International Journal of Computer Sciences and Engineering Systems
Vol. 11 No. 2 (December, 2017)

Distributed Positioning Technique in Wireless Ad Hoc Sensor Networks

M. K. Jeya Kumar[1] and R. S. Rajesh[2]

[1]Department of Computer Applications, Noorul Islam College of Engineering.,Kumaracoil, Thuckalay, India
E-mail: jeyakumarmk@yahoo.com
[2]Department of Computer Science & Engineering, Manonmaniam Sundaranar University, Tirunelveli, India
E-mail: rsrajesh1_tvl@yahoo.co.in

Abstract: *To support processing and collaboration, every node in the network needs to know the identity and location of its neighbors. This can be easily obtained in a planned network and quite hard in ad hoe networks whose topology is constructed and updated in real time. Self-location using* ***GPS*** *may not be feasible but an effective distributed positioning algorithm is to be used. The proposed technique is a distributed infrastructure free positioning algorithm which achieves robustness through iterative propagation of information through a network. The two primary obstacles to positioning in an ad-hoc network like sparse anchor node problem and the range error problem are considered. In order to meet the above problems the new distributed algorithm is generated. This algorithm consists of two phases: start-up and refinement. For the start-up phase the Hop-TERRAIN algorithm is used to overcome the sparse anchor node problem, and refinement algorithm is used to refine the position estimates generated by Hop-TERRAIN. The experiment is carried out in a network containing 400 nodes randomly placed in a square area and the resulting error magnitudes are represented as percentages of each node's radio range.*

Keywords: *Sensor Nodes, Node positioning, Hop-TERRAIN, Refinement, Anchor Nodes, Range error*

1. INTRODUCTION

Large numbers of sensor nodes are densely deployed over physical space in sensor networks. Many applications require knowing the positions of the nodes, sometimes relative positions among them. In this paper, a new distributed technique for discovering relative locations of nodes based on local distance information is proposed. The proposed technique is a healthy algorithms that are capable of handling the wide set of possible scenarios left open by so many degrees of freedom. Specifically assume that all the nodes being considered in an instance of the positioning problem are within the same connected network and that there will exist within this network a minimum of four anchor nodes. In this a connected network is a network in which there is a path between every pair of nodes, and an anchor node is a node that is given a priori knowledge of its position with respect to some global coordinate system.

The positioning algorithm[3] being considered relies on measurements, with limited accuracy, of the distances between pairs of neighboring nodes is called range measurements. Several techniques can be used to generate these range measurements, including time of arrival, angle of arrival, phase measurements, and received signal strength. This algorithm is indifferent to which method is used, except that different methods offer different tradeoffs between accuracy, complexity, cost, and power requirements. Some of these methods generate range measurements with errors as large as ± 50% of the measurement. These errors can come from multiple sources, including multipath interference [2], line-of-sight obstruction, and channel in homogeneity [1] with regard to direction. This work, however, is not concerned with the problem of determining accurate range measurements. Instead, assume large errors in range measurements that should represent an agglomeration of multiple sources of error. Being able to cope with range measurements errors is the first of two major challenges in positioning within an ad-hoc space, and will be termed the *range error problem* throughout this paper.

The second major problem in ad-hoc positioning algorithms is sparse anchor node problem [1], comes from the need for at least four reference points with known location in a three-dimensional space [13] in order to uniquely determine the location of an unknown object [6]. Too few reference points results in ambiguities that lead to underdetermined systems of equations. Recalling the assumptions made above, only the anchor nodes will have positioning information at the start of these algorithms, and assume that these anchor nodes will be located randomly throughout an arbitrarily large network. Given limited radio ranges, it is therefore highly unlikely that any randomly selected node in the network will be in direct communication

with a sufficient number of reference points to derive its own position estimate [7].

In this paper, the distributed algorithm is split into two phases: the *start-up* phase and the *refinement* phase. The start-up phase addresses the sparse anchor node problem by cooperatively spreading awareness of the anchor nodes' positions throughout the network, allowing all nodes to arrive at initial position estimates. These initial estimates are not expected to be very accurate, but are useful as rough approximations. The refinement phase of the algorithm then uses the results of the start-up algorithm to improve upon these initial position estimates. The ranger error problem is also addressed in this paper.

The organization of the paper is as follows. Section 2 will elaborate the two-phase algorithm approach, exploring in depth the start-up and refinement phases of the solution and section 3 will discus about the implementation phase of this paper. Section 3 will discuss about the experiment results of the algorithm and section 5 will discuses about the discussion of the algorithm. The conclusion and future work of the paper is included in the section 6.

2. TWO-PHASE POSITIONING ALGORITHM

The two important problems to positioning in an ad-hoc network are the sparse anchor node problem and the range error problem[10]. In order to address each of these problems sufficiently, the proposed algorithm is separated into two phases: start-up and refinement. For the start-up phase the Hop-TERRAIN algorithm[11] is used. The Hop-TERRAIN algorithm is run once at the beginning of the positioning algorithm to overcome the sparse anchor node problem, and the Refinement algorithm is run iteratively afterwards to improve upon and refine the position estimates generated by Hop-TERRAIN. Refinement is concerned only with nodes that exist within a one-hop neighborhood, and it focuses on increasing the accuracy of the position estimates as much as possible.

2.1 Hop-TERRAIN

In sensor networks most of the nodes have no positioning data, with the exception of the anchors. The networks being considered for this algorithm will be scalable to very large numbers of nodes spread over large areas, relative to the short radio ranges that each of the nodes is expected to possess. Furthermore, it is expected that the percentage of nodes that are anchor nodes will be small. This results in a situation in which only a very small percentage of the nodes in the network are able to establish direct contact with any of the anchors, and probably none of the nodes in the network will be able to directly contact enough anchors to derive a position estimate.

In order to overcome this initial information deficiency, the Hop-TERRAIN algorithm finds the number of hops from a node to each of the anchors nodes in a network and then multiplies this hop count by an average hop distance to estimate the range between the node and each anchor. These computed ranges are then used together with the anchor nodes' known positions to perform a triangulation and get the node's estimated position. The triangulation consists of solving a system of linearized equations (Ax=b) by means of a least squares algorithm[5].

Each of the anchor nodes launches the Hop-TERRAIN algorithm by initiating a broadcast containing its known location and a hop count of 0. All of the one-hop neighbors surrounding an anchor hear this broadcast, record the anchor's position and a hop count of 1, and then perform another broadcast containing the anchor's position and a hop count of 1. Every node that hears this broadcast and did not hear the previous broadcasts will record the anchor's position and a hop count of 2 and then rebroadcast. This process continues until each anchor's position and an associated hop count value have been spread to every node in the network. It is important that nodes receiving these broadcasts search for the smallest number of hops to each anchor. This ensures conformity with the model used to estimate the average distance of a hop, and it also greatly reduces network traffic.

As broadcasts may be omni-directional, and may therefore reach nodes behind the broadcasting node. this algorithm causes nodes to hear many more packets than necessary. In order to prevent an infinite loop of broadcasts, nodes are allowed to broadcast information only if it is not stale to them. In this context, information is stale if it refers to an anchor that the node has already heard from and if the hop count included in the arriving packet is greater than or equal to the hop count stored in memory for this particular anchor. New information will always trigger a broadcast, whereas stale information will never trigger a broadcast.

Once a node has received an average hop distance and data regarding at least 3 or 4 anchor nodes for a network existing in a 2 or 3-dimensional space, it is able to perform a triangulation to estimate its location[6]. If this node subsequently receives new data after already having performed a triangulation, either a smaller hop count or a new anchor, the node simply performs another triangulation to include the new data. This procedure is summarized in the following piece of pseudo code:

when a positioning packet is received,
if new anchor **or** lower hop count
then store hop count for this anchor.
broadcast new packet for this anchor with hop count = (hop count + 1).
else do nothing.
if average hop count is known **and** number of anchors >=(dimension of space + 1)
then triangulate.
else do nothing.

The resulting position estimate is likely to be coarse in terms of accuracy, but it provides an initial condition from which Refinement can launch.

2.2 Refinement

Given the initial position estimates of Hop-TERRAIN in the start-up phase, the objective of the refinement phase[12] is to obtain more accurate positions using the estimated ranges between nodes. Since Refinement must operate in an ad-hoc network, only the distances to the direct (one-hop) neighbors of a node are considered.

Refinement is an iterative algorithm in which the nodes update their positions in a 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 computes a least squares triangulation solution 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 factors that influence the convergence and accuracy of iterative Refinement are the accuracy of the initial position estimates, the magnitude of errors in the range estimates, the average number of neighbors, and the fraction of anchor nodes. Assume that redundancy can counter the above influences to a large extent. When a node has more than 3(4) neighbors in a 2(3)-dimensional space the induced system of linear equations is over-defined and errors will be averaged out by the least squares solver[7].

Despite the positive effects from redundancy it is observed that a straightforward application of Refinement did not converge in a considerable number of "reasonable" cases. Close inspection of the sequence of steps taken under Refinement revealed two important causes:

1. Errors propagate fast throughout the whole network. If the network has a diameter *d*, then an error introduced by a node in step *s* has (indirectly) affected every node in the network by step $s + d$ because of the triangulate-hop pattern.
2. Some network topologies are inherently hard, or even impossible, to locate. For example, a cluster of n nodes (no anchors) connected by a single link to the main network can be simply rotated around the 'entry'-point into the network while keeping the exact same intra- node ranges. Another example is given in Figure 1.

To mitigate error propagation the refinement algorithm is modified to include a confidence associated with each node's position. The confidences are used to weigh the equations when solving the system of linear equations. Instead of solving $Ax = b$, now solve $wAx = b$, where wb is the vector of confidence weights. Nodes, like anchors, that have high faith in their position estimates select high confidence values (close to 1). A node that observes poor conditions (e.g., few neighbors, poor constellation) associates a low confidence (close to 0) with its position estimate, and consequently has less impact on the outcome of the triangulations performed by its neighbors. The usage of confidence weights improved the behavior of Refinement greatly: almost all cases converge now, and the accuracy of the positions is also improved considerably.

Another improvement to Refinement was necessary to handle the second issue of ill-connected groups of nodes. Detecting that a single node is ill-connected is easy: if the number of neighbors is less than 3(4) then the node is ill-connected in a 2(3)-dimensional space. Detecting that a group of nodes is ill connected, however, is more complicated since some global overview is necessary. So employ a heuristic that operates in an ad-hoc fashion (no centralized computation), yet is able to detect most ill-connected nodes. The underlying premise for the heuristic is that a sound node has independent references to at least 3(4) anchors. That is, the multi-hop routes[4] to the anchors have no link (edge) in common. For example, node 3 in Figure 1 meets these criteria and is considered sound.

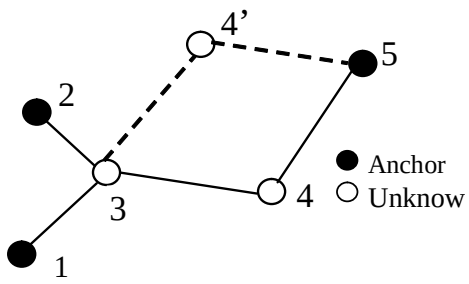

Figure 1: Example Topology

To determine if a node is sound, the Hop-TERRAIN algorithm records the ID of each node's immediate neighbor along a shortest path to each anchor. When multiple shortest paths are available, the first one discovered is used. These IDs are collected in a set of sound neighbors. When the number of unique IDs in this set reaches 3(4), a node declares itself sound and may enter the Refinement phase. The neighbors of the sound node add its ID to their sets and may in turn become sound if their sound sets become sufficient. This process continues throughout the network. The end result is that most ill-connected nodes will not be able to fill their sets of sound neighbors with enough entries and, therefore, may not participate in the Refinement phase. In the example topology in Figure 1, node 3 will become sound, but node 4 will not.

3. IMPLEMENTATION

To analyze study the robustness of the two-phase positioning algorithm it was implemented in NS2 simulation environment and average position error after Hop-TERRAIN and Refinement process are calculated.

The positioning algorithm is designed to be used in an ad-hoc network that presumably employs multi-hop routing algorithms [4][8] and it only requires that a node be able to broadcast a message to all of its one hop neighbors. An important result of this is the ability for system designers to

allow the routing protocols to rely on position information, rather than the positioning algorithm relying on routing capabilities. An important issue is whether or not the network provides reliable communication in the presence of concurrent transmission. In this paper it was assumed that message loss or corruption does not occur and that each message is delivered at the neighbors within a fixed radio range (R) from the sending node.

The functionality of the network layer (local broadcast) is implemented and it is connected to all sensor-node objects in the simulation. This network object holds the topology of the simulated sensor network, which can be read from a "scenario" file or generated at random at initialization time. At time zero the network object sends a pseudo message to each sensor-node object telling its role (anchor or unknown) and some attributes (e.g., the position in the case of an anchor node). From then on it relays messages generated by sensor nodes to the sender's neighbors within a radius of *R* units.

3.2 Hop-TERRAIN

In this algorithm, all of the nodes in the network are waiting to receive hop count packets informing them of the positions and hop distances associated with each of the anchor nodes at time zero. Also at time zero, each of the anchor nodes in the network broadcasts a hop count packet, which is received and repeated by all of the anchors' one-hop neighbors. This information is propagated throughout the network until, ideally, all the nodes in the network have positions and hop counts for all of the anchors in the network as well as an average hop distance. At this point, each of the nodes performs a triangulation to create an initial estimate of its position. The number of anchors in any particular scenario is not known by the nodes in the network, however, so it is difficult to define stopping criteria to dictate when a node should stop waiting for more information before performing a triangulation. To solve this problem, nodes perform triangulations every time they receive information that is not stale after having received information from the first 3(4) anchors in a 2(3)-dimensional space. Nodes also rely on the anchor nodes to inform them of the value to use for the assumed average hop distance used in calculating the estimated range to each anchor. When an anchor node receives a hop count from another anchor it computes its estimate of the average hop distance, and floods that back into the network. Nodes wait for the first such estimate to arrive before performing any triangulation as outlined above. Subsequent estimates from other anchor pairs are simply discarded to reduce network load.

The above details are sufficient for controlling the Hop-TERRAIN algorithm within a simulated environment where all of the nodes start up at the same time. One important consequence of a real network, however, is that the nodes in the network start up or enter the network at random times, relative to each other. This allows for the possibility that a late node might miss some of the waves of propagated broadcast messages originating at the anchor nodes. To solve this, each node is programmed to announce itself when it first comes online in a new network. Likewise, every node is programmed to respond to these announcements by passing the new node their own position estimates, the positions of all of the anchor nodes they know of, and the hop counts and hop distance metrics associated with these anchors. Note that, according to the rebroadcast rules regarding stale information, this information will all be new to the new node, causing this new node to then rebroadcast all of the information to all of its one-hop neighbors.

3.3 Refinement

In refinement algorithm the information in incoming messages is recorded internally, but not processed immediately. This allows for accumulating multiple position updates from different neighbors, and responding with a single reply. The task of an anchor node is to broadcasts its position whenever it has detected a new neighbor in the preceding period. The task of an unknown node is, if new information arrived in the preceding period it performs a triangulation to compute a new position estimate, determines an associated confidence level, and finally decides whether or not to send out a position update to its neighbors.

A confidence is a value between 0 and 1. Anchors immediately start off with confidence 1; unknown nodes start off at a low value (0.1) and may raise their confidence at subsequent Refinement iterations. Whenever a node performs a successful triangulation it sets its confidence to the average of its neighbors' confidences. This will, in general, raise the confidence level. Nodes close to anchors will raise their confidence at the first triangulation, raising in turn the confidence of nodes two hops away from anchors on the next iteration, etc. Triangulations sometimes fail or the new position is rejected on other grounds (see below). In these cases the confidence is set to zero, so neighbors will not be using erroneous information of the inconsistent node in the next iteration. This generally leads to new neighbor positions bringing the faulty node back into a consistent state, allowing it to build its confidence level again. In unfortunate cases a node keeps getting back into an inconsistent state, never converging to a final position/ confidence. To warrant termination it is enough to limit the number of position updates of a node to a maximum. Nodes that end up with a poor confidence (< 0.1) are discarded and excluded from the reported error results; all others are considered to be located and included in the results.

To avoid flooding the network with insignificant or erroneous position updates the triangulation results are classified as follows. First, a triangulation may simply fail because the system of equations is underdetermined (too few neighbors, bad constellation). Second, the new position may be very close to the current one, rendering the position update insignificant. Use a tight cutoff radius of $\frac{1}{100}$ of the radio

range; experimentation showed Refinement is fairly insensitive to this value as long as it is small (under 1% of the radio range). Third, check that the new position is within the reach of the anchors used by Hop-TERRAIN. Similarly to check the convex constraints[12] that the distance between the position estimate and anchor ai must be less than the length of the shortest path to ai(hop-counti) times the radio range (*R*). When the position drifts outside the convex region, then reset the position to the original initial position computed by Hop-TERRAIN. Finally, the validity of the new position is checked by computing the difference between the sum of the observed ranges and the sum of the distances between the new position and the neighbor locations. Dividing this difference by the number of neighbors yields a normalized residue. If the residue is large (residue > radio range) then assume that the system of equations is inconsistent and reject the new position.

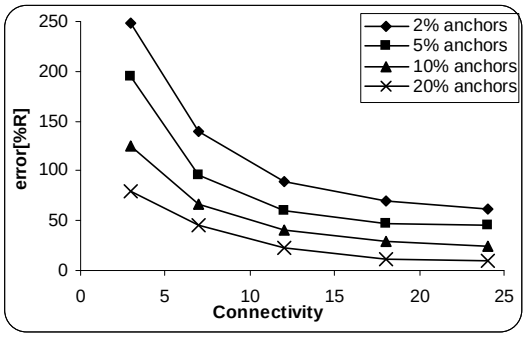

Figure 2: Average Position Error After Hop-TERRAIN (5% Range Errors)

4. EXPERIMENT RESULTS

This algorithm is implemented using NS2 simulation environment. All data points represent averages over 100 trials in networks containing 400 nodes. The nodes are randomly placed, with a uniform distribution, within a square area.. The specified fraction of anchors is randomly selected, and the 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 mean1. The connectivity (average number of neighbors) is controlled by specifying the radio range. 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 half the range of the radio).

Fig. 2 shows the average performance of the Hop-TERRAIN algorithm as a function of connectivity and anchor population in the presence of 5% range errors. As seen in this plot, position estimates by Hop-TERRAIN have an average accuracy under 100% error in scenarios with at least 5% anchor population and an average connectivity level of 7 or greater. In extreme situations where very few anchors exist and connectivity in the network is very low, Hop-TERRAIN errors reach above 250%.

Fig. 3 displays the results from the same experiment depicted in Fig. 2, but now the position estimates of Hop-TERRAIN are subsequently processed by the Refinement algorithm. Its shape is similar to that of Fig. 2, showing relatively consistent error levels of less than 33% in scenarios with at least 5% anchor population and an average connectivity level of 7 or greater. Refinement also has problems with low connectivity and anchor populations, and is shown to climb above 50% position error in these harsh conditions. Overall Refinement improves the accuracy of the position estimates by Hop-TERRAIN by a factor three to five.

Fig. 4 shows the sharp increases in positioning errors for low anchor populations and sparse networks shown in figures 2 and 3. Also this shows that the average connectivity between nodes throughout the network decreases past certain points, both algorithms break down, failing to derive position estimates for large fractions of the network. This is due simply to a lacking of sufficient information, and is a necessary consequence of loosely connected networks. Nodes can only be located when connected to at least 3(4) neighbors; Refinement also requires a minimal confidence level (0.1). It should be noted that the results in Fig. 4 imply that the reported average position errors for low connectivities in fig. 2 and 3 have low statistical significance, as these points represent only small fractions of the total network. Nevertheless, the general conclusion to be drawn from figures 2, 3, and 4 is that both Hop-TERRAIN and Refinement perform poorly in networks with average connectivity levels of less than 7.

In the above experiment place 400 nodes randomly on the vertices of a 200x200 grid, rather than allowing the nodes to sit anywhere in the square area. It is found that the grid layout did not result in better performance for the Refinement algorithm, relative to the performance of the Refinement algorithm with random node placement. It was found that a difference in performance for Hop-TERRAIN though. Fig. 5 shows that placing the nodes on a grid dramatically reduces the errors of the Hop-TERRAIN algorithm in the cases where connectivity or anchor node populations are low. For example, with 5% anchors and a connectivity of 8 nodes, the average position error decreases from 95% (random distribution) to 60% (grid).

Sensitivity to average error levels in the range measurements is a major concern for positioning algorithms. Figure 6 shows the results of an experiment in which the anchor population and connectivity constant at 10% and nodes, respectively, while varying the average level of error in the range measurements. It is found that Hop-TERRAIN was almost completely insensitive to range errors. This is a result of the binary nature of the procedure in which routing hops are counted; if nodes can see each other, they pass on incremented hop counts, but at no time do any nodes attempt to measure the actual ranges between them. Unlike Hop-TERRAIN, Refinement does rely on the range measurements

performed between nodes, and Figure 6 shows this dependence accordingly. At less than 40% error in the range measurements, on average, Refinement offers improved position estimates over Hop-TERRAIN. The results improve steadily as the range errors decrease. For each node performed a triangulation using the true positions of its neighbors and the corresponding erroneous range measurements. The resulting position errors are plotted as the lower bound in Fig. 6.

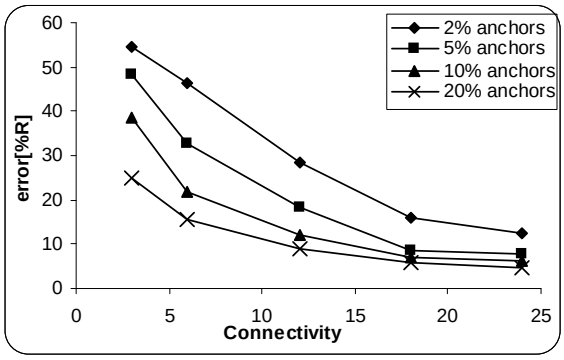

Figure 3: Average Position Error after Refinement (5% Range Errors)

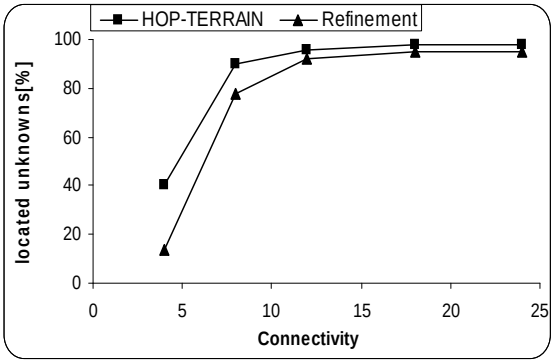

Figure 4: Fraction of Located Nodes (2% Anchors, 5% Range Errors).

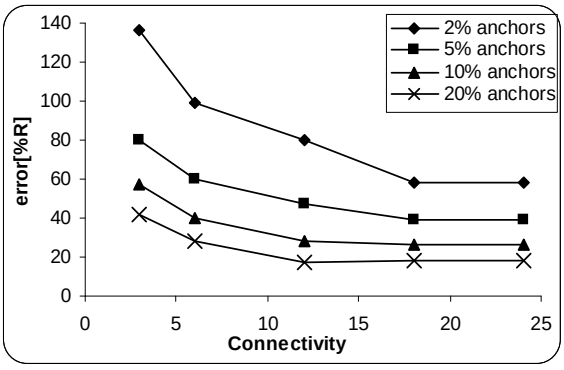

Figure 5: Average Position Error After Hop-TERRAIN (2D Grid, 5% Range Errors)

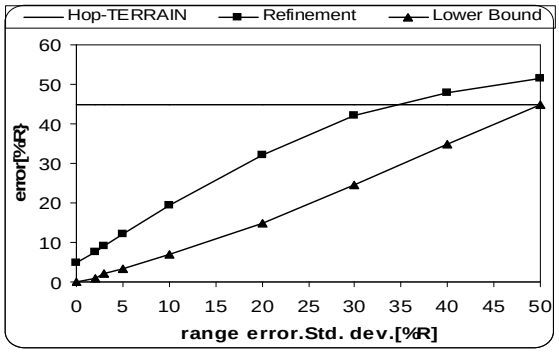

Figure 6: Range Error Sensitivity (10% Anchors, Connectivity 12)

5. DISCUSSION

In this section the performance of the results obtained in the section 4 and some other alternative methods of are compared.

The Hop-TERRAIN algorithm results are compared with DV-hop algorithm [5] The position error for Hop-TERRAIN is 69% and for DV-hop is 35% on a scenario with 10% anchor nodes and a connectivity of 8. Under poorer network conditions though, Hop-TERRAIN is more robust than DV-hop, showing about a factor of 2 improvements in position accuracy in sparsely connected networks. Regardless, the trend observed in both studies is the same: when the fraction of anchors drops below 5%, position errors rapidly increase.

The results of Refinement are comparable to an "iterative multilateration" scenario[9] with 50 nodes, 20% anchors, connectivity 10, and 1% range errors [3]. Their algorithm, however, can handle neither low anchor fractions nor low connectivity, because positioning starts from nodes connected to at least 3 anchors. Refinement still performs acceptably well with few anchors or a low connectivity. Furthermore the preliminary results of their more advanced "collaborative multilateration" algorithm [2] [11] show that Refinement is able to determine the position of a larger fraction of unknowns: 56% (Refinement) versus 10% (collaborative multilateration) on a scenario with just 5% anchors (200 nodes, connectivity 6).

Based on the experimental results it is recommending a number of guidelines for the installation of wireless sensor networks:

- To place anchors carefully (i.e. at the edges), and either
- To ensure a high connectivity (>10), or
- To employ a reasonable fraction of anchors (>5%).

This will create the best conditions for positioning algorithms in general and for Hop-TERRAIN and Refinement in particular.

6. CONCLUSIONS

In this paper, it has been presented a fully distributed algorithm for solving the problem of positioning nodes within

an wireless ad-hoc sensor network nods. The Hop-TERRAIN and Refinement algorithms were used in this problem. The simulation environment used to evaluate these algorithms is explained, including details about the specific implementation of each algorithm. Experiments were done in the simulation environment with respect to several aspects and the performance results achieved under different scenarios were analyzed for each algorithm. The results show that it was able to achieve position errors of less than 33% in a scenario with 5% range measurement error, 5% anchor population, and an average connectivity of 7 nodes. Finally, guidelines for implementing and deploying a network that will use these algorithms are given and explained.

An important aspect of wireless sensor networks is energy consumption. In future this paper can be implemented to analyze the amount of communication and computation induced by running Hop-TERRAIN and Refinement. The accuracy vs. energy consumption trade-off changes over subsequent iterations of Refinement also can be performed.

REFERENCES

[1] A. Nasipuri and K. Li. A Directionality Based Location Discovery Scheme for Wireless Sensor Networks. In 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02), p.p 105-111, Atlanta, GA, Sept. 2002.

[2] A. Savvides, H. Park, and M. Srivastava. The Bits and .ops of the n-hop Multilateration Primitive for Node Localization Problems. In 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02), pp. 112-121, Atlanta, GA, Sept. 2002.

[3] C. Savarese, J. Rabaey, and K. Langendoen. Robust Positioning Algorithm for Distributed Ad-hoc Wireless Sensor Networks. In USENIX Technical Annual Conf., pp. 317-327, Monterey, CA, June 2002.

[4] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks. Technical Report ucla/csd-tr-02-0013, UCLA Computer Science Department, 2002.

[5] D. Niculescu and B. Nath. Ad-hoc Positioning System. Proceedings of the Twenty Second Annual Joint Conference of the IEEE Computer and Communication Societies, INFOCOM 2003, **3**, 1734-1743, 2003.

[6] J. Hightower and G. Boriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, **34**(8): 57-66, 2001.

[7] L. Doherty, L. E. Ghaoui, and K. Pister. Convex Position Estimation in Wireless Sensor Networks. In Proc. Infocom 2001, **3**, 1655-1663, Anchorage, AK, April 2001.

[8] M. Chu, H. Haussecker, and F. Zhao. Scalable Information Driven Sensor Querying and Routing for Ad-hoc Heterogeneous Sensor Networks. *Int. Journal on High Performance Computing Applications*, **16**(3), 293-313, 2002.

[9] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, **7**(5): 28-34, 2000.

[10] S. Capkun, M. Hamdi, and J. Hubaux. GPS-free Positioning in Mobile Ad-hoc Networks. In 34th Hawaii Int'l Conf. on System Sciences (HICSS-34), 3481-3490, Maui, Hawaii, Jan. 2001.

[11] Y. Ko and N. Vaidya. Location-aided Routing in Mobile Ad-hoc Networks. In Proc. 4th Annual ACM/IEEE International Conference on Mobile Computing and Networks (Mobicom), Dallas, TX **6**(4) 307-321, 2000.

[12] Y. Shang, J. Meng, H. Shi, A New Algorithm for Relative Localization in Wireless Sensor Networks, 18th International Symposium on Parallel and Distributed Processing, 24-35, 2004.

[13] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere Connectivity. In ACM MobiHoc, 201-212, Annapolis, MD, June 2003.