The Bits and Flops of the N-hop Multilateration Primitive For Node Localization Problems

Andreas Savvides, Heemin Park and Mani B. Srivastava

Networked and Embedded Systems Lab
Electrical Engineering Department
University of California, Los Angeles
{asavvide,hmpark,mbs}@ee.ucla.edu

ABSTRACT

The recent advances in MEMS, embedded systems and wireless communication technologies are making the realization and deployment of networked wireless microsensors a tangible task. Vital to the success of wireless microsensor networks is the ability of microsensors to "collectively perform sensing and computation". In this paper, we study one of the fundamental challenges in sensor networks, node localization. The collaborative multilateration presented here, enables ad-hoc deployed sensor nodes to accurately estimate their locations by using known beacon locations that are several hops away and distance measurements to neighboring nodes. To prevent error accumulation in the network, node locations are computed by setting up and solving a global non-linear optimization problem. The solution is presented in two computation models, centralized and a fully distributed approximation of the centralized model. Our simulation results show that using the fully distributed model, resource constrained sensor nodes can collectively solve a large non-linear optimization problem that none of the nodes can solve individually. This approach results in significant savings in computation and communication, that allows fine-grained localization to run on a low cost sensor node we have developed.

Categories and Subject Descriptors

C.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance

Keywords

Ad-Hoc Localization, Distributed Localization, Sensor Networks

WSNA'02, September 28, 2002, Atlanta, Georgia, USA.

1. INTRODUCTION

Precise knowledge of node location in ad-hoc deployed microsensor networks yields a wide variety of profound advantages. Knowledge of location can be used to report the geographical origin of events, to assist in target tracking, geographic aware routing [10], to administer the sensor network and evaluate its coverage [11]. These together with security and smart environment applications such as the Smart Kindergarten [18] are only a few of the applications where location aware nodes are required.

In many situations, wireless sensor nodes are expected to be deployed in an ad-hoc fashion (i.e air-dropped over an area). With ad-hoc deployment however, one cannot accurately predict or plan a-priori the location of each sensor. Based on this and the fact that direct line-of-sight with beacons is not always feasible, we seek to develop an algorithm that can perform precise localization of sensor nodes with indirect line-of-sight by utilizing location information and distance measurements over multiple hops. To achieve this goal, nodes use their ranging sensors to measure distances to their neighbors and share their measurement and location information with their neighbors to collectively estimate their locations.

In this paper we present *collaborative multilateration* that we also refer to as the *n-hop multilateration primitive*. Collaborative multilateration consists of a set of mechanisms that enables nodes found several hops away from location aware beacon nodes collaborate with each other to estimate their locations with high accuracy. This multihop capability waives the line-of-sight to beacons requirement making fine-grained localization possible while requiring very few beacon nodes. Position estimates are obtained by setting up a global non-linear optimization problem and solving it using iterative least squares. Collaborative multilateration is presented in two computation models, centralized and distributed. These can be used in a wide variety of network setups from fully centralized where all the computation takes place at a base station, to locally centralized (i.e computation takes place at a set of cluster heads) to fully distributed where computation takes place at every node.

The fully distributed computation model presented here is an approximation of its centralized counterpart and has several properties favorable to sensor networks. It offers a significant reduction in computation requirements thus allowing the execution of collaborative multilateration on resource constrained sensor nodes such as the Medusa MK-2

node described in this paper. Using this mechanism, resource-constrained nodes can collaborate with each other to jointly estimate their locations, a task that none of the nodes can perform individually because of their computation and memory limitations. The use of a fully distributed computation model is also tolerant to node failures, and distributes the communication cost evenly across the sensor nodes and does not require any additional supporting mechanisms such as leader election and multihop routing that would be required for a fully centralized implementation.

The algorithms presented in this paper are validated on a combined ns-2 and MATLAB simulation testbed, using the measured parameters of our experimental sensor nodes. The remainder of this paper is organized as follows. Next section briefly describes the related work. Section 3 provides some preliminary information and overview of our approach. Section 4 describes the initial setup configuration for collaborative multilateration. Section 5 explains centralized and distributed computation models. Our simulation results are presented in section 6 and section 7 concludes the paper.

2. RELATED WORK

Node localization has been the topic of active research and many systems have made their appearance in the past few years. A detailed survey of such systems is provided by Hightower and Boriello in [7]. Despite these efforts, very few systems are actually ad-hoc. Even fewer methods are have been proposed that offer a fully distributed operation. Doherty's [3] convex position estimation approach for instance describes a method for localizing ad-hoc nodes based only on connectivity. This method is based on semi-definite programming and requires rigorous centralized computation so it is not always suitable for many ad-hoc setups.

Some forms of ad-hoc localization also exist in the domain of mobile robotics [8, 13]. The localization problem in mobile robotics is similar to the ad-hoc localization problem investigated in sensor networks. One main difference however is that mobile robots have additional odometric measurements that can help with estimating the initial robot positions, something that is not available in sensor networks. Furthermore, localization studies in the sensor network community also consider scalability communication and power consumption issues that are not studied by the robotics community.

The AHLoS system we proposed in [16] uses iterative multilateration which relies on a small set of nodes initially configured as beacons to estimating node locations in an ad-hoc setup. This work identified two main problems: 1) without relatively large beacon densities, the iterative multilateration may get stuck in regions of the network that are relatively sparse, and 2) error propagation becomes an issue in large networks. The collaborative multilateration algorithms presented in this paper address these two issues.

In parallel to our work some other ad-hoc node localization approaches have been independently proposed in [12] and [15]. In both these approaches anchor node location information is propagated in the network. Nodes with unknown location note the shortest hop distance to each of the anchor nodes a multiply this an average hop distance to get an approximate distance to each of the anchor nodes. With this information nodes perform a multilateration to get an initial estimate of their locations. To obtain better estimates, the authors of [15] also use a further refinement phase that uses least squares to refine node positions based on local computation. The authors claim that their approaches are independent of ranging technologies and can deliver localization accuracy within one third of the communication range.

Despite the similarities, the work presented here has several fundamental differences from the aforementioned approaches. We provide a mechanism for limiting error propagation by computing in the context of over-constrained node configurations (the collaborative subtrees). At the same time, we provide a fully distributed computation model and we demonstrate its scalability. Our algorithms are targeted for a fine-grained ad-hoc localization system that we are currently developing.

3. PRELIMINARIES

3.1 Establishing Local Coordinate Systems

The initial locations beacon nodes can be obtained either by manual placement or by automatically establishing a local coordinate system. One method for establishing a coordinate system is to deploy some nodes that are capable of accurate long distance ranging (i.e. long range ultrasound or laser range finders). These nodes establish a local coordinate system as shown in figure 1. In this figure nodes A, B and Γ are equipped with laser range finders. The nodes can communicate with each other and decide on a local coordinate system. One solution is that node A becomes the origin with coordinates $(0, 0)$, while Γ has coordinates $(A\bar{\Gamma}, 0)$ and B has coordinates $(AB \sin\alpha, AB \cos\alpha)$. Such local coordinate systems can be established at different places inside the network. Capkun et. al. in [1] describe a very similar method for forming and merging local coordinate systems. The scheme described in [1] creates a local coordinate systems at each node and merges them together into a global coordinate systems. This approach is purely based on geometric manipulations that can potentially result in significant error accumulation in the network. In our approach, the formation of local coordinate systems is performed as part of a tiered architecture that establishes a set of initial anchor points, the beacon nodes, but the core localization relies on collaborative multilateration in order to prevent error accumulation.

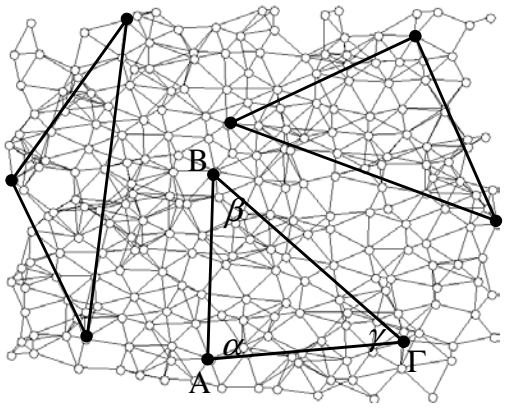

Figure 1: Establishing Local Coordinate Systems

3.2 Example Sensor Node and Ranging Considerations

Figure 2 shows our second generation node, the Medusa MK-2, a low cost wireless sensor node we have developed for experimenting with node localization problems. This node consists of a 4MHz 8-bit AVRMega128L microcontroller from Atmel and a low power RFM radio, in a configuration that is similar to UC Berkeley's Mica motes. In addition the MK-2 node carries a more powerful 40MHz AT91FR4081 ARM THUMB coprocessor with 136KB of RAM and 1MB of on board FLASH memory for more intensive computation tasks. The details and additional features of the MK-2 architecture are described in [17].

For localization, the node is equipped with 40KHz ultrasonic sensors that have an effective range of 5 meters and approximately 1cm accuracy. Similar technologies can produce longer ranges, but we found this range to be more appropriate for indoor settings. The wideband acoustic method presented by Girod and Estrin in [5] is also a notable ranging technology that provides some immunity to common multipath effects and would provide a good ranging alternative.

Figure 2: Our experimental node

3.3 Solution Outline

In this paper the single hop multilateration operation performed by GPS is extended to operate on multiple hops. This enables nodes that are not directly connected to beacon nodes to collaborate with other intermediate nodes with unknown locations situated between themselves and the beacons to jointly estimate their locations. One of the main challenges in this problem is to prevent error accumulation inside the network. To prevent error accumulation, the node localization problem is set up as a least squares estimation problem with respect to the global network topology.

Collaborative multilateration takes place in three main phases: 1) formation of collaborative subtrees, 2) computation of initial estimates, 3) position refinement. During the first phase, the nodes form a well-constrained or over-constrained configuration of unknowns and beacons, the collaborative subtrees (section 4.1). This configuration forms a system of at least n non-linear equations and n unknown variables to be determined. Collaborative subtrees also ensure that each unknown member of the computation subtree has a unique possible solution. This prevents the iterative least squares refinement process in the first phase from computing estimates that are numerically correct but are not the correct solutions. The nodes that do not meet the criteria for collaborative subtrees cannot participate in this configuration. The position estimates for such nodes are determined later in a post-processing phase. In the second phase, each unknown node computes an initial estimate of its location based on the known beacon locations and the inter-node distance measurements. These initial estimates are used to initialize the refinement process in the third phase. The third phase computes a least squares estimate of the node locations. Finally, a post-processing phase uses the computed node estimates to refine the position estimates of nodes that could not participate in the computation subtree configuration. This phase has the similar functionality as the phase for computing initial estimates but it is more constrained by the newly computed location estimates in the computation subtree.

The first and second phases are independent from each other and can take place in parallel in an actual network. Phase three can start as soon as the first two phases are completed and it can terminate at different stages depending on the demands of the application. If computation is done at a central point (either a central computer for the whole network, or a local cluster head), the process will terminate when the unknown nodes receive their position estimates. If the distributed computation form is used, then the processes termination depends on the demands of the application. If the application requires just an indication of proximity, the localization process can only perform the second phase and terminate. If more accurate localization is required, the distributed computation will continue until required precision is achieved.

4. INITIAL CONFIGURATION

4.1 Phase 1: Collaborative subtrees

A computation subtree constitutes a configuration of unknowns and beacons for which the solution to the position estimates of the unknowns can be uniquely determined. This is achieved by obtaining a well determined or preferably over-determined set of equations - n variables to be estimated and at least n equations. Before attempting to solve these equations, the solution uniqueness is determined to prevent the estimation of erroneous locations. Collaborative subtrees have another desirable property that will become apparent when we discuss our distributed computation model in section 5.2. To determine the requirements for solution uniqueness we develop our discussion by reviewing the requirements of the single hop multilateration. Later on, we augment these requirements to cover the multihop case.

4.1.1 One-Hop Multilateration Requirements

In the single hop setup of figure 3a, the basic requirement for one unknown node to have a unique solution on a 2D plane is that it is within range of at least three beacons. If the beacons lie in a straight line, the node configuration is symmetric, and there is more than one possible solution.

4.1.2 Two-Hop Multilateration Requirements

Using the one-hop multilateration requirements as a starting point, the corresponding set of requirements for a two-hop multilateration can be established. A two-hop multilateration represents the case where the beacons are not always directly connected to the node but they are within a two-hop radius from the unknown node. In this situation, two or more unknown nodes can utilize the beacon location information and the intermediate distance measurements between themselves and the beacons to jointly estimate their

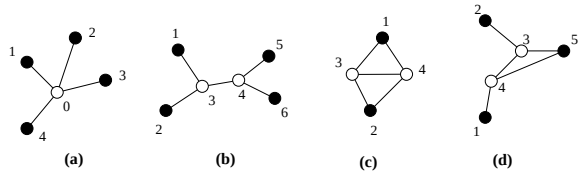

Figure 3: a) One-hop multilateration, b)Two-hop multilateration, c) Symmetric case, d) Each unknown has one independent reference

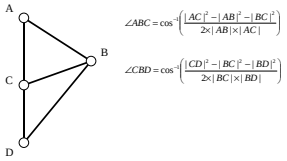

Figure 4: Detecting collinear configurations

locations. Like the one-hop case, each unknown node needs to be connected to at least three nodes, but these nodes are not required to be beacons. Instead, unknown nodes need to determine which of their neighbors have only one possible position solution and use them as reference points to determine if their position solution is unique. From this perspective, a position solution is tentatively unique if it has at least three neighbors that are either beacons or their solutions are tentatively unique. Figure 3b illustrates the most basic case. Nodes 3 and 4 are unknown and they are both connected to three nodes. Note that from the perspective of node 3, one of its links terminates to an unknown, node 4. Node 4 however, has two more outgoing links to beacons 5 and 6. If we assume that node 3 has a unique position solution, then node 4 also has a unique position solution. If however, node 4 has a unique position solution, then node 3 is also collaborative because it is connected to 3 collaborative nodes - 1,2 and 4. This condition is necessary but not sufficient to guarantee that there is only one possible node position estimate. Many symmetric topologies that meet the above requirement can yield more than one possible position estimate.

CONDITION 1. *To have a unique possible position solution, it is necessary that an unknown node be connected to at least three nodes that have unique possible positions.*

The fist symmetric case follows from the conditions of the single hop setup - the nodes with tentatively unique solutions used as references for an unknown should not lie in a straight line. If they lie in a straight line, then the unknown node will have two possible positions so the solution to the location estimate is not unique.

CONDITION 2. *It is necessary for an unknown node to use at least one reference point that is not collinear with the rest of its reference points.*

Although the positions of the reference points are not known, one can test for this condition using basic trigonometry. In figure 4, assuming that nodes A,C and D are known to have unique solutions, node B tries to establish if its position solution is unique. To do so node B computes the angles ABC, CBD and ABD. Using the angle ABD, node B can calculate the distance —AD—. If the computed distance AD is equal to the sum of distances AC and CD then the nodes are collinear [1] hence node B decides that its solution is not unique.

[1]Here we loosely use the term 'equal' for clarity and simplicity of the explanation, in practice we also need to consider the noise incurred by the distance measurement process

Another type of setup that can cause symmetry problems is shown in figure 3c. Nodes 3 and 4 both have 3 links to nodes with tentatively unique positions but the setup is symmetric since the two nodes can be swapped without any violation of the constraints imposed by the intra-node distance measurements. To avoid this situation where the whole network can be rotated over two pivot points (nodes 1 and 2 in this example) an additional condition is set.

CONDITION 3. *In each pair of unknown nodes that use the link to each other as a constraint, it is necessary that each node has at least one link that connects to a different node from the nodes used as references by the other node.*

The network in figure 3d is an example configuration that satisfies this property. Both unknown nodes 3 and 4 have at least one independent reference. Node 4 has beacon 1 and node 3 has beacon 2. The above three conditions are individually necessary but jointly sufficient to guarantee that if an unknown node is within two hops from at least three beacons then the unknown has a single possible position solution.

4.1.3 N-Hop Multilateration Requirements

To determine if nodes located within n hops from the beacons have unique solutions a similar set of criteria is applied. Starting from an unknown node we test if it has at least three neighbors with tentatively unique positions. If the node has three neighbors that do not already know if their solution is unique, then a recursive call is executed at each neighbor to determine if its position is unique. To meet the requirements of condition 3 each node used as an independent reference is marked as used. This prevents other nodes from subsequent recursive calls to re-use that node as an independent reference. At every step, each node checks if the criteria for condition 2 are also met.

4.2 Phase 2: Obtaining Initial Estimates

The initial estimates are obtained by applying the distance measurements as constraints on the x and y coordinates of the unknown nodes. Figure 5 shows how the distance measurement from two beacons A and B can be used to obtain the x coordinate bounds for the unknown node C. If the distance between an unknown and the beacon A is a then the x coordinates of node C are bounded by a to the left and to the right of the x coordinate of beacon A, $x_A - a$ and $x_A + a$. Similarly, beacon B which is two hops away from C, bounds the coordinates of C through the length of the minimum weight path to C, $b + c$, so the bounds for C's x-coordinates with respect to B are $x_B - (b + c)$ and $x_B + (b + c)$. By knowing this information C can determine that its x coordinate bounds with respect to beacons A and B are $x_B + (b + c)$ and $x_A - a$. This operation selects the

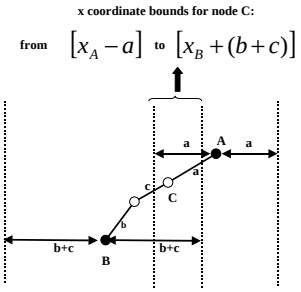

Figure 5: Initial Estimates

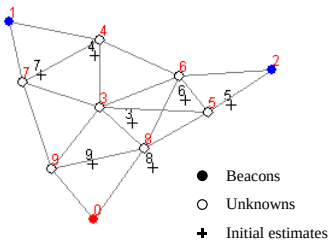

Figure 6: Initial estimates over multiple hops

tightest left hand side bound from and the tightest right hand side bound from each beacon. The same operation is applied on the y coordinates. The node then combines its bounds on the x and y coordinates, to obtain a bounding box of the region where the node lies. To obtain this bounding box, the locations of all the beacons are forwarded to all unknowns along a minimum weight path. This forwarding is the same idea as distance vector routing but using the measured distances instead of hops as weights.

The initial position estimate of a node is taken to be as the center of the bounding box. When these constraints are combined with the conditions for position uniqueness, they provide a good set of initial estimates for the position refinement phase. The resulting initial estimates for a 10 node network with 3 beacons is shown in figure 6. One challenge in this method is that the quality of the initial estimates suffers when the beacon-unknown topology is not convex. The unknown nodes that lie within a convex hull created by the beacons can produce good initial estimates. In some configurations, where some of the unknown nodes lie outside the convex hull the initial estimates are still sufficient. In our experiments with large networks we therefore assume that beacons surround the unknown nodes.

5. PHASE 3: POSITION REFINEMENT

In the third phase, the initial node positions are refined, using least-squares estimation. Our implementation uses a Kalman Filter [20], which provides the same location estimates as iterative least squares in a static network [6]. The Kalman Filter was chosen because of its ability to fuse measurements from multiple sensing modalities and to track the nodes after the localization process is complete. Position refinement can be implanted in one of two possible computation models, centralized or distributed that are described next.

5.1 Computing at a Central Node

Using the collaborative subtrees and the initial position estimates, the unknown node position estimates can be computed at a central point. The edges of the computation subtree give a well-determined or over-determined set of equations, which can be solved using non-linear optimization. The non-linear of equations for the network in figure 3b is shown in equations 1 [2]. As in the one hop case, the objective is to minimize the residuals between the measured distances between the nodes and the distances computed using the node location estimates.

$$\begin{aligned} f_{2,3} &= R_{2,3} - \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ f_{3,5} &= R_{3,5} - \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ f_{4,3} &= R_{4,3} - \sqrt{(ex_4 - ex_3)^2 + (ey_4 - ey_3)^2} \\ f_{4,5} &= R_{4,5} - \sqrt{(ex_4 - x_5)^2 + (ey_4 + y_5)^2} \\ f_{4,1} &= R_{4,1} - \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \end{aligned} \quad (1)$$

The $R_{i,j}$ quantities represent the measured distances between two nodes and the quantities under the square root indicate the estimated distances. $f_{i,j}$ represent the residual between the measured and estimated quantities. The objective function in 2 is to minimize the mean square error over all equations. The difference of this from its one hop counterpart is that in this process, unknown-unknown links are also used as constraints.

$$F(x_3, y_3, x_4, y_4) = min \sum f_{i,j}^2 \quad (2)$$

The solution to this optimization problem can be obtained using some of the standard least squares methods. In our implementation, this is done using a Kalman Filter.

5.1.1 *Kalman Filter Implementation*

A Kalman Filter consists of two phases, a time update phase and a measurement update phase. The former is a prediction in time (equations 3, and 4). This time prediction $\hat{x}_k^-$ is based on a known model of the system behavior, represented by matrix A. u_k is a zero mean gaussian random variable and B is the error covariance matrix for this random variable. P_k^- is the apriori estimate of the error covariance and Q is the process noise. The latter is an update of the current time estimate based on a measurement that was just obtained. K represents the Kalman Filter gain and it serves a weight to the residual of the filter. The residual is the difference between the measurement, (represented by z_k) and the predicted measurement $H\hat{x}_k^-$. $\hat{z}_k$ the distance between nodes, based on the current position estimate. Matrix H is the Jacobian of $\hat{z}_k$ with respect to the apriori estimates (found in $\hat{x}_k^-$) of the locations. Matrix R is the measurement noise covariance matrix. This contains the known noise covariance of the distance measurement system (i.e based on the characterization of our ultrasonic system we assume white gaussian noise with standard deviation = 20 mm). $\hat{x}_k$ is the new estimate obtained after the prediction and measurement are combined. This new prediction

[2]the prefix e in front of x, y denotes estimated coordinates, as opposed to known coordinates

measurement has a new error covariance matrix P. Matrix I stands for the identity matrix.

For the purposes of the collaborative multilateration, the network is assumed to be static. Since the positions of the nodes, do not change in time, the time update phase is not used. Based on this, the discussion in this section focuses on the second part of the Kalman Filter, the measurement update phase.

$$\hat{x}^- = A\hat{x}_{k-1} + Bu_k \tag{3}$$

$$P_k{}^- = AP_{k-1}A^T + Q \tag{4}$$

$$K = P_k{}^- H^T (HP_k{}^- H^T + R)^{-1} \tag{5}$$

$$\hat{x}_k = K_k(z_k - H\hat{k})^{-1} \tag{6}$$

$$P_k = (I - K_k H)P_k^- \tag{7}$$

To estimate the unknown locations, our algorithm proceeds as follows:

1. Set the vector to the initial estimates obtained in section 4.2
2. Evaluate equations 5, 6 and 7 - the measurement update phase
3. Evaluate the stopping criterion $\sqrt{(\hat{x}_k)^2 - (\hat{x}_k^-)^2} \leq \Delta$ where Δ is some predefined tolerance (0.01 in our experiments). If the criterion is met then the algorithm terminates and has the new position estimates. Otherwise,
4. Set the prediction $\hat{x}_k^-$ to the new estimate $\hat{x}_k$ and goto step 2.

The term Δ is used as an indicator of the gradient of the Kalman Filter and shows how far the process is from convergence. For illustrative purposes we provide the setup of the Kalman Filter in terms of figure 3c. The initial estimates (denoted by ex and ey) are placed in vector, $\hat{x}_k^-$, so $\hat{x}_k^- = [ex_3, ey_3, ex_4, ey_4]$. The ranging measurements are placed in vector z_k, $z_k = [R_{2,3}, R_{3,5}, R_{3,4}, R_{4,1}, R_{4,5}]$. Vector $\hat{z}_k$ contains the ranging distances based on the current estimates

$$\hat{z}_k^T = \begin{bmatrix} \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ \sqrt{(ex_3 - ex_4)^2 + (ey_3 - ey_4)^2} \\ \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \\ \sqrt{(ex_4 - x_5)^2 + (ey_4 - y_5)^2} \end{bmatrix}$$

Matrix H is the jacobian of $\hat{z}_k$ with respect to $\hat{x}_k^-$.

$$H = \begin{bmatrix} \frac{ex_3 - x_2}{\hat{z}(1)} & \frac{ey_3 - y_2}{\hat{z}_k(1)} & 0 & 0 \\ \frac{ex_3 - x_5}{\hat{z}_k(2)} & \frac{ey_3 - ey_5}{\hat{z}_k(2)} & 0 & 0 \\ \frac{ex_3 - ex_4}{\hat{z}_k(3)} & \frac{ey_3 - ey_4}{\hat{z}_k(3)} & \frac{ex_4 - ex_3}{\hat{z}_k(3)} & \frac{ey_4 - ey_3}{\hat{(z)}_k(3)} \\ \frac{ex_4 - x_1}{\hat{z}_k(4)} & \frac{ey_4 - y_1}{\hat{z}_k(4)} & 0 & 0 \\ \frac{ex_4 - x_5}{\hat{z}_k(5)} & \frac{ey_4 - y_5}{\hat{z}_k(5)} & 0 & 0 \end{bmatrix}$$

The above illustration shows how the matrix size and subsequently the amount of computation increases with the number of nodes. Each edge in the collaborative subtree contributes one entry in the measurement matrix z_k. In matrix H, the number of unknown nodes determines the number of columns and the number of edges determines the number of rows. Each beacon-unknown edge adds two entries to the row and each unknown-unknown edge adds four entries. The noise covariance matrices P_k^- and P_k are square matrices whose size depends on the number of unknown nodes and the measurement noise matrix R is a square matrix and it size determined by the number of edges in the collaborative subtree. These changes in matrix sizes dramatically increase the amount of computation that has to be performed. Furthermore, from our simulation experience, we noted that when the Kalman Filter has more variables to estimate, it takes more iterations to converge. Unfortunately, since the estimation process is an iterative process we cannot quantify the amount of computation required for the filter to converge analytically. Instead, we evaluate this empirically in our simulations by measuring the number of FLOPS MATLAB consumes [3] until the Kalman Filter converges. This description also shows that although node positions can be estimated accurately using this method, such computation cannot be performed using a low cost microcontroller available on the sensor nodes. To facilitate this type of computation, we have developed a distributed approximation where every node participates in the computation.

5.2 Computing at Every Node

In the distributed version, of our algorithm, computation is spatially distributed across the network and each unknown node is responsible for computing its own location estimate. This is achieved by performing local computation and communication with the neighboring nodes. This idea is similar to using a distributed Kalman Filter [13, 14], but also has some differences:

- The Kalman Filter executes in the context of a collaborative subtree and each node executes a one hop multilateration based on its distance measurements and the location information from its neighbors.
- Instead of using a decentralized Kalman Filter, we use an approximation in which nodes do not exchange covariance information is used. This conserves energy since it reduces communication, and simplifies implementation.
- The computation is driven by ad-hoc networking protocols.

The underlying principle of our distributed scheme is that after the completion of the first two phases, each node inside the computation tree computes an estimate of its location. Since most unknown nodes, are not directly connected to beacons, they use the initial estimates (obtained in section 4.2) of their neighbors as the reference points for estimating their locations. As soon as an unknown computes a new

[3]Note that the FLOPS measure obtained from MATLAB is used as a means for relative comparison of the computation cost between our centralized and distributed schemes. In our actual implementation, the Kalman Filters are implemented in C and they execute on the node processor.

estimate, it broadcasts this estimate to its neighbors, and the neighbors use it to update their own position estimates. This computation is repeated from node to node across the network until all the nodes reach the pre-specified tolerance Δ, $\left(\sqrt{(\hat{x}_k)^2 - \left(\hat{x}_k^-\right)^2} \leq \Delta\right)$. Figure 7 is a pictorial representation of the computation process. First node 4 computes its location estimate using beacons 1 and 5 and node 3 as references. Once node 4 broadcasts its update, node 3 recomputes its own estimate using beacons 2 and 5 and the new estimate received from node 4. Node 3 then broadcasts the new estimate and node 4 uses this to compute a new estimate that is more accurate than its previous estimate.

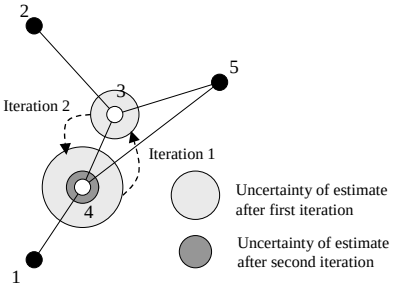

Figure 7: Initial estimates over multiple hops

If this process proceeds uncontrolled, then the nodes will converge at local minimal and erroneous estimates will be produced. Imagine a computation subtree with many unknown nodes (i.e 20). If two neighboring unknown nodes A and B that compute and broadcast their updates as soon as an update from each other is received, then their updating process will proceed faster than the remaining nodes in the computation subtree. This introduces a "local oscillation" in the computation that makes the nodes converge to their final estimates much faster but without complying with the global gradient, thus yielding erroneous estimates.

To prevent this problem, the multilaterations at each node are executed in a sequence across all the unknown members of the computation subtree. This sequence is repeated until the multilaterations of all the members of the computation subtree converge to a pre-specified tolerance. The in-sequence execution of the multilaterations inside the computation subtree establishes a gradient with respect to the global topology constraints at each node, thus enabling the node to compute its global optimum locally.

Figure 8 is an excerpt from our combined ns-2-matlab simulation which demonstrates the execution of distributed position refinement on a network of 34 nodes and 6 beacons [4]. The unknown nodes in the network have an average degree of 4, 15 meters [5] range and the 20 mm white gaussian noise in the distance measurement system. The x-axis shows the number of packets (estimate updates) transmitted by each node, the node ids are shown in the y-axis and the z-axis shows the error in millimeters. The values of the error

[4]For clarity and good visibility purposes the graph only shows how the process proceeds on the even numbered nodes.

[5]In our actual node testbed this is reduced to 5 meters to facilitate multiple hops in a lab setting

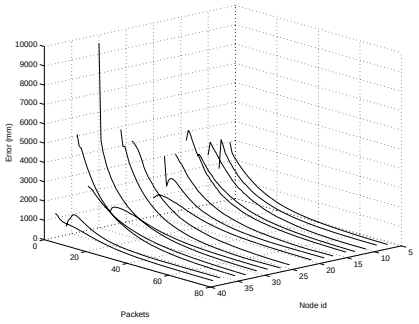

Figure 8: Progress of distributed computation

before any packets are transmitted, at packets = 0 on the x-axis, represent the errors of the initial estimates (obtained in section 4.2). As it can be seen form the figure, each node starts at different levels of error. After a few iterations of executing the node sequences in the computation subtree, a global gradient is established that drives the error down across the whole subtree. In the end, each node succeeds in estimating its node location with a 3-centimeter accuracy. Error accumulation is prevented by the global constraints of the collaborative subtree.

The order of nodes executing in the computation subtree sequence does not need to be specified but it needs to be consistent over successive iterations of the sequence. This entails that the order with which nodes compute their position updates has to be consistent across iterations. One possible way to initiate this distributed computation process is to use Distributed Depth First Search (DDFS). DDFS search is started at an arbitrary unknown node within the computation tree and it runs for two iterations. During the traversal of the subtree by DDFS, when each node is marked visited, the node it computes and broadcasts its location estimate and starts a timer. In the second iteration of DDFS, nodes compute and broadcast their locations. At this point the nodes also stop the previously set timer. The time between the two visits denotes the time interval at which each node should recompute and broadcast a new position update. The distributed algorithm for driving the distributed computation process is shown in figure 9. The DDFS algorithm for this example was obtained from [19].

Before finalizing this design decision we verified that the results obtained by this approximation do not compromise the overall quality of our estimates. We tested this by comparing the outputs of our centralized and distributed position refinement phases. The tests where run on a test suite of 42 networks of different sizes varying from 10 to 100 nodes. From this comparison we found out that the difference in the results between the centralized Kalman Filter the distributed approximation we used is very small. Figure 10 depicts the result of the comparison. The mean difference is 0.015 millimeters with a standard deviation of 0.54mm. Based on this result we verified that our distributed approximation of the Kalman Filter does not compromise our computation accuracy.

6. EVALUATION

We evaluate the performance of the collaborative multilateration through a set of simulations. The Kalman Filters are

Start the algorithm (initiator only!)
$visited_u := true$
for i:=1 : 2
  for $w \in Neigh_u$
    do begin send ⟨**bfs**⟩ to w; $status_u[w] := cal$ **end**

Upon receipt of ⟨**bfs**⟩ from v:
if not $vistited_u(i)$ **then**
  begin
    $visited_u(i)$:=$true$;$status_u(i)[v]$:=$father$
    compute new location estimate and broadcast it
    if i=0 $t1$:=time(); i:=$i+1$
    else $updatePeriod$ = time()-$t1$;
    timer.schedule($updadePeriod$)
  end
if $status_u(i)[v]$=$unused$ **then**
  begin send ⟨**bfs**⟩ to v;$status_u(i)[w]$:= ret **end**
else if there is a w with $status_u(i)[w]$:=$unused$ **then**
  begin send ⟨**bfs**⟩ to w $status_u(i)[w]$:=cal **end**
else if there is a w with $status_u(i)[w]$=$father$ **then**
  begin send ⟨**bfs**⟩ to w **end**
else (* initiator *) **stop**

Upon a timeout:
if $converged = false$ **or** $update\_needed = true$
  begin
    compute a new location estimate and broadcast it
  end
if $\sqrt{(\hat{x}_k)^2 - \left(\hat{x}_k^-\right)^2} \leq \Delta$
  begin $converged := true$ **end**
begin timer.schedule($updadePeriod$) **end**

Upon receipt of a updated location broadcast:
  if $converged = true$
    begin $update\_needed := true$ **end**

Figure 9: Distributed computation algorithm driven by DDFS

implemented in MATALB and they are linked into the ns-2 simulator using the MATLAB compiler and the MATLAB C++ library. The required protocols for communication are implemented inside ns-2. Using this simulation setup we carried out a series of experiments on a test suite of 200 different scenarios. Our simulation parameters are set to match the parameters of our experimental node. Each node has an effective radio range of 15 meters and each node can measure distances between its neighbors with the same range as the radio. The measurement noise is modeled as a zero mean gaussian random variable with 20 mm standard deviation.

The primary goal of our ns-2 implementation is to verify the correct operation of our distributed computation scheme over a wireless ad-hoc network. The distributed version of collaborative multilateration is implemented as a routing layer in ns-2. This routing layer also contains a minimal protocol that discovers the two-hop neighborhood of each node, and a forwarding mechanism for forwarding packets with beacons locations as described in section 4.2. These minimal routing requirements are sufficient to form collaborative subtrees in a fully distributed fashion, to obtain the initial estimates and to perform the distributed localization process. At the MAC layer we use a modified version of the IEEE 802.11 protocol with a 15-meter transmission range and an effective data rate of 20kbps.

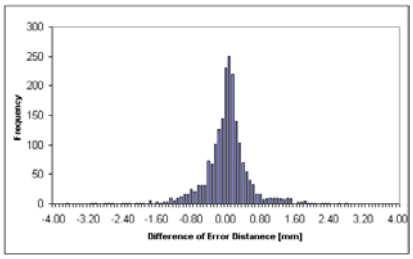

Figure 10: Comparison of centralized and distributed outputs

For the centralized case, DSR was used as the routing protocol, IEEE 802.11 as the MAC and the Kalman Filter was placed at the application layer.

6.1 Computation Cost Comparison

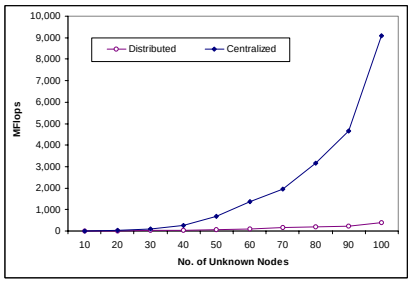

Figure 11: Computation cost comparison

Our first experiment compares the computation overhead between the distributed and centralized computation methods by recording the number of FLOPS consumed by MATLAB to compute the position estimates in each case. The scenarios used for this test have 6 beacons and varying number of unknowns ranging from 10 to 100 nodes. The number of unknowns was used in increments of 10, and the results show the average for 20 scenarios of each type. In all cases the network density is kept constant and each node has an average of 6 neighbors. The cumulative number of MFLOPS for the centralized and distributed implementation are shown in figure 11. From this result, we found that the computation overhead of the centralized computation model increases fast with the number of unknown nodes. In this particular test, the computation overhead appears to be cubic with the number of nodes. The distributed computation model on the other hand scales linearly with the number of nodes. The slope for the distributed case in figure 11 is 3.7MFLOPS, meaning that each node spends approximately 3.7MFLOPS to compute an estimate of its location. Similar trends were also observed when we simulated networks with increasing density.

From this comparison, we conclude that the distributed

computation model is a better choice for computing node locations. Even if computation is performed at a central point, the use of the distributed computation model will help to reduce computation latency. This is particularly the case for low cost processors like the AT91FR4081 processor on the Medusa MK-2 node.

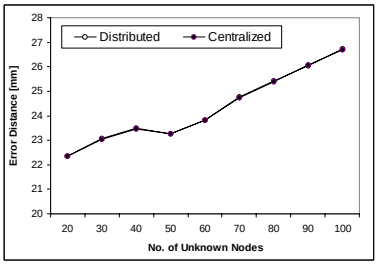

Figure 12: Localization error as unknown nodes increase

6.2 Localization Accuracy

To quantify the accuracy of the localization error we applied two tests. The first test evaluates accuracy of the localization process based on the measurement noise parameters of our ultrasonic distance measurement system. Figure 12 shows how the error in the estimates increases as the network scales. The ratio of beacons with respect to the unknowns is kept constant at 20 percent. The error in the estimates increases very gracefully as the network scales.

Figure 13 shows the cumulative error distribution over all scenarios used in this experiment for both the distributed and centralized cases. In both cases the average error was 27.7 millimeters with a standard deviation of 16 mm.

We observed a similar trend when we computed the error in the estimates at different variances in the ranging measurement error. This reflects how the collaborative multilateration would perform with less accurate distance measurement systems as the percentage of beacons decreases. The results of our simulation are shown in figure 14, and are based on different network sizes ranging from 10 to 100 unknown nodes, 20 networks for each set. In all cases the number of beacons is kept constant, 6 beacons were used in each network. In this particular test we also found that the performance of the distributed version degrades faster in networks of more than 100 nodes. This occurred in cases where the Kalman Filter sequence started by estimating the locations of nodes with accurate initial estimates using neighboring unknown nodes with less accurate initial estimates. One possible approach to mitigate this effect is to take the size of the bounding box (computed in section 4.2)into account. We expect that if the Kalman Filter computation sequence starts at the nodes with the largest bounding boxes then the problem mentioned above could be avoided and the convergence will be faster.

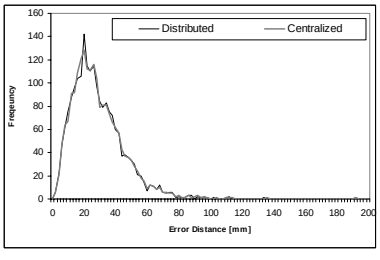

Figure 13: Estimated error distribution

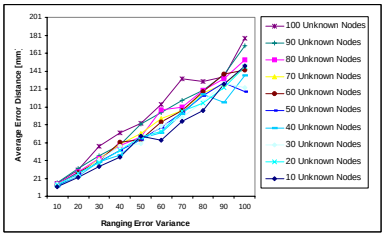

Figure 14: Errors in estimates at different measurement noise levels

6.3 Communication Cost and Convergence Latency Considerations

The convergence latency and communication aspects of collaborative multilateration are more difficult to evaluate because of their dependence on multiple system attributes. In the fully distributed case, convergence latency depends on the available communication bandwidth and the node processing power. Convergence latency also depends on the size of the computation tree. As the number of nodes increases, the sequence of Kalman Filter executions will take longer to complete and more iterations of the sequence are required. The communication pattern is uniform across all the nodes.

When computing at a single point in the network, the convergence latency of the collaborative multilateration is a function of the communication latency for transmitting the packets from each member of the computation subtree and back, and also depends on the power of the central processor. If the computation tree is large, the computation latency will be the dominant component. Evaluating the communication cost is more complex. In a clustered architecture, the communication cost depends on the cost of electing a cluster head and the routing cost for propagating the information back and forth from the cluster head.

To illustrate the communication trends, figure 15 shows a comparison of the communication cost for in the centralized and fully distributed cases, executed on a network of 44 unknown nodes and 6 beacons. The figure shows the total number of bytes transmitted by each node during the collaborative multilateration process. The average number of bytes transmitted is 4596 for the centralized scheme and 4485 for the distributed scheme. Although on average the communication cost is almost the same, the distributed scheme has an even distribution of transmitted bytes.

From a latency prespective, distributed collaborative multilateration offers a favorable tradeoff. This can be observed in figure 8. During the latest part of the process, a lot of computation and communication effort is spent to achieve a relatively small refinement of the position estimate. At this point, higher-level applications can have the option stopping the position refinement phase by adjusting the stopping cri-

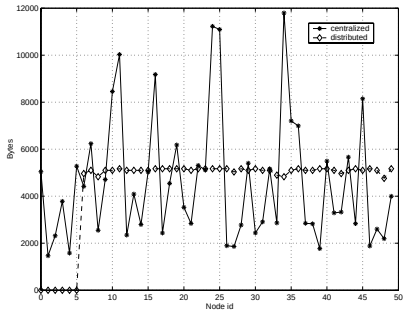

Figure 15: Communication cost on a 50 node network (6 beacons, 44 unknowns)

terion Δ in the interest of energy conservation.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have described collaborative multilateration for node localization problems. We have shown that using this three phase approach nodes that are indirectly connected to beacon nodes can estimate their locations with similar accuracies at the single hop multilateration. Also, with our distributed approach colonies of constrained sensor nodes can collectively solve a global optimization problem that an individual node cannot solve. The use of a global gradient for computing a global optimum locally reinforces a distributed computation model with other potential applications in sensor networks. In addition to the distributed computation model the collaborative multilateration appears to be an attractive choice for assisting infrastructure based localization systems to better handle obstructions. In this paper we have developed the computational part of collaborative multilateration. The remaining challenge is to study its feasibility with respect to the physical effects. To this end, as part of our future work we plan to study the interaction of our algorithms with the physical world using our sensor network testbed of Medusa MK-2 nodes.

Acknowledgments

This paper is based in part on research funded through NSF under grant number ANI-008577, and through DARPA SensIT and Rome Laboratory, Air Force Material Command, USAF, under agreement number F30602-99-1-0529. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF, DARPA, or Rome Laboratory, USAF.

8. REFERENCES

[1] S. Capkun, M. Hamdi, J. P. Hubaux, *GPS-Free Positioning in Mobile Ad-Hoc Networks*,

[2] awaii International Conference on System Sciences, HICCSS-34 Jan. 2001

[3] L. Doherty, L. El Ghaoui, K. S. J. Pister, *Convex Position Estimation in Wireless Sensor Networks*, Proceedings of Infocom 2001, Anchorage, AK, April 2001.

[4] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, *Next Century Challenges: Scalable Coordination in Sensor Networks*, Proceedings of the fifth annual international conference on Mobile computing and networking, Seattle, Washington, 1999, Pages: 263 - 270

[5] L. Girod and D. Estrin , *Robust range estimation using acoustic and multimodal sensing* Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), Maui, Hawaii, October 2001.

[6] E. Foxlin, M. Harrington, and G. Pfeiffer *Constellation(tm): A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications*, Proceedings of Siggraph 98, Orlando, FL, July 19 - 24, 1998

[7] J. Hightower and G. Boriello, *Location Systems for Ubiquitous Computing*, IEEE Computer, 34(8):57-66, Aug 2001

[8] A. Howard, M. J Mataric and G. S. Sukhatme, *Relaxation on a mesh: a formalism for generalized localization,* Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS01), pages 1055–1060, 2001

[9] Intersense Inc http://www.isense.com

[10] M. Mauve, J. Widmer and H. Hartenstein, *A Survey on Position Based Routing in Mobile Ad-hoc Networks*, IEEE Network Magazine, 15(6):30–39, November 2001

[11] S. Meguerdichian, F. Koushanfar, G. Qu, M. Potkonjak, *Exposure In Wireless Ad Hoc Sensor Networks*, International Conference on Mobile Computing and Networking (MobiCom '01),pp. 139-150, Rome, Italy, July 2001..

[12] D. Nicolescu and B. Nath, *Ad-Hoc Positioning System* Proceedings of IEEE GlobeCom, November 2001

[13] Roumeliotis, S.I.; Bekey, G.A. *Synergetic localization for groups of mobile robots*, Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, NSW, Australia, 12-15 Dec. 2000.) Piscataway, NJ, USA: IEEE, 2000. p.3477-82 vol.4. 5 vol. (lxiii+li+5229)

[14] BS Rao and HF Durrant-Whyte, *Fully Decentralized algorithm for multisensor Kalman filtering* IEE Proceedings-D, Vol. 138, No.5 September 1991

[15] C. Savarese, J. Rabay and K. Langendoen, *Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks* USENIX Technical Annual Conference, Monterey, CA, June 2002

[16] A. Savvides, C. C. Han and M. B. Srivastava *Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors,* Proceedings of the seventh annual international conference on Mobile computing and networking, Mobicom 2001, pp 166-179,Rome, Italy, July 2001

[17] A. Savvides and M. B. Srivastava, *A Distributed Computation Platform for Wireless Embedded Sensing* to appear in the proceedings of ICCD 2002, Freiburg, Germany

[18] M. B. Srivastava, R. Muntz and M. Potkonjak, *Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments*, Proceedings of the seventh annual international conference on Mobile computing and networking, Mobicom 2001, pp 132-139 Rome, Italy, July 2001

[19] G. Tel *Distributed Graph Exploration* Obtained from http://carol.wins.uva.nl/ de-laat/netwerken_college/explo.pdf

[20] G. Welch and G. Bishop *An Introduction to the Kalman Filter* Available from http://www.cs.unc.edu/ welch/kalman/kalmanIntro.html