Aggregation Methods for Large Scale Sensor Networks

LAUKIK CHITNIS
University of Florida
ALIN DOBRA
University of Florida
and
SANJAY RANKA
University of Florida

The ability to efficiently aggregate information – for example compute the average temperature – in large networks is crucial for the successful deployment of sensor networks. This paper addresses the problem of designing *truly scalable* protocols for computing aggregates in the presence of faults, protocols that can enable million node sensor networks to work efficiently. More precisely, we make four distinct contributions. First, we introduce a simple fault model and analyze the behavior of two existing protocols under the fault model: *tree aggregation* and *gossip aggregation*. Second, since the behavior of the two protocols depends on the size of the network and probability of failure, we introduce a hybrid approach that can leverage the strengths of the two protocols and minimize the weaknesses; the new protocol is analyzed under the same fault model. Third, we propose methodology for determining the *optimal* mix between the two basic protocols; the methodology consists in formulating an optimization problem, using models of the protocol behavior, and solving it. Fourth, perform extensive experiments to evaluate the performance of the hybrid protocol and show that it usually performs better, sometimes orders of magnitude better, than both the tree and gossip aggregation.

Categories and Subject Descriptors: C.2.4 [**COMPUTER-COMMUNICATION NETWORKS**]: Distributed Systems—*Distributed applications*; G.1.6 [**Numerical Analysis**]: Optimization—*Constrained optimization*; G.3 [**Probability and Statistics**]: —*probabilistic algorithms (including Monte Carlo)*; H.2.4 [**DATABASE MANAGEMENT**]: Systems—*Query processing*

General Terms: Algorithms, Design, Performance, Reliability

Additional Key Words and Phrases: In-network processing and aggregation, Fault tolerance, Sensor fusion and distributed inference, Modeling faults

Author's address: Laukik Chitnis,
CSE Building Room E301, University of Florida, Gainesville, FL 32611-6120 USA
lchitnis@cise.ufl.edu

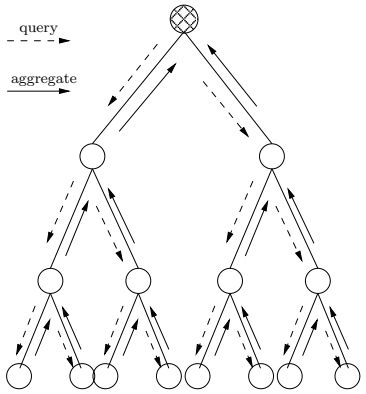


Fig. 1. Tree-based aggregation

1. INTRODUCTION

The modern sensors can sense and measure odors, vibration, acceleration, pressure, temperature and other physical quantities; all this information is useful in gathering data about our surrounding world. Improving processor performance has made it possible to manufacture and embed small but sophisticated chips at considerably low costs. Silicon scaling has also drastically reduced the cost of communication for both wired and wireless systems. The small size and low cost of the components allow deployment of network of very large number of such devices in environments that are otherwise out of reach of human beings. Technologies like smart buildings, ingestible health monitors, smart dust, etc., which are now available in experimental settings will become commonplace [Werner-Allen et al. 2005] [Mainwaring et al. 2002]. We envision that sensors networks consisting of hundred thousand to a million sensors will not be uncommon in these settings and will be indispensable, for example, for large scale scientific and civilian applications. Given the current state of the research on aggregate computation in sensor networks, it is not apparent how the existing protocols/approaches can be adapted for such large networks since no scientific study up to date considered networks of such large size.

The promise of sensors networks is to enable spatial and temporal data collection at a much higher rate than currently prevalent. An important set of data collection activities involves aggregation of information collected from each of the sensors. Simple examples of aggregation include calculating the maximum pressure in an experiment chamber, or calculating the average temperature of a surface. However, these simple aggregations can be easily generalized to complex operations that have underlying associative and commutative properties.

The main challenge in designing algorithms/protocols for distributed computation of aggregation in sensor networks is to keep the resource utilization to the minimum by reducing the communication among nodes and computation performed. The only hope to significantly reduce the resource utilization for aggregate computation is *in-network aggregation*, i.e. perform (partial) aggregation as the information is transmitted. Depending on how/if the in-network aggregation is performed, there are three main techniques published in the literature:

Algorithm: Protocol Push-Sum

1: Let $\{(\hat{s}_r, \hat{w}_r)\}$ be all pairs sent to i in round $t-1$
2: Let $s_{t,i} := \sum_r \hat{s}_r, w_{t,i} := \sum_r \hat{w}_r$
3: Choose a target $f_t(i)$ uniformly at random
4: Send the pair $(\frac{1}{2}s_{t,i}, \frac{1}{2}w_{t,i})$ to $f_t(i)$ and i (yourself)
5: $\frac{s_{t,i}}{w_{t,i}}$ is the estimate of the average in step t

Fig. 2. Gossip aggregation

Centralized processing:. The most straight-forward solution involves transmitting the values collected at each sensor node directly to a centralized processing unit. Aggregation operation can then be easily performed using for example, current database technology. The main drawback of this approach is the sheer amount of communication required since each piece of information has to make its way to the central location. This translates immediately into large processing times – which rules out time critical applications – and large power consumption since each node has to either use a strong signal for the wireless transmission or the message has to be routed through a large number of nodes.

Tree aggregation:. The main idea here, depicted in Figure 1, is to organize all the sensor nodes into an aggregation tree[Madden et al. 2002]. The root of the tree is the node where the query is injected and where the aggregation result is retrieved. The query/request is propagated from parent to children. The leaf nodes send the value of the measurement to the parent. Intermediate nodes wait for values from the children, do a local aggregation of these values and its own measurement and send the aggregate to the parent. The root node computes in the same manner as any intermediate node and presents the value of the overall aggregate to the user. The algorithm requires a spanning-tree among the sensor nodes [Madden et al. 2002] [Madden et al. 2003] [Bawa et al. 2003] [Gupta et al. 2001]. The efficiency of the algorithm depends on the longest route from root to the leaf, i.e how balanced this tree is.

Gossip-based aggregation:. These techniques are based on randomized algorithms that proceed in rounds until convergence. In each round, each node contacts some of its neighbors (either physical neighbors or, in the case routing messages through the network is possible, any other nodes) and exchanges information with these nodes. We discuss two variations of this protocol proposed in the literature. In the *Push-Sum protocol* [Kempe et al. 2003], depicted in Figure 2, each node maintains two quantities: a weight and the aggregate. In each round, each node chooses randomly a node in the network and sends this node half the weight and half the value of the aggregate (i.e. the sending node halves its own weight and aggregate and the receiving node adds these quantities to its own). If every node can uniformly randomly contact any other node the protocol will compute the aggregate in time proportional to the log of the number of nodes. The *Distributed Random Grouping protocol* [Chen et al. 2005] proceeds by randomly selecting a small set of broadcasting nodes. Each such node will send a broadcasting message to all its immediate neighbors and get from them their current value of the aggregate. Then,

the new local aggregate value is computed and disseminated to all the nodes that responded to the broadcast. The convergence of this protocol depends on the *algebraic complexity* [Fiedler 1973] of the graph and it is usually slightly better than the convergence of push-sum protocol.

Clearly, any method using in-network aggregation is superior in terms of the number of messages sent/received to the centralized algorithm if the base station cannot directly contact all the sensors. Even if this direct contact with base station is possible, the total transmission power required would be very high and the messages to/from the sensors have to be sequenced (i.e. sent sequentially through the communication medium). A more interesting comparison is between tree and gossip aggregation. Tree aggregation, if there are no faults in the system and the spanning-tree is relatively balanced, is preferable to the gossip aggregation since not all nodes participate at all times. While gossip can achieve the same asymptotic efficiency, the tree aggregation is more efficient by at least a significant constant factor in these scenarios, as seen in Figure 4. The place where gossip aggregation excels, and indeed the motivation for such protocols [Kempe et al. 2003], is in networks where nodes can fail with significant probability. The tree aggregation suffers because if one intermediate node in the tree is faulty, none of the descendants are reachable; a reasonable alternative then, as we will see later, is to rebuild the tree. On the other hand, gossip aggregation is mostly immune to faults but it is slower. Since no large scale practical application of sensor networks is likely to be in one of the two extreme scenarios, it is worth investigating *hybrid* aggregation schemes that can be customized for a particular application. Such schemes could combine a tree and gossip architecture and, hopefully, perform significantly better than any single method. The goal of this paper is to develop such a hybrid protocol and to analyze its properties for large or highly faulty sensor networks. More precisely, our contributions are:

(1) We propose a simple but realistic fault model which we use to analyze the behavior, in the presence of faults, of both tree and gossip aggregation.
(2) We propose a realistic notion of correctness for aggregation protocols in sensor networks, that we call *robust correctness*. We explain why this notion of correctness is the best that can be hoped for when failures are possible.
(3) We propose a hybrid protocol that partitions the sensors into groups and uses gossip for aggregation within the groups and a tree for aggregation between groups. We analyze the fault behavior of the hybrid protocol under the fault model. While the simplicity of the fault model might be of concern, as experiments we carry show, the hybrid protocol designed using the assumed fault model behaves reasonably when correlated faults are actually present.
(4) We introduce methodology and algorithms to determine the *right* combination between tree and gossip aggregation in the hybrid model. This problem results in the formulation of optimization problems; we explain how the optimal solution can be determined.
(5) We study empirically the efficiency of the hybrid protocol and show that it can be significantly better, sometimes by as much as a factor of 1000, than the best of the tree and gossip aggregation. The results also show that the hybrid

protocol has the potential to scale to hundreds of thousands of sensor nodes and that the protocol is robust with respect to variations to the fault model.

In the rest of the paper, we first develop a fault model, in Section 2, and apply it to existing techniques. Then we describe and analyze the hybrid protocol in Section 3. We empirically evaluate the performance of the hybrid protocol in Section 4 and address various issues in Section 5. We then discuss related work in Section 6 and conclude in Section 7.

2. SENSOR NETWORK AGGREGATION UNDER FAULTS

In the absence of any faults, the tree aggregation is clearly the best choice for two main reasons: (a) Even if a significant effort is required to build a good tree, the absence of faults would make the maintenance of the tree trivial, and (b) Aggregation using a (well balanced) tree is the most efficient method we can hope for both in terms of the number of messages exchanged and the time to completion.

If the nodes are highly prone to faults, the gossip-based aggregation methods are favored since, by design, they do not depend on any stable distributed aggregation infrastructure. These methods would be merely slowed down by faults; the ability to accommodate a lot of faults is their strongest point. In practical situations, however, the fault behavior is unlikely to be in any of these two extreme situations. For this reason, in this section, we introduce a simple parameterized fault model – a parameter will control the *number* of faults in the system – and use it to predict the behavior of tree and gossip aggregation in the presence of faults.

2.1 A Simple Fault Model

For a fault model to be useful for analyzing the behavior of aggregation techniques for sensor networks in faulty environments, the model must have two crucial properties: (a) It must be *realistic* in order for the facts derived using the model to be reasonably extensible to real scenarios, and (b) It must be *simple* enough to allow theoretical analysis of the protocols – due to the scale of the problem, only theoretical analysis is feasible.

One important point to make is the fact that the fault model is exclusively used for the design of the hybrid protocol for aggregation. Even if the actual fault behavior is different, this assumption can be made as long as it leads to good performance of the resulting protocol. The main concerns with choosing a *sophisticated* fault model are:

—Sophisticated models depend on large numbers of parameters. Accurately estimating these parameters for a given practical scenario can be challenging. Inaccurate values can lead to models that are so far from the actual fault behavior to the point that bad decisions are made.

—Sophisticated models might not make a significant difference in the performance of the resulting protocol since they are capturing secondary effects that may have little influence.

—In order for a sophisticated fault model to be useful, all nodes have to be aware of the model. Especially when such models capture correlations between failures, large amounts of information have to be communicated to all sensor nodes. This can be problematic from both, local storage and communication, point of views.

2.1.1 *Fault model.* The fault model we consider here makes the following assumptions:

—Each of the nodes can fail independently with probability p in the time it takes to make one transmission (this includes the computation time required). Probability of failure in unit time can be selected but, we believe, this particular definition leads to simpler and more intuitive formulas. If the probability to fail in unit time is known, the probability to fail in the time it takes to make a transmission can be readily obtained.

—The failures are not correlated, i.e. they are independent from each other. This is a realistic assumption in most situations. We discuss the impact of correlations after we give the details of our solution.

—Link failures are absorbed in the probability p. This is not a limiting assumption since it is impossible for a node to make the difference between a node failure and link failure if only direct communication is used, as is the case in most aggregation protocols. If communication is achieved using routing, link failures are masked (dealt with at the communication level).

—Transient failures, which might trigger message retransmissions which do succeed later, are ignored. Short term transitory failures would simply increase the overall time required for the aggregate computation and would affect all the methods we consider in the same way (i.e. by increasing the time and the number of messages by a constant factor). For this reason, the design of the hybrid protocol is not influenced by the transient failures, hence we ignore them.

As it is apparent from the above description, apart from the size of the sensor network, the only parameter is the probability of failure per time to make a transmission. Since the failures are independent, the theoretical analysis is greatly simplified and the design of the *best* hybrid protocol is reasonably simple.

We believe that this fault model captures the dominant behavior in a large scale sensor network and it is simple enough to allow analysis of the various aggregation techniques. This is confirmed by the experiments we report in Section 4.3 where we considered correlated fault models but ran the protocol that assumes the simple fault model described here with the result that the performance is similar.

2.2 Correctness under Faults

If faults are possible in the system, the usual notion of correctness (i.e. the value of the computed aggregate being exactly equal to the *true* value of the aggregate) has to be revised. Since the nodes do fail, defining the true aggregate is problematic: should it be the aggregate over the sensors *alive at the start* or *alive at the end* of the computation? There are a number of problems with such notions of correctness which are insurmountable or impractical in the presence of faults and asynchronous communication: (a) Correctness with respect to nodes alive at the start of the computation is not achievable since nodes can fail even before they communicate with any other node so the information is lost, and (b) Correctness with respect to the nodes alive at the end of computation implies complete knowledge about what nodes are alive which is not possible in faulty environments[Fischer et al. 1985].

A less restrictive correctness criterion is to provide a result for the aggregate that

is *between* (or approximates well) the smallest and largest value of the aggregate computed over the values generated by the sensors that were alive at some point during the computation. For example, if only one of the nodes become faulty and the aggregate is the sum of all the values (one per sensor) then, under this notion of correctness, we allow the result to be between the value of the sum with and without the faulty node. Formally, we require the result to be a linear combination[1] of the values of non-faulty sensors at the beginning of the computation. Furthermore, the weight of all non-faulty nodes at the end of the computation have to be approximatively equal. In the rest of the paper we will refer to this notion of correctness as *robust correctness*.

While it might seem that robust correctness is overly complicated, we believe that this is the strongest notion of correctness that can be ensured by aggregation protocols for the reasons stated earlier. All the intuitive notions of correctness we considered depend on common knowledge that some event happened (nodes were alive at some stage); since common knowledge cannot be achieved in distributed systems with failures, no protocol can be correct which renders the other intuitive notions of correctness not very useful. We believe that the guarantee of our robust correctness is strong enough for the end user to feel confident about the result obtained by the aggregation protocol.

A very important point about the correctness in sensor networks with node failures, is the fact that correctness cannot be checked in the protocol that computes the aggregate since there is no way to determine what nodes are/were alive at any given moment[Fischer et al. 1985]. The goal of the correctness protocol we proposed is not to provide a mechanism to check the correctness but to characterize a reasonable behavior of the sensor network. The manner we will use the robust correctness is to ensure that the protocols we consider acceptable are implicitly correct (i.e. satisfy this notion of correctness) since no explicit check can be performed.

2.3 Tree Aggregation under Faults

The most important question we have to ask, for each aggregation method, is how it can deal with faults and still achieve robust correctness. In the case of aggregation trees, there are three basic approaches to deal with faults:

Ignore fault. When a fault occurs, simply ignore it. However, the information in the subtree rooted at the faulty node is not aggregated, i.e. not only the information of the faulty node is lost, but the information in the entire subtree is lost. This solution does not ensure robust correctness, unless the faulty node is a leaf, and is not usually acceptable in practice since nothing can be guaranteed about the answer.

Local fix. When a fault occurs for an intermediate node – faults in leaves can just be ignored – attempts can be made to fix the tree locally. This possibility, which has not been really explored in depth in the research literature to the best of our knowledge, has the following disadvantages:

—A crucial requirement for the tree aggregation is the fact that the graph formed by

[1] A linear combination between values $x_1, \ldots, x_n$ has the form $\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$ for positive weights w_i.

the parent-child connections is actually a tree. It is not clear how the requirement can be ensured only using local knowledge; network wide communication might be required to ensure this happens, or conventions on what nodes can be connected (to ensure loops cannot be formed).

—Reconnection of the children/descendants of a faulty node is complicated by the fact that the aggregation procedure is in progress. If the computation has to be restarted, the local fix would most likely cost almost the same amount of time as the complete tree reconstruction[Jia et al. 2006].

—Implementing complicated protocols on sensor networks can be problematic since the computation and communication capabilities of the sensors are low and likely to remain low since the primary goal is prolonging battery life.

—When the rate of failure is high, the local fixes may not be able to keep up with the rate at which the faults occur and severe interference problems between parallel fixes are likely to occur.

For the reasons listed above, we believe that local fixes are impractical and alternatives have to be considered. It is possible to combine tree aggregation with other techniques such as caching old values and using them when communication with a child cannot be established [Madden et al. 2002], and using multiple trees [Madden et al. 2002]. A careful look at these methods reveals that they do not satisfy our notion of robust correctness – we extensively comment on this issue in Section 6. Hence, we seek an approach that avoids altogether the need for local fixes but retains good performance. Since no strict notion of correctness is ensured by the local fix protocols proposed in the existing literature[2] we do not explore this option any further.

Global fix. Each parent keeps track of the state of their children. When the fault of one of the children is detected – or link failure is detected, which would have the same consequence – a message is pushed up the tree to inform the root to rebuild the tree. A simple flooding algorithm [Ding et al. 2003] [Madden et al. 2002] can be used to build/rebuild the tree and the aggregation query dissemination can be performed simultaneously. In order for the aggregation to be successful, the tree has to *survive* the length of the computation – this means that both the construction/query dissemination and the aggregation from leaves to the root must be fault free. An interesting observation is the fact that failure of nodes after the aggregate value has been sent to the parent do not invalidate the correctness. Modeling this detail does not add significant precision, as we discuss below.

From the discussion above, the approach we chose for tree aggregation is the global fix, i.e. rebuild the tree when a fault occurs. In what follows, we estimate, under the simple fault model in Section 2.1, the amount of time and number of messages required on average to compute an aggregate in a sensor network of size N with maximum depth $d(N)$. The maximum depth $d(N)$ depends both on the actual topology, manner of establishing connections between sensors and the tree construction algorithm. In the best case $d(N) = \log N$ but it can be as bad as N when the tree is a line.

[2]For the reasons we listed in the text, we believe that no notion of correctness can actually be ensured for local fixes unless global knowledge can be achieved.

Rather than complicating the fault model we observe that the time for the bottom-up aggregation is equal to the time for aggregation query dissemination/tree construction.

PROPOSITION 1. *Suppose we have a sensor network of size N for which aggregation trees of maximum depth $d(N)$ can be constructed. Assume the* simple fault model *and let p be the independent probability of failure for any of the nodes in the network. Then, the average number of restarts (tree reconstructions) before success is $\frac{1}{(1-p)^{d(N)N}}$ and the average time to successfully finish the aggregation, as a multiplier of transmission duration, is $d(N)\frac{1}{(1-p)^{d(N)N}}$*

PROOF. First, we have the following result: if X is a random variable that encodes the number of coin flips it takes to see a head (as opposed to tail) when the probability of each independent trial is q, then X has a Geometric distribution with parameter q and $E[X] = \frac{1}{q}$ (see for example[Shao 2003]). The variance of such a random variable is $\frac{1-q}{q^2} \leq \frac{1}{q^2}$ thus the standard deviation is upper-bounded by $\frac{1}{q} = E[X]$. This means that the behavior of X is really dictated by $\frac{1}{q}$; the values observed are within a small constant multiplier of this quantity.

To estimate the probability of failure in one round, we make the following observations: for the computation to fail, it is enough to have any of the N nodes fail in any of the $d(N)$ time slots. Since all failures are independent, the probability that this does not happen is $(1-p)^{d(N)N}$ (this is equivalent to having $d(N)N$ independent coin flips all turning tails with p the probability for a head). Linking the two results together, we have the required results, i.e. the number of tree reconstructions is $\frac{1}{(1-p)^{d(N)N}}$ and the time is $d(N)\frac{1}{(1-p)^{d(N)N}}$. □

To get an intuitive idea of what the behavior of the tree aggregation as a function of the probability of failure is, we provide the following result:

PROPOSITION 2. *Using the same setup as in Proposition 1 and defining $q = pd(N)N$, i.e. the ratio of p and $\frac{1}{d(N)N}$, for large $d(N)N$ the average number of times the tree has to be reconstructed until aggregation succeeds is approximately e^q (e is the base of the natural logarithm).*

PROOF. Using the result in Proposition 1, the number of tree reconstructions is $R = \frac{1}{(1-\frac{q}{n})^n}$ where $n = d(N)N$. When n is large we have:

$$R \approx \lim_{n\to\infty} \left(1 - \frac{q}{n}\right)^{-n} = e^q$$

Thus, the number of reconstructions until success is e^q. □

This theoretical result suggests that, when $q \leq 1$, the average number of tree reconstructions is $\leq e \approx 2.7$, thus the impact of the tree reconstruction for tree aggregation is minimal – this is the best scenario for tree aggregation. When $q > 1$ even by a small margin, the number of reconstructions required might be prohibitively large, thus rendering tree aggregation unacceptably slow. For example, when $q = 10$, about 22000 tree reconstructions are required on an average for the tree aggregation to succeed.

To check the correctness, we observe that the result returned is the aggregate over the nodes that were alive in the last, successful round, thus the result of the aggregation has *robust correctness* (it is one of the possible values of the aggregate obtained with the nodes that were alive at the start and the end of the computation).

This fault model for tree aggregation assumes the fact that the failures are uncorrelated. When this is not the case, a new fault model specific to the particular type of correlation can be constructed. As we will see later, the design of the hybrid protocol depends mainly on the behavior of the tree aggregation in the presence of faults (i.e. the probability that at least one node fails during the tree aggregation), and is not very sensitive to common cases of correlated fault behavior.

2.4 Gossip Aggregation under Faults

Following the description in the introduction, the gossip algorithms do not depend on any distributed data-structure to work. If a node fails, the other nodes will carry out the distributed computation without the failed node and simply not communicate with this node. The only concern is whether the result obtained has *robust correctness*.

2.4.1 *Push-sum Protocol.* When using the Push-sum protocol [Kempe et al. 2003], sensors maintain two quantities: a weighted sum of the values to be aggregated and the sum of weights. The ratio of these two quantities is the local estimate of the aggregate. If none of the nodes fail, Kempe et al.[Kempe et al. 2003] proved that the protocol converges for fully connected networks when each sensor picks a random sensor to communicate with (the sending sensor will send the receiver half its sum and half its weight).

If a sensor's connection is faulty, the communication is never initiated thus nothing is lost – this assumes some form of handshaking protocol but this is the standard for communication protocols and it is usually implemented in hardware. A complete node failure can be modeled by simply not allowing the sum and the weight to change. To determine if the Push-sum protocol has *robust correctness*, we make the following observations. First, the sum of sums and the sum of weights (including the faulty nodes) in the sensor network is conserved (this is the main observation in [Kempe et al. 2003]). Second, for an active node, the ratio of the sum and the weight converges to a constant value throughout the system. Third, due to the nature of the protocol, the ratio of the sum and the weight for any node is a linear combination of the sensor values including the faulty sensors. At convergence, the weight, as seen by any live node, of all the nodes alive is approximately the same. This immediately implies that Push-sum satisfies *robust correctness*.

2.4.2 *The Distributed Random Grouping (DRG) algorithm.* The *DRG protocol*[Chen et al. 2005] consists in having each node deciding, with small probability, whether it wants to be a broadcasting node. Then, the broadcasting node sends a broadcast message to all neighbors, gathers their current value, averages all the values received and distributes the information to all nodes that replied. Since averaging transforms linear combinations into new linear combinations, the partial aggregate value at any node at any time is a linear combination of the values in

the sensor network. The protocol is guaranteed to converge[3] for all the nodes alive. This immediately implies that the protocol satisfies *robust correctness*.

Note an important difference between the gossip-based protocols and tree aggregation: the correctness is achieved without significantly slowing the computation down – the slowdown is due only to the multiple retries to communicate with a node and not due to the need to reconstruct large distributed data-structures. This robustness to faults is the primary motivation for these types of protocols.

3. A HYBRID PROTOCOL

As we have shown in the previous section, when the probability of faults is small (smaller than $\frac{1}{d(N)N}$), the behavior of the tree aggregation is excellent. In this case, the gossip type protocols have reasonable performance as well but, as is apparent from the experimental results in Section 4, they are inferior to tree aggregation by a significant constant factor.

When the probability of failure is large ($\gg \frac{10}{d(N)N}$) tree aggregation is poor but gossip protocols' performances are mostly unaffected; in these circumstances the gossip protocols are clearly preferable. Since we already have a reasonable fault model and the characterization of the two types of protocols, a simple approach for deciding what protocol to use in a particular practical situation is to estimate the failure probability and to compare it with $\frac{1}{d(N)N}$. If it is small, tree aggregation should be preferred, if it is larger, gossip protocols are a better choice.

A third alternative is possible and potentially interesting: a combination between tree and gossip aggregation. The reason such a hybrid protocol might work better than either tree and gossip aggregation is based on the observation that the performance of the tree aggregation is dramatically influenced by the size of the tree. For example, if $q = pd(N)N$ is large, say 16, the number of restarts needed for the tree aggregation, according to Proposition 2, is $e^{16} \approx 8.8 \times 10^6$. If we reduce the number of nodes that participate in tree aggregation by a factor of 12, then even at $d(N) = 13$, $d(N)N$ is reduced by at least a factor of $\frac{12*\log N}{\log N/12} \approx 14.84$; thus q becomes ≈ 1.07. So now, the number of restarts is ≈ 2.94, a 6 order of magnitude reduction from the previous case. The reduction in the number of nodes in the tree can be achieved by partitioning sensors into groups, computing the aggregate within each group using a *more robust* approach and using a tree to compute the aggregate over the groups. In the above example we only need groups of size 12 to make the protocol practical.

Based on the above observation, the particular hybrid protocol we propose is depicted in Figure 3. First, the sensors are organized, based on local proximity, into m groups, each of average size $\frac{N}{m}$. Within each group, gossip type protocols are used to compute the value of the aggregate within the group. At the same time of running the aggregation, a leader is selected for each group (any of the gossip based leader selection algorithms can be used [Gupta et al. 2000]). The group leaders then organize into an aggregation tree and aggregate the group values using the tree aggregation protocol. If the group leader fails, any other node in the group

[3]The convergence speed depends on the *algebraic complexity* of the graph and is guaranteed for any connected graph [Chen et al. 2005].

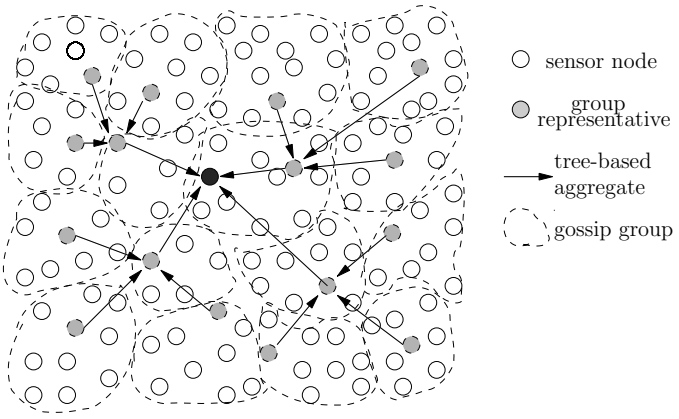


Fig. 3. Hybrid protocol.

can take over the role and participate in tree aggregation since, by the nature of gossip protocols, all nodes converge to the same value. The groups can be reused between queries; there is no need to repartition the network for each query. Thus, any algorithm for partitioning the sensor network can be used since efficiency is not crucial.

The hybrid protocol has *robust correctness* since it is a simple composition of tree and gossip aggregation, both of which have this property (assuming that the tree is reconstructed when a node fails).

3.1 Choosing the Group Size for Hybrid Protocol

Given the qualitative observations about the efficiency of the tree aggregation as a function of the number of nodes, a straightforward approach to determining a reasonable value for the size of the groups is to choose $\frac{N}{m}$ so that $pd(\frac{N}{m})\frac{N}{m} \approx 1$ (m is determined by solving the equation). This would ensure that the tree aggregation part of the protocol needs few restarts; the rest will be taken care of by the gossip within the groups. This approach, even though reasonable, is based on qualitative observations and ignores the fine details of the actual difference in efficiency between the tree and gossip aggregation, which could be substantial (see Figure 4. The important observation here is the fact that no protocol dominates the other for the whole spectrum of probabilities of failure.

A more principled approach is to formulate and solve an optimization problem that gives the *optimal* value for the number of groups given the characteristics of the sensor network. This approach can be used for determining the optimal group size in order to minimize the time to completion or the total number of messages (the energy used in the system depends mostly on the total number of messages). We explain here in detail how to optimize for the time to completion; we then briefly mention how to change the optimization problem to optimize for the number of messages.

As before, $d(N)$ is the amount of aggregation time in a fault-free tree of size N that can be constructed, using the preferred tree construction method, on sensor networks with similar architecture. $d(N)$ is simply the product of the maximum depth of the tree and the number of actual messages required for the parent and

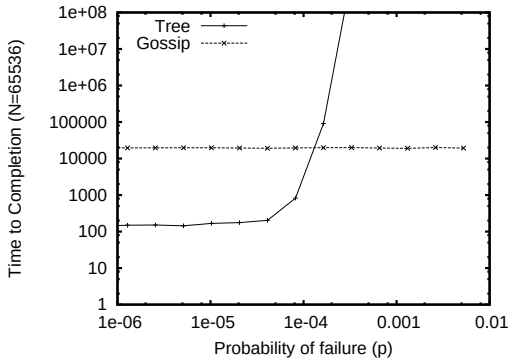


Fig. 4. Comparison of performance of tree and gossip aggregation

child to communicate, i.e. the length of the path between groups. Let $g(N)$ be the convergence time for the particular gossip protocol and communication graph as a function of the size of the network. This function would take into consideration the fact that the communication is local or long distance by measuring the time in local transmissions. With this, the time to completion is the sum of the completion times for running gossip within each group (maximum time across the groups) and tree aggregation between groups, that is:

$$\begin{aligned} T &= T_{\text{gossip}} + T_{\text{tree}} \\ &= c_1 g\left(c_2 \frac{N}{m}\right) + d(m) \frac{1}{(1-p)^{md(m)}} \end{aligned}$$

In the above equations, factor $c_1 \geq 1$ is introduced to account for the fact that $g(N)$ is the expected time, and not the actual running time of the gossip protocol; an actual run of the algorithm might require more time. The factor $c_2 \geq 1$ is introduced to account for the fact that the size of individual groups will be N/m only on average; individual groups could be much larger. The product $c_1 g(c_2 \frac{N}{m})$ models the pessimistic behavior of the gossip aggregation: the *unlucky* speed of the slowest gossip group. We discuss how c_1 and c_2 are determined later in the section.

With this, the optimization problem is:

$$m_{\text{opt}} = \text{argmin}_{m \in [1 \ldots N]} \left[c_1 g\left(c_2 \frac{N}{m}\right) + d(m) \frac{1}{(1-p)^{md(m)}} \right] \tag{1}$$

When the number of messages has to be optimized, the optimization criterion is:

$$N g\left(\frac{N}{m}\right) + m \frac{1}{(1-p)^{md(m)}}$$

The factors c_1 and c_2 are not necessary here since we want the average, not extreme behavior of the groups.

About the gossip speed factor c_1. Determining the variability due to the running of the gossip protocol is very involved theoretically. For this reason, the method we

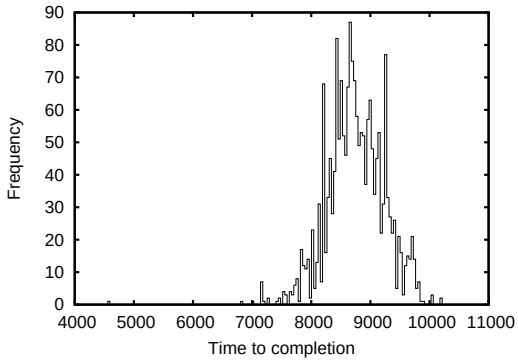


Fig. 5. Frequency distribution of the Time to Completion of gossip protocol for a group of size n=4096 and p=0.00065536

propose here is to determine an appropriate value for the factor c_1 using an empirical approximation of the distribution obtained by simulating the behavior of gossip on the given network. Figure 5 depicts the empirical approximation of the distribution of the convergence speed for a network of size n=4096 and p=0.00065536. As is apparent from the figure, the distribution is concentrated around the expected value and the probability, that the speed is twice as slow as the average, is very small. An immediate consequence is the fact that the maximum, even over a large number of such distributions, will be within a factor of 2 of the average thus 2 is an upper bound of the contribution to the constant c_1.

About the maximum group size factor c_2. The variability due to uneven group size has to be determined based on the particular grouping algorithm used. Unless the grouping algorithm strives to achieve uniform groups, the grouping can be considered random and each group size modeled by a binomial variable – the group of all sizes forms a multinomial distribution. The average of each such variable is $\frac{N}{m}$, but here we need an estimate for the largest group – this group will likely have the slowest gossip. Such an estimate is given by a classic result by observing that this is a *balls into bins* problem [Rao 2004][Raab and Steger 1998] . A good estimate of the size of the largest group is $\frac{N}{m} + \log m$. When the number of groups is small, the logarithmic factor has no influence but becomes dominant when m approaches N. The contribution to the constant c_2 is determined by dividing this quantity by the expected size of the group.

Putting these two contributions together, the value of T_{gossip} we used in equation 1 and recommend in such scenarios is:

$$T_{\text{gossip}} = 2g((1 + \frac{m \log m}{N})\frac{N}{m})$$

4. EMPIRICAL EVALUATION OF THE HYBRID PROTOCOL

In the previous section, we introduced a hybrid tree/gossip protocol. The optimal combination between the two protocols is determined by formulating, based on the

characteristics of the application, and then solving an optimization problem. The main goal of this section is to present empirical evidence that the hybrid protocol is always at least as good as the best of the tree and gossip protocols and sometimes significantly better, by as much as two orders of magnitude.

4.1 Experimental Setup

In this section we give a detailed description of the experimental setup used to produce the results reported later in the section. Due to lack of space, we restrict the type and parameter values for the synthetic sensor networks for which we report results; we observed qualitatively similar results for different types of networks and different parameter values.

The experiments we present here were performed on simulated sensor networks constructed by randomly placing the nodes in a unit square field. For each node, the transmission radius is selected such that the number of direct neighbors is 32. We present results for the number of nodes N varying between 16384 and 262144.

The hybrid protocol that we introduced can use any gossip protocol for aggregation within a group. In all the experiments reported here, we used the push-sum protocol described in [Kempe et al. 2003] for performing aggregation using gossip; we use three variations of this protocol: (a) nodes are allowed to talk only with the immediate neighbors, (b) nodes are allowed to talk, using routing, with any node – this is the model assumed in [Kempe et al. 2003], and (c) nodes are allowed to talk to a logarithmic number of nodes that are distributed randomly throughout the network – this is the model used in peer-to-peer networks. For experiments in sections 4.2 and 4.3, the partitioning of nodes into groups is achieved by splitting the space into a grid with as many cells as groups (the possible number of groups has been restricted so that this is possible). By assuming random placement of the sensors, we can indeed use the formula for the factor c_2 derived in Section 3. We provide some experiments based on a different grouping technique in Section 4.4. A point to be noted here is that the groups, once formed, can be used for multiple instances of aggregation. Thus, the grouping cost when amortized for one instance of aggregation is not significant. In fact, when the gain of using the hybrid protocol, as estimated theoretically by our methodology, is very small (say, < 1.5), we default to the pure protocols (just tree or gossip), instead of incurring the overhead costs of grouping which may not be justified in such cases.

In the tree-based approach, aggregation occurs by forming a spanning tree among the participating nodes by starting with the root node, splitting the space into 4 regions, selecting a random node in each part to be the child of this node and recursing until no nodes are left. This method of splitting will produce a reasonably balanced tree with good locality for the lower part of the tree. This approach is similar to the method of [Gupta et al. 2001]. An alternative is to use flooding – as in [Madden et al. 2002] – in which case the communication between parent and children will be more efficient but the tree may be unbalanced.

4.1.1 *Simulating Very Large Sensor Networks.* The ideal testbed for evaluating the hybrid protocol is a real sensor network. Since the hybrid protocol we proposed is designed to deal primarily with large networks – hundreds of thousands to millions of nodes – empirical evaluation on a real sensor network is outside the

possibilities of a small research group. Even more, accurately simulating such large networks, for example using some of the available emulators [Levis 2003] [Polley et al. 2004] [Girod et al. 2004], is computationally infeasible; the largest sensor network simulations can reasonably accommodate only thousands of nodes and the benefits of the hybrid protocol will be significant usually for much larger networks. Under these circumstances, the only reasonable solution is to use a coarse simulator that will ignore most details of sending the messages using radio communication but will simulate faults. Even simulating the multi-hop routing is an overkill for simulating large networks; for this reason we approximate the number of hops with the distance between the communicating nodes divided by the radius of communication. It is important to mention though that the simulator does not just use the model introduced in Section 2; it actually simulates sending the messages between sensors and acting on behalf of the sensors as they would behave in a real implementation. While the simplifications in how the sensor network is simulated/tested are likely to introduce inaccuracies, we believe that qualitatively the behavior of the protocols will be the same in practice.

4.2 Experimental Results

We perform two types of experiments to determine the efficiency of the hybrid protocol:

(1) We minimize the total number of messages required for an aggregation. This is a fair indicator of the amount of power consumed in the execution of the aggregation algorithm since communication consumes the most significant percentage of power in sensor networks [Shnayder et al. 2004].

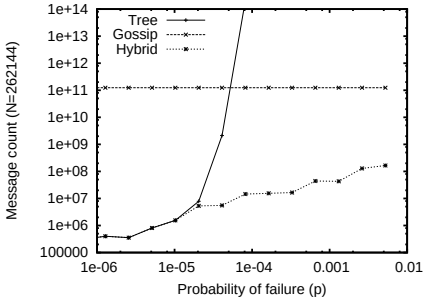
(2) We minimize the time to completion of the hybrid protocol and compare it with the best of the tree and the gossip protocol.

We perform such experiments for the three types of gossip communication: local communication with immediate neighbors, communication with any node using routing, and communication with a random subset of size $\log N$ using routing.

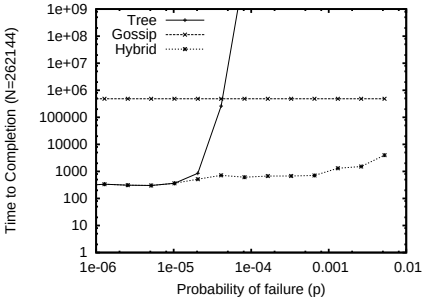
For all the experiments reported in the next three subsections, we use the following conventions:

—For each individual experiment, we provide three plots grouped in a single figure. The first plot is the absolute performance of the quantity measured/optimized for – we plot these values for each of the tree aggregation, gossip aggregation and hybrid aggregation but only for the largest network since otherwise the graphs would be too crowded. The second plot is the relative gain in performance of the hybrid protocol over the best of the tree and gossip aggregation for various network sizes. The third plot is the group size determined as the solution of the optimization problem for the hybrid protocol. For all three plots, the dependency is on the probability of failure which covers a four order of magnitude range.

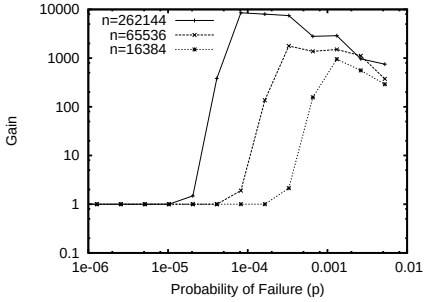
—The optimization criterion for selecting the group size of the hybrid protocol is always the quantity plotted, e.g. for plots of the total number of messages, the hybrid protocol is optimized to minimize the number of messages.



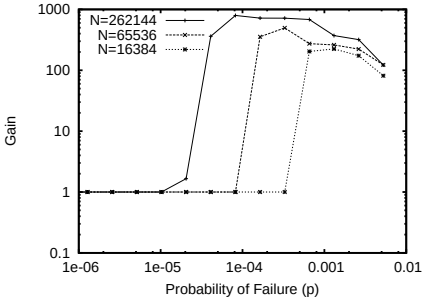
(a) Absolute values



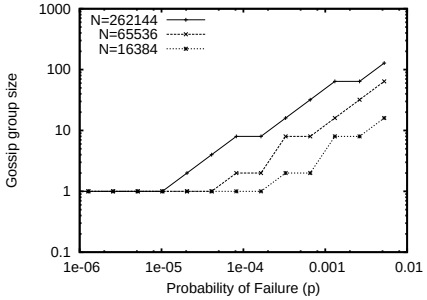
(a) Absolute values



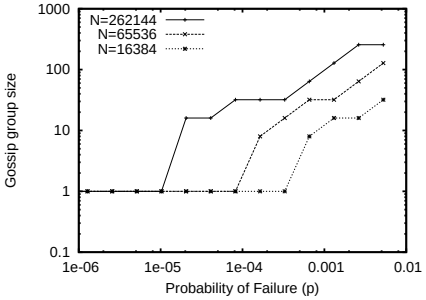
(b) Gain of hybrid protocol



(b) Gain of hybrid protocol



(c) Optimal gossip group size



(c) Optimal gossip group size

Fig. 6. Comparison of the message counts when using local communication for gossip.

Fig. 7. Comparison of the time to completion when using local communication for gossip.

4.2.1 *Local gossip communication.* When using local gossip communication, nodes are allowed to communicate only with their immediate neighbors when running the gossip (part of the) protocol – no long distance communication is allowed for gossiping but it is allowed for tree aggregation. Results of experiments using local gossip communication are reported in Figure 6 for message count and in Figure 7 for total time to completion.

From the plot of the absolute values of the total number of messages for the three protocols in Figure 6(a), we observe that the intuitions about the behavior of the tree and the pure gossip protocol is confirmed experimentally. In particular:

—For small probabilities of failure, the tree performance is five orders of magnitude better than gossip – this behavior is well predicted by the tree fault model and intuition.

—As the failure probability increases, the performance of the tree deteriorates exponentially to the point that it becomes significantly worse than the rather poor performance of the gossip. Notice that the performance goes from good to very bad when the probability of failure increases four-fold from 10^{-5} to $4 \cdot 10^{-5}$. We predicted this behavior with the tree fault model.

—The performance of gossip is mostly immune to failures – this property is intuitive as we pointed out earlier but it was never confirmed experimentally.

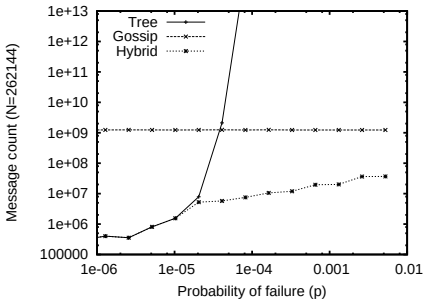
—The performance of the hybrid protocol coincides with the performance of the tree for small failure probabilities but it stays reasonable even when the tree performance becomes unacceptable. Also, at all times, the hybrid protocol significantly outperforms the gossip protocol.

—The dependency on the probability of failure of the hybrid protocol is almost linear but not as flat as it is the case for gossip.

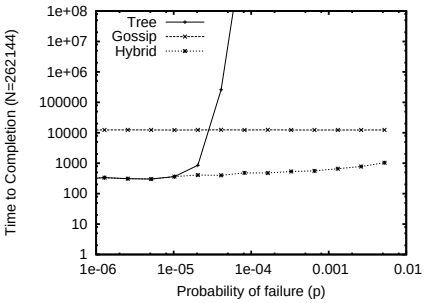
The same trends are observed in the plot of the absolute values of the time to completion in Figure 7(a) except that the gap between the tree and gossip for small probabilities of failure is significantly smaller.

From the plots of the gain of the hybrid protocol over the best of tree and gossip in Figures 6(b) and 7(b) we observe that, except for small probability of failure, the hybrid protocol is significantly more efficient both in terms of total number of messages and total time. For different network sizes, the shape of the curve of the gain is very similar; the variations being the facts that for larger networks the transition from no gain to significant gain happens for smaller failure probabilities and that the largest gain is more significant. This means that the advantage of the hybrid protocol widens for larger networks – its use is crucial instead of the pure protocols. Large gains, in the order of 10000, are observed for local communication because the gossip is very inefficient.

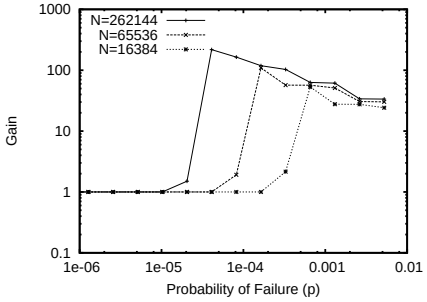
More insight can be gained on the way the hybrid protocol works by analyzing the group size determined using the optimization problem. From the plots in Figures 6(c) and 7(c) we observe that, as suspected, for small probabilities of failure, only the tree aggregation is used (group size is 1). At some point – the larger the network the sooner – the group size is increased almost linearly as a function of the probability of failure.



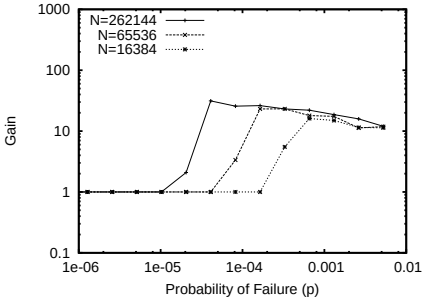
(a) Absolute values



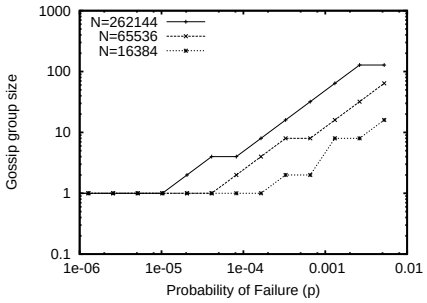
(a) Absolute values



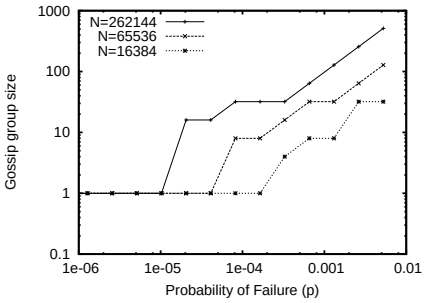
(b) Gain of hybrid protocol



(b) Gain of hybrid protocol



(c) Optimal gossip group size



(c) Optimal gossip group size

Fig. 8. Comparison of the message counts for global gossip communication.

Fig. 9. Comparison of the time to completion for global gossip communication.

4.2.2 *Global gossip communication.* When using global gossip communication, nodes are allowed to communicate with any node in the network – routing is used to accomplish this. Results of experiments using global gossip communication are reported in Figure 8 for message count and in Figure 9 for total time to completion. The same trends we observed for local gossip communication are observed in this case with the following exceptions:

—The gap between the tree and gossip communication for small probability of failures is significantly smaller: $2 \cdot 10^3$ for message counts in Figure 8(a) and only about 50 for the time to completion in Figure 9(a).

—Gains of the hybrid protocol are significantly smaller (the opportunities are not as big since the gap between tree and gossip is smaller) as can be observed from Figures 8(b) and 9(b). Nevertheless, the hybrid protocol still performs significantly better.

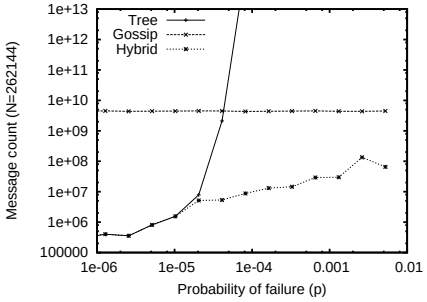
4.2.3 *Limited global communication.* Limited global communication allows nodes to communicate with any node in the network but fixes the number of such nodes for any specific node to the logarithm of the size of the network. In this way, knowledge about only a small number of nodes in the network is required. Thus, this type of communication is more practical than the global gossip. As it can be observed from the experimental results in Figures 10 and 11, the behavior of the limited global communication is very similar to the global communication except that the gossip performance is about 5 times worse both in terms of the number of messages and time to completion (Figures 10(a) and 11(a)). This gets reflected as well in the performance of the hybrid protocol. We believe that this is an acceptable compromise between performance and the need to keep every node informed about all the other nodes in the network (which might be impossible to achieve).

4.3 Robustness and Sensitivity Testing

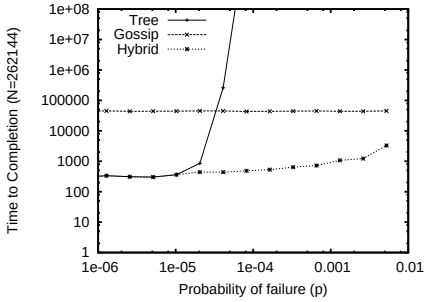
The simplicity of the fault model used to determine the values of the parameters for the hybrid protocol allows for easy theoretical analysis. The results presented so far validate the gain in performance achieved using the hybrid protocol designed using the fault model. Now, we experimentally verify the robustness of the simple fault model to differences in the actual fault behavior from the assumed fault model. We consider the following scenarios:

(1) Inaccuracies in estimation of the parameter p for the fault model: It might not always be possible to determine accurately the estimated probability of failure p. The actual probability of failure might differ from the estimated p. Consequently, we also investigate how sensitive the performance of the hybrid protocol, designed from the simple fault model, is, to deviation of actual failure rate from the estimated p used as input to the fault model.

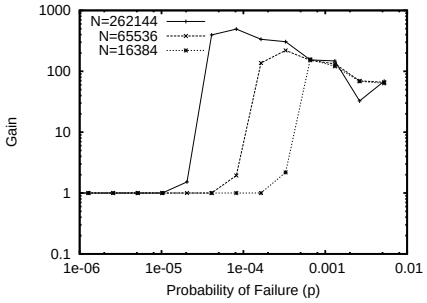
(2) Presence of correlated failures: Though the simple fault model assumes absence of any correlations in the failure rate, environmental conditions might actually create some kind of correlations – for e.g. in a battlefield, some sensors might be overrun by a tank or in habitat monitoring, sensors in damp, swampy areas may fail more often. We investigate how well the hybrid protocol, designed



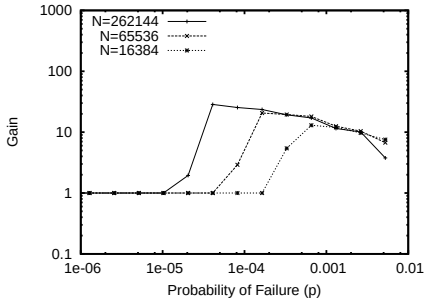
(a) Absolute values



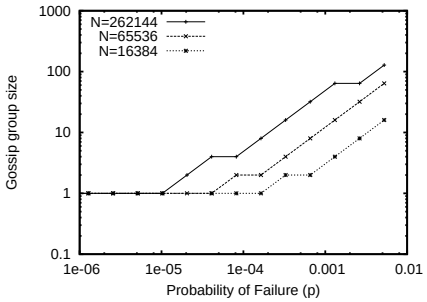
(a) Absolute values



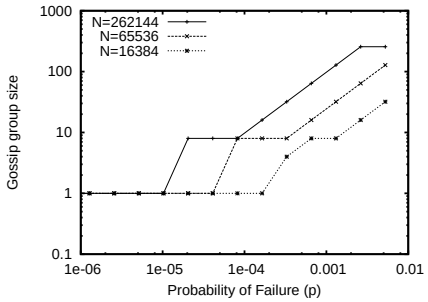
(b) Gain of hybrid protocol



(b) Gain of hybrid protocol



(c) Optimal gossip group size



(c) Optimal gossip group size

Fig. 10. Comparison of the message counts when using limited global communication for gossip.

Fig. 11. Comparison of the time to completion when using limited global communication for gossip.

from this simple fault model, copes up in situations where faults are in fact correlated.

4.3.1 *Sensitivity to failure rate.* To test the sensitivity of the fault model to the actual failure rate, we run experiments wherein the actual failure rate p_a is different from the estimated probability of failure p_e. We observe the change in gain of the hybrid protocol when $p_a = x \times p_e$. Here, we report the results for $x = \frac{1}{16}$, $x = \frac{1}{4}$ and $x = \frac{1}{2}$ in Figure 13(b) and for $x = 2$, $x = 4$ and $x = 16$ in Figure 13(a).

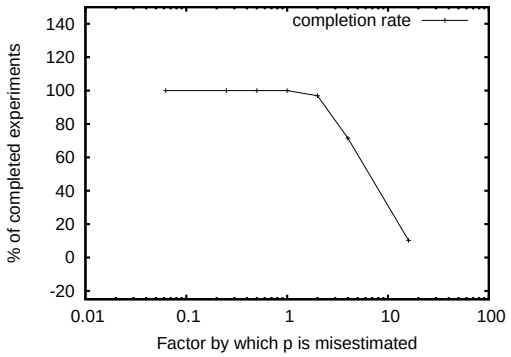
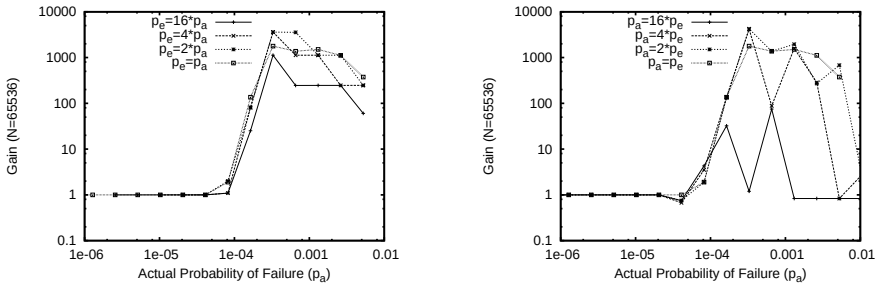


Fig. 12. Completion rate of sensitivity experiments.

From Figure 13, we observe that when the actual probability of failure is overestimated, the overall gain obtained by using the hybrid protocol is slightly worse in general than the gain achieved when the estimation is accurate i.e. when $p_e = p_a$. However, the gain is still significant and follows the trend as observed so far when the probability of failure is estimated correctly.

However, when the probability of failure is underestimated, the performance deteriorates exponentially. An important thing to note when the probability of failure is underestimated is that the number of instances of aggregation experiments that do not complete[4], as seen in Figure 4.3.1, rapidly increases. Due to the difficulties of dealing with the results the experiments that did not complete, the results in Figure 13(b), that includes the results of the experiments which failed to complete, should be interpreted qualitatively rather than quantitatively. In particular, the performance of the hybrid protocol varies considerably due to wide variation in the time it takes to successfully perform the tree aggregation (the distribution of this time is geometric and, as explained in Section 2.3, for small p, the standard deviation is almost equal to the expected value, which is high) and in general the performance is poor. This suggests that care should be taken not to underestimate the probability of failure.

[4]In our experiments we used a cutoff time of $5 \cdot 10^5$ rounds.



(a) Overestimated p (b) Underestimated p

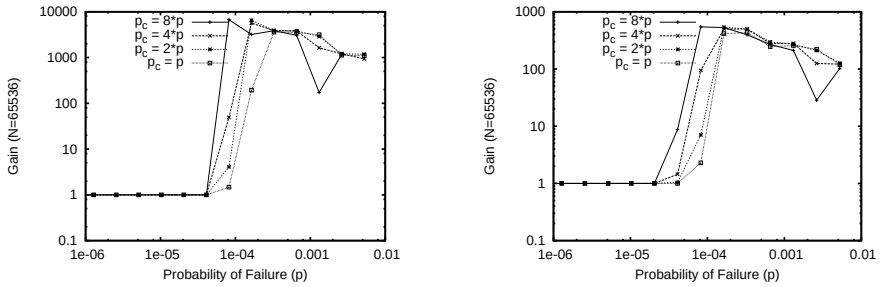
Fig. 13. Gain of the hybrid protocol when the estimated p varies from the actual p. The gossip phase of the hybrid protocol here uses local communication.

These results confirm the theoretical analysis of the fault model in Section 2.3 where we showed that the expected number of times a tree has to be reconstructed is e^q with $q = pd(N)N$. When the probability of failure is underestimated, q is underestimated thus the number of required reconstructions is underestimated. If the target value for q is 1 and we choose a tree size that achieves this for probability of failure p_e, if the probability of failure was in fact $p_a = 4p_e$, then we get $e^4 = 54.6$ times more tree reconstructions than we think we are, which leads to the poor performance observed. When we overestimate the probability, say also by a factor of 4, the number of restarts is $e^{1/4} = 1.28$ instead of the estimated $e^1 = 2.7$. However, this leads to some performance degradation due to the fact that the gossip groups are made larger than the *optimal size*; but, the performance degradation is small compared to the underestimation case. These experimental results thus confirm our theoretical analysis and suggest that it is better to be on the safer side and overestimate the probability of failure rather than underestimating it.

4.3.2 *Robustness to correlated failures.* We induce spatial correlation in the actual fault behavior by making a set of sensor nodes in the center of the field fail at a different rate than the rest of the sensor nodes. More specifically, the rate of failure (p_c) of nodes falling in the circular region at the center encompassing l% of the unit square field area is x times higher. The results presented here are for $x = 2$,$x = 4$ and $x = 8$ and $l = 20\%$.

As we can see in Figures 14(a) and 14(b), the gains of the hybrid protocol even in cases of correlated failures, are comparable to the gains obtained in cases where all failures are uncorrelated. It is important to note that the hybrid protocol used here is in fact designed from the simple fault model which assumes no correlation. Note that the slight increase in gains in some cases of correlated faults ($8 \cdot 10^{-5} < p < 4 \cdot 10^{-4}$ here) is due to the fact that the pure tree protocol performs even worse in such situations and that the gain plotted is the ratio of the performance of the hybrid protocol to that of the best of pure tree and gossip protocols.

This implies that the proposed fault model, which assumes that the faults are uncorrelated, is in fact robust to common cases involving correlated faults.



(a) Message count

(b) Time to completion

Fig. 14. Gain of the hybrid protocol in presence of spatially correlated failures. The gossip phase of the hybrid protocol here uses local communication.

4.4 Seeded Clustering

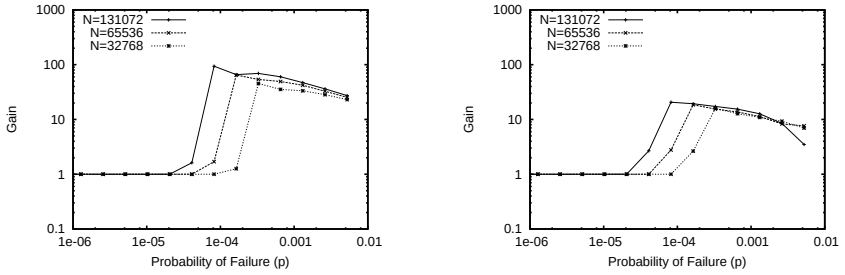
The *grid clustering* technique used so far for clustering sensor nodes into groups needs location information for each of the sensors. If such a requirement is to be avoided, other clustering techniques can be used [Amis et al. 2000] [Bandyopadhyay and Coyle 2003]. We experimented using a simple seeded clustering technique similar to [Bandyopadhyay and Coyle 2003] – each node selects itself as a cluster head with a probability proportional to the required number of clusters and *grows* clusters around it using flooding messages. Since the balls in bins theory cannot be used here, we used a learning technique to estimate the size of the largest gossip group – which is expected to be the slowest – for the factor c_2 in Equation 1. We derive, experimentally, the size, say $y(x)$, of the largest group that is formed when the expected group size is x. Now, the convergence time of $y(x)$ gossip nodes is considered when we want to count in the maximum convergence time for an expected group size of x. The factor c_1 can also be just set to 1 if we can learn $z(x)$ – the expected value of the *slowest* convergence rate when the expected size of the gossip group is x – and use $z(x)$ as $T_{\text{gossip}}(x)$ in Equation 1. The experimental results provided in Figures 15(a) and 15(b) show that the gains achieved follow a trend similar to those(8(b) and 9(b)) with the *grid clustering* technique used previously.

While this scheme might seems more involved than grid clustering, it is worth pointing out that the clustering of nodes into groups is a one-time operation whose cost is amortized over time.

5. DISCUSSION

We make the following important observations about the hybrid protocol we proposed:

—Even though the fault model we proposed and used to determine the parameters of the hybrid model ignores correlated failures, the hybrid protocol obtained in this manner is robust with respect to such correlations and deviations from the estimated failure probabilities. This suggests that considering complicated failure



(a) Message count

(b) Time to completion

Fig. 15. Gain of the hybrid protocol using seeded clustering. The gossip phase here uses global communication. Similar results are obtained when local or limited global communication is used in the gossip phase.

models might not provide a significant benefit.

—As we have shown, the greatest benefit of the hybrid protocol is obtained either when the failure rate is very high – this would happen for example when sensors are deployed in harsh environments such as volcanoes or hurricanes – or when the size of the network is large – for example for large scale scientific experiments say for ocean monitoring.

—Implicitly we assumed that all sensors have the same capabilities with respect to communication rates and, to a lesser degree, reliability. Should this not happen, the more capable nodes should always participate in the tree protocol where reliability is crucial. We believe that a modification of the hybrid protocol we propose here can naturally accommodate such situations. We plan to explore this possibility in future work.

—In this paper we did not explore local fixes as a possibility, mostly because this issue is not explored in the literature and, in our opinion, is nontrivial. While some implementations of simple local fixes exist, it is not obvious how to attach any notion of correction to the type of fixes proposed/implemented. Providing some guarantees is crucial when sensor networks are used.

—The notion of correction we proposed in this work, *robust correctness*, looks complicated and not very intuitive. As we explained previously, it is not possible to give a simple notion of correctness since it would imply achieving global knowledge of one sort or another. Some of the difficulty with understanding robust correctness comes from the fact that any notion of correctness has to be expressed mathematically – an intuitive/descriptive notion of correctness would not be satisfactory since it would not provide any quantifiable guarantee.

—The work in this paper should be viewed more as exploratory work rather than an actual complete solution. Large sensor networks were not explored in previous literature and, as we have shown, they pose severe difficulties that are not encountered in small sensor networks. In this paper we showed the feasibility of combining, in a principled way, different simple protocols to significantly improve the performance. We believe this is the only way to scale protocols to large sensor

networks.

6. RELATED WORK

Madden et al. [2002] provided the first system implementation of aggregate computation in sensor networks. They opted for simple solutions – this is highly desirable due to the limitations of the hardware used – with a focus on acceptable, if not guaranteed, solutions. While not providing a fault model, a number of methods to deal with the faults were proposed: child-caches – caching old values of aggregates at parents in case communication with the child cannot be established – and redundant trees – instead of sending the aggregate value to a single parent, the aggregate value is split between multiple parents. However, in the presence of faults, these do not guarantee robust correctness (or any strict notion of correctness). While this solution works extremely well for small sensor networks – up to 1000 – it is problematic for the large sensor networks we consider in this paper.

As we mentioned in the text, Chen et al. [2005] proposed a variation of the Push-sum protocol of Kempe et al. [2003] that, in the case when wireless broadcast is available, performs slightly better (20% in most cases). While we performed the experiments in this paper using Push-sum, this algorithm can be used as a drop in replacement for Push-Sum protocol in the gossip phase of the hybrid protocol and would have resulted in slightly different, but qualitatively the same behavior. This algorithm essentially has the same properties as Push-sum does; thus is too slow when compared to tree aggregation for small failure probabilities.

Bawa et al. [2003] proposed an alternative to the gossip protocols for fault resilient aggregate computation. As is the case for gossip, it consists in nodes contacting other nodes (the particular scenario considered in this paper is peer-to-peer network based communication – which is essentially the limited global communication we considered in this paper) randomly and exchanging information. In order to limit the amount of information transmitted, approximation techniques based on Flajolet-Martin(FM) sketches [Flajolet and Martin 1985] are used. The FM sketches can estimate the number of distinct values and are thus immune to duplicates – this is desirable for the type of communication in this paper([Bawa et al. 2003]) since the value of a node can make its way to other parts of the network via multiple paths, thus creating duplicates. It is possible to compute approximately aggregates like sum using this scheme. The main disadvantage of this protocol is the fact that the messages exchanged between nodes are large since they have to contain an entire sketch (the size of the sketch dictates the precision of the computation). The speed of convergence of the algorithm is similar to the speed of gossip (it is essentially the *algebraic complexity* of the communication graph as shown in [Chen et al. 2005]).

Nath et al. [2004] also identified the vulnerability of the tree-based aggregation techniques at higher loss rates and proposed a multi-path aggregation algorithm, called synopsis diffusion. By introducing redundancy, duplicates are possibly created, as was the case in the work of Bawa et al. [2003]. The method used to deal with this problem is essentially the same, namely FM sketch techniques. While the convergence is faster than gossip in terms of the number of messages, this method suffers from the same problem – messages have to be large to achieve reasonable

precision. Manjhi et al. [2005] extended this work to take advantage of regional differences in the failure probability. Here, a hybrid algorithm is proposed that utilizes trees in the low failure regions and synopses diffusion in the high failure regions. This is very different from the hybrid protocol we are proposing which uses the fact that the performance of the tree deteriorates as their size increases. The technique of Manjhi *et al.* [Manjhi et al. 2005] is orthogonal on our approach; we believe that the two can be combined to obtain a protocol that adapts to regional differences of probability of failures as well.

Gupta et al. [2001] introduced grid-box hierarchies and use gossiping to reach consensus among members of a grid box. The protocol works by escalating a member of the group – that reaches consensus using gossip – to the next level in the tree where gossip is again used to reach consensus in the enclosing grid box. The gossip used in this paper is not a gossip aggregation protocol like Push-sum (which was introduced later) but gossip communication. Each message has to contain the values contributed by all the nodes a particular node has heard about, thus the message size will be proportional to the size of the group instead on a small constant (2 for computing averages with the gossip aggregation algorithms, the tree or the hybrid protocol).

Tree-assisted gossiping has also been used for video-on-demand in [Zhou and Liu 2005] for combining speed and robustness. However, they are using trees in conjunction with gossiping, rather than in stages as the case is in this paper and the method does not apply to the problem of computing aggregates in sensor networks.

Our methodology for optimizing the gain of the hybrid approach involves the use of results obtained from theoretical and simulation runs. A similar strategy is used in [Pastor and Pena 2003] and [Hu 2005]. Pastor and Pena [2003] exploit the information obtained from simulation to effectively guide the symbolic traversal for verification of concurrent systems. Hu [2005] too use the engineering knowledge of the system to guide their dynamic probabilistic risk assessment to achieve higher efficiency and accuracy.

7. CONCLUSION

In this paper we proposed a new protocol, that we called hybrid protocol, for computing aggregates in large – hundreds of thousands – and/or highly faulty sensor networks that combines in a principled way two existing protocols: tree and gossip aggregation. Our approach is to first determine the behavior of the two existing protocols under faults to identify when they are/are not acceptable and then to combine them in a way that capitalizes on their strengths and minimizes their weaknesses. Instead of an heuristic combination, based on the fault model we formulated and solved an optimization problem that gives the *optimal* combination between the pure protocols. We empirically evaluated the performance of the hybrid protocol and observed that it is at least as good but often significantly outperforms both tree and gossip aggregation.

We believe that, for large networks, no simple (pure) protocol is likely to be the best in most scenarios; a hybrid protocol like the one we described in this paper would be required to achieve acceptable performance.

REFERENCES

Amis, A. D., Prakash, R., Huynh, D., and Vuong, T. 2000. Max-min d-cluster formation in wireless ad hoc networks. In *IEEE INFOCOM*. IEEE press, Tel Aviv, Israel, 32–41.

Bandyopadhyay, S. and Coyle, E. 2003. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*. Vol. 3. Springer-Verlag, San Franciso, CA, USA, 1713– 1723.

Bawa, M., Garcia-Molina, H., Gionis, A., and Motwani, R. 2003. Estimating aggregates on a peer-to-peer network.

Chen, J.-Y., Pandurangan, G., and Xu, D. 2005. Robust aggregates computation in wireless sensor networks. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, Los Angeles, California, USA.

Ding, M., Cheng, X., and Xue, G. 2003. Aggregation tree construction in sensor networks. In *Vehicular Technology Conference*. IEEE, Orlando, Florida, USA.

Fiedler, M. 1973. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal 23(98)*, 298–305.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. 1985. Impossibility of distributed consensus with one faulty process. *J. ACM 32,* 2, 374–382.

Flajolet, P. and Martin, G. 1985. Probabilistic counting algorithms for database applications.

Girod, L., Elson, J., Cerpa, A., Stathopoulas, T., Ramanathan, N., and Estrin, D. 2004. Emstar: a software environment for developing and deploying wireless sensor networks. In *USENIX Technical Conference*. USENIX, Boston, MA, USA.

Gupta, I., van Renesse, R., and Birman, K. 2001. Scalable fault-tolerant aggregation in large process groups.

Gupta, I., van Renesse, R., and Birman, K. P. 2000. A probabilistically correct leader election protocol for large groups. In *DISC '00: Proceedings of the 14th Conference on Distributed Computing*. Springer-Verlag, London, UK, 89–103.

Hu, Y. 2005. A guided simulation methodology for dynamic probabilistic risk assessment of complex systems. Ph.D. thesis, Dept of Mechanical Engineering, Univeristy of Maryland.

Jia, L., Noubir, G., Rajaraman, R., and Sundaram, R. 2006. Group-independent spanning tree for data aggregation in dense sensor networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Springer Berlin / Heidelberg, San Francisco, CA, USA.

Kempe, D., Dobra, A., and Gehrke, J. 2003. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Cambridge, MA, USA.

Levis, P. 2003. Tossim: Accurate and scalable simulation of entire tinyos applications.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2002. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev. 36,* SI, 131–146.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2003. The design of an acquisitional query processor for sensor networks. In *SIGMOD Conference*. ACM, San Diego, California, 491–502.

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J. 2002. Wireless sensor networks for habitat monitoring. In *ACM Workshop on Wireless Sensor Networks and Applications (WSNA'02)*. ACM, Atlanta, GA.

Manjhi, A., Nath, S., and Gibbons, P. B. 2005. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD conference on Management of data*. ACM Press, New York, NY, USA, 287–298.

Nath, S., Gibbons, P. B., Seshan, S., and Anderson, Z. R. 2004. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys '04: Proceedings of the 2nd Conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 250–262.

Pastor, E. and Pena, M. A. 2003. Combining simulation and guided traversal for the verification of concurrent systems. In *Design, Automation and Test in Europe Conference and Exhibition*. IEEE, Messe Munich, Germany.

Polley, J., Blazakis, D., McGee, J., Rusk, D., and Baras, J. S. 2004. Atemu: A fine-grained sensor network simulator. In *SECON'04, The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*. IEEE, Santa Clara, CA.

Raab, M. and Steger, A. 1998. "balls into bins" - a simple and tight analysis. In *RANDOM '98: Proceedings of the Second Workshop on Randomization and Approximation Techniques in Computer Science*. Springer-Verlag, London, UK, 159–170.

Rao, S. 2004. Lecture notes on balls and bin analysis.

Shao, J. 2003. *Mathematical Statistics*, 2 ed. Springer Texts in Statistics.

Shnayder, V., Hempstead, M., Chen, B., Allen, G. W., and Welsh, M. 2004. Simulating the power consumption of large scale sensor network applications. In *SenSys*. ACM, Baltimore, Maryland, USA.

Werner-Allen, G., Swieskowski, P., and Welsh, M. 2005. Motelab: A wireless sensor network testbed.

Zhou, M. and Liu, J. 2005. Tree-assisted gossiping for overlay video distribution. In *Kluwer Multimedia Tools and Applications*. Springer Netherlands.