Distributed Localization Algorithms

Koen Langendoen and Niels Reijers
Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4, 2628CD Delft, The Netherlands
K.G.Langendoen@ewi.tudelft.nl
N.Reijers@ewi.tudelft.nl

1 Introduction

New technology offers new opportunities, but it also introduces new problems. This is particularly true for sensor networks where the capabilities of individual nodes are very limited. Hence, collaboration between nodes is required, but energy conservation is a major concern, which implies that communication should be minimized. These conflicting objectives require unorthodox solutions for many situations.

A recent survey by Akyildiz et al. discusses a long list of open research issues that must be addressed before sensor networks can become widely deployed [1]. The problems range from the physical layer (low-power sensing, processing, and communication hardware) all the way up to the application layer (query and data dissemination protocols). In this chapter we address the issue of localization in ad-hoc sensor networks. That is, we want to determine the location of individual sensor nodes without relying on external infrastructure (base stations, satellites, etc.).

The localization problem has received considerable attention in the past, as many applications need to know where objects or persons are, and hence various location services have been created. Undoubtedly, the Global Positioning System (GPS) is the most well-known location service in use today. The approach taken by GPS, however, is unsuitable for low-cost, ad-hoc sensor networks since GPS is based on extensive infrastructure (i.e. satellites). Likewise solutions developed in the area of robotics [2, 13, 20] and ubiquitous computing [10] are generally not applicable for sensor networks as they require too much processing power and energy.

Recently a number of localization systems have been proposed specifically for sensor networks [4, 5, 6, 7, 9, 14, 16, 17, 19]. We are interested in truly distributed algorithms that can be employed on large-scale ad-hoc sensor networks (100+ nodes). Such algorithms should be:

1. self-organizing (i.e. do not depend on global infrastructure),

2. robust (i.e. be tolerant to node failures and range errors), and

3. energy efficient (i.e. require little computation and, especially, communication).

These requirements immediately rule out some of the proposed localization algorithms for sensor networks. In this chapter, we carry out a thorough sensitivity analysis on three algorithms that do meet the above requirements to determine how well they perform under various conditions. In particular, we study the impact of the following parameters: range errors, connectivity (density), and anchor fraction. These algorithms differ in their position accuracy, network coverage, induced network traffic, and processor load. Given the (slightly) different design objectives for the three algorithms, it is no surprise that each algorithm outperforms the others under a specific set of conditions. Under each condition, however, even the best algorithm leaves much room for improving accuracy and/or increasing coverage.

In this chapter we will

1. identify a common, 3-phase, structure in the selected distributed localization algorithms.
2. identify a generic optimization applicable to all algorithms.
3. provide a detailed comparison on a single (simulation) platform.
4. show that there is no algorithm that performs best, and that there exists room for improvement in most cases.

Section 2 discusses the selection, generic structure, and operation of three distributed localization algorithms for large-scale ad-hoc sensor networks. These algorithms are compared on a simulation platform, which is described in Section 3. Section 4 presents intermediate results for the individual phases, while Section 5 provides a detailed overall comparison and an in-depth sensitivity analysis. Finally, we give conclusions in Section 6.

2 Localization algorithms

Before discussing distributed localization in detail, we first outline the context in which these algorithms have to operate. A first consideration is that the requirement for sensor networks to be self-organizing implies that there is no fine control over the placement of the sensor nodes when the network is installed (e.g., when nodes are dropped from an airplane). Consequently, we assume that nodes are randomly distributed across the environment. For simplicity and ease of presentation we limit the environment to 2 dimensions, but all algorithms are capable of operating in 3D. Figure 1 shows an example network with 25 nodes; pairs of nodes that can communicate directly are connected by an edge. The connectivity of the nodes in the network (i.e. the average number of neighbors) is an important parameter that has a strong impact on the accuracy of most localization algorithms (see Sections 4 and 5). It is initially determined by the node density and radio range, and in some cases it can be adjusted dynamically by changing the transmit power of the RF radio.

In some application scenarios, nodes may be mobile. In this paper, however, we focus on static networks, where nodes do not move, since this is already a challenging condition for distributed localization. We assume that some *anchor* nodes have *a priori* knowledge of their own position with respect to some global coordinate system. Note that anchor nodes have the same capabilities (processing, communication, energy consumption, etc.) as all other sensor nodes with unknown positions; we do not consider approaches based on an external infrastructure with specialized *beacon* nodes (access points) as used in, for example, the GPS-less [4], Cricket [15], and RADAR [3] location systems. Ideally the fraction of anchor nodes should be as low as possible to minimize the installation costs. Our simulation results show that, fortunately, most algorithms are rather insensitive to the number of anchors in the network.

The final element that defines the context of distributed localization is the capability to measure the distance between directly connected nodes in the network. From a cost perspective it is attractive

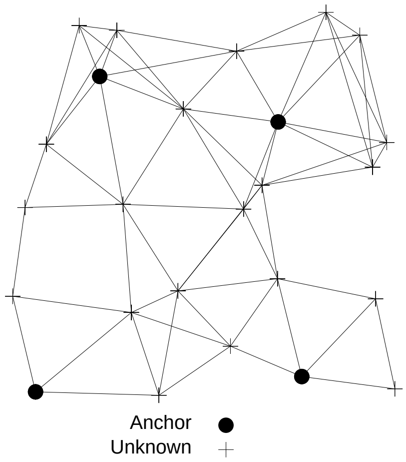

Figure 1: Example network topology.

to use the RF radio for measuring the range between nodes, for example, by observing the signal strength. Experience has shown, however, that this approach yields poor distance estimates [11, 24, 18]. Much better results are obtained by time-of-flight measurements, particularly when acoustic and RF signals are combined [8, 18, 19]; accuracies of a few percent of the transmission range are reported. However, this requires extra hardware on the sensor boards.

Several different ways of dealing with the problem of inaccurate distance information have been proposed. The APIT [9] algorithm by He et al. only needs distance information accurate enough for two nodes determine which of them is closest to an anchor. GPS-less [4] by Bulusu et al. and DV-hop [14] by Niculescu and Nath. do not use distance information at all, and are based on topology information only. Ramadurai and Sichitiu [16] propose a probabilistic approach to the localization problem. Not only the measured distance, but also the confidence in the measurement is used.

It is important to realize that the main three context parameters (connectivity, anchor fraction, and range errors) are dependent. Poor range measurements can be compensated for by using many anchors and/or a high connectivity. This chapter provides insight in the complex relation between connectivity, anchor fraction, and range errors for a number of distributed localization algorithms.

2.1 Generic approach

From the known localization algorithms specifically proposed for sensor networks, we selected the three approaches that meet the basic requirements for self-organization, robustness, and energy-efficiency:

1. **Ad-hoc positioning** by Niculescu and Nath [14],
2. **N-hop multilateration** by Savvides et al. [19], and
3. **Robust positioning** by Savarese et al. [17].

The other approaches often include a central processing element (e.g., ‘convex optimization’ by Doherty et al. [7]), rely on an external infrastructure (e.g., ‘GPS-less’ by Bulusu et al. [4]), or induce too much communication (e.g., ‘GPS-free’ by Capkun et al. [5]). The three selected algorithms are fully distributed and use local broadcast for communication with immediate neighbors. This last feature allows them to be executed before any multi-hop routing is in place, hence, they can support efficient location-based routing schemes like GAF [23].

Table 1: Algorithm classification.

phase	Ad-hoc positioning [14]	Robust positioning [17]	N-hop multilateration [19]
1. distance	Euclidean	DV-hop	Sum-dist
2. position	Lateration	Lateration	Min-max
3. refinement	no	yes	yes

Although the three algorithms were developed independently, we found that they share a common structure. We were able to identify the following generic, 3-phase approach[1] for determining the individual node positions:

1. Determine the distances between unknowns and anchor nodes.
2. Derive for each node a position from its anchor distances.
3. Refine the node positions using information about the range (distance) to, and positions of, neighboring nodes.

The original descriptions of the algorithms present the first two phases as a single entity, but we found that separating them provides two advantages. First, we obtain a better understanding of the combined behavior by studying intermediate results. Second, it becomes possible to mix-and-match alternatives for both phases to tailor the localization algorithm to the external conditions. The refinement phase is optional and may be included to obtain more accurate locations.

In the remainder of this section we will describe the three phases (distance, position, and refinement) in detail. For each phase we will enumerate the alternatives as found in the original descriptions. Table 1 gives the breakdown into phases of the three approaches. When applicable we also discuss (minor) adjustments to (parts of) the individual algorithms that were needed to ensure compatibility with the alternatives. During our simulations we observed that we occasionally operated (parts of) the algorithms outside their intended scenarios, which deteriorated their performance. Often, small improvements brought their performance back in line with the alternatives.

2.2 Phase 1: Distance to anchors

In this phase, nodes share information to collectively determine the distances between individual nodes and the anchors, so that an (initial) position can be calculated in Phase 2. None of the Phase 1 alternatives engages in complicated calculations, so this phase is communication bounded. Although the three distributed localization algorithms each use a different approach, they share a common communication pattern: information is flooded into the network, starting at the anchor nodes. A network-wide flood by some anchor A is expensive since each node must forward A's information to its (potentially) unaware neighbors. This implies a scaling problem: flooding information from all anchors to all nodes will become much too expensive for large networks, even with low anchor fractions. Fortunately a good position can be derived in Phase 2 with knowledge (position and distance) from a limited number of anchors. Therefore nodes can simply stop forwarding information when enough anchors have been "located". This simple optimization presented in the Robust positioning approach proved to be highly effective in controlling the amount of communication (see Section 5.3). We modified the other two approaches to include a *flood limit* as well.

[1]Our three phases do not correspond to the three of Savvides et al. [19]; our structure allows for an easier comparison of all algorithms.

Sum-dist

The most simple solution for determining the distance to the anchors is simply adding the ranges encountered at each hop during the network flood. This is the approach taken by the N-hop multilateration approach, but it remained nameless in the original description [19]; we name it *Sum-dist* in this paper. Sum-dist starts at the anchors, who send a message including their identity, position, and a path length set to 0. Each receiving node adds the measured range to the path length and forwards (broadcasts) the message if the *flood limit* allows it to do so. Another constraint is that when the node has received information about the particular anchor before, it is only allowed to forward the message if the current path length is less than the previous one. The end result is that each node will have stored the position and minimum path length to at least *flood limit* anchors.

DV-hop

A drawback of Sum-dist is that range errors accumulate when distance information is propagated over multiple hops. This cumulative error becomes significant for large networks with few anchors (long paths) and/or poor ranging hardware. A robust alternative is to use topological information by counting the number of hops instead of summing the (erroneous) ranges. This approach was named *DV-hop* by Niculescu and Nath [14], and *Hop-TERRAIN* by Savarese et al. [17]. Since the results of DV-hop were published first we will use this name.

DV-hop essentially consists of two flood waves. After the first wave, which is similar to Sum-dist, nodes have obtained the position and minimum hop count to at least *flood limit* anchors. The second *calibration* wave is needed to convert hop counts into distances such that Phase 2 can compute a position. This conversion consists of multiplying the hop count with an average hop distance. Whenever an anchor a_1 infers the position of another anchor a_2 during the first wave, it computes the distance between them, and divides that by the number of hops to derive the average hop distance between a_1 and a_2. When calibrating, an anchor takes all remote anchors into account that it is aware of. When later, information on extra anchors is received, the calibration procedure is repeated. Nodes forward (broadcast) calibration messages only from the first anchor that calibrates them, which reduces the total number of messages in the network.

Euclidean

A drawback of DV-hop is that it fails for highly irregular network topologies, where the variance in actual hop distances is very large. Niculescu and Nath have proposed another method, named Euclidean, which is based on the local geometry of the nodes around an anchor. Again anchors initiate a flood, but forwarding the distance is more complicated than in the previous cases. When a node has received messages from two neighbors that know their distance to the anchor, and to each other, it can calculate the distance to the anchor.

Figure 2 shows a node ('Self') that has two neighbors n1 and n2 with distance estimates (a and b) to an anchor. Together with the known ranges c, d, and e, Euclidean arrives at two possible values ($r1$ and $r2$) for the distance of the node to the anchor. Niculescu describes two methods to decide on which, if any, distance to use. The *neighbor vote* method can be applied if there exists a third neighbor (n3) that has a distance estimate to the anchor and that is connected to either n1 or n2. Replacing n2 (or n1) by n3 will again yield a pair of distance estimates. The correct distance is part of both pairs, and is selected by a simple voting. Of course, more neighbors can be included to make the selection more accurate.

The second selection method is called *common neighbor* and can be applied if node n3 is connected to both n1 and n2. Basic geometric reasoning leads to the conclusion that the anchor and n3 are on the same or opposite side of the mirroring line n1-n2, and similarly whether or not Self

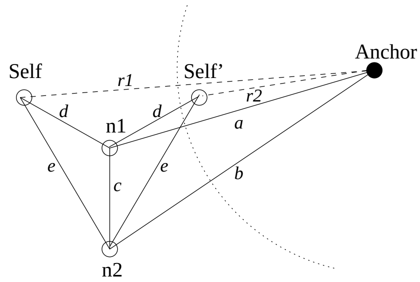

Figure 2: Determining distance using Euclidean.

and n3 are on the same side. From this it follows whether or not self and the anchor lay on the same side.

To handle the uncertainty introduced by range errors Niculescu implements a safety mechanism that rejects ill-formed (flat) triangles, which can easily derail the selection process by 'neighbor vote' and 'common neighbor'. This check verifies that the sum of the two smallest sides exceeds the largest side multiplied by a threshold, which is set to two times the range variance. For example, the triangle Self-n1-n2 in Figure 2 is accepted when $c + d > (1 + 2RangeVar) \times e$. Note that the safety check becomes more strict as the range variance increases. This leads to a lower *coverage*, defined as the percentage of non-anchor nodes for which a position was determined.

We now describe some modifications to Niculescu's 'neighbor vote' method that remedy the poor selection of the location for Self in important corner cases. The first problem occurs when the two votes are identical because, for instance, the three neighbors (n1, n2, and n3) are collinear. In these cases it is hard to select the right alternative. Our solution is to leave equal vote cases unsolved, instead of picking an alternative and propagating an error with 50% chance. We filter all indecisive cases by adding the requirement that the standard deviation of the votes for the selected distance must be at most 1/3rd of the standard deviation of the other distance. The second problem that we address is that of a bad neighbor with inaccurate information spoiling the selection process by voting for two wrong distances. This case is filtered out by requiring that the standard deviation of the selected distance is at most 5% of that distance.

To achieve good coverage, we use both the neighbor vote and common neighbor methods. If both produce a result, we use the result from the modified 'neighbor vote' because we found it to be the most accurate of the two. If both fail, the flooding process stops, leading to the situation where certain nodes are not able to establish the distance to enough anchor nodes. Sum-dist and DV-hop, on the other hand, never fail to propagate the distance and hop count, respectively.

2.3 Phase 2: Node position

In the second phase nodes determine their position based on the distance estimates to a number of anchors provided by one of the three Phase 1 alternatives (Sum-dist, DV-hop, or Euclidean). The Ad-hoc positioning and Robust positioning approaches use *lateration* for this purpose. N-hop multilateration, on the other hand, uses a much simpler method, which we named *Min-max*. In both cases the determination of the node positions does not involve additional communication.

Lateration

The most common method for deriving a position is lateration, which is a form of triangulation. From the estimated distances (d_i) and known positions (x_i, y_i) of the anchors we derive the following system of equations:

$$
\begin{aligned}
(x_1 - x)^2 + (y_1 - y)^2 &= {d_1}^2 \\
&\vdots \\
(x_n - x)^2 + (y_n - y)^2 &= {d_n}^2
\end{aligned}
$$

where the unknown position is denoted by (x, y). The system can be linearized by subtracting the last equation from the first $n-1$ equations.

$$
\begin{aligned}
{x_1}^2 - {x_n}^2 - 2(x_1 - x_n)x \; + \\
{y_1}^2 - {y_n}^2 - 2(y_1 - y_n)y &= {d_1}^2 - {d_n}^2 \\
&\vdots \\
{x_{n-1}}^2 - {x_n}^2 - 2(x_{n-1} - x_n)x \; + \\
{y_{n-1}}^2 - {y_n}^2 - 2(y_{n-1} - y_n)y &= {d_{n-1}}^2 - {d_n}^2
\end{aligned}
$$

Reordering the terms gives a proper system of linear equations in the form Ax=b, where

$$
A = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix}
$$

$$
b = \begin{bmatrix} {x_1}^2 - {x_n}^2 + {y_1}^2 - {y_n}^2 + {d_n}^2 - {d_1}^2 \\ \vdots \\ {x_{n-1}}^2 - {x_n}^2 + {y_{n-1}}^2 - {y_n}^2 + {d_n}^2 - {d_{n-1}}^2 \end{bmatrix}
$$

The system is solved using a standard least-squares approach: $\hat{x} = (A^T A)^{-1} A^T b$. In exceptional cases the matrix inverse cannot be computed and Lateration fails. In the majority of the cases, however, we succeed in computing a location estimate $\hat{x}$. We run an additional sanity check by computing the residue between the given distances (d_i) and the distances to the location estimate $\hat{x}$:

$$
\text{residue} = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \hat{x})^2 + (y_i - \hat{y})^2} - d_i}{n}
$$

A large residue signals an inconsistent set of equations; we reject the location $\hat{x}$ when the length of the residue exceeds the radio range.

Min-max

Lateration is quite expensive in the number of floating point operations that is required. A much simpler method is presented by Savvides et al. as part of the N-hop multilateration approach. The main idea is to construct a bounding box for each anchor using its position and distance estimate, and then to determine the intersection of these boxes. The position of the node is set to the center of the intersection box. Figure 3 illustrates the Min-max method for a node with distance estimates to three anchors. Note that the estimated position by Min-max is close to the true position computed through Lateration (i.e. the intersection of the three circles).

The bounding box of anchor a is created by adding and subtracting the estimated distance (d_a) from the anchor position (x_a, y_a):

$$
[x_a - d_a, y_a - d_a] \times [x_a + d_a, y_a + d_a].
$$

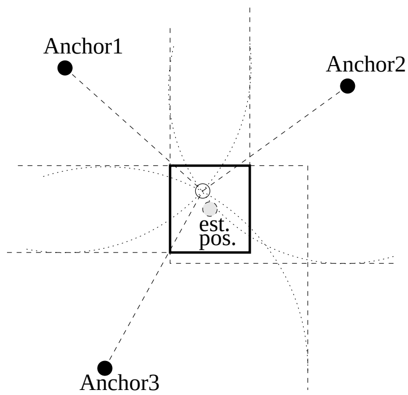

Figure 3: Determining position using Min-max.

The intersection of the bounding boxes is computed by taking the maximum of all coordinate minimums and the minimum of all maximums:

$$[\max(x_i - d_i), \max(y_i - d_i)] \times [\min(x_i + d_i), \min(y_i + d_i)]$$

The final position is set to the average of both corner coordinates. As for Lateration, we only accept the final position if the residue is small.

2.4 Phase 3: Refinement

The objective of the third phase is to refine the (initial) node positions computed during Phase 2. These positions are not very accurate, even under good conditions (high connectivity, small range errors), because not all available information is used in the first two phases. In particular, most ranges between neighboring nodes are neglected when the node-anchor distances are determined. The iterative Refinement procedure proposed by Savarese et al. [17] does take into account all inter-node ranges, when nodes update their positions in a small number of steps. At the beginning of each step a node broadcasts its position estimate, receives the positions and corresponding range estimates from its neighbors, and performs the Lateration procedure of Phase 2 to determine its new position. In many cases the constraints imposed by the distances to the neighboring locations will force the new position towards the true position of the node. When, after a number of iterations, the position update becomes small, Refinement stops and reports the final position.

The basic iterative refinement procedure outlined above proved to be too simple to be used in practice. The main problem is that errors propagate quickly through the network; a single error introduced by some node needs only d iterations to affect all nodes, where d is the network diameter. This effect was countered by 1) clipping undetermined nodes with non-overlapping paths to less than three anchors, 2) filtering out difficult symmetric topologies, and 3) associating a confidence metric with each node and using them in a weighted least-squares solution (wAx = wb). The details (see [17]) are beyond the scope of this paper, but the adjustments considerably improved the performance of the Refinement procedure. This is largely due to the confidence metric, which allows filtering of bad nodes, thus increasing the (average) accuracy at the expense of coverage.

The N-hop multilateration approach by Savvides et al. [19] also includes an iterative refinement procedure, but it is less sophisticated than the Refinement discussed above. In particular, they do not use weights, but simply group nodes into so-called computation subtrees (over-constrained configurations) and enforce nodes within a subtree to execute their position refinement in turn in a fixed sequence to enhance convergence to a pre-specified tolerance. In the remainder of this paper we will only consider the more advanced Refinement procedure of Savarese et al.

3 Simulation environment

To compare the three original distributed localization algorithms (Ad-hoc positioning, Robust positioning, and N-hop multilateration) and to try out new combinations of phase 1, 2, and 3 alternatives, we extended the simulator developed by Savarese et al. [17]. The underlying OMNeT++ discrete event simulator [21] takes care of the semi-concurrent execution of the specific localization algorithm. Each sensor node 'runs' the same C++ code, which is parameterized to select a particular combination of phase 1, 2, and 3 alternatives.

Our network layer supports localized broadcast only, and messages are simply delivered at the neighbors within a fixed radio range (circle) from the sending node; a more accurate model should take radio propagation effects into account (see future work). Concurrent transmissions are allowed if the transmission areas (circles) do not overlap. If a node wants to broadcast a message while another message in its area is in progress, it must wait until that transmission (and possibly other queued messages) are completed. In effect we employ a CSMA policy. Furthermore we do not consider message corruption, so all messages sent during our simulation are delivered (after some delay).

At the start of a simulation experiment we generate a random network topology according to some parameters (#nodes, #anchors, etc.). The nodes are randomly placed, with a uniform distribution, within a square area. Next we select which nodes will serve as an anchor. To this end we superimpose a grid on top of the square, and designate to each grid point its closest node as an anchor. The size of the grid is chosen as the maximal number s that satisfies $s \times s \leq$ #*anchor*s; any remaining anchors are selected randomly. The reason for carefully selecting the anchor positions is that most localization algorithms are quite sensitive to the presence, or absence, of anchors at the edges of the network. (Locating unknowns at the edges of the network is more difficult because nodes at the edge are less well connected and positioning techniques like lateration perform best when anchors surround the unknown.) Although anchor placement may not be feasible in practice, the majority of the nodes in large-scale networks (1000+ nodes) will generally be surrounded by anchors. By placing anchors we can study the localization performance in large networks with simulations involving only a modest number of nodes.

The measured range between connected nodes is blurred by drawing a random value from a normal distribution having a parameterized standard deviation and having the true range as the mean. We selected this error model based on the work of Whitehouse and Culler [22], which shows that, although individual distance measurements tend to overshoot the real distance, a proper calibration procedure yields distance estimates with a symmetric error distribution. The connectivity (average number of neighbors) is controlled by specifying the radio range.

At the end of a run the simulator outputs a large number of statistics per node: position information, elapsed time, message counts (broken down per type), etc. These individual node statistics are combined and presented as averages (or distributions), for example, as an average position error. Nodes that do not produce a position are excluded from such averaged metrics. To account for the randomness in generating topologies and range errors we repeated each experiment 100 times with a different seed, and report the averaged results. To allow for easy comparison between different scenarios, range errors as well as errors on position estimates are normalized to the radio range (i.e. 50% position error means a distance of half the range of the radio between the real and estimated positions).

Standard scenario The experiments described in the subsequent sections share a standard scenario, in which certain parameters are varied: radio range (connectivity), anchor fraction, and range errors. The standard scenario consists of a network of 225 nodes placed in a square with sides of 100 units. The radio range is set to 14, resulting in an average connectivity of about 12. We use an anchor fraction of 5%, hence, 11 anchors in total, of which 9 (3×3) are placed in a grid-like position. The standard deviation of the range error is set to 10% of the radio range. The default flood limit for Phase 1 is set to 4 (Lateration requires a minimum of 3 anchors). Unless specified otherwise, all data will be based on this standard scenario.

4 Results

In this section we present results for the first two phases (anchor distances and node positions). We study each phase separately and show how alternatives respond to different parameters. These intermediate results will be used in Section 5, where we will discuss the overall performance, and compare complete localization algorithms. Throughout this section we will vary one parameter in the standard scenario (radio range, anchor fraction, range error) at a time to study the sensitivity of the algorithms. The reader, however, should be aware that the three parameters are not orthogonal.

4.1 Phase 1: Distance to anchors

Figure 4 shows the performance of the Phase 1 alternatives for computing the distances between nodes and anchors under various conditions. There are two metrics of interest: first, the *bias* in the estimate, measured here using the mean of the distance errors, and second, the *precision* of the estimated distances, measured here using the standard deviation of the distance errors. Therefore, Figure 4 plots both the average error, relative to the true distance, and the standard deviation of that relative error. We will now discuss the sensitivity of each alternative: Sum-dist, DV-hop, and Euclidean.

Sum-dist

Sum-dist is the cheapest of the three methods, both with respect to computation and communication costs. Nevertheless it performs quite satisfactorily, except for large range errors (≥ 0.1). There are two opposite tendencies affecting the bias of Sum-dist. First, without range errors, the sum of the ranges along a multi-hop path will always be larger than the actual distance, leading to an overestimation of the distance. Second, the algorithm searches for the shortest path, forcing it to select links that underestimate the actual distance when range errors are present. The combined effect shows non-intuitive results. A small range error reduces the bias of Sum-dist. Initially, the detour effect leads to an overshoot, but the shortest-path effect takes over when the range errors increase, leading to a large undershoot.

When the radio range (connectivity) is increased, more nodes can be reached in a single hop. This leads to straighter paths (less overshoot), and provides more options for selecting a (incorrect) shortest path (higher undershoot). Consequently, increasing the connectivity is not necessarily a good thing for Sum-dist.

DV-hop

The DV-hop method is a stable and predictable method. Since it does not use range measurements, it is completely insensitive to this source of errors. The low relative error (5%) shows that the calibration wave is very effective. DV-hop searches for the path with the minimum number of

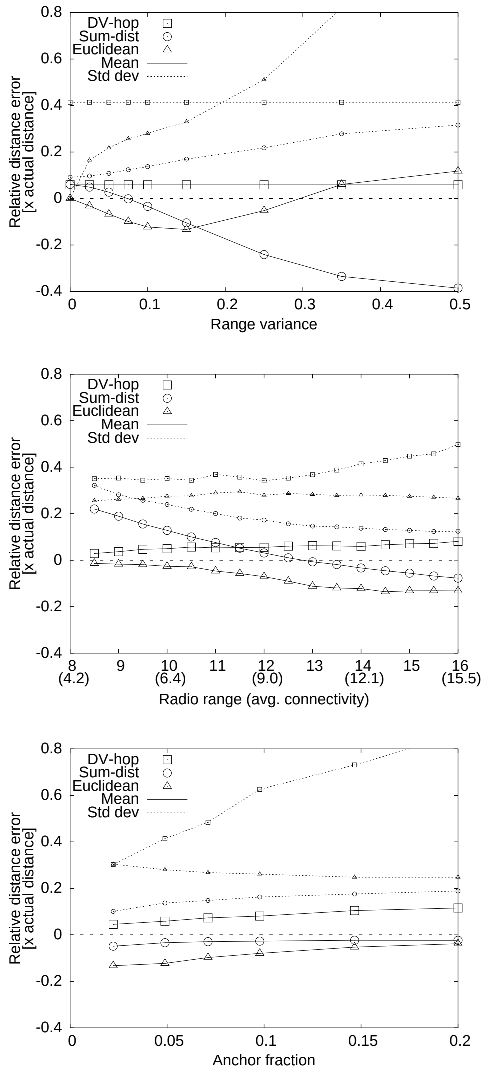

Figure 4: Sensitivity of Phase 1 methods: distance error (solid lines) and standard deviation (dashed lines).

hops, causing the average hop distance to be close to the radio range. The last hop on the path from an anchor to a node, however, is usually shorter than the radio range, which leads to a slight overestimation of the node-anchor distance. This effect is more pronounced for short paths, hence the increased error for larger radio ranges and higher anchor fractions (i.e. fewer hops).

Euclidean

Euclidean is capable of determining the exact anchor-node distances, but only in the absence of range errors and in highly connected networks. When these conditions are relaxed, Euclidean's performance rapidly degrades. The curves in Figure 4 show that Euclidean tends to underestimate the distances. The reason is that the selection process is forced to choose between two options that

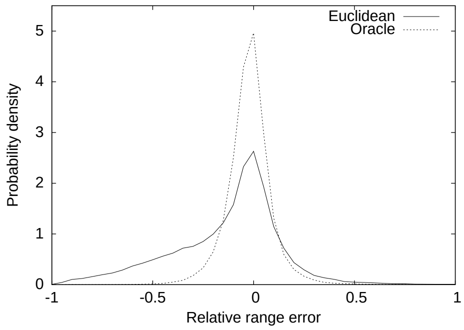

Figure 5: The impact of incorrect distance selection on Euclidean.

are quite far apart and that in many cases the shortest distance is incorrect. Consider Figure 2 again, where the shortest distance $r2$ falls within the radio range of the anchor. If $r2$ would be the correct distance then the node should be in direct contact with the anchor avoiding the need for a selection. Therefore nodes simply have more chance to underestimate distances than to overestimate them in the face of (small) range errors. This error can then propagate to nodes that are multiple hops away from the anchor, causing them to underestimate the distance to the anchor as well.

We quantified the impact of the selection bias towards short distances. Figure 5 shows the distribution of the errors, relative to the true distance, on the standard scenario for Euclidean's selection mechanism (solid line) and an oracle that always selects the best distance (dashed line). The oracle's distribution is nicely centered around zero (no error) with a sharp peak. Euclidean's distribution, in contrast, is skewed by a heavy tail at the left, signalling a bias for underestimations.

Euclidean's sensitivity for connectivity is not immediately apparent from the accuracy data in Figure 4. The main effect of reducing the radio range is that Euclidean will not be able to propagate the anchor distances. Recall that Euclidean's selection methods require at least three neighbors with a distance estimate to advance the anchor distance one hop. In networks with low connectivity, two parts connected only by a few links will often not be able to share anchors. This leads to problems in Phase 2, where fewer node positions can be computed. The effects are quite pronounced, as will become clear in Section 5 (see the coverage curves in Figure 10).

4.2 Phase 2: Node position

To obtain insight into the fundamental behavior of the the Lateration and Min-max algorithms we now report on some experiments with controlled distance errors and anchor placement. The impact of actual distance errors as produced by the Phase 1 methods will be discussed in Section 5.

Distance errors

Starting from the standard scenario we select for each node the five nearest anchors, and add some noise to the real distances. This noise is generated by first taking a sample from a normal distribution with the actual distance as the mean and a parameterized percentage of the distance as the standard deviation. The result is then multiplied by a bias factor. The ranges for the standard deviation and bias factor follow from the Phase 1 measurements.

Figure 6 shows the sensitivity of Lateration and Min-max when the standard deviation percentage was varied from 0 to 0.25, and the bias factor fixed at zero. Lateration outperforms Min-max for precise distance estimates, but Min-max takes over for large standard deviations (≥ 0.15).

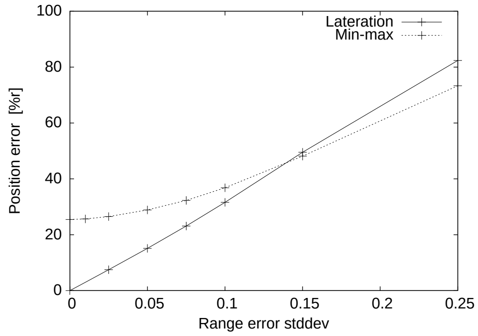

Figure 6: Sensitivity of Phase 2 to precision.

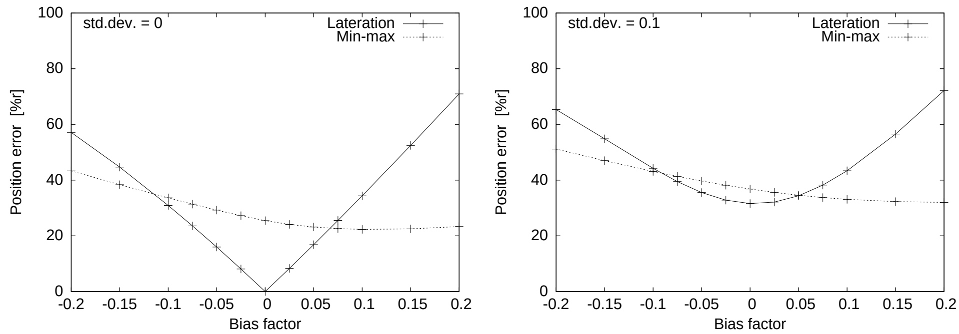

Figure 7: Sensitivity of Phase 2 to bias.

Figure 7 shows the effect of adding a bias to the distance estimates. The curves show that Lateration is very sensitive to a bias factor, especially for precise estimates (std.dev. = 0). Min-max is rather insensitive to bias, because stretching the bounding boxes has little effect on the position of the center. For precise distance estimates and a small bias factor Lateration outperforms Min-max, but the bottom graph in Figure 7 shows that Min-max is probably the preferred technique when the standard deviation rises above 10%.

Although Min-max is not very sensitive to bias, we do see that Min-max performs better for a positive range bias (i.e. an overshoot). This is a consequence of the error introduced by Min-max using a bounding box instead of a circle around anchors. For simplicity we limit the explanation to

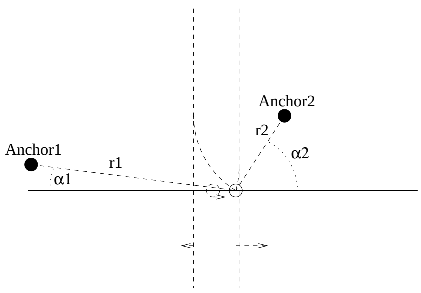

Figure 8: Min-max scenario.

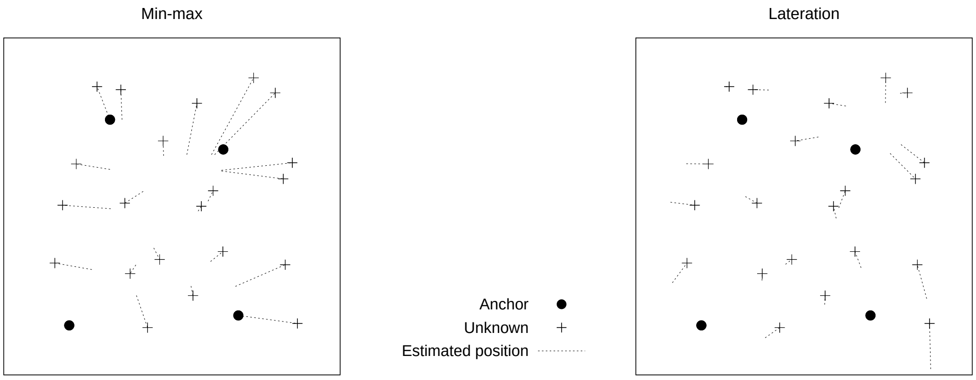

Figure 9: Node locations computed for network topology of Figure 1.

the effects on the x-coordinate only. Figure 8 shows that Anchor1 making a small angle with the x-axis yields a tight bound (to the right), and that the large angle of Anchor2 yields a loose bound (to the left). The estimated position is off in the direction of the loose bound (to the left). Adding a positive bias to the range estimates causes the two bounds to shift proportionally. As a consequence the center of the intersection moves into the direction of the bound with the longest range (to the right). Consequently the estimated coordinate moves closer to the true coordinate. The opposite will happen if the anchor with the largest angle has the longest distance. Min-max selects the strongest bounds, leading to a preference for small angles and small distances, which favors the number of "good" cases where the coordinate moves closer to the true coordinate if a positive range bias is added.

Anchor placement

Min-max has the advantage of being computationally cheap and insensitive to errors, but it requires a good constellation of anchors; in particular, Savvides et al. recommend to place the anchors at the edges of the network [19]. If the anchors cannot be placed and are uniformly distributed across the network, the accuracy of the node positions at the edges is rather poor. Figure 9 illustrates this problem graphically. We applied Min-max and Lateration to the example network presented in Figure 1. In the case of Min-max, all nodes that lie outside the convex envelope of the four anchor nodes are drawn inwards, yielding considerable errors (indicated by the dashed lines); the nodes within the envelope are located adequately. Lateration, on the other hand, performs much better. Nodes at the edges are located less accurately than interior nodes, but the magnitude of and variance in the errors is smaller than for Min-max.

The differences in sensitivity to anchor placement between Lateration and Min-max can be considerable. For instance, for DV-hop/Lateration in the standard scenario, the average position accuracy degrades from 43% to 77%, when anchors are randomly distributed instead of the grid-based placement. The accuracy of DV-hop/Lateration also degrades, but only from 42%to 54%.

5 Discussion

Now that we know the behavior of the individual phase 1 and 2 components, we can turn to the performance effects of concatenating both phases, followed by applying Refinement in Phase 3. We

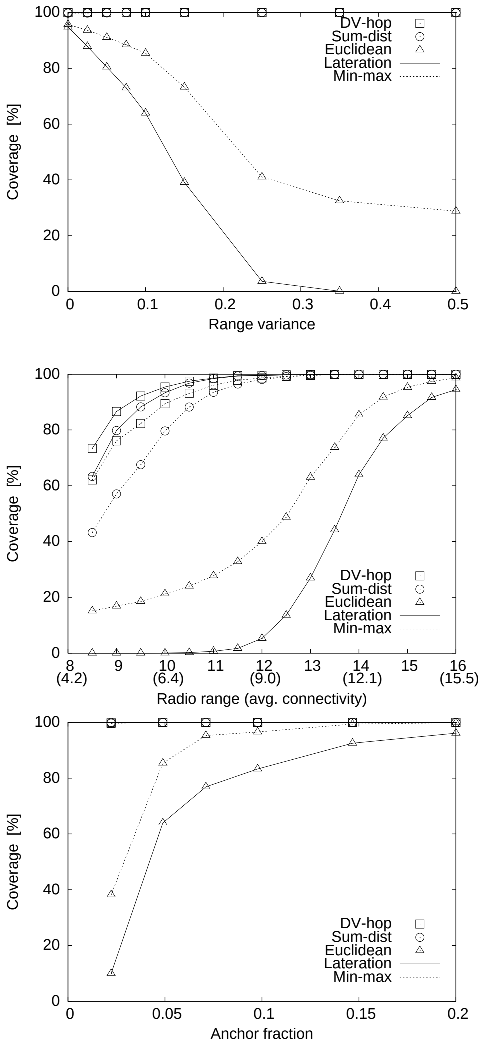

Figure 10: Coverage of Phase 1/2 combinations.

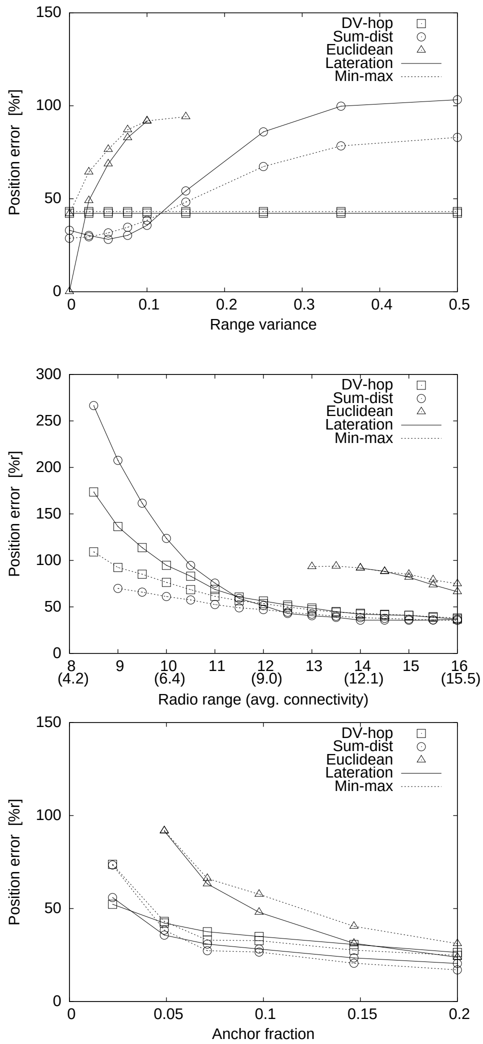

Figure 11: Accuracy of Phase 1/2 combinations.

will study the sensitivity of various combinations to connectivity, anchor fraction, and range errors using both the resulting position error and coverage.

5.1 Phases 1 and 2 combined

Combining the three Phase 1 alternatives (Sum-dist, DV-hop, and Euclidean) with the two Phase 2 alternatives (Lateration and Min-max) yields a total of six possibilities. We will analyze the differences in terms of coverage (Figure 10) and position accuracy (Figure 11). When fine-tuning localization algorithms, the trade-off between accuracy and coverage plays an important role; dropping difficult cases increases average accuracy at the expense of coverage.

Coverage

Figure 10 shows the coverage of the six Phase 1/Phase 2 combinations for varying range error (top), radio range (middle), and anchor fraction (bottom). The solid lines denote the Lateration variants; the dashed lines denote the Min-max variants. The first observation is that Sum-dist and DV-hop are able to determine the range to enough anchors to position all the nodes, except in cases when the radio range is small (≤ 11), or equivalently when the connectivity is low (≤ 7.5). In such sparse networks, Lateration provides a slightly higher coverage than Min-max. This is caused by the sanity check on the residue. A consistent set of anchor positions and distance estimates leads to a low residue, but the reverse does not hold. Occasionally if Lateration is used with an inconsistent set, an outlier is produced with a small residue, which is accepted. Min-max does not suffer from this problem because the positions are always constrained by the bounding boxes and thus cannot produce such outliers. Lateration's higher coverage results in higher errors, see the accuracy curves in Figure 11.

The second observation is that Euclidean has great difficulty in achieving a reasonable coverage when conditions are non-ideal. The combination with Min-max gives the highest coverage, but even that combination only achieves acceptable results under ideal conditions (range variance $\leq$ 0.1, connectivity $\geq$ 15, anchor fraction $\geq$ 0.1). The reason for Euclidean's poor coverage is twofold. First, the triangles used to propagate anchor distances are checked for validity (see Section 2.2); this constraint becomes more strict as the range variance increases, hence the significant drop in coverage. Second, Euclidean can only forward anchor distances if enough neighbors are present (see Section 4.1) resulting in many nodes "locating" only one or two anchors. Lateration requires at least three anchors, but Min-max does not have this requirement. This explains why the Euclidean/Min-max combination yields a higher coverage. Again, the price is paid in terms of accuracy (cf. Figure 11).

Accuracy

Figure 11 gives the average position error of the six combinations under the same varying conditions as for the coverage plots. To ease the interpretation of the accuracies we filtered out anomalous cases whose coverage is below 50%, which mainly concerns Euclidean's results. The most striking observation is that the Euclidean/Lateration combination clearly outperforms the others in the absence of range errors: 0% error versus at least 29% (Sum-dist/Min-max). This follows from the good performance of both Euclidean and Lateration in this case (see Section 4). The downside is that both components were also shown to be very sensitive to range errors. Consequently, the average position error increases rapidly if noise is added to the range estimates; at just 2% range variance, Euclidean/Lateration looses its advantage over the Sum-dist/Min-max combination. When the range variance exceeds 10%, DV-hop performs best. In this scenario DV-hop achieves comparable accuracies for both Lateration and Min-max. Which Phase 2 algorithm is most appropriate depends on anchor placement, and whether the higher computation cost of Lateration is important.

Notice that Sum-dist/Lateration actually becomes more accurate when a small amount of range variance is introduced, while the errors of Sum-dist/Min-max increase. This matches the results found in sections 4.1 and 4.2. Adding a small range error causes Sum-dist to yield more accurate distance estimates (cf. Figure 4). Lateration benefits greatly from a reduced bias, but Min-max is not that sensitive and even deteriorates slightly (cf. Figure 7). The combined effect is that Sum-dist/Lateration benefits from small range errors; Sum-dist/Min-max does not show this unexpected behavior.

All six combinations are quite sensitive to the radio range (connectivity). A minimum connectivity of 9.0 is required (at radio range 12) for DV-hop and Sum-dist, in which case Sum-dist slightly outperforms DV-hop and the difference between Lateration and Min-max is negligible. Euclidean does not perform well because of the 10% range variance in the standard scenario.

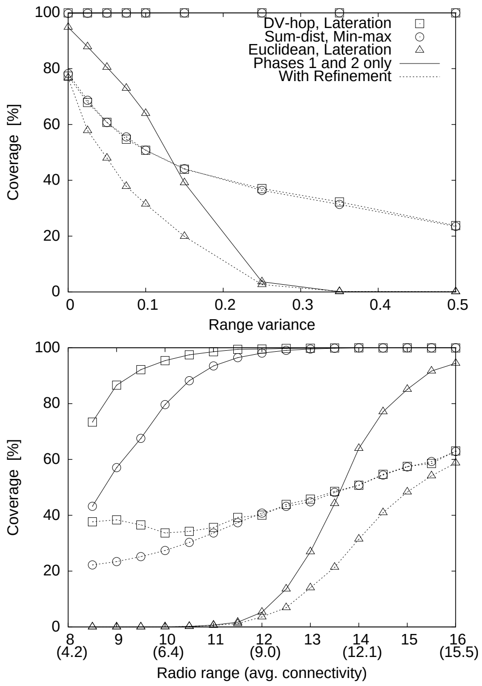

Figure 12: Coverage after Refinement.

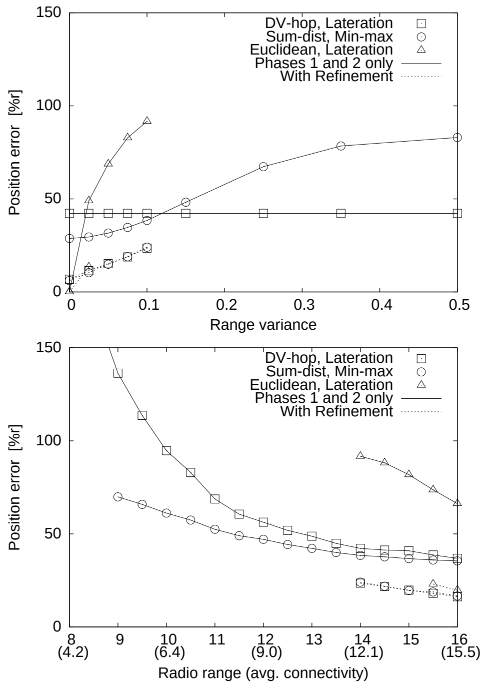

Figure 13: Accuracy after Refinement.

The sensitivity to the anchor fraction is quite similar for all combinations. More anchors ease the localization task, especially for Euclidean, but there is no hard threshold like for the sensitivity to connectivity.

5.2 Phase 3: Refinement

For brevity we do not report the effects of refining the initial positions produced by all six phase 1/2 combinations, but limit the results to the three combinations proposed in the original papers: Sum-dist/Min-max, Euclidean/Lateration, and DV-hop/Lateration (cf. Table 1). Figure 12 shows the coverage with (solid lines) and without (dashed lines) Refinement for the three selected combinations. Figure 13 shows the average position error, but only if the coverage exceeds 50%.

The most important observation is that Refinement dramatically reduces the coverage for all combinations. For example, in the standard case (10% range variance, radio range 14, and 5% anchors) the coverage for Sum-dist/Min-max and DV-hop/Lateration drops from 100% to a mere 51%. For the nodes that are not rejected Refinement results in a better accuracy: the average error decreases from 42% to 23% for DV-hop, and from 38% to 24% for Sum-dist. Other tests have revealed that Refinement does not only improve accuracy by merely filtering out bad nodes; the initial positions of good nodes are improved as well. A second observation is that Refinement equalizes the performance by Sum-dist and DV-hop. As a consequence the simpler Sum-dist is to be preferred in combination with Refinement to save on computation and communication.

Table 2: Average number of messages per node.

Type	Sum-dist	DV-hop	Euclidean
Flood	4.3	2.2	3.5
Calibration	-	2.6	-
Refinement	32	29	20

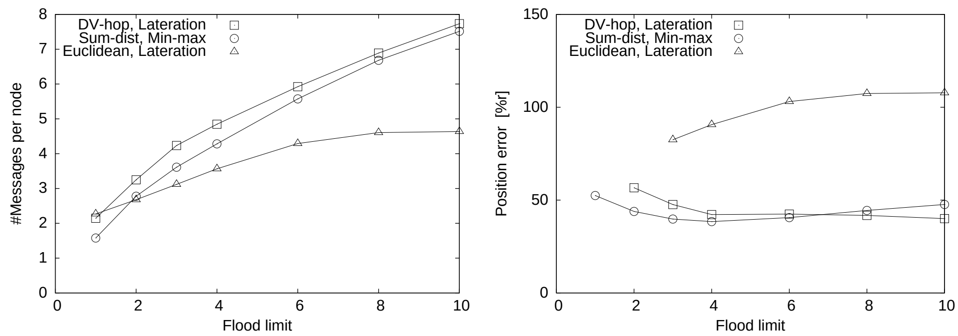

Figure 14: Sensitivity to flood limit.

5.3 Communication cost

The network simulator maintains statistics about the messages sent by each node. Table 2 presents a breakdown per message type of the three original localization combinations (with Refinement) on the standard scenario.

The number of messages in Phase 1 (Flood + Calibration) is directly controlled by the *flood limit* parameter, which is set to 4 by default. Figure 14 shows the message counts in Phase 1 for various flood limits. Note that Sum-dist and DV-hop scale almost linearly; they level off slightly because information on multiple anchors can be combined in a single message. Euclidean, on the other hand, levels off completely because of the difficulties in propagating anchor distances, especially along long paths.

For Sum-dist and DV-hop we expect nodes to transmit a message per anchor. Note, however, that for low flood limits the message count is higher than expected. In the case of DV-hop, the count also includes the calibration messages. With some fine-tuning the number of calibration messages can be limited to one, but the current implementation needs about as many messages as the flooding itself. A second factor that increases the number of messages for DV-hop and Sum-dist is the update information to be sent when a shorter path is detected, which happens quite frequently for Sum-dist. Finally, all three algorithms are self-organizing and nodes send an extra message when discovering a new neighbor that needs to be informed of the current status.

Although the flood limit is essential for crafting scalable algorithms, it affects the accuracy, see the bottom graph in Figure 14. Note that using a higher flood limit does not always improve accuracy. In the case of Sum-dist, there is a trade-off between using few anchors with accurate distance information, and using many anchors with less accurate information. With DV-hop, on the other hand, the distance estimates become more accurate for longer paths (last-hop effect, see Section 4.1). Euclidean's error only increases with higher flood limits because it starts with a low coverage, which also increases with higher flood limits. DV-hop and Sum-dist reach almost 100% coverage at flood limits of 2 (Min-max) or 3 (Lateration).

With the flood limit set to 4, the nodes send about 4 messages during Phase 1 (cf. Table 2). This

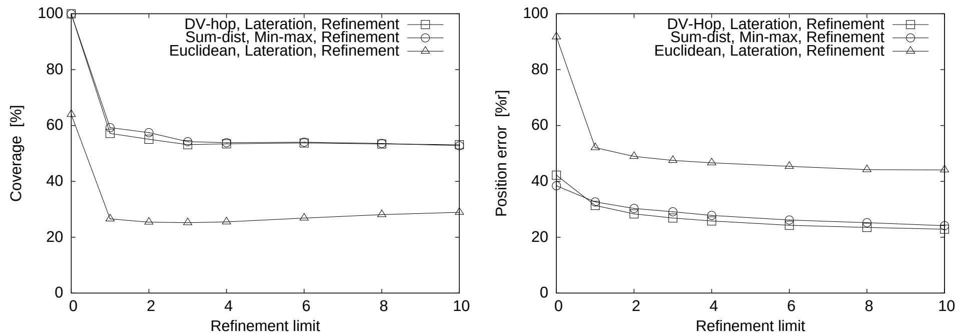

Figure 15: Effect of Refinement limit.

is comparable to the three messages needed by a centralized algorithm: set up a spanning tree, collect range information, and distribute node positions. Running Refinement in Phase 3, on the other hand, is extremely expensive, requiring 20 (Euclidean) to 32 messages (Sum-dist). The problem is that Refinement takes many iterations before local convergence criteria decide to terminate. We added a limit to the number of Refinement messages a node is allowed to send. The effect of this is shown in Figure 15. A Refinement limit of 0 means that no refinement messages are sent, and Refinement is skipped completely.

The position errors in Figure 15 show that most of the effect of Refinement takes place in the first few iterations, so hard limiting the iteration count is a valid option. For example, the accuracy obtained by DV-hop without Refinement is 42% and it drops to 28% after two iterations; an additional 4% drop can be achieved by waiting until Refinement terminates based on the local stopping criteria, but this requires another 27 messages (29 in total). Thus the communication cost of Refinement can effectively be reduced to less than the costs for Phase 1. Nevertheless, the poor coverage of Refinement limits its practical use.

5.4 Recommendations

From the previous discussion it follows that no single combination of phase 1, 2, and 3 alternatives performs best under all conditions; each combination has its strengths and weaknesses. The results presented in Section 5 follow from changing one parameter (radio range, range variance, and anchor fraction) at a time. Since the sensitivity of the localization algorithms may not be orthogonal in the three parameters, it is difficult to derive general recommendations. Therefore, we conducted an exhaustive search for the best algorithm in the three-dimensional parameter space. For readability we do not present the raw outcome, a $6 \times 6 \times 5$ cube, but show a two-dimensional slice instead. We found that the localization algorithms are the least sensitive to the anchor fraction, so Table 3 presents the results of varying the radio range and the range variance, while keeping the anchor fraction fixed at 5%. In each case we list the algorithm that achieves the best accuracy (i.e. the lowest average position error) under the condition that its coverage exceeds 50%. Since Refinement often results in very poor coverage, we only examine Phases 1 and 2 here.

The exhaustive parameter search, and basic observations about Refinement, lead to the following recommendations:

1. **Euclidean** should always be used in combination with Lateration, but only if distances can be measured very accurately (range variance $< 2\%$) and the network has a high connectivity (≥ 12). When the anchor fraction is increased, Euclidean captures some more entries in the left-upper corner of Table 3, and the conditions on range variance and connectivity can be

Table 3: Comparison; anchor fraction fixed at 5%, no Refinement.

range variance	Radio range (avg. connectivity)				
	16 (15.5)	14 (12.1)	12 (9.0)	10 (6.4)	8 (4.2)
0	Euclidean/Lateration	Euclidean/Lateration	Sum-dist/Min-max	Sum-dist/Min-max	DV-hop/Lateration
0.025	Sum-dist/Lateration	Sum-dist/Min-max	Sum-dist/Min-max	Sum-dist/Min-max	DV-hop/Lateration
0.05	Sum-dist/Lateration	Sum-dist/Lateration	Sum-dist/Min-max	Sum-dist/Min-max	DV-hop/Lateration
0.1	Sum-dist/Min-max	Sum-dist/Lateration	Sum-dist/Min-max	Sum-dist/Min-max	DV-hop/Lateration
0.25	DV-hop/Lateration	DV-hop/Min-max	DV-hop/Min-max	DV-hop/Min-max	DV-hop/Lateration
0.5	DV-hop/Lateration	DV-hop/Min-max	DV-hop/Min-max	DV-hop/Min-max	DV-hop/Lateration

relaxed slightly. Nevertheless, the window of opportunity for Euclidean/Lateration is rather small.

2. **DV-hop** should be used when there are no or poor distance estimates, for example, those obtained from the signal strength (cf. the bottom rows in Table 3). Our results show that DV-hop outperforms the other methods when the range variance is large ($>$ 10% in this slice). The presence of Lateration in the last column, that is, with a very low connectivity, is an artifact caused by the filtering on coverage. DV-hop/Min-max has a coverage of 49% in this case (versus 56% for DV-hop/Lateration), but also a much lower error. Regarding the issue of combining DV-hop with Lateration or Min-max, we observe that overall, Min-max is the preferred choice. Recall, however, its sensitivity for anchor *placement* leading to large errors at the edges of the network.

3. **Sum-dist** performs best in the majority of cases, especially if the anchor fraction is (slightly) increased above 5%. Increasing the number of anchors reduces the average path length between nodes and anchors, limiting the accumulation of range errors along multiple hops. Except for a few corner cases, Sum-dist performs best in combination with Min-max. In scenarios with very low connectivity and a low anchor fraction, Sum-dist tends to overestimate the distance significantly. Therefore DV-hop performs better in the far right column of Table 3.

4. **Refinement** can be used to improve the accuracy of the node positions when the range estimates between neighboring nodes are quite accurate. The best results are obtained in combination with DV-hop or Sum-dist, but at a significant (around 50%) drop in coverage. This renders the usage of Refinement questionable, despite its the modest communication overhead.

A final important observation is that the localization problem is still largely unsolved. In ideal conditions Euclidean/Lateration performs fine, but in all other cases it suffers from severe coverage problems. Although Refinement uses the extra information of the many neighbor-to-neighbor ranges and reduces the error, it too suffers from coverage problems. Under most conditions, there is still significant room for improvement.

6 Conclusions

This chapter addressed the issue of localization in ad-hoc wireless sensor networks. From the known localization algorithms specifically proposed for sensor networks, three approaches were selected that meet basic requirements of self-organization, robustness, and energy-efficiency: Ad-hoc positioning [14], Robust positioning [17], and N-hop multilateration [19]. Although these three algorithms were developed independently, they share a common structure. We were able to identify a generic, 3-phase approach to determine the individual node positions consisting of the steps below:

1. Determine the distances between unknowns and anchor nodes.

2. Derive for each node a position from its anchor distances.

3. Refine the node positions using information about the range to, and positions of, neighboring nodes.

We studied three Phase 1 alternatives (Sum-dist, DV-hop, and Euclidean), two Phase 2 alternatives (Lateration and Min-max) and an optional Refinement procedure for Phase 3. To this end the discrete event simulator developed by Savarese et al. [17] was extended to allow for the execution of an arbitrary combination of alternatives.

Section 4 dealt with Phase 1 and Phase 2 in isolation. For Phase 1 alternatives, we studied the sensitivity to range errors, connectivity, and fraction of anchor nodes (with known positions). DV-hop proved to be stable and predictable, Sum-dist and Euclidean showed tendencies to under estimate the distances between anchors and unknowns. Euclidean was found to have difficulties in propagating distance information under non-ideal conditions, leading to low coverage in the majority of cases. The results for Phase 2 showed that Lateration is capable of obtaining very accurate positions, but also that it is very sensitive to the accuracy and precision of the distance estimates. Min-max is more robust, but is sensitive to the placement of anchors, especially at the edges of the network.

In Section 5 we compared all six phase 1/2 combinations under different conditions. No single combination performs best; which algorithm is to be preferred depends on the conditions (range errors, connectivity, anchor fraction and placement). The Euclidean/Lateration combination [14] should be used only in the absence of range errors (variance $< 2\%$) and requires a high node connectivity. The DV-hop/Min-max combination, which is a minor variation on the DV-hop/Lateration approach proposed in [14] and [17], performs best when there are no or poor distance estimates, for example, those obtained from the signal strength. The Sum-dist/Min-max combination [19] is to be preferred in the majority of other conditions. The benefit of running Refinement in Phase 3 is considered to be questionable since in many cases the coverage dropped by 50%, while the accuracy only improved significantly in the case of small range errors. The communication overhead of Refinement was shown to be modest (2 messages per node) in comparison to the controlled flooding of Phase 1 (4 messages per node).

Future work Regarding the future, the ultimate distributed localization algorithm is yet to be devised. Under ideal circumstances Euclidean/Lateration performs fine, but in all other cases there is significant room for improvement. Furthermore, additional effort is needed to bridge the gap between simulations and real-world localization systems. For instance, we need to gather more data on the actual behavior of sensor nodes, particularly with respect to physical effects like multipath, interference, and obstruction.

Acknowledgements

This work was first published in Elsevier Computer Networks [12]. We thank Elsevier for giving us permission to reproduce the material. We also thank Andreas Savvides and Dragos Niculescu for their input and for sharing their code with us.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[2] S. Atiya and G. Hager. Real-time vision-based robot localization. *IEEE Trans. on Robotics and Automation*, 9(6):785–800, 1993.

[3] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location tracking system. In *Infocom*, volume 2, pages 575–584, Tel Aviv, Israel, March 2000.

[4] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, October 2000.

[5] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Cluster Computing*, 5(2):157–167, April 2002.

[6] J. Chen, K. Yao, and R. Hudson. Source localization and beamforming. *IEEE Signal Processing Magazin*, 19(2):30–39, 2002.

[7] L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *IEEE Infocom 2001*, Anchorage, AK, April 2001.

[8] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Maui, Hawaii, October 2001.

[9] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *ACM International Conference on Mobile Computing and Networking (Mobicom)*, pages 81–95, San Diego, CA, September 2003.

[10] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[11] J. Hightower, R. Want, and G. Borriello. SpotON: An indoor 3D location sensing technology based on RF signal strength. UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA, February 2000.

[12] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Elsevier Computer Networks*, 43:499–518, 2003.

[13] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics and Automation*, 7(3):376–382, 1991.

[14] D. Niculescu and B. Nath. Ad-hoc positioning system. In *IEEE GlobeCom*, November 2001.

[15] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *6th ACM Int. Conf. on Mobile Computing and Networking (Mobicom)*, pages 32–43, Boston, MA, August 2000.

[16] V. Ramadurai and M. Sichitiu. Localization in wireless sensor networks: A probabilistic approach. In *Int. Conf. on Wireless Networks (ICWN)*, pages 275–281, Las Vegas, NV, June 2003.

[17] C. Savarese, K. Langendoen, and J. Rabaey. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX technical annual conference*, pages 317–328, Monterey, CA, June 2002.

[18] A. Savvides, C.-C. Han, and M. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *7th ACM Int. Conf. on Mobile Computing and Networking (Mobicom)*, pages 166–179, Rome, Italy, July 2001.

[19] A. Savvides, H. Park, and M. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *First ACM Int. Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 112–121, Atlanta, GA, September 2002.

[20] R. Tinos, L. Navarro-Serment, and C. Paredis. Fault tolerant localization for teams of distributed robots. In *IEEE Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 1061–1066, Maui, HI, October 2001.

[21] A. Varga. The OMNeT++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, June 2001.

[22] K. Whitehouse and D. Culler. Callibration as parameter estimation in sensor networks. In *First ACM Int. Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 59–67, Atlanta, GA, September 2002.

[23] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad-hoc routing. In *7th ACM International Conference on Mobile Computing and Networking (Mobicom)*, pages 70–84, Rome, Italy, 2001.

[24] J. Zhao and R. Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Net-

works. In *First Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 1–13, November 2003.