Aggregation in Sensor Networks: Optimally Trading Information for Energy

Jen Burge Kamesh Munagala
Department of Computer Science
Duke University
jen,kamesh@cs.duke.edu

Romit Roy Choudhury
Dept. of Electrical Engineering
Duke University
romit@ee.duke.edu

Abstract

In this paper, we consider algorithmic issues in employing lossy compression in order to extend the lifetime of wireless sensor networks. We consider two metrics which trade-off with each other: the communication and computation cost on one hand, and the information loss from lossy compression on the other. We consider the particular setting where the lossy compression scheme is simple averaging. This compression scheme leads to lower computation cost at a node compared to expensive loss-less compression schemes, and in addition saves on the communication cost. However, such a scheme also loses significant information, and must only be employed when node values are highly correlated.

We study the problem of deciding the appropriate combination of loss-less and lossy compression schemes to implement along an aggregation tree. The goal is to optimize information loss given a budget on the communication and computation costs. Our algorithmic framework is fairly general and handles various types of cost functions and information loss measures. We perform extensive empirical studies to validate the need for such an algorithmic framework when network lifetime is highly constrained.

1. Introduction

Sensor networks are becoming increasingly useful for monitoring and studying a wide variety of physical phenomena. Wireless networks of sensors allow researchers to take measurements over large spatial and temporal scales without disturbing the surrounding environment. One important limitation of a sensor network is the battery life of the sensors, and frequent visits to the site to change batteries are often impractical. Therefore one of the major goals in sensor networks research is increasing energy efficiency in order to extend the lifetime of networks.

Physical phenomena such as temperature, humidity, or light are inherently continuous and therefore exhibit a high degree of spatial correlation. This correlation can be exploited to save energy by reducing the transmission of redundant information. The method of choice is to allow nodes in the network to compress information they have received before sending it on using some lossless compression scheme, such as Lempel-Ziv or distributed source coding[12, 10]. This strategy trades off computation with communication by reducing the size of the message that needs to be sent. Usually a node will spend several orders of magnitude less energy on computation than on communication, so this technique is able to save a significant amount of energy if the data from different sources is correlated. Compression saves communication cost all along the path to the base, as each node has a smaller message to transmit.

However, compression is limited in its potential for energy savings. Loss-less compression is one end of a spectrum, which preserves all of the information but has the highest communication cost of any aggregation scheme. The other end of the spectrum uses lossy compression (such as averaging) at every node in the network, which minimizes the communication cost but loses the most information. If the network is nearing the end of its lifetime, loss-less compression may not save enough energy to extend the network's lifetime to the desired length. If we are willing to tolerate inaccuracies, then lossy compression, or aggregation, has the potential to save more communication. Averaging the values in a subtree reduces the size of the message to approximately the size of the data from just one source, independent of the original size of the message. If the nodes are highly correlated, this will lose less information. Averaging also has a much smaller computation cost than any loss-less compression scheme.

Another scenario where aggregation is important is a group of highly correlated nodes connected to the rest of the network by some edge which has a high communication cost. An edge may be expensive in terms of communication cost for a number of reasons: The two endpoints of the edge may be at a large geographic distance from each other, requiring the sender to use a lot of energy just so the message will reach the receiver. Even if the nodes are relatively

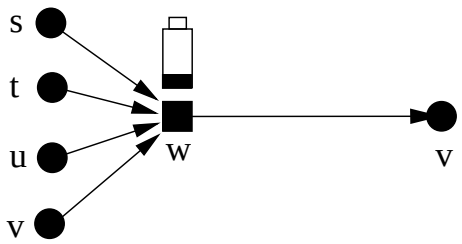


Figure 1. An expensive edge may require aggregation to reduce high-cost communication.

close together, they still may have an expensive edge between them. For instance, if the transmitting node has little battery life remaining, it may not have enough power to reliably reach the receiver, resulting in a lot of retransmissions. Avoiding a lot of such transmissions is necessary in order to keep the network connected longer. As another instance, an edge could be expensive if there is some physical obstruction between the two nodes which causes more messages to fail and require retransmission. In any of these scenarios, we must carefully manage the communication across this edge in order to prevent these nodes from getting disconnected from the network.

Consider the example depicted in Figure 1. There is a small group of nodes very close together, and therefore very highly correlated. The parent of w in the communication tree, x, is connected by an expensive edge, because the battery life of w is very low. By averaging the values from $s, t, u, v,$ and w, we can reduce the communication cost on this expensive edge by $1/5$ over forwarding, with only a small loss of information. Loss-less compression is unlikely to achieve as large a reduction, and is more expensive computationally than averaging. Sending the average will allow node w to use less energy and stay alive longer, preventing $s, t, u,$ and v from becoming disconnected and useless.

We consider combining lossless compression with lossy compression in order to maximize the lifetime of a densely deployed sensor network measuring some continuous physical phenomenon. Our goal is to provide a systematic approach that trades off communication cost with information loss. Given a set of sensors, a communication tree rooted at a base station, and a communication cost budget, we find the appropriate action from $\{compress, average, forward\}$ for each node, such that the total communication cost is less than the budget and the information loss is minimized. Examples of strategies are illustrated in Figures 6,7, and 8. The communication cost budget is determined based on the current battery life of the sensors and the desired lifetime of the network. We will describe our algorithm in the setting where all nodes start with the same battery life, but it is easily adapted to the case of heterogeneous battery lives by assigning appropriate weights to the nodes, as explained in Section 3.5.

Our contributions include

- A model for measuring information loss.
- A dynamic programming algorithm that finds the optimal locations for compression and aggregation for a given network and communication cost budget.
- A distributed implementation of the algorithm.
- An experimental evaluation of the effects of computation cost, correlation, communication tree and communication cost on the information/energy tradeoff.

1.1. Related Work

Energy efficiency in sensor networks is a very active area of research, mainly focussing on communication cost where most energy savings are possible. Our effort integrates computation cost and information loss into this research direction. Though both loss-less and lossy compression schemes have been employed before in this context, the optimal combination of the two schemes has not been looked at previously. Though lossy aggregation schemes such as averaging have been studied previously [3, 8, 9], this has been mostly in the context of implementing aggregate queries over a network. This work has therefore ignored the aspect of information loss which occurs if the average is viewed as an approximate representation of the actual values.

Previous work [7, 11] has considered different in-network loss-less compression techniques, and other work[13, 10] has considered the interaction of such compression with routing. Pattem *et al* [10] analyze the interaction between correlation and the routing/compression algorithms, and show that shortest path trees with opportunistic compression are nearly optimal for a wide range of correlation parameters. Though the overarching question we address still concerns how correlation effects routing, we differ in several regards: First, unlike the hop count metric used in [10], we consider vastly more general cost metrics that take into account the effect of edge lengths on power requirements, as well as battery life and obstacles. Such a cost metric would now be unrelated to the correlation structure in the nodes, which leads to different optimal routing algorithms. Second, our problem formulation takes into account lossy compression schemes and the associated information loss, as well as the computation cost for the compress and decompress operations.

The idea of giving up accuracy to save energy has been explored before by modeling the underlying process generating the sensor values and only sending the values that

differ too much from the predictions[1, 14]. Our work can be thought of as asking a higher level question: Given any such lossy compression scheme, what are the optimal places in the network for implementing these schemes.

We finally note that a line of work[4, 6] has addressed the problem of optimal placement of sensor nodes given a spatial correlation model. We use the same Gaussian correlation model to motivate our problem statement and develop tractable information loss measures. However, our main focus is on optimal placement of compression schemes given a routing tree, and not in sensor node placement, which we assume is given.

1.2. Outline of the Paper

The paper is organized as follows. Section 2 discusses possible methods for estimating the various parameters needed by the algorithm, including communication cost, computation cost and information loss. Section 3 describes the algorithm and possible generalizations and improvements. Section 4 outlines our experimental evaluation of the algorithm and parameters, and presents the results.

2. System Model

We are given a set of locations V where we need to take or estimate measurements, and sensors are placed at some subset W of those locations. We are also given a communication tree T on the sensor locations in W, rooted at the base station. Every node in this tree needs to send the data it collects to the base station. When an interior node in the tree receives data from its children, it has three options: forward, compress, or aggregate. If the node chooses to compress or aggregate, any incoming data that has been compressed must first be uncompressed. Aggregation will produce the smallest output (the size of one measurement if we are averaging) but will result in some loss of information. The aggregation function could be any lossy compression scheme for which we can estimate the information loss. In our work we will consider averaging for its simplicity and robustness.

A solution to this problem is an assignment of forward, aggregate or compress to each node in the tree, which adheres to our constraints. The goal is to find a solution that minimizes the total information loss, given some communication cost budget.

Our algorithm requires some way of estimating communication costs, computation costs and information loss. Communication and computation costs can be determined locally by nodes using profiling tools, or estimated using an appropriate model. Information loss requires a model or global estimation, since the correlation between every pair of nodes in the network is needed. Because global estimation would be expensive, the use of an appropriate model is likely to be the preferred method for estimating information loss. We describe a widely accepted model for spatial correlation in Section 2.2.

2.1. Cost Model

For the purposes of our algorithm, we require a model for both communication and computation costs. We model communication cost as a general function of the size of a message and the distance it travels. We model the computation cost of compressing and decompressing as functions of the initial size of the message being compressed or uncompressed.

Communication Cost: The communication cost of a solution is the sum of the communication costs incurred on each edge. The communication cost on edge e is some function of the size of the message being transmitted across e, b_e, and the distance between the endpoints of e. This function should provide an accurate estimate of the power required to transmit the message across e, and may be defined to incorporate occlusion, battery life, the probability of failures and retransmissions, and any other factors effecting the amount of energy used when transmitting on the edge.

For the purposes of our experiments, we model the communication cost on edge e as the size of the message being transmitted, b_e, times the square of $l(e)$, the length of edge e. Our algorithm is not dependent on the particular communication cost function. The square of the distance is a reasonable function because it is reflective of transmission costs for an outdoor sensor network; for indoor sensor networks the cost would increase as the fourth power of the distance.

$$COST = \sum_{e \in T} b_e l(e)^2$$

Computation Cost: The computation costs for compressing and decompressing the values from a set of sensors could be estimated in three different ways. The costs could be measured before deployment, estimated by each sensor whenever it compresses, or predicted using a model.

Sadler and Martonosi[12] showed experimentally that computation costs are often significantly lower than communication costs, even for expensive loss-less compression routines. This means it is usually a good idea to spend computation energy on compression if it will reduce the size of the message at all. However, when the cost of decompressing a nodes inputs in order to compress them all together is considered, the question of where to compress becomes more interesting.

For our experiments we use a simple model that assumes a fixed constant ratio of communication cost to computation

cost for a set of a particular size. We discuss the effects of varying this ratio in section 4.2

2.2. Correlation Model

Any algorithm requires estimates of the correlations between nodes in order to calculate the information loss of averaging a set of sensor readings. Instead of global estimation of the correlation between each pair of sources, a model of the joint distribution of the sources can be used to estimate the correlations. The model can also be used to estimate the compressed size of a set of readings. We will leave the question of efficient estimation of these parameters for future work, and use a model of the sensors that is standard in the literature. This model assumes that the values are drawn from a multivariate Gaussian distribution. For an n-dimensional Gaussian, each variable X has

$$P(X = x) = \frac{1}{(2\pi)^{n/2}|\Sigma|} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

where Σ is the $n \times n$ covariance matrix, and μ is the length n vector of means.

Since we are dealing with a continuous physical phenomena that occurs at all points in space and not just at the locations where we have placed sensors, we use a generalization of the multivariate Gaussian to an infinite number of variables, called a Gaussian process. A Gaussian process is specified by a mean function $\mu(.)$ and a kernel function $\Sigma(.,.)$, which give the mean for any given location and the covariance between any two locations. The covariance between the values at locations u and v is a function of the distance between them and is given by

$$\Sigma(u, v) = \sigma_u \sigma_v e^{-d(u,v)^2/h^2}$$

where h is a constant parameter that is learned from observations. Our algorithm relies only on this covariance model, which is independent of the means μ_u, μ_v, so we set $\mu(u) = 0$ for all nodes u.

2.3. Compression Size

We calculate the compressed size of a set of sources S using the differential entropy of the set, $H(S)$, which measures the amount of uncorrelated information in the set. Since the measurements are actually continuous random variables, the differential entropy could be infinite, so we need to use the entropy of the discretization of these variables. If the data is discretized to a precision of Δ, the compressed size of S is approximately $DE(S) + n \log(1/\Delta)$.

For a multivariate Gaussian the differential entropy is given by

$$DE(S) = \frac{1}{2} \log((2\pi e)^n \det \Sigma)$$

where Σ is the covariance matrix.

2.4. Information Loss

Quantifying the amount of information lost during aggregation is a difficult challenge in our work. There are two properties we require from an information loss function $f(S)$, which quantifies the amount of information lost when aggregating a set of values S:

- *Non-negativity* $f(S) \geq 0 \quad \forall S$
- *Non-decreasing* $f(A) \leq f(S) \quad \forall A \subseteq S$

One initial attempt is to use the residual information, $f(S) = H(S|\mathsf{avg}(S))$, which would quantify the uncertainty left when using the average of S to try to recover S. However, this could be negative, and does not account for the extra information S could give us about the entire set of values V that $\mathsf{avg}(S)$ does not give us.

To fix this problem we defined *information loss* as

$$IL(S) = MI(V \setminus S, S) - MI(V, \mathsf{avg}(S)) \quad (1)$$

The function $MI(X, Y)$ is the *mutual information* between two sets X and Y, as defined in [2]. This function is a more robust measure of how well set Y can be used to reconstruct set X, and will always be non-negative. In the case where Y is just the average of X, the mutual information is given by

$$MI(X, \mathsf{avg}(X)) = DE(X) - DE(X|\mathsf{avg}(X)) \quad (2)$$

The information loss function is only positive when S is small in relation to V (note that $MI(V \setminus S; V) = 0$ when $S = V$). For this reason we need to use a set of locations V to compute information loss which is larger than the set of all sensor locations W.

To calculate $DE(X|\mathsf{avg}X)$, we use the chain rule of entropies.

$$DE(X_1, X_2, ...X_m|Y) = \sum_{i=1}^{m} DE(X_i|X_1...X_{i-1}, Y) \quad (3)$$

The differential entropy for a single random variable X_i given a set of other random variables A is

$$DE(X_i|A) = \frac{1}{2} \log(2\pi e(\sigma^2_{X_i} - \Sigma_{X_i A} \Sigma^{-1}_{AA} \Sigma^T_{X_i A})) \quad (4)$$

where $\Sigma_{X_i A}$ is the covariance vector with entries $Cov(X_i, a)$ for each $a \in A$, and Σ_{AA} is the covariance matrix restricted to set A.

Using equations (3) and (4) to find $DE(X|\mathsf{avg}X)$ requires the computation of the correlations between values

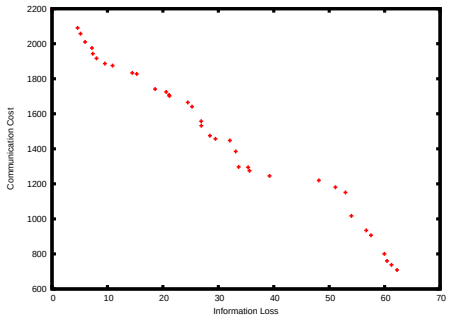


Figure 2. Pareto optimal information loss/communication cost tradeoff points. From any point on this graph any decrease in communication cost requires an increase in information loss and any decrease in information loss requires an increase in communication cost.

in X and the random variable $Y = \sum_{i=1}^{m} X_i/m$ for the covariance vector and covariance matrix. The sum of Gaussian random variables, and therefore also the average, is Gaussian. The covariance between X_j and Y is given by

$$Cov\left(X_j, \frac{1}{m}\sum_{i=1}^{m} X_i\right) = \frac{1}{m}\sum_{i=1}^{m} Cov(X_j, X_i) \qquad (5)$$

Equation (5) combined with equations (1) through (4) allow us to compute $IL(S)$ using the covariance matrix Σ described in Section 2.2.

3. Dynamic Programming Algorithm

Recall that we are trying to find a strategy that assigns an action from $\{compress, average, forward\}$ to each node, in order to minimize the information loss while respecting the communication cost budget. We are trying to compute the Pareto optimal points shown in Figure 2. These points can be computed either by fixing a communication cost and minimizing information loss or fixing an information loss threshold and minimizing communication cost. Since the latter is simpler to work with, we will flip around the problem and minimize communication cost subject to an information loss threshold.

The algorithm to compute a strategy for a given network uses bottom-up dynamic programming, with the table storage distributed across all nodes. Each node will determine all of the points at which a change in the strategy for its subtree results in a smaller communication cost with more information loss. We will call these Pareto optimal points the *information loss thresholds*. These thresholds are sent to the parent of the node, which computes its own thresholds. The algorithm makes the assumption that information loss across different children is additive. This may be overestimating the information loss, since the values in one subtree may be used to reconstruct the values in another subtree. However, if the tree is constructed using some distance-based method such as minimum spanning tree or shortest path tree, the sensors in different subtrees will be farther away and thus likely to be uncorrelated or only weakly correlated, so that using values from another subtree may not help much with reconstruction. Finally, the base station can compute the set of thresholds for the entire network, which gives it a list of communication cost values and the associated minimum information loss. When given a communication cost budget, the base station looks up the point with the largest communication cost that falls under the budget and uses this strategy.

The information loss thresholds are a list of pairs $p_1, p_2, ...p_m$, where $p_i = (loss_i, cost_i)$. The list is sorted in increasing order of $loss_i$ and decreasing order of $cost_i$, representing places where an increase in information loss will decrease the communication cost. For pair p_i, $loss_i$ is the minimum possible information loss for the corresponding communication cost, $cost_i$. There are a finite number of these pairs since each pair corresponds to a certain strategy, and the information loss only changes when the set of aggregating nodes changes. Therefore the maximum possible number of thresholds is 2^n, where n is the size of W, the set of locations with sensors.

Assumption: We make the assumption that when a node receives any average values, it cannot perform compression. This assumption is necessary in order to solve the problem with dynamic programming. The assumption is reasonable, because compression of averages from different subtrees is unlikely to achieve a significant further reduction in size. The assumption allows us to introduce three cases representing the three distinct situations that node v may be in.

- **Case A:** Some ancestor of v will aggregate, but no ancestor of v will compress. In this case, v must consider that the cost of uncompressing will be payed later if it chooses to compress.
- **Case C:** Some ancestor of v will compress. Then v must not only consider the uncompress cost but also is not allowed to aggregate.
- **Case N:** Every node along the path from v to the base station will forward. In this case v is free to choose any of the three actions and will pay no uncompress cost.

For every case i, a table of information loss thresholds, T_v^i, is computed and stored at each node. These thresholds enumerate the levels of information loss where the strategy of the network changes in a way that lowers communication

cost by allowing the loss of more information. A row of the table is a tuple

$$\langle loss, cost, size, action, t_1, t_2, ...t_{deg(v)} \rangle$$

where $deg(v)$ is the number of children v has and t_j is the corresponding threshold for the jth child. After the base station has computed its tables, the strategy is propagated down by passing the appropriate case number and along with t_j from the parent to each child, which the child uses to choose its action.

The leaves of the tree always compress. For cases A and C, a leaf's table consists of one threshold with loss 0 and cost

$$csize(1) * d(l, p(l))^2 + 2 * ccost(1)$$

where $csize(1)$ is the compressed size of a single source, $d(l, p(l))$ is the distance from the leaf to its parent, and $ccost(1)$ is the cost of compressing or uncompressing one source. For case N the cost of the threshold is only $csize(1) * d(l, p(l))^2 + ccost(1)$.

After computing its tables a node sends them to its parent. When the parent, v, receives the tables of all its children, it computes a temporary sum table S_i for each of the three cases by considering all possible combinations of the information loss thresholds of its children. S_i holds the combinations of thresholds when v passes down case i to its children. The sum tables start with a single entry each, computed for an imaginary leaf v' with v as its parent (with $d(v', v) = 0$), to account for v's own measurement. Then for each child j, each entry $T_v^i(t)$ is combined with $T_j^i(1)$. The two thresholds are combined by summing the information loss, communication cost, and size, and adding 1 to the end of split $T_v^i(t)$. Then for each additional entry $T_j^i(u)$ a new threshold is created by combining $T_v^i(t)$ with $T_j^i(u)$.

Node v uses the sum tables to compute thresholds for its own tables. Each table is stored in increasing order of information loss (the first threshold will always have zero information loss). When v attempts to insert a new threshold with loss l_{new} and cost c_{new} into a table, the insertion position j is found so that $T_v^i(j).loss < l_{new}$ and $T_v^i(j+1).loss > l_{new}$. Then there are three different situations.

- If $c_{new} > T_v^i(j).cost$, the new threshold is thrown away, because it represents an increase in both information loss and communication cost.
- If $c_{new} < T_v^i(j+1).cost$, the new threshold replaces $T_v^i(j+1)$, because it has both smaller information loss and smaller communication cost than $T_v^i(j+1)$.
- Otherwise, the new threshold's cost falls between $T_v^i(j).cost$ and $T_v^i(j+1).cost$, as does the information loss. In this case, the new threshold is inserted after $T_v^i(j)$.

Variable	Definition
$d(u, v)$	Distance between nodes u and v
T_v^i	Table of thresholds at node v for case i
S_i	Table of sums of $v's$ children's thresholds
$Sources(v)$	Set of sources in subtree rooted at v
$csize(S)$	Compressed size of set S
$ccost(S)$	Computation cost of compressing S

Figure 3. Notation used in the algorithm description.

3.1. Case C

The simplest table to compute is the table for case C. In this case, v and all nodes in its subtree are not allowed to aggregate. This means there is never any information loss in the subtree and the table only has one entry.

The node needs to determine whether compressing or forwarding is cheaper and enter the appropriate threshold. The cost of forwarding is

$$S_C(1).size * d(v, p(v))^2 + S_C(1).cost$$

and the cost of compressing is

$$csize(Sources(v)) * d(v, p(v))^2 + 2 * ccost(Sources(v)) + S_C(1).cost$$

If compressing is cheaper,

$$T_v^C = \langle 0, csize(Sources(v)) * d(v, p(v))^2 + 2 * ccost(Sources(v)) + S_C(1).cost, csize(Sources(v)),' C', 1, 1, ... \rangle$$

The string of ones at the end represents the threshold numbers passed to each of the children. Since they will each receive case 2, these point to the only entry in T_w^C, for child w. If forwarding is cheaper,

$$T_v^C = \langle 0, S_C(1).size * d(v, p(v))^2 + S_C(1).cost, S_C(1).size,' F', 1, 1... \rangle$$

3.2. Cases A and N

In both cases A and N, v has to choose between the three actions: forward, compress, aggregate. The difference between these two cases is an extra $ccost$ must be added to the cost of any threshold in case A if v chooses to compress.

- *Aggregate* In both cases, if v chooses to aggregate, case A will be passed to its children, and the information loss will be $f(Sources(v))$, using the information loss function we defined in Section 2.4. This loss will be

larger than the sum of the losses if all of v's children aggregate. If there are k thresholds in S_A, the full entry into T_v^A and T_v^N is given by

$$\langle f(Sources(v)), a * d(v, p(v))^2 + S_A(k).cost, a,' A', split(S_A(k))\rangle$$

where a is the aggregate size (equivalent to the size of one datum) and $split(S_A(k))$ is the last $d(v)$ entries of $S_A(k)$, which give the information loss thresholds to be passed to the children.

- *Compress* In both cases, if v chooses to compress, case C will be passed to the children. The threshold added to T_v^N will be

$$\langle 0, csize(Sources(v)) * d(v, p(v))^2 + ccost(Sources(v)) + S_C(1).cost, csize(Sources(v)),' C', split(S_C(1))\rangle$$

The same threshold is added to T_v^A, but with an additional $ccost(Sources(v))$ added to the cost.

- *Forward* If v chooses to forward, whichever case was passed to v gets passed on to its children. For case i, we try to add a new threshold to T_v^i for every sum in S_i. For the tth sum in S_i, we add

$$\langle S_i(t).loss, S_i(t).size * d(v, p(v))^2 + S_i(t).cost, S_i(t).size,' F', S_i(t).split\rangle$$

3.3. Correctness of the Dynamic Program

We will now argue that for a certain information loss threshold, our algorithm will find the strategy that minimizes communication cost. We will prove this for a centralized dynamic program with the information loss threshold as a parameter. This accomplishes the same thing as the distributed algorithm described above but it is easier to reason about.

Let $COST(u, \alpha, \lambda)$ be the minimum cost for the subtree rooted at u for case α with the information loss threshold λ. Let $SIZE(u, \alpha, \lambda)$ be the size of the message transmitted by u in the minimum cost strategy. The base case occurs at the leaves, where the only option is to compress, no matter the case or threshold. We set $d(base, p(base)) = 0$, so the minimum cost action at the base is always to forward.

Let $\Gamma(v)$ be the set of all children of the node v. Node v takes the threshold λ and considers all possible ways to split this among its children, given some discretization level. Let the information loss given to child w by the jth split be $LOSS_j(w)$. The minimum cost for node v, given case α and threshold λ, is

$$COST(v, \alpha, \lambda) = \min_{\texttt{splits}_j} \{$$
$$asize * d(v, p(v)) + \sum_{w \in \Gamma(v)} COST(w, A, LOSS_j(w)),$$
$$csize * d(v, p(v)) + \sum_{w \in \Gamma(v)} COST(w, C, LOSS_j(w)),$$
$$(\sum_{w \in \Gamma(v)} SIZE(w, \alpha, LOSS_j(w)) * d(v, p(v)) + \sum_{w \in \Gamma(v)} COST(w, \alpha, LOSS_j(w)\}$$

If $COST(w, \alpha, \lambda)$ gives the minimum cost for the subtree rooted at w when w receives case α and information loss λ, then this formula finds the minimum cost action for each split and finds the minimum over all possible splits. This is the minimum communication cost for the subtree, for the particular case and information loss. Our distributed algorithm removes the need for a discretization level on the thresholds by building them from the bottom up, so that v knows exactly which splits will change the outcome for its children.

3.4. Distributed Implementation

The algorithm we have described is easily implemented in a distributed manner. Each sensor node is responsible for storing its own table of thresholds. The algorithm requires one bottom-up sweep, where each node calculates its table and then forwards the table to its parent. Then the parent is able to combine the tables of its children to create its own table and these thresholds propagate up to the base station. Then there is a top-down sweep when the appropriate threshold is selected by the base station (based on the communication cost budget) and the (case, threshold number) pairs are sent from parent to child. Each node looks up the appropriate action from its tables using the case and threshold number it receives.

Although the strategy only has to be computed once unless cost estimates or covariances change and the storage of the thresholds is distributed across the network, there could potentially be a large number of thresholds and the number of thresholds can increase rapidly as the network size increases. One of the following methods can be employed to reduce the communication overhead:

- Each node compresses its threshold tables. It will only have to uncompress them when a change in the communication cost budget causes a change in the network strategy.

- Each node filters out very similar thresholds. Set some precision ϵ for the cost thresholds, and if one threshold has cost less than ϵ better than its predecessor in the threshold list, it will be thrown out (since it also has higher information loss).

3.5. Extensions

The algorithm we have presented can be extended in a few straightforward ways.

- **Temporal Correlations**: We focus on a strategy for sending a single round of measurements to the base station. We will ignore temporal correlations, which present the potential for more compression or aggregation. One approach to exploiting temporal correlations is to allow a node to average its time series using a certain block size B. Our algorithm could be extended to handle this case by replacing each node with B nodes, one for each of B time-steps. The distances between these nodes could be set to 0 and the correlation coefficients set according to the temporal correlation of the values seen at the particular node.

- **Heterogeneous Battery Life**: If the battery life of nodes varies widely across the network, an algorithm should take this into account when attempting to extend network lifetime by favoring nodes with a lot of remaining battery life over those that are almost dead. Our framework can be easily adapted to handle this by assigning weights to the edges that make edges incident to dying nodes more expensive. This case then becomes a special case of occlusion, discussed in Section 4.4.

4. Experiments

We have run a series of experiments using MATLAB to examine the trade-off between information loss and communication cost using synthetic data. The sensor locations are chosen uniformly at random from a 10 by 10 square. The rest of the parameters are chosen as follows:

- The values are drawn from a multivariate Gaussian, so information loss and compressed size can be calculated as discussed in Sections 2.4 and 2.3.

- Communication cost for sending a message of size x on edge e is $xl(e)^2$.

- Cost of compressing S = cost of decompressing S = $1/CR*$ communication cost of S on edge of length 1, where CR is a constant cost ratio between communication cost and compression cost.

- The cost ratio is $CR = 15$ unless stated otherwise.

- The precision for discretization is $\Delta = .01$.

- The correlation parameter is $h = 1$ unless stated otherwise.

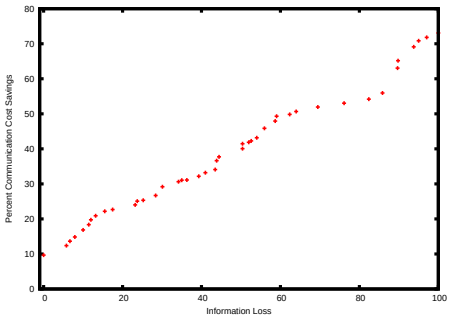


Figure 4. Percent communication cost savings for a set of 80 sensors.

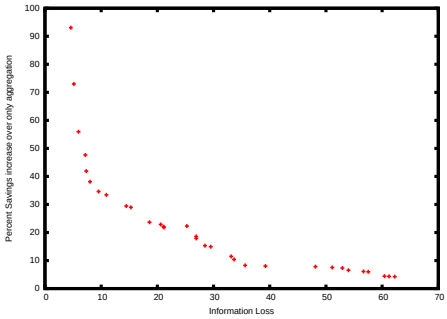


Figure 5. The percent increase in savings we achieve by using compression and aggregation, over using aggregation alone.

- The communication tree is a shortest path tree unless stated otherwise.

In the following experiments we will explore the effects of varying some of these parameters: CR, h, and the communication tree, as well as the impact of occlusion, where an edge may be more expensive than its distance alone would indicate.

4.1. Information Loss vs. Cost

Our first experiments simply validate the need to combine aggregation and compression by showing the potential for savings over either compression or aggregation alone. Figure 4 illustrates the tradeoff between information loss and communication cost savings for a particular 80 node network. The percent savings are relative to the cost of transmitting the data from each node to the base station with no compression scheme. The point where information loss is zero represents the total savings possible with compression alone, which is only about 10%. By allowing only about 15% information loss, this savings can be increased

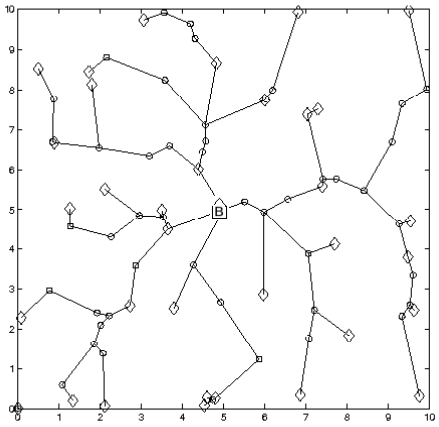


Figure 6. The strategy if we require all the information, regardless of communication cost. There is no aggregation, and thus zero information loss. Diamonds indicate compression and circles indicate forwarding. This strategy saves 3.8% over the maximum communication cost–the cost of a strategy with no compression at all.

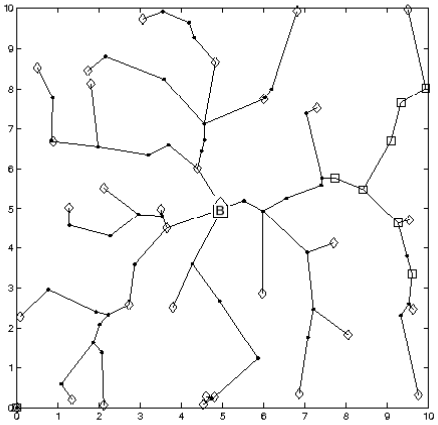


Figure 7. This strategy minimizes the information loss if we can only pay 83% of the communication cost. Squares indicate the points of aggregation.

to 20%. Figure 5 shows that a significant portion of the savings from Figure 4 comes from the use of compression in addition to aggregation, especially for low information loss tolerance.

The trees in Figures 6, 7, and 8 illustrate the locations in the network where we start to perform aggregation, as the communication cost budget is reduced. The diamonds indicate the points of compression and the squares indicated the points of aggregation. As we start to aggregate more, the number of internal nodes that choose to compress is reduced, because of our restriction on the compression of aggregate data.

4.2. Cost Ratio

The dynamic program allows the network to find the optimal tradeoff between the communication cost savings of compression and the computation cost for compressing and decompressing. We have experimented with a few different ratios of communication cost to compression cost. Figure 9 illustrates that the communication cost savings increase for a particular information loss as compression becomes cheaper. When computation is cheap compared to communication, the network will choose to compress at more internal nodes. As computation becomes more expensive, it will become too expensive to compress more than once on

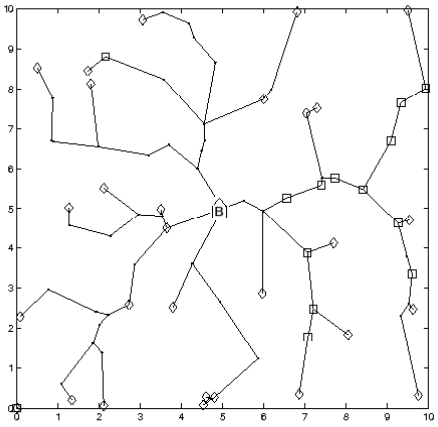
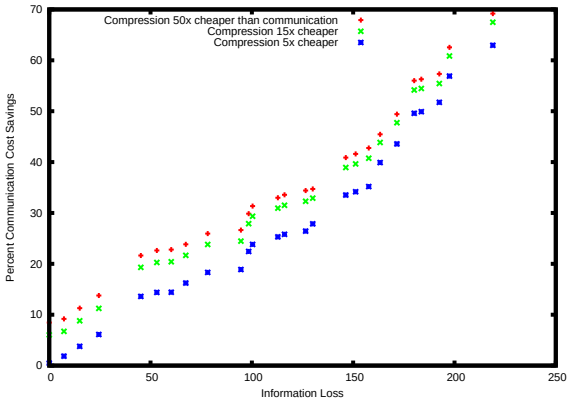


Figure 8. A third strategy with a lower communication cost budget, only 75% of the maximum communication cost.



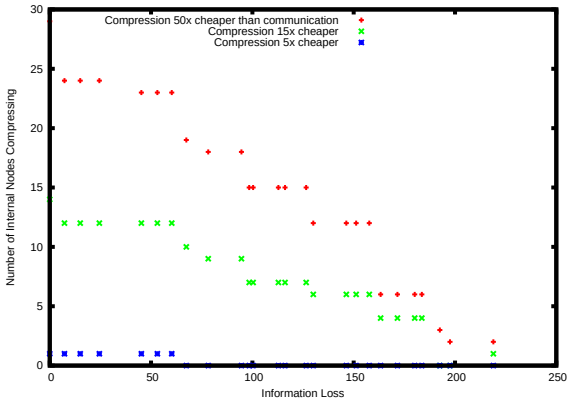


Figure 9. (a) Effects of varying the ratio of communication cost to computation cost in an 80 node network. (b) The number of internal nodes compressing for different cost ratios.

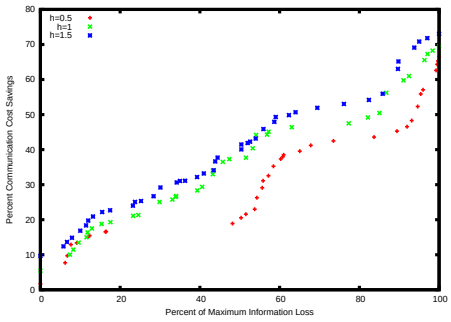


Figure 10. Effects of varying the correlation parameter in an 80 node network.

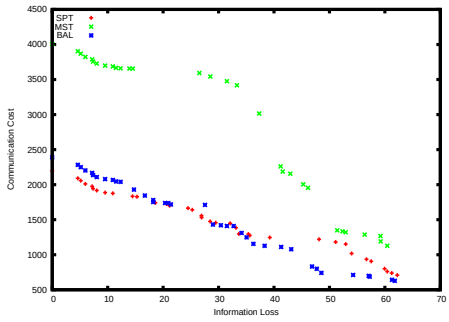


Figure 11. Comparison of MST,SPT and BAL on 80 nodes. Performance of MST is much worse than SPT.

a path to the base station, as shown in Figure 9.

4.3. Communication Tree

The communication tree makes a large difference both in the communication cost and information loss. Ideally the tree should connect highly correlated nodes to provide opportunities for good compression and low information loss aggregation while providing short paths to the base station to keep communication costs low. We experimented with three different communication trees. The first was the minimum spanning tree, which will favor connecting nodes that are close together. The second tree was the shortest path tree (using the squared distance), which will favor short paths to the base station. The last tree we tried was the balanced tree from [5], which is a parameterized combination of the minimum spanning tree and shortest path tree. The MST is constructed and then any paths to the base station longer than $\alpha * d_{SPT}$ get shortcut to the base using the path from the SPT. We used $\alpha = 2$ for our experiments. We expected this tree to perform well because it should provide a combination of the two desirable properties: connecting nearby nodes and keeping short paths to the base station.

In our experiments we have seen that the shortest path tree tends to outperform the other two trees (see Figure 11) for small values of information loss. The squared distance measure helps the SPT to avoid long edges while finding short paths to the base. However, there are cases, such as the one shown in Figure 12, where the MST outperforms the SPT if we are willing to tolerate a significant loss in information. The MST prioritizes connecting with nearby nodes over finding short paths to the base, so it provides more opportunities for useful aggregation. The balanced tree is an attempt to incorporate the best properties of each tree, and indeed its performance is close to that of the best tree in both cases. This experiment highlights the need to carefully choose the communication tree to get the best information

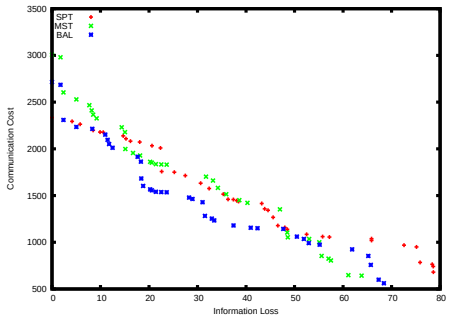


Figure 12. A different experiment on 80 nodes. This time the performance of MST is closer to that of the SPT, and outperforms MST for high information loss tolerance.

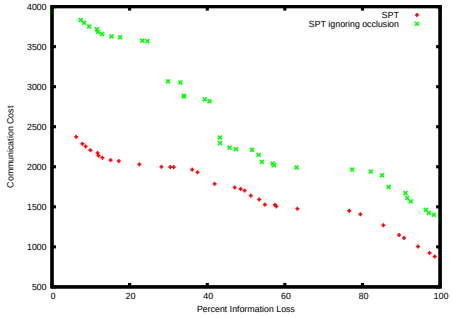


Figure 13. Comparison of communication costs between an SPT that counts occlusion in distances and one that does not.

loss/communication cost tradeoff. In fact the best tree may depend on the information loss tolerance; if the tolerance is high, clustering the nodes for better averaging should be prioritized over short paths to the base, and if the tolerance is very low, the tree should be close to the SPT. We plan to address this question in more depth in future work.

4.4. Occlusion

There are situations in sensor networks (especially in environmental modeling) where the communication cost between two nodes may not be a function of their distance, due to some obstructions such as leaves or deer, for instance. We will call such nodes "occluded". A model of where and how occlusion effects the network is a vital ingredient for finding a low cost, low information loss strategy, as we will show in the following experiment.

For this experiment we have random selected 10% of the nodes in the network to be occluded. If an edge is occluded it is assigned a random weight from $[1, M]$, which the cost of communication on any adjacent edge will be multiplied by. If both endpoints of an edge (u, v) are occluded, the cost of communication on that edge will be multiplied by $w(u) * w(v)$. Figure 13 compares the costs of an SPT built using the occlusion weights and one that ignores the weights. Even with only 10% of the network occluded, there is a dramatic difference between accounting for the occlusion and ignoring it.

The complication introduced by occlusion is that correlation (and therefore compression and aggregation performance) is tied to distance, but communication cost has been decoupled from distance in some places. This means that the best tree when only forwarding is allowed may no longer be the best tree when the abilities to compress and aggregate are introduced. We plan to explore new tree building algorithms to address this issue in future work.

5. Conclusion

We have presented a framework for optimally combining loss-less compression and lossy compression, given a communication tree and communication cost budget. We have shown experimentally that the combination of compression and aggregation is able to save more energy than either of these techniques alone, if some information loss can be tolerated.

Our algorithm and experiments have made several assumptions that we hope to address in future work.

- We have assumed a model of the joint distribution of the sources that allows us to calculate compression size and information loss using simple formulas. Could these parameters be estimated efficiently in a distributed manner?

- Our dynamic programming algorithm requires the assumption that nodes can't compress if they receive any aggregate information. Removing this assumption may require a more heuristic approach, but could potentially save more energy.

- In this paper we have addressed the problem of finding the best locations for aggregation and compression in a given network. We also compared the performance of a few different networks. However, the question of what network our algorithm would give the best performance on has been left unanswered, especially in the case of occlusion, where the best network when only forwarding is allowed may not be the best network when compression and aggregation are introduced.

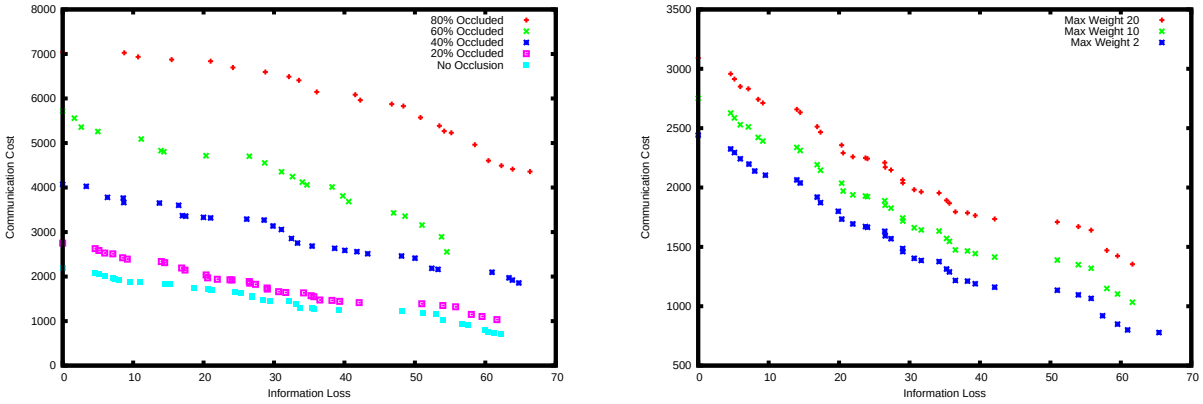


Figure 14. (a) The effects of varying the percentage of occluded nodes in an 80 node network. (b) The effects of varying the maximum weight of occlusion in an 80 node network.

References

[1] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. *ICDE*, 2006.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[3] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. *VLDB*, 2004.

[4] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. *ICML*, 2005.

[5] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 1995.

[6] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. *ISPN*, 2006.

[7] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. *ICDCS Workshop on Distributed Event-based Systems*, 2002.

[8] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *OSDI*, 2002.

[9] S. Nath, B. Gibbons, S. Srinivasan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *SenSys*, 2004.

[10] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ISPN*, 2004.

[11] S. S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, 2002.

[12] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. *SenSys*, 2006.

[13] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. *MobiCom*, 2002.

[14] A. Silberstein, R. Braynard, and J. Yang. Constraint-chaining: On energy-efficient continuous monitoring in sensor networks. *SIGMOD*, 2006.