Localized Algorithms In Wireless Ad-Hoc Networks: Location Discovery And Sensor Exposure

Seapahn Meguerdichian[1], Sasa Slijepcevic[1], Vahag Karayan[2], Miodrag Potkonjak[1]
[1] Computer Science Department, University of California Los Angeles
[2] Electrical Engineering Department, University of California Los Angeles
{seapahn, sascha, miodrag}@cs.ucla.edu
vahag@ee.ucla.edu

Abstract

The development of practical, localized algorithms is probably the most needed and most challenging task in wireless ad-hoc sensor networks (WASNs). Localized algorithms are a special type of distributed algorithms where only a subset of nodes in the WASN participate in sensing, communication, and computation. We have developed a generic localized algorithm for solving optimization problems in wireless ad-hoc networks that has five components: *(i)* data acquisition mechanism, *(ii)* optimization mechanism, *(iii)* search expansion rules, *(iv)* bounding conditions, and *(v)* termination rules. The main idea is to request and process data only locally and only from nodes who are likely to contribute to rapid formation of the final solution.

The approach enables two types of optimization: The first, guarantees the fraction of nodes that are contacted while optimizing for solution quality. The second, provides guarantees on solution quality while minimizing the number of nodes that are contacted and/or amount of communication. This localized optimization approach is applied to two fundamental problems in sensor networks: *location discovery* and *exposure*-based coverage. We demonstrate its effectiveness on a number of examples.

I. INTRODUCTION

A. Motivation

Wireless ad-hoc sensor networks (WASNs) have emerged as a premier research and development direction in the last several years. This strong activity impetus is due to strong and convincing reasons provided by both economic and scientific potential. From the economic viewpoint WASNs provide the potential for both superior business models as well as a jump in the quality of personal life. For example, it is easy to envision scenarios where work and living spaces are instrumented with sensors in such a way that one can customize, optimize, and personalize how the environment is used. From the scientific point of view, sensors represent the last missing link between the Internet and the physical world. In addition, sensor networks can greatly enhance social science though improved and significantly more accurate monitoring of individuals and groups.

There is a wide consensus that probably the most needed and most challenging component of wireless ad-hoc sensor networks is the development of practical localized algorithms. Localized algorithms are necessary for a number of reasons: First, the relative communication delay in sensor networks is significantly larger than in traditional computational systems. It is interesting to note that in modern deep submicron (DSM) designs, delay on a single system on chip will be up to 20 clock cycles. However, even the fastest communication protocols in WASNs will have delays in millions of cycles. Furthermore, communication generally dominates both sensing and computation in terms of energy (currently, image/video sensors are exceptions). Again, it is interesting to draw parallels with DSM designs. In DSM, communication will also dominate power consumption, maybe eventually by as high as a 10:1 ratio with respect to computations. In WASNs, technology trends are much more difficult to predict, but at least in current and pending technologies, this ratio is much higher, often estimated at 1000:1. In addition, due to their wide deployment and relatively high cost of servicing, we can expect that fault tolerance will be essentially mandatory for large scale WASNs. In any case, it is reasonable to expect that at least some nodes will exhaust their energy supply. Thus, expecting that all nodes will always be able to communicate and compute is not realistic. Even if latency (real-time constraints), power, and fault tolerance were not an issue, very often security and privacy issues will mandate that only a subset of nodes is participating in a task. Sensor nodes are often deployed outside strictly controlled environments, they communicate using wireless media, and hence will be highly susceptible to security attacks. Finally, as WASNs evolve into the Internet-like scale and organization and span the whole Earth and beyond, eventually, the only realistic possibility for all tasks will be execution in highly localized scenarios.

Distributed algorithms, and even distributed sensor networks algorithms, have been addressed in the past, often in great detail. However, the advent of the new generations of wireless ad-hoc sensor networks not only attract new attention, but also require completely new approaches. It is interesting and important to analyze why the already available distributed algorithm techniques are not applicable to wireless ad-hoc sensor networks. There are at least five major reasons: *(i)* WASNs are intrinsically related to the physical, geometric world and therefore have very special properties. *(ii)* Relative communication costs are much higher than they were assumed in all previous distributed computing research. *(iii)* Accuracy of physical measurements is intrinsically limited and therefore, there is little advantage on insisting on completely accurate computation. *(iv)* Power is probably the single most important limiting factor. *(v)* Finally, and maybe most importantly, data acquisition is naturally distributed and unpredictable, implying a strong need for new sensing, computation, and communication models.

We have developed a generic localized algorithm for solving optimization problems in wireless ad-hoc networks. The key idea is to request and process data only at the node which requested a task and some limited number of nodes that are geographically close. More specifically, the goal is to communicate only with nodes that are likely to contribute to rapid formation of the final solution. The approach naturally enables two types of optimization. In the first, one is able to provide that the amount of communication is within user specified bounds in terms of the number of nodes that are contacted, how many times communication is conducted, or which nodes are contacted (e.g. in the case when security and privacy is of the prime importance). The goal under these constraints is to obtain a solution that is as close to optimal as possible. The second case is the dual problem of the first: Quality of solution is within the user specified bounds while communication is minimized.

We apply this approach to two fundamental problems in sensor networks, namely location discovery and exposure. We selected these two problems as driver examples not just because of their importance for proper operation of a WASN, but also because of their very different nature. For example, it is much easier to determine when to terminate the search in location discovery than in exposure. The exposure problem, defined as an integral over a path, is intrinsically global. On the other hand, it is well known that location discovery is very error sensitive and therefore poses unique challenges along that line.

B. Paper Organization

This paper is organized as follows: In the next section, we provide an overview of the related work on localized algorithms, location discovery in wireless networks, and coverage in sensor networks. In Section III we present the generic localized algorithm framework. Section IV contains the technical preliminaries, assumptions, and background information that pertain to the technical discussions in Section V and VI. In Section V we present the Localized Location Discovery and in Section VI the Localized Minimal Exposure Path algorithms. In Section VII we present several experimental results and analysis followed by the conclusion.

II. Related Work

Recently, wireless sensor networks have been attracting a great deal of commercial and research interest. Ad-hoc networks in general, and ad-hoc sensor networks in particular, have no fixed infrastructure and therefore can be deployed and adapted very rapidly. Furthermore, the integration of inexpensive, power efficient and reliable sensors in nodes of traditional wireless ad-hoc networks, with significant computational and communication resources, has opened the door to new research and engineering opportunities. A number of high profile applications for wireless sensor networks have been proposed [Ten00, Est00]. At the same time, wireless sensor networks pose a number of demanding new technical problems, including the need for new DSP algorithms [Pot00], operating systems [Adj99], low power design [Abi00], and integration with biological systems [Abe00].

A. Distributed and Localized Algorithms

Distributed algorithms and architectures has been a commonly used term for a long time in computer science. There are at least six large computer science communities which address specific types of distributed algorithms and architectures: traditional theoretical computer science community [Gal83, Lam78] and in particular computational models and languages communities [Hoa85, Mil89], parallel computation community, distributed algorithms community [Lyn96, Ray88, Tel94, Lam90], distributed artificial intelligence [Dur89, Rum86], operating systems [Ben93, Kis92] and client-server research and development community [Dow98]. Unfortunately none of the already proposed approaches are applicable to wireless ad-hoc networks. For example, parallel computation research is mainly concerned with exploiting concurrency. In addition distributed sensing has been a popular topic for at least two decades [MIT82]. There are also a number of wireless networks related efforts in development of efficient distributed algorithms [Bad93].

In order to address the needs of distributed computing in wireless ad-hoc sensor networks, one has to address how key goals, such as power minimization, low latency, security and privacy, are affected by the algorithms used. Depending on what the primary optimization goals are, different distributed communication and computation models are appropriate. However, some common denominators are almost always present. They include high relative cost of communication to computation, need to take communication and routing protocols into account, and importance of security and privacy.

B. Location Discovery

Knowledge of physical locations of sensor nodes is essential for many applications for which wireless sensor networks can be used. Examples of such applications are target tracking [Int00] and coverage management [Meg01a, Meg01b]. Importance of the location information for effective operation of sensor networks was recognized as an important problem and several solutions have been proposed.

In [Sav01a] and [Beu99] sensor nodes are equipped with Pico Radios with a transmission range of up to 10m. Nodes, whose locations are known in advance, serve as reference points for other nodes. A node approximates its position based on estimated distances to three or more reference points. It is not necessary that the reference points reach every node in the network. Each reference point initializes a process of *cooperative ranging* where a partial topology of the network, with that reference point as the origin, is propagated through the network. Consequently, each node can determine its distance to a reference point from the acquired topology. Typically, with three or more distances to reference points, a node can determine its location by multilateration. One method used to approximate distances between neighboring nodes is using Received Signal Strength Indicator (RSSI) measurements. RSSI measurements are inaccurate and can result in errors up to 50% of the measured distance. To reduce the effects of this error, nodes estimate their positions by measuring distances to as many reference points as possible. This has been shown to reduce the node location errors to less than 5% in certain cases.

In [Bul00] RSSI measurements are not used due to their imprecision. There, when a reference node sends a message, a receiving node only concludes that the distance between the two nodes is shorter than the transmission range of the reference node. The reference nodes are positioned in predefined locations. Under this framework, the position of a node is estimated as the centroid of all reference nodes that the node can hear from.

An alternative approach to distance estimation presented in [Sav01b] is measuring the time difference of arrival (TDoA) between radio and ultrasound signals. Currently, this technique is limited by the short range of ultrasound (up to 3m as reported in [Sav01b]) and by a lower capability of ultrasound in penetrating obstacles in the signal path. As in previous cases, reference nodes have their locations predefined or estimated using GPS receivers. After the nodes estimate the distances, a distributed process of *iterative multilateration* starts, where each node that estimates its location becomes a reference point for other nodes. This process allows for a maximal number of nodes to approximate their location having only a minimal number of initial reference nodes.

C. Coverage

Several different coverage formulations arise naturally in many domains. The Art Gallery Problem for example, deals with determining the number of observers necessary to cover an art gallery room such that every point is seen by at least one observer. This problem has several applications such as for optimal antenna placement problems in wireless communication. The Art Gallery problem was solved optimally in 2D [ORo92] and was shown to be NP-hard in the 3D case. Reference [Kan00] describes a general systematic method for developing a sensor network for monitoring complex systems such as a nuclear power plant. Coverage studies to maintain connectivity in a wireless network have also been the focus of study for many years. For example, [Lie98] calculates the optimum number of base stations required to achieve certain service objectives. The connectivity coverage is even more important in the case of ad-hoc wireless networks since the connections are peer-to-peer. Reference [Has97] shows the improvement in the network coverage due to multi-hop routing features and optimizes the coverage constraint with a limited path length.

References [Meg01a] and [Meg01b] present several formulations of sensor coverage in sensor networks. The formulations include calculations based on best- and worst-case coverages for agents moving in a sensor field and exposure-based methods. In the best- and worst-case formulations, the distance to the closest sensors are of importance while in exposure-based methods the detection probability (observability) in the sensor field is represented by a path- and speed-dependent integral of multiple sensor intensities.

III. Generic Localized Algorithms

In this section, we very briefly and in general terms introduce the generic localized algorithm. The objective of the section is just to introduce the main ideas. These ideas are explained in much more tangible ways in the next sections where we apply them to two important WASN problems: location discovery and exposure-based coverage.

We have developed a generic localized algorithm for solving optimization problems in wireless ad-hoc networks. As stated earlier, the technique has five components: *(i)* data acquisition mechanism, *(ii)* optimization mechanism, *(iii)* search expansion rules, *(iv)* bounding conditions, and *(v)* termination rules. The data acquisition mechanism facilitates which sensed data is obtained from which node. The optimization mechanism provides a partial or complete solution to the targeted task. Search expansion rules indicate which nodes are best to contact next. Bounding conditions indicate which nodes should not be considered further, since information that they have is irrelevant for the final solution. Finally, termination criteria indicate when search expansion and optimization mechanism can be halted.

The idea is to request and process data only locally and only from nodes who are likely to contribute to both final solution as well as to provide good bounds to determine non-promising search directions. It is important to note that initialization may start from a single point (as in the case of minimal exposure path coverage) or multiple points (as in the case of location discovery). In the second case, the search is continued simultaneously on more than one cluster of communicating nodes. Note that the clusters can overlap.

The approach enables two types of optimization. In the first, one guarantees the percentage of nodes that are contacted, while trying to optimize the quality of solution. In the second, one provides guarantees on the quality of solution, while minimizing the number of nodes that are contacted and/or amount of communication. The approach can be summarized using the following pseudo-code:

```
Generic Localized Optimization Procedure
Initiate Search;
Request Information from Neighbors;
While (termination criteria = NO) {
   Form Partial Solution;
   Decide which Nodes to Contact;
   Decide which Nodes to Terminate;
   Contact Selected Nodes;
}
```

There are two initiation steps which start the search at single or multiple locations. After that we enter a loop. The termination criteria are user specified and count either the number of contacted nodes or measures amount of communication in one case or how far we are from the final solution in the other case. The first step in the loop is to form or elaborate on partial solutions with the available information. The solution is next analyzed in terms of its distance from optimal and which direction (sensor nodes) should be contacted next. After that we terminate the search along all lines that will not yield the final optimal solution, governed by the bounding conditions.

IV. TECHNICAL PRELIMINARIES

A. Location Discovery

In the localized location discovery algorithm we use *multilateration* as an atomic procedure. Consider the simple example depicted in Figure 1. Node 0 can estimate its location based on information received from nodes 1, 2, 3, and 4. These nodes send estimates of their locations to node 0. We refer to the nodes that send an estimate of their locations as *beacons*. Beacons can acquire their location information either from previous multilateration procedures or from other sources such as GPS. Distances can be estimated using either RSSI measurements [Sav01a], ultrasound measurements [Gir01], or a combination of both [Sav01b].

To estimate the location of a node we compute the local minimum of the function L_2, express as:

$$L_2(x,y) = \sum_{i=1}^{N} \left(D_i(x,y) - R_{i0} \right)^2$$

where D_i is the approximate distance between the estimated location of node i and the location (x,y) and R_{i0} represents the RSSI- (or other) based estimate of the distance between node i and node 0. We compute the local minimum for L_2 using exhaustive search in a region that depends on the positions of the beacons and the distance measurements.

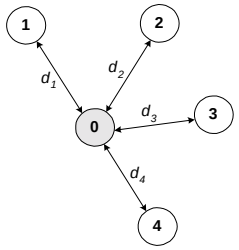


Figure 1. Multilateration Example

B. Coverage and Exposure

The exposure framework we use for the localized coverage computation is adopted from [Meg01b]. In order to make our discussion self contained, here we briefly state a summary of the important definitions and details of exposure-based coverage calculations. For a sensor s, the sensing model S at an arbitrary point p is:

$$S(s,p) = \frac{\lambda}{\left[d(s,p) \right]^K}$$

where $d(s,p)$ is the Euclidean distance between the sensor s and the point p, and positive constants λ and K are sensor technology dependent parameters. We must note here that this sensing model is mainly for experimentation purposes only. The only real requirement for the sensing model for exposure calculations is that it be non-negative and defined at every point in the field. The sensor field intensity model used for the localized algorithm is essentially the *All-Sensor Field Intensity* model $I_A(p)$ as defined in [Meg01b], with the exception that each node performing the computation may not have knowledge of the entire sensor set. Thus, each node s_j computes the sensor field intensity $I(p)$ as:

$$I(p) = \sum_{1}^{n_j} S(s_i, p)$$

where n_j is the total number of sensors (and s_i corresponds to each sensor) that s_j is aware of. As defined in [Meg01b], the exposure for an object O in the sensor field during the interval $[t_1,t_2]$ along the path $p(t)$ is:

$$E\left(p(t), t_1, t_2\right) \stackrel{\Delta}{=} \int_{t_1}^{t_2} I\left(p(t)\right) \left| \frac{dp(t)}{dt} \right| dt \,.$$

C. Voronoi Diagram, Delaunay Triangulation

In 2D, the Voronoi diagram of a set of discrete sites (points) partitions the plane into a set of convex polygons such that all points inside a polygon are closest to only one site. Reference [Aur91] presents a detailed survey of Voronoi diagrams and their applications in many domains. Delaunay triangulations are directly related to Voronoi diagrams and can be constructed by connecting the sites whose Voronoi polygons share a common edge. Neighborhood information can be extracted from the Delaunay triangulation since sites that are close together are connected. In fact the Delaunay triangulation can be used to find the closest neighbor for each site. For localized exposure calculations, we refer to nodes that are connected in the Delaunay triangulation as neighbors. Note that Delaunay neighbors are not necessarily network neighbors and may not be in communication range of each other.

V. LOCATION DISCOVERY

From the basic mechanism of multilateration described in Section IV we see that a node requires only information about the locations of its neighbors, but not the other nodes in the network to make an initial estimate of its own location. This property of multilateration makes the location discovery problem a prime candidate for localized implementation. We make two assumptions: *(i)* all nodes know their neighbors, and *(ii)* all messages are received correctly.

The theoretical background for distributed algorithms for localization is established in great detail in [Sav01a] and [Sav01b]. The localized location discovery algorithm proposed here is one of the possible approaches to collaborative location discovery. What is unique in our algorithm is that we specify in which order nodes in a network should estimate their locations such that location errors are reduced. Instead of accepting the result of a first trilateration as its final location, a node continues to accept location information from other nodes and adjust its position estimate.

If the node has more than three neighbors, it can select different groups of three nodes for each trilateration. If more than one trilateration is performed, an estimate of the location of a node is computed as the center of mass of the all generated locations. When selecting nodes that should be first to estimate their locations we specify the priority as follows:

1) Nodes whose locations generated by trilateration procedures are more consistent measured by the variance of the locations relative to the center of the mass of the locations;

2) Nodes whose majority of neighbors have already accepted their estimated locations and became reference points for other nodes.

```
Procedure Localized_Location_Discovery {
1. Initialization
If (gotGPS)
   finalLocation = GPS location ; gotFinal = true;  orphan =false;
Else If (number of neighbors >= 3)
   gotFinal = false;  orphan = false
Else
   gotFinal=false; orphan=true
Broadcast M(transmit power);
For each received message M
   calculate distance;
While (NOT (each node in neighborhood gotFinal||orphan))
   2. Location information exchange
   Broadcast M(gotFinal,orphan,location if any);
   For each received message
      record (gotFinal,orphan,location)
   N=number of neighbors;
   K=number of neighbors with location
   O=neighbors orphans
   If (N –O < 3 AND !gotFinal) orphan = true;
   If (K<3 OR gotFinal OR orphan)go to 4
   3. Trilateration
   For (i = 1 to #attempts)
      randomly select three nodes with locations
      L(i) = trilateration(locations of selected nodes)
   L = center of mass(Li);
   calculate OF
   4. Objective function comparison
   If (gotFinal OR orphan) OF = MAX_INT
   Else If (K<3) OF = MAX_INT-1;
   Broadcast OF
   If (gotFinal OR orphan) continue
   For each received message record OF
   If (OF lower than all neighbors' OFs)
      If (OF == MAX_INT-1) orphan = true;
      Else gotFinal = true; accept location;
```

Figure 2. Localized Location Discovery Algorithm

We use an objective function to evaluate these two properties of a node and its estimated location. Once nodes evaluate their objective functions they compare them with the values reported by the nodes within a predefined number of hops. In the centralized algorithm we can compare the values of the objective function for all nodes. Figure 2 lists the pseudo code of the localized algorithm where only neighbors within transmission range compare their values. If the value of the objective function for a node is lower than the values of all its neighbors, the node accepts its estimated position and becomes a reference point. Other nodes then start new rounds of multilaterations.

The algorithm presented in Figure 2 applies the generic localized procedure from Section III by limiting the area where the values of the objective function are compared to the one-hop neighborhood of a node. The algorithm contains four parts:

(1) Initialization: In the initialization part, the nodes exchange messages that may include the transmit power for that message, so receivers can use RSSI measurements to

estimate the distance to the sender. Nodes also initialize Boolean variables they use to inform their neighbors about their connectivity to the network. If a node has less than three neighbors and does not have a GPS-receiver, it cannot determine its location. Such a node sets its *orphan* flag to be true, and sends it to its neighbors in the following steps of the algorithm. Thus the neighbors will know that they should not expect any location information from that node. A node that determines the final estimate of its location, either by having a GPS-receiver or through the trilateration mechanism, sets *gotFinal* to *true*. The algorithm for a node halts when all the nodes within one hop, including the node itself, set either *gotFinal* or *orphan* to true.

(2) Information exchange: Each node receives messages from its neighbors about their state, and a location estimate, if any. If a node that still does not have its final location determines that only two or fewer of its neighbors are not *orphans*, it becomes an *orphan*, since it will not be able to estimate its location through trilateration.

(3) Trilateration: Once a node receives estimated locations from the neighbors that have that information available and it estimates the distances to its neighbors, the node performs a number of trilaterations. The number of trilaterations depends on the number of neighbors with locations (reference points). For each trilateration, the neighboring reference nodes are chosen randomly. Each trilateration produces one estimate for the location of the node. At the end of this step, a node determines an estimate of its location as a center of mass of the locations generated by all trilaterations. Each node then calculates the value of its objective function. The purpose of the objective function is to determine the order in which the nodes should accept the estimates of their locations. The objective function evaluates the properties of a node: consistency of the estimates by measuring the variance of the trilateration-generated estimates relative to their center of the mass, and a percentage of the neighbors that already estimated their locations. The second property ensures that the nodes that cannot get many new information updates, estimate their locations before the nodes that may get a significant number of new neighbors to be included in trilaterations.

(4) Objective function comparison: After the objective functions are calculated, the neighbors exchange their values. Only nodes whose objective function value is lower than all values announced by their neighbors will accept the estimates from the previous step as their locations. All other nodes discard their estimates. A node terminates the algorithm when its location is determined or the node becomes an orphan and each of the neighbors are in one of those two states. Since in every pass of the algorithm at least one node either gets its location or becomes an orphan, the algorithm always terminates.

VI. EXPOSURE

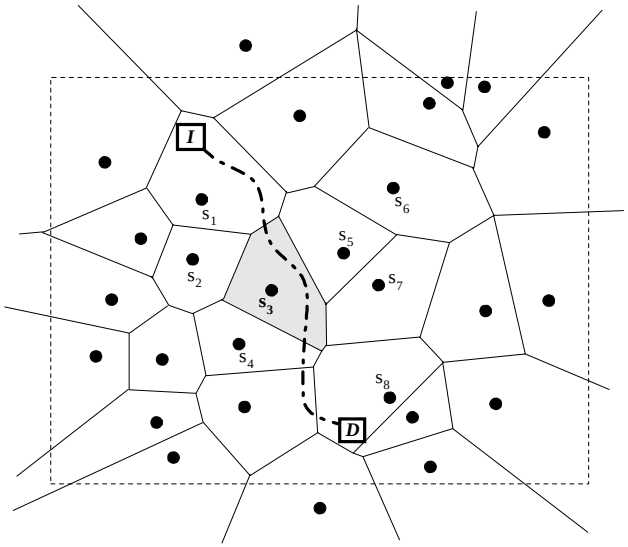


Figure 3. Overview of the Localized Minimal Exposure Path algorithm.

To introduce the localized exposure-based coverage algorithm, we start by presenting an overview of the computation. Consider the sensor network instance presented in Figure 3. Each dark circle represents a sensor node in the field. The problem we are trying to solve is finding the Minimal Exposure Path from a given initial point *I* to a destination point *D*. The Minimal Exposure Path P_{minE} is a path in the sensor field that connects the arbitrary points *I* and *D*, along which the total exposure $E(P_{minE})$ is minimized. Reference [Meg01b] presents a detailed discussion on exposure formulations and their application in characterizing coverage in sensor networks, as well as a centralized algorithm for finding P_{minE}. Here, our goal is to provide a framework for a localized implementation of the Minimal Exposure Path algorithm. The essential underlying factor that necessitates the design and use of a localized algorithm as opposed to the centralized one is that often, in ad-hoc environments, each node may not have complete information about all other nodes in the network. The only assumption that we make is that nodes know the position of their immediate (Delaunay) neighbors. Hence, the formal problem statement becomes:

Given: A field *A* instrumented with sensors *S;* Areas *I* and *D* corresponding to initial (*I*) and destination (*D*) locations of the agent.

Problem: Identify $P_{minE}(I,D)$, the minimal exposure path starting in *I* and ending in *D*.

Localization Assumption: Each node $s_i \in S$ is assumed to have the location information about itself and its immediate neighbors, i.e. all nodes that are in neighboring Voronoi polygons (Delaunay neighbors).

Note that each node $s_i \in S$ may have knowledge of many other, potentially all other, nodes in S. In certain cases, nodes that are far apart (relative to the communication range) may be Delaunay neighbors. In such cases the algorithm will still work and the specific implementation may choose to ignore such neighbors that are far apart since their discovery may be too costly. Although this design decision may result in globally sub optimal solutions, it generally will not effect the optimality of the localized algorithm (local results).

Dividing the space for the purpose of localization is the first task in designing the Minimal Exposure Path algorithm. Several potential strategies include the use of a global grid, circular regions around each node, and the Voronoi Diagram corresponding to the sensor nodes. As shown in Figure 3, we opt to use the Voronoi Diagram to partition the sensor field such that each sensor node s_i is responsible for exposure calculations in its Voronoi polygon VP_i. Such a partitioning has several advantages:

1) Only one sensor can exist in each polygon VP_i therefore eliminating multiple, possibly different, exposure estimations in a specific region in the field.

2) Each node can compute its Voronoi polygon efficiently, knowing only the locations of its immediate neighbors (*Localization Assumption*).

3) Areas of higher node density are naturally divided with more polygons (smaller) hence increasing the computation granularity where it is needed most.

One of the drawbacks of using the Voronoi based field partitioning is that potentially, a single node may be responsible for servicing the computation for a physically larger area compared to another node in a denser region. The larger Voronoi polygon results in a potentially higher number of requests processed by the node since on average, more minimal exposure paths are likely to traverse larger polygons than smaller ones. Such issues are important when considering performance metrics like balanced computation workloads among sensor nodes and optimal energy consumption and their study will be the focus of future research efforts.

Now that we have established the partitioning method of the field and the sensor nodes responsible for each region, we can focus on the details of finding P_{minE}. First, let us briefly review the framework for calculating path exposures within the Voronoi Polygon VP_i of a sensor node s_i. Since the space is continuous and the exposure integral does not have simple analytic solutions, we use numerical methods and a generalized grid, similar to what is presented in [Meg01b]. The generalized grid consists of regular squares that span the VP_i, with the addition that each square has a finite number of divisions along each edge. The algorithm constructs the graph G_i corresponding to the grid covering VP_i while in each grid square, the division points along each edge are connected to all other division points in the square. This is to ensure that path movements are not restricted to only horizontal, vertical, and diagonal directions. The exposure integral along each line-segment is approximated using numerical techniques and is assigned as the weight to the corresponding edge in the graph. The minimal exposure paths can be calculated from each point to every other point on the boundary of the polygon VP_i using the Floyd-Warshall All-Pairs-Shortest-Paths algorithm or generalized forms of Dijkstra's Shortest-Path algorithm [Cor90] (dynamic programming). The grid resolution parameters play an important role in the optimality of the solutions. Increasing the grid detail will yield higher quality solutions at the cost of higher processing, storage, and energy consumption.

To compute the point-to-point exposure (*PPE*) along a path segment, we need the sensor field intensities $I(p)$ along the path. As stated in the localization assumption, each node s_i has the location information of its neighbors and thus can approximate $I(p)$. Clearly, if node s_i has location information of other active nodes in the sensor field, it will include them in the intensity function thus increasing the accuracy of results. In practical instances, sensor nodes that are far away do not contribute significantly to the sensing intensity $I(p)$.

To describe the node interactions during the minimal exposure path computation, we now focus on the different cases that each node must handle. We categorize the algorithm tasks into four cases:

1) ***Path_request***: Node s_i receives a request from an agent to find P_{minE} from I to D .

2) ***Edge_update****:* Node s_i receives an update notification from a neighbor to continue "search" for $P_{minE}(I,D)$.

3) ***Abort_update***: Aborting conditions notification.

4) ***Dest_update***: Destination reached notification.

<u>*Path_request:*</u> The *path_request* query generated by an agent contains the initial location I and the destination location D. This query is initially received and handled by the node closest to I which we call s_i. Node s_i then computes the

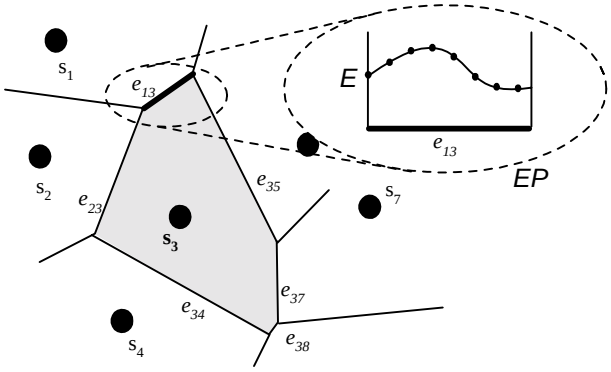


Figure 4. An Edge Exposure Profile (EP) Example

minimal *PPEs* from *I* to points along the VP_i boundary edges and thus builds the initial Exposure Profile (*EP*) for each boundary edge. There are several ways of encoding and sending the *EP*s: curve fitting, various data-compression techniques, and using discrete values at specific intervals. As shown in Figure 4, we use the latter method in our implementation due to its simplicity.

<u>*Edge Update:*</u> Node s_i sends an *edge_update* message to each neighbor s_j that contains the $EP(e_{ij})$, where e_{ij} is the Voronoi edge between node *i* and *j*. We use the example depicted in Figure 5 to discuss the details of this process. Here, node s_1 has received a *path_request*, computed the $EP(e_{13})$ and sent an *edge_update* message to node s_3. If s_3 does not have an existing *EP* for edge e_{13}, the received *EP* is stored as the current $EP(e_{13})$ at node s_3. Otherwise, s_3 computes the combined $EP(e_{13})$ by comparing the received $EP(e_{13})$ in the *edge_update* message to its current $EP(e_{13})$ and keeping the lower exposure values for each point (see Figure 6).

Using the new $EP(e_{13})$, node s_3 computes and updates the *EP*s for all other edges: $EP(e_{23})$, $EP(e_{34})$, $EP(e_{38})$, $EP(e_{37})$, $EP(e_{35})$. A new *edge_update* is sent to the corresponding neighbors. For each edge_update message sent by node s_i, s_i must receive one termination message (acknowledgement) in order to know when the search for a particular request has been completed. This requires that s_i store bookkeeping information about each request and is appending a unique *request_id* identifiers on each *edge_update* message.

<u>*Abort update:*</u> To limit the extent of the search, we use a global parameter λ which is the bound on maximum path exposure. Before a node s_i sends out an *edge_update* message to s_j, it checks to see whether the minimum point of $EP(e_{ij})$ is above the current value of λ. If that is the case s_i will not send the *edge_update* message to s_j thus terminating further search from that edge. Node s_i will instead send an *abort_update* message to the node that had sent the *edge_update* message. This mechanism ensures that the search for the minimal exposure path does not expand arbitrarily far.

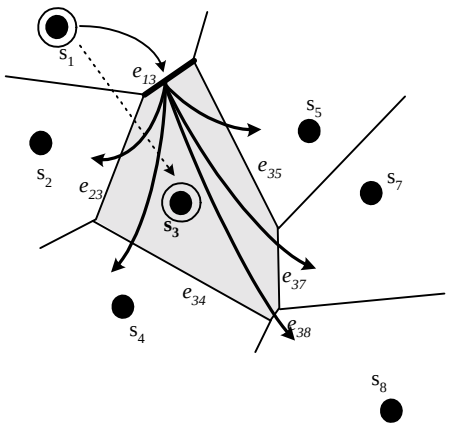


Figure 5. Localized minimal exposure path calculation at each node.

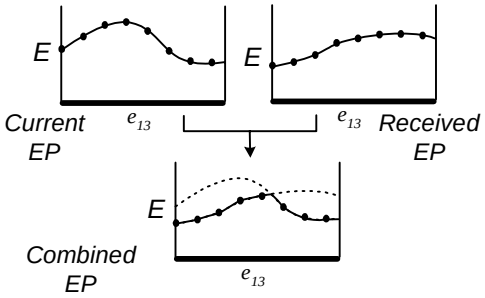


Figure 6. Combining Exposure Profiles for edge e_{13}.

<u>*Dest_update:*</u> Once a node computes the exposure to the destination *D* it broadcasts the result in a *dest_update* message. The parameter λ is updated by taking the minimum of its current value and the exposure at *D*. Since we have already found a path to *D* there is no reason to expand the search on paths that have larger exposures. The algorithm is done once all nodes terminate their search and report back to the initiating node. Note that the information about the minimum exposures at the boundaries of the search edges are propagated using this mechanism and the search continues until all boundary exposures are above the minimal exposure up to *D* (latest λ value).

The path P_{minE} is constructed by tracing from the destination *D* back to *I*. This is done in a separate phase of the algorithm the details of which we omit here.

The initial termination parameter λ is determined heuristically at the node that services the agent query. Several alternatives exist for the calculation of the initial λ such as by considering the average local or global node density and estimating the exposure based on the path length from *I* to *D*. However, in our current implementation, the initial λ is a user-specified parameter. If the search is unsuccessful, i.e. the $E(P_{minE}) \geq \lambda$, then the initiating node increases λ by a constant value (linearly) and repeats the search. The overhead of increasing λ can be reduced significantly by caching previous computations at each node. Also, note that although λ is a global constant, we do not make any specific assumptions about its consistency among all nodes. Although we assume that each node will eventually receive the messages that contain updated λ information (reliable message delivery) we do not require that these messages arrive synchronously or in order.

In order to avoid the excessive messages that can be generated if the path crosses an edge in several places, we augment the algorithm and only generate an *edge_update* message if the improvement in *EP* of an edge is better than a specified threshold. For example, as the parameter *K* in the equation for *S(s,p)* increases, the minimal exposure paths tend to lie very close to the edges of the Voronoi Diagram. This mechanism ensures that the number of unnecessary update messages among nodes is minimized.

Procedure Localized_Minimal_Exposure_Path
For each *edge* $e_{ij} \in VP$ *update_count*$(e_{ij}) = 0$

EVENT LOOP:
Case *path_request(I,D)* received from agent *a*
  For each *edge* $e_{ij} \in VP$
    Calculate $EP(e_{ij})$ from *I*.
    Send *edge_update*(e_{ij})

Case *edge_update*$(EP_r(e_{pq}))$ received from s_p
  If $EP_r(e_{pq}) \geq EP(e_{pq})$
    Send *abort_update(request_id)* to s_p
  Else
    $EP(e_{pq}) = min\ (EP_r(e_{pq}), EP(e_{pq}))$
    For each *edge* $e_{ij} \in VP$
      Compute $EP_{new}(e_{ij})$
      If $EP_{new}(e_{ij}) \leq min(\lambda, EP(e_{ij}))$
        $EP(e_{ij}) = min\ (EP_{new}(e_{ij}), EP(e_{ij}))$
        Store s_p *in requestor list, assign new request_id*
        Send *edge_update*$(e_{ij}$, *request_id)*
    If $D \in VP$
      E_D = *min exposure to D*
      Broadcast *dest_update*(E_D)

Case *result_request* received from s_p
    If $D \in VP$
      $P_{minE} = \emptyset$
      Lookup P_{minE} ending at *D* to boundary of *VP*
      Send *result_request*(P_{minE}) *to next neighbor*
    Else
      Lookup part of P_{minE} in *VP*
      Send *result_request*(P_{minE}) *to next neighbor*

Figure 7. Minimal Exposure Path Algorithm

VII. EXPERIMENTAL RESULTS

A. Location Discovery

To test the localized location discovery algorithm, we randomly generate the locations of 50 nodes within a square (40m wide). Transmission range of the nodes is 10m. Except in simulations where the effect of the number of nodes with GPS-receivers is examined, 10 nodes are equipped with GPS-receivers. The inaccuracy of GPS-receivers is simulated by adding an error to the coordinate of the exact location of each node. The error is generated following the Gaussian distribution with the parameters (0;*GPS_Error*/3). The distance error is modeled as a percentage of each particular correct distance *d*. The parameter *Distance_Error* represents percentage of the distance *d*. The error added to the distance *d* is generated from the Gaussian distribution (0;(*Distance_Error**d)/3).

In the first set of simulations we wanted to find out whether, under any circumstances, nodes with GPS-receivers could benefit from measurements received from other nodes in the network. We compared the average location error in a network when GPS-equipped nodes adapt their GPS-based location information to the locations of other nodes and distances to them, and the average error in a network where GPS-based location information is accepted as the final estimate of a node's location. For two values of *Distance_Error*, 0.05 and 0.25, we varied the value of *GPS_Error* as shown in Figure 8. As shown, while both cases (especially for higher values of the *GPS_Error*) achieve similar performances, better results can be achieved by retaining GPS-based locations, and use GPS-equipped nodes as anchors without imposing constraints from distance measurements on them.

After we answered the question about the best way to use GPS-based estimates, we compared the centralized and a localized version of the algorithm in order to estimate what, if any, improvement in precision can be made if a centralized version is used. The parameters of the simulation are the following: *Distance_Error*=0.25, *GPS_Error*=10, the number of the nodes with GPS receivers varies from 7 to 12. Although in all cases the centralized version of the algorithm achieves better results, the biggest difference is for 7 nodes with GPS in the network, which is 0.5 (around 7%). Such a small difference affirms that the location discovery is inherently a distributed problem, and there is no need for centralized processing. It is interesting to note that the more

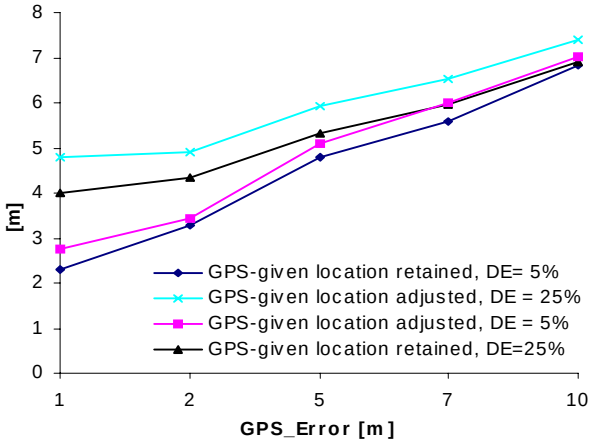


Figure 8. Average distance between the real location of a node and the estimate, for two different approaches of using GPS beacon location estimates.

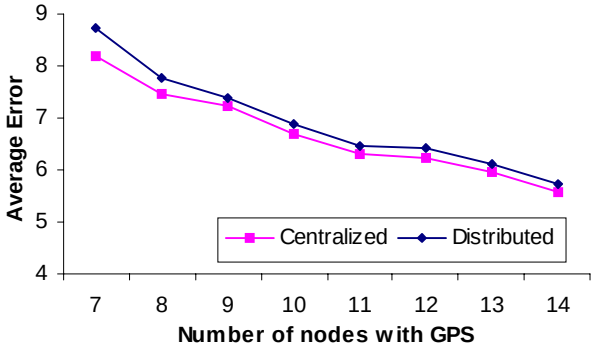


Figure 9. Performance of localized and distributed versions of the location discovery algorithm.

nodes with GPS-receivers are in the network, the difference between two approaches decreases. This can be explained with the fact that with more GPS nodes, the majority of nodes receive a location estimate through trilateration with three GPS nodes whose locations do not change during the location discovery process. Thus, the ordering of trilaterations is less important.

B. Exposure

We use Glomosim 2.0 [Glo2] to implement and test the localized Minimal Exposure Path algorithm. We randomly generate the topology following the uniform distribution in a square. The simulations are executed on a 1GHz Intel Pentium III-based workstation with 512MB memory. We have also created a user friendly graphical interface using Visual C++ and OpenGL running in Microsoft Windows environment. Table 1 lists the results of four such topology instances with 30 sensor nodes and the results obtained using the centralized and localized versions of the algorithm. For all cases the minimal exposure path was computed starting and ending in opposite corners of the field. The results show that when using all sensor information in the localized algorithm the outcomes are similar to the centralized version with the exception of numerical and rounding errors (boundary effects) at the edges of the polygons separating neighboring nodes. The table also lists the results obtained using only information from immediate neighbors in exposure computation. As expected, the exposure levels are significantly lower in this case due to incomplete information while processing.

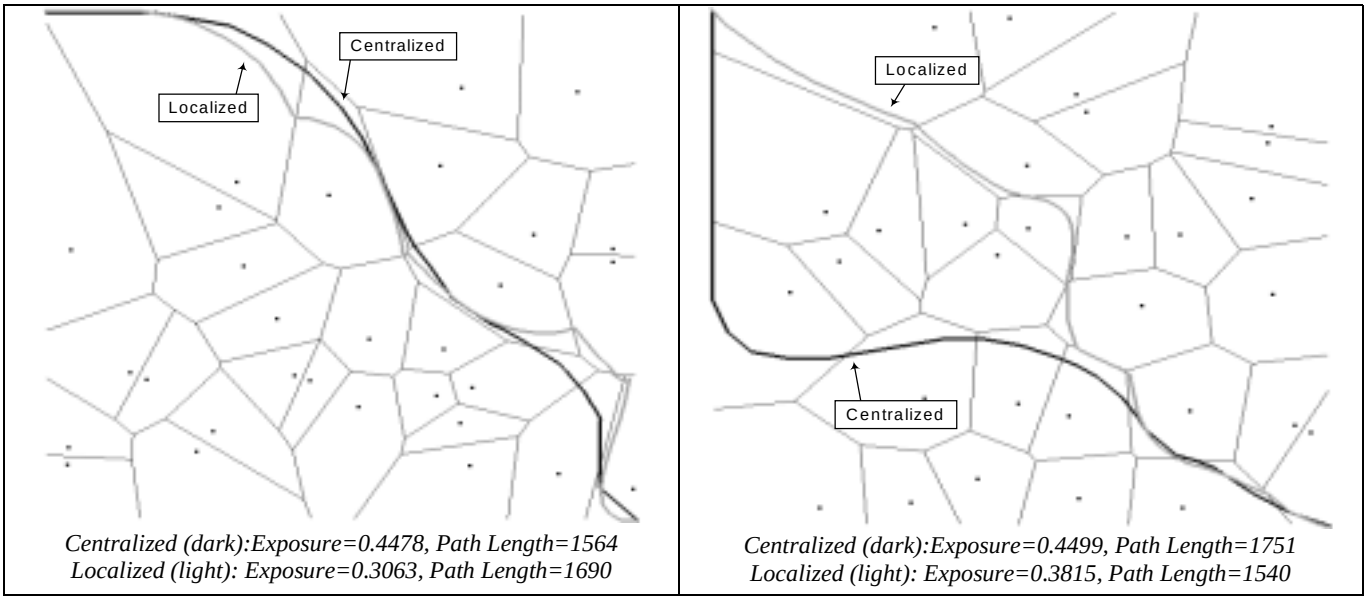
Figure 9 shows the minimal exposure paths obtained using the centralized and the localized algorithms for two randomly generated sensor network instances (*I* and *D* placed at opposite corners of the field). Figure 10 shows a more global picture of the localized algorithm in action. The shaded region in the figures depict the extent of the search expansion to find the minimal exposure paths.

	Centralized		Localized	
Name	**Exposure**	**Length**	**Exposure**	**Length**
30-1-all	0.3865	1903.5	0.3961	1890.2
30-2-all	0.4478	1563.7	0.4595	1672.4
30-3-all	0.4590	1584.7	0.5050	1563.3
30-4-all	0.3972	1630.1	0.4176	1560.3
30-1-del			0.2572	1921.9
30-2-del			0.2964	1683.4
30-3-del			0.3560	1582.6
30-4-del			0.2665	1572.7

Table 1. Minimal Exposure Path results using *all* sensors and *Del*aunay neighbors only (localized).

VIII. CONCLUSION

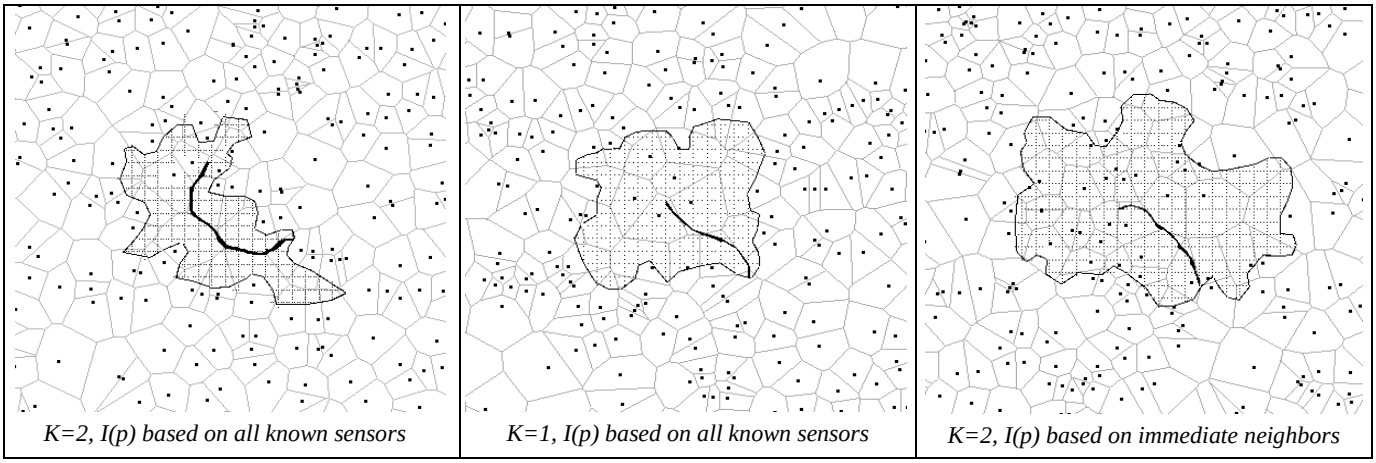
We presented a generic framework for designing localized optimization algorithms in wireless ad-hoc sensor networks. The two driver examples, namely location discovery and minimal exposure path algorithms were presented in detail. In each case, the number of nodes that are contacted and participate in the computation are generally confined within a geographical region while still producing solutions of high quality, compared to centralized schemes. In addition to the detailed algorithmic discussions, we also presented a sampling of experimental results in order to illustrate the effectiveness of the localized approach for location discovery and minimal exposure path. The results indicate that localized algorithms in ad-hoc sensor networks can produce results comparable to their centralized counterparts, while providing practical implementation and deployment options.



Centralized (dark):Exposure=0.4478, Path Length=1564
Localized (light): Exposure=0.3063, Path Length=1690

Centralized (dark):Exposure=0.4499, Path Length=1751
Localized (light): Exposure=0.3815, Path Length=1540

Figure 9. Minimal Exposure Path results using centralized and localized algorithms for two randomly generated network topologies. In the localized version only Delaunay neighbors are used in exposure computations.



K=2, I(p) based on all known sensors | *K=1, I(p) based on all known sensors* | *K=2, I(p) based on immediate neighbors*

Figure 10. Extent of search expansion in finding Minimal Exposure Path for three sensor network instances.

REFERENCES

[Abe00] H. Abelson, et. al. "*Amorphous Computing*." Communications of the ACM, Volume 43, No. 5, pp. 74-82, May 2000.

[Abi00] A.A. Abidi, G.J. Pottie, W.J. Kaiser, "*Power-Conscious Design Of Wireless Circuits And Systems*." Proceedings of the IEEE, vol.88, (no.10), pp. 1528-45, Oct. 2000.

[Adj99] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, "*The Design And Implementation Of An Intentional Naming System.*" Operating Systems Review, vol.33, (no.5), pp.186-201, Dec. 1999.

[Bad93] B.R. Badrinath, et. al. "*Impact Of Mobility On Distributed Computations*." ACM Operating Systems Review, April 1993.

[Aur91] F. Aurenhammer, "*Voronoi Diagrams – A Survey Of A Fundamental Geometric Data Structure*," ACM Computing Surveys 23, pp. 345-405, 1991.

[Ben93] M. Bender et. al. "*Unix For Nomads: Making Unix Support Mobile Computing*." In Proceedings of the USENIX Symposium on Mobile & Location Independent Computing, August 1993.

[Beu99] J. Beutel, "*Geolocation In A PicoRadio Environment*." M.S. Thesis, ETH Zürich, Electronics Lab, 1999.

[Bul00] N. Bulusu, J. Heidemann, D. Estrin, "*GPS-less Low Cost Outdoor Localization For Very Small Devices*." IEEE Personal Communications, Special Issue on "Smart Spaces and Environments", Vol. 7, No. 5, pp. 28-34, October 2000.

[Cor90] T. Cormen, C. Leiserson, R. Rivest, *Introduction to Algorithms*, MIT Pres, June 1990.

[Dow98] T.B. Downing. *Java RMI: Remote Method Invocation*. IDG Books Worldwide, New York, 1998.

[Dur89] E.H. Durfee, V.R. Lesser and D.D. Corkill, "*Trends in Cooperative Distributed Problem Solving*." IEEE Transactions on Knowledge and Data Engineering, Vol.1, pp. 63-83, March 1989.

[Est00] D. Estrin, R. Govindan, J. Heidemann. "*Embedding the Internet: Introduction*." Com. of the ACM, Vol. 43, pp. 38-42, May. 2000.

[Has97] Z.J. Haas, "*On The Relaying Capability Of The Reconfigurable Wireless Networks.*" IEEE 47th Vehicular Technology Conference, Vol.2, pp. 1148-52, May 1997.

[Int00] C. Intanangonwiwat, D. Estrin, R. Govindan, "*Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks.*" Procs. of ACM MobiCOM 2000, August 2000.

[Gal83] R.G. Gallager, P.A. Humblet, P.M. Spira, "*A Distributed Algorithm For Minimum Spanning Tree*." ACM Transactions on Programming Languages and Systems, pp. 66-77, 1983.

[Gir01] L. Girod, D. Estrin, "*Robust Range Estimation Using Acoustic And Multimodal Sensing*." IEEE/RSI Int. Conf. on Intelligent Robots and Systems (IROS 2001), Maui, Hawaii, October 2001 (To Appear).

[Hoa85] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall International, 1985.

[Kan00] C.W. Kang, M.W. Golay, "*An Integrated Method For Comprehensive Sensor Network Developement In Complex Power Plant Systems*." Reliability Engineering & System Safety, vol.67, pp. 17-27, Jan. 2000.

[Kis92] J. Kistler, M. Satyanarayanan. "*Disconnected Operation In The Coda File System*." ACM Trans. on Computer Systems, Feb. 1992.

[Lam78] L. Lamport, "*Times, Clocks, And The Ordering Of Events In A Distributed System*." CACM, pp. 558-565, July 1978.

[Lam90] L. Lamport, N. Lynch, "*Distributed Computing: Models And Methods*." Handbook of Theoretical Computer Science, pp. 1158-1199, Elsevier Science Publishers, 1990.

[Lie98] K. Lieska, E. Laitinen, J. Lahteenmaki, "*Radio Coverage Optimization With Genetic Algorithms*." IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications, pp. 318-22, Sept. 1998.

[Lyn96] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, San Francisco, 1996.

[Meg01a] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. Srivastava, "*Coverage Problems in Wireless Ad-Hoc Sensor Networks*." IEEE Infocom 2001, Vol. 3, pp. 1380-1387, April 2001.

[Meg01b] S. Meguerdichian, F. Koushanfar, G. Qu, M. Potkonjak, "*Exposure In Wireless Ad Hoc Sensor Networks*." Procs. of Int. Conf. on Mobile Computing and Networking (MobiCom '01), July 2001.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.

[MIT82] MIT Lincoln Laboratories. Workshop on Distributed Sensor Networks. 1982.

[Glo2] Glomosim 2.0. http://pcl.cs.ucla.edu/projects/glomosim/

[ORo92] J. O'Rourke, *Computational geometry column 15. (Open problem from art gallery solved)*. Int. Journal of Computational Geometry & Applications, vol.2, pp.215-17, June 1992.

[Pot00] G. J. Pottie, W. Kaiser, "*Wireless Integrated Network Sensors.*" Communications of the ACM, Vol. 43, No. 5, pp. 51-58, May 2000.

[Ray88] M. Raynal, *Distributed Algorithms and Protocols*, John Wiley and Sons, 1988.

[Rum86] D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986.

[Sav01a] C. Savarese, J. Rabaey, J. Beutel, "*Locationing in Distributed Ad-Hoc Wireless Sensor Networks.*" Proceedings of the ICASSP, May 2001.

[Sav01b] A. Savvides, C.C. Han, M.B. Srivastava, "Dynamic *Fine-Grained Localization in Ad-Hoc Networks of Sensors.*" Procs. of MobiCom 2001, July 2001.

[Tel94] G. Tel, *Introduction to Distributed Algorithms.* Cambridge University Press, Cambridge, U.K., 1994.

[Ten00] D. Tennenhouse, "*Proactive computing.*" Communications of the ACM, vol. 43, No. 5, pp. 43-50, May 2000.

This material is based upon work partially supported by the National Science Foundation under Grant No. NI-0085773 and DARPA and Air Force Research Laboratory under Contract No. F30602-99-C-0128. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, DARPA, or Air Force Research Laboratory.