An Architecture for Distributed Wavelet Analysis and Processing in Sensor Networks

Raymond S. Wagner
Richard G. Baraniuk
Dept. of Electrical and Computer Engineering
Rice University
Houston, Texas

Shu Du
David B. Johnson
Dept. of Computer Science
Rice University
Houston, Texas

Albert Cohen
Laboratoire Jacques-Louis Lions
Universite Pierre et Marie Curie
Paris, France

ABSTRACT

Distributed wavelet processing within sensor networks holds promise for reducing communication energy and wireless bandwidth usage at sensor nodes. Local collaboration among nodes de-correlates measurements, yielding a sparser data set with significant values at far fewer nodes. Sparsity can then be leveraged for subsequent processing such as measurement compression, de-noising, and query routing. A number of factors complicate realizing such a transform in real-world deployments, including irregular spatial placement of nodes and a potentially prohibitive energy cost associated with calculating the transform in-network. In this paper, we address these concerns head-on; our contributions are fourfold. First, we propose a simple interpolatory wavelet transform for irregular sampling grids. Second, using *ns-2* simulations of network traffic generated by the transform, we establish for a variety of network configurations break-even points in network size beyond which multiscale data processing provides energy savings. Distributed lossy compression of network measurements provides a representative application for this study. Third, we develop a new protocol for extracting approximations given only a vague notion of source statistics and analyze its energy savings over a more intuitive but naïve approach. Finally, we extend the 2-dimensional (2-D) spatial irregular grid transform to a 3-D spatio-temporal transform, demonstrating the substantial gain of distributed 3-D compression over repeated 2-D compression.

Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]: Data compaction and compression

General Terms: Algorithms, Theory

Keywords: distributed wavelet analysis, irregular grid wavelet analysis, sensor networks, compression, multiscale analysis

1. INTRODUCTION

The nodes in a sensor network typically engage in three main tasks: measuring phenomena, processing the measurements, and sharing processed data with a sink or with other nodes in the network via multihop wireless links. With the utility of the network fundamentally constrained by both wireless bandwidth and onboard node power supplies, communication emerges as the most costly of these activities. Thus, it can behoove the network to transmit *answers* to users' *questions* about measured data, rather than the raw data itself. Such distributed processing trades potentially long-haul transmission of raw data to the sink for less costly local communication and processing among neighboring nodes at various spatial scales.

Realizing gains from distributed multiscale processing in sensor networks is complicated by two factors, however. First, nodes in real-world sensor network deployments are typically distributed irregularly in space, a complication highlighted in [6]. Second, depending on the characteristics of the network (number of nodes, transmission range, etc.), the energy and bandwidth overhead associated with distributed processing may be prohibitive, rendering less sophisticated techniques, such as a raw data dump to the sink, more cost effective.

Our main contributions are fourfold. First, we present in Section 3 a protocol for computing a distributed wavelet transform (WT) for irregularly sampled sensor data. This multiscale transform, which is simple to construct and iterate, extends that in [12] and removes the latter's dependence on building and coarsening a distributed mesh within the network. Second, in Section 4 we explore the fundamental tradeoffs between transform overhead cost and energy/bandwidth savings for the application of distributed lossy compression of a measurement field. We accomplish this via *ns-2* emulation of the traffic induced by the transform, using a variety of network configurations. Third, in Section 5 we present a protocol for successive refinement of a lossy measurement reconstruction to reach a specified error target and analyze the savings from such an approach over a more natural but naïve method. Fourth, in Section 6 we extend the two-dimensional (2-D) spatial transform to a 3-D spatio-temporal transform for representing time-series measurements at different sensors, demonstrating the substantial compression benefit to be reaped by exploiting temporal as well as spatial correlations between measurements. Finally, Section 7 concludes with a discussion of ongoing work. We now present a brief review of related techniques for distributed, multiscale processing of sensor network data.

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.

2. RELATED WORK

Multiscale algorithms such as distributed wavelet processing hold much promise for sensor network applications. Measurement fields often exhibit high local correlation and more moderate global correlation, leading naturally to a paradigm of *local processing at fine scales* between spatially proximate nodes and *global processing at coarse scales* between more distant nodes. DIMENSIONS [5] proposes using an in-network wavelet transform for query-routing and storage of network measurements, though a regular-grid node placement and wavelet transforms are assumed in its implementation. To solve only the query routing problem, multiscale techniques are also employed in Fractional Cascading [7], which allows each node its own multiresolution viewpoint rather than distributing a single multiscale decomposition of measurements throughout the network as does DIMENSIONS.

Separable application of 1-D wavelet transforms are proposed in [9] to solve the 2-D sensor broadcast problem, although regular node placement is again assumed. A wavelet-based protocol for data gathering is proposed in [1], but its application is limited to 1-D, regularly spaced grids. In a similar spirit, [4] develops a 2-D transform by tracing a 1-D path through a measurement field and employing a 1-D wavelet filter along the path. It is not clear how to choose an optimal path for data compaction, however, and even given one this 1-D approach will not be able to exploit 2-D spatial dependencies among data as well as a fully 2-D method.

The technique described here extends that found in [12], which develops a distributed, irregular-grid WT based on the theory of lifting [11]. To stabilize the transform data for procedures such as data compression and de-noising, [12] relies on the construction and repeated coarsening of a distributed mesh among nodes in the network. Maintaining this mesh requires a good deal of communication overhead, and the technique does not tolerate occasional loss of connectivity between nodes, requiring global re-meshing when connectivity is lost. We formulate here a much more elegant, meshless approach to building a stable transform.

3. IRREGULAR GRID WAVELET ANALYSIS

Given a time snapshot of a measured spatial field, distributed wavelet analysis replaces the 2-D [1] set of node measurements with a set of transform coefficients that, for piecewise smooth fields, are more sparse than the original data — that is, there are fewer important wavelet coefficients than measurements. This process involves local collaboration among nodes and is multiscale in nature — that is, collaborative neighborhoods become broader as the transform progresses from fine to coarse scales.

3.1 Multiscale Hierarchies

Let $j \in \{J, ..., 1, 0\}$ represent the transform scale, with $j = J$ corresponding to the original (finest) scale of measurements and $j = 0$ to the final (coarsest) scale of the transform. Label the complete set of nodes E_J and the set of original measurements $\{s_{J,n}\}_{n \in E_J}$ — the finest-scale set

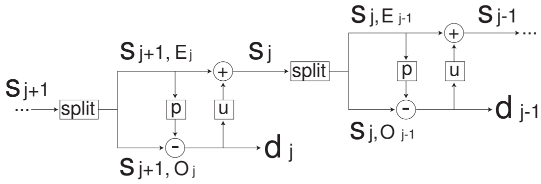


Figure 1: A cascaded pair of lifting stages. Scaling coefficients from scale $j+1$ are first split into an even set (abbreviated s_{j+1,E_j}) and odd set (s_{j+1,O_j}). Each scale-j wavelet coefficient in d_j is found as the difference of the value in s_{j+1,O_j} and values from s_{j+1,E_j} processed by a predict operator. Each new scale-j scaling coefficient in s_j is then found as the sum of the value in s_{j+1,E_j} and values from d_j processed by an update operator. The transform repeats at scale $j-1$ using s_j.

of *scaling coefficients*. We describe the transform in the language of wavelet lifting, which factors each transform stage into three steps: *split, predict,* and *update* [11].

Figure 1 illustrates a pair of cascaded lifting stages. Starting with $j = J - 1$, we first split the set of sensors E_{j+1} into an *even* set E_j and an *odd* set O_j. This partitions the scale–$(j+1)$ scaling coefficients into sets $\{s_{j+1,m}\}_{m \in E_j}$ and $\{s_{j+1,n}\}_{n \in O_j}$. We then predict a scale-j *wavelet coefficient* $d_{j,n}$ at each node $n \in O_j$ using the odd coefficient $s_{j+1,n}$ and a predict operator applied to some set of even coefficients $\{s_{j+1,m}\}_{m \in \mathcal{N}(n)}$ in a spatial neighborhood $\mathcal{N}(n) \in E_j$ around node n. The complete set of predicted values comprises the set of scale-j wavelet coefficients $d_j := \{d_{j,n}\}_{n \in O_j}$. Finally, we update the scaling coefficient at each node $m \in E_j$ using the even coefficient $s_{j+1,m}$ and an update operator applied to a set of scale-j wavelet coefficients $\{d_{j,n}\}_{n \in \mathcal{N}(m)}$ from nodes in some neighborhood $\mathcal{N}(m) \in O_j$ of m. The complete set of updated coefficients $s_j := \{s_{j,m}\}_{m \in E_j}$ provides the input to the next coarser scale of transform, $j - 1$. The transform iterates to the coarsest scale $j = 0$, and the terminal set of scaling coefficients, s_0, along the set of wavelet coefficients from each scale, $\{d_j\}_{j=0}^{J-1}$, number the same as and completely describe the original measured data.

The following three subsections discuss the implementation of the predict, split, and update stages, which depend solely on and assume access to nodes' self-localized positions [8]. Though transform parameters could be computed in-network akin to the method suggested in [12], we propose here a more *centralized* solution to reduce the communication overhead. For transform invertibility, the data sink must know the network's transform parameters. Since the sink must also know nodes' self-localized positions to make sense of their measurements, it can compute the transform parameters itself with zero transmission overhead and inform the nodes, rather than the reverse.

These calculations assume perfect wireless connectivity among nodes, and nodes must then locally repair the initial transform as needed to deal with imperfect wireless connectivity, informing the sink of new neighborhood memberships. Such a situation only occurs when a pair of nodes cannot find *any* multi-hop path to connect themselves and should be rare. The cost of repair can be amortized over multiple iterations of the distributed transform. As discussed further in Section 3.5, this technique also provides the transform

[1]Of course, sensor locations actually populate a 3-D space (such as a building), but we assume WLOG a 2-D sensor field.

Procedure 1 $[E_j, O_j] = \text{Split}(E_{j+1})$

$O_j \leftarrow \emptyset$
$n \leftarrow \text{first}(E_{j+1})$
while $n \neq \text{last}(E_{j+1})$ **do**
    for each n' s.t. $d(n, n') < 2^{-j}$ **do**
        $O_j \leftarrow O_j \bigcup n'$
        $E_{j+1} \leftarrow E_{j+1} \setminus n'$
    end for
    if $n \neq \text{last}(E_{j+1})$ **then**
        $n \leftarrow \text{next}(E_{j+1})$
    end if
end while
$E_j \leftarrow E_{j+1}$

with a means of accommodating time-varying network connectivity among nodes.

3.2 Split Design

To split nodes into even and odd sets at each scale j, we take inspiration from the regular grid setting of image processing, where the distance between neighboring nodes doubles in both the horizontal and vertical directions following a round of decimation. We outline a split procedure that guarantees a minimum distance of 2^{-j} between any pair of nodes in the even set E_j following the split. Call the minimum distance between any two nodes $d_{\min} = \min d(n, n')$, where $n, n' \in E_J$ (the original set of nodes) and $n \neq n'$, with $d(\cdot)$ measuring Euclidean distance in the plane. The starting scale $J-1$ is set so that $J - 1 = -\lceil \log_2 d_{\min} \rceil$. To split at a scale j, we iteratively examine each sensor $n \in E_{j+1}$, identifying all $n' \in E_{j+1}$ for which $d(n, n') < 2^{-j}$. These n' are placed in the odd set O_j for that scale and removed from the working copy of E_{j+1}, whose next element is then examined for close neighbors. Once the last element of E_{j+1} has been reached, the elements not removed from E_{j+1} comprise the even set E_j for scale j. The method is summarized in Procedure 1.

3.3 Predict Design

For each point n placed in O_j by the split routine, we must define a vector $p_{j,n} := p_{j,n}(m), m \in \mathcal{N}(n)$ used in an inner-product with scale-$(j+1)$ scaling values at sensors in a neighborhood $\mathcal{N}(n) \in E_j$. The difference between the value returned by the inner product and $s_{j+1,n}$ gives us $d_{j,n}$, the scale-j wavelet coefficient at sensor n

$$d_{j,n} = s_{j+1,n} - \sum_{m \in \mathcal{N}(n)} p_{j,n}(m) s_{j+1,m}, \tag{1}$$

We design a predict stage that is *second-order accurate* — that is, one that generates zero-valued wavelet coefficients when data comprise a plane over sensors in $\mathcal{N}(n)$. This amounts to requiring that the predictor regress a plane centered at n through the scale-$(j+1)$ scaling coefficients at $\mathcal{N}(n)$. We can easily choose to regress higher-order polynomials for greater predict accuracy — a decision which yields superior sparsity of the wavelet coefficients but comes at the price of higher communication cost to traverse a larger $\mathcal{N}(n)$. For the ease of explanation, though, we discus planar regression.

Rather than choosing $\mathcal{N}(n)$ directly, we pick a radius h around n and designate neighbors within that radius as $\mathcal{N}(n)$. This allows us to achieve desired stability in the predict step by tuning h. Linear least-squares regression of a plane centered at n and normalized radially by h involves minimizing the following quantity over plane parameters α_1, α_2, and α_3

$$\sum_{m \in \mathcal{N}(n)} \left[s_{j,m} - \alpha_1 - \alpha_2 \left(\frac{x_n - x_m}{h} \right) - \alpha_3 \left(\frac{y_n - y_m}{h} \right) \right]^2, \tag{2}$$

where x_m and y_m give the x and y coordinates of node m. Solving this minimization and re-formulating the result to yield the predict vector $p_{j,n}$ from (1), we have that

$$p_{j,n} = [1, 0, 0] \cdot G^{-1} \cdot X^T, \tag{3}$$

where $G = X^T X$ and the $\#\mathcal{N}(n) \times 3$ matrix X is given by $X = \left[1, \frac{1}{h}\left(x_{\mathcal{N}(n)} - x_n\right), \frac{1}{h}\left(y_{\mathcal{N}(n)} - y_n\right)\right]$.[2]

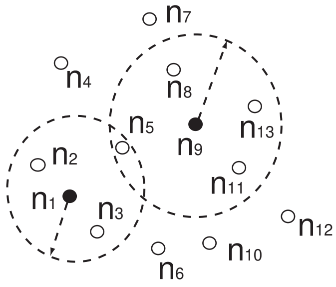


Figure 2: Each node $\{n_1, n_9\} \in O_j$ (marked as $\bullet$) grows its radius until it has found enough neighbors $\mathcal{N}(n) \in E_j$ to stabilize its predict.

Rather than relying on a distributed mesh to guide neighbor selection and stabilize the prediction as in [12], we simply grow the predict radius h until enough neighbors are found to stabilize the inversion of G — that is, until G exhibits a desired condition number. If h is grown enough to encompass all of E_j and the target still cannot be reached, then n should be removed from O_j and inserted into E_j. In simulations, a target condition number of 25 with a random, uniform distribution of node locations typically gives desired stability while allowing the bulk of points selected for prediction at each scale to be predicted. Figure 2 illustrates the radius growth process to find a stable neighborhood.

3.4 Update Design

Once wavelet values are predicted at the sensors O_j, we must update the scale-$(j+1)$ scaling values $\{s_{j+1,m}\}_{m \in E_j}$ at the un-decimated sensors to yield a smoother set of scale-j scaling values s_j. This has the effect of further stabilizing the overall transform. Analogous to the predict stage, we construct for each non-predicted point $m \in E_j$ a vector $u_{j,m} := u_{j,m}(n), n \in \mathcal{N}(m)$ whose inner product with wavelet values at predicted sensors in a neighborhood $\mathcal{N}(m) \in O_j$ adds to m's scale-$(j+1)$ scaling value to give a scale-j scaling value

$$s_{j,m} = s_{j+1,m} + \sum_{n \in \mathcal{N}(m)} u_{j,m}(n) d_{j,n}. \tag{4}$$

We define $\mathcal{N}(m)$ as the set of nodes whose scale-j wavelet values were predicted from m in the previous step.

We adopt with slight modification the update technique proposed in [12]. The goal of the update is relatively straightforward — we want to preserve the average value

[2] $\#$ denotes the number of elements in a set.

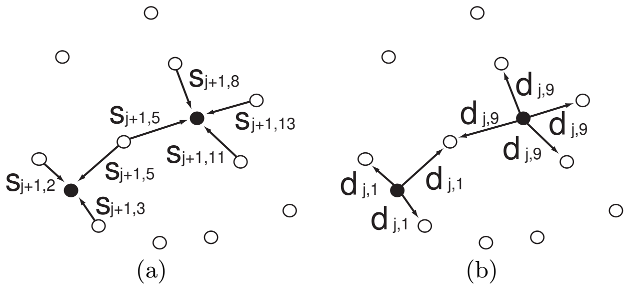


Figure 3: Communication flow at scale j: (a) First, each predicted node $\{n_1, n_9\} \in O_j$ (marked as $\bullet$) receives a scale-$(j+1)$ scaling coefficient from each neighbor in its predict neighborhood $\{\mathcal{N}(n_1), \mathcal{N}(n_9)\} \in E_j$. (b) Then $\{n_1, n_9\}$ each transmits its scale-j wavelet coefficient to each updated neighbor in $\{\mathcal{N}(n_1), \mathcal{N}(n_9)\}$.

of the field across scales. This amounts to computing update filters that keep the sum of scaling function integrals weighted by scaling coefficients constant across scales. The integrals here are treated as discrete sums over the original measurement grid E_J — the main departure from the update procedure of [12], which considers continuous integrals over spatial indictor functions derived from a mesh. We must specify these integrals at each scale, beginning at the finest scale by defining scale-J scaling functions as delta functions so that the integral $I_{J,m} = 1 \; \forall \; m \in E_J$. Wavelet lifting provides the mechanics to relate the integral of node $m \in E_j$ at a scale j to its scale-$(j+1)$ integral and those of its predicted neighbors $\mathcal{N}(m) \in O_j$ as

$$I_{j,m} = I_{j+1,m} + \sum_{n \in \mathcal{N}(m)} p_{j,n}(m) I_{j+1,n}. \tag{5}$$

Using these integrals, we can solve for the appropriate update filter values. Rather than calculating update filters directly for each updated node, it is much easier to find filter values with respect to each *predicted* node. That is, we calculate for each predicted node n the value $u_{j,m}(n)$ with which each $m \in \mathcal{N}(n)$ should weight the wavelet coefficient $d_{j,n}$ during m's update

$$u_{j,m}(n) = \frac{I_{j,m} I_{j+1,n}}{\sum_{k \in \mathcal{N}(n)} I_{j,k}^2}, \tag{6}$$

Iterating this procedure at each $n \in O_j$ determines the entire set of scale-j update vectors. Note that nodes in E_j which are not used to predict wavelet values at O_j are not updated, but retain their scale-$(j+1)$ scaling values through scale j.

3.5 Synchronization and Robustness

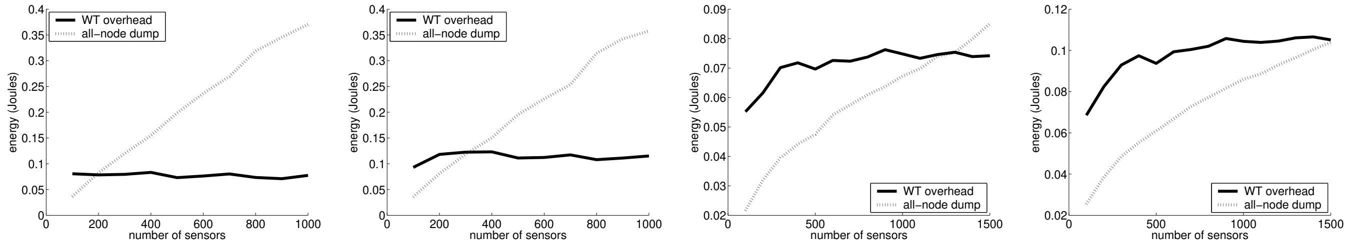
As mentioned above, the WT parameters can be computed in a centralized fashion at the data sink and then sent through the network to each node, which must know: (i) the scale at which it is predicted, (ii) the neighbors it uses for its prediction at that scale, and (iii) the neighbors it helps predict at all finer scales. The transform begins at scale $J-1$ with sensors in $E(J-1)$ sending their scale-J scaling coefficients (raw measurements) to predicted neighbors in $O(J-1)$. Once a sensor in $O(J-1)$ has heard from each neighbor used in its prediction, it can compute a scale-$(J-1)$ wavelet value and send that value back to each updated neighbor. When a sensor in $E(J-1)$ receives wavelet values from each predicted neighbor, it can compute its scale-$(J-1)$ scaling value. An updated sensor can then participate in scale-$(J-2)$ of the transform, contacting predicted neighbors or waiting to hear from neighbors used in its own prediction as specified by the scale-$(J-2)$ split of sensors in $E(J-1)$. Since a node does not attempt to compute a scale-j coefficient until all relevant neighbor data is received, there is no danger of synchronization errors due to slower nodes. Figure 3 illustrates the communications traffic among the predicted and updated nodes at scale j of the WT.

On occasion, a pair of nodes may not be able to find *any* multihop path to share their coefficients, and so the transform must locally adapt to guarantee predict stability. If a predicted node n at scale j is unable to hear from a neighbor $m \in \mathcal{N}(n)$, it must recompute its predict filter according to (3) with a neighborhood of $\mathcal{N}(n) \setminus m$, which may result in ill-conditioning of matrix G in the equation. In such a case, it can begin looking for new predict neighbors by searching in a progressively widening neighborhood radius $\widehat{h}$ until the new neighborhood $\widehat{\mathcal{N}}(n)$ enables a new, well-conditioned $\widehat{G}$. Given this new neighborhood, node n can re-compute its set of predict filters as well as the new update values required by its neighbors in $\widehat{\mathcal{N}}(n)$. Note that, to maintain proper integral bookkeeping, nodes in $\widehat{\mathcal{N}}(n)$ must re-compute their scale-j integrals to account for n's new set of predict coefficients, and disconnected node m must remove n's contribution to its scale-j integral once it realizes n has not responded with a scale-j wavelet coefficient. This implies that m must keep track of n's contribution to m's integral at scale j. Additionally, this will trigger re-computation of all update filters at coarser scales which descend from the scaling functions of m and $\widehat{\mathcal{N}}(n)$ at scale-$(j-1)$. Clearly, this in-network link repair technique is suited only to networks with occasional connectivity losses, as the repair overhead will become prohibitive as the frequency of failing network links increases. To control the degree of repair needed, the terminal scale of the transform can be set at some scale $j > 0$ — in fact, such a setting is typical even in traditional, regular-grid wavelet applications. Finer terminal scales lead to less sparsity in the wavelet coefficients but limit the spatial scope of repair traffic.

4. TRANSFORM COMMUNICATION COST

Computing the WT in a distributed fashion requires a nontrivial amount of communication between nodes in the sensor network. And while this communication is considered local within scales, the spatial area covered by local neighborhoods increases as scale coarsens. Thus, the communication overhead for obtaining the transform data may become substantial, and in some cases, dominant. While this may be the case for certain network configurations, we now show that there exist *break-even points* in both network size and communication density beyond which the cost of multiscale analysis is acceptable.

Though the potential applications of wavelet analysis are many, including feature extraction, measurement de-noising, and query routing, we explore here in detail one of the most common: lossy compression of a measurement field. Wavelet compression leverages the WT's strong *energy compaction property* — that is, wavelet coefficients generated from a



(a) (b) (c) (d)

Figure 4: *ns-2* simulation of the energy cost of computing the WT (solid) and dumping all raw measurements to the data sink (dashed) versus network size. Energy is computed in the bottleneck metric for (a) 10 and (b) 20 radio neighbors per sensor and in the network average metric for (c) 10 and (d) 20 radio neighbors per sensor.

piecewise-smooth measurement field tend to exhibit greater sparsity than do the original measurements. This concentrates signal energy into relatively fewer wavelet coefficients, permitting the user pick the N largest coefficients as the best N-term nonlinear approximation of the field. With a properly designed WT, reconstruction error signal energy is proportional to the energy of the discarded wavelet coefficients. In the sensor network setting, coefficients are selected by broadcasting a threshold to the network, and nodes with coefficient magnitudes above the threshold report their data to the sink.[3] We assume in this section *a priori* knowledge of the threshold and leave the details of a protocol for harvesting coefficients using only general knowledge of their statistics to Section 5.

Wavelet compression allows the network to trade reconstruction quality for communication energy and bandwidth usage. But any energy savings is offset by the overhead cost of computing the wavelet coefficients. We now characterize the break-even point for distributed wavelet processing.

4.1 Break-Even Analysis of Distributed Wavelet Processing

For a given network configuration (say, fixed communication density), there will be a network size below which wavelet-based compression is less efficient than straightforward forwarding of the set of raw measurements to the data sink. In other words, the overhead cost of the WT, prior to sending *any* significant wavelet coefficients to the sink, will dominate the cost to offload all measurements. The same logic applies to fixed network size with varying size of node radio neighborhoods.

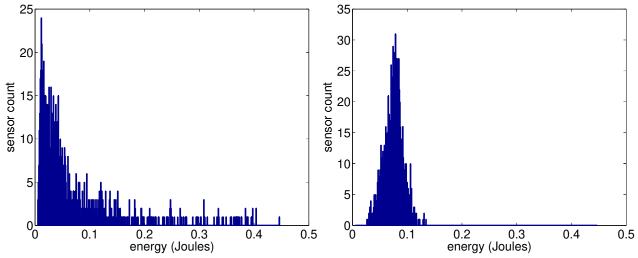
For a relatively small number of sensors with large transmission ranges, each sensor has a large fraction of the network as its radio neighbors. The collaborative overhead of the WT is wasted — a single direct communication with the sink is less expensive than communication with a set of neighbors. For a network of many sensors with few radio neighbors, however, hop count begins to approach geographic distance, and the localized nature of WT communication will require paths on average much shorter than a node's expected distance to the sink. This allows the WT overhead to easily win with plenty of energy left over for streaming significant coefficients to the sink.

To explore this tradeoff, we simulate the energy cost of WT network traffic and the traffic to dump all measurements to the data sink. The average number of radio neighbors for a network is fixed, and networks of varying size are generated with uniform random node placement. Two energy metrics are used. The first is average energy consumed at the one-hop neighbors of the sink, which we refer to as *bottleneck energy.* This is a critical metric, since it gives a direct measure of network lifetime. When the one-hop neighbors of the sink deplete their power supplies, the sink will be cut off from the network, rendering it useless. The second metric is *average energy* consumed by all nodes in the network.

Energy expenditure is simulated using version 2.1b8a of *ns-2* with the Monarch wireless and mobile extensions [3]. These extensions model an IEEE 802.11-based network with a wireless physical rate of 2Mbps, a nominal wireless transmission range of 250 m and a carrier sensing range of 550 m. The RTS/CTS of 802.11 are turned off during our simulations so that all the unicast traffic use DATA/ACK frames in the MAC layer. For the node energy model, we use the default energy model provided by *ns-2*, which defines a transmission power of 0.6W and receiving power of 0.3W; the network area is scaled so that node radios achieve a desired coverage (average number of radio neighbors per node). All of the nodes are initialized with enough power so that their supplies are not depleted during the simulations. We assume an omnipotent routing protocol has already been deployed to provide the shortest path between any two nodes before the WT process starts, so that there is no additional routing traffic during the process of wavelet transformation or querying and harvesting. In all cases, the sink is located at the center of the measurement field, and packets are 24 bytes in length, allowing each to carry a single coefficient as an 8-byte double-precision floating point number with 16 bytes left over for potential header information. Results for each network size are averaged over 5 instantiations using randomly, uniformly distributed node locations.

Figures 4(a) and 4(b) show results for the bottleneck energy metric, plotting energy consumed versus number of sensors in the network. Figure 4(a) sets the network to have an average of 10 radio neighbors per sensor, while Figure 4(b) permits an average of 20 per sensor. The crossover point is seen for a relatively small number of sensors in each — around 200 for the 10-neighbor case and around 300 for the 20-neighbor case. This result is not surprising. The bottleneck cost of the network dump scales with the number of sensors, so we expect to see a roughly linear increase in cost. For the WT, each node requires on average 3.5 neighbors

[3]Note that each node with a scaling coefficient at the terminal scale — that is, a node in $E(0)$ — must always transmit its scale-0 scaling coefficient to the sink.



(a) (b)

Figure 5: Histograms of relative energy usage for (a) a dump of all measurements to the sink and (b) the WT calculation overhead for a network of 1000 nodes with an average of 10 radio neighbors per node. Note the much smaller variance of WT energy consumption.

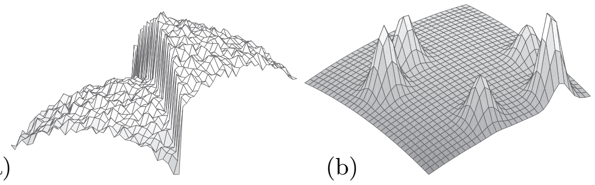
to predict its wavelet coefficient using the target condition number mentioned in Section 3.3. As the network size increases, this cost remains fixed for one-hop neighbors of the sink, and the remainder of their energy expenditure depends on (i) the number of sensors they help predict, and (ii) the amount of network traffic routed through them. The net effect of these two costs will tend to increase slowly, but it is dependent on random node placement and the trend is not as visible with the bottleneck energy metric's average over a small number of nodes.

Figures 4(c) and 4(d) track energy expenditures averaged among all nodes in the network — again, for 10 and 20 radio neighbors per sensor, respectively. Here the cost of the wavelet transform is averaged among a larger number of sensors, and we can see more clearly its gradual growth with increasing network size. Not surprisingly, a larger number of nodes are required to achieve a crossover in the WT energy cost, since average energy does not fixate on the sink the way bottleneck energy does. For the 10-neighbor case, a crossover is evidenced around 1300 nodes, and for the 20 radio neighbors the WT curve is trending toward a break-even point near 1500 nodes.

An alternate look at global energy expenditure provides a bit of additional intuition to the benefits of the WT. Consider Figure 5, which presents histograms of node energy consumptions for the 1000-node, 10-neighbor network. Figure 5(a) gives the energy distribution for the network-wide dump while Figure 5(b) shows distribution of WT overhead energy. Clearly, though the mean of the WT expenditure is a bit higher, its variance is much lower than the dump, indicating that the WT *spreads the energy cost* more uniformly across the network, thereby avoiding concentrating it in regions such as the bottleneck. Thus, even for networks where the global expenditure is greater for the WT, it can help spread the cost of data gathering over all nodes in the network, prolonging the lifetime of the bottleneck at the expense of higher power output at non-critical sensors.

4.2 Distortion/Energy Analysis of Distributed Compression

Given that the cost of the WT is non-prohibitive for networks of a certain size, we can evaluate its utility for the distributed compression application. Wavelet compression enables trading reconstruction error for energy spent transporting wavelet coefficients to the sink, tracing a decreasing distortion curve along an energy axis (E/D curve). The first point on the curve begins at the WT's energy overhead, where zero wavelet coefficients are sent to the sink for maximum distortion. It effectively ends at the energy cost for transporting all raw measurements to the sink. With good energy compaction properties, the reconstruction should have nearly zero error using much less energy than a network-wide measurement dump.



(a) (b)

Figure 6: Prototype measurement fields: (a) a noisey, discontinuous quadratic field, and (b) random Gaussian bumps populating a smoothly-varying quadratic field.

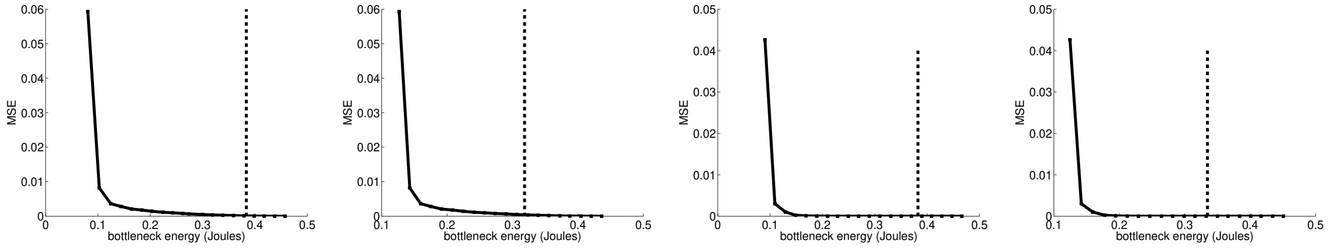
For all the query traffic, we use a threshold broadcast scheme. The sink node broadcasts the query to its one-hop neighbors, and a node forwards the query to its neighbors upon first receipt. Reply traffic follows a unicast model, where the source node and all the intermediate forwarding nodes deliver packets along the shortest path provided by the omnipotent routing protocol.

We consider 1000-node samples of measurement fields such as those illustrated in Figure 6. Figure 6(a) shows a noisy, inverted quadratic bowl with a discontinuity along the line $x = y$. Figure 6(b) gives a set of randomly located Gaussian bumps of random height populating a slowly-varying quadratic field. Both fields exhibit super-planar features well beyond the first vanishing moment of the WT and yield significant sets of wavelet coefficients. Mean squared error is measured in the ℓ^2 norm, the average of the squares of the differences between approximated and actual sensor values. To match the error metric, we implement ℓ^2 thresholding of wavelet coefficient magnitudes: a scale-j coefficient $d_{j,n}$ is compared against the uniform threshold as $2^{-j}|d_{j,n}|$ [2].

Distortion versus bottleneck energy is plotted for the discontinuous quadratic field in Figures 7(a) and (b) for 10 and 20 radio neighbors on average per node, respectively. In both cases, the dashed vertical line marks the energy consumed by dumping all node measurements to the sink. The distortion drops substantially in both instances when only a small fraction of coefficients are sent to the sink, and the network dump energy lies well within the effectively zero-distortion regime of the curves. Small reconstruction error is realizable using upwards of 30% of the dump energy in the 10-neighbor case and 50% of the dump energy in the 20-neighbor case. Results are even more impressive for the smooth Gaussian bump field, shown in Figures 7(c) and (d) for 10 and 20 average radio neighbors. To achieve nearly zero reconstruction error using wavelet compression, we require only 30% of the dump energy in the 10-neighbor case and 50% of the dump energy for the 20-neighbor case.

5. APPROXIMATION REFINEMENT PROTOCOL

As mentioned in Section 4, achieving a desired reconstruc-



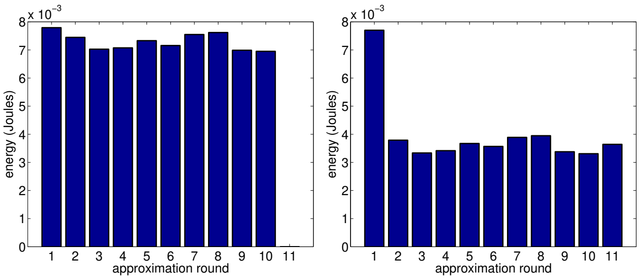
(a) (b) (c) (d)

Figure 7: Distortion versus energy curves for 1000-node samplings of the discontinuous quadratic field with (a) 10 radio neighbors per node and (b) 20 neighbors per node on average and the Gaussian bump field with (c) 10 and (d) 20 average radio neighbors. In all cases, the vertical dotted line marks the energy required to send all 1000 original measurements to the sink.

tion fidelity involves choosing a proper threshold for judging coefficients' significance. Picking the threshold to achieve a desired MSE from the outset requires not only access to all coefficient values but also truth data against which to measure the error — neither of which are available at the sink. Thus, in practice we must pick a reasonable starting threshold, query the network for significant coefficients, and then repeat the process with subsequently lower thresholds until the signal energy difference between successive approximations lies below some target value. Intelligent choices for threshold values can be gleaned from network history. Note that this is rather unlike the classic image processing approach, where the user has access to all coefficients at no cost and can sort them by magnitude to determine the best n-term approximation. Thus, compression in the sensor network setting is *fundamentally different* from standard image compression, since the user must pay a price to harvest data. It therefore behooves us to reduce this cost as much as possible.

We must flood the network with at minimum two threshold queries to evaluate the stop criterion, but in general we may need to issue multiple requests. Rather than issuing multiple network-wide query floods, we propose a more efficient approach. The first query flood contains a set of decreasing threshold values to sweep along the E/D curve from high to low distortion regimes. Nodes will then respond to the sink in a time delayed fashion. Those above the first threshold will immediately respond. Those between the first and second thresholds — in the second threshold "band" — will respond after some delay. Those in the third band reply after a greater delay, and so on. As coefficients stream in, the sink periodically measures the signal energy differences between successive approximations and issues a "stop" flood when some target has been reached. The savings from issuing two rather than arbitrarily many queries can be substantial, provided that the threshold bands can be contained in a single flood packet. Such a feat can be accomplished by either describing thresholds at less than full double floating-point precision or storing a finite number of candidate thresholds at each node and referring to these with integer indices in query packets.

To illustrate the potential savings, we compare the cost of issuing multiple queries with that of the banded-query approach. A series of 10 approximation rounds are conducted on coefficients from the 1000-node discontinuous quadratic field. The bar plot of Figure 8(a) depicts the energy expended per round for repeated querying, and Figure 8(b) gives the energy for issuing a single banded query followed



(a) (b)

Figure 8: Energy expenditure per approximation round for 10 rounds of (a) repeated querying and (b) single banded-query with stop message after round 10.

by a stop flood. Each technique issues a query and expends the same energy in the first round, but the banded-query technique consumes approximately half the energy in subsequent rounds by not issuing a query. Even the final "stop" query it issues after the 10th round does not substantially raise its total energy cost, which is about 60% of the repeated querying approach's cost of 0.07 J.

6. SPATIO-TEMPORAL WAVELET ANALYSIS

The WT presented thus far operates on a set of data sampled in space at the same instant in time at all nodes. In practice, nodes will collect time-series of measurements that not only exhibit spatial correlations among measurements at the same time index but also temporal correlations among the set of data gathered at each node. Thus, to implement a truly effective compression algorithm, we must design a 3-D wavelet transform that operates in both the space and time. While the irregular spatial sampling grid motivates the need for novel wavelet theory, the regularly-spaced temporal samples at each sensor make extending the transform to the time dimension much easier. We build an overall 3-D wavelet transform via separable application of 1-D and 2-D transforms. First, a 1-D regular grid WT is applied to each node's time series of measurements, generating a series of temporal wavelet coefficients at each node. Then, the 2-D irregular-grid wavelet transform is computed for each 2-D plane of the wavelet coefficient series.

The centralized nature of this 1-D compression at each sensor allows us to build more efficiency into the 3-D coder. Nodes now share vectors of wavelet coefficients during the

2-D encoding process, and as these vectors arise from correlated time-series they can be efficiently represented by classic wavelet zerotree coding [10]. In other words, a node shares a zerotree-compressed (rather than full-bitrate) vector of scaling or wavelet coefficients with neighbors during the encoding process. For correlated time series the compression factor can be quite high while introducing only a modest amount of quantization error. As we demonstrate below, the 2-D transform is very robust to this intermediate error.

For fields that vary in both time and space, the compression benefits of 3-D encoding can be substantial. As an example, we consider step function whose transition edge moves slowly across a measurement field from left to right, and we sample the time-varying field at 64 instants with a network of 300 uniformly, randomly distributed nodes. To study repeated 2-D compression, we first compute a 2-D WT for each snapshot in time and then query each transform plane for an equal number of wavelet coefficients.

We reconstruct the full 3-D field and plot the MSE of the approximation versus the number of coefficients used to generate it, giving the dotted curve of Figure 9(a). We then compare this against a full 3-D encoder using D-7,9 wavelets along the time-axis and *no* intermediate zerotree compression, querying the 3-D coefficient set each time for the same number of nonzero coefficients used over 64 queries of the 2-D transform data.[4] This gives the solid curve of Figure 9(a), which exhibits a much faster MSE decay than that of repeated 2-D compression.

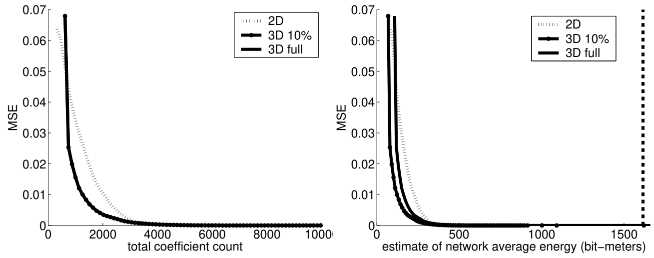
We then repeat the 3-D experiment with intermediate zerotree compression that reduces nodes' coefficient vectors to 10% of their original size, giving the curve marked by crosses in Figure 9(a). The results are nearly identical to those for the 3-D transform with no intermediate compression, indicating that the 2-D spatial transform step is robust to the quantization errors induced by even a high temporal compression rate. This is a testament not only to the 2-D encoder's robustness but also to the zerotree coder's high efficiency.

Finally we estimate the average energy used by each node in both the transform and the approximation in *bit-meters*, which approximate energy consumption by accounting for the distance bits must be moved to the sink, located at the center of a $1m^2$ box containing all sensors. Figure 9(b) plots these curves with the same curve symbols and an estimate for the average energy needed to transport all original measurements to the sink given by a dashed, vertical line similar to Figure 7. The benefit of 3-D encoding with intermediate 1-D zerotree compression is obvious.

7. CONCLUSION

In this paper, we have developed a new distributed wavelet transform and data harvesting architecture for sensor networks. By design, the transform adapts to irregular sampling grids and sparsifies piecewise-smooth sensor measurement fields. Using *ns-2* simulations, we have established the fundamental result that as the network size grows there exists a break-even point beyond which multiscale data processing provides energy savings over merely exporting the raw measurements. Finally, we have extended the 2-D spatial transform to a 3-D spatio-temporal transform to provide further compression gains. Our current research focuses on new protocols to support in-network operations beyond compression with our 2-D and 3-D transforms.

[4]Note that each node in $E(0)$ of the 2-D transform must send all 64 terminal scaling coefficients to the sink; all remaining nodes must only send unaltered scaling coefficients from the 1-D WT of their time series.



(a) (b)

Figure 9: (a) MSE versus coefficient count for repeated 2-D compression (dotted), 3-D compression with full-bitrate intermediate messages (solid), and 3-D compression with zerotree coding of messages at 10% bitrate (crosses). (b) MSE versus estimated average network energy (in bit-meters). The dashed vertical line estimates average energy for a sink dump of original 3-D measurements.

8. ACKNOWLEDGMENTS

Supported by NSF-NeTS, AFOSR, ONR, and the Texas Instruments Leadership University Program.
Web: compass.cs.rice.edu, dsp.rice.edu.

9. REFERENCES

[1] J. Aćimović, R. Cristescu, and B. Beferull-Lozano. Efficient distributed multiresolution processing for data gathering in sensor networks. In *Proc. IEEE Int. Conf. on Acoustic and Speech Sig. Proc. (ICASSP)*, pages 837–840, Mar. 2005.

[2] S. Amat, F. Aràndiga, A. Cohen, R. Donat, G. Garcia, and M. von Oehsen. Data compression with ENO schemes: A case study. *App. and Comp. Harmonic Analysis*, 11:273–288, 2001.

[3] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. Int. Conf. on Mobile Comp. and Net. MobiCom*, pages 85–97, 1998.

[4] A. Ciancio and A. Ortega. A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting. In *Proc. IEEE Int. Conf. on Acoustic and Speech Sig. Proc. (ICASSP)*, pages 825–828, Mar. 2005.

[5] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan. Multi-resolution storage and search in sensor networks. *ACM Trans. on Storage*, V(N), Apr. 2005.

[6] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. *SIGCOMM Comput. Commun. Rev.*, 34(1):125–130, 2004.

[7] J. Gao, L. Guibas, J. Hershberger, and L. Zhang. Fractionally cascaded information in a sensor network. In *Proc. Int. Symp. Inf. Proc. in Sensor Networks (IPSN)*, pages 311–319, 2004.

[8] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proc. Int. Conf. on Mobile Comp. and Net. (MobiCom)*, pages 45–57, 2004.

[9] S. Servetto. Distributed signal processing algorithms for the sensor broadcast problem. In *Proc. Conf. on Information Sciences and Systems (CISS)*, Mar. 2003.

[10] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41:3445–3462, Dec. 1993.

[11] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, Mar. 1998.

[12] R. Wagner, H. Choi, R. Baraniuk, and V. Delouille. Distributed wavlet transform for irregular sensor network grids. In *Proc. IEEE Stat. Sig. Proc. Workshop (SSP)*, Jul. 2005.