A Distributed Swarm-Intelligent Localization for Sensor Networks with Mobile Nodes

Pontus Ekberg and Edith C.-H. Ngai
Department of Information Technology
Uppsala University, Sweden
{pontus.ekberg, edith.ngai}@it.uu.se

***Abstract*—We present a novel distributed localization algorithm, called Swarm-Intelligent Localization (SIL), for computing the physical locations of nodes in wireless sensor networks. The algorithm assumes that only a small fraction of the nodes have a priori knowledge of their positions, and that noisy distance measurements are available between all neighboring nodes. The algorithm has no explicit global state and it can handle nodes that are both mobile and that can arrive in the network at any time. SIL works in two different phases, a coarse phase where nodes compute rough positions for themselves based on information about remote anchors, and a fine phase where nodes iteratively refine their positions from the coarse phase by collaborating with their neighbors. The average computational complexity per node running SIL is very low, namely constant in the network size and linear in the connectivity of the network. We evaluate the algorithm through extensive simulations. The results indicate that SIL is able to compute accurate positions for the majority of nodes in a wide range of network topologies and settings, and that it can handle difficulties such as large distance measurement errors and low network connectivity.**

***Index Terms*—Sensor networks, Mobile environments.**

I. Introduction

Many wireless sensor network applications require information about *where* a certain phenomenon has been sensed. For an example, in a tracking application we want to know where the nodes that sense the tracked entity are positioned; in a fire-detection system we want to know where a fire has been detected; and in an inventory system we might want to know where certain goods are placed. Location information is also useful when employing techniques such as location-based routing [1]. The process of determining the physical positions of nodes is known as *localization*.

Usually a few nodes are assumed to have a priori knowledge of their own positions. These *anchor nodes* often either have GPS receivers or were mapped by hand. The rest of the nodes are referred to as *non-anchor nodes*. The problem of localization is then to find the positions of the large number of non-anchor nodes, given the positions of the anchors.

In this work we present a new distributed algorithm, called *Swarm-Intelligent Localization* (SIL), for solving the localization problem. The algorithm's computational complexity per node is constant to the size of the network, and is on average linear to the connectivity (average number of one-hop neighbors) of the network. In the description of the algorithm in subsequent sections, we assume that the only information available to nodes are noisy distance measurements to their one-hop neighbors, and that each anchor has a priori knowledge of its own position.

SIL is capable of handling nodes arriving to the network at different times, as well as mobile nodes. It uses a light-weight general optimizer, Particle Swarm Optimization (PSO) [2], to compute positions. SIL is an iterative algorithm, where the computed position of a node is refined in each iteration based on fresh information acquired from other nodes. We evaluate the performance of SIL through simulations of different network topologies with varying parameters.

II. Related Work

Different approaches to localization algorithms for wireless sensor networks have been proposed in recent years. They could be centralized [3] or distributed [4]–[7], range-based [8], [9] or range-free [10], and provide absolute [8] or relative positions [11] etc. The algorithm presented in this paper is distributed, range-based and absolute, so we will focus on this type of algorithms in the remainder of this section.

One class of range-based localization algorithms attempts to localize a few non-anchor nodes at a time [8], [12], [13]. The basic idea is that all non-anchor nodes perform position estimations if they have enough direct anchor neighbors (generally three or four). When they finish the position estimations, they will turn themselves into *pseudo-anchors* and enable other non-anchor nodes to calculate their positions. This process iterates until there are no more nodes that can calculate a position. Error propagation and accumulation can be a problem for this type of algorithms, because the calculations are based on pseudo-anchors that themselves may have erroneous position estimations. Nodes may also be left unlocalized, because they never get enough (pseudo-)anchor neighbors to calculate a position at all.

Another class of algorithms lets non-anchor nodes position themselves with the help of each other, without turning them into pseudo-anchors in some order. Savarese et al. [4] suggested that non-anchor nodes first find rough initial position estimations, and once they have those they iteratively recompute and refine their positions based on each others' guesses, and broadcast any changes back to their neighborhoods. The algorithm we propose in this work also takes this approach.

Lastly, a popular localization technique is to independently localize small subnetworks consisting of a handful of neigh-

978-1-4244-9538-2/11/$26.00

boring nodes, and then stitch these small subnetworks together into a global coordinate system. Many methods have been used for localizing the subnetworks. For an example, Shang and Ruml [11] used Multidimensional Scaling (MDS) to localize the subnetworks. Li and Kunz [14] used a similar approach, but used Curvilinear Component Analysis (CCA) instead of MDS. One of the strengths of the network-stitching approach is that small subnetworks can be localized into relative coordinate systems without the help of anchor nodes, and then stitched together and aligned to an absolute coordinate system using relatively few anchors. However, this approach may require some global decision about when to do the different steps, which can make it difficult to handle mobile or newly arriving nodes. Different from that, our work can support mobile nodes arriving at different times.

III. Swarm-Intelligent Localization

In this section we present our new localization algorithm, which we call *Swarm-Intelligent Localization* (SIL). SIL is a distributed, range-based, absolute localization algorithm that can work in either two or three spatial dimensions. It works in two different phases, referred to as the *coarse phase* and the *fine phase*. All non-anchor nodes start out in the coarse phase, in which they calculate rough position estimations through a number of iterations, and then carry on to the fine phase where those estimations are further improved through more iterations.

The nodes communicate with their one-hop neighbors during each iteration by transmitting one message that can ideally be heard by all the neighbors. Each message contains information about the sender's estimation of its own current position, its confidence in that estimation, and its stored data about its one-hop neighbors and the anchors that it knows. Each piece of data concerning the transmitting node's one-hop neighbors includes that neighbor's last reported position estimation, its reported confidence in that estimation, and the measured distance to it (this data was gathered by listening to previous messages transmitted by the neighbor). Similarly, for each anchor is included its reported position, the shortest known hop-count to it, as well as the added measured distance along that path. In each iteration a node listens to one message broadcast by each of its one-hop neighbors, broadcasts one message itself, and then updates its position estimation based on its currently available information. The node's new position estimation and the information received from other nodes might be relevant to its neighbors. Therefore we iterate the whole sequence again, thus making nodes cooperate in finding a global solution as illustrated in Fig. 1.

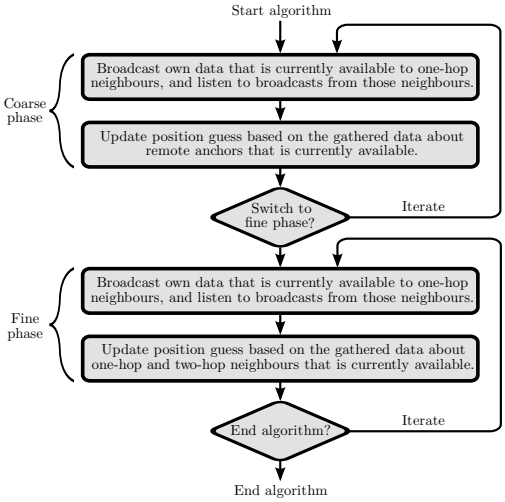


Fig. 1. In both the coarse phase and the fine phase we iterate until some end condition is fulfilled. In each iteration we exchange information between neighbors and calculate new position estimations in turn.

In order to compute a position estimation, each non-anchor node creates a Particle Swarm Optimization (PSO) instance where the solution space of the optimizer maps to the physical space (two or three spatial dimensions, represented as $\mathbb{R}^2$ or $\mathbb{R}^3$) in which we want to find the position. PSO is a population-based, stochastic optimization algorithm proposed by Kennedy and Eberhart [2]. In PSO, a number of *particles*, together called the *swarm*, move or "fly" around in the solution space of the problem, sampling the objective function at the discrete points they move to. By using information of other nodes' reported positions, as well as the measured distances to those nodes, we form a function over the solution space, called the *objective function*, that should be maximized by the PSO. The point in the solution space where we find the optimal (maximal) value of the objective function corresponds to the node's current best guess of its own position in physical space. We have slightly modified Shi and Eberhart's PSO variant [15] to handle the dynamics of a changing objective function (as new information is available each iteration, the objective function will change) while incurring little overhead. Due to the page-limit we have omitted the details about this modification.

A. The Coarse Phase

1) Overview of the Coarse Phase: In the coarse phase, the non-anchor nodes will gather position information about a number of anchors, together with the hop-count and total measured distance along a fewest-hops path to each of these anchors as shown in Fig. 2. When receiving information about an anchor, a node will store it and include it in future messages that it sends, thus making sure that this information spreads in the network.

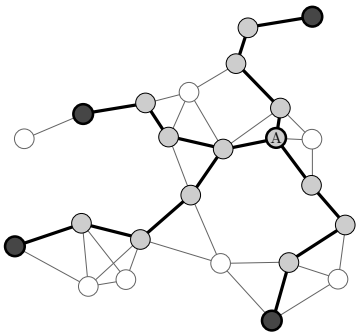


Fig. 2. In order to estimate its own position in the coarse phase, the non-anchor node labeled A uses the known positions of the anchors (darker), together with the hop-count and total measured distance along a fewest-hops path to each of them.

At the end of each iteration, a non-anchor node will position itself at the point that seems to best match its current information about the anchors. It does this by encoding the available information in its objective function, and then iterating its PSO instance on the newly formed objective function a fixed number of times. Each anchor partakes in the communication in exactly the same way as the non-anchor nodes, but does not try to update its position afterwards.

Since anchors are sparsely deployed, we can expect the majority of non-anchor nodes to have few, if any, anchors as direct neighbors. As a consequence, the distance estimations to the anchors will often have to be the sum of several hop-by-hop distance measurements. These added distances are likely to be inaccurate, especially if the nodes on the path are not collinear, and we can therefore expect the position estimations calculated in the coarse phase to be imprecise. As such, the coarse phase of SIL serves to provide rough position estimations for the non-anchor nodes, that will then be further improved in the fine phase. The fine phase is capable of computing much more precise positions than the coarse phase, but is sensitive to the initial position estimations.

Since the number of anchors in a network is potentially unbounded, non-anchor nodes never store or forward information about more than a maximum number, $anchors_{\max}$, of their closest anchors in terms of hops. We use $anchors_{\max} = 7$ in the simulations, but the optimal value will vary with factors such as network size, topology and available resources.

2) The Objective Function in the Coarse Phase: Given knowledge about an anchor's position and an estimated distance to that anchor, we can draw the conclusion that we are, approximately, somewhere on the perimeter of a circle that has the distance estimation as the radius and the centre at the anchor's reported position. In three dimensions it would be the surface of a sphere instead of the perimeter of a circle, but the reasoning is otherwise the same.

We encode this in the objective function by adding a term to it that has its maximum value on the circle's perimeter, and then gradually becomes smaller the farther away from it we get. In order to leave some room for errors in the position and distance data, we let the term flatten out as we approach the perimeter. We also let it to flatten out as we get some distance away, so that very remote anchors, or anchors to which the distance estimation is very bad, can not contribute unbounded penalties to the objective function. We achieve this by letting the term be a Gaussian function of the distance between the anchor's reported position and the evaluated point in the solution space (the non-anchor node's candidate solution), which has its center at the measured distance.

We set the variance of the Gaussian to be $2R^2$, where R is the nominal radio range, so that it can be appropriately shaped for deployments with different radio ranges. Let a be an anchor, a_{dist} be the added measured distance to that anchor, a_{pos} be its reported position and let $\|\mathbf{x}\|$ denote the Euclidean norm of $\mathbf{x}$. The contribution of anchor a to the objective function of a non-anchor node is then proportional to

$$f_a(\mathbf{x}) = e^{-\frac{(\|a_{\text{pos}} - \mathbf{x}\| - a_{\text{dist}})^2}{4R^2}}. \quad (1)$$

Since an error in the distance estimation might be introduced for each hop, especially if the nodes along the path are not collinear, we weigh the contribution of each anchor by the hop-count to it. We do not weigh by the estimated distance, since that would make us biased toward anchors to which we have underestimated the distance. Let A be the set of anchors that a non-anchor node knows about, and let a_{hops} be the hop-count from that node to the anchor a. The objective function of that node in the coarse phase is then the weighted sum of the contributions described in Eq. (1):

$$f_{\text{coarse}}(\mathbf{x}) = \sum_{a \in A} \frac{f_a(\mathbf{x})}{a_{\text{hops}}}. \quad (2)$$

B. The Fine Phase

1) Overview of the Fine Phase: In the fine phase, the non-anchor nodes will base their estimations not on remote anchors, but on their close neighbors (which may be non-anchor nodes themselves). Nodes switch to the fine phase after they have run $iter_{\max}$ iterations in the coarse phase, or they have heard from the maximum number of anchors, $anchors_{\max}$, and not moved their position estimation more than 10% of the nominal radio range in the last iteration. The number of iterations in the coarse phase is limited to at most $iter_{\max}$ to keep the total number of iterations low. Since information about anchors propagate at least one hop per iteration, any anchor that we have not heard about in many iterations will be too far away to be really useful anyway. We use $iter_{\max} = 8$, which we found to be a reasonable trade-off. Note that different non-anchor nodes may be in different phases at any specific time.

A non-anchor node will listen to the reported positions that its neighbors currently *believe* they have, and then try to position itself based on these reported positions and the measured distances to these same neighbors. Note that each node include the latest gathered information about its one-hop neighbors in the messages it sends. By listening to messages sent by its one-hop neighbors, a node can therefore gather data about its two-hop neighbors. Both one-hop and two-hop neighbors are used for the position estimation in the fine phase (though we could generalize it to use n-hop neighbors, we believe that returns in precision would be sharply diminishing on going further than two hops away).

Once a non-anchor node has found a new position estimation, this new estimation will have an effect on the estimations of its neighbors, that based *their* estimations partly on the estimation of the first node. Its neighbors will consequently update their positions again, which will in turn affect the estimation of the first node, and so on. What grows out of this is a distributed, dynamic system, where nodes collaborate, by using only local information, to achieve a global, coherent solution. This is allowed to iterate for either a fixed number of iterations, or until the nodes converge to solutions.

2) The Objective Function in the Fine Phase: For the one-hop neighbors we have distance measurements that can be used to form contributions to the objective function in a similar way to the anchors in the coarse phase. Just as with the anchors,

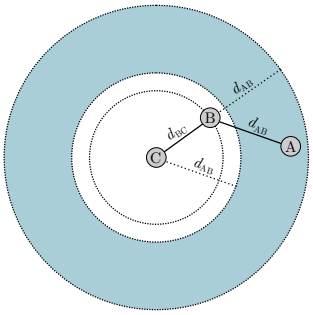


Fig. 3. A pair of two-hop neighbors, A and C, are connected through a third node, B. The distance measurements are d_{AB} and d_{BC}. Then A should likely be between $\max(d_{\mathrm{AB}}, d_{\mathrm{BC}})$ and $d_{\mathrm{AB}} + d_{\mathrm{BC}}$ distance units away from C, indicated by the filled area.

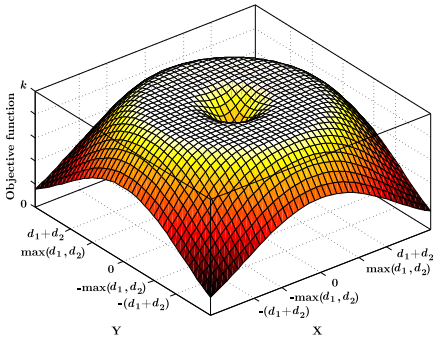


Fig. 4. In this example, the other node's reported position is at $(0, 0)$, and the two nodes are connected through a third node to which they have the measured distances d_1 and d_2, respectively. The term is maximal at any point that is between $\max(d_1, d_2)$ and $d_1 + d_2$ distance units away from the other node's reported position, and then decreases as we deviate from that area.

the contribution of a one-hop neighbor is proportional to a Gaussian of the distance to the perimeter of a circle centered on the neighbor's reported position. The contribution to the objective function of a one-hop neighbor, n, will therefore be proportional to

$$f_n(\mathbf{x}) = e^{-\frac{(\|n_{\mathrm{pos}} - \mathbf{x}\| - n_{\mathrm{dist}})^2}{4R^2}}. \quad (3)$$

For two-hop neighbors we can not directly measure the distance, and estimating it by simply adding two distance measurements together is too inexact for our purposes in the fine phase. However, we can find a larger space in which a node is likely to be in, given these two distance measurements. We note that a node is unlikely to be further away from one of its two-hop neighbors than the added distance measurements to it. Also, since two-hop neighbors are *not* one-hop neighbors, they are unlikely to be closer to each other than any of them are to the node that connects them (if they were, they would probably have been able to communicate directly with each other). Following this reasoning, we can conclude that it is likely that a node is positioned somewhere in a ring-shaped area (or space for the three-dimensional case), as is shown in Fig. 3.

By using information about two-hop neighbors we can successfully localize some nodes that can not be unambiguously localized using only one-hop neighbors. Even if there are several one-hop neighbors connecting a node to the same two-hop neighbor, it only considers the two-hop neighbor once, using the distances it gets from the one-hop neighbor that is farthest from it.

We let a two-hop neighbor add to the objective function a term that is proportional to a Gaussian of the distance from the area described in Fig. 3. An example term is shown in Fig. 4. Let m_{pos} be the reported position of the two-hop neighbor m, and let d_1 and d_2 be the measured distances of the two hops to it, respectively. The contribution to the objective function by the two-hop neighbor, m, is then proportional to

$$f_m(\mathbf{x}) = e^{-\frac{h(\|m_{\mathrm{pos}} - \mathbf{x}\|)^2}{4R^2}}, \quad (4)$$

where $h(y)$ is the distance to the optimal area, when y is the distance to m:

$$h(y) = \max(0, \max(d_1, d_2) - y, y - (d_1 + d_2)). \quad (5)$$

We weigh contributions to the objective function based on the *confidence factors* of the neighbor nodes. The reasoning is that if a node knows which neighbors' guesses are likely to be more accurate than others', it can weigh its own guess based on this information. We let all anchors have a fixed confidence factor of 1, and let higher values denote lower confidence. A non-anchor node's confidence factor is calculated once as it switches to the fine phase. The confidence is based on the closeness in hops to the minimum number of anchors that are required for a unique solution to be possible (three in two dimensions, four in three dimensions). If the node does not know about enough anchors, we use a somewhat arbitrary large number, 100 in our case, as the hop-counts for the "missing" ones. Let d be the number of dimensions of the solution space, and A_{closest} be the closest $d + 1$ anchors in in terms of hop-counts that a node n knows about (or as many as are available). The confidence factor of n is then given by

$$n_{\mathrm{confidence}} = \sum_{a \in A_{\mathrm{closest}}} (a^2_{\mathrm{hops}}) + (d + 1 - |A_{\mathrm{closest}}|) \cdot 100^2. \quad (6)$$

We chose this metric for the confidences because of its simplicity and because it closely relates to *why* some nodes have better position estimations early on. We can not simply compute a node's confidence factor based on how well it has been able to find a solution to its objective function, though it might seem like a good metric. This is because clusters of nodes can position themselves perfectly with respect to each other, but still be far away from their true positions—something which seems to happen almost exclusively to nodes that do not have enough anchors nearby.

The objective function in the fine phase is then a weighted sum of the contributions from the one-hop and two-hop neighbors in (3) and (4), respectively. Let N be the set of all one-hop neighbors of a node, and M the set of all two-hop neighbors. The objective function of that node in the fine phase is then

$$f_{\mathrm{fine}}(\mathbf{x}) = \sum_{n \in N} \frac{f_n(\mathbf{x})}{n_{\mathrm{confidence}}} + \sum_{m \in M} \frac{f_m(\mathbf{x})}{m_{\mathrm{confidence}}}. \quad (7)$$

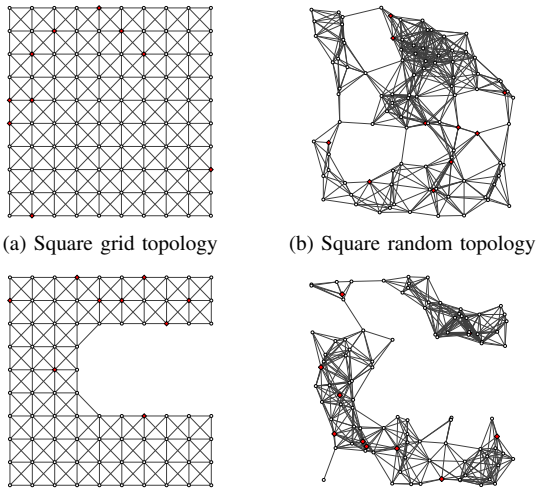
IV. Evaluations

A. Simulation Setup

We simulate and evaluate the performance of the SIL algorithm by varying parameters such as ranging errors, network connectivity and node mobility. Four different topology types are considered, as shown in Fig. 5. We define the *location error* of a node as the distance between its true position and estimated position. In the figures we plot the *mean location error* and the *median location error* of all the non-anchor nodes. All distances are normalized with the nominal radio range R (i.e., if $R = 30$ m, then a location error of 1 means an error of 30 m). Distance measurement noise is modeled as Gaussian. Let d be the true distance between two nodes, the measured distance $\tilde{d}$ is then

$$\tilde{d} = d(1 + \omega Z), \tag{8}$$

where Z is a random variable drawn from a standard normal distribution, and ω is a parameter for controlling the noise level.



(a) Square grid topology (b) Square random topology

(c) C-shaped grid topology (d) C-shaped random topology

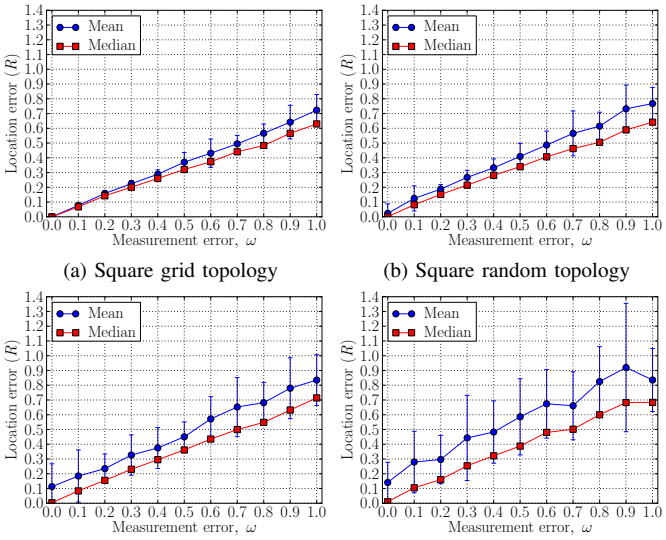
Fig. 5. The different types of network topologies.

B. Effects of Ranging Errors

Fig. 6 shows the location errors for varying values of the ranging error parameter ω. All simulated networks had a size of 100 nodes, of which 10% were chosen at random to be anchors. The radio range and network area were set so that the connectivity was about 10 for each network.

As we might have expected, the grid topologies had lower location errors than their random counterparts. The explanation is that the grid topologies' regular structure avoid "weak" spots that are inherently hard to localize. However, a larger difference is to be seen between the square and the C-shaped topologies. All of the topology types had similar median location errors, but the C-shaped ones had larger mean location errors with a larger standard deviation.

Location errors grow rather slowly when the ranging errors increase, which indicates that the algorithm is robust even in the face of large distance measurement errors. Compared to other proposed localization algorithms, SIL's robustness



(a) Square grid topology (b) Square random topology

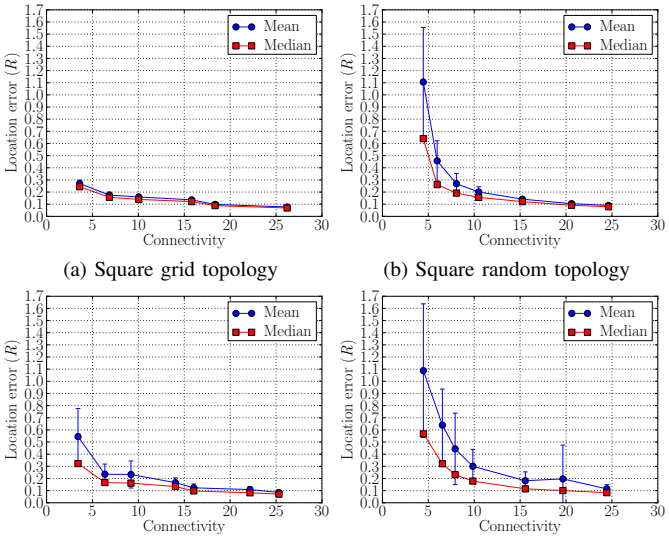
(c) C-shaped grid topology (d) C-shaped random topology

Fig. 6. The effects of ranging errors.

to large ranging errors seems comparable or favorable. To the best of our knowledge there are no algorithms that are significantly more robust in this sense. We should note that the ranging errors when $\omega = 1$ are very large. When ω is larger than around 0.5 we would get comparable or even better results by simply setting all distances to the same approximate average distance, effectively making the algorithm *range-free*.

C. Effects of Network Connectivity

Next, we studied the effects of varying network connectivity. The simulated networks had a size of 100 nodes, 10% of them randomly picked to be anchors. The ranging error parameter ω was set to 0.2. We then varied the connectivity by changing the radio range of the nodes. The results are shown in Fig. 7.



(a) Square grid topology (b) Square random topology

(c) C-shaped grid topology (d) C-shaped random topology

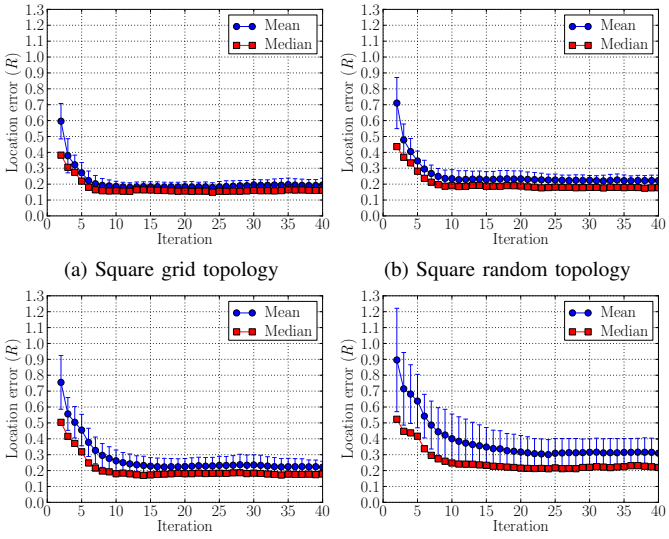
Fig. 7. The effects of connectivity.

Clearly, higher connectivity leads to better results. This can

be explained by the fact that with higher connectivity, the average node has more neighbors, and thus more constraints on its position estimation in the fine phase. The square grid topology shows a remarkable resilience to low connectivities, which can be explained by its regular structure's lack of weak, ill-connected spots. Even with very low connectivities, no nodes in the grid network are really ill-connected. The other network types were more difficult. In the random topologies, low connectivity leads to more nodes that are ill-connected, which makes them hard to localize.

D. Mobility

To evaluate the effects of mobility we studied the long term behavior of the location errors as nodes were moving, and plotted these in Fig. 8. All networks were of size 100, 10% of the nodes were picked to be anchors, the connectivity was about 10 and $\omega = 0.2$. We let 50% of the non-anchor nodes be mobile and gave each of them an individual velocity $v = (\Delta x, \Delta y)$, such that Δx and Δy were drawn from a uniform distribution on the interval $[-R/5, R/5]$, where R is the nominal radio range. In other words, when nodes moved as fast as possible, they traveled the full radio range in each direction during the time it took to run 5 SIL iterations. When a mobile node got to the edge of the network area, it "bounced" back by negating its velocity in that direction.



(a) Square grid topology (b) Square random topology

(c) C-shaped grid topology (d) C-shaped random topology

Fig. 8. The effects of mobility.

After a few iterations the location errors reached a relatively steady level, and the algorithm was capable of keeping them at that level while the nodes were moving. Many sensor network deployments might need to keep track of some mobile nodes, but most localization algorithms do not handle mobility in any other way than rerunning the entire algorithm from start at periodic intervals. For SIL, this is not necessary.

V. Conclusions

We have presented a new distributed, simple and lightweight algorithm, named SIL, for localization in wireless sensor networks. The algorithm is flexible in that users easily can modify it by adding or removing terms in the objective function as is suitable for particular applications. For example, if we know that the deployment has a particular shape or structure, terms can be added to the objective function that rewards estimations that agree with this knowledge.

SIL also support mobile nodes arriving at any time. Mobility is handled by iterating the algorithm either when the nodes move (controlled mobility) or at periodic intervals (uncontrolled mobility). In each iteration, we build on prior knowledge, thus enabling efficient refinements and updates. Through extensive simulations we have demonstrated that SIL is capable of computing precise positions for the majority of nodes under different network topologies and settings. All nodes, except completely isolated ones, are localized on a best-effort basis. The location errors of SIL have been shown to be low even for networks with large ranging errors and low connectivity. The algorithm is also computational efficient and very scalable.

Acknowledgement

This work was supported by the VINNOVA VINNMER program funded by the Swedish Governmental Agency for Innovation Systems. We would also like to thank Dr. David Black-Schaffer for his valuable comments on the paper.

References

[1] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Elsevier Ad Hoc Network Journal*, vol. 3, 2005, pp. 325–349.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[3] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks," in *IEEE INFOCOM*, vol. 3, 2001, pp. 1655–1663.

[4] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *USENIX ATEC*, 2002, pp. 317–327.

[5] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler, "The effects of ranging noise on multihop localization: An empirical study," in *IPSN 2005*, April 2005, pp. 73–80.

[6] V. Savic, A. Poblacion, S. Zazo, and M. Garcia, "An experimental study of RSS-based indoor localization using nonparametric belief propagation based on spanning trees," in *SENSORCOMM*, July 2010, pp. 238-243.

[7] B. H. Cheng, L. Vandenberghe, and K. Yao, "Distributed algorithm for node localization in wireless ad-hoc networks," *ACM Trans. Sensor Networks*, vol.6, issue 1, Article 8, December 2009, pp. 1-20.

[8] A. Savvides, C.-C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *ACM MobiCom*, 2001, pp. 166–179.

[9] M. L. Sichitiu and V. Ramadurai, "Localization of wireless sensor networks with a mobile beacon," in *IEEE MASS*, 2004, pp. 174–183.

[10] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, 2000, pp. 28–34.

[11] Y. Shang and W. Ruml, "Improved MDS-based localization," in *IEEE INFOCOM*, March 2004, pp. 2640–2651.

[12] J. Liu, Y. Zhang, and F. Zhao, "Robust distributed node localization with error management," in *ACM MobiHoc*, 2006, pp. 250–261.

[13] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *ACM WSNA*, 2002, pp. 112–121.

[14] L. Li and T. Kunz, "Localization applying an efficient neural network mapping," in *ICST Autonomics*. Brussels, Belgium, 2007, pp. 1–9.

[15] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of Evolutionary Computation*, 1998, pp. 69–73.