ENERGY-EFFICIENT GRAPH-BASED WAVELETS FOR DISTRIBUTED CODING IN WIRELESS SENSOR NETWORKS

Godwin Shen, Sundeep Pattem and Antonio Ortega

Ming Hsieh Department of Electrical Engineering
University of Southern California
godwinsh@usc.edu, pattem@usc.edu, ortega@sipi.usc.edu

ABSTRACT

This work presents a class of unidirectional lifting-based wavelet transforms for an arbitrary communication graph in a wireless sensor network. These transforms are unidirectional in the sense that they are computed as data is forwarded towards the sink on a routing tree. We derive a set of conditions under which a lifting transform is unidirectional, then find the full set of those transforms. Among this set, we construct a unidirectional transform that allows nodes to transform their own data using data forwarded to them from their descendants in the tree and data broadcasted to them from their neighbors not in the tree. This provides a higher quality data representation than existing methods for a fixed communication cost.

Index Terms— Data Compression, Wavelet Transforms, Wireless Sensor Networks

1. INTRODUCTION

Wireless sensor networks (WSN) have garnered much attention given the low-cost of sensor devices (nodes) and their potential for distributed and autonomous operation. One of the main challenges WSN face is that sensor devices are often battery-powered, and so are severely energy-constrained. In order to achieve energy-efficient data gathering in WSNs it is important to study how to effectively exploit spatial data correlation to lower total power consumption. We consider scenarios where a set of samples is captured by sensors in the network. Then, sensors cooperate to transmit this set of samples to a single sink node. The goal is to minimize the total power consumption in the network needed to achieve a given quality for the reconstruction of this set of samples achievable at the sink.

In-network compression (e.g., [1, 2, 3, 4]) can lead to overall lower communication costs and power consumption: sensor nodes compress data they receive from other nodes as they relay it to the sink. This raises the question of how best to organize data gathering through the network. If there was no in-network compression and an equal number of bits were used for each sample, then it would be best to simply gather data through a shortest path routing tree in order to minimize total power consumption. However when using in-network compression it will be necessary to search for the best combination of routing and compression [1, 4]. Radio power levels for each sensor determine which pairs of sensors are in direct communication with each other. We will define an undirected graph where each edge represents one of these communication links and we will focus on designing strategies for data transfer (how data is routed in the graph) and data processing (how sensors compressed data) that can optimize overall power consumption.

This work was supported in part by NASA under grant AIST-05-0081.

In this work we focus on distributed signal transforms, which can be efficient tools for in-network compression. Our work is based on two observations. First, performance of these transforms depends on how well they exploit local correlation, i.e., for maximal spatial de-correlation data sampled in a node should be transformed along with data from all, or most, of its neighboring nodes. However, performing a transform incurs communications cost, since data needs to be exchanged across nodes. For example, the 2D wavelet in [2] de-correlates data using a lifting transform [5] constructed on a graph such as the one we consider here. Nodes are partitioned (split) into even and odd sets by choosing odd nodes that give maximal de-correlation. Then data is filtered across these sets. In terms of de-correlation, this transform is efficient since each odd (even) node transforms its own data using data from all of its even (odd) neighbors. However, this transform is inefficient in terms of the overall number of communications since even nodes must first transmit raw data to odd neighbors, then wait for odd neighbors to transmit transform coefficients back to them and then compute their own transform coefficients and forward them to the sink. This produces significant communication overhead since many nodes are transmitting data twice (and possibly data is transmitted "away" from the sink).

Then, our second observation is that it is best to design transforms with low communications cost, by requiring nodes to transmit data just once[1] and to do so in the direction of the sink. Transforms have been proposed that only require a node to transmit data once. This is achieved by computing the transform as data is forwarded to the sink along a given routing tree. We say that such transforms have *unidirectional operation*. The wavelet transform proposed in [1] has unidirectional operation since each node only transforms its own data using data from neighbors along a 1D routing path to the sink. The tree-based 2D wavelet transform proposed in [3, 4] is constructed on an arbitrary routing tree, resulting in a critically sampled transform (one coefficient per node) that exploits correlation across routing paths and also has unidirectional operation. These transforms are more efficient than that of [2], but nodes only exploit correlation using data from their neighbors in the routing tree, whereas in [2] correlation is exploited using data from all neighbors. In this paper, we develop transform designs that enable more correlation to be exploited across more neighbors (as in [2]), while preserving unidirectional operation as in [3, 4].

Note that both communication cost and the transform itself depend on how data is routed to the sink. Thus we assume that we have a routing tree describing how data from each node flows to the sink. Moreover, since each node transmits its sensed data just once, the time at which transmission occurs determines how this sample can

[1] More precisely, each node communicates once for each data sample it captures, but also relays data from other nodes

be used by other nodes for compression. Thus if a node i schedules to communicate its sample after a neighbor j has already transmitted its own, then node j may not be able to use information from i for coding. Therefore, in addition to the routing tree, we define a transmission schedule that determines the time at which each node is supposed to transmit a sample.

Assuming a fixed communication graph, routing tree, and transmission schedule, the *main goal* of this work is to find (i) all feasible transforms with unidirectional operation, and (ii) ways to select the best transform. To do so, in Section 2 we first establish conditions under which a lifting transform has unidirectional operation. We also find sub-graphs on which these transforms exist. In Section 3, we construct a unidirectional lifting transform on these sub-graphs and provide a unidirectional computation algorithm. This provides a unidirectional transform on a graph that is more general than that proposed in [1, 3, 4] and also exploits more of the existing correlation using data received over links not used for routing, as in [2]. The performance improvements that this transform provides are demonstrated in Section 4. These transforms also have more general relevance since they can be applied to any type of data on any graph.

2. UNIDIRECTIONAL TRANSFORMS ON GRAPHS

Let $G = (V, E)$ be an undirected communication graph of a WSN with N nodes indexed by $n \in \mathcal{I} = \{1, 2, \ldots, N\}$, with the sink node having index $N+1$ and where each edge $(m, n) \in E$ denotes a communication link from node m to node n. Let $T = (V, E_T)$ be a routing tree in G along which data, denoted by $x(n)$, flows towards the sink . Let depth(n) be the number of hops from n to the sink on T and let ρ_n denote the parent of n, $\mathcal{C}_n$ the set of children of n and $\mathcal{D}_n$ the descendants of n in T. Also let $\mathcal{A}_n$ denote the set of nodes that n routes data through to the sink excluding the sink, i.e., ancestors of n. We define a *transform* as a set of linear operations on data x specified by the computations $y(n) = \alpha_0^n x(n) + \sum_{i=1}^{M} \alpha_i^n x(n_i)$ for each node n with some set $\mathcal{N}_n = \{n_1, n_2, \ldots, n_M\}$. In addition, let $y_p(n) = \alpha_0^n x(n) + \sum_{j=1}^{M_p} \alpha_{i_j}^n x(n_{i_j})$ denote the "partial" coefficient of n for some $\mathcal{N}_n^p = \{n_{i_1}, n_{i_2}, \ldots, n_{i_{M_p}}\} \subset \mathcal{N}_n$.

Our goal is to find transforms that have unidirectional operation. To do so, the data a node can use in the transform depends on the order in which nodes transmit data. This is illustrated in Fig. 1. For example, node 2 can use $x(6)$, $x(7)$ and $x(12)$ to compute $y(2)$, e.g., $\mathcal{N}_2 = \{6, 7, 12\}$, since 6, 7 and 12 transmit before 2. Data $x(n)$ can also be processed at any node $m \in \mathcal{A}_n$, i.e., n's parent or grandparent, using $x(m)$. For instance, node 5 has neighbors $\mathcal{N}_5 = \{1, 11\}$ so it can generate $y_p(5) = \alpha_0^5 + \alpha_2^5 x(11)$ and forward $y_p(5)$ and $x(11)$ to node 1. Once node 1 receives $y_p(5)$ it completes the computation as $y(5) = y_p(5) + \alpha_1^5 x(1)$. Note that not all data can be used, i.e., $x(5)$ cannot be used to process $x(10)$ since node 5 transmits after node 10. To make these ideas more precise, we now define a transmission schedule and unidirectional operation.

Definition 1 (Transmission Schedule)**.** *A transmission schedule is a function $t : \mathcal{I} \to \{1, 2, \ldots, M_{slot}\}$, such that i) $t(n) = j$ when node n transmits in the j-th time slot[2] and ii) n transmits data before node m whenever $t(n) < t(m)$.*

[2]Note that these time slots are not necessarily of equal length; they simply allow us to describe the order in which communications proceed in the network; before time $t(n)$, node n is listening to other nodes, and at time $t(n)$ node n starts transmitting its own data, along with data from its descendants in the routing tree.

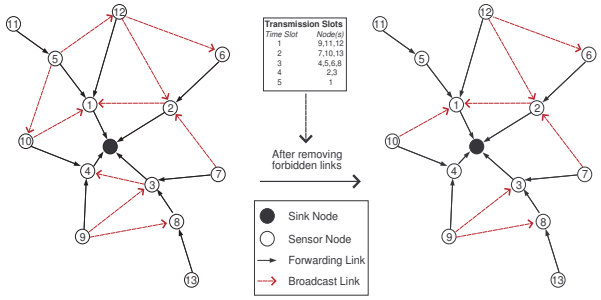


Fig. 1. Toy example to illustrate the theory. Solid arrows indicate "forwarding" links over which data is routed to the sink, dashed arrows indicate "broadcast" links and nodes transmit in the order of the time slot specified.

Definition 2 (Unidirectional Operation)**.** *Let $\mathcal{B}_n = \{k : t(k) < t(n), (k, n) \in E\}$ be the set of one-hop neighbors of n that transmit before n does. We say a transform has unidirectional operation on a routing tree T under transmission schedule t if, for each node n, (i) data is only forwarded along T according to the schedule specified by t, i.e., from n to ρ_n in slot $t(n)$, and (ii) n only transmits the coefficients of its descendants and either $x(n)$ or $y_p(n)$ or $y(n)$.*

This definition allows each node n to use data from any $m \in \mathcal{B}_n$ to transform its own data $x(n)$. Node n can also use data from some node $m \in \mathcal{A}_n$ since either $x(n)$ or $y_p(n)$ will be available at m. We simply delay that processing of $x(n)$, a principle we call "delayed processing". In particular, one of three things happens when n forwards $x(n)$ or $y_p(n)$ to ρ_n: ρ_n either i) completes the computation of $y(n)$ or ii) generates or updates $y_p(n)$ or iii) does nothing. If n forwards $y(n)$, ρ_n also does nothing. This leads to Proposition 1 which provides conditions for unidirectional operation.

Proposition 1. *Let T be a routing tree in a graph G with transmission schedule t. Then a transform has unidirectional operation whenever $y(n) = \alpha_0 x(n) + \sum_{i=1}^{|\mathcal{N}_n|} \alpha_i x(n_i)$ for $\mathcal{N}_n \subset \mathcal{B}_n \cup \mathcal{A}_n$ chosen so that n need not forward data from any $m \in \mathcal{B}_n - \mathcal{D}_n$.*

For a lifting transform, at each level of decomposition, nodes are first partitioned into even and odd sets, $\mathcal{E}$ and $\mathcal{O}$ respectively, with $\mathcal{E} \cap \mathcal{O} = \emptyset$. Each odd node n generates detail coefficient $d(n)$ using data from its even neighbors. Then, each even node m generates smooth coefficient $s(n)$ using coefficients of its odd neighbors. The conditions for a unidirectional lifting transform are presented in Proposition 2. This follows from Definition 2 and Proposition 1.

Proposition 2. *Let T be a routing tree in a graph G with transmission schedule t. Let $\mathcal{E}$ and $\mathcal{O}$ denote the even and odd sets of a lifting transform on nodes of G. This lifting transform is unidirectional on T using schedule t if, for all nodes n, (i) n only forwards coefficients from itself and its descendants, (ii) if $n \in \mathcal{O}$, $d(n) = x(n) + \sum_{i=1}^{|\tilde{\mathcal{N}}_n|} p_i x(i)$ for $\tilde{\mathcal{N}}_n = (\mathcal{B}_n \cup \mathcal{C}_n \cup \{\rho_n\}) \cap \mathcal{E}$, and (iii) if $n \in \mathcal{E}$, $s(n) = x(n) + \sum_{i=1}^{|\tilde{\mathcal{N}}_n|} u_i d(i)$ for $\tilde{\mathcal{N}}_n = (\mathcal{C}_n \cup \{\rho_n\}) \cap \mathcal{O}$.*

A unidirectional multi-level lifting transform is also guaranteed if we apply some split method at each level $j > 1$ to get even and odd sets $\mathcal{E}_j$ and $\mathcal{O}_j$, then re-apply Proposition 2 to the sets $\mathcal{E}_j$ and $\mathcal{O}_j$ on only the smooth coefficients of $\mathcal{E}_{j-1}$.

If we examine the allowable edges for each n, i.e., $E_n = \{(m, n) : m \in \mathcal{B}_n \cup \{\rho_n\}\}$, we see that the sub-graph $G' = (V, \cup_{n \in \mathcal{I}} E_n)$ can be used to define every lifting transform that satisfies Proposition 2. This is formalized in Proposition 3.

Proposition 3. *Let T be a routing tree in a graph G with transmission schedule t. For each $n \in \mathcal{I}$, let $E_n = \{(m,n) : m \in \mathcal{B}_n \cup \{\rho_n\}\}$. Let $E' = \cup_{n\in\mathcal{I}} E_n$. Then a unidirectional lifting transform satisfies Proposition 2 only if it is defined on the sub-graph $G' = (V, E')$. We say that G' contains all such lifting transforms.*

Proof. Suppose a lifting transform, with even and odd sets $\mathcal{E}$ and $\mathcal{O}$ respectively, satisfies Proposition 2. Define $\tilde{\mathcal{N}}_n$ as in Proposition 2 using $\mathcal{E}$ and $\mathcal{O}$. Each odd node n will only use data from $l \in \mathcal{B}_n \cup \mathcal{C}_n \cup \{\rho_n\}$ and clearly $(l,n) \in E'$ for each l. Each even node m only uses data from $k \in \mathcal{C}_m \cup \{\rho_m\}$ and $(k,m) \in E'$ for each k. Therefore, this transform can be constructed on $G' = (V, E')$. □

3. LIFTING TRANSFORM CONSTRUCTION

We achieved our first goal in Section 2. We now study our other goal by proposing a new unidirectional lifting transform. To define a lifting transform we must decide on a splitting rule and filter design strategy. Proposition 3 specifies the set of possible unidirectional lifting transforms. Note that G' contains the lifting transform proposed in [3] since it is constructed exactly on T but it does not necessarily contain the lifting transform in [2] since it is not necessarily unidirectional. However, it will contain unidirectional transforms close to that in [2]. Thus, constructing a transform on G' will combine the benefits of these transforms and eliminate their deficiencies.

3.1. Split Design

We split nodes on the sub-graph G' using a slight modification of the strategy in [2]. In this construction, all nodes are initially unassigned. In a lifting transform, data at odd nodes is predicted using data from even neighbors and residual prediction errors are used to represent their data. Thus, odd nodes are chosen first based on the number of neighbors of each node since using more data tends to produce better predictions (and smaller errors). Thus, we first assign the node in G' with the most neighbors as odd, then assign its neighbors as evens. Then among the remaining unassigned nodes we assign the node in G' with the most neighbors as odd, assign its neighbors as evens, then repeat until all nodes are assigned. This method produces a very uneven split since there are always fewer odds than evens. To enforce a more flexible assignment of parity, we do this until all nodes are assigned, then run this procedure again using only the evens until a certain number of odds is reached. Over L levels of decomposition on G', this produces disjoint sets of odd and even nodes, $\mathcal{O}_j$ and $\mathcal{E}_j$, respectively, for $j = 1, 2, \ldots, L$.

We then apply Proposition 2 to achieve unidirectional operation. This forces nodes to distinguish between "broadcast" links, over which data is only used for transform, and "forwarding" links, over which data is used for both transform and routing. In this case, broadcast links for $n \in \mathcal{O}$ are the $(m,n) \in E'$ such that $m \in (\mathcal{B}_n \cap \mathcal{E}) - \mathcal{C}_n$ and forwarding links are the $(m,n) \in E'$ such that $m \in \mathcal{C}_n \cup \{\rho_n\}$ (with $\mathcal{B}_n$, $\mathcal{C}_n$ and ρ_n defined as in Section 2). The "transform" neighbors of n are $\mathcal{N}_n = (\mathcal{B}_n \cup \mathcal{C}_n \cup \{\rho_n\}) \cap \mathcal{E}$ for $n \in \mathcal{O}$. Moreover, it is clear that there are no broadcast links for $n \in \mathcal{E}$ and the forwarding links are the $(m,n) \in E'$ such that $m \in \mathcal{C}_n \cup \{\rho_n\}$. Also, $\mathcal{N}_n = (\mathcal{C}_n \cup \{\rho_n\}) \cap \mathcal{O}$ for $n \in \mathcal{E}$.

3.2. Filter Design and Computation

Define linear prediction operators $\mathbf{p}_{n,j}$ and update operators $\mathbf{u}_{m,j}$ at nodes $n \in \mathcal{O}_j$ and $m \in \mathcal{E}_j$, respectively. These filters can be designed in a variety of ways. For instance, the prediction and update filters of [3] use simple averaging and smoothing filters and the filters in [2] use prediction filters that can perfectly de-correlate piecewise planar data and update filters that keep the average value of the transform coefficients the same at every level of decomposition. We adopt the latter design since it produces better predictions.

Let $\mathcal{N}_{n,1} = \mathcal{N}_n$ be the constrained set of neighbors defined in Section 3.1 and define neighbors $\mathcal{N}_{n,j}$ for all n and $j = 1, 2, \ldots, L$. Then for each $m \in \mathcal{O}_j$ we compute detail coefficient $d_j(m)$ as:

$$d_j(m) = s_{j-1}(m) + \sum_{k\in\mathcal{N}_{m,j}} \mathbf{p}_{m,j}(k) s_{j-1}(k) \tag{1}$$

and given every $d_j(m)$, for each $n \in \mathcal{E}_j$, we get smooth coefficient $s_j(n)$ computed as follows (note that $s_0(n) = x(n)$):

$$s_j(n) = s_{j-1}(n) + \sum_{l\in\mathcal{N}_{n,j}} \mathbf{u}_{n,j}(l) d_j(l). \tag{2}$$

3.3. Unidirectional Computation Algorithm

For simplicity of presentation, let $\mathcal{O} = \mathcal{O}_1$, $\mathcal{E} = \mathcal{E}_1$, $\mathcal{A}_n = \{\rho_n\}$, $\mathbf{p}_n = \mathbf{p}_{n,1}$ for each $n \in \mathcal{O}$ and $\mathbf{u}_m = \mathbf{u}_{m,1}$ for each $m \in \mathcal{E}$. Also let $d(n) = d_1(n)$ for $n \in \mathcal{O}$ and $s(m) = s_1(m)$ for $m \in \mathcal{E}$. Let this define a lifting transform satisfying Proposition 2. We now describe how to compute the transform as data is routed towards the sink. When an odd (even) node receives all data from evens (odds) in previous time slots, it filters its own data with that data, processes the coefficients of its descendants when necessary, then forwards its coefficients and those of its descendants to the sink. Note that an odd node n may receive data from some $m \notin \mathcal{D}_n$ via broadcast. By Proposition 2, it can use that data to filter its own but it must not forward it to the sink. This is detailed in Algorithm 1.

Algorithm 1 Unidirectional Computation Algorithm

for $m = 1$ to M_{slot} **do**
  $\mathcal{I}_m = \{n \in \mathcal{I} : t(n) = m\}$
  for all $n \in \mathcal{I}_m \cap \mathcal{E}$ (evens in the m-th time slot) **do**
    for all $l \in \mathcal{C}_n \cap \mathcal{O}$ **do**
      $d(l) = d_l + \mathbf{p}_l(n)x(n)$
      For all $k \in \mathcal{C}_l \cap \mathcal{E}$, $s(k) = s(k) + \mathbf{u}_k(l)d(l)$
    end for
    $s(n) = x(n) + \sum_{l\in\mathcal{C}_n\cap\mathcal{O}} \mathbf{u}_n(l)d(l)$
  end for
  for all $n \in \mathcal{I}_m \cap \mathcal{O}$ (odds in the m-th time slot) **do**
    $d(n) = x(n) + \sum_{l\in\mathcal{B}_l\cap\mathcal{E}} \mathbf{p}_n(l)\hat{x}(l)$
    for all $l \in \mathcal{C}_n \cap \mathcal{E}$ **do**
      If $l \in \mathcal{E}$ and $\rho_n \in \mathcal{O}$, $s(l) = s(l) + \mathbf{u}_l(n)d(n)$
    end for
  end for
end for

3.4. Discussion

This transform construction provides greater de-correlation than in [3] and is unidirectional unlike in [2]. There is, however, an intimate connection between T and t and the performance of the transform. We propose the following transmission schedule, which assigns nodes further from (closer to) the sink earlier (later) time slots to provide a natural flow of data towards the sink. In other words, nodes furthest from the sink first forward to their parents in T, then nodes second furthest process their data and forward to their parents, etcetera. Let $\mathcal{L}_m = \{k : \text{depth}(k) = m\}$ and

$d_{max} = \max(\text{depth})$. We first uniquely assign $|\mathcal{L}_{d_{max}}|$ time slots to the nodes in $\mathcal{L}_{d_{max}}$, i.e., each $n \in \mathcal{L}_{d_{max}}$ is assigned a unique time slot $t(n) \in \{1, 2, \ldots, |\mathcal{L}_{d_{max}}|\}$. For each set $\mathcal{L}_m$, we assign time slots to nodes in the same way, where each $n \in \mathcal{L}_m$ is assigned a unique time slot $t(n) \in \{N_m + 1, N_m + 2, \ldots, N_m + |\mathcal{L}_m|\}$, with $N_m = \sum_{i=m+1}^{d_{max}} |\mathcal{L}_i|$. At each depth m, nodes with fewer neighbors are assigned earlier time slots and those with more are assigned later time slots. This allows nodes with many neighbors to receive, and hence utilize, data over most of their available links in G'. Note that, though joint optimization of compression and routing was considered in our previous work [4], better overall performance (in terms of cost, reconstruction quality and transmission delay) may be achieved if compression is jointly optimized with routing and transmission scheduling. This is a topic for future work.

4. EXPERIMENTAL RESULTS

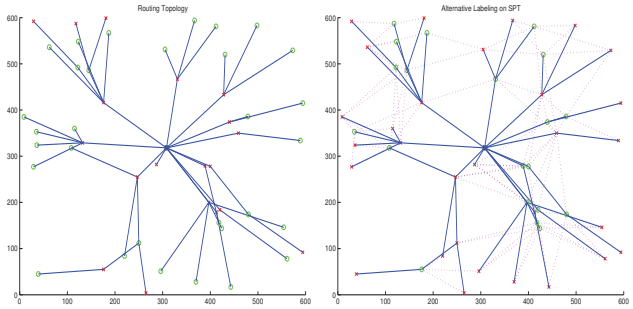
In this section we compare the tree-based wavelet in [3] with the unidirectional lifting transform presented here. As discussed here and as demonstrated in [3], the cost for using the transforms in [1, 2] is higher than the tree based wavelet and so we omit it from our comparisons. For both transforms, we use the predict and update filter design proposed in [2]. We also compare against the use of "delayed processing" for the tree-based wavelet. In particular, since even (odd) depth nodes in the tree are even (odd) in the transform, the grandchildren of each even node will also be even. Since such grandchildren are even nodes themselves, their coefficients are low-pass coefficients. Moreover, every node will have access to the coefficients of its grandchildren. Thus, each even node can apply an additional level of decomposition to the low-pass coefficients of their grandchildren (i.e. more de-correlation) with no added cost. Nodes of depth one can do the same for their even children. This transform is still unidirectional since the coefficient of each even node (for the first level of decomposition) will be completed at its grandparent.

We use an AR-2 model to generate (noise-free) simulation data with high spatial correlation. A randomly generated 50 node network is used with a graph obtained using a fixed radio range at each node. The shortest path routing tree (SPT) is shown in Fig. 2(a). We use the transmission schedule discussed in Section 3.4. The structure of the transform presented here is shown in Fig. 2(b).

Performance comparisons are shown in Fig. 3, which plots energy consumption versus reconstruction quality (in terms of Signal to Quantization Noise Ratio). Energy consumption is modeled as in [6]. Each point corresponds to a different quantization level with sequential entropy coding applied to coefficients at each node. The tree-based wavelet has worst performance, but improves significantly when adding delayed processing. The transform proposed here does best since it exploits more correlation and is unidirectional.

5. CONCLUSIONS

Given an arbitrary communication graph and a routing tree, we have defined a set of conditions under which a lifting transform is unidirectional. A sub-graph which contains all such transforms was also found and a lifting transform was constructed that exploits most of the correlation in the network by allowing nodes to use data they are responsible for forwarding and data they receive via broadcast. Experimental results show performance improvements with respect to a lifting transform computed only along a routing tree. As future work, we can consider other problems including selecting the tree, transmission schedule and transform jointly for a given graph.



(a) Transform on SPT (b) Transform on Graph

Fig. 2. Transform definition on SPT and on graph. Circles denote even nodes and x's denote odd nodes. The sink is shown in the center as a square. Solid lines represent forwarding links. Dashed lines denote broadcast links.

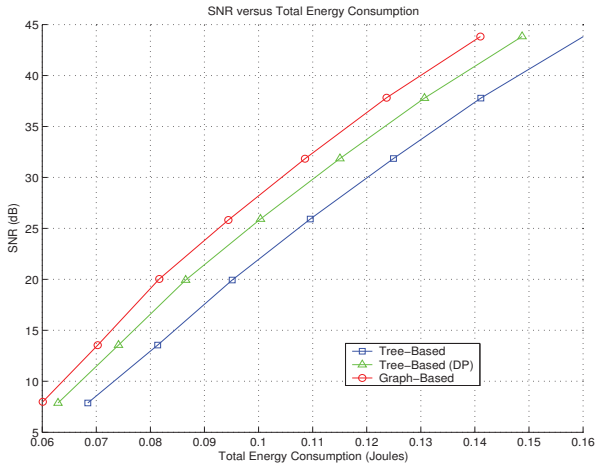


Fig. 3. Performance comparisons.

6. REFERENCES

[1] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari, "Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm," in *IPSN '06*, April 2006.

[2] R. Wagner, R. Baraniuk, S. Du, D.B. Johnson, and A. Cohen, "An architecture for distributed wavelet analysis and processing in sensor networks," in *IPSN '06*, April 2006.

[3] G. Shen and A. Ortega, "Optimized distributed 2D transforms for irregularly sampled sensor network grids using wavelet lifting," in *Proc. of ICASSP'08*, April 2008.

[4] G. Shen and A. Ortega, "Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting," in *IPSN '08*, April 2008.

[5] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," Tech. report 1995:6, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, (ftp://ftp.math.sc.edu/pub/imi_95/imi95_6.ps), 1995.

[6] A. Wang and A. Chandraksan, "Energy-efficient DSPs for wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 4, pp. 68–78, July 2002.