

Localization in Ad Hoc and Sensor Wireless Networks with Bounded Errors

Mark Terwilliger^{1,2}, Collette Coullard², and Ajay Gupta¹

¹Computer Science Department, Western Michigan University, Kalamazoo, MI, USA

²Computer Science Department, Lake Superior State University, Sault Ste. Marie, MI, USA

Abstract: With the proliferation of wireless networks and mobile computing devices, providing location-aware technology and services to new applications has become important for developers. Our main contribution is an efficient location discovery algorithm that bounds the localization error. Providing an efficient localization technique is critical in resource-constrained environments that include mobile devices and wireless networked sensors. Applicable in centralized and distributed environments, our algorithm, based on finding the smallest circle enclosing the intersection of n disks, runs in $O(n^2)$ time. We then extend our work to the problem of finding the smallest disk that includes the set of points common to n disks and excluded from the interiors of m other disks. Finally, we show performance results from the implementation of our algorithms in which, under some conditions, localization estimates for 500 nodes in a 500x500 ft region can be found with a mean error of one foot and a two-foot error bound.

Keywords: Localization, error bound, sensor network, position, distance estimates

1 Introduction

Advancements in low-power electronic devices integrated with wireless communication capabilities and sensors have opened up an exciting new field in computer science. Wireless sensor networks (WSN) can be developed at a relatively low-cost and can be deployed in a variety of different settings. A WSN is typically formed by deploying many sensor nodes in an ad hoc manner. These nodes sense physical characteristics of the world. The sensors could be measuring a variety of properties, including temperature, acoustics, light, and pollution. Base stations are responsible for sending queries to and collecting data from the sensor nodes. Some of the main characteristics of a networked sensor include: (1) small physical size, (2) low power consumption, (3) limited processing power, (4) short-range communications, and (5) a small amount of storage.

Localization is the process of determining the positions of nodes in an ad hoc network. It is an important problem that has attracted much attention in this decade [1]. Providing robust localization services remains a fundamental research challenge facing the entire WSN development community [2].

A common practice when locating an object is to use estimated distances to known positions, or *anchors*. Suppose two objects are actually separated by distance d . If the estimate to a position is given by d' and the possible error for this distance is $\pm\epsilon$,

then the object must be located within a disk with radius $d'+\epsilon$ as shown in Fig. 1(i). If we remove the inside disk with radius $d'-\epsilon$, we can further say that the object must be located in a washer as illustrated in Fig. 1(ii).

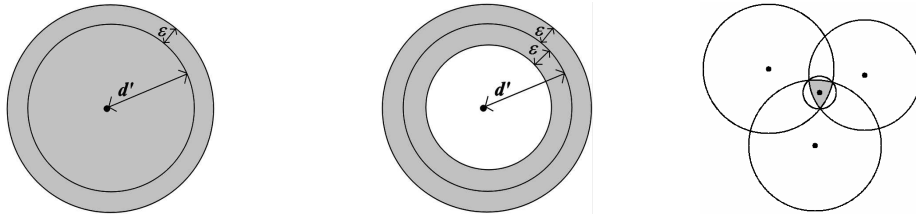


Fig. 1. (i) A disk, (ii) A washer, (iii) smallest enclosing circle covering intersection of disks.

If we have distance bounds $d'+\epsilon$ from n known positions to an object, we know that the object must be located in the intersection of those n disks. We find the smallest circle enclosing this intersection. By choosing the center of this circle as the location estimate, we guarantee that the localization error cannot be larger than the circle's radius. See Fig. 1(iii).

In this paper, we provide algorithms that find the smallest enclosing circle for both the intersection of disks, as well as washers. In addition to providing each step of our location discovery algorithms and we analyze their efficiency. We first provide an $O(n^3)$ -time complexity algorithm for finding a location estimate along with a corresponding error bound. We then use a novel approach to improve our technique to $O(n^2)$ for the intersection of n disks. Our technique has the following advantages.

- 1) For every location estimate that is made, a bound on the maximum possible error is also provided.
- 2) The network *just* has to be connected. As opposed to many localization techniques, we have no restrictions on the number of neighbors each node must have to other nodes or anchors.
- 3) The algorithms work in both a centralized, as well as, a distributed environment.
- 4) This technique applies to both ad hoc mobile computing and sensor networks.
- 5) The algorithms are independent of the ranging technique used by nodes to estimate distances between neighbors.

The remainder of this paper is organized as follows: In Section 2, we define specifically the problems that we are attempting to solve. We describe related work in Section 3. We provide our solution to the Disk Problem in Section 4 and the Washer Problem in Section 5. Performance results are given in Section 6. We make final conclusions and talk about future work in Section 7.

2 Problem Description

The standard localization problem can be defined as follows: "Reconstruct the positions of all the nodes in a network given the relative pairwise distances among all the nodes that are within some radius r of each other." While we are given 1-dimensional measures of the relative distances, we are required to compute the

positions either in a 2-dimensional or a 3-dimensional space, which makes the problem interesting and challenging. Throughout this paper, without loss in generality, we target our algorithms for the resource constrained and energy-critical WSNs, however, our solutions are applicable to more general wireless ad-hoc networks.

The localization problem is even more important in wireless sensor networks for the following reasons:

1. Many WSN protocols and applications simply assume that all nodes in the system are location-aware.
2. If a sensor is reporting a critical event or data, we must know the location of that sensor.
3. If a WSN is using a geographical routing technique, all of the nodes must be aware of their location.

Given known exact distances between neighbors, the localization problem has been shown to be NP-hard [3]. An added challenge is the fact that in practice, the exact distances between pairs of sensor nodes are not known. Instead, estimates are used to approximate the distances.

Therefore, there are two sources of errors in localization techniques – errors in relative pairwise distance estimates and even if exact distances are known, errors in computing the global coordinates. One must try to minimize these for heuristic techniques to be effective.

This paper addresses the problem of finding the location of *all* the nodes in a network given the location of a small subset of the nodes and estimates of the relative distances between pairs of nodes (i.e., relative distances are not known exactly). Our solution is in the form of a point estimate along with an error bound.

The first problem that we address is called the **Disk Problem**: Given centers and radii of n distinct disks, the goal is to find the center and radius of the smallest disk $D^* = [x^*, y^*, r^*]$ that includes the intersection of the n input disks.

The second problem that we address is the **Washer Problem**. This is similar to the Disk Problem, but the goal is to now find the center and radius of the smallest disk $D^* = [x^*, y^*, r^*]$ that includes the set of points included in all of the input washers.

3 Related Work

Most localization techniques consist of two steps or phases. In the first phase, distances or angles are measured between known points and the object to be located. This first phase is referred to as the **ranging phase**. In the second phase, these distance or angle measurements are combined to produce the location of the object. This phase is referred as the **localization phase**.

Some of the prominent techniques for the ranging phase include: 1. Received Signal Strength Indicator (RSSI), 2. Incremental Stepping of Transmission Power, 3. Time of Arrival (ToA), and 4. Angle of Arrival (AoA) [1,4].

Depending on the method used for ranging, an appropriate localization technique is applied in the second phase. The following localization strategies have been proposed [1,4]:

1. **Trilateration** – This is one of the more popular strategies and is used when

the exact distances between known points and an object to be located are available. When the distance between an object and three points are given, the object's location can be computed as the intersection of three circles.

2. **Bounded Intersection** – The trilateration technique works well when the three circles intersect at a single point, but this is rarely the case when estimates are used in ranging. When using incremental stepping of transmission power for ranging, maximum values can be used for estimating the distances. The object to be located would fall into a geometric region that is the intersection of three circles.

3. **Triangulation** – The triangulation method is useful if the angle between two objects can be measured.

4. **Maximum Likelihood** – When estimates are used for ranging, it is possible that region of intersection is empty. This will occur if at least one ranging estimate is too small. One method that overcomes this problem selects the point for localization that gives the minimum total error between measured estimates and distances.

The most obvious solution to the localization problem is to simply equip every node with its own GPS device. This strategy might be feasible in some scenarios, but it suffers from several limitations of GPS such as it does not work indoors or when the line-of-sight is blocked. The size, cost and power consumption of a GPS receiver are also factors that make it impractical to equip all of the nodes in a WSN with this technology. Therefore, one must develop alternate low-cost and low-power solutions.

We presented one such solution based on evolution strategies that combined information about anchor positions with distance estimates between neighboring nodes [5]. In this paper, we present a deterministic algorithm that includes an error bound with each localization estimate.

The current landscape of location sensing systems is filled with a variety of technologies. The most popular system, GPS [6], uses radio time-of-flight lateration via satellites, but has some limitations. A good discussion of location systems is found in [1]. Most of the location systems discussed rely on known positions or distances in the location or calibration process and they rely on an a priori *infrastructure*. This leads to two problems: (1) The system will not scale well to a large topology, and (2) It is very difficult to do location sensing in an ad-hoc manner.

The problem of finding the location of *all* nodes in a wireless sensor network given the location of a subset of nodes has been approached by many researchers. A system called AHLoS (Ad-Hoc Localization System) [7] assumed that *beacon* nodes are aware of their positions. The rest of the nodes in the system are referred to as *unknown*, as these nodes will try to discover their location. The beacon nodes broadcast their location. An unknown node within range of three or more beacons estimates its position to minimize the mean square error. A technique called *iterative multilateration* is then used to handle the localization of all the nodes in the system. The accuracy of ranging in AHLoS was very precise, but it comes with a substantial cost in CPU power, energy consumption, and hardware circuitry. The percentage of beacons necessary to perform collaborative multilateration is relatively high. For example, for 90% of the network to localize in a network of 300 nodes, it is necessary for 45 of these nodes to be designated as beacons.

Many of the other existing localization algorithms, such as ABC [8], TERRAIN [9], and the work proposed by Meguerdichian et al [10], consist of two phases: 1) Estimate Position, and 2) Iterative Refinement The iterative refinement phase consists

of approximately 25 iterations of **every** node sending its location to all of its neighbors. This process must be repeated when changes to the topology occur. Although this technique seems to provide good results as far as localization accuracy is concerned, the energy utilization in the wake of every node continuously broadcasting its location can be overwhelming, particularly when energy is one of the most precious resources for nodes in sensor networks.

Our algorithms do not include an iterative refinement phase, making them more efficient as far as energy conservation is concerned. In [11], rectangular bounds were placed around possible positions of nodes by using linear programming. To the best of our knowledge, no one has done work on bounding the localization error by finding the intersection of disks or washers.

4 The Disk Problem

As described in the Introduction, our work relies on finding the smallest circle that encloses the intersection of an arbitrary number of input disks. The intersection of the disks is the region in which the object of interest lies. The center of the smallest enclosing circle is our estimate of its location; the radius is our bound on the error. By choosing the center as our estimate, it minimizes the error bound. In this section, we first present an $O(n^3)$ -time algorithm for finding this smallest circle, analyze its computational complexity, and then describe how to improve it to $O(n^2)$. In [12], we prove the correctness of our algorithms.

$O(n^3)$ Algorithm for the Disk Problem

Input: Centers and radii of n distinct disks: $D_i = [x_i, y_i, r_i]$, $i = 1, \dots, n$, where $r_1 \leq r_2 \leq \dots \leq r_n$.

Output: Center and radius of the smallest disk $D^* = [x^*, y^*, r^*]$ that includes the intersection S_n of the n input disks.

1. [$O(n^2)$] Find all pairwise *intersection points* of the associated circles of the input disks. (There can be at most 2 intersection points for each pair of circles, for a total of $n(n-1)$ intersection points.)
2. [$O(n^3)$] For each intersection point, determine if this point lies in all of the input disks. Let $\{(a_k, b_k)\}$ be the set of these *corner points*, each of which lies in all of the input disks. (Although this step takes $O(n^3)$ -time, we show that the number of corner points can be reduced to $O(n)$, and they can be found in $O(n^2)$ time.)
3. [$O(n^2)$] For each corner point that lies on the smallest input circle C_1 , check to see if its antipodal point on C_1 lies in all of the input disks. If no corner points lie on C_1 , then pick any antipodal pair of C_1 and check if both points lie in all of the input disks. If any antipodal pair lies in all of the input disks, return C_1 as the smallest enclosing circle. Otherwise, proceed to Step 4.

4. [$O(n)$] Return $\text{Smallest}(\{(a_k, b_k)\})$, the smallest disk containing the $O(n)$ corner points $\{(a_k, b_k)\}$. Megiddo's linear programming algorithm finds the smallest disk containing a set of input points in linear time [13].

Next, we explain the steps of the algorithm, improve Step 2 to $O(n^2)$, and verify the $O(n^2)$ time complexity of the overall algorithm.

STEP 1: Finding the Pairwise Intersection Points

In [14], an $O(1)$ -time complexity algorithm is provided for finding the intersection of two circles. Using this algorithm $n(n-1)/2$ times, once for each pair of the n circles, we can find all pairwise intersection points in $O(n^2)$ -time.

STEP 2: Finding the Corner Points in $O(n^2)$ time

In [12], we establish that the number of corner points can be reduced to at most $2n-2$. Now, we will show how these corner points can be found in $O(n^2)$ time.

For each input circle C_i , order the intersection points with all circles $C_j : 1 \leq j < i$, in a counter-clockwise (increasing-angle) direction. For each intersection point, label that point “+” if the segment from that point in the positive direction lies in both circles meeting at that point, and label it “-” otherwise. Traverse the circle in a positive-angle direction until a “+” is followed immediately by a “-”. Those two points, which we will call a $+ -$ pair, are the only corner points of S_j on C_i . This process is illustrated for three circles in Fig. 2(i). If there are two such $+ -$ pairs, then we can conclude C_i contributes no corner points. Doing this for all input circles except for the smallest one, C_1 , results in an $O(n^2 \log n)$ method for finding all the corner points, which is better than $O(n^3)$, at least.

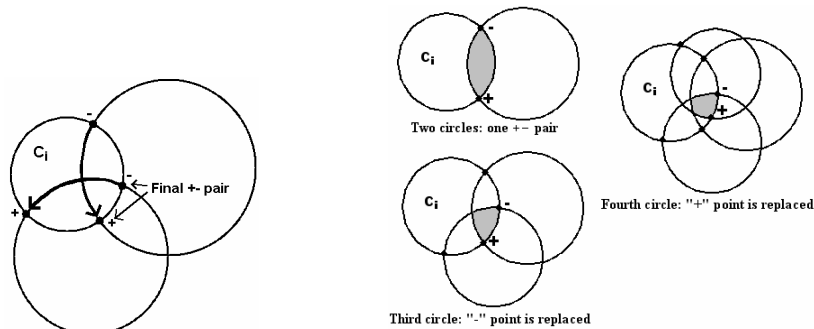


Fig. 2. (i) final plus/minus pair, (ii) processing plus/minus pairs.

In this approach, we are finding all of the corner points of $S_2, S_3, \dots,$ and S_n , and we proved this is at most $2n-2$ total points. A final step to eliminate those not in S_n , then can be done in $O(n^2)$ time.

We can modify the above approach to avoid the $O(n \log n)$ work required to sort each circle's intersection points, as follows. While processing circle C_i , keep the

current $+-$ pair. If the “+” point of the next intersection pair is greater than the current “+” point, then replace the current with the next. If the “-” point of the next intersection pair is less than the current “-” point, then replace the current with the next. If either the next “+” point is greater than the current “-” point or the next “-” point is less than the current “+” point, then we can conclude C_i contributes no corner points to S_i . If there is an input circle $C_j : 1 \leq j < i$ that does not intersect C_i , then there are 2 cases: If C_j is contained in C_i , we again conclude C_i contributes no corner points to S_i . If C_i and C_j are disjoint, the intersection set S_n is empty. This approach requires only $O(n)$ work for each input circle, leading to an $O(n^2)$ method for finding all the corner points.

This process of using the plus/minus pairs to find the contributing corner points of C_i is illustrated in Fig. 2(ii). With two circles, there is one plus/minus pair indicating the two corner points of C_i . When a third circle is added, the “-” point is replaced. The “+” point is replaced when the fourth circle is added, and the final plus/minus pair represents the two corner points contributed by C_i .

We leave as an open question whether this can be improved to $O(n \log n)$, or even $O(n)$.

STEPS 3 and 4: Checking the Antipodal of each Corner Point

If a diameter of D_1 is contained in S_n (see Fig. 3), then D_1 is the solution to the Disk problem. Otherwise, the solution is the smallest enclosing disk containing all of the corner points. This smallest disk can be obtained by Megiddo’s linear-time algorithm [13]. The checks are made in Steps 3 and 4 of the algorithm and their validity was established in [12].

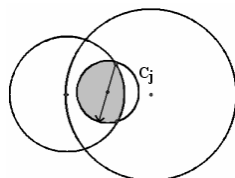


Fig. 3. Antipodal lies in S_n , thus $D_1 = D^*$.

Next we discuss the computational aspects of Steps 3 and 4.

Given the center of a circle (x_c, y_c) and a point on that circle (x_p, y_p) , the antipodal point (x_a, y_a) can be found as $x_a = x_c + (x_c - x_p)$ and $y_a = y_c + (y_c - y_p)$.

To check if an antipodal point (x_a, y_a) lies in all n input disks $C_i = (x_i, y_i, r_i)$, you would simply examine if $(x_a - x_i)^2 + (y_a - y_i)^2 \leq r_i^2$ is true for each $i = 1, \dots, n$.

To test whether each of the $O(n)$ corner points of C_1 has an antipodal point in each of the n input disks exhaustively would require $O(n^2)$ work. Perhaps we can do better. Can we improve this step to $O(n)$? It is true that there can be $O(n)$ corner points on the smallest circle. Therefore, we would like to be able to check to see if the

antipodal point of each is in the intersection, without having to explicitly check its inclusion in all the input disks.

By using the Disk Method to find D^* , the point (x^*, y^*) is used to estimate an object's location and the localization error is bounded by r^* .

5 The Washer Problem

We now consider the *Washer Problem*. If we can place a minimum bound, $d' - \epsilon$, on the distance between an object and an anchor, we are sure that the object will lie outside of the disk centered at the anchor and having a radius of $d' - \epsilon$. This forms the basis of the Washer Problem detailed below:

$O(n^3)$ Algorithm for the Washer Problem

Input: Centers and radii of n (inclusion) disks: $D_i = [x_i, y_i, r_i]$, $i = 1, \dots, n$, where $r_1 \leq r_2 \leq \dots \leq r_n$, and m (exclusion) disks: $D'_i = [x'_i, y'_i, r'_i]$, $i = 1, \dots, m$, where $r'_1 \leq r'_2 \leq \dots \leq r'_m$. All input disks are pair-wise distinct.

Output: Center and radius of the smallest disk $D^* = [x^*, y^*, r^*]$ that includes the set of points $S = I - E$, where $I = (D_1 \cap D_2 \cap \dots \cap D_n)$ and $E = (D'_1 - C'_1) \cup (D'_2 - C'_2) \cup \dots \cup (D'_m - C'_m)$. That is, S is the set of points included in all the inclusion disks and excluded from the interiors of all the exclusion disks. In Fig. 4(i), the set S is indicated by the shaded region. In the following algorithm, we assume that the number of inclusion disks and exclusion disks are approximately the same ($m \leq 2n$). Therefore, $O(n) = O(m)$.

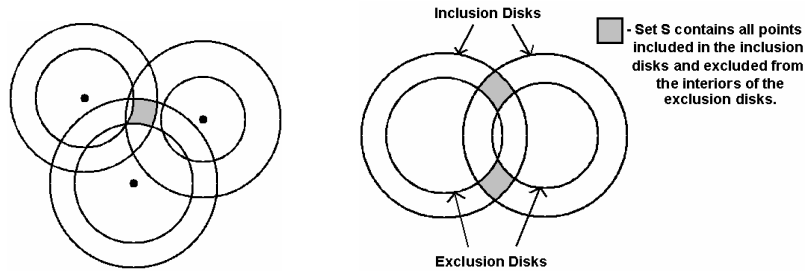


Fig. 4. (i) the Washer problem, (ii) set S broken into multiple pieces.

1. [$O(n^2)$] Find all pairwise *intersection points* of the associated circles of the n inclusion disks and m exclusion disks. There can be at most 2 intersection points for each pair of circles, for a total of $(m+n)(m+n-1)/2$ intersection points.
2. [$O(n^3)$] For each intersection point, determine if this point lies in all of the inclusion disks and outside all of the exclusion disks. Let $\{(a_k, b_k)\}$ be the set of these *corner points*.
3. [$O(n^2)$] For each of the $O(n)$ corner points that lies on the smallest input circle C_1 , check to see if its antipodal point on C_1 lies in all of the inclusion disks and

outside all of the exclusion disks. If any such antipodal passes this check, return C_1 as the smallest enclosing circle. Otherwise, proceed to Step 4.

4. [$O(n^2)$] Return $\text{Smallest}(\{(a_k, b_k)\})$, the smallest disk containing all of the corner points $\{(a_k, b_k)\}$ using Meggido's linear-time linear programming algorithm [13].

Note that corner points = $O(n^2)$.

As shown in Fig. 4(ii), it is possible that the inclusion disks and exclusion disks will break the set S into multiple disjoint pieces. Because of this, our proof of the $O(n)$ bound on the number of corner points does not carry over to the washer case. We leave open the questions of whether the number of corner points is less than $O(n^2)$ and whether they can be found in less than $O(n^3)$ time.

We can, however, still establish the validity of Step 4, which results in our $O(n^3)$ algorithm above. The proof is similar to that of the Disk Method. As with the Disk Method, the smallest enclosing circle D^* is used in the localization problem by estimating an object's location at (x^*, y^*) with an error bound of r^* .

6 Performance Results

A program was developed to implement our algorithms in order to simulate the localization process under varying conditions. We used the Delphi programming environment because its excellent graphics capabilities make it ideal for illustrating visually the washers used in our technique.

The system assumes a node can estimate the distance between itself and each of its neighbors. Although more accurate ranging techniques will produce smaller localization errors, our approach is not dependent on any one ranging technique. The system also assumes that a small subset of the nodes, *anchors*, are aware of their location. Anchor nodes are either physically placed at known positions or they are equipped with a positioning technology such as GPS. Finally, for simplicity, the system assumes: (1) signals are omni directional and symmetric, (2) all nodes have the same transmission range, and (3) the network is connected.

In our simulation experiments, we randomly deployed 500 sensor nodes over a 250,000 ft² region (500 x 500 foot square). The anchor nodes were placed in a mesh configuration. When running trials based on radio transmission, the radio range was assumed to be 100 feet. This range was based on experimental results using second-generation MICA2 motes [15].

When calculating perceived distance between 2 nodes, a random normally distributed error $e \in [-\epsilon, \epsilon]$ is generated and added to the actual distance d . The resulting perceived distance $d' = d + e$ is thus within $d \pm \epsilon$, where ϵ is the maximum ranging error. Next, to ensure a correct bound, the disk radius of $d' + \epsilon$ is used in the algorithm. When we use the Washer method, the inner disk's radius is taken to be $d' - \epsilon$.

By having each node find an upper bound on its multi-hop distance to every anchor node, we can localize the nodes using any or all of the anchors, not just the neighboring ones. To accomplish this, each anchor node broadcasts its position to

initialize the localization process. All other nodes will then broadcast these anchor positions as well as their maximum distance estimate to each anchor. After several iterations of this process, each node will now have an upper bound on the distance to every anchor node in the network. These upper bounds are used as the circles in the Disk method. If a node is a neighbor to an anchor node, it uses the distance estimate $d' - \epsilon$ as the radius of the washer's inner circle. If the node is not a neighbor to an anchor, the best we can do is to use the maximum transmission range (e.g., 100 feet) as the washer's inner circle.

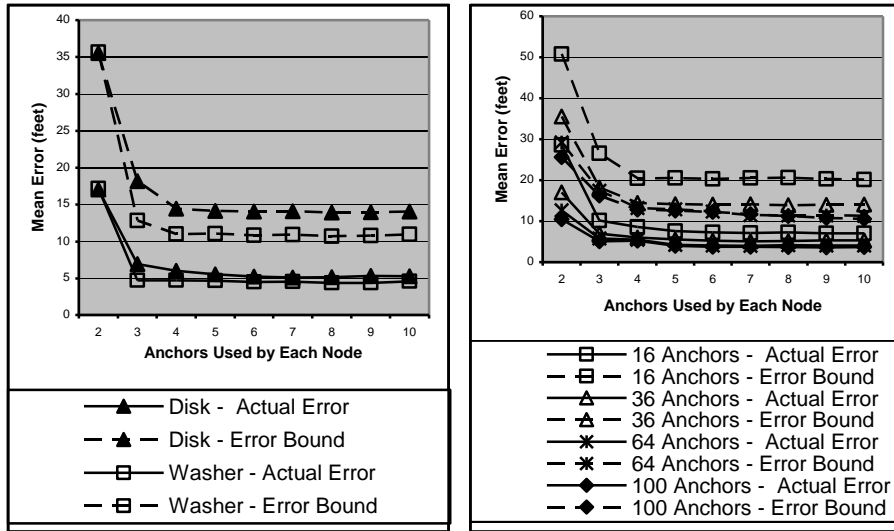


Fig. 5. (i) Disk versus Washer, (ii) Varying the number of anchors.

When using the exclusion disks in addition to the inclusion disks, the size of the smallest enclosing circle is sometimes reduced (see Fig. 4(i)). A node uses the distances, $d' \pm \epsilon$, from multiple anchors to form the washers for localization. Each node chooses its closest anchors when selecting anchors to be used. In our experiments, we varied the number of anchors, or washers, used in the localization calculations from 2 to 10. A 500-node network was generated and a 6x6 mesh of 36 anchors was used. An ϵ of 10 feet was used for the maximum ranging error.

Fig. 5(i) shows averaged results of localizing the 500-node network. The Washer method had a slightly smaller mean actual localization error than Disk. When using 10 anchors, for example, the actual error of the Disk method was 5.29 feet and Washer was 4.60 feet. When looking at the bound on the localization error, recall that it is the radius of the smallest enclosing circle of the intersecting washers. The Washer method gave a slightly tighter bound on the error than Disk. The error bound for Disk was 14.07 feet and Washer was 10.96 feet.

First, note that the average actual error is quite small on the order of $\epsilon / 2$, or half of the maximum ranging error. This is a bit surprising since the actual error is in 2 dimensions, whereas the ranging error is a 1-dimensional measure. Second, our error bound is also generally quite small, being approximately twice the actual error.

In Fig. 5(ii), we show how increasing the density of the anchors affects the localization error. Keeping constant the 500x500 feet region and the 100-foot radio range, we ran experiments using a 500-node network and varied the mesh sizes of anchors to be 16, 36, 64, and 100. As one might expect, the networks with more anchors per square foot produced lower errors. Having more anchors as neighbors provided smaller inclusion disks compared to the disks resulting from multi-hop paths. With 10 anchors contributing, the mean actual error for 16, 36, 64, and 100 anchors were 7.07, 5.29, 4.08, and 3.63 feet, respectively. The mean error bounds for these same networks were 20.19, 14.07, 11.33, and 10.58 feet.

In most cases, increasing the number of anchors, or washers, used for localization improved the results. This increasing of anchors used eventually plateaus. As you can see from Fig. 5(i and ii), this plateau appears to occur at about 4 or 5 anchors. As more anchors are used, however, this increases the number of computations a node must perform to localize. This can have an effect on energy utilization of a node, which in turn, affects the lifetime of a wireless sensor network. Therefore, even though we showed the efficiency for the Disk and Washer algorithms to be $O(n^2)$ and $O(n^3)$, respectively, it appears that using values of n larger than 4 or 5 will require more work but add no significant degree of accuracy.

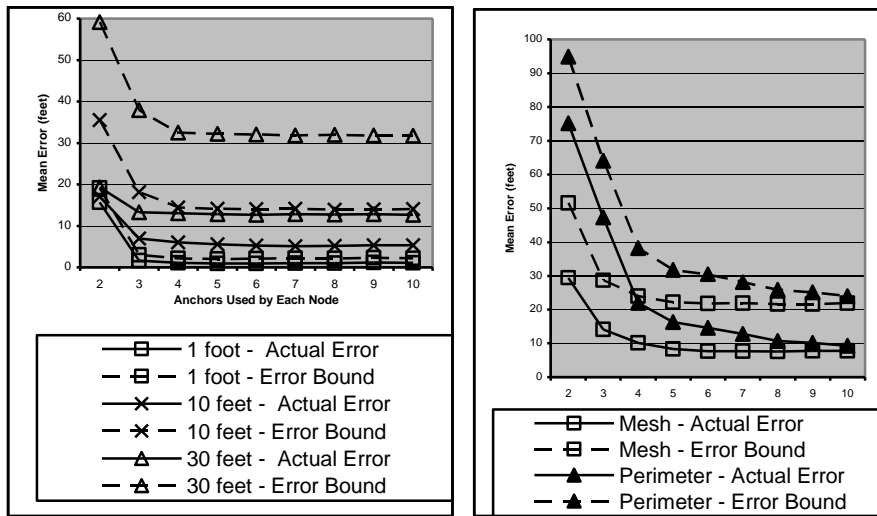


Fig. 6. (i) Varying the ranging error, (ii) Varying the number of anchors.

As mentioned previously, our technique does not rely on any specific ranging technique. In Fig. 6(i), we show how the accuracy of the ranging method affects the localization error. We are using disks with radius $d'+\epsilon$ based on distance estimates using values of 1, 10, and 30 feet for the maximum ranging error ϵ . In these experiments when 10 inclusion disks were used, a maximum ranging error ϵ of 1, 10, and 30 feet produced mean actual errors of 1.05, 5.29, and 12.66 feet respectively. The error bounds were 2.18, 14.07, and 31.72 feet. As one would expect, the smaller values of ϵ produce better localization accuracy. When ϵ of 30 feet was used, the

error bound was approximately ϵ and the actual errors were less than $\epsilon/2$.

In all of the previous examples, we deployed the anchors in a mesh configuration. We decided to consider an anchor configuration that places all of the anchors along the perimeter of the region. The perimeter configuration might make sense when placing anchors in a region was difficult (military, volcano, heavy foliage, etc.). We used a 500-node network of 16 anchors and compared the accuracy of a 4x4 mesh of anchors versus a perimeter arrangement with 5 anchors along each side of the region. In the perimeter scenario, over half of the nodes did not have any neighbors that were anchors.

As illustrated in Fig. 6(ii), when only a few washers were used in the computation, the perimeter configuration produced much larger errors than the mesh. As more washers contributed to the localization, however, the two configurations produced comparable results. With 10 anchors used, for example, the mean actual error was 7.76 feet for mesh and 9.31 feet for perimeter. The mean error bounds for mesh and perimeter were 21.87 and 23.97 feet, respectively. It is worth noting that when more anchors are used in the perimeter configuration, it tended to give larger errors. This was because a node would choose all of its neighboring anchors from the same boundary. The resulting smallest enclosing circle would thus be along that border instead of towards the interior of the region where the node actually lies. When using perimeter, one could address this by choosing anchors from different boundaries.

7 Conclusions and Future Work

We have shown that we can bound the error on the localization of a node by finding the smallest enclosing circle that covers the intersection of multiple disks. The disks are constructed by taking the maximum distances $d'+\epsilon$ from anchor nodes as the disk radius and use the anchor's position as the disk center. We use the center of the smallest enclosing circle as the location estimate of the node and the radius of the circle as the error bound. We provided a novel $O(n^2)$ algorithm for finding this location estimate and error bound, where n is the number of anchors used by each node. We believe that Step 2 of our Disk algorithm in which we compute the corner points can be improved and we leave this problem as part of our future work.

We extended our Disk technique to what we called the Washer method. Based on the distance to anchors, the washers are constructed using the inclusion disks as the exterior circle and exclusion disks with radius $d'-\epsilon$ as the inner circle. We provided an $O(n^3)$ algorithm for solving the Washer problem and are hopeful that we can improve on this efficiency in the future.

Providing a location estimate for a device is more meaningful if you can say something about the confidence that you have in the estimate. In our technique, we are saying that there must be a bound on the ranging estimate between two neighboring devices. For example, we say that there exists some maximum error ϵ in which distance estimates between the two devices will always fall into the range $d \pm \epsilon$, where d is the actual distance between the devices. In some conditions, the ranging estimate may be very accurate most of the time, but occasionally produces a large error. This type of environment would yield a large value of ϵ and, as illustrated

in Fig. 6(i), a bigger localization error. To address this, we would like to extend our work so that confidence intervals can be used in conjunction with the location estimates and error bounds.

It is important to note that our technique does not depend on the number of nodes in the network or the percentage of nodes that need to be anchors. Given a region and a set of anchor nodes in that region, our system will produce the same accuracy and amount of work per device for locating 10,000 devices as it does for locating one device. The important factors in the localization accuracy, as reported in the previous section, are anchor density, maximum ranging error, number of anchors used in the computations, and anchor configuration.

References

1. Hightower, J., Borriello, G.: *Location Systems for Ubiquitous Computing*, IEEE Computer, August 2001.
2. Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., MainWaring, A., Estrin, D.: "Habitat Monitoring with Sensor Networks", *Communications of the ACM*, Volume 47, Number 6, pp. 34-40, June 2004.
3. Aspnes, J., Goldenberg, D., Yang, R.: On the computational complexity of sensor network localization. *Algorithmic Aspects of Wireless Sensor Networks: First International Workshop, ALGOSENSORS 2004*, Turku, Finland, July 16, 2004.
4. Terwilliger, M., Gupta, A., Bhuse, V., Kamal, Z., Salahuddin, M.: "A Localization System using Wireless Network Sensors: A Comparison of Two Techniques", *Proceedings of the First Workshop on Positioning, Navigation and Communication*, Hannover, Germany, March 2004.
5. Terwilliger, M., Gupta, A., Khokhar, A., Greenwood, G.: "Localization Using Evolution Strategies in Sensornets", *Proceedings of the IEEE Congress on Evolutionary Computation 2005*, Edinburgh, Scotland, September 2005.
6. Enge, P., Misra, P.: *Special Issue on GPS: The Global Positioning System*, Proceedings of the IEEE, Volume 87, Number 1, pp. 3-172, January 1999.
7. Savvides, A., Han, C., Srivastava, M.: "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors", In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (MOBICOM '01), 2001.
8. Savarese, C., Rabaey, J., Beutel, J.: "Locationing in Distributed Ad-Hoc Wireless Sensor Networks," Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 4, (Salt Lake City, UT), pp. 2037.
9. Savarese, C., Rabaey, J., Langendoen, K.: "Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks", Proceedings of the General Track: 2002 USENIX Annual Technical Conference, p.317-327, June 10-15, 2002.
10. Meguerdichian, S., Slijepcevic, S., Karayan, V., Potkonjak, M.: "Localized Algorithms in Wireless Ad-Hoc Networks: Location Discovery and Sensor Exposure", *Proceedings of MobiHOC 2001*, Long Beach, CA, 2001.
11. Doherty, L., Pister, K., Ghaoui, L.: "Convex Position Estimation in Wireless Sensor Networks", In *Proceedings of the IEEE InfoCom'01*, Vol. 3. pp. 1655-1663.
12. Terwilliger, M., Gupta, A., Coullard, C.: "On Bounding the Localization Errors", Technical Report TR/05-07, Department of Computer Science, Western Michigan University, 2005.
13. Meggido, N.: "Linear-Time Algorithms for Linear Programming in R^3 and Related Problems", *SIAM Journal on Computing*, Volume 12, pages 759-776 (1983).
14. Bourke, P.: "Intersection of Two Circles", <http://astronomy.swin.edu.au/~pbourke/geometry/2circle/>, April 1997.
15. Crossbow Technology, <http://www.xbow.com>, last viewed August 2005.