Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Using smart vehicles for localizing isolated Things

Walid M. Ibrahim[a], Abd-Elhamid M. Taha[b,*], Hossam S. Hassanein[a]

[a] School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada
[b] College of Engineering, Alfaisal University, PO Box 50927, Riyadh 11533, Saudi Arabia

ARTICLE INFO

Article history:
Available online 12 June 2014

Keywords:
Internet of Things
Localization
Kalman filter
Multi-hop wireless localization
Mobile anchors

ABSTRACT

Elementary to the success of the Internet of Things (IoT) is the capability to accurately and efficiently localize its network components, information, and processes. In this paper, we focus on enabling localization of Things that have limited capabilities deployed in isolated areas. Specifically, we explore the scenario where the deployment or the utilization of dedicated anchor nodes becomes costly or practically unfeasible, and where the dependence on multi-hop localization techniques becomes inevitable. We further advocate the use of emerging IoT components such as smart vehicles, capable of self-localization and short-range communication. The proposed scheme thus illustrates the feasibility of a multi-hop wireless localization scheme dependent on mobile anchors (reference points). A key advantage of the proposed scheme is overcoming collinear trajectory (flip-ambiguity) problem, which arises whenever the smart vehicle moves in a straight trajectory. A Kalman Filter (KF) is used to decrease the location error introduced from the multi-hopping during the localization process. Through simulation, we show that the use of our localization scheme with KF reduces errors by 31% compared to localization using anchors from a single direction and 16% compared to a weighted means approach. Moreover, our scheme with KF consistently outperforms the typical range-based DV-Distance scheme with fixed anchors.

1. Introduction

Rapid evolutions in wireless communication and electronic technologies have substantially decreased the cost and size of embedded devices with sensing, processing and communication capabilities. Such devices have made ubiquitous monitoring and tracking applications cost-effective by enabling the collection of data from hundreds of different locations in large-scale scenarios [1]. These advancements have facilitated a new vision where information from millions or even billions of devices can be collected, processed and exploited collaboratively within a global Internet of Things (IoT) [2].

In IoT, "Things" communicate and collaborate with each other. Energy consumption, storage management, heterogeneity of devices and communication bandwidth are major challenges facing this emerging paradigm. As well, Things have to be locatable and addressable in order to be trackable and accessible in application domains such as geographic routing, marketing, data aggregation algorithms and environmental monitoring applications [3]. However, many Things cannot autonomously identify their position, and may require multiple anchors to estimate their location. Given the important role assumed by Sensor Nodes (SNs) in IoT, our focus in this paper is on location of SNs with limited capabilities.

Many Wireless Sensor Networks (WSNs) applications involve a random deployment of SNs in isolated areas with challenging terrains and no central access roads, e.g., dense rain forest or rocky mountainous areas. Due to the limited transmission range of WSNs, SNs collect information about the environment and send the collected information to the sink node at the edge of the topology using multi-hop communication. Fig. 1 shows an example of isolated SNs. To localize SNs isolated from the network edge, a multi-hop localization scheme is needed. As the processing of the position information propagates to the isolated nodes, error accumulates, decreasing the estimation accuracy as the number of hops increases [4].

Localizing SNs using multi-hop schemes involves deploying anchor nodes that broadcast their location information with operation instructions to the SNs. In turn, SNs would utilize this information to estimate their own positions. Such schemes

* Corresponding author.
E-mail addresses: walid@cs.queensu.ca (W.M. Ibrahim), ataha@alfaisal.edu (A.-E.M. Taha), hossam@cs.queensu.ca (H.S. Hassanein).

http://dx.doi.org/10.1016/j.comcom.2014.05.013
0140-3664/© 2014 Elsevier B.V. All rights reserved.

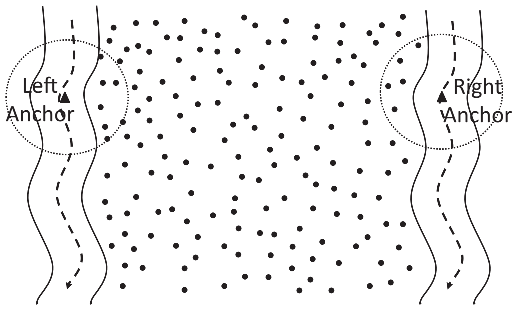


Fig. 1. WSN with anchor nodes from multiple directions.

commonly relied on high-density deployment of costly anchor nodes to ensure the availability of sufficient reference points for all SNs. Such assumptions become problematic in the context of IoT, where densities of SNs or Things are expected to be higher, more ad hoc, and spread over wider areas, and where the use of dedicated “anchoring” becomes, eventually, both costly and ineffective.

While IoT emerges with its unique challenges, it also brings forth unique opportunities. A relevant example can be readily seen in the ubiquity of today’s smartphones that possess a collective capability of communication, processing, storage, recording (audio, image and video), and localization (GPS and assisted GPS). However, a more pronounced manifestation of an IoT opportunistic resource are smart vehicles that interact not only with navigation and broadcast satellites, but also with passenger smartphones, roadside components, and other vehicles on the road.

In this work we abstract on the emergence of these smart vehicles, specifically by using them as mobile anchors. When smart vehicles move in straight trajectories, the flip ambiguity problem results [5]. Traditionally, the term “flip ambiguity” labels the confusion resulting from collinear anchor nodes. As illustrated in Fig. 2, anchor nodes a, b, and c are collinear. Node n estimates its position through measurements d_a, d_b, and d_c. Each measurement defines a ranging circle centered at the anchor node. Due to measurement errors, the three measured circles do not intersect at a common point, which causes ambiguities in determining whether the position of the SN is n or n' [6].

We propose a new and robust localization scheme that uses smart vehicles to localize isolated SNs. In the proposed scheme, the SNs estimate their positions from multiple directions, which decrease the effect of the error propagation. After this process, a Kalman Filter (KF) is used to decreases the localization error coming from the longer hop direction, based on the information

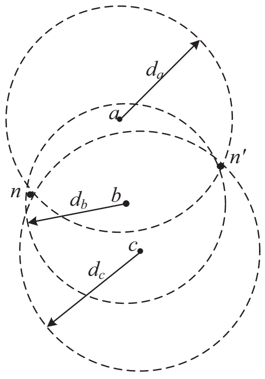


Fig. 2. Collinear anchor nodes a, b and c causing a flip ambiguity for SN n.

coming from the shorter hop direction. Simulation results show that using information from two different directions significantly increases localization accuracy.

The **contributions** presented in this work are listed as follows:

1. Illustrating the use of smart vehicles as mobile anchors (reference points) for localization.
2. Facilitating wireless multi-hop localization based on location information received from multiple directions.
3. Overcoming collinearity to account for smart vehicles moving in straight trajectories.
4. Employing Kalman filtering to reduce localization error in multi-hop wireless localization scheme which utilize mobile anchors (reference points).

The remainder of this paper is organized as follows. In Section 2, we present the motivation behind our work through reviewing related effort in the literature and establishing the addressed void. Section 3 offers a concise definition of the problem addressed, and details the proposed solution. Section 4 details the simulation environment used for validation and analysis, along with results and discussions. Finally, conclusions are made in Section 5, along with an elaboration on possible future directions.

2. Related work and motivation

The specific tool with which a thing can be localized depends on its capabilities, in addition to the type of nodes or localization services available in its direct context. Readers interested in surveys on various localization schemes should refer to references [7,8]. Our interest in this work is in localizing Things that are isolated from direct access to location information, i.e., have no direct access or interaction with a self-localizing entity, or a location broadcasting network element. To this end, the use of mobile anchors and multi-hop localization becomes inevitable if the Thing is to label its communication with accurate location information. The independent application of these localization techniques have been extensively discussed within the context of localizing sensors in WSNs. In motivating our work, we review the state of the art in three relevant aspects: (1) multi-hop localization techniques, (2) addressing the problem of flip ambiguity, and (3) localization using mobile anchors.

2.1. Multi-hop localization

Multi-hop localization schemes are based on either distance-based or connectivity-based strategies. In connectivity-based strategies the SNs obtain the absolute measurements of node distances using Receive Signal Strength Indicator (RSSI), Time of Arrival (ToA), or Time Difference of Arrival (TDoA) [4,9,10], while in distance-based strategies the SNs use the connectivity information to estimate the location of SNs based on the position of the anchor nodes [4,11–13].

Niculescu and Nath propose two localization schemes, one based on distance measurement, the other is based on connectivity information [4]. The authors’ distance-based scheme is called Distance Vector (DV)-distance, and has the anchor node sending beacon messages to all its immediate neighbors. Immediate (first-hop) neighbors to the anchor node estimate the distance to the anchor by using signal strength measurement. These neighboring nodes then forward the beacon message to the second-hop neighbors to infer the distance to the anchor, and so on until the network is completely covered in a controlled flooding manner. Once an unknown node has three or more distances estimated to different anchor nodes, it computes its position using multi-lateration.

The second scheme proposed in [4] is the DV-hop, which operates in three stages. First, the algorithm computes the number of hops for all the SNs to the anchor nodes. Next, the anchor node gets the number of hops required to reach the other anchor nodes, calculating the average length for one hop by dividing the total distance by the number of hops. SNs then estimate the distance by multiplying the number of hops by the average length for one hop.

Stoleru et al. propose a scheme called MDS-MAP that uses multidimensional scaling (MDS) to determine SN locations by using only connectivity information [9]. The operation of MDS-MAP consists of three steps: (1) Finding the shortest paths for all pairs; (2) applying classical MDS to the distance matrix; and (3) using three or more anchor nodes to transform the relative map to positions based on the positions of anchor nodes.

Wu et al. propose a self-configurable positioning technique for multi-hop wireless networks [10]. A number of nodes at each corner of the network serve together as an anchor for estimating the distances by a Euclidean distance estimation model. The authors use ToA to estimate the distance for each hop. Once ToA information is received by an SN, the sum of these distances is computed by minimizing an error objective function.

The above solutions work well in isotropic networks, i.e., networks where the hop count between two nodes is proportional to their geometric distance. The schemes, however, exhibit a dramatic decrease in performance when used in anisotropic networks, i.e., in networks with non-uniform node distribution where there is a concave region at its center. Fig. 3 shows the difference between isotropic and anisotropic networks. Li and Wang mined the characteristic of anisotropic WSN when anchor SNs send non-uniform beacon messages [14]. They use the mined network connectivity characteristics to make appropriate adjustments on measured distance between nodes based on the directions of message and degrees of inflections. Their simulation results show that their method outperforms DV-distance especially in anisotropic networks. Xiao et al. solve the problem of anisotropic network by defining 3 different patterns based on number of hops and line-of-sight rule [15]. The three patterns are (1) Concentric Ring (CR) pattern, in which the SN is within few hops from the anchor node, in this case SN is treated as it is in isotropic anchor. (2) Centrifugal Gradient (CG) pattern, in which the SN is far from the anchor node, in this case SN is treated as anisotropic where they use a proposed solution named DiffTriangle that tolerate the inaccurate HopSize estimates. (3) Distorted Gradient (DG) pattern which is considered the worst case, in which the line-of-sight rule is breach by an object between them, in this case the message is dropped. A SN estimates its location using weighted MMSE multi-lateration after it collects sufficient distance estimates from different anchors using CR or CG patterns only. They show that using analytical analysis and simulation that their solution for anisotropic gives higher accuracy compared to previous localization solutions that claims to tolerate network anisotropy.

For connectivity based multi-hop localization Savarese et al. [11] propose AHLoS (Ad-Hoc Localization System) algorithm, where a small fraction of nodes have the knowledge of their position to estimate the location of other SNs using collaborative and iterative multi-lateration algorithm. In AHLoS at least three SNs know their position in order to estimate the position of other nodes. Nagpal et al. [12] calculate a global coordinate system for the whole network by estimating the Euclidian distance of each hop between SNs. The SNs use the number of communication hops to estimate how far they are from anchor nodes. When an SN receives at least three different positions from different anchor nodes, the SN combines the distance from the anchor nodes and estimates its position based on the hop count to each anchor. Akbas et al. [13] localize the position of SNs flooding in the Amazon river based on stationary anchor nodes placed at the bank of the river. Their localization algorithm uses multi-hop between SNs and anchor nodes. Each SN keeps a single weight value for each anchor it is associated with. The saved weight represents how far the SNs are to each anchor node. The anchor node uses these weights to estimate the SNs position.

2.2. Flip ambiguity

The problem of flip ambiguity is approached from different perspectives in the literature. The work done by Eren et al. and Goldenberg et al. test the unique localization conditions and construct localizable networks using rigidity theory [16,17]. The authors show that maintaining a global rigidity in the localized networks decreases the collinearity of anchor nodes. However, it is hard to maintain the global rigidity of the network unless it is compensated by a priori information from the network [8].

Localization algorithms in [6,18] identify possible flip ambiguities caused by collinearity of anchor nodes and decrease the effect of flip ambiguity during the localization processes. Moore et al. propose a robust quadrilaterals localization scheme to identify possible flip ambiguities in fully connected sensor quadruples [6]. The scheme has two steps. In the first step, the distance measurement between two anchor nodes S_A and S_B is used to estimate the two possible locations of the un-localized SN S_D. In the second step, a third anchor SN S_C is used to decide which of the

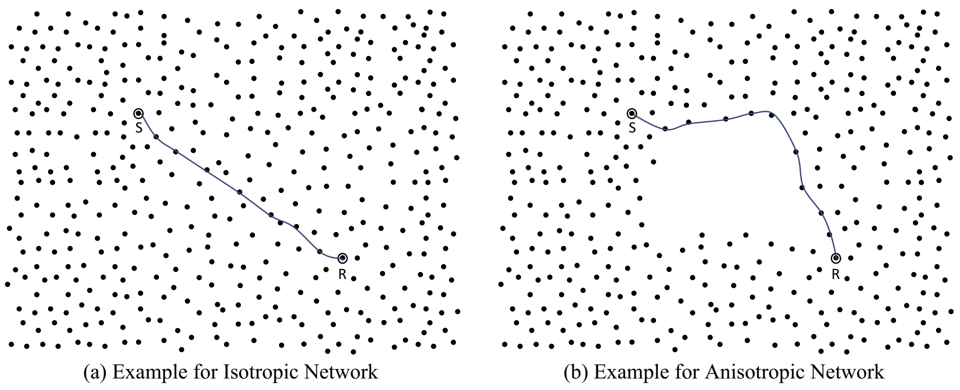

S
R
(a) Example for Isotropic Network
S
R
(b) Example for Anisotropic Network

Fig. 3. (a) The shortest path between source and destination is close to a straight line. (b) The shortest path between source and destination is curved caused by the hole between them.

two possible locations for the un-localized SN satisfy the distance constraint. If both locations satisfy the condition, the scheme will ignore this SN. In [18], it was noted by Sittile that if sensors S_A and S_C are used in the first step in [6] instead of sensors S_A and S_B, and sensor S_B is used in step 2 instead of sensor S_C, this may result in a different value for the robustness criterion, which would affect the overall localization performance. Such dependency is eliminated by including all three permutations when localizing S_D, i.e., (S_A, S_B, S_C), (S_A, S_C, S_B) and (S_B, S_C, S_A). This inclusion, however, increases the computational complexity of the algorithm.

To reduce the error caused by trilateration, Yang et al. [19] propose a sequential localization scheme that estimates SNs location and controls the errors introduced in each step. In their sequential scheme, a set of anchor nodes is chosen and the expected error is tracked in each step to minimize the error. However, flip ambiguity cannot be avoided by error control alone as it can be triggered even by the smallest errors if the anchor nodes used to localize the SN are collinear. Basu et al. solved the problem of collinearity by using both distance and angle measurements [20], where the localization problem is transferred to a convex form and solved using linear programming. However, the scheme by Basu et al. cannot work if either the distance or angle measurement does not have a clear boundary. Moreover, the scheme depends on the knowledge of both distance and angle measurements, which requires additional hardware. To identify and reduce the error caused by flip ambiguities, Kannan et al. introduce a scheme that recognizes SNs with possible flips using simulated annealing, and offer a refined scheme through the use of a ranging model and a bounder check, despite the refinement, however, the scheme may not identify all flips [21].

2.3. Localization using mobile anchors

Pathirana et al. use mobile anchor nodes that move in random paths to localize SNs in a delay-tolerant sensor network [22]. Han et al. show that localization using mobile anchor nodes that move randomly results in poor performance in terms of localization time and accuracy [5]. Another work [23] uses a mobile anchor node that adopts Gauss–Markov motion model, and uses the weighted centroid algorithm to localize the position of SNs. They use a genetic algorithm to decrease the estimated error.

To overcome the poor performance of random movement for mobile anchor nodes, Koutsonikolas et al. propose a pre-determined path for a single mobile anchor node [24]. They also address the collinearity problem when a single anchor node is used. Different fixed trajectory types such as Circle, S-Curves, Rectangle, Spiral and Triangle are proposed for a single mobile anchor node to overcome the collinearity problem [5,25,26]. Predetermined paths are efficient if the deployment area has a regular shape (i.e., square or rectangle) and the density of sensors is uniform, but can lead to wasteful anchor movement in irregular areas and non-uniform sensors' density. Wang et al. propose a scheme that handles non-uniform placement scenarios [27]. In Wang et al. scheme, the mobile anchor node sends a start message all over the network and when an SN receives the start message it adds the neighbor SNs surrounding it and then the SN forwards the message. When the anchor node receives the start message back, it calculates the shortest path to localize all SNs. The anchor node moves in half-circle movements to avoid the collinearity property for the mobile anchor node.

2.4. Motivation for our work

From the above, it can be seen that existing multi-hop localization approaches have only exploited the use of stationary anchor nodes. Existing schemes also either avoid or try to refine the result of flipped SNs. When mobile anchors are utilized, localization is only applied to SNs within a single-hop range from the anchor node. Collectively, these drawbacks eliminate the possibility of localizing isolated Things or nodes.

The objective of this work is to demonstrate the possibility of localizing isolated Things using mobile anchors. We realize this possibility through combining the use of multi-hop localization and mobile anchors. As mobile anchors may travel in straight paths, the designed scheme will need to accommodate both collinear and non-collinear movements. In presenting our solution, we first illustrate its overall operation with mobile anchors moving in pre-defined paths that deliberately avoid collinear measurements. We then introduce a scheme that relaxes this assumption, allowing for any trajectory for mobile anchors.

3. Robust multi-hop localization scheme

The robust multi-hop localization scheme using multiple directions is described in detail in this section.[1] The two main goals for this approach are: (1) to enhance the position estimation of localized SN without deploying anchor nodes in the sensing area as the cost of anchor nodes is much higher than normal SNs and (2) to propose a solution that overcomes the collinearity problem that appears from using a mobile vehicle that moves in straight lines. To simplify the description of our scheme, and without loss of generality, we review the operation of the scheme using only two mobile anchors, each sending position messages from a different direction.

To overcome flip ambiguity, we propose a new localization scheme that estimates the distance between two nodes using RSSI measurements. SNs then estimate their position using the estimated distance and laws of trigonometry [29]. In the following, we first formulate the localization problem. The proposed scheme is then described. Next, the Kalman Filter is used to reduce the localization errors introduced in the localization process.

3.1. Problem formulation

We consider a two-dimensional WSN localization problem, where there are two roads at both ends of the sensing area as shown in Fig. 1. Assume that there are M SNs that are deployed randomly in the sensing area, where the SNs need to localize their positions. The position of ith SN is denoted by $\mathbf{x}_i = [x_i \quad y_i]^T$. The distance measured between the ith and jth SN is

$$d_{i,j} = d_{j,i} = r_{i,j} + \varepsilon_{i,j} \qquad \forall i,j = 1,2,\ldots,M \tag{1}$$

where $r_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|$ is the noise-free distance between SN i and j, and $\varepsilon_{i,j} \sim \mathcal{N}(0, \sigma^2_{i,j})$ represents the uncorrelated noise. The $\sigma^2_{i,j}$ is assumed to be accurately estimated and is known a priori [30]. Let α^l_i and $\alpha^r_i, \forall i = 1,2,\ldots,n$, respectively be the positions where the left and right mobile anchor nodes broadcast their positions while they are moving on the edges of the sensing area. The mobile anchor sends its position in the position packet that is sent to localize the SNs. Each SN localizes its position twice from the left and right sides and saves the number of hops to the left and right edge. The estimated positions of ith SN from the left and right side that are p and q hops away from the left and right anchor nodes are represented by $\tilde{\mathbf{x}}^{l,p}_i$ and $\tilde{\mathbf{x}}^{r,q}_i$, respectively. For example, $\tilde{\mathbf{x}}^{l,3}_k$ means SN k received a packet that is 3 hops away from the left edge.

[1] A simplified version of this scheme where mobile anchors know their direction of movement was introduced in [28].

Algorithm 1. Processing the position message at Node *k*.

```
Function CheckPosMsg(recSeqNum, recHopNum, srcIP, $\mathbf{x}_i$, $d_{i_k}$)
    switch unknownPosMap, leftPosMap and rightPosMap do
        case unknownPosMap, leftPosMap, rightPosMap are empty
            Add $\mathbf{x}_i$ to unknownPosMap;
            savedUnknwonHopNum ← recHopNum;
            if recHopNum == 1 then
                unknownNodeState ← borderNode;
            else unknownNodeState ← normalNode ;
        case leftPosMap and rightPosMap are empty
            $y_{avg}$ ← GetYAverage(unknownPosMap);
            if $|y_{avg} - y_i| < \varepsilon$ then
                if recHopsNum > savedUnknownHopsNum then
                    return (Message coming from longer route);
                Add $\mathbf{x}_i$ and $d_{i_k}$ to unknownPosMap;
            else
                if $y_i < y_{avg}$ then
                    Add $\mathbf{x}_i$ and $d_{i_k}$ to leftPosMap;
                    Move unknownPosMap to rightPosMap;
                    rightNodeState ← unknownNodeState;
                    if recHopNum == 1 then
                        leftNodeState ← borderNode;
                    else leftNodeState ← normalNode ;
                else
                    Move unknownPosMap to leftPosMap;
                    Add $\mathbf{x}_i$ and $d_{i_k}$ to rightPosMap;
                    leftNodeState ← unknownNodeState;
                    if recHopNum == 1 then rightNodeState ← border
                    else rightNodeState ← normalNode ;
            Forward the received message;
        case leftPosMap and rightPosMap are not empty
            $y_{avg}$ ← GetYAverage(leftPosMap);
            if $|y_{avg} - y_i| < \varepsilon$ then
                if recHopsNum > savedLeftHopsNum then
                    return (Message coming from longer route);
                Add $\mathbf{x}_i$ to leftPosMap;
            else
                if recHopNum > savedRightHopsNum then
                    return (Message coming from longer route);
                Add $\mathbf{x}_i$ and $d_{i_k}$ to rightPosMap;
            if leftNodeState is borderNode then
                EstimateNodeKPos(leftPosMap,left);
            if rightNodeState is borderNode then
                EstimateNodeKPos(rightPosMap,right);
            else Forward this message ;
```

Algorithm 2. Message direction is known. The message is coming from the left direction.

```
Function CheckPosMsgDirKnown (seqNum, hopNum, srcIP, $\mathbf{x}_i$,$d_{i_k}$
    if msgDir == left then
        if hopNum > savedLeftHopsNum then
            return (Message coming from longer route);
        savedLeftHopsNum ← seqNum for srcIP;
        Add $\mathbf{x}_i$ and $d_{i_k}$ to the leftPosMap;
        if size of leftPosMap > 2 then
            EstimateNodeKPos(leftPosMap,left);
    else if msgDir == right then
        if hopNum > savedRightHopsNum then
            return (Message coming from longer route);
        savedRightHopsNum ← seqNum for srcIP;
        Add $\mathbf{x}_i$ and $d_{i_k}$ to the rightPositionMap;
        if size of rightPosMap > 2 then
            EstimateNodeKPos(rightPosMap,left);
```

3.2. Processing the position message

Each SN estimates its position $\tilde{\mathbf{x}}_i$ using position location coming from left ($\tilde{\mathbf{x}}_i^{l,p}$) and right ($\tilde{\mathbf{x}}_i^{r,q}$) directions. The SNs, therefore, need not know the direction of the message to estimate position. Each SN requires a minimum of two SNs, with a known position from each direction, in order to estimate its position from one direction. This localization scheme contains two phases of position messages. The first position message phase is when the direction of message is unknown, while the second position the message phase is when the direction of the message is known and the position of the SNs at the border is estimated.

The the first phase position message has three different maps: *unknownPosMap*, *leftPosMap* and *rightPosMap*. The *unknownPosMap* saves anchor SN positions along with the distance between the anchor node and itself when the direction of the message is unknown at the beginning. The *leftPosMap* and *rightPosMap* are used when the SN has enough information that enables the SN to identify whether the message is coming from the right or left direction. This localization scheme has three cases, as shown in Algorithm 1, to process the position message: case 1 is used when the three Maps are empty; case 2 when the leftPosMap and rightPosMap are empty; and case 3 when leftPosMap and rightPosMap contains data.

In **case 1**, when the three maps are empty, this means that this is the first position message received by the SN. The position of the anchor and the estimated distance is saved in unknownPosMap. After that, the SN checks the number of hops, if equal to 1, then the SN declares itself to be a border SN otherwise it is a normal SN.

For **case 2**, when leftPosMap and rightPosMap are empty, it means the direction of the message is not identified yet. Thus the SN has to identify whether the received message is coming from the same direction or from the other direction. This process is done as follows. First, the SN calculates the average of y in unknownPos-Map. It then compares the y_{avg} with the received y_i. If the difference between them is smaller for a given threshold, it means that the change in y is very small, and the message is coming from the same direction as the previous messages. In this case, the SN verifies that the message is not coming from a longer route and then adds the received anchor SN position to the unknownPosMap. However, if the difference between y_{avg} and received y_i is greater than the given threshold, then this means the message is coming from the other direction. If the received y_i is less than the saved average y_{avg}, then the received message is coming from the left direction. Thus the position of the anchor node is saved in leftPosMap and unknownPosMap is copied to rightPosMap and vice versa if the received y_i is greater than the saved average y_{avg}. Finally, the SN forwards the position message.

For **case 3**, when leftPosMap and rightPosMap are not empty, it means that the direction of the message can be determined. To identify the direction of the received message, the SN estimates the average y of one of the saved Maps (in our case, we chose average of leftPosMap). If the difference between y_{avg}^{left} and received y_i is less than a given threshold, this means the message is received from the left direction, otherwise it means it is coming from the right direction. If the message is coming from the left direction, the SN checks that the message is not coming from a longer route. After that, the anchor node position is added to the leftPosMap and vice versa if it is coming from the right direction. Then the SN checks its status, if it is a normal SN then it will forward the received position message. But if it is a border SN, it estimates its position then forwards its position to the surrounding SNs.

When an SN in the middle receives a second phase position message, it processes the second phase position message as follows. The SN checks the number of hops of the received message. If the received number of hops is larger than the saved number of hops, the SN discards the message as it is coming via a longer route. Otherwise, the SN saves the received number of hops and this hop number represents how far the SN is from the edge in the direction of the received message. The SN then saves the position of the SN that sends the position message. Algorithm 2 shows the main procedure to check the message with known direction.

3.3. Estimating the node position

After an SN receives two or more location packets that have the same number of hops, it estimates the three distances $d_{i,j}, d_{i,k}, d_{j,k}$ for each pair as shown in Fig. 4, where the positions of $\mathbf{x}_i$ and $\mathbf{x}_j$ are previously known (i.e., two different locations for two mobile anchor nodes or normal SNs that have estimated their position in a previous step) and $\mathbf{x}_k$ is the location of SN k that needs to estimate its position.

In order to estimate $\mathbf{x}_k$, we need to estimate the coordinates of point $\mathbf{x}_l$ representing the intersection between $d_{i,j}$ and the height h of triangle $d_{i,j}, d_{i,k}, d_{j,k}$. The coordinates of $\mathbf{x}_l$ are calculated as follows:

$$\begin{bmatrix} x_l \\ y_l \end{bmatrix} = \begin{cases} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{l}{d_{i,j}} \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} & \text{for } \widehat{D}_{j,k} \leqslant 90 \\ \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{l}{d_{i,j}} \begin{bmatrix} x_j - x_i \\ -(y_j - y_i) \end{bmatrix} & \text{otherwise} \end{cases}, \tag{2}$$

where l is calculated using laws of sines, cosines and tangents.

$$l = \sqrt{h^2 + d_{i,k}^2 - \left(2 \times h \times d_{i,k} \times \cos\left(\widehat{L}\right)\right)}. \tag{3}$$

In order to calculate l we need to calculate h and angle $\widehat{L}$. h is given by

$$h = \frac{2 \times d_{i,k} \times d_{i,j} \times \sin\left(\widehat{D}_{j,k}\right)}{d_{i,j}}. \tag{4}$$

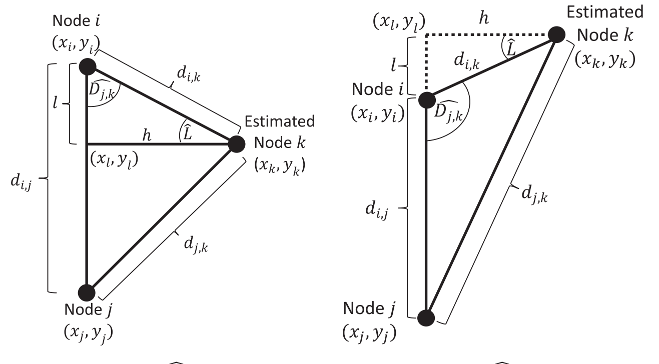
where the angle $\widehat{D}_{j,k}$ is calculated using:

$$\widehat{D}_{j,k} = \cos^{-1}\left(\frac{d_{i,j}^2 + d_{j,k}^2 - d_{i,k}^2}{2 \times d_{i,j} \times d_{j,k}}\right), \tag{5}$$

and the angle $\widehat{L}$ as

$$\widehat{L} = \begin{cases} 90 - \widehat{D}_{j,k} & \text{for } \widehat{D}_{j,k} \leqslant 90 \\ \widehat{D}_{j,k} - 90 & \text{otherwise} \end{cases}. \tag{6}$$

After estimating the coordinates of $\mathbf{x}_l$, we get the slope between SN i and j to calculate $\mathbf{x}_k$ to consider the shift in x and y coordinates caused by the slope of the line $m_{d_{i,j}} = \tan^{-1}\frac{y_j - y_i}{x_j - x_i}$. This allows us to estimate the position of the SN using collinear and non-collinear anchor nodes.



(a) When $\widehat{D}_{j,k} < 90$ (b) When $\widehat{D}_{j,k} > 90$

Fig. 4. Estimating node k using the estimated distance.

SN k estimates its position based on the direction of the message. Thus, if the message is coming from the left direction, SN K estimates $\tilde{\mathbf{x}}_k^{l,p}$ by

$$\tilde{\mathbf{x}}_k^{l,p} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_l \\ y_l \end{bmatrix} + h \begin{bmatrix} \sin(m_{d_{i,j}}) \\ -\cos(m_{d_{i,j}}) \end{bmatrix}. \tag{7}$$

Otherwise, if the message is coming from the right direction, SN k estimates $\tilde{\mathbf{x}}_k^{r,q}$ by

$$\tilde{\mathbf{x}}_k^{r,q} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_l \\ y_l \end{bmatrix} + h \begin{bmatrix} -\sin(m_{d_{i,j}}) \\ \cos(m_{d_{i,j}}) \end{bmatrix}, \tag{8}$$

where $\tilde{\mathbf{x}}_k^{l,p}$ is the estimated position from the left direction that is p hops away from the left edge and $\tilde{\mathbf{x}}_k^{r,q}$ is the estimated position from the right direction that is q hops away from the right edge. Algorithm 3 shows the procedure for estimating the SN's position from the left and right directions.

After SN k estimates its direction from both directions, the SN can use the mean to estimate its position. However, the estimated position from the direction with the larger number of hops contains more errors than the direction with smaller hops number (i.e., if $q < p$, then $\tilde{\mathbf{x}}_k^{r,q}$ is more accurate than $\tilde{\mathbf{x}}_k^{l,p}$). Using the mean the SN does not take into consideration the error propagated for each hop. Thus, the weighted mean can be used to consider the propagation error for each hop. The weighted mean estimation is calculated as follows:

$$\tilde{\mathbf{x}}_k = \frac{(\tilde{\mathbf{x}}_k^{l,p} \times q) + (\tilde{\mathbf{x}}_k^{r,q} \times p)}{p + q}. \tag{9}$$

Algorithm 3. Estimate the position of Node k

```
Function EstimateNodeKPos (posMap, msgDir)
    counter ← 0 foreach $\mathbf{x}_i$ and $d_{i,k}$ in posMap do
        foreach $\mathbf{x}_j$ and $d_{j,k}$ in posMap do
            if $\mathbf{x}_i == \mathbf{x}_j$ then Continue;
            counter + +;
            $d_{i,j} \leftarrow \sqrt{(x_j - x_i)2 + (y_j - y_i)2}$;
            Calculate $l = \sqrt{h^2 + d_{i,k}^2 - (2 \times h \times d_{i,k} \times \cos(\widehat{L}))}$;
            if $\widehat{D}_{j,k} \leq 90$ then
                $\widehat{L} \leftarrow 90 - \widehat{D}_{j,k}$;
                $x_l \leftarrow x_i + \frac{l}{d_{i,j}} \times (x_j - x_i)$ ;
                $y_l \leftarrow y_i + \frac{l}{d_{i,j}} \times (y_j - y_i)$;
            else
                $\widehat{L} \leftarrow \widehat{D}_{j,k} - 90$;
                $x_l \leftarrow x_i + \frac{l}{d_{i,j}} \times (x_j - x_i)$ ;
                $y_l \leftarrow y_i + \frac{l}{d_{i,j}} \times -(y_j - y_i)$;
            Calculate $m_{d_{i,j}} = \tan^{-1}\frac{y_j - y_i}{x_j - x_i}$;
            if msgDir == left then
                $p \leftarrow$ savedLeftHopNum;
                estimate $\tilde{\mathbf{x}}^{l,p}$[counter] using eq 7;
            else if msgDir == right then
                $q \leftarrow$ savedRightHopNum ;
                estimate $\tilde{\mathbf{x}}^{r,q}$[counter] using eq 8;
    if msgDir == left then
        $\tilde{\mathbf{x}}_k^{r,p} \leftarrow$ GetAverage($\tilde{\mathbf{x}}^{l,p}$)
    else if msgDir == right then
        $\tilde{\mathbf{x}}_k^{r,q} \leftarrow$ GetYAverage($\tilde{\mathbf{x}}^{r,q}$)
```

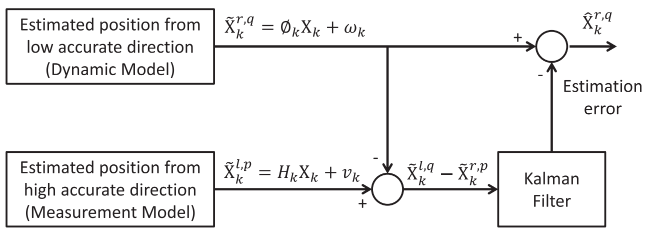

Estimated position from low accurate direction (Dynamic Model)
$\tilde{X}_k^{r,q} = \phi_k X_k + \omega_k$
$\hat{X}_k^{r,q}$
Estimation error
Estimated position from high accurate direction (Measurement Model)
$\tilde{X}_k^{l,p} = H_k X_k + v_k$
$\tilde{X}_k^{l,q} - \tilde{X}_k^{r,p}$
Kalman Filter

Fig. 5. The Kalman filter left/right integration for $p < q$ [31].

However, the weighted mean does not take into consideration the error gained from each hop, which motivates our use of Kalman filtering.

3.4. Location enhancement using Kalman filter

We propose to use KF in place of the weighted mean. KF is an optimal estimation tool that enhances one measurement given a more accurate measurement from another source using a sequential recursive algorithm [31]. We use KF that corrects the estimated location of the side that has the larger number of hops using the information provided from the side that has the smaller number of hops. This helps to estimate the error resulting from the larger number of hops. Fig. 5 shows the KF block diagram used in this study.

In order to complete the development of the state-space of the discrete time KF equations, the system dynamic and measurement models for the SN have to be defined. The system's dynamic and measurement model equations when $p < q$ (the values of $\tilde{\mathbf{x}}_k^{l,p}$ and $\tilde{\mathbf{x}}_k^{r,q}$ are switched if $q < p$) are represented as follows, respectively:

$$\mathbf{x}_k^{r,q} = \phi_k \mathbf{x}_k + \omega_k^q \tag{10}$$

$$z_k = \mathbf{x}_k^{l,p} = H_k \mathbf{x}_k + v_k^p \tag{11}$$

where $\mathbf{x}_k$ is the actual location of the SN, ϕ_k is a static transmission matrix that relates $\mathbf{x}_k$ with its previous state. Since there is no change in the SN state, i.e., location, the $\phi_{k,k-1}$ matrix is represented as an identity matrix, $Q_k^p = E[\omega_k^p\,(\omega_k^p)^T]$ and $R_k^q = E[v_k^q\,(v_k^q)^T]$ are the covariance matrices for the p and q hop count coming from the left and right directions. Q_k and R_k are assumed to be uncorrelated as they are received from two different directions with different numbers of hops.

Cho et al. calculate Q_k and R_k for single hop as $\frac{R_3}{2} \times I$, where R is the normal distribution of error placement for a single hop [32]. In order to calculate Q_k and R_k for multi-hops, we expanded their proof to calculate Q_k and R_k for multiple hops. Q_k^p and R_k^q are calculated in our work as follows:

$$E[\omega_k^p\,(\omega_k^p)^T] = \begin{bmatrix} \frac{\sum_{i=1}^{p}(\sigma_i^2)^3}{2} & 0 \\ 0 & \frac{\sum_{i=1}^{p}(\sigma_i^2)^3}{2} \end{bmatrix} \tag{12}$$

$$E[v_k^q\,(v_k^q)^T] = \begin{bmatrix} \frac{\sum_{i=1}^{q}(\sigma_i^2)^3}{2} & 0 \\ 0 & \frac{\sum_{i=1}^{q}(\sigma_i^2)^3}{2} \end{bmatrix} \tag{13}$$

where $\sum_{i=1}^{p}\sigma_i^2$, is the summation of the uncorrelated noise in Eq. (1) from hop 1 to hop p and similar for $\sum_{i=1}^{q}\sigma_i^2$.

The KF equations used in this study are summarized in Table 1. The steps using KF are as follows if $p < q$ ($\tilde{\mathbf{x}}_k^{l,p}$ and $\tilde{\mathbf{x}}_k^{r,q}$ are switched if $q < p$). First, the covariance matrix is initialized at the left border SN using Eq. (1). After that, the SN calculates the priori covariance and Kalman gain matrices using Eqs. (1) and (1). Then, the right position $\tilde{\mathbf{x}}_k^r$ is updated to $\hat{\mathbf{x}}_k^r$ using Eq. (1). Later, the SN calculates the posteriori covariance matrix using Eq. (1) and forwards its value to the next hop SNs. The SNs that are away from the edge of the network do the same steps except they use the received posteriori covariance matrix instead of creating a new one. Finally the SNs estimate their new position using the following equation:

Table 1
A summary of Kalman filter equations for $p < q$.

Kalman filter algorithm	
Covariance matrix initialization:	
$P_0 = E\langle(\mathbf{x} - \tilde{\mathbf{x}}_0)(\mathbf{x} - \tilde{\mathbf{x}}_0)^T\rangle$	(14)
State estimate extrapolation:	
$\hat{\mathbf{x}}_k^r(-) = \phi_k \hat{\mathbf{x}}_{k-1}^r(+)$	(15)
A priori covariance matrix:	
$P_k(-) = \phi_k P_k(+) \phi_k^T + Q_{k-1}$	(16)
Kalman gain matrix:	
$K_k = P_k(-)H_k^T(H_k P_k(-) H_k^T + R_k)^{-1}$	(17)
Update the estimated location:	
$\hat{\mathbf{x}}_k^r(+) = \tilde{\mathbf{x}}_k^r(-) + K_k(\tilde{\mathbf{x}}_k^l - H_k \tilde{\mathbf{x}}_k^r(-))$	(18)
A posteriori covariance matrix:	
$P_k(+) = (I - K_k H_k) P_k(-)$	(19)

$$\tilde{\mathbf{x}}_k = \frac{\tilde{\mathbf{x}}_k^{l,p} + \hat{\mathbf{x}}_k^{r,q}}{2} \tag{20}$$

4. Performance evaluation

In this section we evaluate the performance of the proposed scheme in three different scenarios. The first scenario compares the error between using fixed anchor nodes against using mobile anchor nodes. To do so, we first calculate the number of fixed anchors required to cover a road with a given length, then perform the comparison between using fixed and mobile anchors. In the second scenario we investigate the accuracy of the localization estimation as the number of hops increases. Finally in the third scenario we compare the effect of increasing the number of hops by increasing the width of the simulation area.

The metric utilized in the first scenario is the localization mean error after estimating the position using KF as in Eq. (20). In the second and third scenarios, we compare four different estimation techniques for our localization scheme: (1) using one direction that has fewer number of hops; (2) using the mean of both sides; (3) using the weighted mean of both sides using Eq. (9); and (4) using KF using Eq. (20) against DV-Distance localization scheme [4]. The anchor nodes used for DV-Distance are fixed on the edge of the simulated area.

Our simulations are made in ns-3 [33]. The communication range of anchor and SN is set to 30 m. The range measurement noise $\varepsilon_{i,j}$ is a zero-mean white Gaussian process with variance $\sigma_{i,j}^2 = d^2/\text{SNR}$, where SNR is the signal-to-noise ratio received by the SN [30]. All results are averages of ten different independent runs with distinct random seeds. Table 2 summaries the experimental parameters we used in our simulation model.

4.1. Minimum number of static anchor nodes

Before we compare the result between using mobile against static anchor nodes, we need to identify the minimum number of static anchors on each side of the road that are required to replace the mobile anchor. To facilitate the illustration, we assume that the

Table 2
Summary of the experimental parameters.

Parameter	Value
Number of SNs	200
Transmission range	30 m
Measurement noise $\varepsilon_{i,j}$	$\sigma^2_{i,j} = d^2/\text{SNR}$
Sensing area width (variable)	100–400 m
Sensing area length (variable)	100–400 m

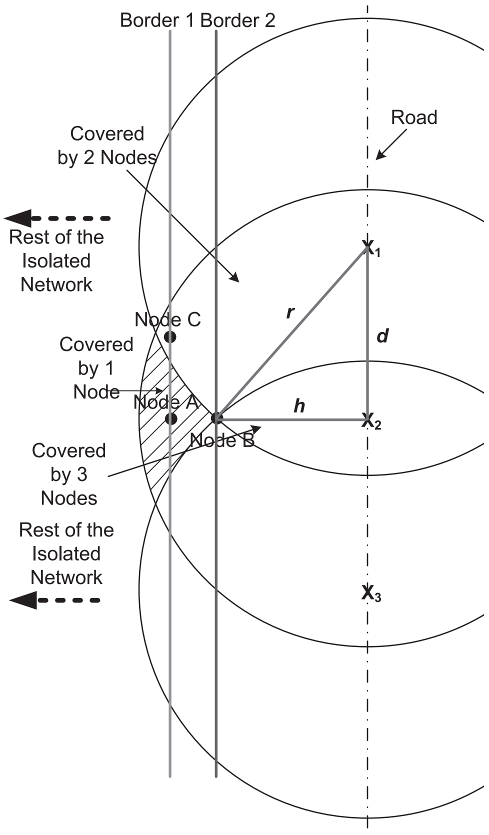


Fig. 6. Determining the minimum number of fixed anchor nodes required.

static anchor nodes are placed in a straight line, and that transmission ranges are fixed and are not effected by signal distortion (i.e., have a perfectly circular shape).

The scenario considered in Fig. 6 where SN A and C are almost on the same border line. However SN A is only covered by 1 anchor node "x_2", while SN C is covered by 2 anchor nodes "x_1 and x_2". This means that not all SNs on border line 1 are guaranteed to be covered by 2 anchors nodes. On the other hand, SN B by comparison is covered by 3 anchors nodes, which means that SNs on border 2 are guaranteed to be covered by at least 2 anchors. This line is defined by the point of intersection between the two circles x_1 and x_3, which is the position of SN B. Thus, to measure the distance between the anchor nodes "d", we have to know how far the border line of the isolated SNs is from the line where the anchor nodes are located. The distance between the line of anchor nodes and the border SN is represented by the symbol h.

The distance between two anchor nodes can be calculated using triangle x_1, x_2 and SN$_B$. Since we have two known sides, which are how far the SNs are from the anchor nodes represented by h and the transmission range for the anchor node represented by r, we can get the length of the third side using Pythagoras theorem formula. Thus the distance between two anchor nodes can be calculated as

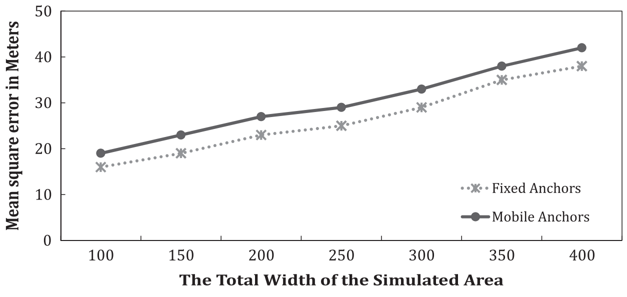


Fig. 7. Comparison between static and mobile anchor nodes.

$$d = \sqrt{r^2 - h^2} \tag{21}$$

Therefore, the number of SNs required to cover each side is given by the following equation:

$$\text{Number of anchor nodes} = \frac{l}{\sqrt{r^2 - h^2}} + 1 \tag{22}$$

where l is the length of the road. Thus the number of SNs is directly proportional with the height of the simulated area, and inversely proportional with the transmission range and how far the SNs are from the border.

4.2. Static vs. mobile anchors

After we identified the minimum number of static nodes that are required to cover each side of the road in the previous subsection, in this subsection we compare the performance of static and mobile anchor nodes. 200 SNs are deployed randomly in a simulated area with a width of 100 m and the length of the road is changed from 100 m to 400 m in 50 m increments. For the static nodes approach, SNs are fixed in their position and the distance between fixed anchor nodes is calculated based on Section 4.1. The position of the static SN is assumed to arrive accurately at the SNs, while a fixed error is introduced in the mobile anchors location broadcasts. The error is equal to 10% of the distance between the road and the sensor network.

Fig. 7 shows the average location error for fixed and mobile anchor nodes, these results were obtained after using the KF. Fig. 7 shows that the accuracy of using fixed anchors is almost the same as when using mobile anchors. However, static anchors give a little higher location accuracy than mobile anchors. Although there is a position error introduced to the position of the mobile anchor, the difference between the accuracy of using a mobile anchor with inaccurate position is less than 0.5 m, which is much lower than the error introduced to the mobile anchors. This is because the KF enhances the estimated localization.

4.3. Localization error per number of hops

In this scenario, we examine the localization error for each hop as the number of hops of the shortest side increases in the same simulation area. We randomly deploy 200 SNs in a simulation area with dimension of $400\,\text{m} \times 100\,\text{m}$, since we are interested in studying the effect of number of hops on our localization accuracy, which is affected by the width of the simulated area. Thus we increase the width of the simulated area to be 4 times its length. The maximum number of hops from one end to the other using the above dimension is 17. For DV-Distance, the number of fixed nodes is calculated using Eq. (6) in Section 4.1.

Fig. 8 illustrates that using mean estimation for our techniques gives a similar trend as using DV-Distance, as the localization accuracy is worse at the edges and improves in the middle of the simulation area. This shows that DV-Distance scheme works simi-

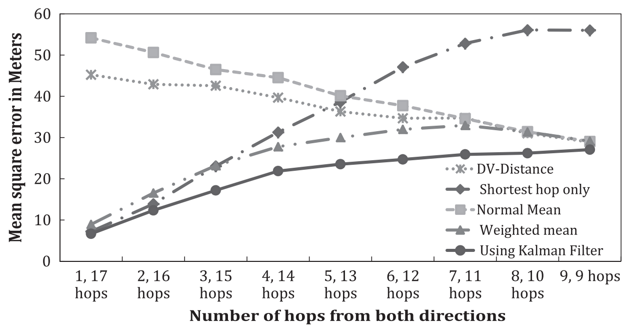


Fig. 8. Relation between localization error and number of hops.

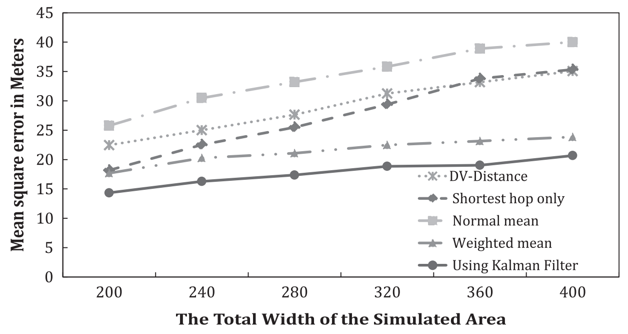


Fig. 9. Relation between localization error and the width of the simulated area.

lar to the mean estimation i.e., give similar weight to the longer and shorter hops. For all other estimation techniques as the numbers of hops increases the localization error increases. Fig. 8 shows that using KF gives the least estimation error, while the mean estimation gives the highest estimation error. The mean estimation gives the worst results when the difference between the number of hops is larger as the error from the direction that has a larger number of hops is huge, which affects the overall estimation accuracy when we take the mean. However by taking the weighted mean, we give a lower weight for the estimation from the direction that has a larger number of hops. The improvement of KF over the weighted mean is between 19% and 13% with an overall mean of 15.6%. Estimating the position using shortest hop only, weighted mean and KF gives a very high accuracy when the difference between the two directions is the maximum (i.e., near the edge of the simulation area). However, the estimation error for shortest hop only is higher than weighted mean and KF for SNs that are four hops away from the edge of the network and reaches the maximum in the middle of the network performance is worse than the KF by 51.7%. This is because the shortest hop only does not benefit from the information coming from the other direction. Moreover, using our scheme with KF is better than using DV-Distance scheme by 28% on average.

4.4. Localization error per width change

In this scenario, we compare the overall localization error as we increase the number of hops by increasing the simulation area. We randomly deployed 200 SNs in a simulation area with a length of 100 m and the width of the simulation area is changed from 200 m to 400 m in 40 m increments.

Fig. 9 shows that using the mean gives the worst localization accuracy, while using the KF gives the best accuracy. KF gives better localization accuracy than weighted mean by 15.6% on average and better than a single side by 31% on average. The DV-Distance localization techniques gives a better accuracy than using the mean, however its localization performance is worse than Shortest distance only, weighted mean and using KF. The reason that the KF gives a better result than the weighted mean is the KF estimates and assigns the weights automatically. Moreover, KF takes into consideration the propagation error per hop, while the weights in the weighted mean are fixed and the propagation errors per hop are not taken into consideration.

5. Discussion

In IoT, decreasing energy and resource consumption is on of the key components to enable Things to function for a longer time. The number of messages transmitted between Things has a direct impact on energy consumption and bandwidth consumed. Thus by decreasing the number of messages, we decrease the energy and resources used during the localization process. In this paper we propose a new scheme that consists of two phases. The first phase, which localizes the position of SNs consumes the same amount of energy as other localization schemes. While the second phase, which is used to enhance the localization accuracy using KF, is an optional phase that can be used to enhance the localization accuracy at the expense of increasing the message transmitted between Things. In order to decrease the transmitted message a similar technique used in our previous work [34] can be used. In the scheme in [34], a SN receiving message, would store it for a pre-defined interval, as opposed forwarding it directly. The SN then filters redundant information and forwards the most accurate message. Using such technique significantly reduces wasted resources.

Estimating the distance between a pair of SNs is the main component of localizing SNs. RSSI and Time-based measurement techniques are the most common ranging techniques used in WSN localization. Both techniques are prone to noise causing the estimated distance to be imprecise [35]. Time-based techniques are relatively immune to most sources of noise including signal attenuation, refraction and reflection as Time-based techniques rely on the signal speed. The main source of errors are the absence of LoS between SNs and the processing time of the packets. On the other hand, RSSI techniques are sensitive to channel noise, interference and reflections as they estimate the distance using the strength of the received radio frequency signal. RSSI techniques use either RSSI profiling measurements or estimate the distance via analytical models by mapping the RSSI to distance using path-loss propagation models. Analytical mapping models are attractive since they do not require additional hardware and can provide reasonable localization accuracy [8,36].

6. Conclusion

Our intent in this work is to demonstrate the viability of localizing isolated Things through the use of a combination of multi-hop wireless localization and mobile anchors (or mobile reference points). We advocate the high feasibility of this proposal in IoT through the emergence of smart vehicles. A novel multi-hop localization scheme was introduced for Sensor Nodes (SNs) with limited capabilities, with the scheme capable of utilizing location information from multiple mobile anchor nodes. The scheme operates in two stages. In the first stage, the scheme estimates the location for each SN from multiple directions using the estimated distance between SNs and the flow direction of the message. In the second stage, we apply Kalman filtering to improve localization accuracy. Simulation results illustrated our scheme's superiority over other mechanisms that rely on location information from a single direction. As well, and unlike existing schemes, our proposal was shown to accurately perform position estimation regardless of mobile anchor collinearity.

With the above mentioned viability demonstrated, a fundamental concern in IoT remains to be addressed, and that is the energy consumption in localization schemes. In previous work [34], we investigated the general limitations inhibiting scalability in localizing Things using state-of-the-art multi-hop wireless localization techniques. There we highlighted the impact of minor modifications in behaviors for forwarding location information on the aggregate network signaling and, thereby, on consumed energy. Such findings can be readily applied to the work presented herein, where error reduction was optimized by the use of Kalman filtering. These issues motivate a deeper understanding of the trade-offs between localization accuracy, localization delay, signaling overhead, and energy consumption. Indeed, our future work seeks this very understanding.

Acknowledgements

This work was made possible by NPRP Grant #NPRP4-553-2-210 from the Qatar National Research Fund (a member of The Qatar Foundation). The work was also supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The statements made herein are solely the responsibility of the authors.

References

[1] O. Vermesan, P. Friess, Internet of Things: Global Technological and Societal Trends, River Publishers, 2011.
[2] R.H. Weber, R. Weber, Internet of Things, Legal Perspectives, first ed., Springer, 2010.
[3] Z. Chen, F. Xia, T. Huang, F. Bu, H. Wang, A localization method for the Internet of Things, J. Supercomput. 63 (3) (2013) 657–674.
[4] D. Niculescu, B. Nath, DV based positioning in ad hoc networks, Telecommun. Syst. 22 (2003) 267–280.
[5] G. Han, H. Xu, J. Jiang, L. Shu, T. Hara, S. Nishio, Path planning using a mobile anchor node based on trilateration in wireless sensor networks, Wirel. Commun. Mob. Comput. 13 (14) (2013) 1324–1336.
[6] D. Moore, J. Leonard, D. Rus, S. Teller, Robust distributed network localization with noisy range measurements, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys, ACM, 2004, pp. 50–61.
[7] J. Wang, R. Ghosh, S. Das, A survey on sensor localization, J. Control Theory Appl. 8 (2010) 2–11.
[8] G. Mao, B. Fidan, B.D. Anderson, Wireless sensor network localization techniques, Comput. Netw. 51 (10) (2007) 2529–2553.
[9] R. Stoleru, T. He, J. Stankovic, Range-free localization, in: R. Poovendran, S. Roy, C. Wang (Eds.), Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks, Advances in Information Security, vol. 30, Springer, US, 2007, pp. 3–31.
[10] H. Wu, C. Wang, N.-F. Tzeng, Novel self-configurable positioning technique for multihop wireless networks, IEEE/ACM T. Netw. 13 (3) (2005) 609–621.
[11] C. Savarese, J. Rabaey, J. Beutel, Location in distributed ad-hoc wireless sensor networks, in: International Conference on Acoustics, Speech, and Signal Processing, ICASSP, vol. 4, 2001, pp. 2037–2040.
[12] R. Nagpal, H. Shrobe, J. Bachrach, Organizing a global coordinate system from local information on an ad hoc sensor network, in: Proceedings of the 2nd International Conference on Information Processing in Sensor Networks, IPSN, vol. 2634, Springer-Verlag, 2003, p. 553.
[13] M. Akbas, R. Matthias, D. Turgut, Local positioning for environmental monitoring in wireless sensor and actor networks, in: IEEE 35th Conference on Local Computer Networks, LCN, 2010, pp. 806–813.
[14] K. Li, Y. Wang, Distance estimation by mining characteristics in anisotropic sensor networks, Clust. Comput. 13 (2) (2010) 167–180.
[15] Q. Xiao, B. Xiao, J. Cao, J. Wang, Multihop range-free localization in anisotropic wireless sensor networks: a pattern-driven scheme, IEEE Trans. Mob. Comput. 9 (11) (2010) 1592–1607.
[16] T. Eren, O. Goldenberg, W. Whiteley, Y. Yang, A. Morse, B. Anderson, P. Belhumeur, Rigidity, computation, and randomization in network localization, in: 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, vol. 4, 2004, pp. 2673–2684.
[17] D.K. Goldenberg, A. Krishnamurthy, W.C. Maness, Y. Richard, Y.A. Young, A.S. Morse, A. Savvides, B.D.O. Anderson, Network localization in partially localizable networks, in: IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, vol. 1, 2005, pp. 313–326.
[18] F. Sottile, M. Spirito, Robust localization for wireless sensor networks, in: Proceedings IEEE 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON, 2008, pp. 46–54.
[19] Z. Yang, Y. Liu, Quality of trilateration: confidence-based iterative localization, IEEE Trans. Paral. Distrib. Syst. 21 (5) (2010) 631–640.
[20] A. Basu, J. Gao, J.S. Mitchell, G. Sabhnani, Distributed localization using noisy distance and angle information, in: 7th International Symposium on Mobile ad hoc Networking and Computing, MobiHoc, 2006, pp. 262–273.
[21] A. Kannan, G. Mao, B. Vucetic, Simulated annealing based wireless sensor network localization with flip ambiguity mitigation, in: IEEE 63rd Vehicular Technology Conference, VTC, vol. 2, 2006, pp. 1022–1026.
[22] P.N. Pathirana, N. Bulusu, A.V. Savkin, S. Jha, Node localization using mobile robots in delay-tolerant sensor networks, IEEE Trans. Mob. Comput. 4 (3) (2005) 285–296.
[23] J. Huanxiang, W. Yong, T. Xiaoling, Localization algorithm for mobile anchor node based on genetic algorithm in wireless sensor network, in: International Conference on Intelligent Computing and Integrated Systems, ICISS, 2010, pp. 40–44.
[24] D. Koutsonikolas, S.M. Das, Y.C. Hu, Path planning of mobile landmarks for localization in wireless sensor networks, Comp. Commun. 30 (13) (2007) 2577–2592.
[25] D.-K. Kim, Y.-S. Kim, J.-W. Chong, Hybrid RSS/TOA wireless positioning with a mobile anchor in wireless sensor networks, in: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, ICUIMC, ACM, New York, NY, USA, 2012, pp. 116:1–116:8.
[26] R. Huang, G. Zaruba, Static path planning for mobile beacons to localize sensor networks, in: 5th International Conference on Pervasive Computing and Communications Workshops, PerCom, 2007, pp. 323–330.
[27] H. Wang, W. Qi, K. Wang, P. Liu, L. Wei, Y. Zhu, Mobile-assisted localization by stitching in wireless sensor networks, in: IEEE International Conference on Communications, ICC, 2011, pp. 1–5.
[28] W.M. Ibrahim, A.M. Taha, H.S. Hassanein, Robust wireless multihop localization using mobile anchors, in: IEEE International Conference on Communications, ICC, 2013, pp. 1506–1511.
[29] M.D. Hestenes, R. Hill, Algebra and Trigonometry, second ed., Prentice Hall Professional Technical Reference, 1986.
[30] H.-W. Wei, Q. Wan, Z.-X. Chen, S.-F. Ye, A novel weighted multidimensional scaling analysis for time-of-arrival-based mobile location, IEEE Trans. Signal Process. 56 (7) (2008) 3018–3022.
[31] M. Grewal, A. Andrews, Kalman Filtering: Theory and Practice Using MATLAB.
[32] H. Cho, J. Lee, Y. Lee, S.W. Kim, Localization for a mobile node based on chrip spread spectrum ranging, in: International Conference on Computational Science and Its Applications, ICCSA, 2010, pp. 211–216.
[33] The Network Simulator ns-3 <http://www.nsnam.org/>.
[34] W. Ibrahim, A.-E.M. Taha, H. Hassanein, Scalability issues in localizing Things, in: IEEE 36th Conference on Local Computer Networks, LCN, 2011, pp. 673–680.
[35] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, D. Culler, The effects of ranging noise on multihop localization: an empirical study, in: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, IPSN, IEEE Press, 2005, pp. 73–80.
[36] A. Koubaa, M.B. Jamaa, Taxonomy of fundamental concepts of localization in cyber-physical and sensor networks, Wirel. Pers. Commun. 72 (1) (2013) 461–507.