RLISE: Relative Location with Incomplete Stationary Emitters

Roy Friedman  Guy Korland
Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000
Israel

February 8, 2005

Abstract

Geographic location information of nodes in ad hoc networks is useful for improved routing, lookup services, and location aware applications. This paper proposes a scheme that enables nodes of an ad-hoc network to discover their geographical location with good accuracy when only a few nodes know a-priori their exact location (e.g., through GPS) and the only additional information available to other nodes is the relative distance from their immediate neighbors (e.g., based on the quality of the signal they receive). The proposed scheme is investigated using simulations to explore its efficiency as a function of the number of nodes that have accurate knowledge of their position (we refer to such nodes as stationary emitters), the density of the network, and the impact of errors in distance measurements on the accuracy of this scheme.

Keywords: Ad-Hoc Networks, Ah-Hoc Positioning, Relative Location.
1 Introduction

Ad-hoc networking technology enables users to spontaneously form a dynamic wireless communication network [15]. Thus, an ad-hoc network can be used to provide devices with continuous network connection, even when a device is out of an infrastructure based network scope. The two most common examples of ad-hoc networks include Mobile Ad-Hoc Networks [18] (MANET) and sensor networks [16]. Examples of the possible uses of ad-hoc networks include e.g., soldiers in a battlefield, emergency disaster relief personnel, networks of laptops, seismic sensor networks, and floor control and monitoring networks.

As been demonstrated, various basic services and applications can greatly benefit from positioning knowledge. For example, it was shown that geographical based routing can be efficient and scalable, e.g., in [8, 9, 20]. Moreover, in sensor networks, knowing the location of a node is vital for being able to correctly analyze and respond to sensors’ readings.

There are several existing techniques that enable nodes to learn their position. The simplest approach is to equip every node with a GPS receiver. However, this option is relatively expensive, it is not always available (for instance, most laptops and PDAs sold today are still not equipped with GPS), and GPS requires line of sight, which means that it does not always work in an urban environment, and in particular inside buildings. Another known option is too assume the existence of several base stations, also known as stationary emitters or landmarks, who know their exact position and from which other nodes can learn their own location by using triangulation principles. These schemes, however, assume that each node is covered by the transmission range of at least three stationary emitters. This requires a large number of stationary emitters and is also sensitive to interference caused by concrete walls etc.

In order to overcome the above obstacles, several ad-hoc positioning systems (APS) have been proposed, e.g., [1, 3, 4, 6, 7, 10, 12, 13, 14, 17]. These schemes can be largely categorized as DV-hop propagation methods, Euclidean and DV-radial methods, and DV-position methods. In the DV-hop propagation method, it is assumed that each node knows the shortest path to the landmark nodes (or stationary emitters), which know their position. The landmarks publish the estimated physical distance for each hop, which allows nodes to learn their estimated distance from each landmark, thereby also compute their estimated location. In the Euclidean and DV-radial methods, nodes are assumed to be able to measure their distance or angle from landmark nodes. Once they have this information, they can use triangulation and trilateration techniques to compute their own location. Finally, in DV-position methods nodes combine both distance and angle measurements for improved accuracy. Notice, though, that in these methods, a node can only learn its location if it is within the transmission range of three other nodes that already know their location.

Contribution of this Work: In this paper we propose a novel method for achieving positioning knowledge based on distance measurements. In our scheme, nicknamed RLISE, we allow for a gradual iterative learning process in which nodes can eventually learn their location even if no node is initially placed within the transmission range of three stationary emitters (or landmarks). Thus, this enables positioning with much fewer initial landmarks.

As in existing techniques, RLISE is based on simple geometric principles, but it also employs partial location gossiping in order to enhance each node’s perception of its own location in an iterative process. Thus, nodes can learn their position much faster than in the Euclidean methods, and moreover, there are situations in which the Euclidean method does not converge while our method does. The distance measurement capabilities we require can be obtained by employing Time of Arrival (ToA) [10] and Received Signal Strength Indicator (RSS) [1] techniques. Thus, RLISE does not require any special hardware.

Also, we would like to stress that all computations are local, and do not rely on any central entity.
Another feature of RLISE is that when the network connectivity is not sufficient, nodes can still maintain partial information on where they might be. For example, for a while, a node might only be able to tell that it is located somewhere on a circle, or that it is definitely in one of two possible locations. This information might still be useful for many applications, and in particular, once the network connectivity improves, it allows for quick learning of the exact location. These aspects are elaborated later in this paper.

Finally, the paper includes an analysis of our protocol, RLISE. In particular, we discuss in detail the benefits and the limitations of RLISE, and present extensive simulations. In these simulations we explore the performance of RLISE vs. the number of stationary emitters, the density of the network, the impact of mobility, and its sensitivity to errors in distance measurements. These measurements generally confirm our assumptions about RLISE’s behavior.

**Paper’s road-map:** Section 2 discusses related works. Section 3 presents the model and basic assumptions. Section 4 presents our RLISE positioning scheme and discusses its benefits as well as its limitations and some extensions. The simulation results are presented in Section 5. We conclude with a discussion in Section 6.

## 2 Related Works

There are several methods for obtaining geographic localization. The Global Positioning System (GPS) [5] is probably the most commonly used for outdoor localization. GPS relies on satellites to achieve triangulation. Yet, line of sight access to satellites is often not available indoors, which largely limits the use of GPS to being outside.

Several attempts to mimic the use of satellites indoor were made using full cover of stationary antennas. For example, the RADAR system [1] is a radio-frequency (RF) based system for locating and tracking users inside buildings. RADAR uses multiple base stations positioned to provide overlapping coverage in the area of interest, using received radio signal strength (RSS) to determine an object’s location. Another example is the LEASE [10] system. LEASE is composed of three different stationary devices that should be deployed across the building. In LEASE, all positioning computations are done by a global server.

There are also several works on learning location information using cooperative sharing between mobile nodes. For example, Savarese, Rabaey and Beutel proposed the Assumption Base Coordinates (ABC) algorithm to calculate relative positions of nodes based on range measurements between the nodes in [17]. This algorithm determines the location of unknown nodes one at a time in the order they establish communication. Yet, this approach makes an assumption that all nodes are aware of each other, and therefore can apply messages in a specific order.

A variation of the ABC protocol called TERRAIN is also presented in [17]. The main differences between TERRAIN and RLISE is that in TERRAIN the computation cannot proceed with partial information. Also, in TERRAIN, anchor nodes start calculating their locations in semi-virtual coordinates. When the computation discovers another coordinate system, a correction phase is started until eventually the system converges to a single coordinate system.

SpotON [4] provides absolute and relative location capability for sensors networks using tags based on RSS. However, in SpotON, all computations are done by a global server.

Another method that helps determine nodes positions is the Angle of Arrival (AOA), which is used in several Ad Hoc Positioning System (APS) [20, 12]. These protocols offer a hybrid mixture of two major concepts: distance vector (DV) routing, and beacon based positioning (GPS). A mechanism for positioning...
learning that is based on region intersections was proposed in [2]. By this scheme, nodes maintain an estimate on the maximal transmission range, and the region(s) it might be in. Nodes exchange this information, and when a node receives such a message, it calculate the intersection between the sender’s regions and its own, based on the maximal transmission range. This method also allows for gradual learning of the location, with improved accuracy as more messages are received. On the other hand, it is sensitive to errors in the assumed maximal transmission range.

3 Model and Basic Assumptions

We assume a collection of nodes (also referred to as devices), each equipped with a wireless transmitter and receiver. The transmitter of each node $p_i$ has a bounded transmission range $r_i$. Thus, when a node $p_i$ sends a message, all other nodes whose distance from $p_i$ is less than $r_i$ may potentially receive this message. Yet, not all such nodes always receive each message sent by $p_i$, e.g., due to interference with other messages, obstacles such as concrete walls, etc.

We refer to two nodes that are in the transmission range of each other as neighbors. We further assume that when a node $p_j$ receives a message from another node $p_i$, then $p_j$ can calculate its distance from the sender $p_i$, e.g., by examining the received signal strength indicator (RSS) [1] or by Time of Arrival (TOA) [10]. In RSS, if the energy in which a signal is sent is known, then the distance can be computed based on the energy drop between sending and receiving the event. In TOA, on the other hand, the distance is calculated based on the latency of a ping message. Such capabilities are considered reasonable [1, 10].

Moreover, we assume that some of the nodes have precise a-priori knowledge of their geographical location. These nodes are called stationary emitters. Other nodes have no a-priori knowledge of their position. However, nodes can transmit to each other messages containing their location, if it is known, or a summary of possible locations for them. For example, $p_i$ might know that it is somewhere on a circle, or that it is in one of two locations (but not in any other location), etc. Also, nodes can indicate an error estimate for their location.

4 Calculating Relative Positions

4.1 The Basic Protocol for Dynamic network

In this section, we present the basic algorithm for calculating nodes’ position. For clarity of presentation, the presentation deals with two dimensions only. However, extending the protocol to three dimensions is obvious. Also, initially we assume that each node can detect its distance from a sender of a message it receives accurately and the clocks of all nodes are tightly synchronized. Later, we relax these two assumptions.

In RLISE, a node can be in one of the four states: Accurate, SemiAccurate, Circle, and Unknown. Accurate indicates that the node knows its exact location represented by coordinates $(X, Y)$. SemiAccurate indicates that the node knows that it must be in one of two optional locations, represented by two coordinates $(X_1, Y_1)(X_2, Y_2)$. Circle indicates that the node knows that it is somewhere on a continuous circle represented by the middle coordinates and radius $(X, Y, R)$.

1It is known that this simplistic transmission disk model is not completely realistic. However, it simplifies the definitions and the presentation. Moreover, our protocol does not depend on it and our simulation results also use a more realistic model.

2Yet, let us note that neither method is perfect. For example, the amount of energy used to send a signal varies over time in WiFi, often without advanced warning. Also, ToA’s accuracy depends on the clocks drift between nodes.
Figure 1: Relative location finding - trivial cases  
Figure 2: Combining two SemiAccurate states

Note that location information may become obsolete due to mobility. To overcome this problem, we devise our protocol to work in asynchronous rounds. Each node that is not a stationary emitter starts a round in the Unknown state, which is maintained in the STATE variable, and tries to gradually upgrade its state during the round. This is facilitated by having each node maintain a variable cur_round, which is initialized to 0. The cur_round and the STATE variables are updated as described below.

Stationery emitters start in the Accurate state while all other nodes start in the Unknown state. Each stationery emitter periodically increases its round number, broadcast its state and round number to all its neighbors, and then sleeps for a given time period. Non-stationery emitters wait for incoming messages: Whenever a node \( p_i \) receives a state message from another node \( p_j \), it checks whether the newly received round number is larger than cur_round. If it is, then \( p_i \) sets its state to Unknown and update its cur_round accordingly.

Following this, \( p_i \) computes its new state according to the geometric possible solution. For example, some trivial cases are showed in Figure 1. Here, if a node \( p_i \) that is in the Unknown state receives an Accurate message from another node \( p_j \), then \( p_i \) can upgrade its state to Circle with \( p_j \)'s location as the center of the circle and the distance between them as the radius. Finally, if indeed the state of the node was changed as a result of processing this message, than it broadcasts it updated STATE and cur_round variables to its neighbors. The full pseudo-code is listed under Algorithm 1 and Algorithm 2.

4.2 Optimizations

Often, in a sensor network, and even in a MANET, the region’s boundaries are known. In these cases, it is possible to eliminate potential locations that are outside the network region. This helps improving the accuracy of the location calculations, as illustrated in Figure 6.

An important variant on the basic protocol is allowing a process to hold more than one potential circle in the Circle state and more than two points in the SemiAccurate state. That is, suppose a node \( p \) that is in the
Algorithm 1 Relative Location Algorithm for non-Stationary Emitters

OnReceive( $r_{state}$, round)
1: $STATE \in$
   \{Accurate($x$, $y$), SemiAccurate($x_1$, $y_1$, $x_2$, $y_2$),
   Circle($x$, $y$, $r$), Unknown\}
2: if $cur\_round < round$ then
3: $STATE := Unknown$
4: $cur\_round := round$
5: else if $cur\_round > round$ then
6: drop message
7: end if
8: if $r_{state} = Accurate(-)$ then
9: if $STATE = SemiAccurate(-)$ then
10: $STATE := Semi2Accurate(STATE, r_{state}, d)$ \{See Figure 3\}
11: else if $STATE = Circle(-)$ then
12: $STATE := Circle2Semi(STATE, r_{state}, d)$ \{See Figure 3\}
13: else if $STATE = Unknown$ then
14: $STATE := Unknown2Circle(STATE, r_{state}, d)$ \{See Figure 3\}
15: end if
16: else if $r_{state} = SemiAccurate$ then
17: if $STATE = SemiAccurate(-)$ then
18: $STATE := SemiPlusSemi(STATE, r_{state}, d)$ \{See Figure 4\}
19: else if $STATE = Circle(-)$ then
20: $STATE := CirclePlusSemi(STATE, r_{state}, d)$ \{See Figure 4\}
21: end if
22: else if $r_{state} = Circle \land STATE=SemiAccurate$ then
23: $STATE := SemiPlusCircle(STATE, r_{state}, d)$ \{See Figure 4\}
24: end if
25: if $STATE$ changed then
26: Broadcast $STATE$, $cur\_round$
27: end if

Algorithm 2 Relative Location Algorithm for Stationary Emitters
1: $STATE := Accurate(x, y)$
2: $round := 1$
3: loop
4: Broadcast $STATE$, $round$
5: SLEEP $\Delta t$
6: $round := round + 1$
7: end loop
Semi2Accurate (SemiAccurate(x₁, y₂, x₂, y₂), Accurate(x', y'), d)
if (x₁, y₁) is on the circle (x', y', d) then
    return Accurate(x₁, y₁)
else if (x₂, y₂) is on the circle (x', y', d) then
    return Accurate(x₂, y₂)
else
    return SemiAccurate(x₁, y₁, x₂, y₂)
endif

Circle2Semi (Circle(x, y, r), Accurate(x', y'), d)
if the circles (x, y, r) and (x', y', d) intersect then
    let (x₁, y₁) and (x₂, y₂) be the two intersection points of (x, y, r) and (x', y', d)
    return SemiAccurate(x₁, y₁, x₂, y₂)
else
    return Circle(x, y, r)
endif

Unknown2Circle (Unknown, Accurate(x', y'), d)
return Circle(x', y', d)

Figure 3: Simple new state calculations - see drawings in Figure 1

SemiPlusSemi (SemiAccurate(x₁, y₂, x₂, y₂), SemiAccurate(x₁', y₁', x₂', y₂'), d)
if dist((x₁, y₁), (x₁', y₁')) = d or dist((x₂, y₂), (x₂', y₂')) = d then
    return Accurate(x₁, y₁)
else if dist((x₂, y₂), (x₁', y₁')) = d or dist((x₂, y₂), (x₂', y₂')) = d then
    return Accurate(x₂, y₂)
else
    return SemiAccurate(x₁, y₁, x₂, y₂)
endif

CirclePlusSemi (Circle(x, y, r), SemiAccurate(x₁', y₁', x₂', y₂'), d)
if there is a single point on the circle (x, y, r) whose distance from either (x₁', y₁') or (x₂', y₂') is d then
    let (x₃, y₃) be that point
    return Accurate(x₃, y₃)
else if there are two points on the circle (x, y, r) whose distance from either (x₁', y₁') or (x₂', y₂') is d then
    let (x₃, y₃) and (x₄, y₄) be these points
    return SemiAccurate(x₃, y₃, x₄, y₄)
else
    return Circle(x, y, r)
endif

SemiPlusCircle (SemiAccurate(x₁, y₁, x₂, y₂), Circle(x', y', r'), d)
if only (x₁, y₁) is such that its distance from the circle (x', y', r) is d then
    return Accurate(x₁, y₁)
else if only (x₂, y₂) is such that its distance from the circle (x', y', r) is d then
    return Accurate(x₂, y₂)
else
    return SemiAccurate(x₁, y₁, x₂, y₂)
endif

Figure 4: More complex state calculations - see illustrations in Figures 2 and 5
Figure 5: Combining a Circle and SemiAccurate states

Unknown state hears a message from another node q that is in the SemiAccurate state. In this case, p can tell that it is in one of the circles whose centers are the two possible positions of q, but it does not know which of these circles. Thus, if we allow p to maintain both of them, then eventually by interacting with additional nodes, it might be able to rule out one of these circles. In turn, this can later even help q to rule out one of its possible locations and upgrade to the Accurate state. On the other hand, as we would like to keep the space and computation requirements of RLISE small, it is recommended to allow each node to keep at most 2-3 circles in the Circle state and at most 4-6 potential locations in the SemiAccurate state. This optimization is not incorporated into Algorithm 1 as presented above for clarity of presentation reasons. Yet, adding this to the code is obvious. In particular, our simulation measurements have explored this optimization, as reported below in Section 5.

Another possible optimization on the basic protocol is to collect several messages before applying them in Line 4 of the protocol. This way, it is possible to choose messages that provide the best location information. For example, ones that bring the state to Accurate within the least number of steps. If such messages do not exist, then it may be possible to choose messages that result in the two closest points in the SemiAccurate state. If the latter is not achievable, then one can pick the messages that produce the Circle state with the smallest radius. Another variant of this idea is to apply the first message that arrives, yet to keep the last few messages. Then, on each message arrival, to try to choose the best set of messages among the last saved ones, and apply these messages to recalculate the best possible location.

4.3 Unsynchronized Clocks

It is possible to eliminate the synchronized clocks assumptions simply by making the round numbers behave like logical timestamps [11]. That is, each time an emitter hears about a round r that is larger than its own by more than one, it assigns its current round number to be r. The details are obvious and are therefore emitted due to space limitations.
4.4 Calculating Relative Positions in Static Networks

The same algorithm can be applied to static networks but for such networks there is no need to periodically rebuild the position information. Specifically, the original algorithm can be reduced, removing lines 2-5 from Algorithm 1 and minimizing Algorithm 2 to line 4.

4.5 Handling Distance Estimation Errors

As can be noticed from the code of the algorithm, the correctness of the location estimates depends on the ability to accurately measure the distance from a sender. In this section, we explore the impact of errors in measuring this distance on the outcome of the protocol, and discuss heuristics to alleviate this problem. The most obvious problem in inaccurately measuring the distance from a sender is that a node will think that it is on a circle with a different radius than the real one. This can lead to finding incorrect circle intersection points, and eventually to an incorrect position.

In most situations, this process should lead to a bounded error propagation. In particular, if the distance measurement error is equally probable to be positive and negative, there is a good chance that it will more or less balance out. Yet, there is one pathological case in which the error can be quite large. Interestingly, this happens when a node is in the Circle state and it receives a message from another node, who is very close to the circle’s center. In this case, as illustrated in Figure 7, the calculated intersection points can be very far from the real ones.

A way to overcome this problem is to ignore such messages. That is, if a node is in the Circle state and it receives a message with a position that is very close to the center of its circle, it simply ignores this message. The rational here is that it is better to remain in a less accurate state, with the hope of later receiving another message that can upgrade the state to a more accurate one, than to adopt a message that is likely to result in a large position error. Another optimization in order to mitigate the propagation of errors is based on the assumption that the distance measurement error is proportional to the distance itself [3]. Thus, a node can collect a few messages before reacting to any of them, and then decide to adopt the ones that are sent from the closest nodes, while avoiding the false circle intersection mentioned in the previous paragraph.
4.6 Analysis and Limitations

One of the main benefits of our RLISE protocol is that all computations are simple and local. They rely on simple, efficiently computable, geometric principles. Also, all exchanged messages carry only a few integer values. These messages are exchanged only between immediate neighbors, and there is no need to forward messages along multiple hops. Moreover, the memory requirements of this protocol are also minimal, i.e., remembering the current state of the protocol, which can be represented as a handful of integers. Even in the extensions mentioned above to handle errors, at most a small number of recent messages need to be remembered. As becomes evident in our simulations detailed below, the vast majority of nodes learn their position within a very small number of message exchanges. Yet, RLISE does not require that any node will be within the transmission range of at least three stationary emitters.

RLISE works very well when nodes are either static, or when nodes move at a walking speed, typically up to 1-2 meters per second, which is much slower than the rate at which information propagates. However, when nodes move much faster, e.g., at the speed of cars and faster, their physical position changes rapidly. In this case, in order to prevent nodes from sending obsolete location information to other nodes, thereby contaminating the system with false location information, we need to set the $\Delta t$ timer of Algorithm 2 to a small value. Note also that most ad-hoc positioning systems suffer from the same problem so in that sense RLISE is no worse than other ad-hoc positioning approaches.

If the number of fast moving nodes is very small, and they are aware that they are moving very fast, we can simply disallow such nodes from sending their location information to others. Otherwise, in order for RLISE to work well with many very fast moving nodes, these fast moving nodes must be aware of the direction and speed that they are moving (e.g., using a compass and a speedometer). In this case, mobile nodes can simply add the offset from the position information they receive based on their movement since obtaining that knowledge. Interestingly, with this assumption in place, our protocol in fact lends itself easily to rapid mobility in the sense the all calculations are local, and thus even when a node moves fast, this does not require any type of global re-computation. Moreover, due to the gradual learning process of RLISE, rapid mobility can be used in these cases to obtain accurate location information even when the network is not fully connected!

Another limitation of RLISE is that not all nodes are guaranteed to eventually learn their exact location (although as shown in Section 5 below, above a certain density, this happens only to a tiny fraction of nodes). Similarly, when the distance estimates are not perfect, the computed location of a small fraction of the nodes might be far from their physical location (although most nodes compute a location close to their real one). At the same time, it always performs better then Euclidean methods.

4.7 Utilizing Inaccurate Location Estimates

As discussed before, many works have explored how to utilize exact location information. In this section we discuss how the SemiAccurate and Circle states can also be used in location based routing algorithms and services.

When considering geographical routing protocols, there are several alternatives. For example, if a node is in the Circle state and the radius of the circle is small, a node can publish its circle. Then, the message will be routed to the correct geographical region of the network. Once the message arrives near the actual target node, most geographical routing protocols maintain proactive neighborhood information that is sufficient to complete the forwarding to its destination. The same is true when a node is in the SemiAccurate state and both possible locations are close to each other, as they are likely to fall in the same region of the routing protocol. During our simulations we discovered that very often this was indeed the case.
If a node is in the *SemiAccurate* state and the possible locations are far, then the message can be routed to both locations. This way messages will still arrive quickly to the node, and will only require a redundancy factor of 2. When a node is in the *Circle* state with a moderate radius, it is possible to initially route the message to the closest point on the circle. From there, the message can be routed along the circle edge, until it reaches the vicinity of the target node.

When considering location based services such as location dependent information, again, if the possible locations of the *SemiAccurate* state are near each other, then the information can be generated, e.g., according to the middle position on the straight line connecting them. If the positions are far, then it is possible to generate the information relevant to both. If a node is in the *Circle* state with a small radius, then the information can be sent based on the middle of the circle. This way the information might not be as accurate as with perfect location knowledge, but still at a much higher quality than without any knowledge.

## 5 Results

Our performance evaluation was carried out by simulations. In each simulation, we assumed a simulation area of 100x100 meters, in which nodes are spread in a random manner with uniform distribution. Also, a certain percentage of the nodes act as stationary emitters, i.e., they know their exact position right from the start of the simulation. All other nodes are unaware of their location at the beginning of each simulation, but gradually learn it while executing Algorithm 1. In the simulations we varied the number of nodes between 20 and 160, and the transmission range between 10 and 40 meters. Each data point is the average of 10 runs. We initially present simulations in which nodes are assumed to be able to measure the exact distance from a sender, and later explore how errors in these measurements affect the accuracy of the learned positions. The former indicates how well our scheme, RLISE, can do in optimal situations, while the latter serves as a guideline for how it behaves in actual environments, and how advances in distance measurements technology will affect RLISE’s accuracy.
5.1 Performance with Exact Distance Measurements

Figure 8 explores the percentage of nodes that end up in each of the protocol states, after it reaches steady state, as a function of the number of nodes in the system. Since the area is fixed, this in fact indicates the relation between the density of the network and the accuracy of the system. In this case we fixed the number of stationary emitters to 10 in all runs, and the transmission range to 20 meters. As can be expected, the accuracy obtained by RLISE is proportional to the density of the network. Yet, this accuracy grows very fast until it reaches the point where with only 100 nodes, more than 90% of the nodes end up in the Accurate state. From there on, the improvement in accuracy grows much slower, until with 160 nodes we have 99% of the nodes in Accurate state.

Figure 9 presents how many protocol iterations (or local message exchanges) are needed until the system reaches a steady state, in which all nodes are in the best accuracy state as a function of the network density. As can be seen, initially, the number of iterations for convergence increases dramatically with the number of nodes. However, this is somewhat misleading. Recall from Figure 8 that with fewer than 80 nodes, the number of nodes that reach the Accurate state is small. Moreover, the number of Accurate nodes grows dramatically until we reach 80 nodes (after which is levels). However, once we bypass 80 nodes, in which most nodes end up Accurate, the number of iterations for convergence drops in a concave manner and eventually goes down to fewer than 6 iterations.

Figure 10 presents how many protocol iterations are needed until the system reaches a steady state as a function of the transmission range. In this graph we fixed the number of nodes to 100. As can be seen, here we have a similar phenomenon. When the transmission range is below 15 meters, a relatively small number of nodes reach the Accurate state, and thus the converges speed in that part of the graph grows very quickly as a function of the range. Beyond 15 meters, we have a significant concaved drop that levels at just above 2 rounds (on the average). Notice that when nodes and SEs are placed using uniform distribution, the probability that a node is in transmission range of at least 3 SEs is roughly equivalent to $\pi R^2 \times \frac{\text{number-of-stationary-emitters}}{3 \times \text{size-of-area}}$, where $R$ is the transmission range. Thus, with a transmission range of 35 meters and above, each node it likely to be within the transmission range of at least 3 stationary emitters.

Figure 11 illustrates the percentage of nodes that reach the Accurate state during each protocol iteration. Here again we have 100 nodes, 10 stationary emitters, and a transmission range of 20 meters. Also, in Figure 11 we compare three variants of RLISE to pure Euclidean positioning [13]. The RLISE variants are the most basic scheme, denoted RLISE, a variant that allows storing up to 2 potential circles and 4 points, denoted RLISE2, and a variant that allows storing up to 4 possible circles and 8 points, denoted RLISE4. As can be seen, RLISE converges much faster than the Euclidean method. This is because in the Euclidean method, nodes cannot make any use of partial location information and must wait until they have at least three neighbors that already established their location. Also, in this comparison, 98.5% of the nodes in RLISE reached the Accurate state, whereas in the Euclidean method only 93.5% eventually found their location. Thus, not only RLISE is faster, but indeed RLISE also enables more nodes to find their location.

Additionally, as can be seen in Figure 11, when considering all variants of RLISE, the second iteration is the most effective. After that, the number of additional nodes that become Accurate drops exponentially with each iteration. This indicates that iterations 2-4 are the most effective and most nodes converge during these iterations. Moreover, RLISE4 converges faster than RLISE2, which converges faster than RLISE. However, as can be seen, the main difference between RLISE2 and RLISE4 is in the final iterations. This indicates that there is not much point in maintaining more than 4 circles, and even 2 is probably good enough. As for

---

3 Each protocol round includes multiple iterations defined by the number of message exchanges experienced by each node within this round.
nodes that reach the Accurate state in the first iteration, these are nodes that are placed initially within the transmission range of at least three stationary emitters.

For the Euclidean method, in the first three iterations, a growing number of nodes find their location. After that, a slowly decreasing number continues to find their location. This highlights the fact that in Euclidean positioning, a node does not learn anything about its location until at least three of its neighbors know their accurate location.

Figure 12 explores the convergence rates of RLISE and the standard Euclidean method with different numbers of stationary emitter (with 100 nodes and a transmission range of 20 meters). As can be seen, with RLISE many more nodes reach the Accurate state than with the Euclidean.

### 5.2 The Impact of Distance Measurements Errors

In order to study the impact of distance measurements errors on the accumulated accuracy error, we run simulations in which distance measurements errors varied in a random manner, with a gaussian probability, up to a given maximum [3]. The measurements we present here are on the basic algorithm, without the improvements detailed in Section 4.5. Figure 13 indicates the impact of the maximal error in each single distance measurements on the accumulated error in the calculated position of each node vs. its true location. In this graph we present the percentage of nodes whose accumulated error is below a given distance from their true location. Moreover, we present four lines, representing maximal single distance measurement error of 0.1 meters, 0.3 meters, and 0.7 meters. The number of nodes was fixed to 100, the number of stationary emitters set to 10, and the transmission range at 20 meters. As can be seen, in all cases the location calculated by the vast number of nodes is less than 10 meters.

Figure 14 exhibits the maximum accumulated error obtained in each protocol iteration. As can be expected, the maximum accumulated error grows polynomially with the iteration number. This is highlights the propagation of errors in subsequent calculations.
6 Conclusions

In this paper we have demonstrated a local distributed scheme for geographic localization, which uses relative measurements and cooperative knowledge. The algorithm assumes the existence of a small number of nodes that know their exact location a-priori, e.g., through GPS or cellular triangulation. It also assumes that each node can determine its distance from a sender of a message, e.g., using known range measurements techniques such as ToA and RSSI.

With these two reasonable assumptions, our protocol allows nodes to gradually learn their location without relying on any central authority and while only exchanging local information with their immediate neighbors. Unlike standard Euclidean methods, our protocol allows nodes to maintain partially accurate states, in which they only know about a potential circle they might be on, or a couple of potential exact positions. This, in turn, allows our protocol to converge faster, and moreover, to converge in situation in which pure Euclidean methods do not.

We have presented extensive simulations that investigate the performance of our scheme, and explored its behavior as well as its limitations, including the impact of mobility and of errors in distance measurements on the accuracy of the location estimations. These simulations have shown that indeed when the network connectivity is reasonable, or better, the vast majority of nodes learn their accurate location very quickly. Also, despite reasonable potential errors in distance measurements, the vast majority of nodes compute a location that is within 10 meters of their true location (and in fact, even within 1 meter of their correct position).

Our work can still be extended in several ways. An interesting open topic is to exploit the past in order to improve the convergence and accuracy of our scheme. For example, in order to reduce the impact of errors in distance estimations, it is possible to try computing the location based on multiple messages, and then take the average. Similarly, it may be possible to use multiple such messages to rule out gross errors, etc. Here again, the trade-off is how to keep memory consumption, message sizes, and computation reasonably small.

It would be interesting to extend our scheme to three dimensions, to explore how the initial stationary emitters placement affect the accuracy of our scheme, and try to find an optimal such placement. Finally, it
might be possible to use a hybrid AOA and TOA [12] scheme for improved accuracy. Additional optimizations include incorporating TTL based techniques for improving the accuracy of the learned positions along the lines of [3]. Also, utilizing mobility, and in particular the fact that movement is continuous, to eliminate impossible locations, thereby converging faster to accurate locations [19].

Acknowledgments We would like to thank Gabi Kliot and Roman Tcharny for helpful discussions and comments.
References


