UCLA
Papers

Title
Aggregation in sensor networks: An energy-accuracy trade-off

Permalink
https://escholarship.org/uc/item/8t06k2rw

Authors
A. Boulis
S. Ganeriwal
M. B. Srivastava

Publication Date
2003

Peer reviewed

Aggregation in Sensor Networks: An Energy-Accuracy Trade-off

Athanassios Boulis, Saurabh Ganeriwal, and Mani B. Srivastava

Networked and Embedded Systems Lab, EE Department, University of California at Los Angeles
email: { boulis, saurabh, mbs }@ee.ucla.edu

Abstract – **Wireless ad hoc sensor networks (WASNs) are in need of the study of useful applications that will help the researchers view them as distributed physically coupled systems, a collective that estimates the physical environment, and not just energy-limited ad hoc networks. We develop this perspective using a large and interesting class of WASN applications called *aggregation applications*. In particular, we consider the challenging periodic aggregation problem where the WASN provides the user with periodic estimates of the environment, as opposed to simpler and previously studied snapshot aggregation problems. In periodic aggregation our approach allows the spatial-temporal correlation among values sensed at the various nodes to be exploited towards energy-efficient estimation of the aggregated value of interest. Our approach also creates a system level energy vs. accuracy knob whereby the more the estimation error that the user can tolerate, the less is the energy consumed. We present a distributed estimation algorithm that can be applied to explore the energy-accuracy subspace for a sub-class of periodic aggregation problems, and present extensive simulation results that validate our approach. The resulting algorithm, apart from being more flexible in the energy-accuracy subspace and more robust, can also bring considerable energy savings for a typical accuracy requirement (five-fold decrease in energy consumption for 5% estimation error) compared to repeated snapshot aggregations.**

Keywords: sensor networks, aggregation applications, distributed estimation, energy vs. accuracy trade-off.

I. INTRODUCTION

The technological advances in embedded computers, sensors, and radios have led to the emergence of wireless ad-hoc sensor networks (WASNs) as a new class of system with uses in diverse and useful applications. Indeed, the early papers in the area [6][7][13][15] talk about the vision of cheap self-organizing ad-hoc networks that are able to perform a higher level sensing task through the collaboration of a large number of cheaper and resource constrained wireless sensor nodes. Leveraging numerous sensing devices placed close to the actual physical phenomena, the information that such networks can provide is more accurate and richer than the information provided by a system of few, expensive, state-of-the-art sensing devices. Since WASNs operate largely unattended, often in environments where the access cost of deploying or maintaining nodes is high, a key problem in designing WASNs is how to prolong their useful lifetime by conserving energy. Consequently, a large fraction of research in WASNs has been dedicated to aspects of the energy-efficiency problem.

The original vision and promise of WASNs was that multiple nodes collectively perform the sensing task requested by the users and communicate the results to the users. However, most of the research so far has simply viewed WASNs as just another kind of wireless ad hoc networks, albeit one composed of nodes that are more energy-constrained and whose data sources are sensors. So, for example, much work has focused on issues such as energy-efficient MAC and ad hoc routing protocols to realize the needed point-to-point and point-to-multipoint communication patterns in WASNs. But, little has been done to develop an understanding of a WASN as a *collective* or an *aggregate* where sensor nodes collaborate to jointly estimate the desired answer about the sensed environment. In part this is because not many actual applications useful to the end-user have been studied. The only notable exception is the target-tracking problem, which has drawn attention from several research groups. Otherwise, the applications that have been examined are usually "toy" scenarios used to showcase the abilities of protocols and programming frameworks (e.g., [10]), or very specific applications examined for the sake of some energy-saving technique (e.g., [11]).

In this research we have made a first attempt at exploring and understanding the performance of a WASN as a collective that performs a sensing task. We examine a general class of WASN applications that we call *aggregation applications* where the desired answer depends on the sensed value at multiple nodes. In particular, we explore the energy vs. accuracy subspace, i.e. how much energy savings can one get by relaxing some accuracy requirements and vice versa. We propose an algorithm that exploits this trade-off and jointly considers networking and signal processing issues to create a distributed estimation mechanism.

A. Aggregation Applications

Many of the examples and simple applications presented in WASN research are based around some kind of aggregation function. The most popular and simple examples of aggregation functions are "maximum" and "average". That is, a user may be interested in knowing the max (or average) of a value in the WASN or in some restricted area of the WASN. If this function needs to be performed once, we refer to it as "snapshot aggregation". If the user needs an update in periodic intervals we refer to it as "periodic aggregation".

The snapshot aggregation problem is trivial for a single static user. The user sends a request to flood the sensor network (or the area of interest). Upon reception of a request

message a node sets the sender of the message as its parent, leading towards the user. This way an aggregation tree is formed with the user node at its root. Data are flowing along the aggregation tree towards the user while being aggregated at intermediate nodes. For instance, in the max function a node receiving multiple values (i.e., its own local reading and values sent by other nodes) finds their maximum and sends it to its parent. For more details on snapshot aggregation the reader can refer to [2][3].

More generally, in aggregation applications, the user seeks a condensed view of the physical environment the WASN is monitoring, or a condensed view of the network's state. To achieve this, the values from all the nodes (i.e., sensor readings or node state values) are aggregated to a size-bounded vector describing this condensed view. Furthermore, several important properties hold for the aggregation process: (i) multiple local values can be combined to an aggregated description with a single pass, (ii) multiple aggregated descriptions and multiple local values can be combined to an aggregated description with a single pass. These properties permit the aggregation process to be done easily within the network, without the need for multiple passes of the data. A counter-example is the calculation of median, as it requires two passes of the data. The bound on the aggregated description (i.e., vector) is $O(1)$. In order to include more specific cases of applications (like some referenced in related work) the bound can be relaxed to $O(N)$, where N is the number of nodes in the network.

Some examples more advanced than "max" and "average" include: (i) approximate contours of nodes' residual energy (similar to the specific case studied in [16]), (ii) approximate the boundary line between sensing and no-sensing (e.g. of light) with a straight line or a parabola (similar to the specific case studied in [11]). Whatever the aggregation function is, the basic structure of in-network processing in snapshot aggregation remains the same, combining values and descriptions at intermediate nodes until the final aggregated description reaches the user. This kind of in-network processing is similar to traditional distributed/parallel computing where precise information is handled and the correct execution of the algorithm often depends on the right number and order of messages exchanged. We call such processing type-I.

If periodic aggregation is needed then one could run snapshot aggregations periodically, executing accurate type-I algorithms periodically. This indeed is the conventional approach.

B. Our contribution: Distributed Estimation Approach

In WASNs though, unlike traditional distributed computing systems, there is a strong coupling to the physical world as we are monitoring some parameters of a physical process. The WASN by its nature can only sample this physical process, which in turn implies that we are only getting an approximation of the parameters sought. Once this point is understood, it is realized that the algorithms in WASNs have an inherent extra dimension: **accuracy**. We can exploit this extra dimension to produce more energy efficient algorithms. The less accuracy is required, the more energy can be saved using proper algorithmic techniques. For example, in our particular case, the spatial-temporal correlation of sensor node's values is leveraged to create estimates of the aggregated descriptions of the environment. The less accurate the estimates need to be, the fewer messages need to be exchanged. We argue that approximation and estimation are an inherent part of WASN algorithms and should be taken into account while designing algorithms for WASN. We call such type of in-network processing type-II. Type-II algorithms are essentially ***distributed estimation*** algorithms.

Essential to our development of such distributed estimation algorithms is the joint consideration, or co-design, of the signal processing algorithms and networking protocols that have thus far been treated separately in WASNs. In this paper, we propose a distributed estimation algorithm that explores the energy/accuracy subspace for the periodic aggregation domain.

We have validated our approach through simulation using the "max" aggregation function as an example. The "min" aggregation function is completely symmetrical. Note that the "max" aggregation function can be on any property of the actual local measurements. For instance, if we are interested in the maximum rate of change among local measurements, the aggregation function will be done on the derivatives of the local measurements. We are currently working to extend our algorithm to the more general k^{th} max case. In its current form, the algorithm cannot be applied to summary aggregation functions [17] like "average", "count" etc.

In section II we examine related work. In section III we introduce some initial approaches to the problem. In section IV we describe our distributed estimation algorithm. Section V presents simulation results. Finally, section VI concludes the paper.

II. RELATED WORK

Aggregation applications have been popular among researchers in WASN. This is mainly because the concept of aggregation is simple enough to be executed at each node with minimal effort. The main problem is how to orchestrate the whole procedure to save energy. Efforts such as [4] and [9] seek to provide a framework for flexible aggregation in sensor networks. They investigate constructs to program the sensor network to perform arbitrary aggregation functions. However, all the prior work has been mainly concerned with snapshot aggregation and has only superficially acknowledge the energy/accuracy trade-off present when continuous periodic aggregation is considered.

For example, in [17], authors have developed an aggregation service for ad hoc networks of TinyOS motes. The aggregates are processed in the network by computing over the data as it flows through the sensors; what we term as

snapshot aggregation. The main contribution of the paper is to provide a simple, declarative interface for carrying out snapshot aggregation using database query languages. More specifically, authors have developed a query processing system based on SQL for extracting information from a network of sensors. The authors show a significant performance improvement as compared to traditional centralized, out-of-network methods, emphasizing the need of carrying out in-network processing in WASNs. They are restricted to type-I processing though. We take a step further to acknowledge the energy-accuracy space and propose type-II algorithms for the aggregation problem, which are more robust and flexible than type-I algorithms, as shown in this paper. Research work that is trying to provide a general framework for programmable sensor networks, such as [10], also considers aggregation to illustrate the capabilities. They too are restricted to snapshot aggregation.

More relevant to our work are efforts that view a very restricted portion of the aggregation realm but nevertheless try to provide algorithms for the periodic aggregation case. In [16] the authors are concerned with the problem of providing a residual energy map of the network. They do so by calculating the "equi-potential" curves of residual energy with some tolerance. Their aggregation function is quite simple, as they do not consider approximation in location (two curves engulf accurately all nodes with residual energy within the tolerance unit). A more advanced aggregate function could try to fit an accurate convex curve (having arbitrary many points) with a fixed-point convex curve. Furthermore when the authors are concerned with the problem of periodically update the residual energy map they resort to the simple scheme of incremental updates. That is, if a node changes its value over the tolerance limit its value is transmitted and aggregated again in some intermediate node. The final change should eventually reach the user. There is no provision to predict changes or try to estimate correlation between values so one can set the threshold less conservatively.

In [8] the authors are concerned with the problem of monitoring the values of every sensor over time. This can be considered as a degenerate version of an aggregation function where a node does not actually aggregate or merge any data. However the basic notion of spatial-temporal correlation between the sensor values exists and a periodic monitoring scheme can try to exploit it, as an aggregation scheme would do. The authors note that this spatial-temporal redundancy is exploited in video compression to save storage size and/or bandwidth. They try to use and modify the basic techniques of MPEG-2 for sensor network monitoring to save communication traffic and thus energy. Essentially a central node is calculating predictions and transmits them to all the nodes. The nodes send their update only if it is different from the prediction. There is a hierarchy of nodes to perform this task more distributed. The scheme proposed in [8] raises questions on whether the prediction transmission is actually more efficient in a large random network. Unfortunately their algorithms are tested in small networks where every node belongs to just one cluster (i.e., every node is one hop away from the central node).

III. INITIAL APPROACHES

Since distributed estimation for aggregation applications is not a well-researched area, it is beneficial to start by describing some of the intuitive approaches to the problem. The different approaches are individually useful under certain circumstances but they cannot be applied universally. For each approach we will identify some of its major weak points. Through this process the reader can get a deeper insight into the problem and appreciate its difficulty in the general case. We will be considering the "maximum" as the aggregation function to make our statements more concrete, without loss of generality.

A. Centralized - simple

The user node gets the values from all nodes at some update rates. As the user learns about the dynamics of the process it computes the optimum update rate for each node and communicates that decision to each node. Problems with this approach: (a) centralized, (b) the processing power at the nodes is not leveraged, and (c) although the nodes locally know their values continuously, this solution only samples the local values at different update rates. In essence the user "visits" the different node locations and samples the values at different update rates (depending on his previous measurements). The WASN is used *only as a network*, connecting the locations to the user, so that the user does not have to physically take the trip. One can do better by leveraging the computation capabilities of the nodes in WASNs.

B. Centralized - leveraging more computational and sensing power

The user node gets the values from all nodes at non-regular intervals. The user computes the global maximum and based on previous values it predicts how the global maximum is going to evolve. This prediction (a function of time) is transmitted to all the nodes. At every sampling interval the node compares this prediction with its own value. If the local value is larger then it should be transmitted towards the user. This way the processing in the nodes is leveraged, as well as the capability of the nodes for continuous sensing. Problems with this approach: (a) still centralized, (b) does not adapt well to fast changing physical processes (because the prediction should be send too often), (c) the computational capabilities at the nodes are used minimally.

C. Distributed - simple heuristic

An aggregation tree is formed with data flowing from children to parents. The data are aggregated in intermediate nodes until they reach the user. Each node receives locally aggregated values from its children at different update rates. Depending on the values received by the children and its own value, a node decides how to assign new update rates based on

its own update rate (i.e., the rate that its parent is notified). So every child gets a number [0...1] and the parent's update rate. If a node's update rate is changed (by its parent), the node notifies its children of the new update rate. This is essentially the distributed case of the first approach. The problem with this approach is that computational and sensing capabilities are still not leveraged.

D. Distributed - leveraging more computational and sensing power

If we want to leverage the ability of the nodes to sense constantly we can enhance the previous scheme with feedback. A node sends the current local aggregated value back to its children. The children decide to send or not their value according to their own readings. There is no need for specified update rates. A node transmits based on the local values constantly sampled, and the partial information on the globally aggregated value received by its parent. Problems with the approach: (a) temporal-spatial correlation is not used in a predictive scheme, (b) poor performance in fast global max decreases.

IV. DISTRIBUTED ESTIMATION ALGORITHM

In this section, we propose a distributed estimation algorithm that can be applied to a large class of aggregation problems. More specifically we classify aggregation functions into two categories: (i) aggregation functions whose result is determined by the values of *a few* nodes (e.g., the max result is based on one node), and (ii) aggregation functions whose result is determined by the values of *all* the nodes (e.g., the average function). We have found that different kinds of distributed estimation algorithms are required in order to deal efficiently with each of the categories. In this paper we only consider the first case, i.e., aggregation function that find or approximate some kind of boundaries (e.g., minima, maxima), and hence the aggregation result is determined by the values of few nodes. This still includes a large number of interesting applications (e.g., approximate a shadow of light using a straight line and tracking it through time). In the rest of the paper, if we need to refer to a specific aggregation function we will use the simple scalar max function. Finally, another point to note is that the proposed algorithm does not assume any knowledge about the underlying physical process.

A. Basic Approach

Instead of transmitting a partially aggregated result, as in the snapshot aggregation case, each node keeps and possibly transmits an ***estimation*** of the global aggregated description. The global aggregated description in its most general form would be a vector quantity. An estimate in its most complete form is a probability density function (pdf) of the value (or vector) that is being estimated. Most of the times though, due to insufficient information, ease in computation or lack of proper estimation theory tools, an estimate has the form: (estimated value, confidence indication), which in estimation theory nominally means: (average of estimated vector,

covariance matrix of estimated vector). For both reasons of compactness and manipulability with estimation theory methods, we chose to represent estimates in the form (A, P_{AA}) with A being the mean of the aggregated vector and P_{AA} being the covariance matrix of vector A. For the max aggregation function, vector A becomes a scalar denoting the mean of the estimated max, and P_{AA} becomes simply the variance of A.

In snapshot aggregation, a node cannot change the rate at which they send predictions to their parents. Whatever the initial user update rate is, a node has to keep it. This is because a node receives little information about the global maximum, as it has no idea what is happening beyond its parent. In our approach a node accepts estimations from all of its neighbors. This way the node gradually gains knowledge of the global situation and can also decide whether its own information is useful to other nodes or not. If the decision is positive (i.e., the estimate could be useful to other nodes) it should transmit the new estimate. Unlike snapshot aggregation where the node transmits its estimate to a designated node (also called its parent), in our scheme the node broadcasts its estimate. Thus every node in the neighborhood shall receive this new estimate.

Note that with this approach there is no need to create and maintain a hierarchical parent-child structure. Every node has information about the globally aggregated value. This is a strong point of our approach since it can operate effortlessly with multiple users, mobile users, failing nodes, and failing connections between the nodes, events that are very common in WASNs. The global information is everywhere not just at the root of an aggregation tree as in the snapshot aggregation case. If the distributed estimation algorithm we are applying is successful then the nodes should not have great variance among their estimates and the estimates should also be close to the actual aggregated value. Indeed we show these desired properties of our algorithm in section V via simulations.

B. Modular Structure

The working of the algorithm can be subdivided into the following major points:

- Every node has an estimated value of the global aggregated value (henceforth called "global estimate" or "estimated global value") in the form: (mean, covariance matrix).

- When a node makes a new local measurement, it updates its global estimate. It then decides if it is worthwhile to transmit this new global estimate to its neighbors.

- When a node gets a global estimate from a neighboring node, it combines this estimate with its own global estimate and forms a new global estimate. It again takes a decision whether it's worthwhile to transmit this updated estimate or not.
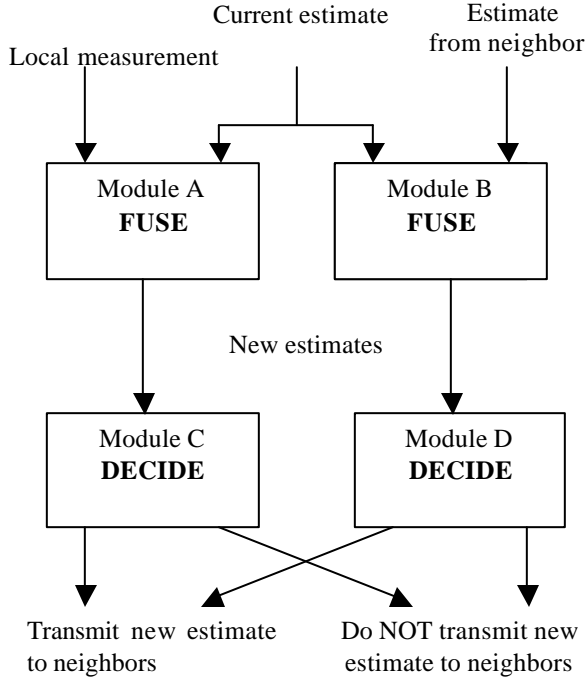
Figure 1. Modules of the generic algorithm.

Figure 1 represents the modular breakup of this algorithm. In the next subsections, we discuss the modules *A, B, C* and *D* in detail.

C. Fusing two global estimates

This subsection corresponds to module *B* shown in Figure 1. We need a data fusion algorithm to combine two global estimates into a new estimate. A popular and effective algorithm to fuse two estimates is the Kalman Filter. Unfortunately, the Kalman Filter requires that the two estimates are independent or that we know their dependency by means of their covariance matrix P_{AB}. In the distributed environment we are operating, we cannot keep track on "which node received what information by which node". Thus we do not know how much redundant (dependent) information exists in an estimate a node receives. The most suitable algorithm in our case is covariance intersection [5]. Covariance Intersection (CI) can be viewed as a generalization of the Kalman Filtering. The principal advantage of CI is that it permits filtering and data fusion to be performed on probabilistically defined estimates without the need to know the degree of correlation among those estimates. Thus, CI makes no assumptions on the dependency of the two pieces of information, when it fuses them.

Given two estimates *(A, P_{AA})* and *(B, P_{BB})*, the covariance intersection combined estimate *(C, P_{CC})* is determined by the following equations:

$$P_{CC} = (wP_{AA}^{-1} + (1-w)P_{BB}^{-1})^{-1}. \qquad (1)$$

$$C = P_{CC}(wP_{AA}^{-1}A + (1-w)P_{BB}^{-1}B). \qquad (2)$$

Here P_{AA}, P_{BB}, and P_{CC} represents the covariance matrix associated with estimates *A, B* and *C* respectively. The main computational problem with CI is to compute w, $0 \le w \le 1$, such that the trace of the P_{CC} determinant is minimized.

For the case of the max aggregation function, covariance matrixes are simple scalars. It can be observed from equations 1 and 2, that for such a case w can be either *1* or *0*. Subsequently, P_{CC} is equal to the minimum of P_{AA} and P_{BB}. Accordingly, *C* becomes equal to either *A* or *B* depending on the value of P_{CC}. Even if the estimates are vectors of a small number of scalars (e.g., find the 2nd or the 3rd max/min) there are computationally efficient algorithms [5] to determine w.

D. Fusing a local measurement with a global estimate

This subsection corresponds to module *A* shown in Figure 1. Until now nothing was assumed about the specific aggregation function (apart from the fact that it belongs to one of the major categories of aggregation functions as discussed in the beginning of section IV). The functionality of this module depends on the aggregation function performed (e.g., this module should be different for finding the maximum or the minimum aggregated value). Since the quantities we want to fuse are so different, there is no general fusion theory to combine them (like fusing two global estimates), thus we resort to heuristics. The heuristic fusion rule has to answer the question: "How does a local measurement affect a global estimate?" We describe the module's functionality for the problem of finding the global maximum. The methods used in our example can be used as guidelines in more complex aggregation functions.

We begin by modeling the local measurement, as a mean and a variance, like the global estimate. The mean is the actual measured value and the variance represents the variance of the measurement noise. Since the quantities we want to fuse are so different, there is no general fusion theory to combine them (like fusing two global estimates). Thus, we resort to breaking up the problem in two different cases and use heuristics to fuse the data. The proposed heuristics need the actual distribution of the global estimate and the local measurement. Since only mean and variance information is available the best choice is the gaussian distribution (gaussian distributions are the maximum entropy distributions for given mean and variance). Let us call the local measurement distribution *l(x)* and the global estimate distribution *g(x)*. The two different cases of the fusion rule are given below:
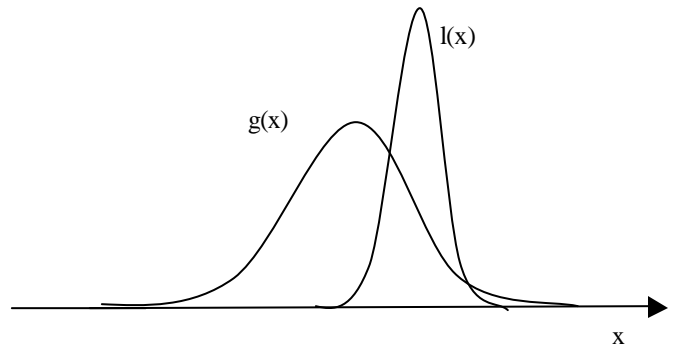

Figure 2. Distributions of global estimate g(x) and local measurement l(x).

D.1 Mean of local measurement is larger than the mean of the current global estimate

To analyze this case, consider the simplistic case where the current estimate and local measurements are simple scalars and not distributions. Clearly, the new estimate should be the local measurement. We need to extend this argument for the case when estimate and local measurements are modeled as distributions. Figure 2 represents this case.

In this case, the distribution corresponding to the new estimate cannot be simply $l(x)$. This is because although mean of $l(x)$ is greater than that of $g(x)$, there is a non-zero probability that the value of the new measurement is smaller than the old estimate. Hence, the resultant distribution should be calculated taking into account both the local measurement as well as the old estimate. Recall that the value of the distribution at any x' $(g(x')$ or $l(x'))$ represents the confidence (or certainty) which can be associated with this x' in the estimate $(g(x)$ or $l(x))$. We use a simple heuristic that the confidence associated with any x' in the new estimate should be simply the sum of the confidence associated with this x' in the measurement $(l(x'))$ and the earlier estimate $(g(x'))$. Also, as the objective is to calculate the maximum aggregated value we assign a higher weight to a potentially higher value (in this case the local measurement). Thus the resultant distribution, corresponding to the new estimate, is calculated by taking a simple weighted sum of the two distributions as follows:

$$g_{new}(x) = \{w_1(x) * g(x)\} + l(x) \forall 0 \le w_1(x) \le 1. \quad (3)$$

Finally, we calculate the mean and variance of $g_{new}(x)$ so that we can represent the new global estimate in the form: (mean, covariance matrix).

D.2 Mean of local measurement is smaller than the mean of current estimate

We again combine the two distributions as in the previous section, assigning a higher weight to the distribution with the larger mean (in this case the old estimate).

$$g_{new}(x) = \{w_2(x) * l(x)\} + g(x) \forall 0 \le w_2(x) \le 1. \quad (4)$$

However, the situation is not so trivial in this case. If the value decreases at every node, our estimate should track this decrease rapidly. Although equation (4) can handle minor fluctuations it performs poorly in sharp falls of the global maximum. To account for this issue we use the following rule: If the previous local measurement was close to the global estimate (closeness determined by a fixed fraction value) we assign the higher weight to the local measurement, reasoning that the specific local measurement is still the global aggregated value. We achieve this by calculating the new estimate using equation (3) instead of (4).

Note that even if a new (smaller) estimate is wrongly generated following this rule, the particular node will be promptly corrected by one of its neighbors, which will transmit the "correct" (higher) estimate.
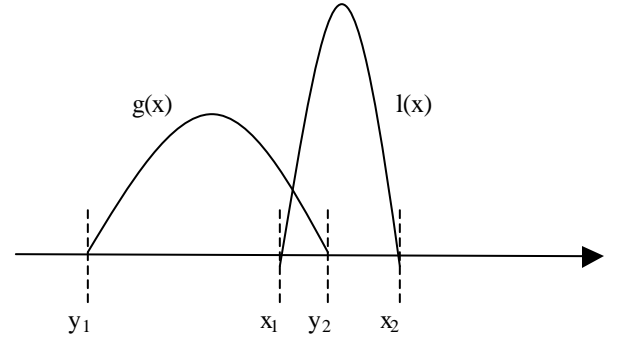

Figure 3. Bounded distributions

D.3 Calculating $w_1(x)$ and $w_2(x)$

In this subsection, we present a simple heuristic way in which $w_1(x)$ and $w_2(x)$ can be calculated. We make an approximation that all Gaussian distributions are bounded within the interval $[m-3s, m+3s]$, where m and s^2 represent the mean and variance respectively. The approximation is derived from the well-known property of gaussian distributions stating that a gaussian distribution's values lie in the interval $[m-3s, m+3s]$ with a probability of 99.7%.

Let us first consider the case, when the mean of local measurement is larger than the mean of the current estimate. Figure 2 is redrawn in Figure 3, but now the distributions are bounded within finite limits. This means that $l(x)$ and $g(x)$ can take nonzero values only in interval $[x_1, x_2]$ and $[y_1, y_2]$ respectively.

The global aggregated value should at least be equal to the current measurement. Hence we should assign zero probability to the points on the left of x_1. Thus $w_1(x)$ can be defined as:

$$w_1(x) = \begin{cases} 0 \forall x \le m_1 - 3s_1 \\ 1 \forall x > m_1 - 3s_1 \end{cases} \quad (5)$$

Note that x_1 is equal to $m_1 - 3s_1$, where m_1 and s^2_1 represent the mean and variance of $l(x)$ respectively. Figure 4 represents the case, when the mean of local measurement is smaller than the mean of the current estimate. We assign a zero probability to points on the left of y_1. The assumption is that the value won't fall too drastically within a sampling period. Moreover as explained earlier, we should assign zero probability to the points on the left of x_1. Thus $w_2(x)$ can be defined as:

$$w_2(x) = \begin{cases} 0 \forall x \le \max(m_1 - 3s_1, m_2 - 3s_2) \\ 1 \forall x > \max(m_1 - 3s_1, m_2 - 3s_2) \end{cases} \quad (6)$$

Note that y_1 is equal to $m_2 - 3s_2$, where m_2 and s^2_2 represent the mean and variance of $g(x)$ respectively.

We normalize the resultant function appropriately so that it represents a valid probability distribution. Although we are combining two bounded gaussian distributions, we make an assumption that the obtained distribution is still represented by a gaussian distribution. This is done in order to maintain the consistency of the estimates. The mean and variance of the
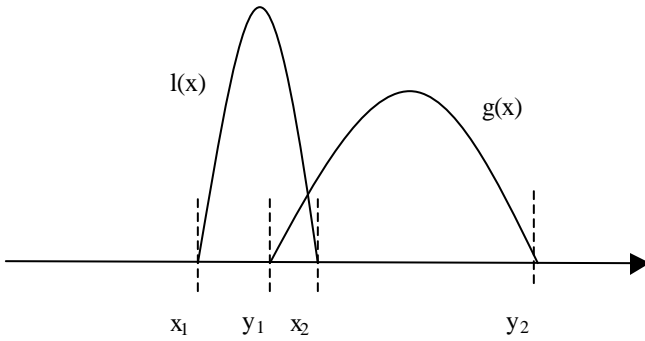
Figure 4. Bounded distributions

new gaussian distribution represents the new estimate and the confidence (or certainty) associated with this new estimate respectively.

E. Making the decision

This subsection corresponds to modules *C* and *D* in Figure 1. These modules directly affect the tradeoffs one can achieve in the energy-accuracy subspace. When a node calculates a new estimate, it should not necessarily transmit it. In fact it should transmit only if this new estimate can bring a *significant change* in one of its neighbor estimate.

This raises a question, how does a node know whether its new estimate is going to affect the estimates of its neighbors. For this purpose every node maintains a table where it stores the most recent estimates received from its neighbors. Let us call this table the *estimate_table*. A node, after calculating its new estimate (by using modules *A* or *B*), combines this estimate with every estimate in the *estimate_table,* by using the same algorithm as in module B. The node then calculates the difference between the new generated estimate for every neighbor and the old estimate. If this difference is beyond a preset threshold for any of the neighboring nodes, the node broadcasts its new estimate. The threshold is expressed in terms of a fraction, i.e., a number in [0…1].

Note that the threshold should be on both the mean and the variance. So even if the mean remains the same, but the variance has changed significantly, the node should still broadcast its value. We represent the thresholds on mean and variance as $mean_T$ and $variance_T$ respectively. In the current implementation, we keep both $mean_T$ and $variance_T$ to the same value.

Let us observe closely what does the threshold physically mean. Suppose we set $mean_T$ to be 0.05. This means if the node observes a change greater than *5%*, it shall broadcast its estimate. This also means that all changes less than *5%* are left unnoticed. Thus, the algorithm can converge to a global aggregated value that is off by at most 5% from the actual value. Thus, the threshold directly relates to the accuracy of the global estimate in each node, and consequently the accuracy of the estimate that the user receives. The threshold also relates directly to the energy expenditure of the sensor network, as it decides the amount of traffic being generated in the network. Thus, the higher the threshold is, the lower the

accuracy is, and the higher the energy savings are. In fact, the threshold gives the user a control knob in the energy-accuracy subspace. This parameter is set depending on the user requirements.

We should also try to control the overhead of passing on redundant information. This will be clear from the scenario shown in Figure 5. Suppose that nodes *I, J* and *K* are in each other's neighborhood. Suppose node I measures a local value, which changes its estimate. After combining this estimate with the values in its *estimate_table*, node *I* decides to broadcast its new estimate. Both node *J* & *K* get this new value and will update their respective estimates accordingly. Now node *K* (or *J*) is unaware that the estimate of *J* (or *K*) has also changed. So when node *K* (or *J*) runs its decision making step, it will find out that it should broadcast this new estimate as it can change the estimate of *J* (or *K*) considerably. Thus the same estimate is propagated again.

This overhead can easily be avoided by simply maintaining information about two-hop neighbors at every node. When a node gets a reading from another node (say *X*), it only checks for those nodes (in the *estimate_table*) that are in its own neighborhood but not in the neighborhood of *X*. Thus, in the above example when *K* (or *J*) gets the reading from *I*, it does not broadcast as it realizes that *J* (or *K*) is also in the neighborhood of *I*. The two-hop neighborhood information can easily be acquired with a protocol like the one in [1], at the network boot-up phase. However when nodes die this information might get outdated. If a node does not keep track of such dying nodes, it might just potentially spend some unnecessary energy in making sure that information reaches dead nodes (which it believes are still alive). Note that irrespective of the strategy that a node follows, the accuracy of the algorithm remains unaffected.

F. Insights

In this subsection, we give some insight on the current model, as well as motivate more complex models.

F.1 Robustness of the algorithm

In this subsection we will discuss the robustness of our proposed scheme and compare it with that of snapshot aggregation. First let us summarize some of the key properties for our algorithm:

- The algorithm is completely distributed and localized (nodes exchange information only with immediate one-hop neighbors).
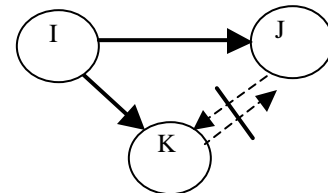

Figure 5. Eliminating redundant transmissions

- There is no need for time synchronization between the nodes. Moreover, there is no requirement on the number or order of messages that a node should receive. If a new estimate is received the current estimate is updated.

- All the nodes perform the same functions, and compute the same results. Every node has an estimate of the global aggregated value. There are no special nodes.

Contrast these properties to the properties of snapshot aggregation:

- There is a hierarchical parent-child structure.

- The parent should receive an exact number of messages, equal to the number of its children.

- The final result is only available at the user node.

As a result our algorithm performs exactly the same with mobile users and multiple users (the global estimate is available everywhere). Failing nodes, failing links, non-reliable transport protocols and MACs, deteriorate accuracy slowly, as only the information of the failing nodes/links is lost and this is usually redundant information. This robust operation of the algorithm is a great advantage in the dynamic environment of WASNs where node failures and mobile users are the norm rather than the exception.

Snapshot aggregation on the other hand is very sensitive to the stability of the hierarchical structure. If a parent node dies, all its descendents are cut off from the functionality of the algorithm and it results in generating poor estimates of the global aggregated value. Further more, if a user moves, a new aggregation tree needs to be formed in order to receive the aggregated value. This is translated in more energy consumption.

F.2 Choosing the threshold

The energy-efficiency and accuracy of the algorithm heavily depend on the chosen threshold. As the simulation results reveal in section V there is a very concrete dependency of threshold, energy spent, and accuracy achieved. If a user knows some of the macro characteristics of the physical process being monitored, then based on the graphs provided in section V he can knowledgably chose a threshold that fit his needs. What the user needs to know is how dynamic the process is. For example, how fast do the values fluctuate and in what dynamic range? Intuitively a more dynamic physical process will put a greater strain in our distributed estimation process, requiring more energy for the same accuracy. Indeed this intuition is validated from the simulation results.

The more interesting problem rises though when the user has no information on the physical process being monitored. In such a case, setting a threshold does not provide immediate information on the actual accuracy achieved. The only known fact is that the accuracy is less or equal than the chosen threshold. The only other parameter than can provide the user with some indirect information about the accuracy is the

variance associated with the mean estimate. Recall that variance represents the degree of certainty that should be associated with the estimate. Hence, if a user gets an estimate with a large variance, he should try to decrease the threshold and vice versa. Currently we are in the process of developing an analytical model to provide guidelines to the user, so that he can adjust the threshold on-line. Furthermore, the energy spent at the user node can be used as an estimate of the average energy spent in nodes (recall that our algorithm is fairly homogenous in energy consumption). So even with no physical process information the user can acquire an estimate for the accuracy achieved, as well as for the energy spent, on the fly. In some rare cases with very dynamic processes and little spatial-temporal correlation between the nodes' values the user could be advised to resort to the periodic execution of snapshot aggregations.

F.3 Data Staleness

The estimates generated by every node are based on information, the node has acquired until that time. These estimates are going to be valid at that time but as time passes the data will become stale. Thus, we need to differentiate between the freshness of the estimates. Ideally, we would like to give more weight to an estimate that has been recently generated. A very simple way of doing this is to timestamp the prediction. Any old predictions (before a certain time) will be neglected. However, this way we do not achieve a graceful degradation in estimate importance. A more proper way of achieving the gradual staleness of the estimates is to increase their variance over time. In our current implementation, we increase the variance associated with the estimate at a constant rate.

This approach motivated us to look for more general models, where mean and variance are represented as time functions rather than simple scalars. This is a subject of on-going work.

V. SIMULATION RESULTS

In this section we present the simulation platform, a general model to describe diverse physical processes, and finally measurements on several aspects of the energy and accuracy quantities.

A. The simulation platform

The algorithms described in the previous section were implemented in our simulation platform [18], based on PARSEC (PARallel Simulation Environment for Complex systems) [12]. In our simulation platform a sensor network is modeled as: (i) a collection of sensor nodes, (ii) a channel, and (iii) a supervising entity to create the nodes, trigger the sensing devices, and keep statistics of several quantities. Each sensor node consists of two entities. 1) A processing entity, where all node-specific algorithms run, and 2) a radio entity in order to interact with other sensor nodes. The radio entity implements different MAC protocols and measures the energy spent due to transmissions and receptions. Note that energy is spent on

receptions even if there is a collision of packets. For our measurements a TDMA MAC was used. To transform transmission and reception times to energy we used the data from the hardware specifications provided in [14].

B. Modeling a physical process

In order to acquire concrete measurements, the underline physical process needs to be defined. Since we do not wish to bind our algorithm to specific physical process scenarios we have built a general model to define a large class of physical processes. We adopt a diffusion model of multiple, additive, mobile, and variable sources. That is, if there is a source at some point, its value is diffused to the environment according to some power law of distance. For a single point in the WASN coverage area, the influences from multiple sources are added. The sources can move and change their value in arbitrary ways. An example of a physical process described by our model could be a terrain with a fire moving through it. The measured heat follows our model. More specifically our model states:

$$V(p,t) = \sum_{i:allsources} (k \cdot dist(i) + 1)^{-a} \cdot V(p(i),t) + noise \quad (7)$$

Here $V(p, t)$ is the value of the sensed quantity at point p at time t, $dist(i)$ is the distance between point p and source i at time t, $p(i)$ is the position of source i at time t, and k, a are parameters. The noise is gaussian with zero mean and unit variance and it models the measurement noise. For our current experiments we used one source with a specific waveform (shown in Figure 6), with k=0.25 a=1. We altered the period of the waveform and also made the source mobile in order to test the response of our algorithm in increasingly dynamic processes.

C. Measurements

All results are derived using networks of *50* nodes distributed in a uniformly random fashion over a sensor terrain of size *45m* x *45m.* Each node has a transmission range of *15m* and a sampling period of 0.5 seconds. Simulation is run for a time interval of 200 seconds. All simulation results are averaged over 500 individual cases.

Figure 6 shows how the actual maximum value varies over time and the way it is tracked by our algorithm, for two different thresholds. The waveform of the maximum value also reveals the shape of the source waveform used in all our measurements. As stated earlier the shape of the source's waveform remains the same and only its period changes in different simulation scenarios.

Figure 7 shows the cumulative energy consumption of all the nodes in the sensor network versus different threshold values. The figure shows energy consumption using repeated snapshot aggregation and our algorithm in three different scenarios. As can be seen from Figure 7, the energy consumption in our algorithm varies from 5% to 67% of the energy consumed in repeated snapshot aggregation according to the threshold and the physical process examined. As

expected with increasing thresholds, energy consumption decreases. Reducing the period of the waveform of the source or making the source mobile results in a more dynamic setting. The energy consumption naturally increases but in a graceful manner. The figure clearly highlights the stiffness associated with snapshot aggregation. The energy consumption for snapshot aggregation is the same for all the scenarios and for all threshold values.

Figure 8 plots the estimation error, versus different threshold values. The estimation error basically gives us the accuracy of the algorithm. In the case of repeated snapshot aggregation, the user node gets the actual global aggregated value and hence there is no discrepancy between the estimated value and the actual aggregated value (i.e. the error is 0). Again one can see that the snapshot aggregation case is rigid, not able to accommodate different threshold values. In our algorithm, as the threshold increases the accuracy decreases (estimation error increases).

As expected, the performance is worse in a more dynamic setting. A key thing to notice from Figure 8 is that the estimation error is much smaller than the threshold used. Theoretically we know that the threshold acts as an *upper bound* to the estimation error. In practice we discover that the error is considerably smaller. This is a very positive observation as one can get significant energy savings for typical accuracy levels. For instance, if 95% accuracy is needed (i.e., 5% error) for the processes examined above, then we can set the threshold to 0.125 and consume only 20% of the energy that the repeated snapshot aggregation approach consumes.

C.1 Spatial variation

In this subsection, we examine the spatial variation of energy consumption and accuracy in the sensor network. The simulations are carried out for a fixed threshold of 0.1. The source is static and the period of the source is 200s. The source is located at (25,25) i.e., approximately at the center of the sensor terrain. We average out the results for 100 simulation runs.

Figure 9 plots the spatial variation of energy consumption in the sensor terrain. The nodes located near the center of the terrain, closer to the source, are the nodes that are going to measure a higher reading as compared to the boundary nodes. Thus these nodes generate new estimates of the global aggregated value and transmit them with greater frequency (since they are tracking the global max), consuming larger amounts of energy. On the other hand, boundary nodes that are measuring lower values do not need to transmit their estimates since they notice that they cannot influence the estimates of other nodes. Although the plot for repeated snapshot aggregation is not shown, it can be easily inferred that every node consumes the same amount of energy irrespective of its location. Note that even for nodes located at the center of the sensor terrain, the absolute value of energy consumption is much larger for repeated snapshot aggregation as compared to our algorithm.
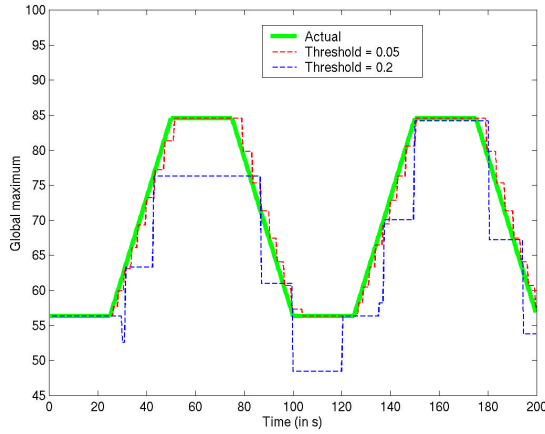
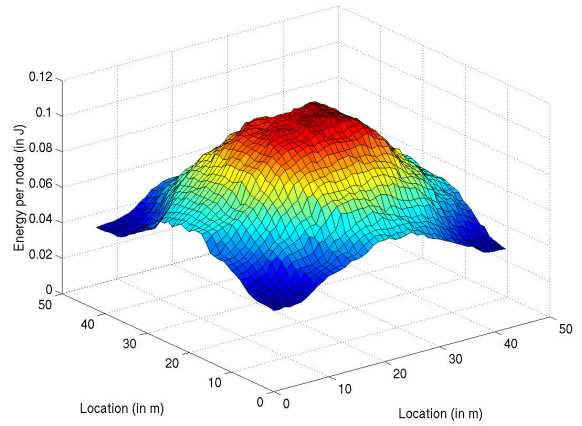Figure 6. Global maximum tracking in time.

Figure 7. Energy consumed vs threshold used.

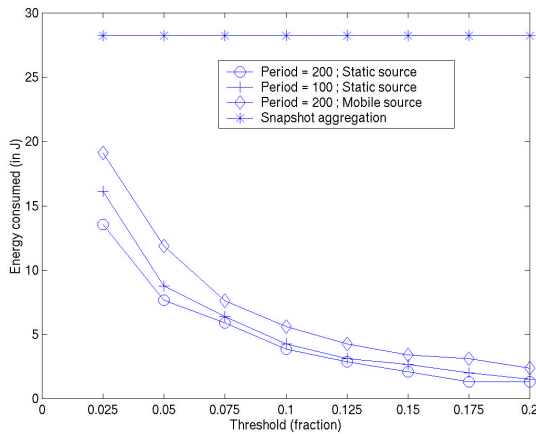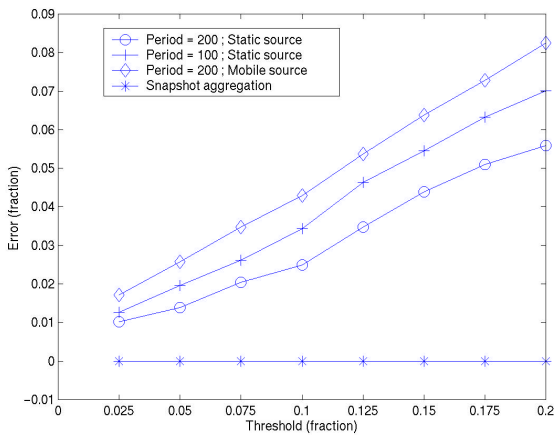
Figure 8. Estimation Error vs threshold used.

Figure 10 plots the spatial variation of estimation error in the sensor terrain. As can be observed from the figure, almost every node in the sensor terrain achieves the same accuracy. This is because of the basic operation of our algorithm according to which every node in the sensor terrain should be

Figure 9. Spatial variation of energy consumption.

Figure 10. Spatial variation of estimation error.

having the same global picture as any other node in the network. Only the node with the actual maximum has an error very close to zero. In fact in every simulation run at least 95% of the nodes had a similar estimate as the user node. This is clearly in contrast to snapshot aggregation, in which only few of the nodes (the nodes in path of the aggregation tree from the node measuring the global maximum and the user node), have the same accuracy (zero) as the user node.

Summarizing, the simulation results show how one can achieve a desired energy consumption and accuracy by choosing the right threshold. Furthermore, the energy consumption is not homogeneous in the network, at least in scenarios where the maximum remains in the same location. The nodes, which possibly have the global maximum, end up spending more energy. However, every node in the network has the same global picture and hence has the same accuracy level. In a more dynamic setting (for example where the source is mobile), we expect both energy consumption and accuracy to be homogeneous throughout the network.

VI. CONCLUSIONS

In this paper we study a large class of applications in WASNs, namely aggregation applications. We propose a distributed estimation algorithm that can be applied to a subclass of periodic aggregation problems. Our algorithm exploits the energy-accuracy trade-off in WASNs to provide the user with a larger solution space than the conventional approach of periodically running snapshot aggregations. We validate our algorithm through simulation, achieving promising results.

VII. ACKNOWLEDGMENTS

VIII. REFERENCES

[1]. S. Borbash, M. McGlynn, "Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks", Proceedings of MobiHoc 2001, Long Beach, USA, 2001.

[2]. A. Boulis and M. B. Srivastava, "Aggregation applications in resource-constrained distributed systems" TM- UCLA-NESL-2001-11-002, http://nesl.ee.ucla.edu/TM/

[3]. A. Boulis, "Illustrating Distributed Algorithms for Sensor Networks", http://www.ee.ucla.edu/~boulis/phd/Illustrations.html

[4]. N. H. Cohen, A. Purakayastha, J. Tuter, L. Wong, D. Yeh, "Challenges in Flexible Aggegation of Pervasive Data" IBM Technical Report.

[5]. Covariance Intersection Working Group (CIWG), "A Culminating Advance in the Theory and Practice of Data Fusion, Filtering, and Decentralized Estimation", http://www.ait.nrl.navy.mil/people/uhlmann/CovInt.html, 1997.

[6]. D.Estrin, R.Govindan, J.Heidemann (Editors), "Embedding the Internet", Communications of the ACM. Vol. 43, no 5, pp. 38-41, May 2000.

[7]. D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", ACM Mobicom Conference, Seattle, WA, August 1999.

[8]. S. Goel, T. Imielinski, "Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG", ACM Computer Communications Review, vol. 31, no. 5, October 2001.

[9]. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govidan, D. Estrin, D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming", Proceedings of Symposium od Operating Systems Principles, October 2001

[10]. C. Jaikaeo, C. Srisathapornphat, and C. Shen, "Querying and Tasking of Sensor Networks", SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V), Orlando, Florida, April 26-27, 2000.

[11]. J. Liu, P. Cheung, L. Guibas, F. Zhao, "A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks", Palo Alto Research Center Technical Report P2002-10077, March 2002.

[12]. PARSEC: PARallel Simulation Environment for Complex systems, http://pcl.cs.ucla.edu.

[13]. G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors", Communications of the ACM. Vol. 43, no 5. May 2000.

[14]. A. Savvides, C. C. Han, M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors", Mobicom 2001, Rome, Italy, pp.166-179, July 2001.

[15]. D. Tennenhouse, "Proactive Computing", Communications of the ACM. Vol. 43, no 5, pp.43-50, May 2000.

[16]. Y. J. Zhao, R. Govidan, D. Estrin, "Residual Energy Scan for Monitoring Sensor Networks", Proceedings of WCNC 2002.

[17]. Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "TAG: A Tiny AGgregation Service for Ad-hoc Sensor Networks", OSDI Conference, 2002.

[18]. NESLSim, http://www.ee.ucla.edu/~saurabh/NESLsim/