

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

Andreas Savvides, Heemin Park and Mani B. Srivastava
{asavvide,hmpark,mbs}@ee.ucla.edu
Networked and Embedded Systems Lab, EE, UCLA

ABSTRACT

The recent advances in MEMS, embedded system 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 relies is the ability of microsensors to “collectively perform sensing and computation” [1]. In this paper, we study one such challenge, that of ad-hoc node localization. The distributed computation method described here enables ad-hoc deployed wireless microsensors to discover their physical locations with an accuracy of a few centimeters. Each sensor “senses” its physical distance from its neighbors and computes an accurate estimate of its physical location. Although individual microsensors have neither the information nor the computation and memory resources to individually perform the vital task of node localization, they can collectively solve this problem in a robust, distributed manner. The key to our approach is the *n-hop multilateration primitive*, an operation that can estimate the locations of nodes that are multiple hops away from known reference points. Our simulation results show that a 3-centimeter accuracy¹ position estimates with respect to the global topology can be computed locally with linear computation cost (3 to 4 MFLOPs per node), as opposed to the centralized computation cost which appears to be cubic with respect to the unknown nodes and without compromising the accuracy of the computed result.

1. INTRODUCTION

The marriage of ever tinier and cheaper embedded processors and wireless interfaces with micro-sensors based on micro-mechanical systems (MEMS) technology has led to the emergence of wireless sensor networks as a novel class of networked embedded systems. Many interesting and diverse applications for these systems are currently being explored. In indoor settings, sensor networks are already being used for condition-based maintenance of complex equipment in

¹Result based on average, details in figure 15

factories. In outdoor environments, these networks can monitor the natural habitats, remote ecosystems, endangered species, forest fires, and disaster sites. The ultimate goal of these networked sensors is to “coordinate amongst themselves to achieve a higher sensing task” [1]. Because of these requirements, sensor networks are expected to act as large distributed systems where computation is embedded inside the network. The nature of this problem imposes new challenges on how to coordinate and distribute computation on cheap, energy-constrained devices with limited memory and computation capabilities. UC Berkeley’s Rene mote for example has a 4MHz 8-bit micro controller from Atmel with 512 bytes of FLASH memory and 8 kilobytes of RAM, a low power radio and some sensors. With such limits, it is often the case that many nodes are required to perform collaborative computation on a sensed event. In this paper we examine one such problem, that of ad-hoc node localization.

Our work is motivated by the fact that many sensor network applications are based on the ad-hoc deployment of wireless sensors in dangerous or highly toxic environments to monitor different events and report their findings back to a remote operator. In such context, it is of utmost importance that each node is aware of its location. Such knowledge of node location is needed for reporting the geographic origins of sensed events and to perform other network level functions such as geo-routing [7] and network management. Furthermore, recent work in ad-hoc and sensor networks [9], [7] and [8] which build up on the assumption of known node locations reinforces the premise that multihop node localization is becoming an indispensable feature for applications in this new era of ubiquitous computing. For this purpose we have developed the *n-hop multilateration primitive* to estimate node locations over multiple hops. To estimate their locations, wireless devices use different technologies (i.e ultrasound, acoustics, laser, radio signal strength) to measure distances (perform ranging) to fixed landmarks. Using this ranging information and the known locations of landmarks, an optimization problem can be formulated and solved to obtain estimates of node physical locations. Besides ad-hoc deployment, the n-hop multilateration primitive would be useful in other setups of deeply embedded systems and context aware applications such as security applications and asset and personnel tracking inside buildings. In indoor infrastructure settings for instance, the n-hop multilateration primitive can assist localization in the presence of obstacles. In other situations, it can improve system robustness or simply complement existing methods to improve local-

ization accuracy. We refer to our method as a primitive because it can be used as a component in larger systems in many different configurations. The fully distributed form of the primitive on one hand, is sufficient to perform localization on a minimal system that uses a simple medium access control protocol does not require any routing protocol. The centralized computation model on the other hand, can be used as part of a greater system that is either centralized or locally centralized. In such case, the communication requirements can be easily adjusted to work in harmony with routing or any other communication and coordination protocols in the system.

1.1 What is presented in this paper

Our contribution is a method with which ad-hoc sensor nodes can coordinate among themselves to solve a global non-linear optimization problem of multihop localization, thus waiving the line of sight requirement with beacons. In this paper we present two computation models: centralized and distributed. The main goal of our discussion is to demonstrate the capability of our distributed approach to compute the global optimum locally. In the setup we investigate, the optimal position estimate for each node is the one computed from a global vantage point that considers all the available location information and all distance measurements between neighboring nodes. The proposed scheme addresses this challenge by using a very small number (3 or more) of initial landmarks (the beacon nodes) and by utilizing location and distance measurement information over multiple hops. Although none of the nodes has sufficient memory and computation resources to solve this problem individually, the nodes collectively solve this problem by performing local computation and communication. A distinct feature of our approach is the introduction of a three-phase process for estimating the locations of nodes that are physically located multiple hops away from the beacon nodes. In the first phase, nodes organize themselves in special configurations, the *computation subtrees*, which allow nodes to estimate their location over multiple hops while preventing error accumulation. During the second phase, the nodes use simple geometrical relationships to obtain a crude initial estimate of their locations. These estimates are later on refined in the third phase (position refinement phase) using a Kalman filter [18]. The third phase has two possible models of computation, centralized and distributed. The distributed computation model presents another novelty of our approach. By performing the Kalman Filter computation in the context a computation subtree, nodes in a multihop network establish a gradient with respect to the global topology constraints. This gradient enables each unknown node member of the computation subtree to estimate its globally optimal position estimate by performing computation locally. Our results demonstrate that n-hop multilateration makes it feasible for a group of small computationally constrained nodes to collectively solve the non-linear optimization problem of node localization, a problem that none of the nodes can solve individually.

This paper has two main goals 1) to describe the computation model of the n-hop multilateration primitive (centralized and distributed) and 2) to illuminate some of the inherent properties of the n-hop multilateration primitive by evaluating the different trends in computation, local-

ization accuracy, communication and latency. The results presented here are demonstrated by a set of simulations in ns-2 and MATLAB that compare two different implementations, one centralized and one distributed. Furthermore, we developed our discussion around our experimental wireless sensing device for localization, described in section 7.4. We use this device to demonstrate the viability of our approach by implementing this algorithm on a research oriented wireless sensing device we have developed for conducting experiments related to node localization.

2. RELATED WORK

Node localization has been the topic of active research and many systems have made their appearance in the past few years. Microsoft's RADAR system [16] and UCLA's GPS-less localization system [17] provide RF based localization. In both cases the unpredictability of wireless links hindered localization accuracy and enforces these systems to rely on preplanned infrastructures. The AT&T Laboratories Active Bats [14], Intersense's Constellation [15] and MIT's Crickets [6] are all infrastructure-based localization systems based on ultrasonic distance measurements. These three systems can perform localization with a few centimeter accuracy but they are all infrastructure based. So forms of ad-hoc localization made its appearance in the domain of mobile robotics. An example of the latest work in this field is Roumeliotis's Synergetic Localization Scheme [13] in which a group of mobile robots localize themselves by combining a set of inertial measurements and distance measurements using a distributed Kalman Filter. Like the method presented here this approach uses a distributed Kalman Filter but has a different setup. The robots start from the same starting point and they keep track of each other by exchanging measurement data and location measurements. This enables the robots to localize themselves with respect to each other. Another difference is that this approach does not consider computationally constrained systems in large numbers, which have to utilize measurements and location information over multiple hops.

The AHLoS system [2] is another ad-hoc localization system. It is based on an iterative multilateration in which nodes unknown nodes that estimate their locations become beacons that in turn help other nodes to localize themselves. A problem with this approach is error accumulation in the network. Also compared to the system presented here, iterative multilateration requires more nodes to be initially configured as beacons. The authors of the AHLoS paper also mention another method, collaborative multilateration but they do not provide any details of how this operation can be efficiently performed. We show how this operation can be done with very few beacons.

Finally, Doherty's [10] convex localization approach describes a method for localizing nodes in an ad-hoc network. This method is based on semi-definite programming and requires rigorous centralized computation so it is not always suitable for low-cost wireless devices. Our approach is different from the previously proposed approaches in the sense that it can estimate locations of nodes that are found multiple hops away from the beacons while avoiding error propagation. We limit error propagation by performing node computation in the context of a computation tree. Additionally, the

proposed approach is flexible, since it can run in many different configurations: fully distributed, locally centralized at a cluster head or purely centralized.

3. MOTIVATION AND ASSUMPTIONS

In many situations, wireless sensor networks 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. As mentioned earlier however, for many applications, sensor nodes are expected to know their physical locations. It is therefore essential to endow wireless sensor nodes with the capability to dynamically estimate their locations. The authors of the GPS-less localization system [17] argue that GPS cannot be used in all occasion due to power, form factor and the line of sight requirement. We share a similar viewpoint hence we seek to develop a system that does not require direct line of sight with beacons so that can be easily deployed in an ad-hoc manner, yet can achieve the same level of localization accuracy. To meet this goal, nodes with unknown node positions collaborate and share information with other unknown nodes over multiple hops to collectively estimate their locations. This problem has a mutual relationship to ad-hoc networking. From the localization perspective on one hand, information needs to efficiently propagate over multiple hops before the problem is solved. For wireless devices on the other hand, localization is an important feature for enabling context awareness.

3.1 Problem Statement

Given a network of sensor nodes where a) only a small fraction of the nodes is aware of their initial locations (the beacons), and b) sensor nodes within radio range of each other can also measure the distance between each other, estimate the locations of the remaining nodes (the unknowns).

3.2 Ranging Technologies

The only assumption that we make in our algorithms is that nodes can accurately measure distances between themselves and their neighbors. This assumption is supported readily available technologies, which can be found in many existing systems. The Medusa node in [2] performs some basic ultrasonic distance measurements has a 3-meter range and 2-centimeter accuracy. More advanced systems can achieve higher ranges and better resolutions. Hexamite [20] advertises ranging devices based on modulated ultrasound with a 20-meter range and 0.3 centimeter accuracy. InterSense [4] Soni discs are higher end devices that also have sub centimeter precision. All three devices use 40KHz ultrasonic transceivers and are based on time-of-arrival measurements. Medusa measures the time-of-arrival between RF and ultrasonic transmissions. In Hexamite's system the beacons transmit an ultrasonic scan that queries each of the devices. Distance is measured by recording the time taken for each device to respond. Soni Disks measure time-of-arrival by recording time between the reception infrared and ultrasonic distances. Girod's NLOS wideband acoustic ranging system can also deliver very accurate distance measurements. In outdoor environments this system has a 50-meter range. Reference [5] provides a detailed study of this system. Laser ranging technologies can make accurate distance measurements at longer ranges. The SICK's [11] laser range finders

have an effective range of 80 meters and 1 centimeter accuracies.

For indoor applications of pervasive computing we are in favor of ultrasonic distance measurements because it is silent, accurate, low cost and relatively low power. In our distance measurement experiments we have experimented two families of with 40KHz ultrasonic transducers, ceramic and polymer. Ceramic transducers have greater degree of directivity while ceramics can transmit in wider angles. Ceramic transducers produce a conical transmission pattern and polymer transducers for a cylindrical pattern. Our ceramic transducers have a 60-degree beam angle. The polymer transducers have a 360-degree horizontal coverage for the transmitters and 180-degree horizontal coverage for the receivers. With these transducers our bench top prototypes can achieve an effective range of 5 meters² with 0.5-centimeter measurement accuracy.

3.3 Establishing and Merging 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 $(A\Gamma, 0)$ and B has coordinates $(AB \sin \alpha, AB \cos \alpha)$. Such local coordinate systems can be established at different places inside the network. Later on, nodes coordinate and merge their coordinate systems using coordinate system transformations that place the nodes in a unified coordinate system. This operation was previously done in the field of mobile robotics and it is explained in detail in [3]. Our discussion begins on how to estimate node locations within such coordinate systems.

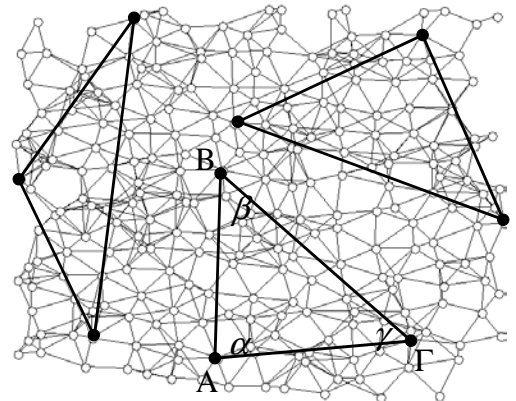


Figure 1: Establishing Local Coordinate Systems

²Ranges up to 20m are also achievable but they are not desirable since it would increase multipath effects and increase power consumption

3.4 Sensor Network Viewpoint on Localization

In a sensor network wireless microsensors collaborate with each other to achieve a common sensing task. Such a setup implies that each sensor node views its neighboring sensor nodes as trusted parties, therefore sharing location information does not violate privacy. Furthermore, we view sensor networks as large colonies of sensors that are locally proactive and globally reactive. From this perspective sensor nodes perform local coordination within their neighborhood and then use the locations to perform network scale functions. We consider localization as one of the local coordination functions of the network that takes place during the initial stages of network formation. Later on, during the network lifetime, the acquired knowledge of location is applied to perform tasks such as reporting the origins of events, target tracking, geo-routing and geo-casting, network coverage and network maintenance.

4. SOLUTION OUTLINE

In its simplest form the node localization problem is similar to GPS [12]. If three or more beacons surround an unknown node, the location of the unknown is estimated by minimizing the residuals between the measured and estimated distances between each beacon and the unknown node. In an ad-hoc network since one cannot guarantee that each unknown node will have at least three beacons as neighbors, a different method for estimating node positions needs to be applied. In this paper we extend this basic, single hop multilateration operation performed by GPS to a multihop operation. This operation enables nodes that are not directly connected to a node with known position to collaborate with other intermediate nodes with unknown locations situated between itself and the beacons so that they can jointly estimate their locations. One of the main challenges in this problem is to prevent error accumulation in the network. To prevent this we use least squares estimation to estimate all the unknown node positions simultaneously. In our solution the unknown nodes collaborate to set up a non-linear optimization problem that can then be solved either at a central node or in a fully distributed manner.

The n -hop multilateration takes place in three main phases depicted in figure 2. During the first phase, the nodes form a well constrained or over constrained configuration of unknowns and beacons, the computation subtrees (section 5.1). This configuration forms a system of at least n non-linear equations and n unknown variables to be determined. Computation subtrees also ensure that each unknown member of the computation subtree has a unique possible solution. The nodes that do not meet the criteria for computation subtrees cannot participate in this configuration. The position estimates for such nodes are determined later on in the process. 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. This is described in section 5.2. The initial estimates obtained in this phase are used to initialize the Kalman Filters of the third phase. The third phase utilizes a Kalman Filter to refine the initial position estimates and compute the final estimate of the node positions (section 6). Finally, the fourth phase uses the computed node estimates to refine the

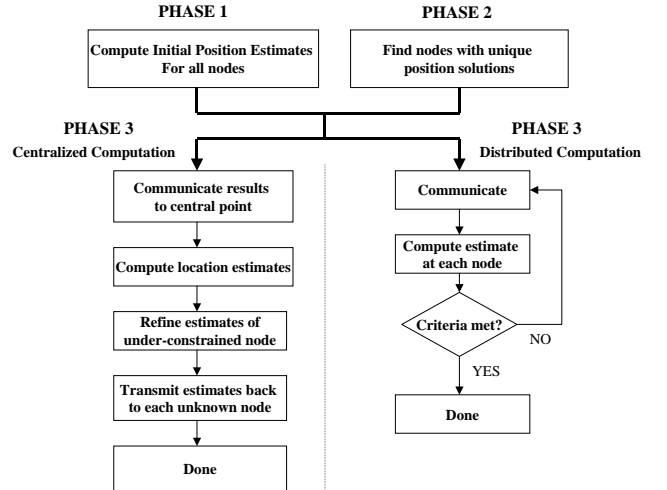


Figure 2: Location Estimation Phases

position estimates of nodes that could not participate in the computation tree configuration of the first phase.

The first and second phases are independent from each other in an actual network they can take place in parallel. Phases three and four can start as soon as the first two phases are completed and they 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.

5. INITIAL CONFIGURATION

Before attempting to solve an optimization problem using a Kalman Filter (a gradient descent method for least-squares), one must try to prevent two possible problems may arise: 1) the input problem may have multiple possible solutions and 2) the optimization process may converge at a local minimum. To compute position estimates successfully, one must ensure that the input configuration to the Kalman Filter has a unique solution, otherwise the Kalman Filter may compute a false positive, that is, it will compute an estimate that is numerically correct but it is the wrong estimate of a node's position. In addition to this, if the Kalman Filter is not properly initialized, it may converge to local minima thus a good set of initial position estimates is essential. In this section we describe how our approach avoids these two problems. In order to ensure that the solution is unique, we introduce the creation of computation trees. To avoid converging at local minima we obtain a set of initial estimates that is close to the final estimates using a simple geometrical relationship.

5.1 Phase 1: Computation 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 usually 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, we first determine a set of requirements that ensure the solution we are about to compute is unique. Computation subtrees have another desirable property that will become apparent when we discuss our distributed computation model in section 6.2. 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.

5.1.1 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.

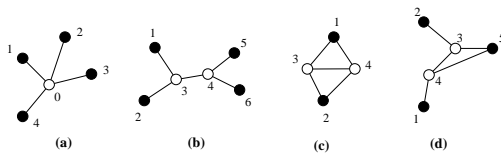


Figure 3: a) One-hop multilateration, b) Two-hop multilateration, c) Symmetric case, d) Each unknown has one independent reference

5.1.2 Two-Hop Multilateration Requirements

Using the one-hop multilateration requirements as a starting point, we establish the corresponding set of requirements for a two-hop multilateration. 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 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, we assume that a node 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 first 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 $\angle ABC$, $\angle CBD$ and $\angle ABD$. Using the angle $\angle ABD$, 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 collinear³ hence node B decides that its solution is not unique.

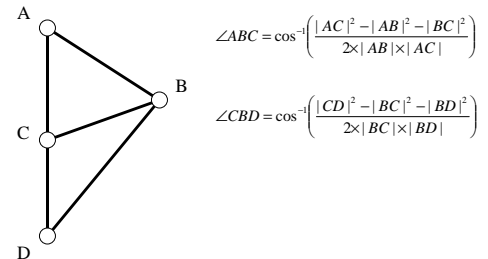


Figure 4: Detecting collinear configurations

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) we set an additional constraint.

CONDITION 3. *In each pair of unknown nodes that use the link to each other as a constraint, it is necessary that*

³Here 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

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.

5.1.3 N-Hop Multilateration Requirements

To determine if nodes located within n hops from the beacons have unique solutions we use a similar set of criteria as in the single hop case. Starting from the 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. We refer to these nodes as used nodes. This prevents other from subsequent recursive calls to use that node as an independent reference. At every step, each node checks if the criteria for condition 2 are also met. This recursive algorithm is given in figure 5. The algorithm initially starts at one unknown node and constructs a computation subtree by traversing the network in a depth first manner checking for the multilateration requirements. For illustrative purposes, we provide the static version of the algorithm⁴. At each node the algorithm assumes that its caller has a tentatively unique solution and, tries to find out whether the node currently visited has a tentatively unique solution. The algorithm terminates when a computation subtree is formed. In the distributed version of the algorithm, each unknown node only needs to know whether it is a part of the computation subtree, and which of its neighbors are part of the same subtree. Using this information, the node is ready to respond to a call to enter the third phase, position refinement.

5.2 Phase 2: Computing Initial Estimates

Before a Kalman Filter can be started it needs to be properly initialized. To prevent the filter from converging at a local minimum, we apply a simple geometrical relationship based on the accurate ranging measurements taken by the sensors and the knowledge of beacon locations to compute an initial position estimate for each unknown node. The initial estimates are obtained by applying the distance measurements as constraints on the x and y coordinates of the unknown nodes. Figure 6 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

⁴In practice we apply a distributed algorithm that has multiple starting points and can create a computation tree in a fully distributed manner using by only considering the 2-hop neighborhood of each node. This algorithm is omitted due to space restrictions

Inputs: node id, node id of the caller and a boolean flag that specifies if the caller node is the node that initiated the computation tree search.
Output: if success full a list of edges that make up the computation subtree, otherwise, it returns NULL

```

list isUnique(u, caller, initiator)
begin
if we are at the initiator
begin U := 0 ; m := 0 end
else begin U := 1; m := 1 end
for each beacon neighbor b begin
if b is marked begin
check if we can unmark
if b is not the only beacon neighbor
of the node that marked it begin
mark b as visited by u;
decrement U for the node that previously
marked b; U := U + 1
end
else begin
if m < 2 begin
m := m + 1; U := U + 1;
end
end
end
else begin
edgeList ← edge(b, u); U := U + 1
mark b as visited by u
end
end
for each unknown marked neighbor z begin
if m < 2 begin
edgeList ← edge(u, z)
m := m + 1 ; U := U + 1
end
if U = 3 return edgeList
end
for each unknown unmarked neighbor x
if (edgeListB := isUnique(x, u, false)) not NULL
edgeList ← edge(x, u)
edgeList ← edgeListB
U := U + 1
if U = 3 return edgeList
end
end
return NULL
end

```

Figure 5: Forming computation subtrees

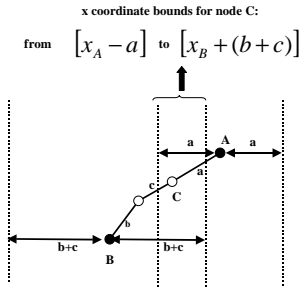


Figure 6: Initial Estimates

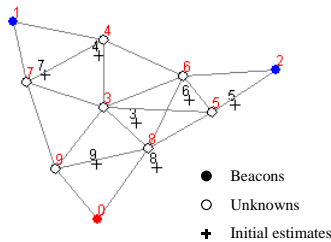


Figure 7: Initial estimates over multiple hops

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 tightest left hand side bound from and the tightest right hand side bound from each beacon. The same idea is applied on the y coordinates. The node then combines its bounds on the x and y coordinates, to construct 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 Kalman Filter used in the next phase. The resulting initial estimates for a 10 node network with 3 beacons is shown in figure 7. 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.

6. PHASE 3: POSITION REFINEMENT

In the third phase, the initial node positions are refined, using least-squares estimation. Given that the ranging measurements are noisy this method will give the optimal posi-

tion estimates in the least-square sense. In our implementation we decided to use a Kalman Filter, which is a recursive solution for least-squares estimation. A Kalman Filter was chosen mostly because of its flexibility and expandability. A Kalman Filter provides a recursive solution to least squares estimation. The main properties of the Kalman Filter that made it an attractive choice are its ability to fuse information from multiple sensing modalities and time prediction based on the previous system state and a system model. These properties are useful for localization, tracking and for other sensing tasks so the Kalman Filter is a valuable component in our infrastructure. The other property of the Kalman Filter that makes it an attractive choice for our application is that it can also track the nodes after the localization process is complete. As described in section 7.4, our experimental node is equipped with sensors that can track the node by dead reckoning once the node's position estimate is known.

The position refinement phase has two possible formulations, centralized and distributed. The first one computes many unknown estimates at a single point (i.e a central node a locally central cluster head). The second is a distributed approximation of the central implementation in which each node iteratively refines its position with local computation and communication.

6.1 Computing at a Central Node

Using the computation subtrees and the initial position estimates, we can compute the initial node estimates 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 1⁵. As in the one hop case, the objective is to minimize the residual between the measured distances between the nodes and the computed estimates, which are the result of the estimation process.

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

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.

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

⁵the prefix e in front of x, y denotes estimated coordinates, as opposed to known coordinates

The solution of these equations using a Kalman Filter is described in the following subsection.

6.1.1 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 a priori 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 a priori 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 measurement has a new error covariance matrix P . Matrix I stands for the identity matrix. For a good introduction discussion of Kalman Filters we refer the reader to [18] and for a more in-depth discussion we recommend [19].

For the purposes of the n-hop multilateration primitive, we assume that the network is static. Since the positions of the nodes, do not change in time, the time update phase is not used. In fact, the result of the Kalman Filter would be the same as the result of iterative least squares [15]. Based on this we focus our discussion on the second part of the Kalman Filter, the measurement update phase.

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

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

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

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

$$P_k = (I - K_k H)P_k^- \quad (7)$$

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

1. Set the vector to the initial estimates obtained in section 5.2

2. Evaluate equations 5, 6 and 7 - the measurement update phase
3. Evaluate the convergence 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 restart from 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 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^- .

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

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 its 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 ⁶ until the Kalman Filter converges. This

⁶Note that the FLOPS measure obtained from MATLAB

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 to this method - distributed n-hop multilateration.

6.2 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. This is achieved by performing local computation and communication with the neighboring nodes. This idea of using a decentralized Kalman Filter to distributed computation across nodes is not new. Durrant Whyte presented a decentralized Kalman Filter in [22]. Roumeliotis [13] is also applying a distributed Kalman Filter in his synergetic localization scheme. The novelties of our scheme that makes it different from the previous approaches are:

- The Kalman Filter executes in the context of a computation subtree. As we will describe in this section, this enables every unknown node to obtain its global optimal position estimate locally.
- We use an approximation of the Kalman Filter instead of the full form in order to conserve computation and we provide a scalability comparison of centralized vs. distributed in terms of computation overhead. This is crucial for the viability of our approach since we want to run this on resource-constrained devices.
- The computation is applied over a multihop network thus it is inherently related the concepts of ad-hoc networks, in fact, the estimation itself is a least-squares-communication hybrid.

The underlying principle of our distributed scheme is that after the completion of phases one and two, 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 5.2) of their neighbors as the reference points for estimating their locations. As soon as an unknown computes a new 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 , $\left(\sqrt{(\hat{x}_k)^2 - (\hat{x}_k^-)^2} \leq \Delta\right)$. Figure 8 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.

is used 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.

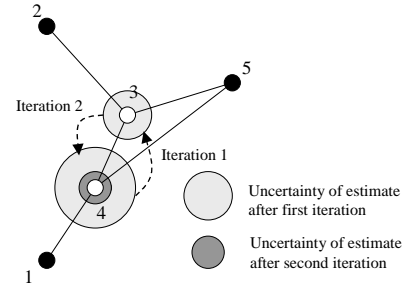


Figure 8: Initial estimates over multiple hops

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. Because of this the nodes will produce incorrect position estimates.

To prevent this problem, the Kalman Filters at each node are executed in a sequence across all the unknown members of the computation subtree. This sequence is repeated until the Kalman Filters of all the members of the computation subtree converge to a pre-specified tolerance. The in-sequence execution of the Kalman Filters 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.

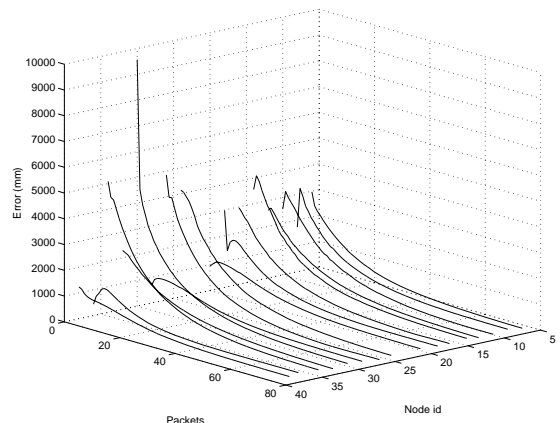


Figure 9: Progress of distributed computation

Figure 9 is an excerpt from our combined ns-2-matlab simulation. As we will describe in the next section, we developed our simulations in ns-2. To be able to measure FLOPS, we implemented the Kalman Filters in MATLAB and then using the MATLAB compiler and the MATLAB C++ libraries, we translated the filters to C++ so we can use them

at the routing layer in ns-2. The figure demonstrates the execution of distributed position refinement on a network of 34 nodes and 6 beacons⁷. The unknown nodes in the network have an average degree of 4, 15 meters 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 before any packets are transmitted, at packets = 0 on the x-axis, represent the errors of the initial estimates (obtained in section 5.2). As it can be seen from 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. The fact that error accumulation is eliminated is due the properties of the computation subtree. As explained in 5.1, computation subtrees constitute a constrained configuration of nodes and beacons. This constrained configuration prevents the estimates from going in the wrong direction.

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 by using Distributed Depth First Search (DDFS). DDFS search is started at an arbitrary unknown node within the computation tree and its run 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 location. 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 10. Figure 9 shows how the computation proceeds on a network of 6 beacons and 34 unknowns. The DDFS algorithm for this example was obtained from [21].

Finally, note that the Kalman Filter used in our experiments is an approximation of the distributed Kalman Filter. The results of the two methods can be made equivalent if nodes also exchange and update their corresponding covariance matrices, as it is done in [13]. In our implementation, we do not communicate the covariance matrices to the neighboring nodes, to conserve computation and communication. 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 were 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 dis-

⁷For clarity and good visibility purposes the graph only shows how the process proceeds on the even numbered nodes.

```

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

Upon receipt of ⟨bfs⟩ from v:
if not visited_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(updatePeriod)
  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  $\sqrt{(\hat{x}_k)^2 - (\hat{x}_k^-)^2} > \Delta$ 
  begin converged := true end
begin timer.schedule(updatePeriod) end

Upon receipt of a updated location broadcast:
if converged = true
  begin update_needed := true end

```

Figure 10: Distributed computation algorithm driven by DDFS

tributed approximation we used is very small. Figure 11 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.

7. EVALUATION

We evaluate the performance of the n-hop multilateration primitive through a set of simulations. The Kalman Filters are 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

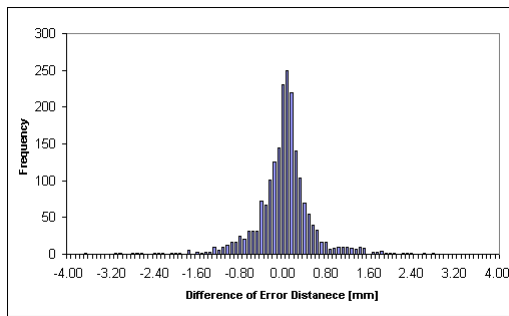


Figure 11: Comparison of centralized and distributed outputs

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 algorithm version of the n -hop 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 5.2. These minimal routing requirements are sufficient to form computation 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.

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

7.1 Computation Cost Comparison

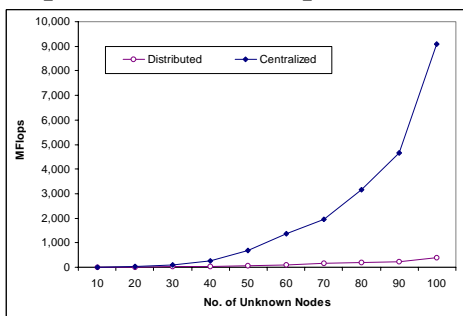


Figure 12: Computation cost comparison

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 re-

sults 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 12. 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 12 is 3.7MFLOPS, meaning that each node spends approximately 3.7MFLOPS to compute an estimate of its location. Network density has a similar trend on computation overhead. As the network density increases, the amount of computation required by the centralized model increases exponentially. Figure 13 shows an example of this case. In this example, the number of unknowns and beacons in the network is kept constant. Changing the area of the network varies the network density. The amount of computation at each node in the centralized case is much higher with respect to the number of unknown nodes n . In the distributed case, density only increases the number of edges at each node hence the computation scales linearly with the number of nodes.

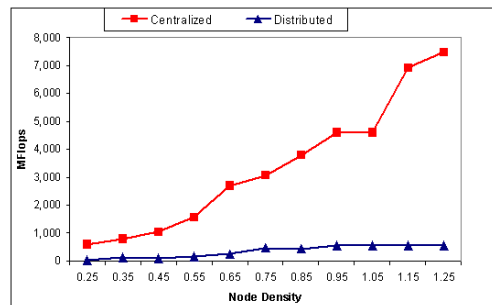


Figure 13: Computation vs. network density

From this comparison we conclude that the distributed computation model is a better choice for computing node locations. Even if computation is performed a central point, using the distributed model to compute locations will help to reduce computation latency. This is especially important in the case of low cost processors that do not have hardware support for floating point operations. The AT91FR4081 microcontroller of our localization node is one such example. The ARM7TDMI core of our microcontroller has a fixed-point multiplier, so we perform the floating-point operations of our Kalman Filter using a software floating-point library. As a second level of optimization, we are now in the process of implementing the Kalman Filter using fixed-point arithmetic on our experimental node.

7.2 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 14 shows how the error in the estimates increases as the network scales. The ratio of beacons with respect to the unknowns

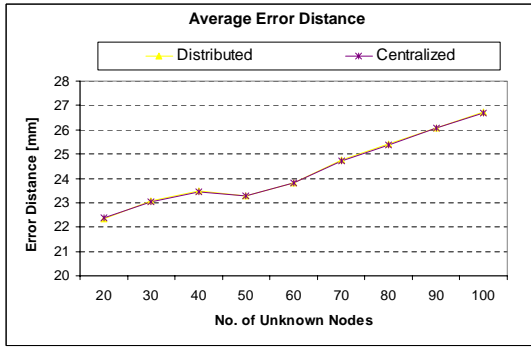


Figure 14: Quality of localizations a unknown nodes increase

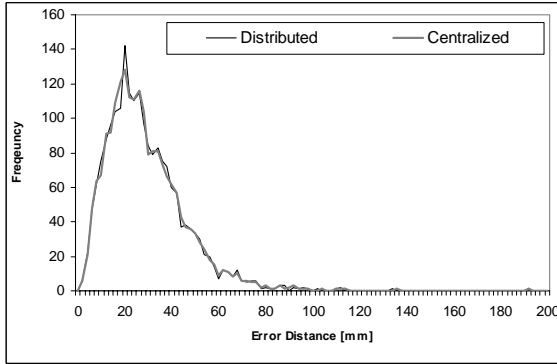


Figure 15: Estimate error distribution

is kept constant at 20 percent. The error in the estimates increases very gracefully as the network scales.

Figure 15 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 measurement noise levels. This reflects how the n-hop multilateration would perform with less accurate distance measurement systems as the percentage of beacons decreases. The results of our simulation are shown in figure 16, and a based on different network sizes ranging from 10 unknown nodes to 100 unknown nodes, 20 networks with 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 5.2) 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. We plan to explore this as part of our

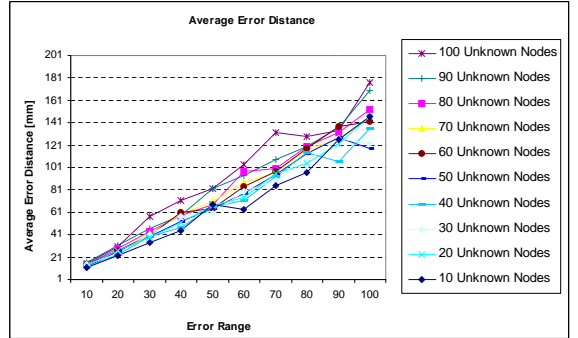


Figure 16: Errors in estimates at different measurement noise levels

7.3 Communication Cost and Convergence Latency

The convergence latency and communication aspects of the n-hop 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 n-hop multilateration primitive 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. As an example a network of 100 unknowns and 6 beacons takes approximately 5 to 7 minutes to converge on our Pentium III 700 MHz workstation. 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 cost to propagate the information back and forth from the cluster head.

Figure 17 is a comparison of the communication cost of the n-hop multilateration process in the centralized and distributed case, 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 n-hop 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. Additionally, a favorable trade of the distributed version on the n-hop multilateration is shown in figure 18. The average convergence latency for two network sizes of 20 and 40 nodes is shown. During the latest part

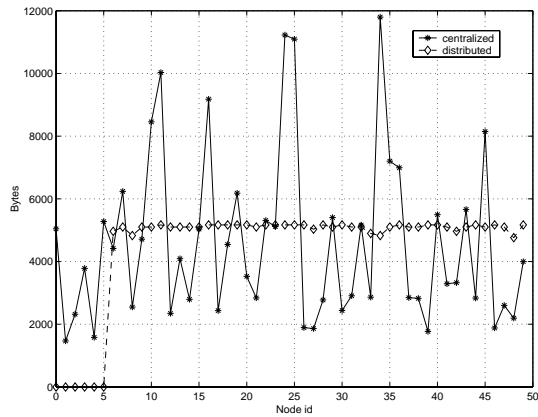


Figure 17: Communication cost on a 50 node network (6 beacons, 44 unknowns)

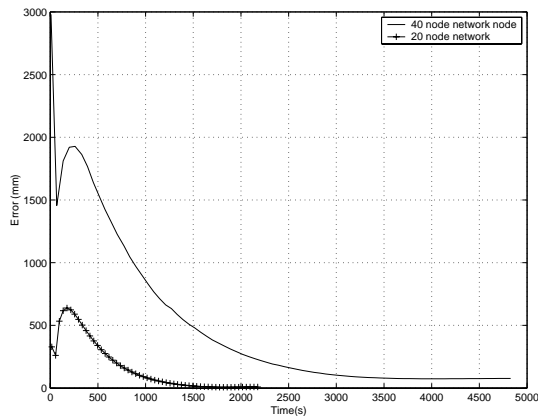


Figure 18: Convergence latency of distributed n-hop multilateration

of the process, a lot of computation is spent to achieve a relatively small refinement of the position estimate. At this point, higher-level applications should have the option to decide when to stop the position refinement phase by adjusting the convergence criterion Δ . Finally, we note that this process is robust since the position refinement phase can continue even if some of the nodes fail during the process.

7.4 Localization Experimental Node

To experiment with different types of node localization problems we developed a wireless sensing device that is geared towards experimentation. The wireless communication front end of this device is similar to UC Berkeley's Rene and MICA motes ?? and UCLA's Medusa node described in [2]. It features a low power radio from RF Monolithics and a 4MHz ATmega103L microcontroller from ATMEL. The design decisions for the remaining part of the node are however different from the nodes mentioned above. In addition to the ATmega103L microcontroller, the node has a 40MHz ARM THUMB microprocessor, also from Atmel. This serves as



Figure 19: Our experimental node

a more powerful computation co-processor⁸. This processor has 1MB of FLASH memory and 136KB of RAM, which is sufficient to run several existing embedded operating systems such as Red Hat's eCos and uLinux. This provides a versatile programming environment for studying the interaction of our localization algorithms with existing protocols currently used by a wide variety of wireless devices.

In addition to the more powerful processor, our research node has 2 pushbuttons that can be used as user interfaces, and an RS-485 transceiver. This provides a 2Mbps bus to facilitate data collection in experiments or to server as a gateway to the infrastructure. The RS-485 link also enables the formation of node arrays (up to 36 nodes can be daisy-chained together) help with experimentation and data collection in various settings. Our node supports several power saving modes and contains three current monitors that can monitor the power consumption of the different component of the node. The main board also carries a MEMs accelerometer that detects if the node has moved during the localization process. Later on we will also use the accelerometer as another sensing modality in our Kalman Filter to track the nodes once the localization process is completed. The distance measurement front end comes on an accessory board that carries an array of 40KHz ultrasonic sensors, and a magnetometer that can serve as a compass. The ultrasonic distance measurement system has an effective range of 5 meters and an accuracy of 0.5 centimeters. The node is power by a single 550mAh Lithium-Ion rechargeable battery. The circuit boards were customized to fit in a 2 x 3 x 1.25 epoxy enclosure shown in the figure.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have described, the n-hop multilateration primitive for node localization problems. We have shown that using this three phase approach nodes that are indirectly connected to nodes with node locations 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 op-

⁸At design time we considered that Moore's law will prevail so we decided to go with a bigger processor that has more memory and a multiplier. This proved to be a good design choice since the new replacement part for the ATmega103L, ATmega128L does have a multiplier

timization 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. Our simulations have shown that although the computation cost scales linearly with the number of nodes, the increasing amount of communication suggests that the n-hop multilateration primitive should be applied hierarchically in large networks. As we have explained at the beginning of this paper, a good starting point for such a hierarchical is to establish several coordinate systems locally using long-range distance measurements and then perform the required coordinate transformations to map the entire network to a single coordinate system. In addition to the distributed computation model the n-hop multilateration primitive 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 the n-hop 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 through the localization node that we have developed.

9. REFERENCES

- [1] 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
- [2] A. Savvides, C. C. Han and M. B. Srivastava *Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors*, Proceedings of the fifth annual international conference on Mobile computing and networking, Mobicom 2001, Rome, Italy, July 2001, pages 166-179
- [3] 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
- [4] Intersense Inc <http://www.isense.com>
- [5] Lewis Girod and Deborah 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] N. Priyantha, A Chakraborty, and H. Balakrishnan, *The Cricket Location Support System*, Proceedings of International Conference on Mobile Computing and Networking, pp. 32-43, August 6-11, 2000 Boston, MA
- [7] J. Li, J Jannotti, D.S J. DeCouto, D. R. Karger and R. Morris, *A Scalable Location Service for Geographic Ad-Hoc Routing*, Proceedings of International Conference on Mobile Communications and Networking, August 11-16, Boston, Massachusetts
- [8] Y. Xu, J. Heidemann and D. Estrin, *Geography-informed Energy Conservation for Ad Hoc Routing*, Proceedings of the ACM SIGMOBILE 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, July 2001
- [9] 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..
- [10] L. Doherty, L. El Ghaoui, K. S. J. Pister, *Convex Position Estimation in Wireless Sensor Networks*, Proceedings of Infocom 2001, Anchorage, AK, April 2001.
- [11] SICK <http://www.sick.de>
- [12] E. Kaplan, *Understanding GPS Principles and Applications* Artech House, 1996
- [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] R. Wand, A. Hopper, V. Falcao and J. Gibbons, *The Active Bat Location System*, ACM Transactions on Information Systems, January 10 1992, pages 91-102
- [15] 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
- [16] P. Bahl, V. Padmanabhan, *RADAR: An In-Building RF-based User Location and Tracking System*, Proceeding of INFOCOM 2000 Tel Aviv, Israel, March 2000, p775-84, vol 2
- [17] N. Bulusu, J. Heidemann and D. Estrin, *GPS-less Low Cost Outdoor Localization For Very Small Devices*, IEEE Personal Communications Magazine, Special Issue on Networking the Physical World, August 2000
- [18] G. Welch and G. Bishop *An Introduction to the Kalman Filter* Available from <http://www.cs.unc.edu/welch/kalman/kalmanIntro.html>
- [19] R. Brown and P. Hwang *Introduction to Signals and Applied Kalman Filtering* Wiley Press 1997
- [20] Hexamite <http://www.hexamite.com>
- [21] G. Tel *Distributed Graph Exploration* Obtained from http://carol.wins.uva.nl/de-laet/netwerken_college/explo.pdf
- [22] BS Rao and HF Durrant-Whyte, *Fully Decentralized algorithm for multisensor Kalman filtering* IEE Proceedings-D, Vol. 138, No.5 September 1991
- [23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister *System Architecture Directions for Networked Sensors*, Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS -IX), pages 93-104, Cmbridge, MA, USA, Novemeber 2000