Position Certainty Propagation: A Location Service for MANETs

Abdallah Sobehy[1], Eric Renault[1](✉), and Paul Muhlethaler[2]

[1] Samovar, CNRS, Télécom SudParis, University Paris-Saclay, 9 Rue Charles Fourier, 91000 Évry, France
{sobehy,eric.renault}@telecom-sudparis.eu

[2] INRIA Roquenourt, BP 105, 78153 Le Chesnay Cedex, France
Paul.Muhlethaler@inria.fr

Abstract. Localization in Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSNs) is an issue of great interest, especially in applications such as the IoT and VANETs. We propose a solution that overcomes two limiting characteristics of these types of networks. The first is the high cost of nodes with a location sensor (such as GPS) which we will refer to as anchor nodes. The second is the low computational capability of nodes in the network. The proposed algorithm addresses two issues; self-localization where each non-anchor node should discover its own position, and global localization where a node establishes knowledge of the position of all the nodes in the network. We address the problem as a graph where vertices are nodes in the network and edges indicate connectivity between nodes. The weights of edges represent the Euclidean distance between the nodes. Given a graph with at least three anchor nodes and knowing the maximum communication range for each node, we are able to localize nodes using fairly simple computations in a moderately dense graph.

Keywords: Location service · MANETs · WSNs

1 Introduction

Localization in MANETs has been the focus of attention in recent years as it is used in various applications such as routing, autonomous air-vehicles, network security, environment surveillance for military purposes, etc. For instance, in [1–3], the location information is used in various ways to improve routing decisions. A straightforward method to locate network nodes is to equip each of them with a location sensor such as a GPS and broadcast the location information. In SFLS location service [4], the location information is forwarded with higher frequency to close neighbors in terms of the number of hops and lower frequency to further nodes to decrease bandwidth consumption while maintaining adequate knowledge of neighbors' positions. Since IoT networks have limiting characteristics, it is rather expensive to equip each node with a GPS sensor. Thus, different solutions attempt to locate nodes where only a subset of network nodes are equipped with a location sensor.

To estimate the position of non-anchor nodes, additional information is required to relate nodes to each other. According to the classification specified in [5], node relations can be as simple as being in connection with anchor nodes or not. This is used in [6] where a node's position is assumed to be the average of the known positions of the other nodes. Another approach is range-based where the distance between nodes is computed either through RSSI (Received Signal Strength Indicator), TOA (Time Of Arrival) or TDOA (Time Difference Of Arrival) [7]. TOA and TSOA require external hardware to synchronize a transmitter and a receiver, while RSSI does not since the distance is derived from a signal attenuation model that relates the signal strength to the distance [7]. Knowing the relative distances is obviously an added benefit that helps to improve the accuracy of the location. However, using RSSI is subject to errors due to environmental factors (indoors or outdoors), multi-path fading and noise [7]. In [8], RSSI was found to vary consistently with a distance up to 50 meters. In [5,9,10], the probability distribution is integrated in the location process instead of concluding a single position for each node. Such a method allows the uncertainty of measurements to be taken into account, be it is the GPS position or the relative distance between nodes [5].

2 Related Work

Consider the location problem as an undirected graph $G = (V, E)$ where V is the set of the nodes in the network and E is the set of edges connecting two nodes located within communication range of each other. In this context, the given information is the position of anchor nodes and edge weights which are relative distances between nodes within communication range. The aim is to compute the position of all non-anchor nodes. This can be seen as an optimization problem that can be solved using numerical techniques such as SDP [11] or linear programming [12]. To avoid inaccuracies of signal propagation models when using RSSI for distance measurements, some studies [13–15] have used interval-analysis. In interval-based analysis, instead of computing direct distances from RSSI, a set of inequalities is formulated to indicate that a node is on a ring between 2 radii based on its relative position to other nodes. Then, the defined areas for nodes can be further restrained using the Waltz algorithm [16].

Some approaches divide the problem into sub problems as in [8] where (1) base stations classify ordinary nodes into clusters based on their proximity to anchor nodes and (2) within each cluster, a node seen by three anchor nodes is located using a simple geometrical computation. In [7], the region where the network is deployed is divided into rectangular grids as a first step, then within each small grid the location is refined. However, in the previously mentioned approaches the number of GPS nodes needs to be high in order to satisfy the necessary constraints.

Generally speaking, if the distance of a non-anchor node to three anchor nodes is known, it can be located with simple geometric computations except in the rare cases where the nodes are collinear or when some nodes overlap. This might imply a conclusion that at least three GPS nodes in the network are needed for location. However, in [5], the proposed solution attempts to locate the network with a single anchor node. Each node starts with a uniform probability distribution over the deployment region and as the roaming anchor node passes by the ordinary nodes it tweaks the distribution to locate nodes. In this context, the authors experiment their solution where the anchor nodes use a random model to traverse the network and have a reasonable claim that traversing the network can be optimized to improve the location process. In [5,9] negative information about the absence of a node in the proximity of the anchor nodes is used in the location process. This kind of information is used as a basis for our algorithm.

An approach which is seemingly far from the mentioned methods, yea interesting to address the node location problem, is Graph Layout Algorithms such as FDP [17] and neato [18]. Even though the objective of these algorithms is generally to create an easy-on-the-eye graph, some variations make it possible to fix the positions of some nodes (the anchor nodes) and to set the suitable edge length (i.e., is the distance between nodes). We have tested these algorithms using graphviz [19] and the estimated node positions are satisfactory.

Other methods, which use a set of distance equations and optimization techniques usually require high computation power, which is impractical for these networks. In this case, the computation is done remotely in a centralized fashion. The graph structure might not yield a unique solution. Even if the graph has a unique solution, finding this solution is proved to be NP-hard [20].

We address the problem from the graph point of view aiming at pinpointing the location of as many anchor nodes as possible. We first introduce our algorithm. Then we move on to explain how it can be implemented distributively over the nodes.

In Sect. 3, the system model is presented and the centralized version of the algorithm is introduced. In Sect. 4, the distributed version of the algorithm is explained and how each node acquires awareness of its location and the location of other nodes. Next, the algorithm is compared with an existing method in Sect. 5. Section 6 contains the conclusion, and possible future work.

3 System Model

3.1 Overview

Let us consider the network as an undirected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ are the nodes of the network and $E = \{(0, 1), (0, 2)...\}$ where each pair represents an edge connecting two nodes located within the communication range of each other. Each edge is assigned a weight indicating the distance between the two nodes. We assume a subset of the network nodes m are anchor nodes where m includes at least three nodes of the network and for our evaluation we use exactly three anchor nodes. We also assume there is at least one node within communication range of at least three anchor nodes.

3.2 Position Certainty Propagation Algorithm

To illustrate the algorithm, let's relate its steps to a 15-node network with three anchor nodes in a region of 20×20 square meters and a maximum communication range of 8 meters. Figure 1 presents an example of such a network.

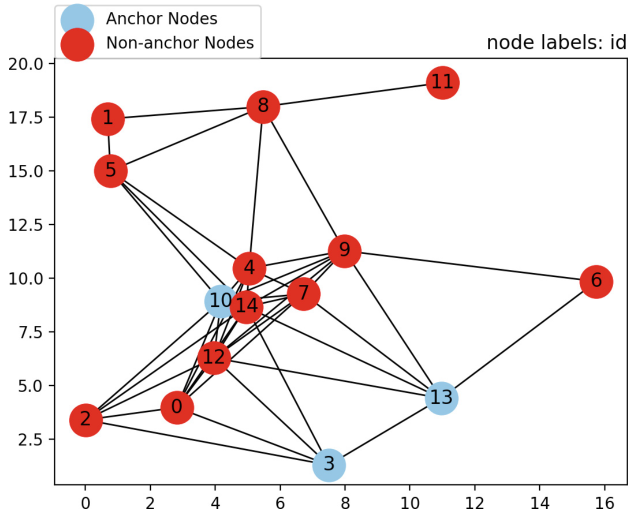

Fig. 1. 15-node network example

The algorithm starts by computing the position of the nodes with the most anchor nodes (or nodes with found positions) in their vicinity. Thurs, the first step is computing for each node the number of the neighbours with known position (whether anchor nodes or computed position nodes). Recall that for the algorithm to start, there must exist at least one node with at least three anchor nodes in its vicinity. Each anchor node forms a circle centered at the anchor node with a radius equal to the edge weight between the anchor node and the node in question. The node position is chosen to be the intersection of the three circles. Whenever a node position is found, it changes its state to a known position node. Whenever a node position is computed, all nodes recompute the number of neighbors with known positions in their vicinity. Then, the algorithm computes the position of the other nodes with the most known position nodes in their vicinity. Figure 2 illustrates the step of computing the number of nodes with known positions in vicinity (left) and the computation of the position of node 12 by computing the intersection of the three circles formed by the known position nodes: 3, 10 and 13 (right).

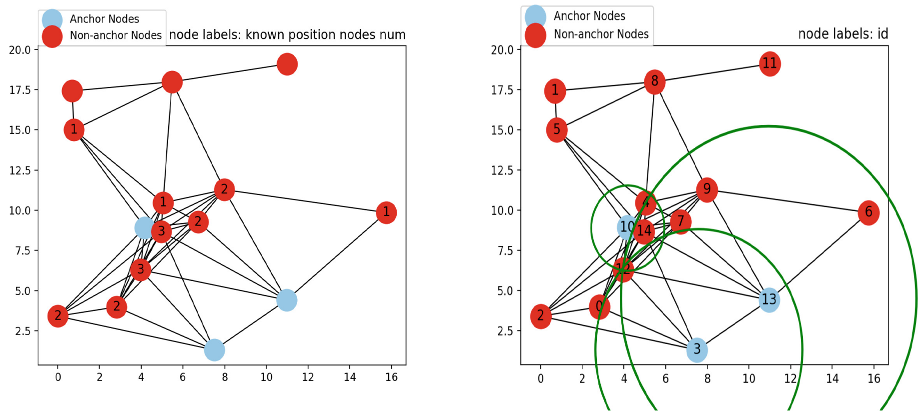

Fig. 2. Counting number of known position neighbours (left). Computing position from the intersection of three circles (right)

As the position of nodes are computed, the probability of finding nodes with known positions in the vicinity increases. Continuing to compute the position of nodes, one can reach the case where the maximum number of known position nodes in the vicinity is 2 for all nodes in the network. In this case, the intersection between the two circles formed by the two known position nodes in the vicinity would generally yield two possible positions. To choose one of these two positions, the node is assumed to be at both positions and the set of neighbors from the known position nodes that are within a radius of maximum communication range is computed at each of the two potential positions. The position where wrong nodes would have been neighbors, i.e. if one potential node position would be within the communication range of another node n while n is not actually sensed as a one-hop neighbor, is eliminated and the other position is chosen for this node. Figure 3 shows this case and the simulated two positions for node 6 are presented as blue octagons. The lower of the two possible positions can be eliminated because at this position node 6 would have had other neighbors which are not it actual neighbors in the network.

There might be cases where the elimination is not needed e.g. when the two generated circles are tangent at one point or when of the two positions are out of the deployment region. The algorithm continues to compute the position of nodes with the two previously mentioned cases until no more positions can be estimated. This is the case when a node sees two known position nodes within its vicinity but neither of the two possible positions introduces wrong neighbors and thus cannot be eliminated. Also, when a node is seen only by one known position neighbor, it may be located anywhere on the circle centered on the known position node and of radius the distance between the two nodes. Therefore, its position cannot be effectively established. This can be seen when attempting to compute the position of node 11, where its only known position

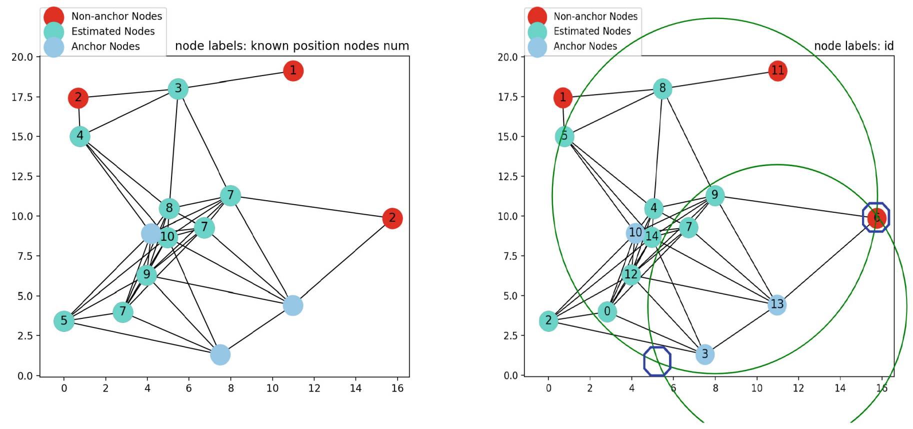

Fig. 3. Counting number of known position neighbours (left). Computing position from the intersection of two circles (right) (Color figure online)

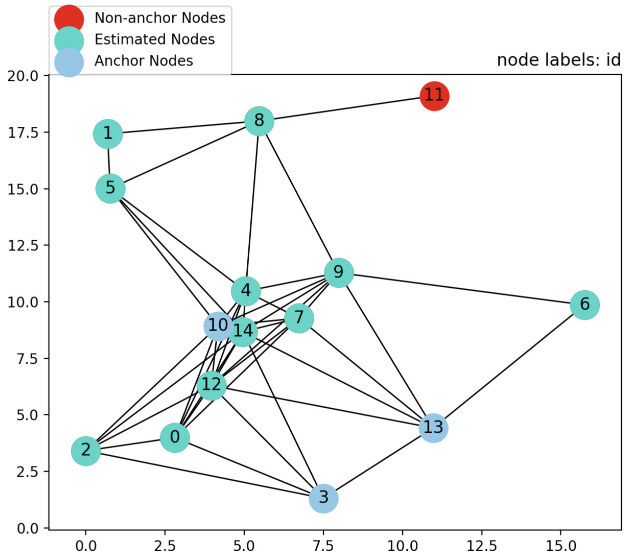

Fig. 4. The final result of the algorithm

neighbor is node 8. The final outcome of the algorithm in Fig. 4 shows that only node 11's position cannot be computed.

The main advantage of this algorithm is its lightness as it comes down to a series of triangulation steps. The next section discusses the implementation of the proposed solution in a distributed manner.

4 Distributed Implementation of the Algorithm

To cope with the distributive nature of MANETs, we introduce here a possible algorithm to run at each node either to establish its own position or the position of all the nodes in the network. First, we assume that a node with a known position broadcasts its own position through the network. To mitigate overloading the network bandwidth, position updates are spread over the network using a bandwidth conserving method like SFLS [4]. This method is based on sending the updates with the highest frequency to one-hop neighbors, and for each increment of hop-count the frequency of updates is halved. This method is compatible with our approach since the information needed to eliminate wrong positions relies on neighbors which are a few hops away. A node that needs to compute its own position receives position information from other nodes. It also computes the distance to one hop neighbors using any of the previously mentioned methods. When sufficient information has been received, it computes its own position as shown in Algorithm 1.

Algorithm 1. Position Certainty Propagation

```
1: $N$ (list of received node positions) and $N_1$ (1-hop neighbours with known positions)
2: $Id_1$ list of IDs of all one-hop neighbors
3: procedure RECEIVE($sender, n, pos, timeStamp$)
4:     if $sender == n$ and ($n$ not in $N_1$ or $timeStamp > N_1[n].timeStamp$) then
5:         update $N_1$ with $n$, $pos$ and $timeStamp$
6:     end if
7:     if $n$ not in $N$ or $timeStamp > N[n].timeStamp$ then
8:         update $N$ with $n$, $pos$ and $timeStamp$
9:     end if
10: end procedure
11: do
12:     $receive(sender, n, pos, time())$
13:     if $N_1$ has 3 or more nodes then
14:         $selfPos = intersection(N_1)$
15:     else if $N_1$ has 2 nodes then
16:         [pos1, pos2] = $intersection(N_1)$
17:         $P1Neighbors =$ n for each $n \in N$ where $distance(n, pos1) <= CommRange$
18:         $P2Neighbors =$ n for each $n \in N$ where $distance(n, pos2) <= CommRange$
19:         $pos1IsCompatible \iff$ for each $n \in P1Neighbors$, $n \in Id_1$
20:         $pos2IsCompatible \iff$ for each $n \in P2Neighbors$, $n \in Id_1$
21:         if $pos1IsCompatible$ and not $pos2IsCompatible$ then
22:             $selfPos = pos1$
23:         else if $pos2IsCompatible$ and not $pos1IsCompatible$ then
24:             $selfPos = pos2$
25:         end if
26:     end if
27: while $selfPos$ not found
```

Let N_1 be the list of detected known position neighbors within communication range of the node. When N_1 includes two or three nodes, the position of the node can be made certain using the *intersection* method. If N_1 includes two nodes, the *intersection* method returns the two possible positions from which one is to be eliminated if possible. Let N be the list containing position information of all received positions of network nodes. The *receive* method is called when a message is received from a *sender* node within communication range that contains the node ID n and the corresponding position pos. Id_1 contains the list of IDs of all one-hop neighbors whether their position is known or not.

When positions are computed, the nodes start broadcasting their own position using SFLS [4]. Also, to take into account the continuous change of node positions, nodes should re-run the position computation algorithm whenever a change in the parameters used to compute its position is received. To make it clearer, if any of the three nodes used to compute a node's position send a message indicating a position change, the node should update its own position.

5 Performance Evaluation

5.1 Comparable Method: Gps-Free

In order to assess our algorithm we compare it with an existing solution [21], which we will refer to as Gps-free. We implemented the first part of the Gps-free algorithm where node positions are computed in a local coordinate system of one of the nodes in the network. The paper further explains how to choose a stable coordinate system for the network, which does not concern us since we evaluate the solution for a static instance of the network. The algorithm compromises the following steps:

1. Each node creates a local map composed of itself and the maximum possible number of 1-hop neighbours via triangulation making itself the origin.
2. Node k can transfer its coordinate system to node i if both nodes exist in the local map of one another in addition to a third node common in both local maps.
3. All nodes in the network attempt to transfer their coordinate system to node i so that all node positions are computed in one common coordinate system.

We refer the reader to the Gps-free article [21] for further details of how these steps are executed. Observing the behaviour of Gps-free in some graphs we figured an improvement that can increase the percentage of localized nodes. The improvement concerns the condition that only nodes who cannot build a local map are transferred to another coordinate system via triangulation. We quote from the Gps-free paper: "The nodes that are not able to build their local coordinate system but communicate with three nodes that already computed their positions in the referent coordinate system can obtain their position in the Network Coordinate System by triangulation" [21]. We extended this behaviour to nodes who built their local map but still cannot transfer their local coordinate system to the referent coordinate system; only the node compute its position in the referent coordinate system.

5.2 Experimental Setup

Our objective is to compute the positions in the global coordinate system using GPS information of three nodes. In the Gps-free method, even though the positions are computed in a local coordinate system, it is possible to transfer the node positions to the global coordinate system if the 3 GPS nodes have their positions computed. Each simulation run is a connected geometric graph (aka disc graph) where nodes are randomly positioned in a $100\,\mathrm{m} \times 100\,\mathrm{m}$ grid. Three anchor nodes are randomly chosen so that at least one non-anchor node is within their communication range. Our algorithm is run to compute the positions of the non-anchor nodes as previously described. Gps-free is run where the node chosen to have all nodes transferred to its coordinate system is the one the maximizes the number of estimated nodes, this ensures extracting the best possible result from the Gps-free method without taking into account whether the GPS nodes are among the nodes with the computed positions. The success percentage is the percentage of nodes with their position computed. Two experiments are conducted, one varying the maximum communication range $[14, 15, 16.., 23]$ while keeping the average number of nodes constant at 100 nodes. In the other experiment the communication range is kept constant at 14 m while the average number of nodes is varied $[100, 115, 130, 145, 160]$. The number of nodes in each experiment follows a poisson distribution with the given average. In each graph configuration the experiment is repeated 1000 times and the confidence interval is shown as a vertical bar around the point.

5.3 Experimental Results

Figure 5 shows the comparison between PCP, Gps-free and Gps-free extended version. We start with a maximum communication range of 14 m and increment by 1 m for each new configuration. We show on the graph the average node degree at each maximum communication range. Put differently, for the first configuration each node has a maximum communication range of 14 m and the average node degree over the 1000 simulations is approximately 5.5, the next configuration is with a range of 15 m which gives an average node degree of $\sim$6.3 etc. It can be clearly seen that the maximum communication range has a direct impact on the number of localized nodes. As the maximum communication range increases and consequently the average node degree, it is possible to estimate positions of more nodes. When the average degree is $\approx$ 10, our algorithm is able to compute the positions of $\approx$ 90% of nodes. Also, our algorithm shows a higher success percent at all configurations compared to Gps-free and the extended version. In another attempt to study the effect of varying the number of nodes keeping the maximum communication range constant at 14 meters, increasing the number of nodes has a similar effect to increasing the maximum communication range. Here, the node density (nodes/meter^2) is shown against the success rate.

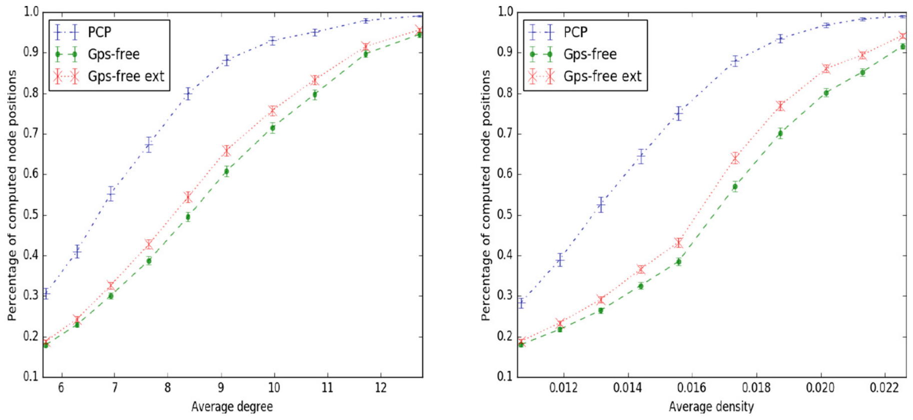

Fig. 5. Comparing PCP to Gps-free and Gps-free ext methods

6 Conclusion

This article presented a simple algorithm to compute the positions of nodes when a network has a very limited number of anchor nodes (the minimum is set to 3). The algorithm has an initial condition, which is the presence of a node that is within the vicinity of at least three anchor nodes. It might appear that the starting condition is hindering the utility of the approach, however it can be partially mitigated by assuming three non-anchor nodes that are within each other's vicinity and satisfy the initial condition as anchor nodes. These virtual anchor nodes are given assumed positions so that they form a triangle from the known distance between them. For instance, assume non-collinear nodes i, j, k see each other and at least a forth node. Node i is positioned at the origin, node j is on the horizontal axis with at a distance equal to $dist(i, j)$ and node k is added to have a positive y-value using triangulation. The algorithm then treat them as anchor nodes and compute the position of the rest of the nodes as previously explained. When at least three real anchor node positions are computed, all the nodes can then be transferred to the global coordinate system using there computed positions and actual positions [21]. It has been shown statistically that with a fairly dense network (average degree $\approx$ 10), the algorithm is able to quickly and efficiently compute the position of $\approx$ 90% the nodes whose positions are not known. The next step is to test our algorithm with the inevitable uncertainty of measurements using a tool such as NS3. Additionally, the impact of the node's velocity, delay of receiving information on the accuracy of the estimation of the node's position are also to be considered.

References

1. Jung, W.-S., Yim, J., Ko, Y.-B.: QGeo: Q-learning-based geographic Ad Hoc routing protocol for unmanned robotic networks. IEEE Commun. Lett. **21**(10), 2258–2261 (2017)
2. Terao, Y., Phoummavong, P., Utsu, K., Ishii, H.: A proposal on void zone aware greedy forwarding method over MANET. In: 2016 IEEE Region 10 Conference (TENCON), pp. 1329–1333. IEEE (2016)
3. Ko, Y.B., Vaidya, N.H.: Location aided routing (LAR) in mobile ad hoc networks. Wireless Netw. **6**(4), 307–321 (2000)
4. Renault, E., Amar, E., Costantini, H., Boumerdassi, S.: Semi-flooding location service. In: 2010 IEEE 72nd conference on Vehicular Technology Conference Fall, VTC 2010-Fall, pp. 1–5. IEEE (2010)
5. Huang, R., Záruba, G.V.: Monte Carlo localization of wireless sensor networks with a single mobile beacon. Wirel. Netw. **15**(8), 978 (2009)
6. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low-cost outdoor localization for very small devices. IEEE Pers. Commun. **7**(5), 28–34 (2000)
7. Chen, Z., Xia, F., Huang, T., Fanyu, B., Wang, H.: A localization method for the internet of things. J. Supercomput. **63**(3), 657–674 (2013)
8. Sallouha, H., Chiumento, A., Pollin, S.: Localization in long-range ultra narrow band IoT networks using RSSI. In: 2017 IEEE International Conference on Communications (ICC), pp. 1-6. IEEE (2017)
9. Peng, R., Sichitiu, M.L.: Robust, probabilistic, constraint-based localization for wireless sensor networks. In: 2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005, IEEE SECON 2005, pp. 541–550. IEEE (2005)
10. Sichitiu, M.L., Ramadurai, V.: Localization of wireless sensor networks with a mobile beacon. In: 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp. 174–183. IEEE (2004)
11. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, pp. 46-54. ACM (2004)
12. Larsson, E.G.: Cramer-Rao bound analysis of distributed positioning in sensor networks. IEEE Sig. Process. Lett. **11**(3), 334–337 (2004)
13. Mourad, F., Snoussi, H., Abdallah, F., Richard, C.: Guaranteed boxed localization in MANETs by interval analysis and constraints propagation techniques. In: Global Telecommunications Conference 2008, IEEE GLOBECOM 2008, pp. 1–5. IEEE (2008)
14. Mourad, F., Snoussi, H., Abdallah, F., Richard, C.: Model-free interval-based localization in manets. In: IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009. DSP/SPE 2009, pp. 474–479. IEEE (2009)
15. Mourad, F., Snoussi, H., Richard, C.: Interval-based localization using RSSI comparison in MANETs. IEEE Trans. Aerosp. Electron. Syst. **47**(4), 2897–2910 (2011)
16. Waltz, D.L.: Generating semantic descriptions from drawings of scenes with shadows (1972)
17. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force directed placement. Softw.: Pract. Exp. **21**(11), 1129–1164 (1991)
18. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inf. Proc. Lett. **31**(1), 7–15 (1989)

19. Ellson, J., Gansner, E., Koutsofios, L., North, S.C., Woodhull, G.: Graphviz—open source graph drawing tools. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 483–484. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_57
20. Eren, T., et al.: Rigidity, computation, and randomization in network localization. In: INFOCOM 2004, Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 4, pp. 2673–2684. IEEE (2004)
21. Čapkun, S., Hamdi, M., Hubaux, J.-P.: GPS-free positioning in mobile ad hoc networks. Cluster Comput. **5**(2), 157–167 (2002)