A Cross Layer, Adaptive Data Aggregation Algorithm Utilizing Spatial and Temporal Correlation for Fault Tolerant Wireless Sensor Networks

Babak D. Beheshti
Electrical & Computer Engineering Technology
New York Institute of Technology
Old Westbury, New York, USA
bbehesht@nyit.edu

Howard E. Michel
Electrical & Computer Engineering Department
University of Massachusetts Dartmouth
North Dartmouth, Massachusetts, USA
hmichel@umassd.edu

Abstract— Wireless Sensor Networks (WSNs) are ad hoc networks formed by tiny, low powered, and low cost devices. WSNs take advantage of distributed sensing capability of the sensor nodes such that several sensors can be used collaboratively to detect events or perform monitoring of specific environmental attributes. Since sensor nodes are often exposed to harsh environmental elements, and normally operate in an unsupervised fashion over long periods of time, within their MTBF, some of them are subject to partial failure in form of A/D readings that are permanently off the correct levels. Additionally, due to glitches in timing and in hardware or software, even healthy sensor nodes can occasionally report readings that are outside of the expected range.

In this paper we present a novel approach that combines spatial and temporal correlation of the data collected by neighboring sensors to combat both error modes described above. We combine the weighted averaging algorithm across multiple sensors, with the LMS adaptive filtering of individual sensor data, in order to improve fault tolerance of WSNs. We present performance gains achieved by combining these methods; and analyze the computational and memory costs of these algorithms.

Keywords—Wireless Sensor Networks; Cross Layer; Aggregation

I. INTRODUCTION

Multi-sensor data aggregation is commonly used to combine the data collected from neighboring sensors to accomplish one or both of the following objectives: 1) To reduce the amount of data required to be transmitted to the sink, and thereby, reducing transmit energy of the battery operated sensor nodes, 2) To improve accuracy of the information obtained, through “fusion” of the data from multiple/complementary sensors. There is a body of work that presents algorithms to benefit from the spatial correlation between the data collected from multiple sensors to reduce the effects of a permanently faulty sensor. These algorithms are based on weighted averages. In the event of a sensor failure, the algorithm adaptively decreases the weight for sensors which have failed or demonstrate probability of failure. Meanwhile, weighting factors for neighboring sensor nodes are increased. These algorithms respond well to sensors reporting out of normal range values due to a permanent failure mode (low battery, failed transducer...), however, do not respond quickly to occasional data misreads. This is because the weight of the misread sensor cannot be changed quickly enough to minimize the effect of a single misread on the averaged data.

Depending on how/if the in-network aggregation is performed, there are three main techniques published in the literature [4]

1. Centralized Processing. The most straightforward 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.

2. Tree Aggregation. Here, we organize all the sensor nodes into an aggregation tree. The root of the tree is the node where the query is injected and where the aggregation result is retrieved. 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 efficiency of the algorithm depends on the longest route from root to the leaf, that is, how balanced this tree is.

3. 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 where routing messages through the network are possible, any other nodes) and exchanges information with these nodes.

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, tree aggregation is more efficient by at least a significant constant factor in these scenarios [4]. The place where gossip aggregation excels is in networks where nodes can fail with significant probability.

We propose a tree based aggregation in our research; as this approach eliminates the need for additional communication cost for gossip based aggregation. In particular, we overlay our aggregation tree over the cluster based communication network already established in the sensor network. It is assumed that the
sensor network established is based on a least cost metric to minimize communications energy costs of transmitting to the sink (base station). We are using the xMesh network formation which is based on the Minimum Transmission (MT) Cost Metric [5]. The purpose of the cost metric is to minimize the total cost it takes to transmit to the base station Mote.

II. PROBLEM STATEMENT

Using the Information Centric face of the Technical Reference Manual (IC-TRM) and behavior centric face of the Technical Reference Manual (B-TRM) [1], [3], we develop a hybrid multi-layer data aggregation algorithm that is more robust with regard to inaccuracies due to:

1. Sensing errors
2. Component errors
3. Communication errors
4. Aggregation errors

We provide qualitative performance evaluation of this hybrid approach. Also, we outline how the information necessary to implement this hybrid algorithm is transmitted between the different layers of our TRM.

III. OUR APPROACH

A. Node Permanent Failures Due to Communication Errors

We define communication errors as total loss of communication to a particular sensor node. 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 the aggregation simply by increasing the time and the number of messages by a constant factor.

Due to its inherent reliance on the underlying mesh network, our approach automatically compensates for these types of permanent communication failures. Therefore once a node fails to report it must be removed from the aggregation scheme. This is accomplished through a number of interlayer messages as follows:

A state machine is in progress at layer 5 of B-TRM continuously monitors reporting status of all sensor nodes. Once a node is deemed failed by exceeding its reporting deadline, a message will be generated destined to layer 4 of I-TRM to exclude this node from any further aggregation. There are two possibilities here:

1. The failed node is a leaf node: In this case our scheme requires a message transmitted from the base to the cluster-head, indicating loss of the failed leaf node. This transmission is a message from the base to the cluster head (n-hops), and within the cluster head, from L4 of B-TRM to L4 of IC-TRM.

2. The failed node is cluster head: This case is more complicated, as once the loss of the cluster head is detected, L5 of B-TRM needs to obtain the ID of the new cluster as assigned by the underlying sensor network protocol. Once layer 5 (L5) of B-TRM obtains the ID of the new cluster head, it sends a aggregation tree reconfiguration request to the L5 of C-TRM. L5 of C-TRM in turn issues a reconfiguration command that will arrive at the new cluster head. There is no need to inform the cluster members of this reconfiguration because they were already involved in the new cluster head selection protocol (xMesh tree formation in our case) [5].

B. Node Temporary Failures Due to Sensing Errors

The Least-Mean-Square (LMS) algorithm is a method to estimate gradient vector with instantaneous value. It changes the filter tap weights so that an error signal e(n) is minimized in the mean-square sense. The conventional LMS algorithm is a stochastic implementation of the steepest descent algorithm. It simply replaces the cost function \( \xi(n) = E[e^2(n)] \) by its instantaneous coarse estimate. The error estimation \( e(n) \) is

\[
e(n) = d(n) - w(n)X(n)
\]

Coefficient updating equation is

\[
w(n+1) = w(n) + \mu(x(n)e(n)
\]

where \( \mu \) is an appropriate step size to be chosen as \( 0 < \mu < 0.2 \) for the convergence of the algorithm. The larger step sizes make the coefficients to fluctuate wildly and eventually become unstable [4].

In Normalized LMS (NLMS), the step size takes the form of

\[
\mu(n) = \frac{\beta}{||x(n)||^2}
\]

Where \( \beta \) is a normalized step size with \( 0 < \beta < 2 \).

When \( x(n) \) is large, the LMS experiences a problem with gradient noise amplification. With the normalization of the LMS step size by \( ||x(n)||^2 \) in the Normalized LMS (NLMS), noise amplification problem is diminished.

Relying on the inherent temporal coherence between consecutive samples collected by one sensor node, we can eliminate transient invalid measurements in a sensor, by applying the NLMS algorithm continuously to the sampled data within each sensor node. Once the learning period of the NLMS adaptive filter is completed, short term glitches in the data are “smoothed out” by the filter. Figure 1. below illustrates simulation results for a 4th order NLMS filter. As can be seen, by the 40th sample, the output of the filter (red curve) converges and from that point on, large transients in the input signal are suppressed significantly. Should the characteristics of the input signal change drastically for the long term, the adaptive filter will go through another learning period before converging on the new signal behavior. Should the behavior of the input signal change frequently, this technique will suffer due the frequent learning periods. Figure 2. from [4], summarizes the convergence rates of LMS algorithms for two step sizes. The results are comparable to our simulation results.
C. Node Permanent Failures Due to Component Errors

What happens to aggregated data, if a node (or some nodes) experiences component failure so that while it is communicating its measurement results to its cluster head, the actual measured data is no longer valid? We propose to solve this problem by taking advantage of the spatial coherence present in measurements made by neighboring sensors. Sridhar et al. in [6] describe their approach that is based on weighted averaging. The authors state that with weighted average, the user can set weights and thus can control the aggregation process. Since weighted average is stable and not computationally intensive, it is well suited for data aggregation in sensor networks.

The process is described as follows: Each sensor node has a weighting factor at any instance of time, given by $w_i(t)$. In the event of sensor failure, the proposed algorithm adaptively decreases the weight for sensors which have failed or demonstrate likelihood to fail. At the same time weighting factors for neighboring sensor nodes increase. Hence, every reading from each sensor is weighted at each predetermined time interval, and weight updates are computed as follows:

$$w_i(t + 1) = w_i(t) \pm \Delta w_i(t)$$

(4)

In order to estimate $\Delta w_i(t)$, they propose the following model:

$$\Delta w_i(t) = |r_i| \times \varepsilon$$

(5)

Where $\tau_i$, the adaptation parameter, is given by:

$$\tau_i = \frac{r_1 + r_2 + \ldots + r_i + \ldots + r_k}{k}$$

(6)

where $r_i$ is the reading from the $i^{th}$ sensor, $k$ is the number of neighboring sensors, and $\varepsilon$, the scaling factor, is a small value and is chosen appropriately for a given application. The scaling factor ensures that $0 < \Delta w_i(t) < 1$.

An algorithm for updating the weights is presented in the following:

1. Initialize all weights at $t=0$, $w_i(0) = 1$.
2. At time $t+1$, calculate $\tau_i$ for all the sensor readings within the region of event.
3. Calculate $\Delta w_i(t) = |\tau_i| \times \varepsilon$. Choose $\varepsilon$ appropriately.
4. For $i = 1$ to $k$:
   a. if $\Delta w_i(t)$ is minimum for all $i$, then $w_i(t + 1) = w_i(t) + \Delta w_i(t)$
   b. if $\Delta w_i(t)$ is above minimum for all $i$ then $w_i(t + 1) = w_i(t) - \Delta w_i(t)$
5. Repeat Steps 2–4 for next time interval.

We propose to use this algorithm to adaptively eliminate the effect of permanent or long term invalid measurements by a sensor node from the data aggregated within a cluster. We have simulated this approach in a three sensor node scenario. As can be seen in Figure 3, sensors 1 and 2, continue measuring a physical phenomenon consistently. Sensor 3 however, experiences a component failure and its measurements start deviating away from the norm as compared to its neighbors.

Figure 4. below illustrates the simulation results for the case above with two possible aggregation methods. One curve shows the Mean Square Error (error between the ideal aggregated data and the actual aggregated data) for the case where no fault tolerance is built in. As can be seen, the MSE simply increases to some saturated value as sensor 3 experiences its component failure. The second curve, however, implements the weighted fault tolerant aggregation, in which after some adaptation period, the faulty sensor’s output is eliminated and the MSE converges back to 0.

In order to estimate $\Delta w_i(t)$, they propose the following model:
In our design, management of this aggregation mechanism resides in L4 of B-TRM, where the actual aggregation takes place in L4 of IC-TRM. In our initial proposal, the effects of component failure on the part of the cluster head were for future study.

L4 of B-TRM manages the parameters associated with the aggregation execution. Such parameters include whether Weighted Fault Tolerant Aggregation is enabled or not, whether approximate replication enabled or not, thresholds for approximate replication etc. Reconfiguration of a cluster for a new aggregation tree formation takes place in this layer as well.

L3 of B-TRM is responsible for management of the NLMS filtering parameters, including whether it is enabled or disabled etc.

3) Cross-Layer Module

To expedite some fast response reconfigurations in the I-TRM, we now propose a limited cross layer module. We envision this module to be limited simply because there are two major risks in any cross layer solution. One risk is that it generally decreases the level of modularity between the layers. This implies a coupling and interdependence between the layers that would make it difficult for future improvements and innovations in an existing cross layer solution. The second risk is the possible instability caused by unintended functional dependencies. In other words the interactions between cross layer parameters may cause issues that are not obvious at design time [16]. However, there are cases where bypassing the traditional layered models will achieve system level efficiencies.

Alongside the cross-layer design proposals discussed earlier, initial proposals on how cross-layer interactions can be implemented are also being made in the literature. These can be put into three categories [17]:

1. Direct communication between layers
2. A shared database across the layers
3. Completely new abstractions

We have selected the shared database approach as it requires no additional API between non-adjacent layers. Currently in the context of the aggregation sub-system, the following parameters will reside in the cross layer module: Current or new cluster head for each cluster, aggregation control parameters and thresholds.

B. Hybrid Data Reduction Based on Approximate Replication of Data Using Adaptive Filters

Approximate replication is highly effective in scenarios involving special cases of distributed systems such as Wireless Sensor Networks (WSN) since it limits the transmission of sensed data from the node to the sink according to the prediction ability of the mechanism. The proposed model [7] introduces coordinating filters that are deployed at the sink and the cluster head (or in general the aggregating node) to adapt and predict the sensed information. The model is primarily divided into two modes:

1. The adaptation mode
2. The prediction mode.

During the adaptation mode the filter uses the sensor measurements to adapt the sensing environment. In this mode the measurements are relayed unaltered to the cluster head while filter at the cluster head is idle. Once the filters are converged they begin to predict and if the difference between...
the predicted output and the measured data remains below a predefined threshold for N samples the model moves to the prediction mode. Once in prediction mode, the sensing nodes do not relay the measured information to the cluster head; rather the estimated output at the cluster head is treated as the sensed data. Therefore while in prediction mode, NO DATA is transmitted from the sensing nodes to the cluster head. The authors in [7] propose the coordinating filter to reside in the sink. We incorporate this hybrid algorithm in our data aggregation sub-system. In our proposal we keep this algorithm local at the cluster level, hence each cluster head maintains adaptive filters for all the nodes within its cluster. This modification avoids multi-hop transmissions during the adaptation mode to the sink, thereby significantly reducing the communication energy cost for clusters far away from the sink. Also the threshold in this algorithm is application specific and will be a user defined parameter that gets relayed to the IC-TRM, through the C-TRM.

![Hybrid Multi-Layer Data Aggregation Block Diagram](image)

**Figure 5. Hybrid Multi-Layer Data Aggregation Block Diagram**

V. FAILURE MODES RECOVERY

A. Node Permanent Failures Due to Communication Errors

As stated earlier we rely on the sensor network communication infrastructure to determine the hierarchical aggregation topology in our sensor network. We now enhance our algorithm with the goal of maximizing the network lifetime. The lifetime of the network is defined as the time from the initial deployment of the network to the time that the first sensor runs out of energy in the network [8]. Assuming equal sensor node capabilities (energy source, processing power etc.), this can be interpreted as the time that the node with highest energy consumption dies. Obviously cluster heads, once saddled with the added functionality of being aggregating nodes, will suffer from the highest rates of energy consumption in the network. To randomize this added energy burden within a cluster, we propose a “rotating” aggregation node function [6]. That is based on an optimization algorithm; the role of aggregation node will be rotated between those nodes in the cluster that are not leaf nodes. This optimization algorithm can be a joint, cross-layer one, incorporating parameters from the physical and routing layers, as in [9] and [10]. However, we propose a simpler round robin algorithm here to decouple the I-TRM from the physical layer constraints. It should be noted however that in some implementations of the I-TRM, such cross layer joint optimizations are permissible. Management and control of this algorithm resides in L4 of the B-TRM of each node, since L4 is the highest layer typically present in a sensor node.

B. Node Temporary Failures Due to Sensing Errors

Our adoption of the hybrid data reduction based on approximate replication of data using adaptive filters, allows us to reduce the effects of transient noise in the sensed data, as described before, coupled with communication cost reduction in the prediction mode of the operation. Furthermore any change in the behavior of the sensed data (beyond a short term transition), will cause the local sensing node’s filter to move to the adaptation mode, and thereby start transmitting data to the aggregating node. If we consider the probability of a sensor node’s data changing its behavior in one transmission period T to be denoted as p, then the probability of it not transmitting data (due to it being in prediction mode) is:

\[ q = 1 - p \]  

(7)

The theoretical reduction in transmission cost per transmission period T for each sensor node is therefore:

\[ TCR = k.q \]  

(8)

where k is a constant describing the cost of transmission of one data packet.

For a cluster with n nodes, the transmission cost reduction is therefore:

\[ TCRn = (k.q)n \]  

(9)

Our more robust aggregation algorithm however, must be capable of handling cases where time behavior of the sensed data changes frequently and therefore the computational cost of the approximate replication is not warranted. Therefore we institute a user defined parameter in the C-TRM that enables/disables this algorithm in the B-TRM.

C. Node Permanent Failures Due to Component Errors

In our initial proposal of the adaptive weighted fault tolerant aggregation, we did not address the situations where a cluster head is also sensing data as well. If we consider a cluster to be a graph where each node is a vertex and the link between each two nodes is an edge in the graph. With a simple graph transformation, we can separate the cluster head into two vertices, one being the aggregating node, and the other the sensing node. Therefore the same analysis would apply to this case as well.

1) Aggregation Errors

Inaccuracies due to aggregation errors were not considered in our initial proposed algorithm. Sources of aggregation errors that we consider in our proposed algorithm are listed as follows:

Error due to some nodes not reporting within an epoch – this is called data availability constraint [9]. An illustrative example of this constraint is as follows. Assume node A and B work as the source nodes. However, B is not directly connected to the aggregation node C. An intermediate node N relays data for B. Suppose at time t1 A has delivered some data to C,
whereas B’s data has not arrived at C since they are delayed at node N. At this moment, although lots of A’s packets are waiting in its buffer, node C needs to wait until B’s data arrives at time t2. If C chooses to deliver A’s packets at time t1, it has to do the same job again when B’s packets arrive. In either case the aggregation result may be inaccurate. For example if the aggregation function is average, with B’s data missing, there will be an aggregation error.

2) Errors due to including data samples from different time instance

This can be exemplified by for example a situation where the aggregating node averages a sample from Node n, collected at time ti ([n, ti]) with a sample from Node m, collected at time tj ([n, tj]). In situations where relative timing of samples from different sensors is important (e.g. determining angle of arrival of a phenomenon), mixing samples from different time instances is not permissible.

3) Duplicate data received by the aggregating node in a duplicate sensitive aggregation function (i.e. sum, average, count)

We address the above issues as follows: problem 1 is addressed by incorporating a timestamp in the aggregating algorithm. We include a timestamp to be attached to every data sample transmitted from a sensing node. This implies existence of a synchronization protocol within the sensor network. We propose a localized synchronization protocol in which each cluster maintains its own time synchronization. This is a hierarchical protocol in which each parent maintains synchronization with its children. Problem 1 above is addressed by including the number of participating nodes in the aggregation function. In addition a confidence factor $\alpha$ is attached to the aggregated data at each aggregating node. At each hop towards the sink, the aggregated data is augmented with the number of samples used to arrive at that result, in addition to the confidence factor. The confidence factor can be a weighted sum of several factors including a sensor node’s position, the importance of its data, cluster’s density $\eta$ (as defined by the number of nodes per unit area) etc.

Problem 2 is automatically addressed by never combining in an aggregation function, samples from differing time stamps. Avoiding introduction of aggregation errors due to asynchronous samples is addressed here. However, we do not address mechanisms that attempt to reduce the probability of samples with differing timestamps. This is a rate optimization problem that is addressed in the literature extensively [11], [12], [13] and [14].

Problem 3 is avoided by including a unique Message Sequence Number (MSN) in each node sensor data packet. This unique MSN (unique per node, not globally) will be used to eliminate all duplicate messages at the sink. In our particular case, since the data flow in xMesh is not multi-path, the probability of a cluster head receiving a duplicate message simply does not exist. We include the MSN however, to accommodate situations where broadcast modes are used for data transfer.

REFERENCES


