Position certainty propagation: a localization service for Ad-Hoc Networks

Abdallah Sobehy, Eric Renault, Paul Mühlethaler

computers

MDPI

Article

Position Certainty Propagation: A Localization Service for Ad-Hoc Networks

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; eric.renault@telecom-sudparis.eu

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

* Correspondence: sobehy@telecom-sudparis.eu

Received: 11 November 2018; Accepted: 2 January 2019; Published: date

Abstract: Location services for ad-hoc networks are of indispensable value for a wide range of applications, such as the Internet of Things (IoT) and vehicular ad-hoc networks (VANETs). Each context requires a solution that addresses the specific needs of the application. For instance, IoT sensor nodes have resource constraints (i.e., computational capabilities), and so a localization service should be highly efficient to conserve the lifespan of these nodes. We propose an optimized energy-aware and low computational solution, requiring 3-GPS equipped nodes (anchor nodes) in the network. Moreover, the computations are lightweight and can be implemented distributively among nodes. Knowing the maximum range of communication for all nodes and distances between 1-hop neighbors, each node localizes itself and shares its location with the network in an efficient manner. We simulate our proposed algorithm in a NS-3 simulator, and compare our solution with state-of-the-art methods. Our method is capable of localizing more nodes ($\approx$ 90% of nodes in a network with an average degree $\approx$ 10).

Keywords: localization service; MANETs; WSNs

1. Introduction

Localization is the process of acquiring the position information of an entity in a coordinate system. Knowledge of the position of network nodes is invaluable for numerous applications, such as routing, autonomous air-vehicles, network security, environment surveillance for military purposes, and so on. The network is modeled as an undirected graph, where vertices represent a set of nodes and edges are links between two nodes within communication range. Nodes which are aware of their coordinates or position are known as anchor nodes. The aim is to compute the position of other nodes within the network with the help of these anchor nodes. We attempt to achieve this goal by using the position information of anchor nodes and the relative distance between their one-hop neighbors. Since, in some ad-hoc networks (e.g., IoT), the devices have limited resources, it is computationally expensive to equip each node with a localization sensor such as GPS. Thus, we attempt to localize nodes where only a subset are equipped with localization sensors.

Using the distance between one-hop neighbors to localize nodes is known as range-based localization, where the distance between nodes is computed either through Received Signal Strength Indicator (RSSI), Time Of Arrival (TOA), or Time Difference Of Arrival (TDOA). TOA and TDOA require external hardware to synchronize a transmitter and a receiver, while RSSI does not need additional hardware as the distance is derived from a signal attenuation model which relates signal strength to the distance. However, using RSSI is subject to error due to environmental factors (indoors or outdoors), multi-path fading, and noise [1]. In [2], RSSI was found to vary consistently, with a distance up to 50 m.

Considering the resource limitation of the network nodes, we propose a lightweight solution based on a series of triangulations. If a non-anchor node lies within the vicinity of three nodes whose positions are known, it can efficiently estimate its own position via triangulation, because the distances to these nodes are known. The node then shares its position with other nodes, to further propagate the location estimation process. However, in some less connected regions of the network, this condition will not hold, prohibiting the location discovery for existing solutions (such as [3,4]). Our contribution involves the use of triangulation to estimate the coordinates of non-anchor nodes within the vicinity of exactly two anchor nodes. The triangulation yields two possible positions for non-anchor nodes. We introduce a mechanism which attempts to eliminate one of the positions, to localize non-anchor nodes in less connected regions of the network. This approach improves significantly the localization of unknown positions of network nodes.

2. Related Work

Localization of network nodes is crucial for plethora of applications; for instance, improvement of routing techniques [5–7]. A straightforward method to address the localization problem is to equip all nodes with a location sensor (such as a GPS), and share the location information. There are two main paradigms to share position information: Rendez-vous-based and flooding-based. In rendez-vous-based methods, some nodes are elected to store node position information and respond to inquiries when position information is requested. The disadvantage of such a method is that it centralizes the information in only a few nodes, and thus there is a risk of information loss if such nodes break down [8]. Flooding-based methods overcome this disadvantage by broadcasting node position information to the network. The cost involved in such an approach is excessive message exchange, contributing to network overhead. In Semi-Flooding based Location Service (SFLS) [8], the location information is forwarde with higher frequency to close neighbors (in terms of the number of hops) and lower frequency to further nodes, in order to decrease bandwidth consumption while maintaining adequate knowledge of neighbors' positions.

In the case where not all nodes are equipped with a GPS, localizing non-anchor nodes require additional information to relate nodes to each other. According to the classification specified in [9], node relations can be connectivity-based, which simply indicates whether a node is in connection with anchor nodes or not. This is used in [10], where a node's position is assumed to be the average of the known positions of the other nodes. Another type of inter-node relation is the distance between one-hop neighbors, which is widely used to estimate the positions of non-anchor nodes [3,4]. Relative distance information is an added benefit, which helps to improve the accuracy of the location. In addition, some solutions use the Angle-of-Arrival to relate nodes, or a combination of the mentioned measurement types [11].

The localization problem can also be seen as an optimization problem, which can be solved using numerical techniques such as Semi-Definite Programming (SDP) [12] or linear programming [13]. To avoid the inaccuracies of signal propagation models when using RSSI for distance measurements, some studies [14–16] 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 [17].

Some approaches divide the problem into subproblems, as in [2], 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 [1], 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. In [9,18,19], the probability distribution is integrated in the location process, rather than attributing a single position for each

node. Such a method allows the uncertainty of measurements to be taken into account—be it the GPS position or the relative distance between nodes [9].

Generally speaking, if the distance of a non-anchor node to three anchor nodes is known the node can be located with simple geometric computations, except in 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 [9], the proposed solution attempts to localize 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 tweak the distribution to localize the 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 [9,18], 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 above-mentioned methods, yet interesting to address the node location problem, are Graph Layout Algorithms such as FDP [20] and neato [21]. 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 a suitable edge length (i.e., the distance between nodes). We have tested these algorithms using graphviz [22], 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 has been proved to be NP-hard [23].

The presented work is an extension of our previous work on localization [24]. We address the problem from the graph point of view, aimed at pinpointing the location of as many nodes as possible. In our approach, we first consider perfect distance measurements and compare it to an existing method [4], and show that our method is able to localize more nodes in a randomly generated network by exploiting the case where a non-anchor node is within vicinity of only two localized nodes. Next, we introduce Gaussian noise to distance measurements to simulate the effect of real world errors in distance measurements. Gaussian noise is widely accepted in simulating distance errors, and is experimentally validated in [25]. Table 1 summarizes different approaches to solve the localization problem.

Table 1. Localization approaches.

Approach	Cost of Sensors	Comments
Flooding based	Expensive	Information well distributed, but bandwidth-consuming.
Rendez-vous based	Expensive	Centralized, but bandwidth-friendly.
Semi-Flooding [8]	Expensive	Sweet spot between flooding and rendez-vous.
Connectivity based [10]	Not Expensive	Low complexity, but low accuracy.
Optimization [12,13]	Not Expensive	Adequate accuracy, but dependent on environment noise.
Interval analysis [14–16]	Moderately Expensive	Adequate accuracy, and mitigates environment noise.
GPS-less [4]	Not Expensive	Adequate accuracy, and position information is in a local coordinate system.
PCP (proposed method)	Not Expensive	Adequate accuracy, and high localization percentage.

In Section 3, the system model is presented and the principal localization steps of the algorithm are introduced. In Section 4, we explain the distributed implementation of the algorithm, which shows how each node acquires awareness of its location and the location of other nodes. Next, the algorithm is compared with the GPS-free [4] method, which is widely referenced in the literature and has some

similarities with our method, allowing clear comparison and explanation of results. Next, the behavior of our method is studied, with distance measurement errors, in the NS-3 [26] network simulator in Section 5. Section 6 contains the conclusion and discusses possible future work.

3. System Model

3.1. Overview

The network can be considered as an undirected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ represent the nodes, each with a given communication range. For simplicity, we assume that all nodes have the same communication range. Any two nodes within a distance smaller than or equal to the communication range has an edge constructed between them, representing two-way communication. The edge is assigned a weight, which represents the distance between the nodes. It is assumed that the network contains at least three anchor nodes, which are within communication range of at least one non-anchor node.

3.2. Position Certainty Propagation Algorithm

The self-localization process is composed of two major cases: (a) When the positions and distances to three or more one-hop neighbors are known, and (b) when the positions and distances to exactly two one-hop neighbors are known. The latter is considered a bottleneck, which inhibits other algorithms [3,4] from localizing nodes in less-connected regions in the network, and thus the position discovery does not propagate.

3.2.1. Three or More Nodes With Known Position in Their Vicinity

When a node collects enough position information through position broadcasts of anchor/localized nodes, so that at least three one-hop neighbor positions are known, it can estimate its position (via triangulation) using the distance to each of these nodes. The localization process can be viewed as the intersection of circles whose centers are nodes with known positions and radii being distances from the known position nodes to the node to be localized. This is visualized in Figure 1, which shows a 15-node network in a 20 m $\times$ 20 m region, where each node has a maximum communication range of 8 m. In this example, three anchor nodes $[10, 13, 3]$ are used to localize node 12.

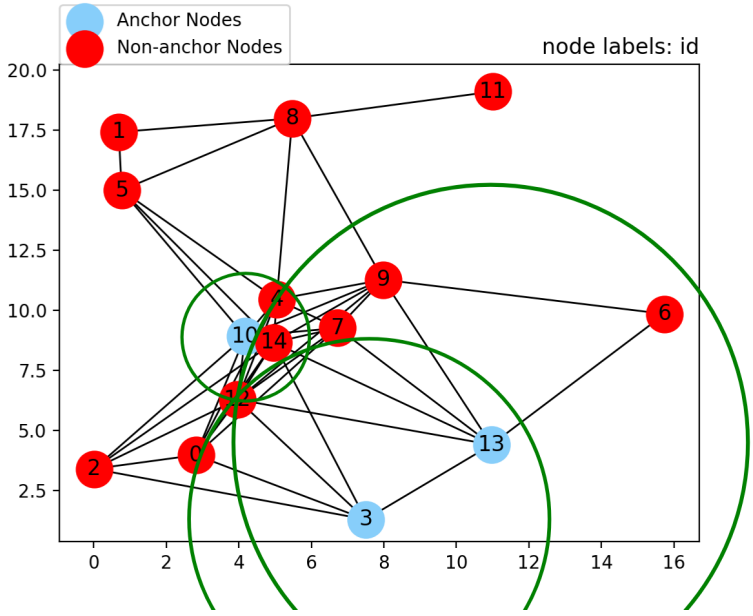

Figure 1. Computing position, based on three known position nodes.

The centers of the green circles are the positions of the anchor nodes $[10, 13, 3]$. The radius of each circle is the distance between the anchor node at its center and the node 12. However, due to measurements errors, the three circles do not necessarily intersect at one exact point. To overcome this, we use a variant of the residual weight algorithm [27]. This algorithm was originally implemented to mitigate the effect of Non-Line-Of-Sight (NLOS) measurement errors. When the distance and position information of more than three nodes is collected, an estimated position is computed for each possible combination of three or more nodes, using least-squares error minimization. To illustrate this, for position/distance information of four nodes, $\binom{4}{3}$ positions are computed for all three-node combinations, and one position for the four nodes together. Each estimated position is given a weight that is inverse to the normalized residual error. Finally, the chosen position is the weighted average of all positions (another possibility is to choose the position with the minimum residual error). Since minimizing the error does not always guarantee an estimation that is the closest to the actual position [3], we choose not to use least-squares error minimization, in order to minimize computation effort. Rather, for position/distance information of three nodes, we compute a position for each combination of node pairs (in this case, three combinations) by computing two intersections of the two circles and eliminating one of the positions, using the third circle. Then, the weighted average of the three positions is chosen as the estimation. Let us consider a node pair $[A, B]$, whose known positions are $[a.x, a.y]$ and $[b.x, b.y]$, respectively. The distances to node C, whose position is to be computed, are ac and bc, respectively. First, the distance between A and B is computed:

$$ab = \sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}. \tag{1}$$

The intersection between two circles generally yields two positions. The computation of the two possible positions is easier if the two circles lie on the horizontal axis and one of them is at the origin. Thus, $\vec{a}$ is subtracted from a and b. Next, the rotation angle shown in Equation (2) is applied, so that a is transferred to the origin and b resides on the x-axis.

$$\theta_{rot} = arctan\left(\frac{b.y - a.y}{b.x - a.x}\right) \tag{2}$$

After applying the translation and rotation, the two intersection points will have the same x-coordinate. The y-coordinate for one intersection point is above the x-axis and the other is below the x-axis. The rotated coordinates of the intersection points are shown in Equations (3) and 4.

$$c.x_{rot} = \frac{ab^2 + ac^2 - bc^2}{2ab} \tag{3}$$

$$c.y_{rot} = \pm\frac{\sqrt{(ab + ac + bc) \times (ab + ac - bc) \times (ab - ac + bc) \times (-ab + ac + bc)}}{2ab} \tag{4}$$

Finally, to restore the intersection points in the original coordinate system, the opposite rotation and translation are applied to the rotated coordinates, as follows:

$$c = \begin{bmatrix} \cos(\theta_{rot}) & -\sin(\theta_{rot}) \\ \sin(\theta_{rot}) & \cos(\theta_{rot}) \end{bmatrix} \begin{bmatrix} c.x_{rot} & \pm c.y_{rot} \end{bmatrix} + \vec{a}. \tag{5}$$

The next step is to eliminate one of the two positions using the third position/distance pair $[d, dc]$ of node D, whose position is also known. The elimination process consists simply of choosing the position for which the distance to the third node is almost equal to the distance dc. The error for the estimated known position is the absolute difference between dc and the distance between the chosen position and d.The weight given for the chosen position is $1/error$. Finally, the estimated position is the weighted average of the chosen positions. If there are more than three one-hop neighbors with known positions, the process is done for all 3 one-hop neighbor combinations, and the estimate chosen is that which yields the least error.

It should be noted that this process is not possible if the three nodes with known positions are collinear. In this case, two of the three nodes are chosen, so that the largest angle of the triangle of the two nodes (with known positions) and the node to be localized is as close as possible to 90°. This criterion is chosen so that the two possible positions of the intersection of two circles are further from each other, which helps in the elimination step that follows. The localization process using the positions and distances to only two one-hop neighbors is explained in the following subsection.

3.2.2. Two Nodes With Known Positions in the Vicinity

The first step is to compute the two possible positions, as previously shown in Equations (1) to (5). The information required to eliminate one of the two positions is the maximum range of communication, a list of received positions of other nodes in the network, and the IDs of one-hop neighbors. The position to be estimated is expected to be within the maximum range of the one-hop neighbors, and outside the maximum range for n-hop neighbors (where $n > 1$). This requires that the node stores the IDs of all one-hop neighbors, and updates them frequently. Also, the received positions of neighbors in the network are stored in a table.

In order to eliminate a position, the distances to all stored positions of other nodes in the network are computed. If a computed distance to a one-hop neighbor is larger than the maximum range of communication, the position is eliminated because it implies that this 1-hop neighbor is outside the maximum range of communication. On the other hand, if the computed distance to an n-hop neighbor is smaller than the communication range, the position is also eliminated because it implies that this n-hop neighbor is one hop away. If one of the two positions is eliminated, the other position is chosen to be the estimated position. If neither of the two positions is eliminated, the estimation is discarded until more information is available.

To illustrate, we continue the localization process of nodes in the 15-node network previously shown in Figure 1. The continuous localization of nodes propagates, until there are no more nodes that can be localized with three or more nodes in their vicinity. This state is shown in Figure 2, where node 6 has only two nodes with known positions in its vicinity, 9 and 13. The two possible positions from the two circle intersections are shown as blue octagons. Since node 6 is aware that it has only two 1-hop neighbors, computing the distance between the lower left blue octagon to any node with known position (e.g., node 0) will yield a distance smaller than the maximum communication range. Since node 0 does not belong to the list of 1-hop neighbors, this position is eliminated and the other position is chosen to be the estimate, as it does not introduce any conflicts. Similarly, node 1 can be localized using nodes 5 and 8.

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 nodes with known positions within its vicinity, but neither of the two possible positions introduces conflict and thus cannot be eliminated. Also, when a node is seen only by one neighbor with known position, it may be localized anywhere on the circle centered at the node with known position and whose radius is 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 neighbor with known position is node 8. The final outcome of the algorithm in Figure 3 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.

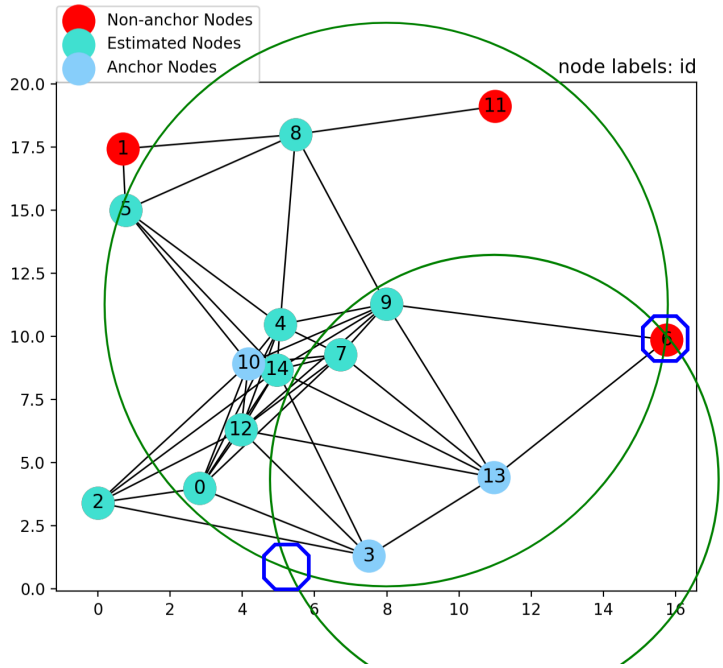

Figure 2. Computing position, based on two nodes with known positions.

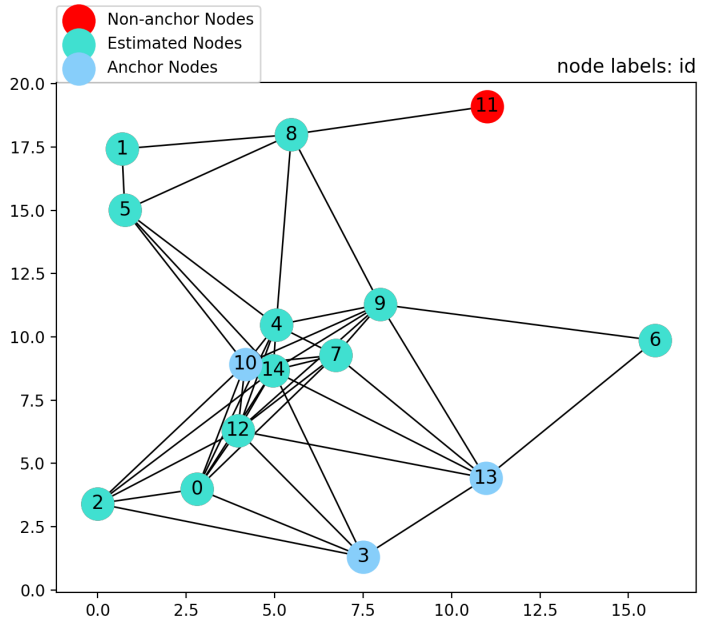

Figure 3. The final result of the algorithm.

4. Distributed Implementation of the Algorithm

In this section, we present the algorithm implemented on each node in the network. The first part of the algorithm is sharing positions (if found) through the network, in a bandwidth efficient manner. The second part is the previously described localization method. We use a semi-flooding method (SFLS) [8] to share known positions, as it takes the advantages of both flooding and rendez-vous based methods, while mitigating their drawbacks. In this method, each node broadcasts its own position (if found) with an identifier, such as a time-stamp or sequence number. A node that receives another node's position with a higher sequence number or time-stamp re-broadcasts once every two receivables. In other words, when a node receives a position for the first time, it rebroadcasts. The next time it receives the new position of the same node, it does not rebroadcast and keeps alternating

the rebroadcast decision for the following received positions. This method is bandwidth-friendly and compatible with our localization algorithm, because nodes need more frequent information about their closer neighbors. Assuming the broadcast interval is t seconds, one-hop neighbors receive the position information every t seconds, two-hop neighbors receive position information every 2 t seconds, three-hop neighbors receive position info every 4 t seconds, and so on. In [8], it has been shown mathematically that the mean number of broadcasts $\overline{m_i}$ performed to update the position information of node i grows approximately linearly with the number of nodes in the network, where nodes are uniformly distributed. This ensures scalability with respect to bandwidth consumption. The broadcasting of positions, receiving them, and initiating the self-localization algorithm is illustrated in Algorithm 1.

Algorithm 1 SFLS Position sharing.

$ownSeqNum$ = 0 (Sequence number starts at 0 for all nodes)
$retransmitDecision$ (stores retransmission decision for all nodes, initialized to true for all nodes)
N (list of received node positions) and N_1 (1-hop neighbors with known positions)
Id_1 : list of IDs of all one-hop neighbors and $distanceTo$ stores distances to 1-hop neighbors
procedure BROADCAST_OWN_POSITION
    if $nodeType == anchor$ **then**
        $ownSeqNum++$
        $pos = anchor\_pos$
    else if position is estimated **then**
        $pos = estimatedPos$
    end if
    packet = [$ownID$, pos, $ownSeq$, $ownID$]
    broadcast packet
end procedure
procedure RECEIVE($sendID, pos, seqNum, ID_{1-hop}$)
    $prevSeqNum = seqNum[sendID]$
    if $seqNum > prevSeqNum$ **then**
        $seqNum[sendID] = seqNum$
        $N[sendID] = pos$
        update $distanceTo[ID_{1-hop}]$
        if $sendID == ID_{1-hop}$ **then**
            $N_1[sendID] = pos$
        end if
        if $nodeType != anchor$ and $seqNum > ownSeqNum$ and $N_1.size >= 2$ **then**
            $selfLocalize()$
        end if
    end if
    $rebroadcast(sendID, pos, seqNum, ownID)$
end procedure
procedure REBROADCAST($sendID, pos, seqNum, ownID$)
    if $retransmitDecision[sendID] == true$ **then**
        packet = [$sendID, pos, seqNum, ownID$]
        broadcast packet
        $retransmitDescision[sendID] = false$
    else
        $retransmitDescision[sendID] = true$
    end if
end procedure

Three procedures are described in Algorithm 1:

(a) Broadcast own position: Startup method is called every t seconds for anchor nodes at the beginning of the simulation. For non-anchor nodes, it is called, once their positions are estimated, to start broadcasting their positions with the interval t. Note that the sequence number (which indicates the freshness of position information) is incremented only by anchor

nodes. This is because the position information originates from the anchor nodes, while the localization of all the other nodes is more or less inherited from them. Thus, when a non-anchor node is localized, it updates its sequence number to be equal to that of the nodes used to localize it.

(b) Receive: Called every time a packet is received. The inputs to the procedure are the ID of the original creator of the packet $sendID$, the position of sending node pos, the associated sequence number $seqNum$, and the ID of the one-hop neighbor that relayed the packet ID_{1-hop}. When new information arrives, this method checks if the gathered information is sufficient to attempt to estimate a node's position. If so, it calls the $selfLocalize$ method, which is described in Algorithm 2.

(c) Rebroadcast: Called by the receive method, after the received position is processed, to decide whether to forward the received packet according to the SFLS rule.

The second part of the algorithm is the localization part, which was explained in the previous section. Thus, each time updated position information is received, an attempt is made to localize a node by calling $selfLocalize()$ in the *receive* method. The localization procedure is further detailed in Algorithm 2.

Algorithm 2 Position Certainty Propagation: $selfLocalize()$.

$mostRecentOneHop$ = list of 1-hop neighbors with the highest sequence number
$maxSeqNumFound$ = seqNum of any of the nodes in the $mostRecentOneHop$
if $mostRecentOneHop$ has 3 or more nodes **then**
    $estimatedPos = intersection(mostRecentOneHop)$
    $ownSeqNum = maxSeqNumFound$
else if $mostRecentOneHop$ has 2 nodes **then**
    [pos1, pos2] = $intersection(mostRecentOneHop)$
    $P1Neighs$ = n **for** each $n \in N$ where $distance(n, pos1) <= CommRange$
    $\overline{P1Neighs} = N - P1Neighs$
    $P2Neighs$ = n **for** each $n \in N$ where $distance(n, pos2) <= CommRange$
    $\overline{P2Neighs} = N - P2Neighs$
    $pos1IsCompatible \iff$ **for** each $n \in P1Neighs, n \in Id_1$ and **for** each $m \in \overline{P1Neighs}, m \notin Id_1$
    $pos2IsCompatible \iff$ **for** each $n \in P2Neighs, n \in Id_1$ and **for** each $m \in \overline{P2Neighs}, m \notin Id_1$
    if $pos1IsCompatible$ and not $pos2IsCompatible$ **then**
        $estimatedPos = pos1$
        $ownSeqNum = maxSeqNumFound$
    else if $pos2IsCompatible$ and not $pos1IsCompatible$ **then**
        $estimatedPos = pos2$
        $ownSeqNum = maxSeqNumFound$
    end if
end if

Let N_1 be the list of detected neighbors with known positions within communication range of the node. When N_1 includes two or three nodes, the position of the node can be estimated using the *intersection* method. If N_1 includes two nodes, the *intersection* method returns the two possible positions, one of which is to be eliminated (if possible). Let N be the list containing position information of all the received positions of network nodes. The first line gets the 1-hop neighbors whose positions are known and have the highest sequence number available. This ensures that the positions used to localize the node are from the same time step, and thus ensures the information is coherent. Then, if the number of 1-hop node positions is three or more, the *intersection* method returns one position that is stored in $estimatedPos$ and the sequence number is updated to be equal to the sequence number of nodes used in localization. If there were two 1-hop nodes with known positions, the *intersection* method returns two possible positions, which are then tested in an attempt to eliminate one of them

using the procedure explained above. Once *estimatedPos* is updated, it will be used together with the sequence number in the next *broadcast_own_position* call.

5. Performance Evaluation

5.1. Comparison with GPS-Free Method

In order to assess our algorithm, we compare it with an existing solution [4], 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 neighbors 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 to 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 [4] for further details of how these steps are executed. Observing the behavior of GPS-free in some graphs, we figured out 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 triangulisation" [4]. We extended this behaviour to nodes that built their local map, but still cannot transfer their local coordinate system to the referent coordinate system; in this case, the node only computes its position in the reference coordinate system.

5.1.1. 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. Similar to our approach, GPS-free is based on triangulations. However, it does not consider the case when a node is within the range of only two known position nodes. The results of this study highlight the added benefit of using such a case. Each simulation run is a connected geometric graph (aka a disc graph), where nodes are randomly positioned in a 100 m $\times$ 100 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 when the node (chosen to have all nodes transferred to its coordinate system) is the one that maximizes the number of estimated nodes. This ensures that the best possible result from the GPS-free method is obtained, without taking into account whether the GPS nodes are among the nodes with the computed positions. The success percentage is the percentage of nodes whose position is computed. Two experiments are conducted, one varying the maximum communication range $[14, 15, 16.., 23]$ m while keeping the average number of nodes constant at 100 nodes. In the second experiment, the communication range is kept constant at 14 m, while the average number of nodes varies $[100, 115, 130, 145, 160]$. The number of nodes in each experiment follows a Poisson distribution with the given average. For each graph configuration, the experiment is repeated 1000 times and the confidence interval is shown as a vertical bar around the point.

5.1.2. Experimental Results

Figure 4 shows a comparison between PCP, GPS-free, and the GPS-free extended version. We start with a maximum communication range of 14 m, and increment it by 1 m for each new configuration. The graph shows 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 has a range of 15 m, which gives an average node degree of 6.3, and so on. 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, consequently so does the average node degree, and it is possible to estimate the positions of more nodes. When the average degree is ≈ 10, our algorithm is able to compute the positions of $\approx 90\%$ of nodes. Also, our algorithm shows a higher success percentage for all configurations than GPS-free and the GPS-free extended version. In another attempt to study the effect of varying the number of nodes while keeping the maximum communication range constant at 14 m, increasing the number of nodes has a similar effect to increasing the maximum communication range. Here, the node density (nodes/m^2) is shown against the success rate.

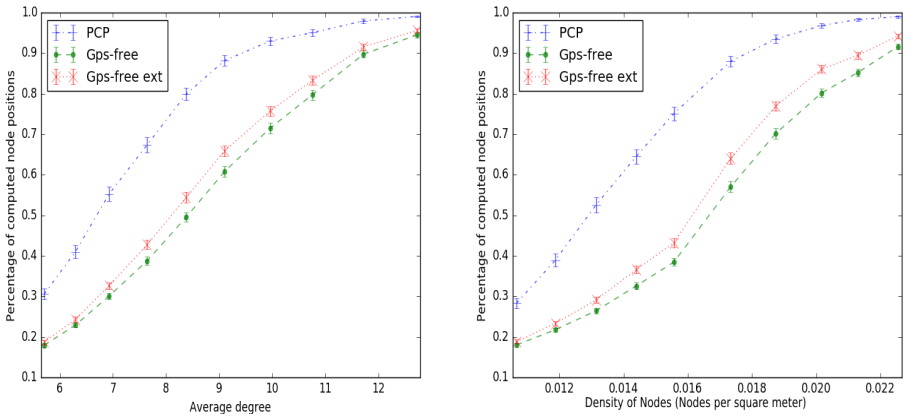

Figure 4. Comparing PCP to the GPS-free and GPS-free extended methods.

5.2. Behavior in Simulated Network

In this section, we study the behavior of our algorithm in a simulated network created using the NS-3 simulator [26]. The network is composed of 49 nodes, spread out on a 7×7 grid. The horizontal and vertical distances between the nodes are 10 m. The anchor nodes are chosen to be in the middle of the networks, as depicted in Figure 5. The maximum range of communication is set to 15 m, using the *RangePropagationLossModel* class. Communication between nodes is through wifi 802 11.b standard, with a data rate equal to 1 Mbps. It is to be pointed out that WiFi is not the only standard to be used for inter-node communication, it is provided here as an example. Generally speaking, any communication technology that allows broadcasting can be used. It is to be noted that performing a broadcast is far easier than point-to-point communication, since the latter might need to store and update routing tables. Thus, this simplicity of just using broadcasts is an advantage of our solution.

The time interval between broadcasts is 1 s. The anchor nodes start broadcasting their own position at slightly different instants, to avoid packet loss due to multiple transmissions in the same instant. Once a non-anchor node estimates its position, it starts its own broadcast cycle with the 1 s interval. This means that after the three anchor nodes start broadcasting their position, the first node to be localized is node 18, which starts broadcasting its own position so that further nodes are localized. If there are no measurement errors, all the nodes are correctly localized within the first second by the successive localization and broadcasting mechanism. We introduce distance measurement errors between nodes by adding Gaussian noise with zero mean and different standard deviation values for each experiment. The average position error for all nodes in the network is computed over a

period of time which includes five broadcasts for all network nodes. The position error for a node is the difference in the distance between the estimated and the actual position. The results are shown in the left-hand graph of Figure 6. Since not all nodes know their position at the same instant, the average localization error is shown against the sequence number. The *i*th sequence number on the x-axis indicates the *i*th anchor node's own position broadcast, which is inherited by non-anchor nodes when localized. Even though the nodes are localized at different time instants, during the first second all the nodes update their position and set their sequence number to one. The experiment is repeated 1000 times at each standard deviation value. The distance measurement error and confidence interval are plotted. For 1 cm standard deviation, the average error fluctuates around 3.5 cm. For 3 cm and 5 cm, the average error is around 11 cm and 18 cm, respectively. The left graph shows the stability of position estimation over a period of time. The right-hand graph shows the average localization error (during one time step) for a wider range of noise values, by varying the standard deviation from 1 cm to 15 cm. The graph shows a linear relation between noise and localization error.

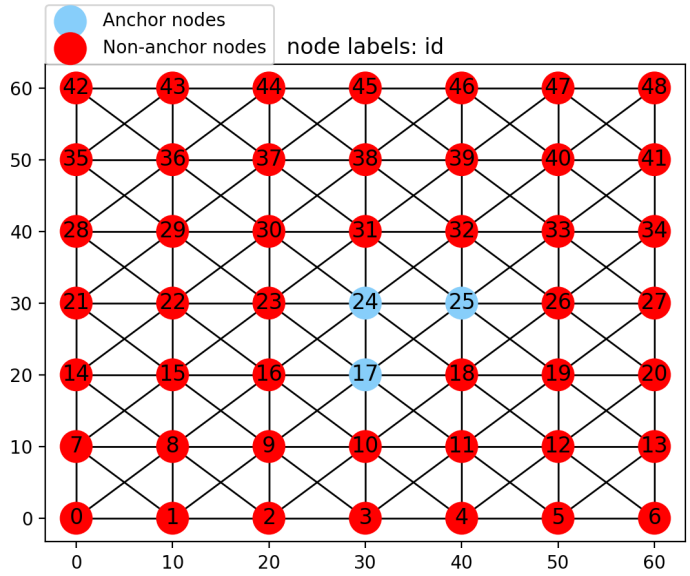

Figure 5. Grid network.

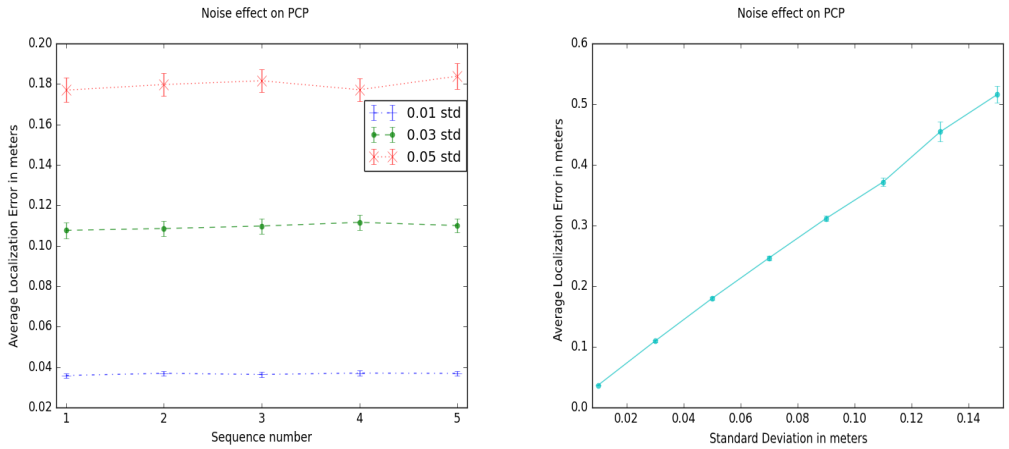

Figure 6. Noise effect during 5 time steps (**left**). Average localization error against noise standard deviation values (**right**).

To study the effect of error propagation as nodes towards the edge of the network are localized, we repeat the experiment for different numbers of nodes in the network. The 3 anchor nodes are always chosen to be in the center of the network, and the standard deviation for the Gaussian noise is set to 5 cm. The results are shown in Figure 7. The first setup is a network with four nodes, three of which are anchor nodes. This means that the fourth node is localized directly through anchor nodes, yielding a very small average localization error. The second setup includes 16 nodes, which means that the anchor nodes are surrounded by one layer of non-anchor nodes colored in green, as shown in the left part of Figure 7. Each subsequent setup adds one extra layer of nodes; a 36-node network has two layers of nodes surrounding the anchor nodes, and so on. It can be seen that, as the layers around the anchor nodes increase, the average localization error increases. This is due to error propagation, which induces larger errors towards the edge of the network.

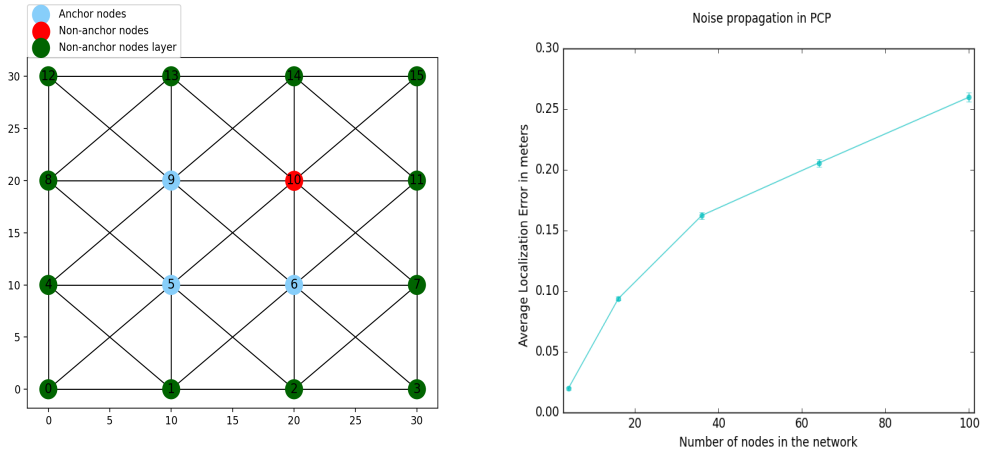

Figure 7. 16-node network example (**left**). Localization error against number of nodes (**right**).

Another aspect to look into is the network traffic consumed in order to maintain position knowledge in the network. As previously mentioned, we use SFLS to share position information. The first broadcast is a traditional flooding, because the first request to rebroadcast is set to true for all nodes. However, the second broadcast is limited to the 1-hop neighbors as they do not rebroadcast received positions. Figure 8 shows the number of broadcasts at each time step, also indicated by the sequence number of the transmitted positions. The number of broadcasts is highest for the first broadcast, as all nodes broadcast the position of all nodes, thus yielding a number of broadcasts equal to n^2 (where n is the number of nodes in the network). The next broadcast wave includes a broadcast to one-hop neighbors only, then the rebroadcasting is cut off, yielding a small number of broadcasts. Nodes alternate allowing and blocking rebroadcasts, such that closer nodes receive higher frequency updates while further nodes are updated less frequently. Using traditional flooding for each position update leads to a number of broad casts $\approx n^2$ for all time steps, as the number of broad casts at sequence number 1 in figure 8. This increase in bandwidth consumption does not yield any improvement to the localization accuracy. Since SFLS does not flood the whole network with position packets for every position update of node i and the position update rate is relatively small compared to channel bandwidth, the collision rate is negligible. Even in the case of losses, since the update frequency is highest for nearby nodes, the position update rebroadcast will reach nearby nodes sooner than further nodes. For further nodes, in the case of node mobility, the relative position change will be less significant compared to nearby nodes. Consequently, the effect of data loss on further nodes is expected to be less significant. If for any reason, the data-loss rate is significantly affecting the solution's accuracy, SFLS can be adjusted to increase the ratio of forwarding received positions. In other words, instead of forwarding once every two received positions (or n times out of p, in general), the ratio n/p can be increased to account for losses.

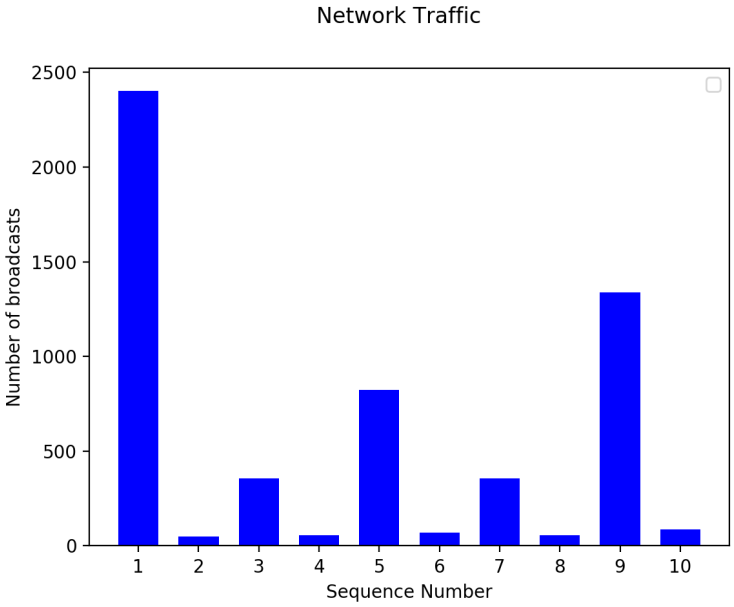

Figure 8. Trace of number of broadcasts per time step.

6. Conclusion

The work presented in this article proposed a location service solution that is lightweight, bandwidth friendly, and cost and energy efficient. Cost and energy consumption is conserved by requiring a minimum of three GPS-equipped nodes. The computations used to localize nodes are not complicated, and do not require many cycles (as in the case of optimization). Sharing positions with other nodes in the network is done using a semi-flooding method, to conserve network bandwidth. The solution requires that the anchor nodes are in the vicinity of at least one non-anchor node, in order to be able to start. This starting condition constraint might appear to hinder the generality of the algorithm, but this can be mitigated by electing some virtual anchor nodes that satisfy the condition. These virtual anchor nodes are given assumed positions, so that they form a triangle from the known distance between them. For instance, let us assume non-collinear nodes i, j, k see each other, and at least a fourth node l. Node i is positioned at the origin, node j is on the horizontal axis at a distance equal to $dist(i, j)$ and node k is added to have a positive y-value using triangluation. The algorithm then treats them as anchor nodes and computes the position of the rest of the nodes, as previously explained. When at least three real anchor node positions have been computed, all the nodes can then be transferred to the global coordinate system, using the computed and actual positions of anchor nodes [4].

The solution was compared to another method [4], and it was shown that our solution achieves a higher percentage of localized nodes. In a network with an average degree $\approx$ 10, around 90% of the nodes are localized. Furthermore, we tested our algorithm using NS-3, while introducing distance measurement errors, and the average error is shown to be stable for a 50-node network. Also, the average localization error increases linearly with the standard deviation of the Gaussian measurement noise. The position sharing technique used conserves bandwidth by reducing position update frequency to further nodes. For future work, the error propagation effect towards the edge of the network needs to be mitigated. Additionally, the impact of node mobility on the accuracy of localization is also to be considered.

Author Contributions: E.R. and P.M. supervised the research work of A.S.; A.S. proposed the idea and implemented the solution; E.R. and P.M. provided guidance and directions to the research methodology, and gave insights as to which results should be obtained; A.S. wrote the article, which was reviewed by E.R. and P.M.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, Z.; Xia, F.; Huang, T.; Bu, F.; Wang, H. A localization method for the Internet of Things. *J. Supercomput.* **2013**, *63*, 657–674.
2. Sallouha, H.; Chiumento, A.; Pollin, S. Localization in long-range ultra narrow band IoT networks using RSSI. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
3. Moore, D.; Leonard, J.; Rus, D.; Teller, S. Robust distributed network localization with noisy range measurements. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004; pp. 50–61.
4. Čapkun, S.; Hamdi, M.; Hubaux, J.P. GPS-free positioning in mobile ad hoc networks. *Cluster Comput.* **2002**, *5*, 157–167.
5. Jung, W.S.; Yim, J.; Ko, Y.B. QGeo: Q-Learning-Based Geographic Ad Hoc Routing Protocol for Unmanned Robotic Networks. *IEEE Commun. Lett.* **2017**, *10*, 2258–2261.
6. Terao, Y.; Phoummavong, P.; Utsu, K.; Ishii, H. A proposal on void zone aware Greedy Forwarding method over MANET. In Proceedings of the Region 10 Conference (TENCON), Singapore, 22–25 November 2016; pp. 1329–1333.
7. Ko, Y.B.; Vaidya, N.H. Location-Aided Routing (LAR) in mobile ad hoc networks. *Wirel. Netw.* **2000**, *6*, 307–321.
8. Renault, E.; Amar, E.; Costantini, H.; Boumerdassi, S. Semi-flooding location service. In Proceedings of the Vehicular Technology Conference Fall (VTC 2010-Fall), Ottawa, ON, Canada, 6–9 September 2010; pp. 1–5.
9. Huang, R.; Záruba, G.V. Monte Carlo localization of wireless sensor networks with a single mobile beacon. *Wirel. Netw.* **2009**, *15*, 978–990.
10. Bulusu, N.; Heidemann, J.; Estrin, D. GPS-less low-cost outdoor localization for very small devices. *IEEE Pers. Commun.* **2000**, *7*, 28–34.
11. Buehrer, R.M.; Wymeersch, H.; Vaghefi, R.M. Collaborative Sensor Network Localization: Algorithms and Practical Issues. *Proc. IEEE* **2018**, *106*, 1089–1114.
12. 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, Los Angeles, CA, USA, 24–27 April 2005; pp. 46–54.
13. Larsson, E.G. Cramer-Rao bound analysis of distributed positioning in sensor networks. *IEEE Signal Process. Lett.* **2004**, *11*, 334–337.
14. Mourad, F.; Snoussi, H.; Abdallah, F.; Richard, C. Guaranteed boxed localization in manets by interval analysis and constraints propagation techniques. In Proceedings of the Global Telecommunications Conference, New Orleans, LA, USA, 30 November–4 December 2008; pp. 1–5.
15. Mourad, F.; Snoussi, H.; Abdallah, F.; Richard, C. Model-free interval-based localization in manets. In Proceedings of the Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, Marco Island, FL, USA, 4–7 January 2009; pp. 474–479.
16. Mourad, F.; Snoussi, H.; Richard, C. Interval-based localization using RSSI comparison in MANETs. *IEEE Trans. Aerosp. Electron. Syst.* **2011**, *47*, 2897–2910.
17. Waltz, D.L. Generating semantic descriptions from drawings of scenes with shadows 1972. Available online: https://dspace.mit.edu/handle/1721.1/6911 (accessed on)
18. Peng, R.; Sichitiu, M.L. Robust, probabilistic, constraint-based localization for wireless sensor networks. In Proceedings of the 2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, CA, USA, 26–29 September 2005; pp. 541–550.
19. Sichitiu, M.L.; Ramadurai, V. Localization of wireless sensor networks with a mobile beacon. *MASS* **2004**, *4*, 174–183.
20. Fruchterman, T.M.; Reingold, E.M. Graph drawing by force-directed placement. *Softw. Pract. Exp.* **1991**, *21*, 1129–1164.
21. Kamada, T.; Kawai, S. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **1989**, *31*, 7–15.

22. Ellson, J.; Gansner, E.; Koutsofios, L.; North, S.C.; Woodhull, G. Graphviz—Open source graph drawing tools. In *International Symposium on Graph Drawing*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 483–484.
23. Eren, T.; Goldenberg, O.K.; Whiteley, W.; Yang, Y.R.; Morse, A.S.; Anderson, B.D.; Belhumeur, P.N. Rigidity, computation, and randomization in network localization. In Proceedings of the Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, 7–11 March 2004; Volume 4, pp. 2673–2684.
24. Sobehy, A.K.; Renault, E.; Muhlethaler, P. Position Certainty Propagation: A location service for MANETs. In Proceedings of the MSPN 2018—4th International Conference on Mobile, Secure and Programmable Networking 2018, Paris, France, 18–20 June 2018.
25. Liu, J.; Zhang, Y.; Zhao, F. Robust distributed node localization with error management. In Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Florence, Italy, 22–25 May 2006; pp. 250–261.
26. Riley, G.F.; Henderson, T.R. The ns-3 network simulator. In *Modeling and Tools for Network Simulation*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–34.
27. Chen, P.C. A non-line-of-sight error mitigation algorithm in location estimation. In Proceedings of the Wireless Communications and Networking Conference, New Orleans, LA, USA, 21–24 September 1999; Volume 1, pp. 316–320.