally compress data at the source, reducing network traffic and power consumption. We present a comprehensive information theoretic basis for SNP and establish its superior performance in comparison to existing approaches. We support our results with a comprehensive experimental evaluation of the performance of SNP in a real-world sensor network testbed.

I. Introduction

With the goal of building a comprehensive systems infrastructure for scalable sensor networks, we have developed a distributed data-driven processing runtime and an associated flexible programming environment [1]. Using this system, we have deployed a large-scale sensornet testbed for structural monitoring at the Bowen Labs for Structural Engineering [1], [2]. Our experience with this testbed, and the experience reported by other real-world deployments [1], [3]–[6] reveal that fine-grained data collection is critical to several applications.

Fine-grained data collection typically requires data from *all* sensors in the network. This is motivated by the fact that data processing operations on sensor data are often too computationally intensive for resource constrained sensor nodes. For example, in structural engineering time domain sensor data needs to be converted to frequency domain spectrograms[1], which requires Fourier Transform over long data sequences. Sensor nodes do not have the processing or memory capabilities to perform such operations. Furthermore, physical phenomenon models, which provide the basis for in-network processing, are often not known a-priori. In fact, a key use of sensornets is to provide empirical measurements that are used to build or verify scientific models [7].

Supporting fine-grained data collection in dense sensornets is rendered challenging by the scarce network and energy resources at sensor nodes. Increasing network density, and hence data rate, results in rapid degradation in wireless neighborhood network capacity [8]. Similarly, increasing data rate results in high energy consumption [9]. Consequently, reducing data traffic is critical to both throughput and longevity of the network. In-network compression using spatio-temporal correlations is a viable application-independent technique for fine-grained data collection.

Several existing protocols (e.g., [7], [10]) exploit spatio-temporal correlations by partitioning the network into disjoint clusters. Data from each node in the cluster is routed to a cluster representative, which then performs correlation driven compression. Compressed data from cluster representatives is then relayed to the sink. The resulting message overhead of such protocols is typically:

$$O(n \cdot k_h) + O(n_c \cdot k_s). \quad (1)$$

Here, n is the number of nodes in the network, k_c is the average number of hops from a source to the cluster representative, n_c is the average number of messages with compressed data, and k_s is the average number of hops from the cluster representatives to the sink. The first term in this overhead results from data sharing for correlation and the second term is the actual compressed data. As the density and number of nodes in the network increase, the first term dominates overhead. Furthermore, while such protocols reduce network congestion near the sink, congestion near the sources increases rapidly. Figure 1, illustrates the decrease in network throughput as the number of nodes generating data at a constant rate in a wireless neighborhood near the sources increases. Improvements to this approach are presented in [10], where data is compressed along tree routing paths to the cluster representative, thus reducing the number of messages. However, the $O(n)$ term still exists because over half the nodes in a tree are leaves. Pattem et al. [11] provide an excellent analysis of data compression using the network partitioning approach. However, in their analysis they assume that the routing topology within a partition is linear. Compression along a linear topology overcomes the $O(n)$ lower bound on overhead. However, this increases the path length offsetting the benefits and increasing load imbalance (nodes near the

[1] Spectrograms are used to observe vibration modes of building structures over the passage of time.

cluster representative must deliver compressed packets from all predecessors).

In this paper, we present a novel protocol called spatial neighborhood protocol (SNP) for exploiting spatio-temporal correlations in dense sensornets. The critical differentiating aspect of SNP is that correlation-based compression is collocated with sensing. Each node independently determines whether it should share its data based on data, communicated using radio broadcast, from other nodes in its spatial neighborhood[2]. Given the assumption that correlations are likely to be spatially localized, a node requires data only from a few other nodes in its neighborhood to (near) optimally compress its data. In this manner, only a subset of the nodes in the network need to share their data, while all nodes can achieve near optimal compression based on this shared data. As a result the overhead of SNP is:

$$O(n - R) + O(n_c \cdot k). \tag{2}$$

Here, k is the average number of hops from a source to the sink, n_c is the number of messages containing compressed data, and R is a measure of redundancy in the network. The increasing redundancy, R, as a function of node density, is key to the scalability of SNP. We show that SNP achieves near optimal compression without the $O(n)$ overhead for computing correlations.

The key contributions of this paper are as follows.

- We present an information theoretic model, called the SN model, for joint network compression, which exploits both the correlation of data in the sensor network and redundancy due to network density. We show that the SN model maximizes compression without incurring much overhead of data-sharing for correlations. We also develop a model for the partitioning-based technique used in prior work, and show that SN delivers higher compression rates and lower compression overhead. (Section II)
- We develop SNP, a practical distributed protocol that implements the SN model, and inherits its performance properties. We discuss the architecture of SNP and demonstrate its robustness. (Section III)
- We present a comprehensive evaluation of SNP on a real-world testbed, and using simulations. We demonstrate the superior performance of SNP in comparison to existing protocols and characterize its behavior over a wide operational range. (Section IV)

II. Information Theoretic Underpinnings

In this section we develop the spatial neighborhood (SN) model (Section II-D), which forms the basis for the SNP protocol. We also present the partitioning model (PT), which forms the basis of existing protocols (Section II-E). We show that the compression overhead of the SN model is much less than that of the PT model. We also show that the compression rate of the SN model is better than that of the PT model.

[2]In a wireless network, a message broadcast transmission can be received by all nodes within the radio range of the broadcasting node.

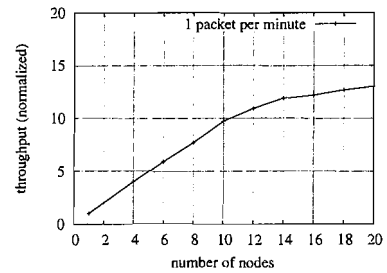

Fig. 1. Throughput of the network decreases rapidly as the number of nodes in a neighborhood increases.

In demonstrating the near-optimality of compression overhead and rate for the SN model, we establish it as the basis for data gathering protocols in sensornets.

A. Preliminaries

Shannon entropy, or simply entropy of a random variable X, denoted by $\mathcal{H}(X)$, is a measure of the uncertainty (randomness) of a variable. If X is a random variable whose values are drawn from the probability distribution of the data generated by a sensor node, then $\mathcal{H}(X)$ denotes the entropy of the source. To model a network with n nodes, we define $\mathbf{N}$ as a set of random variables. $X_i \in \mathbf{N}$ represents the random variable associated with the data originating at node i and $\mathcal{H}(X_i)$ represents its entropy. Joint entropy of multiple sources corresponds to the minimum amount of information that can be used to reconstruct data from each source. Notationally, we represent joint entropy as $\mathcal{H}(X_1, X_2, \ldots, X_n)$, or simply $\mathcal{H}(\mathbf{N})$. Jointly coding data from correlated sources results in transmission of $\mathcal{H}(\mathbf{N})$ bits of information instead of $\sum_{i=1}^{n} \mathcal{H}(X_i)$. Note that $\mathcal{H}(\mathbf{N}) < \sum_{i=1}^{n} \mathcal{H}(X_i)$, in the existence of any data correlations. Temporal correlations further reduce data since only $\mathcal{H}(\mathbf{N}^t|\mathbf{N}^{t-1}, \ldots)$ (i.e., conditional entropy of data at time t, given data from previous time steps) bits of information need to be transmitted, instead of $\mathcal{H}(\mathbf{N})$. In this section we focus primarily on spatial correlations. Temporal correlations can be computed from history buffers at source nodes.

B. Joint Entropy of Two Sources

The joint entropy of two source nodes is expressed as:

$$\begin{aligned} \mathcal{H}(X_1, X_2) &= \mathcal{H}(X_1) + \mathcal{H}(X_2|X_1) \\ &= \mathcal{H}(X_1) + \mathcal{H}(X_2) - \mathcal{I}(X_1, X_2) \\ &\leq \mathcal{H}(X_1) + \mathcal{H}(X_2). \end{aligned} \tag{3}$$

Here, $\mathcal{H}(X_2|X_1)$ is the conditional entropy of X_2 given X_1 and $\mathcal{I}(X_1, X_2)$ is the mutual information between the two random variables X_1 and X_2. Mutual information is a quantity that measures the correlation between two random variables.

As an example, consider a two-node system in which data transmitted by node 2 can be deterministically calculated using data from node 1. In this case, $\mathcal{H}(X_1, X_2) = \mathcal{H}(X_1)$, and only data from source 1 needs to be transmitted. Conversely, if no correlations exist (i.e., there is no mutual information), then $\mathcal{H}(X_1, X_2) = \mathcal{H}(X_1) + \mathcal{H}(X_2)$, and data from both sources must be transmitted. In sensor networks, data from

one node is often correlated with nearby sources. Therefore, $\mathcal{H}(X_1, X_2) < \mathcal{H}(X_1) + \mathcal{H}(X_2)$. In such cases, only uncorrelated bits of X_2 (called error bits or ϵ) need to be transmitted to **exactly** reconstruct the data of node 2 using the data from node 1.

Mutual information in sensor networks quantifies correlations, which typically result from spatial locality of nodes in the network. Based on this spatial locality relation, mutual information for a pair of nodes can be expressed as:

$$\mathcal{I}(X_1, X_2) = \mathcal{D}(d) \cdot min(\mathcal{H}(X_1), \mathcal{H}(X_2)). \quad (4)$$

Here, $\mathcal{D}(d)$, is a correlation scaling function defined in terms of the distance d between nodes 1 and 2, and takes values in the range $0 \leq \mathcal{D}(d) \leq 1$. The lower of the two source entropies provides a trivial bound on the maximum mutual information. The exact characteristics of the function $\mathcal{D}(d)$ depend on applications, and deployments within specific applications of sensornets. In typical applications, though, it is reasonable to expect that correlations are inversely related to proximity. We formally state this as:

Lemma 2.1: Monotonicity of $\mathcal{D}(d)$:

$$\mathcal{D}(d) \geq \mathcal{D}(d') \quad \text{iff} \quad d \leq d'.$$

We have, thus far, discussed the abstract correlation scaling function, $\mathcal{D}(\cdot)$ in terms of spatial distance, i.e., as $\mathcal{D}(d)$. However, in different deployments the scaling function may be defined in terms of other parameters, leading to the generalization $\mathcal{D}(\mathbf{R}_{i,j})$. Here, $\mathbf{R}_{i,j}$ represents the parameter set. Our model does not impose any constraints on these parameters, as long as the generated function meets the following properties: (i) the range of the function should be $0 \leq \mathcal{D}(\mathbf{R}_{i,j}) \leq 1$; and (ii) the function should be a monotonic (either increasing or decreasing) function of its parameters.

Example 2.1: In our structural health monitoring setup, sensors are attached to the frame. In structural response, there is a high correlation between sensors on the same structural element, but there is little correlation between the sensors on different elements, even if the spatial distance between them is small. To account for this, we define a correlation function $\mathcal{D}(d, p)$, where d is the distance between the two sensors and p is variable set to 0 if the nodes are on the **same** element, and 1 otherwise. $\mathcal{D}(d, p)$ satisfies the properties of $\mathcal{D}(\cdot)$ since it is monotonic in d and p.

The above example shows how application specific correlation scaling functions can be designed. In the rest of the paper we use $\mathcal{D}(d)$ as a concrete instance of $\mathcal{D}(\cdot)$. An analogy of the results can be easily constructed for other instances of $\mathcal{D}(\cdot)$.

C. Joint Entropy of N Sources

A precise expression for (optimal) joint entropy must incorporate application features. To provide an application independent description, we define an approximation to the optimal joint entropy of the sensornet in terms of pairwise mutual information. This approximation suffices to show that the spatial neighborhood model, which is the basis for SNP, achieves better compression than existing approaches. The joint entropy of the network is represented by $\mathcal{H}(\mathbf{N})$. Correlation-based compression induces an implicit ordering of sources. This is because data from a node i can only be coded with respect to the data from a node j, which itself is not encoded with respect to i. Therefore, an iterative construction[3] is required to evaluate H_n, our approximation to $\mathcal{H}(\mathbf{N})$:

Procedure 2.1: Evaluating H_n.

1) Initialize: Let, $\mathbf{S}$ be the set of nodes $\{v_2, v_3, \ldots, v_n\}$ and $\mathbf{V}$ be the set of a single node $\{v_1\}$. We set $H_1 = \mathcal{H}(X_{v_1})$, where X_{v_1} is the random variable associated with data from node v_1.
2) Iterate: for $i = 2 \;\; to \;\; n$
   a) Select node v_k from $\mathbf{S}$ and v_j from $\mathbf{V}$ such that $\mathcal{I}(X_{v_k}, X_{v_j})$ is maximized.
   b) Set $H_i = H_{i-1} + (\mathcal{H}(X_{v_k}) - \mathcal{I}(X_{v_k}, X_{v_j}))$.
   c) Remove node v_k from $\mathbf{S}$ and add it to set $\mathbf{V}$.
3) H_n is the approximation of $\mathcal{H}(\mathbf{N})$.

In the above procedure, step 2.a induces the ordering required for coding nodes with respect to each other. By selecting a node v_k that is maximally correlated (maximum mutual information) to some node in $\mathbf{V}$, this step minimizes joint entropy of the iteratively growing set, $\mathbf{V}$. H_n is an approximation to $\mathcal{H}(\mathbf{N})$ since pair-wise mutual information does not capture the information that v_k can extract from (all) other nodes in the set $\mathbf{V}$. Therefore, in general $\mathcal{H}(\mathbf{N}) \leq H_n$.

As stated earlier, the task of a distributed in-network compression protocol is to support efficient sharing of data from sources in the network to enable joint encoding of correlated data. In the context of Procedure 2.1, node v_k must have access to the data from node v_j. Furthermore, it is desirable that this data sharing be independent of the underlying network layout and routing topology. Existing approaches share data between nodes by partitioning the network into disjoint clusters and compressing the data at the cluster representative. However, this process has high compression overhead (i.e., data sharing overhead). Furthermore, the compression performance is sensitive to the optimality of partitioning.

D. Spatial Neighborhood Model (SN)

The SN model is based on the following construction: let $\mathbf{S}$ be the set of n nodes in the network. For *each* node i, we define a spatial neighborhood set $\mathbf{S}_i^{r_i}$, which is a subset of $\mathbf{S}$ containing all nodes within distance r_i (except i itself). Here, r_i is called the correlation radius of node i. Corresponding to each set $\mathbf{S}_i^{r_i}$, we build a set of random variables $\mathbf{M}_i$, such that $\forall k \in \mathbf{S}_i^{r_i} : X_k \in \mathbf{M}_i$. From Equation 4 and Lemma 2.1, nodes that are close to a given node i have a high spatial correlation with i. Therefore, the value of r_i can be chosen such the set $\mathbf{S}_i^{r_i}$ contains all nodes whose mutual information w.r.t. i is above threshold c.

In the SN model, each node i receives messages from nodes in its spatial neighborhood set $\mathbf{S}_i^{r_i}$. Since a node k may be in several spatial neighborhood sets, it can communicate with

[3]This model is similar to the one presented in [11]. However, ours is a more general formulation.

all nodes i for which $k \in \mathbf{S}_i^{r_i}$ using a single radio broadcast message, assuming radio range exceeds r_i.

The construction of SN, thus far, implies that n broadcast messages are required, since each node must be in the spatial neighborhood set of at least one other node. However, due to redundancy in dense networks, we can prune the spatial neighborhood sets in such a way that a number of nodes (R) can be unaffiliated, i.e., are not in *any* set.

Definition 2.1: A node $k \in \mathbf{S}_i^{r_i}$ is redundant w.r.t. to $\mathbf{S}_i^{r_i}$ if there exists a node $j \in \mathbf{S}_i^{r_i}$ such that $\mathcal{I}(X_i, X_k) \approx \mathcal{I}(X_i, X_j)$ and the mutual information between k and j is high (i.e., $\mathcal{I}(X_k, X_j) > c$, for some threshold c).

To maximize R (and therefore, minimize message broadcast count), joint pruning of all the spatial neighborhood sets is needed. This is straightforward to achieve because, if the mutual information of two nodes, say k and j, is high, they are spatially close to each other (due to Equation 4). Therefore, the distance of node k and node j from another node i is approximately the same. Hence, $\mathcal{I}(X_i, X_k) \approx \mathcal{I}(X_i, X_j)$ for all other nodes i. Note that identifying R redundant nodes results in message reduction from n to $n - R$. This result is useful and important because redundancy typically increases with number of network nodes, implying that protocols based on the SN model scale well with increasing density.

Theorem 2.1: Redundancy (R) increases monotonically with network density.

Proof: Consider a pair of spatially proximate nodes k and j. From Equation 4 and Lemma 2.1, their mutual information is potentially high, i.e., $\mathcal{I}(X_k, X_j) > c$. Furthermore, as these nodes come closer (increasing density), they belong to the neighborhood sets of an increasing number of nodes together. It then follows from Equation 4 that $\mathcal{I}(X_i, X_k) \approx \mathcal{I}(X_i, X_j)$. Consequently, one of k or j can be removed from all spatial neighborhood sets (cf. Definition 2.1). It is easy to show that as the density of the network increases, the number of spatially proximate pairs increases linearly. One node from each such pair can be removed, if correlated, increasing R. If the network has uniform density this increase is linear as well. Note, though, that this relies on correlation (mutual information)– if there is no correlation, even with increasing density, R does not increase. ■

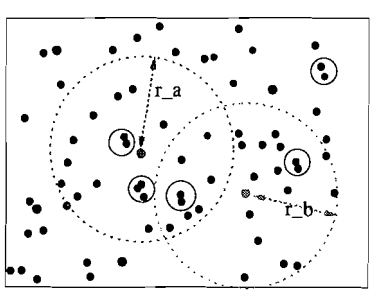

Fig. 2. Overview of the SN model.

Figure 2 illustrates the SN model for a sample network layout. Two nodes a and b (shaded red and green, respectively) and their correlation radius r_a and r_b are shown. Solid (black outline) circles mark a few of the redundant pairs. One such pair is in the intersection of the correlation radius of nodes a and b. One of these nodes need not broadcast its data, without affecting the compression rate of the nodes a and b.

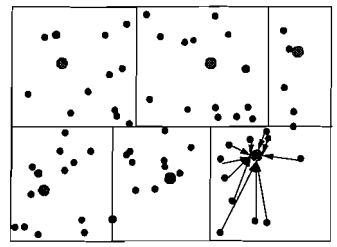

Fig. 3. Overview of the PT model.

We now show that the SN model achieves better compression than the bound quantified by H_n (cf. Section II-C).

Theorem 2.2: The spatial neighborhood model (SN) achieves joint entropy, $\leq H_n$.

Proof: This follows from the observation that the spatial neighborhood set of node i, $\mathbf{S}_i^{r_i}$, includes all nodes that have high mutual information w.r.t. node i (cf. Equation 4 and Lemma 2.1). Thus, it must include the node v_j from step 2.a of Procedure 2.1. Therefore, the SN model achieves joint entropy of $\leq H_n$. ■

An implication of the above theorem is that the SN model can achieve in-network compression such that at most H_n bits are transmitted to the sink. Let n_c denote the number of messages required to transmit H_n bits, and k be the average number of hops from the source to the sink (e.g., in a tree topology k is the height of the tree). Then the overhead of transmitting compressed data is $O(n_c \cdot k)$. As stated earlier, the overhead of data sharing in the network is $O(n - R)$. Therefore, we can derive the following theorem:

Theorem 2.3: The network overhead of the SN model is $O(n - R) + O(n_c \cdot k)$.

The key observation from this theorem is that the SN model scales well with increasing density of the network. This is because: (i) R increases with density, (ii) n_c decreases with density, and (iii) k remains approximately constant as the density increases.

E. Partitioning Model (PT)

An overview of the partitioning model for a random sensor network topology is presented in Figure 3. Here, the sensor network is partitioned into m disjoint clusters of neighboring nodes in the network (in the figure $m = 6$). A cluster representative (shaded nodes in the figure), chosen within each partition, is responsible for receiving data from all nodes in the partition (see lower right partition). At each cluster representative, all collected data is then jointly coded and the compressed information is relayed to the sink. To evaluate the resulting joint entropy we let $\mathbf{C}_i$ be the set of random variables associated with the nodes in the ith cluster. The joint entropy achieved by the PT model is therefore given by: $\mathcal{H}^{PT}(\mathbf{N}) = \sum_{i=1}^{m} \mathcal{H}(\mathbf{C}_i)$.

Theorem 2.4: $\mathcal{H}^{PT}(\mathbf{N}) \geq H_n$.

Proof: If the number of clusters $m = n$, then $\mathcal{H}^{PT}(\mathbf{N}) = \sum_{i=1}^{n}(\mathcal{H}(X_i)) \geq H_n$. If $m = 1$, clearly $\mathcal{H}^{PT} = H_n$. For any other value of m the entropy of each cluster can be found using Procedure 2.1 and the sum of these entropies is greater than or equal to H_n. ■

Corollary 2.1: The spatial neighborhood model (SN) achieves better compression rate than the partitioning model (PT) because $\mathcal{H}^{SN}(\mathbf{N}) \leq H_n \leq \mathcal{H}^{PT}(\mathbf{N})$.

Protocols based on the PT model (e.g., [7], [12], [13]) reduce the number of messages by pushing compression (or processing) of information into the network, i.e., to the cluster representatives. This decreases the network messages delivered to the sink. Unfortunately, such protocols still have a transmission overhead of $O(n \cdot k_h)$ because each node must necessarily transmit its data to the respective cluster representative. Here, k_h is the number of hops to the cluster representative. Thus, the compression overhead of the PT model is at least $O(n)$. Due to its construction, the PT model can not reduce this overhead to $O(n - R)$, which the SN model achieves. Therefore, in dense networks SN has significantly lower compression overhead than the PT model.

From Corollary 2.1, we see that the SN model achieves better compression than the PT model. Let, n_c be the number of messages required for transmitting the compressed data from the cluster representatives to the sink. Then, $O(n_c \cdot k_s)$ messages are required for transmitting data to the sink. Here, k_s is the average number of hops from the cluster representative to the sink. Recall that the SN model requires $O(n'_c \cdot k)$ messages, where $n'_c \leq n_c$ and in general $k > k_s$. However, as the density of the network increases, the $O(n)$ term dominates for PT, while $O(n - R)$ dominates for SN. Since R increases with density, the total overhead of SN is much lower. Therefore, the key aspect of the SN model is that it achieves a low compression overhead, while achieving similar compression rates as prior approaches.

It is worth noting that optimal partitioning of the network for PT is itself a hard problem. Finding an optimal partition with good load balance has exponential complexity in n, though approximate algorithms with polynomial complexity are possible [7]. Secondly, optimal partitioning may change with time resulting in network overhead for re-organizing the clusters in the network. In comparison, the SN model is self-organizing without extra overhead. Furthermore, the PT model places a high computation, communication, and memory burden on the cluster representative.

III. The SNP Protocol

SNP is a distributed and self-organizing protocol that efficiently implements the SN model, achieving high associated compression rates at low overheads. It is practical and can be implemented on lean sensor nodes.

a) Protocol Overview: In the SN model, each node needs data from its neighbors, using which it can compress (correlate) its own data. This data is communicated through broadcasts. Other, correlated nodes, suppress their own broadcasts in response. The key unresolved issue is to construct a symmetric distributed coding and decoding scheme. Specifically, if node i codes its data w.r.t. node j, then node j may not code its data w.r.t. node i. Furthermore, the sink should be aware that node i used data from node j for reconstruction. Clearly, an ordering based on which coding can take place is required. The task of SNP is to induce such an ordering, while conforming to the SN model. The ordering induced by SNP is based on spatial relationships (and consequently, likelihood of correlation) between nodes.

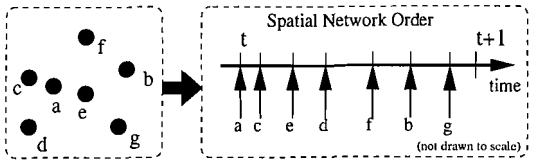

Fig. 4. Spatial neighborhood ordering in the SNP protocol. Node a initiates the ordering process.

b) Protocol Details.: SNP partitions time into intervals of user-defined epochs (based on the data rate). Within each epoch, a node broadcasts (or suppresses its communication) at an allocated time. This time-ordering of nodes in a spatial neighborhood can be established using several protocols. SNP designates a subset of spatially distant (with distance $\approx D$ between them) nodes that initiate this ordering process. In this step, the designated nodes broadcast their data and go to sleep until the next epoch. Upon receiving this broadcast from a designated node, each node time-orders itself based on its distance from the designated node. This is done by initializing a count-down timer at node i to $T_i = \alpha \times d_{i,j}^2 + \beta_i$. Here, $\alpha \propto epoch/D^2$, $d_{i,j}$ represents the distance between node i and designated node j, and β_i is calculated using the hash of node id of i to the space $(0, \alpha)$. Note that SNP does not depend on the exact measurement of distance. A relative measure that is monotonic w.r.t. distance suffices. If nodes do not have a GPS, radio signal strength can be used [14]. Node locations can also be hard-coded into node IDs. The hash term, β prevents collisions between nodes that may be the same distance from a designated node. If a node receives messages from two nodes with different distances from it, the node chooses the closer of the two to synchronize its timer.

Once all timers have been initialized, we have an induced time-ordering of nodes in a spatial neighborhood (Figure 4). We refer to this ordering as Spatial Neighborhood Node ordering (SNO). This technique for deriving SNO has several desirable features: (i) it is resilient to node failures and insertions, (ii) it provides relative synchronization of the nodes and hence has much lower overhead than absolute time division and synchronization protocols, (iii) it is independent of the radio range because nodes synchronize with messages from nearby neighbors. For the same reason it does not suffer from the hidden station problem, and (iv) it minimizes collisions in the network by providing a simple means of time division slotting (TDMA).

c) Prediction Functions.: A prediction function, $\mathcal{F}_\theta$, estimates the data at a node from data at correlated sources.

$$\hat{x}_i^t = \mathcal{F}_\theta(x_i^t | x_j^t, x_k^t, \ldots, x_i^{t-1}, x_j^{t-1}, \ldots).$$

Here, $\hat{x}_i^t$ is an estimate of x_i^t (the data at node i at time step t) computed from data at other nodes. Note that data from previous time steps (e.g., x_j^{t-1}) can also be used by the prediction function. The prediction error $|\epsilon|$ is given by $|x_i^t - \hat{x}_i^t|$. Higher correlation implies lower prediction error. Note that the prediction function has a model parameter θ_i for each node. These parameters are evaluated at the sink and transmitted to the nodes. Thus, the computationally intensive task of calculating parameters is performed at the sink, while the nodes use simple operations to predict data. θ_i can be updated at the sink if the correlations change. This technique has been used in prior systems as well [15], [16]. SNP is, itself, independent of the prediction function. The prediction function used in our implementation is discussed in Section IV-A.

d) Correlation Radius.: Instead of defining correlation radius in terms of distance, SNP keeps two sets of nodes, $PRED_i$ (predecessors in time ordering) and $SUCC_i$ (successors in time ordering) at each node, that serve the same practical purpose. These sets are constructed locally at each node. For example, for node e in Figure 4, $PRED_e = \{a, c\}$ and $SUCC_e = \{g, b, f, d\}$. Note that these sets are sorted in terms of the distance of nodes from node e. A node is allowed predict its data using data from the current time step from nodes in $PRED_i$ and data from the previous time steps using data from nodes $PRED_i \cup SUCC_i$. The number of nodes in these sets is γ_p (predecessors) and γ_s (successors). Large predecessor and successor sets improve compression, however, they also have associated memory overheads. In SNP, these parameters to be tunable by users. We show in our experiments that a small constant set size suffices in practice (Section IV-C).

e) Suppressing Data Broadcasts.: A node determines whether it must broadcast its data or not based on the value predicted using its predecessors (and successors from prior epochs). This results in a self-adjusting mechanism, with varying density and correlations. *Due to this broadcast suppression mechanism, SNP achieves scalability with increasing density, as with the SN model.* The above mechanism is implemented using two thresholds, δ_l and δ_h. If the prediction error $|\epsilon| < \delta_l$, the node does not broadcast its data. In subsequent epochs, the node continues to suppress communication of its data unless $|\epsilon| > \delta_h$. This *hysteresis* based thresholding results in stability across slight correlation changes. Stability is an important part of this decision process, since a change in the decision at a node can affect the $PRED$ and $SUCC$ sets of other nodes. Conversely, if the decision process is over-damped, the system can not adapt to changing correlations. We show using experiments in Section IV-D that this is not a major concern for SNP.

Note that, as nodes broadcast their data, nodes that might not have heard the SNO initiators can set their timers based on messages heard from their neighbors and, thus, find their position in the SNO ordering. This overcomes the hidden station problem.

f) Data Compression and Transmission.: Locally, each node i finds an estimate, $\hat{x}_i^t$, of its data x_i^t as:

$$\hat{x}_i^t = \mathcal{F}_\theta(x_i^t | \ PRED_i^t \ \cup SUCC_i^{t-1} \ \cup PRED_i^{t-1} \cup \ldots). \quad (5)$$

The prediction error is given by $\epsilon_i^t = x_i^t - \hat{x}_i^t$. Users can specify ϵ_m, the maximum error tolerance (which can be zero). If $|\epsilon_i^t| \leq \epsilon_m$, no data is transmitted, otherwise only ϵ_i^t (which uses fewer bits) needs to be transmitted to the sink. Since data is communicated in packets, sending a packet with a few bits will have high overhead. Consequently, we buffer the prediction errors from multiple time epochs until the buffer is large enough to offset the packet overhead. We also use a threshold $thresh$, so that if $|\epsilon_i^t| > thresh$, the sensor measurement is immediately transfered to the sink[4]. In this manner, *outlier* data is immediately transmitted to the sink, while well correlated data is transfered lazily. *By using the data from both its predecessor and successors for compression, SNP faithfully implements the SN model and achieves compression rates of the SN model.*

The final step of the SNP protocol runs at the sink, which reconstructs data from compressed values, i.e., ϵ_i^t, received from each node. For this, the sink must be able to execute the same prediction operation (Equation 5). Once the estimate, $\hat{x}_i^t$, is evaluated at the sink, the actual value can be computed using the compressed bits received from the node.

$$x_i^t = \hat{x}_i^t + \epsilon_i^t$$

Clearly, for the sink to apply the prediction operation it needs to know the $SUCC_i$ and $PRED_i$ sets of a node. Recall that data at node x_i^t is predicted using data from the same time step from its $PRED_i$ set or data from previous time steps from its $PRED_i \cup SUCC_i$ set. Thus, all data required to re-construct x_i^t is available at sink. Each node communicates its $PRED_i$ and $SUCC_i$ sets to the sink. This needs to be done only once, when the sets are first constructed. This amortizes, over time, the overhead of communicating these lists. Note that these sets are stable because the broadcast suppression mechanism (which affects the nodes that can be in the $SUCC$ and $PRED$ sets) is stable, as discussed earlier and demonstrated in our experiments in Section IV-D.

g) Resilience to Packet Losses.: Packet losses can disrupt prediction, since a packet (data) used for prediction at source may not have been received by the sink. Due to the use of spatial neighborhood ordering and data sharing, SNP minimizes packet losses from collisions and radio attenuation (due to spatial locality). Furthermore, the $PRED_i$ and $SUCC_i$ sets can be adapted so that nodes with repeated losses relative to the node i are removed from the sets.

h) Selecting SNO Initiators.: In our implementation we offload to the task of selecting the initiators to the sink, which knows the topology of the network. Note that the sink

[4]Data is transmitted to the sink using the underlying sensor network routing protocol, e.g., tree routing. Note that SNP is independent of this routing layer.

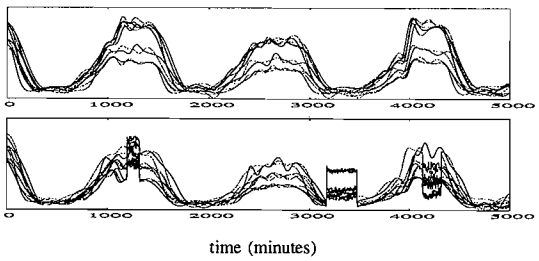

Fig. 5. Trace A (top) and Trace B (bottom), used in evaluating SNP. Each curve in a trace corresponds to the data from one node.

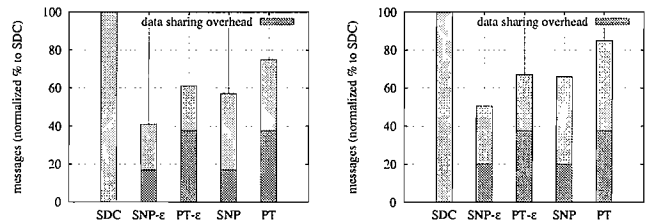

Fig. 6. Overall performance of in-network compression. Number of messages (normalized w.r.t. the number of messages in SDC) using different schemes (Trace A, left, and Trace B, right).

can accurately keep track of failures of such nodes because the initiator node always transmits data to the sink at the start of each epoch, which can be used as a heartbeat. In general randomized algorithms for selecting these nodes can easily be formulated. In real-world deployments, however, there are practical benefits to pre-specifying designated nodes as initiators.

IV. Experimental Evaluation

We present a comprehensive evaluation of the performance of SNP over a 25 Mica2 node deployment, and using detailed simulations for parameter studies. We show that SNP provides up to 60% savings in network messages for fine-grained data collection. We compare SNP with existing approaches for in-network compression based on network partitioning, and show that these protocols require 25% to 50% more messages than SNP. Using simulation we evaluate the performance of SNP with increasing density and number of nodes in the network. Our results show that SNP scales well, exploiting both correlations and redundancy in dense networks. Finally, we evaluate the effect of different parameters of SNP on its performance and describe how they can be used to tune SNP for different environments.

A. Experimental Setup

We have implemented SNP on Mica2 nodes using COSMOS [1]. COSMOS supports a high-level programming model for sensor networks, with a lean runtime environment. The underlying source to sink data delivery uses tree routing. We present results using a lab testbed of 25 nodes. To evaluate SNP, we use two sensor data traces that are seeded on the sensor nodes. Thus, instead of sending data read from its sensors, the Mica2 nodes send data from the trace for repeatability. The two traces are shown in Figure 5. The first trace, Trace A (top plot), is based on temperature data from the Sonoma forest deployment [17]. The second trace, Trace B (bottom plot), is constructed by adding sharp perturbations to the first trace. This allows comprehensive evaluation of in-network compression in highly dynamic environments.

To enable a comparative study we also implement the PT protocol, and a simple data collection (SDC) protocol for baseline measurements. SDC does not use any in-network compression. The PT protocol implementation uses the same prediction function as the SNP protocol. To allow a fair comparison, we do not incorporate the cost of partitioning the network in the PT protocol. All other overheads, e.g., hop-by-hop messages due to the routing tree are incorporated. We also fix the tree routing structure so that measurements are comparable across runs. We have also built a simulator for the SNP, PT, and SDC protocols, which allows us to evaluate different operational ranges in detail.

Prediction Function: We use Autoregressive Moving Average (ARMA) [18] based prediction to exploit spatio-temporal correlations. A node exploits data from multiple neighbors by taking a weighted average, or auto-regression, (based on spatial distance) of data. In addition to spatial correlations, each node exploits temporal correlations by maintaining a history of its own data and the data from its neighbors. A weighted average based on time (older data has lower weight) represents temporal history. Finally, the moving average component of ARMA captures the history of prediction errors making newer predictions more accurate based on the gradient of data. SNP is, itself, independent of the prediction function used.

B. SNP Performance

We evaluate the performance of SNP in terms of total messages (data sharing messages + messages for transmission of compressed data to sink) w.r.t. the baseline SDC protocol and also compare it to the PT protocol. Additionally, we study the impact of approximate compression using the SNP-ϵ and PT-ϵ variants of the orignal protocols. In our experiments we use $\epsilon = 5\%$. We determined the number of messages required for SNP, SNP-ϵ, PT, PT-ϵ and SDC protocols using our testbed composed of 25 Mica2 motes.

The results of this evaluation are shown in Figure 6. The number of messages are normalized to the number of messages required by SDC. As expected, in-network compression offers significant savings in the number of messages. Due to unexpected perturbations (i.e., fewer correlations) the compression of Trace B is lower than that of Trace A. SNP outperforms PT, reducing the message overhead by up to 30%. Furthermore, as expected, the approximate versions of the SNP and PT protocols perform better in terms of the message overhead. A key point to note is that the overhead of data sharing (shaded boxes) is significant. In fact, the superior performance of the SNP protocol compared to PT can be attributed mostly to the lower data sharing overhead. In all cases the overhead of data sharing in SNP is at least 45% lower than the PT protocol.

Another key point to note is that the data sharing overhead of SNP adjusts to the degree of correlations in the network,

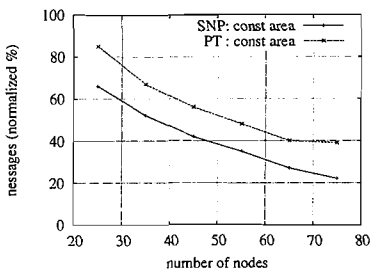

Fig. 7. Messages used by SNP and PT with increasing number of nodes, while the area is kept constant. The number of messages is normalized to the messages used by the SDC scheme.

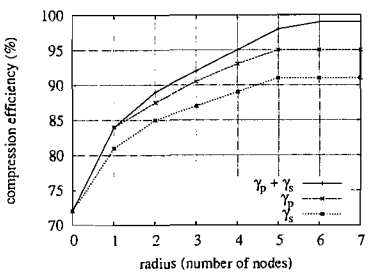

Fig. 8. Effect of changing the number of members in predecessor and successor set of nodes.

while that of the PT protocol remains the same. This is because, for the PT protocol, irrespective of the correlations in the network, each node must send its data to the cluster head for compression. The data sharing overhead of SNP for Trace B is about 20% higher as compared to that of SNP for Trace A due to lower correlations (and thus smaller R) in Trace B. Irrespective of correlations, the data sharing overhead of SNP is lower than PT. This is because, at worst, all nodes broadcast their messages, while in the case of PT all nodes must route their data to the cluster head, which requires $n \cdot k_h$ messages, where k_h is the number of hops to the cluster head.

Impact of Node Density: We study the impact of node density on performance through simulations. To increase density, the spatial area of the network (in the simulator) is kept constant while the number of nodes is increased. We evaluated this scenario using both Trace A and Trace B. Due to space limitation, we discuss results only from Trace B.

We observe that the ratio of messages required for SNP w.r.t. SDC tends to zero, while the same ratio for the PT protocol tends to a non-negligible constant (Figure 7). This is a clear consequence of the linear scaling overhead of PT and SDC and sublinear scaling of SNP.

We now evaluate the characteristics of SNP with respect the tunable parameters of SNP.

C. Changing Correlation Radius.

In the SNP protocol, each node maintains sets $PRED_i$ and $SUCC_i$, whose data is used to predict and, hence, compress data at the node. The sizes of these sets are γ_p, and γ_s, respectively. These parameters capture the correlation radius of a node and impact the memory-correlation tradeoff. We study the impact of γ_p and γ_s on compression. Users may tune γ_p or γ_s or a combination thereof to specific application characteristics.

Evaluation of SNP using both traces were performed. The results from evaluation using Trace B are presented in Figure 8. The metric of evaluation is compression efficiency, which is the ratio of the compression achieved using limited correlation radius with that of the compression achieved using an infinite radius. The three curves in the plots correspond to: (i) increasing γ_s while setting γ_p to zero, (ii) increasing γ_p with γ_s set to zero, and (iii) increasing $\gamma_p + \gamma_s$ with $\gamma_p = \lceil(\gamma_p+\gamma_s)/2\rceil$ and $\gamma_s = \lfloor(\gamma_p+\gamma_s)/2\rfloor$. We observe from Figure 8 that irrespective of the trace, the best performance is archived by using the $\gamma_p + \gamma_s$ approach, since using this approach, nodes are able to use the closest spatial neighbors. This is consistent with the intuition behind the construction of the SN model. An important implication of this result is that small sets are sufficient for achieving high (99%) efficiency. This makes SNP particularly suited to resource constrained nodes such as Mica2, which has only $4KB$ RAM.

D. Broadcast Suppression

An important characteristic of SNP is that it minimizes the number of nodes that need to share their data with neighboring nodes, while achieving high compression. The SNP protocol achieves this by suppressing broadcasts. Two thresholds δ_l and δ_h are defined. If the node's prediction error based on its predecessors is below δ_l, it suppresses broadcast till its prediction error increases beyond δ_h. This results in hysteresis, which is necessary so the predecessor and successor sets of nodes in the network do not change often. Perturbation of these sets results in transmission of message containing the new list of nodes in the sets.

We study the effect of varying these parameters in terms of number of message transmissions as a percentage of the minimum number of messages achieved by varying the hysteresis threshold ($\delta_h - \delta_l$). Based on the characterization, we develop a simple algorithm that automatically discovers the correct hysteresis value. The results for varying the hysteresis threshold while evaluating the SNP protocol using Trace B is shown in Figure 9. The precision threshold δ_l was set at 2%. We observe that there is high overhead associated with low hysteresis. This is because of frequent changes to predecessor and successor sets, which must be communicated to the sink. This overhead decreases rapidly as hysteresis is increased. There is a wide range of hysteresis settings for which the system achieves low overhead. Finally, we note that if the hysteresis is too high, the overhead increases. This is because the selection of nodes does not change often enough to keep up with changing correlations in Trace B, resulting in loss of compression. The fact that there is a wide range of parameters where overhead is low, makes estimation of good hysteresis threshold easy.

V. Related Work

Application dependent in-network processing and aggregation based on data-centric routing has been well studied [12],

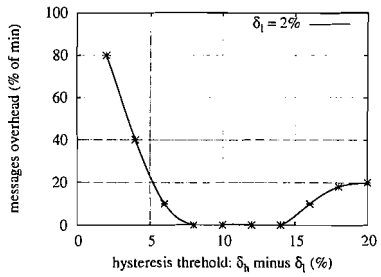

Fig. 9. Characterization of varying hysteresis threshold.

[19]–[22]. The focus of our work is to develop application and routing independent schemes to minimize traffic in sensor networks. To this end, we develop an in-network compression protocol to enable distributed joint coding, achieving a high compression rate, with low overheads.

Traditional data compression schemes [23] can not directly be applied to sensor networks. There have been proposals to apply the much celebrated results of Selpian-Wolf [24] to sensor networks. Selpian-Wolf joint coding can achieve distributed compression without communication between the sources, which is attractive for sensor networks [25]. However, this approach requires precise *a-priori* knowledge of the probability density function of data sources. There have also been efforts aimed at exploiting temporal correlations of each sensor node with its own history [15], [16]. However, the performance of these models is a function of the dynamic variations in data.

In early work on exploiting spatio-temporal correlations in sensed data, researchers have explored the PT model of correlation [7], [10], [11]. We improve on these results, both in terms of compression rates and overhead. Pattem et al. [11] provide an information theoretic basis for the PT protocol. They also propose an implementation of the PT model in which compression can occur on a linear path to the cluster representative, thus, possibly reducing the $O(n)$ overhead of the PT protocol. However, this has the effect of increasing the route-length a data item must travel – thus adversely impacting network capacity. If this is not a consideration, the performance of this model approaches (but does not exceed) that of SNP. Furthermore, a key feature of SNP is that it does not require any support from the underlying routing layer.

VI. Conclusions

In this paper, we present SNP, a novel application independent, lean, in-network compression protocol that achieves high compression rates by exploiting spatio-temporal correlations with low network overheads. We present formal quantification of compression rates, overheads, and scaling, and experimentally demonstrate its performance on real testbeds, as well as through simulations (for parameter studies). We also show that SNP outperforms existing schemes based on these performance parameters.

References

[1] A. Awan, S. Jagannathan, and A. Grama, "Macroprogramming heterogeneous sensor network systems using COSMOS," in *Proc. of EuroSys '07*, March 2007.

[2] "Bowen labs," https://engineering.purdue.edu/CE/BOWEN/Facilities.

[3] K. Chintalapudi, J. Paek, N. Kothari, S. Rangwala, R. Govindan, and E. Johnson, "Embedded sensing of structures: A reality check," in *RTCSA'05*, August 2005.

[4] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler, "A macroscope in the redwoods," in *SenSys'05*, November 2005.

[5] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. of OSDI'06*, November 2006.

[6] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. of WSNA'04*, September 2002.

[7] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proc. of ICDE'06*, April 2006.

[8] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. IT-46, no. 2, March 2000.

[9] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The Emergence of Networking Abstractions and Techniques in TinyOS," in *Proc. of NSDI '04*, March 2004.

[10] D. Petrovic, R. C. Shah, K. Ramchandran, and J.Rabaey, "Data funneling: Routing with aggregation and compression for wireless sensor networks," in *Proc. of SNPA'03*, May 2003.

[11] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," in *Proc. of IPSN'04*, April 2004.

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks," in *Proc. of HICSS*, January 2000.

[13] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "next centur challenges: Scalable coordination in sensor networks," in *Proc. of MOBICOM'99*, August 1999.

[14] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Mobicom'01*, July 2001.

[15] A. Jain, E. Chang, and Y. Wang, "Adaptive stream resource management using kalman filters," in *SIGMOD'04*, June 2004.

[16] M. Li, D. Ganesan, and P. Shenoy, "PRESTO: Feedback-driven data management in sensor networks," in *Proc. of NSDI'06*, May 2006.

[17] G. Tolle, "Sonoma redwoods data, 2005." www.cs.berkeley.edu/~get/sonoma.

[18] G. Box and G. Jenkins, in *Time Series Analysis*. Prentice Hall, 1991.

[19] 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, February 2002.

[20] J. Kulik, W. Rabiner, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. of Mobicom'99*, August 1999.

[21] B. Krishnamachari, D. Estrin, and S. Wicker, "Impact of data aggregation in wireless sensor networks," in *Proc. of ICDCS/DEBS'02*, July 2002.

[22] B. Bonfils and P. Bonnet, "Adaptive and decentralized operator placement for in-network query processing," in *Proc. of IPSN'03*, April 2003.

[23] K. Sayood, *Introduction to data compression*. Morgan Kaufmann Publishers Inc., 1996.

[24] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4.

[25] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, no. 2, March 2002.