Hindawi
Mobile Information Systems
Volume 2017, Article ID 4525380, 16 pages
https://doi.org/10.1155/2017/4525380

Research Article

LEA: An Algorithm to Estimate the Level of Location Exposure in Infrastructure-Based Wireless Networks

Francisco Garcia,[1] Javier Gomez,[1] Miguel Lopez-Guerrero,[2] Victor Rangel,[1] and Michael Pascoe-Chalke[2]

[1] *Department of Telecommunications Engineering, National Autonomous University of Mexico, 04510 Mexico City, Mexico*
[2] *Department of Electrical Engineering, Metropolitan Autonomous University, Iztapalapa, 09340 Mexico City, Mexico*

Correspondence should be addressed to Francisco Garcia; lgarciaj@uxmcc2.iimas.unam.mx

Received 14 August 2016; Revised 20 December 2016; Accepted 4 January 2017; Published 26 February 2017

Academic Editor: Carlos T. Calafate

Location privacy in wireless networks is nowadays a major concern. This is due to the fact that the mere fact of transmitting may allow a network to pinpoint a mobile node. We consider that a first step to protect a mobile node in this situation is to provide it with the means to quantify how accurately a network establishes its position. To achieve this end, we introduce the location-exposure algorithm (LEA), which runs on the mobile terminal only and whose operation consists of two steps. In the first step, LEA discovers the positions of nearby network nodes and uses this information to emulate how they estimate the position of the mobile node. In the second step, it quantifies the level of exposure by computing the distance between the position estimated in the first step and its true position. We refer to these steps as a *location-exposure* problem. We tested our proposal with simulations and testbed experiments. These results show the ability of LEA to reproduce the location of the mobile node, as seen by the network, and to quantify the level of exposure. This knowledge can help the mobile user decide which actions should be performed before transmitting.

1. Introduction

With the proliferation of positioning devices, for example, GPS, location-based services (LBSs) are now available in a variety of mobile devices at all times. This availability facilitates the development of many localization and guiding applications as well as allowing mobile nodes to better interact with their surroundings. For instance, LBSs typically take into account the user's geographic position in order to retrieve valuable information such as the location of the nearest gas station or directions that must be taken to reach a destination. However, these services raise serious location-privacy concerns, since the position of the mobile user becomes available to the service providers. In order to minimize such risks, several studies have focused on protecting the location of the users when they send queries to LBSs (e.g., [1]). In many of these solutions, the mobile node simply provides a false location that is far away from its true position while still receiving meaningful information from LBSs [2, 3]. However, even if a mobile node does not send its position to a service on purpose, as long as it wirelessly transmits packets, it unwillingly exposes its location to nearby network nodes. For instance, several localization methods make use of three or more overhearing network nodes in order to estimate the location of a transmitting wireless source [4]. Furthermore, recently developed localization methods can provide location estimates with fewer than three network nodes [5, 6]. We consider that the ability of a network to localize a transmitting source raises serious location-privacy concerns for mobile users. For example, the network could gather information about the location of users as time passes in order to analyze their behaviors. In [7], how the network could use this knowledge to predict future positions of a mobile node is explained. Furthermore, relating habits and mobility patterns may allow someone to figure out the user's identity [8]. We consider that a first step to incorporate location-privacy guarantees is to provide the mobile user with the means to quantify how accurately the network is able to localize the mobile terminal (i.e., to quantify the level of exposure). To achieve this end, we introduce LEA (*location-exposure*

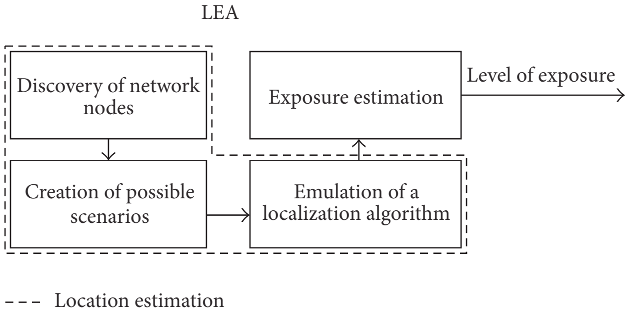

LEA
Discovery of network nodes
Exposure estimation
Level of exposure
Creation of possible scenarios
Emulation of a localization algorithm
Location estimation

Figure 1: LEA operation.

algorithm), a solution that allows a mobile node to quantify its level of exposure when traversing a network comprised of static wireless nodes. This is accomplished in real time with no intervention from the network. In this way, the mobile user can take appropriate actions before transmitting.

LEA operation is divided into two steps. In the first step, a node running LEA overhears data transmissions from network nodes in its vicinity and estimates how far away they are located. This is achieved from methods based on measurements of signal attenuation [9]. This procedure of distance estimation is performed in at least two different locations in order to estimate possible locations for each network node. Then LEA uses these discovered positions to emulate how the network pinpoints the position of the mobile terminal. We refer to this step as *location estimation*. Then, in the second step, LEA quantifies the level of exposure by computing the difference between the location estimated in the first step and the true location of the mobile node (which is acquired from other means, such as GPS). We use this difference as an indication of the level of exposure based on the rationale that the higher the error in the location estimated by the network, the lower the level of exposure, and vice versa. Note that this step can only be performed by the mobile terminal, since the network does not know the actual position of the mobile node. This feature constitutes an advantage of the mobile node over the network. We refer to this step as *exposure estimation* and the set of problems addressed by the two steps is what we collectively refer to as the *location-exposure* problem. The operation of LEA is illustrated in Figure 1, where the location estimation step is comprised of three components which perform the following tasks: discovery of network nodes, creation of possible scenarios, and emulation of a localization algorithm. Then, in the exposure estimation step, LEA generates as its output an estimate of the level of exposure.

LEA operation is not tied to the emulation of a particular localization method. However, the selected algorithm to be emulated will have an effect on the computation of the level of location exposure. At this point, the following two considerations are in order. First, some discrepancies may be introduced if LEA uses a localization method which is different from the one actually being used by the network. Second, there are some localization methods regarded as highly effective whose operating details are not known enough (e.g., Google maps). In both cases, it is very likely that the user will not have access to the missing information. Due to these reasons, we consider as a sensible approach to use the best available localization algorithms whose operating details are known. At this point, it is worth mentioning that there are many localization methods that we could have used in our tests. However, most of them assume that network nodes are uniformly deployed in such a way that there are always k nodes covering each region, where k is typically greater than or equal to three [10–12]. However, we found this limitation to be too restrictive, as it is not always possible to achieve such coverage. In fact, even in dense wireless networks, there might be regions with fewer network nodes than in others. Moreover, factors such as obstacles, irregular areas, power outages, and channel propagation impairments affect the network coverage from region to region. Due to all these factors, we decided to make use of localization methods with minimum coverage requirements to work. In this work, we considered two state-of-the-art localization algorithms [5, 6] as case studies for the emulation of a localization algorithm. Both are capable of pinpointing mobile nodes in sparse wireless networks (i.e., situations where fewer than three fixed network nodes detect the mobile node).

We implemented LEA in a custom computer simulator and also performed testbed experiments using IEEE 802.11 hardware under diverse network conditions. This paper reports our experiments and corresponding results.

In the future, we envision LEA as the basis of various applications trying to increase or decrease the level of exposure of mobile users depending on their particular situation. For example, in the case of an emergency situation, a mobile node might want to increase its level of exposure by moving to a region where the localization error estimated by the network is minimized. LEA also provides information about holes in network coverage, thus allowing the mobile terminal to avoid moving to areas with little or no coverage. Since LEA discovers the location of network nodes, this information can be used, for instance, to get closer to other nodes in order to achieve a better connection. On the other hand, a mobile node might want to reduce its level of exposure in untrusted networks. The algorithms to modify such level are still to emerge and are out of the scope of this paper. However, we envision that modifying its level of exposure might require the mobile node to change its transmission patterns, its trajectory, or even its identity.

The rest of this paper is organized as follows. Section 2 overviews related work in areas of localization, exposure, coverage, and privacy in wireless networks. Section 3 details the main components of LEA, while Section 4 explains its operation in dense and sparse networks. Section 5 presents performance tests under diverse network conditions as well as in real testbed experiments. Finally, Section 6 concludes this paper.

2. Related Work

The problem of establishing the location of mobile or fixed nodes has been given significant attention for several years now. In contrast, there is a more recent interest in studying how to provide location-privacy guarantees to mobile users under diverse circumstances. In this paper, we use the term location privacy to denote a feature consisting in letting a mobile node decide which entities know its location and which ones do not.

By far, the most studied scenario dealing with location privacy is the one where a mobile node sends its current position to a LBS in order to retrieve location-dependent information. Such interaction raises serious privacy concerns, since the mobile node discloses its current location to the service provider. By collecting location data from a mobile terminal over time, it is possible to retrace its recent movements and estimate its future locations. For instance, in [7], the authors used selected traces left behind by a mobile node and a Gaussian regression process in order to retrace its trajectory. Another concern related to collecting user location information is that it could be used for other purposes such as to infer habits or mobility patterns. In [13], for instance, the authors studied some location-privacy concerns related to untrusted LBSs. Furthermore, the same authors proposed a method to minimize the risk associated with the disclosure of mobile node trajectories by selecting which locations are sent to a LBS.

Other research works have addressed a problem somehow related to location privacy, that is, how to determine the number and location of network nodes required to observe a certain region (also known as the coverage problem). For instance, in [14], the authors proposed an algorithm in which the uncertainty about the position of each network node is modeled as concentric circles in order to determine the minimum sensing radius that guarantees k-coverage. Other authors have focused on minimizing the number of network nodes required to observe a region by reducing the overlapping regions among their coverage zones. In this context, a result worth mentioning is that the hexagon grid lattice is a single-coverage optimal grid [15]. However, small displacement of one network node may create holes, resulting in blind regions within the wireless network. A study of a hexagonal grid lattice is proposed in [15], where the authors resized the lattice grid to guarantee full coverage even when some network nodes are not located in their designated positions. Authors in [16] proposed a protocol to protect vulnerable paths in sensing fields by using some mobile sensor nodes. This algorithm utilizes Voronoi diagrams to find vulnerable points in the sensing field. By adjusting the deployment of some nodes, this algorithm protects important areas called TBP (i.e., to-be-protected) regions. These studies let us claim that it is possible for a network to detect and localize a mobile node when it wirelessly transmits packets, even if it does not reveal its position to a service.

The coverage problem is closely related to the exposure problem. The latter tries to measure the ability of the network to track a mobile node as it moves along an arbitrary path [17]. The authors in [18] formally defined the exposure problem as the integral of an intensity function in the interval $[t_1, t_2]$ along a given path $\mathbf{p}(t)$. A problem derived from the exposure problem is the minimal exposure path [10], where a mobile node seeks a path between two given points so that the total exposure is minimized. In a related work, the authors in [19] proposed an algorithm called antidetection in which a mobile node evaluates the best path in order to traverse a sensing field without being detected. This algorithm uses routing principles to move across the sensing field based on the local level of exposure and the current position with respect to the final destination. However, this algorithm assumes that the mobile node has partial or complete knowledge of the network deployment in advance. Although this knowledge greatly simplifies the complexity of exposure-related algorithms, we consider that this assumption does not generally hold in real life. This is due to the fact that it is uncommon for real network nodes to share their positions with mobile nodes. In our approach, it is not assumed that the location of network nodes is known in advance. LEA starts by discovering this information in order to estimate the level of exposure of the mobile node. This important feature clearly sets LEA apart from other works dealing with the problem of exposure estimation. It is worth mentioning, however, that the network has the advantage of knowing the real location of each network node, whereas LEA has to work with estimates.

In the following section, we describe in detail the operation of LEA.

3. LEA

In this section, we explain both the key assumptions and main components behind LEA operation.

Let us consider a wireless network comprised of a set of static nodes scattered over a two-dimensional space. Let us also consider that there is a mobile node roaming over the network and that the network nodes within its transmission range (R_c) are able to detect its presence as soon as it transmits. We denote by $N(t) = \{n_1, n_2, \dots, n_n\}$ the set of network nodes discovered by the mobile node up to time t. Each network node n_i is characterized by two attributes, its position denoted by $p_i = (x_i, y_i)$ and its coverage area which is considered to be circular and centered at point p_i with radius R_c. Finally, it is assumed that the mobile node is able to obtain its geographic position (this can be achieved by means of GPS, for example).

In the following, we describe the two main steps of LEA. First, we describe the location estimation step which is comprised of three components. Then, we explain the exposure estimation step.

3.1. Location Estimation

3.1.1. Discovery of Network Nodes. In LEA, a mobile node starts by discovering the position of nearby network nodes. To accomplish this task, it configures its wireless interface in monitor mode in order to overhear transmissions from network nodes while it moves. From this activity, the mobile node obtains measurements of received signal strength that it uses to compute distance estimates [9]. Note that, with a single distance estimate, the mobile node is only able to determine its distance to a transmitting network node n_i. More estimates are required in order to pinpoint the position of network node n_i. The method used by LEA to achieve this end is derived from [20] and is briefly outlined below. As depicted in Figure 2, let us assume that a network node n_i has

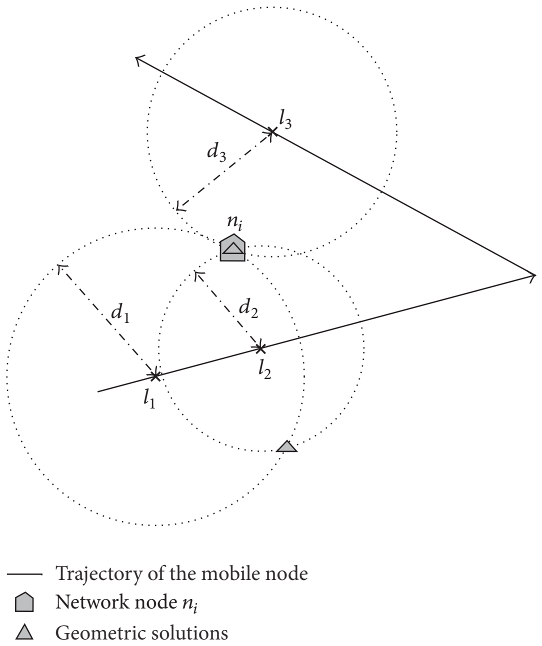


Figure 2: Discovering a network node.

been detected by a mobile node at two different locations. Let us denote by l_1 and l_2 these locations (with $l_1 \neq l_2$) and let d_1 and d_2 be the estimated distances between the mobile node and network node n_i. With two distance estimates, LEA can compute two geometric solutions for the position of network node n_i by intersecting the two overlapping circles, centered at locations l_1 and l_2 with radii d_1 and d_2, respectively. One geometric solution lies at its left side (with respect to its trajectory) while the other one lies at its right side (see the triangles shown in Figure 2). Note that left and right locations are used for descriptive purposes, since, at this point, the mobile node has no means of distinguishing which one is the real position of network node n_i.

In order to solve the location ambiguity just described and to determine the real position of network node n_i, LEA requires the mobile node to perform a change of trajectory while maintaining network node n_i within its coverage area. As a result of this action, one of the two geometric solutions can be discarded. This is illustrated in Figure 2, where the mobile node changed its trajectory and at location l_3 it estimated that its distance to node n_i was d_3. It can be seen in this figure that the solution located outside the boundary of the circle centered at location l_3 can be discarded. Note that this action completes a trilateration process [4]. At this point, there is no location ambiguity about the real location of network node n_i, since one of the two geometric solutions was discarded.

Now we explain how LEA stores the position of discovered network nodes while it roams. It is well known that a binary tree (BT) can be used for fast data access and retrieval in data structures. LEA uses a BT to build local maps of discovered network nodes. In LEA, a tree T is a linked data structure without cycles in which a network node n_i is an entity whose relevant fields (i.e., attributes) are

(i) key[n_i]: node identifier,

(ii) left[n_i]: pointer to the left subtree,

(iii) right[n_i]: pointer to the right subtree,

(iv) color[n_i]: state criterion,

(v) left-side[n_i]: the left position of the two geometric solutions,

(vi) right-side[n_i]: the right position of the two geometric solutions,

(vii) real[n_i]: the real position of n_i.

In LEA, each discovered network node can be in one out of three possible states, which are represented with different colors:

(i) white: this color denotes the fact that a first distance estimation to network node n_i has been performed (first encounter)

(ii) gray: this color denotes the fact that a second distance estimation to network node n_i has been performed (second encounter). At this point, there is a location ambiguity because two geometric solutions can be established

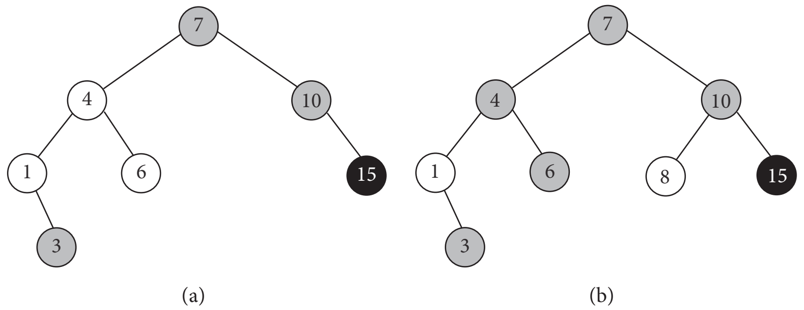
(iii) black: this color denotes the fact that a trilateration process has been performed (third encounter). The location ambiguity has been removed with a change of trajectory and the real location of network node n_i has been established

Whenever a new network node n_j is inserted in T, the following rules must be followed:

(i) Node n_j is in the left subtree of node n_i if and only if key[n_j] < key[n_i].

(ii) Node n_j is in the right subtree of node n_i if and only if key[n_j] > key[n_i].

To accomplish the rules just mentioned, LEA uses the TREE-INSERT-UPDATE procedure shown in Algorithm 1, which operates as follows. Once the first encounter with network node n_j occurs, the algorithm uses function Node(id) to insert a new node n_j in T and sets attribute key[n_j] ← id and color[n_j] ← white. If a second encounter with network node n_j occurs, function Color(n_j) will update the attribute color[n_j] to gray and will set the attributes left-side[n_j] and right-side[n_j] with the two geometric solutions. When the location ambiguity of network node n_j is eliminated, function Color(n_j) updates the attribute color[n_j] to black and stores the real position of network node n_j in the attribute real[n_j]. LEA names the first node inserted in T as the root node. The worst-case time complexity needed to insert/update a network node requires $\Theta(n)$, where n is the number of nodes in T, since each network node is visited at most once. The time complexity of a BT depends on the height of T; however, it is well known that the expected height of a binary tree is proportional to $O(\log n)$ [21]. Information such as the color and positions (i.e., left-side[n_i], right-side[n_i], and real[n_i]) of a network node n_i can be retrieved from T in $O(\log n)$.



(a)

(b)

Figure 3: Example of a binary tree in LEA. (a) Tree at time t_1 and (b) tree at time $t_1 + \Delta t$.

```
TREE-NODE($id$)
if root = NIL then
    root ← Node($id$)
else
    TREE-INSERT-UPDATE(root, $id$)
end if
TREE-INSERT-UPDATE ($n_i$, $id$)
 if $id$ < key[$n_i$] then
    if left[$n_i$] = NIL then
        left[$n_i$] ← Node($id$)
    else
        TREE-INSERT-UPDATE(left[$n_i$], $id$)
    end if
else if $id$ > key[$n_i$] then
    if right[$x$] = NIL then
        right[$n_i$] ← Node($id$)
    else
        TREE-INSERT-UPDATE(right[$n_i$], $id$)
    end if
else
    Color($n_i$)
end if
```

Algorithm 1: TREE-INSERT-UPDATE.

As an example, suppose that at time t_1 nodes 1, 4, and 6 are colored in white, nodes 3, 7, and 10 are colored in gray, and node 15 is colored in black (see Figure 3(a)). At time $t_1 + \Delta t$, the mobile node detects for the second time nodes 4 and 6, while it detects for the first time node 8. This implies that nodes 4 and 6 turn gray (i.e., two geometric solutions for each node can be established), while node 8 is inserted in white color as a new node in T (see Figure 3(b)). Nodes 1, 3, 7, and 10 did not change their colors, since they were not detected again by the mobile node between t_1 and $t_1 + \Delta t$.

3.1.2. Creation of Possible Scenarios. As it has been explained, the information discovered by LEA is organized in a tree structure, where each node may be white, gray, or black depending on the accuracy with which its position has been established. From this information, LEA takes into consideration gray and black nodes in order to create a number of possible scenarios (i.e., all possible combinations of network node locations or network maps). This procedure is explained below.

Let $N_{\text{in}}(t) = \{n_1, n_2, \ldots, n_k\}$ be a subset of $N(t)$ comprised by the k nearby gray and black nodes that would be able to detect the presence of the mobile node if it transmitted at time t (i.e., network nodes within the transmission range R_c of the mobile node). The procedure for the creation of different scenarios can be more clearly understood with the help of matrix $M(t)$, which represents all possible scenarios (combinations) of network node locations in $N_{\text{in}}(t)$. This matrix is comprised of $2^{k-\text{black}}$ rows and k columns, where black denotes the number of network nodes colored in black belonging to $N_{\text{in}}(t)$. Each row of $M(t)$ looks like a binary number of length k bits in the interval $[0, 2^{k-\text{black}} - 1]$ so that consecutive rows constitute an increasing sequence in steps of one unit.

As a first example of the construction of matrix $M(t)$, suppose that at time t the mobile node detected three network nodes in its transmission range whose IDs are n_3, n_7, and n_{10}, respectively. Suppose that in this case all network nodes are colored in gray. Each gray node is represented in matrix $M(t)$ with two values, namely, "0" and "1," in order to indicate left and right locations, respectively. Therefore, matrix $M(t)$ for this example becomes

$$M(t) = \left(\begin{array}{ccc} n_3 & n_7 & n_{10} \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}\right). \qquad (1)$$

Note that each row of $M(t)$ contains a binary number of length 3 in the interval $[0, 2^{3-0} - 1]$ and each column of $M(t)$ is associated with a network node $n_i \in N_{\text{in}}(t)$ ordered by its ID in ascending order from left to right.

Each row of $M(t)$ represents a possible scenario to be considered, since there is no certainty about the real position of nodes n_3, n_7, and n_{10} at time t. For instance, scenario

[0 0 1] consists of the left-side position of node n_3, the left-side position of node n_7, and the right-side position of node n_{10}. These locations and node colors can be obtained by searching the attributes of each network node n_i in T.

Now let us assume a more general case where there are both gray and black nodes in $N_{\text{in}}(t)$. Matrix $M(t)$ is initially constructed by considering the gray nodes and applying the corresponding rules just explained. In addition, black nodes must be associated with the leftmost columns of $M(t)$ and must be ordered by their IDs in ascending order from left to right. For convenience, the corresponding columns for black nodes must be filled with "0's." As an example, suppose that the mobile node detected four network nodes in its transmission range at time t; that is, $|N_{\text{in}}(t)| = 4$ (see matrix (2)), whose IDs are n_4, n_6, n_9, and n_{15}. Nodes n_4 and n_9 are colored in gray, whereas nodes n_6 and n_{15} are colored in black. Matrix $M(t)$ for this example becomes

$$M(t) = \left(\begin{array}{cccc} n_6 & n_{15} & n_4 & n_9 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right). \qquad (2)$$

Note that the two leftmost columns of $M(t)$ are filled with "0's," since they correspond to black nodes 6 and 15. This example clearly illustrates that, in a given scenario, "0" may represent either a real position or a left-side position depending on the color of the node. For instance, scenario [0 0 0 1] consists of the real positions for nodes n_6 and n_{15}, the left-side position of node n_4, and the right-side position of node n_9.

When $N_{\text{in}}(t)$ only contains black nodes, the corresponding matrix $M(t)$ becomes a row vector of all zeroes, which implies that there is only one possible scenario consisting of their real positions.

3.1.3. Emulation of a Localization Algorithm. In the following step, LEA must emulate how the network nodes establish the location of the mobile node. This procedure is carried out assuming that the network nodes are placed at the positions of a given scenario. LEA estimates its distance to each network node and applies trilateration techniques [22, 23] in order to compute the location of the mobile node *from the network point of view*. This procedure is repeated for each scenario, that is, for each row in $M(t)$.

At the end of the process, LEA has as many possible locations for the mobile node as rows in matrix $M(t)$. By using a convex hull (CH) algorithm [21, 24] that takes into account each estimated point associated with each row, LEA builds a unique weighted convex polygon, which represents the area where the wireless network locates the mobile node [4, 25]. Finally, the centroid of such weighted polygon represents the network-estimated location of the mobile node by considering all possible scenarios.

3.2. Exposure Estimation. LEA computes the level of exposure at time t as the difference between the location estimated in the location estimation step and the true location of the mobile node (which is assumed to be known by means of a positioning system, such as GPS). This distance is a measure of location privacy for the mobile node.

In the following section, we discuss in detail diverse issues to be addressed during the emulation step of LEA when it is applied in dense and sparse networks.

4. LEA in Dense and Sparse Networks

In this section, we illustrate the considerations to be taken into account during the emulation of a localization algorithm. As a case study, we use Ghost [5], but similar considerations have to be taken with other localization algorithms.

4.1. Ghost Emulation in LEA. Ghost is a localization algorithm intended to be used by a set of network nodes that cooperate among themselves in order to track a mobile node. To achieve this end, every time a mobile node is detected by at least one network node, the Euclidean space is divided into convex polygons by creating a Voronoi tessellation in order to find the most likely polygon where the node is located. LEA emulates Ghost by performing this division as explained below.

Let r_j denote the jth row of $M(t)$ so that each vector r_j represents a possible scenario. For each r_j, a Voronoi tessellation [24] is computed by taking into account two sets of points:

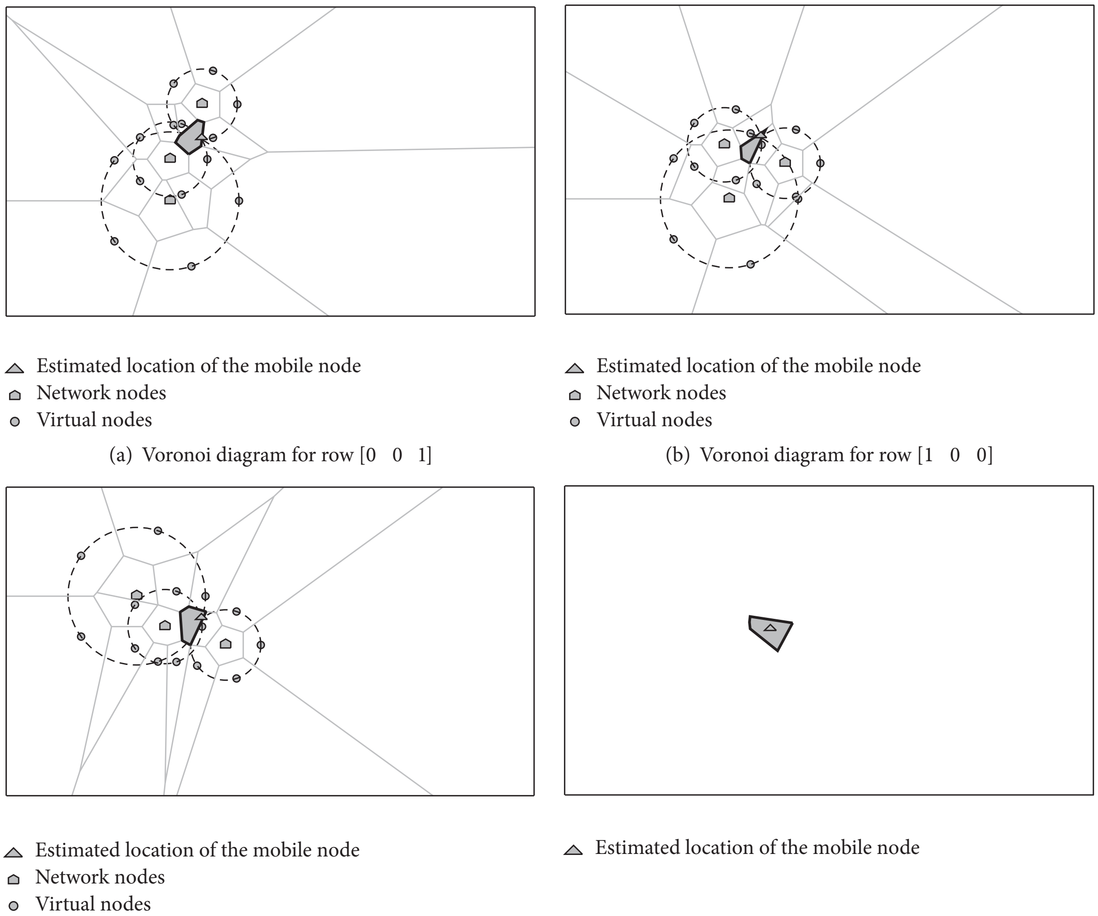
(1) For each network node $n_i \in N_{\text{in}}(t)$ (recall that $N_{\text{in}}(t) = \{n_1, n_2, \ldots, n_k\}$), LEA places m equidistant virtual nodes along a circle centered at the coordinates of n_i having a radius equal to the estimated distance between n_i and the mobile node at time t. Depending on the scenario, these coordinates can be either left-side[n_i] or right-side[n_i] for a gray node or real[n_i] for a black node. See the example depicted in Figure 4.

(2) Let $N_{\text{out}}(t)$ be the set of gray and black network nodes in $N(t)$ which do not belong to $N_{\text{in}}(t)$. For each network node $n_x \in N_{\text{out}}(t)$, LEA places m equidistant virtual nodes as follows:

   (a) If attribute color[n_x] = gray, both left-side[n_x] and right-side[n_x] are considered. Therefore, LEA places m virtual nodes along a circle centered at left-side[n_x] and m virtual nodes along another circle centered at right-side[n_x]. Each one of these circles has a radius equal to the maximum communication range (i.e., R_c).

   (b) If attribute color[n_x] = black, only the real location is considered. LEA places m virtual nodes along a circle centered at real[n_x], having a radius equal to the maximum communication range (R_c).

It is worth mentioning that, in all cases, all procedures, related to the division of the Euclidean space, constitute an atomic operation that has to be performed at time t.


△ Estimated location of the mobile node
⌂ Network nodes
○ Virtual nodes
△ Estimated location of the mobile node
⌂ Network nodes
○ Virtual nodes
△ Estimated location of the mobile node
⌂ Network nodes
○ Virtual nodes
△ Estimated location of the mobile node

(a) Voronoi diagram for row [0 0 1]

(b) Voronoi diagram for row [1 0 0]

(c) Voronoi diagram for row [1 1 0]

(d) Resulting polygon by using CH

FIGURE 4: LEA operation for cases when $|N_{\text{in}}(t)| \geq 3$. (a) LEA selects the most probable polygon where the mobile node is located at time t for row [0 0 1] (see polygon filled in gray color). (b) In a similar way, for row [1 0 0], LEA chooses the most probable polygon where the mobile node is located at time t; (c) shows the associated Voronoi diagram for row [1 1 0] at time t. Finally, LEA uses a convex hull (CH) algorithm to compute a unique resulting polygon where the mobile node is likely located.

In the following subsections, we describe how LEA emulates Ghost in three main scenarios. These cases are referred to as triple coverage, double coverage, and single coverage, depending on the number of network nodes detecting the mobile node. These cases are explained below and further details can be found in [5].

4.2. Triple and Higher-Order Coverage. Let us explain how LEA emulates Ghost when the mobile node detects three or more network nodes at time t; that is, $|N_{\text{in}}(t)| \geq 3$. For each row $r_j \in M(t)$, LEA computes the associated Voronoi diagram considering the gray and black nodes of the scenario (i.e., nodes in both sets $N_{\text{in}}(t)$ and $N_{\text{out}}(t)$) and the virtual nodes placed around these nodes. As an example, consider the case corresponding to the matrix shown in (1) in which the number of rows is equal to eight: $|N_{\text{in}}(t)| = 3$ (i.e., three gray nodes are detected) and $N_{\text{out}}(t)$ is empty. Figures 4(a)–4(c) show some examples of Voronoi diagrams for rows [0 0 1], [1 0 0], and [1 1 0], respectively. For the sake of simplicity, we only show three rows out of eight. Note that each network node in these figures has $m = 5$ virtual nodes (shown as gray circles). As it can be observed, the purpose of the virtual nodes in the Ghost algorithm is to create smaller polygons during the partitioning of the Euclidean space and, therefore, to locate the mobile node with greater accuracy. For each Voronoi diagram, LEA uses a trilateration algorithm [9, 12, 22, 23] to locate the mobile node by using the distances computed between each network node n_{i} and the location of the mobile node at time t. Then, according to Ghost, LEA selects the most probable convex polygon where the mobile node is located by means of a point location algorithm (e.g., [26]). This is also illustrated in Figures 4(a)–4(c). We emphasize that a trilateration algorithm can be applied only if $|N_{\text{in}}(t)| \geq 3$.

After obtaining the associated convex polygon for each row $r_j \in M(t)$, the location of the mobile node can be obtained by computing a weighted *centroid* as mentioned in Section 3.1.3. To accomplish this task, we use a convex hull (CH) algorithm [21, 24] that takes into account the centroid of each convex polygon associated with each row

$r_j \in M(t)$. As a result of computing a CH algorithm, there is a unique weighted convex polygon, which represents the area where the wireless network localizes the mobile node [25]. From now on, we refer to the centroid of this polygon as a *hook* point, as it is also referred to in Ghost. For instance, Figure 4(d) shows the weighted convex polygon corresponding to the matrix shown in (1) mentioned above where the gray-filled triangle represents the likely location of the mobile node (i.e., hook point) as seen by the network (i.e., Ghost).

4.2.1. Dealing with Special Cases in Triple Coverage. Building a polygon using a convex hull algorithm requires as input at least three noncollinear centroids. That is, $M(t)$ must have at least three rows in order to generate one centroid from each scenario. As an example, let us consider that there are three nodes in set $N_{\text{in}}(t)$. It is clear that if there is one black node and two gray nodes, $M(t)$ is comprised of $2^{3-1} = 4$ rows and a CH algorithm can be applied. The same case applies if there are three gray nodes, since $M(t)$ consists of eight rows. However, in the following two remaining cases, it would not be possible to apply a convex hull algorithm. Suppose that one network node in $N_{\text{in}}(t)$ is colored in gray, while the others are colored in black. Thus, the number of rows in $M(t)$ is equal to two. It follows that both the number of polygons computed by the point location algorithm and their associated centroids are equal to two. Clearly, a CH algorithm cannot be applied in this case. LEA overcomes this situation by intersecting the two computed polygons to find the area where the wireless network detects the mobile node. Consider now the case when all network nodes in $N_{\text{in}}(t)$ are colored in black. The likely location of the mobile node for this scenario is the centroid of the unique polygon computed by means of the point location algorithm, since there is only one row in $M(t)$. This situation appears only when the location ambiguity of all network nodes in $N_{\text{in}}(t)$ was solved by time t. Since no ambiguity exists in this scenario, the network and LEA establish similar locations of the mobile node. Thus, this is the ideal scenario from the localizability point of view.

4.3. Double Coverage. Now let us explain how LEA operates when the mobile node only detects two network nodes at time t; that is, $|N_{\text{in}}(t)| = 2$. According to what was explained in Section 3.1.2, whenever the mobile node detects at least one network node, a matrix $M(t)$ can be constructed by using network nodes belonging to $N_{\text{in}}(t)$. For example, consider that $N_{\text{in}}(t) = \{n_1, n_2\}$, $N_{\text{out}}(t)$ is empty, and the two network nodes in $N_{\text{in}}(t)$ are colored in gray. The associated matrix $M(t)$ for this example is

$$M(t) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}. \tag{3}$$

An example of the Euclidean space division for rows [0 0], [0 1], [1 0], and [1 1] and $m = 9$ is depicted in Figures 5(a)–5(d), respectively. These figures also illustrate that, for each $r_j \in M(t)$, the estimated position of the mobile node has two geometric solutions as explained before (shown as gray triangles). Let us denote these solutions by $\widehat{M}_n^j$ and $\widehat{M}_f^j$, where subindexes n and f stand for near and far, respectively (in the original description of Ghost [5] these solutions are called *real* and *mirrored* locations) and superindex j denotes the jth row of $M(t)$ for $1 \leq j \leq 2^{k-\text{black}}$. Then, for each $r_j \in M(t)$, LEA selects the most probable convex polygons where the mobile node is located by means of a point location algorithm which takes as inputs $\widehat{M}_n^j$ and $\widehat{M}_f^j$. This is also depicted in Figures 5(a)–5(d).

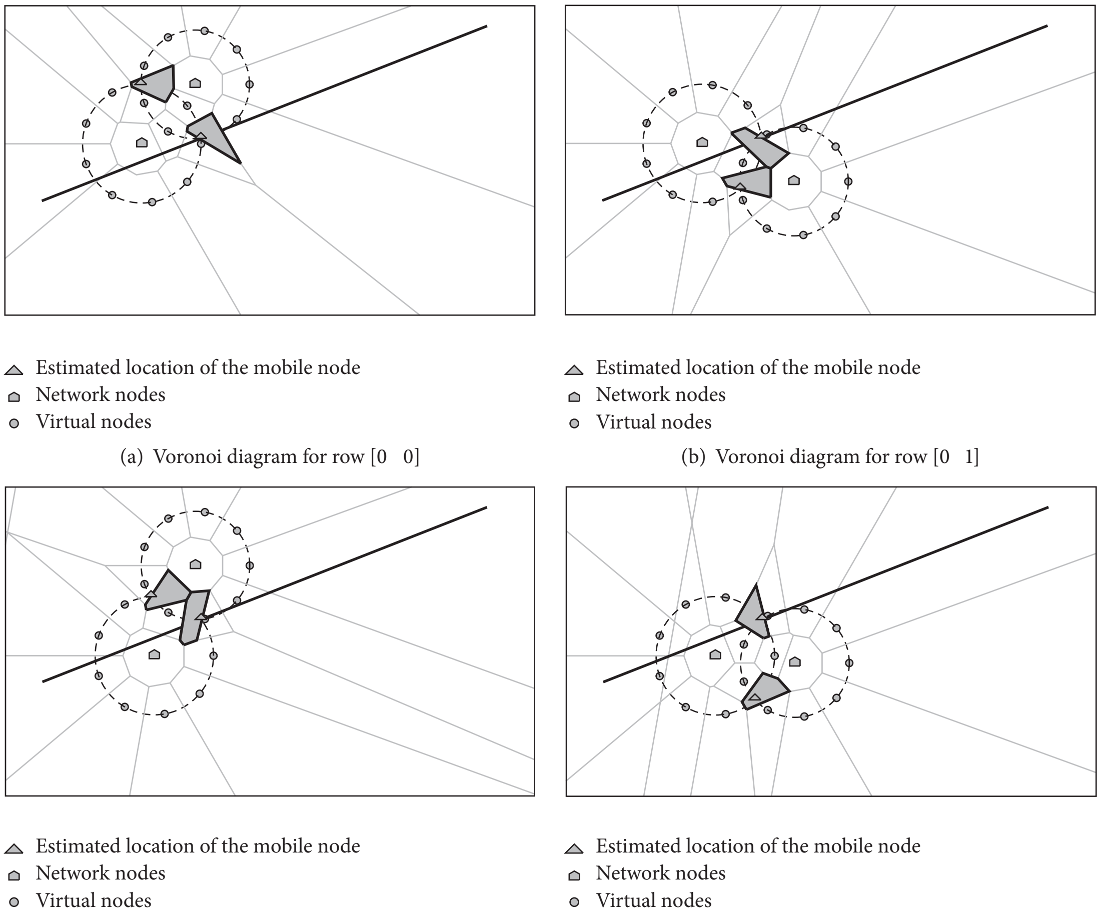
LEA cannot apply a convex hull algorithm in a double-coverage scenario in order to find a weighted convex polygon where the mobile node is likely located. In this case, LEA combines different scenarios r_j as follows. Let us consider a mobile node moving from location l_1 to location l_2 as depicted in Figure 6. At location l_2, geometric solutions $\widehat{M}_n^1$, $\widehat{M}_f^1$, $\widehat{M}_n^2$, $\widehat{M}_f^2$, $\widehat{M}_n^3$, $\widehat{M}_f^3$, $\widehat{M}_n^4$, and $\widehat{M}_f^4$ are established. Since locations $\widehat{M}_n^1$, $\widehat{M}_n^2$, $\widehat{M}_n^3$, and $\widehat{M}_n^4$ are the same point, let us refer to them by $\widehat{M}_n^*$ (see the corresponding triangle in Figure 6). For each $r_j \in M(t)$, we can observe that locations $\widehat{M}_f^1$ and $\widehat{M}_f^4$ (see the corresponding triangles in Figure 6) are symmetrical with respect to the trajectory of the mobile node (i.e., $l_1 \to l_2$). In the same figure, we also observe that locations $\widehat{M}_f^2$ and $\widehat{M}_f^3$ are also symmetrical with respect to the trajectory of the mobile node. Due to these symmetry conditions, LEA may discard rows [1 0] and [1 1] from $M(t)$, since, from the network perspective, locations $\widehat{M}_f^4$ and $\widehat{M}_f^3$ produce similar location errors in comparison to locations $\widehat{M}_f^1$ and $\widehat{M}_f^2$. As a result, matrix $M(t)$ becomes

$$M'(t) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{4}$$

for cases when $|N_{\text{in}}(t)| = 2$ and the two network nodes are colored in gray.

Now, let us explain how LEA computes the likely location of the mobile node in this case. For each $r_j \in M'(t)$, two polygons are determined by means of a point location algorithm: one of them (the one associated with the "near" solution) always lies on the trajectory of the mobile node, whereas the other (the one associated with the "far" solution) is situated at either the left side or the right side, with respect to the same trajectory. This situation is illustrated in Figures 5(a) and 5(b). Since at this point LEA cannot discard any of these polygons (under some circumstances it will be able to use a method described in Section 4.3.3 to reduce the location ambiguity), it computes three polygons as likely locations of the mobile node at time t. The first polygon is constructed by intersecting polygons located along the trajectory of the mobile node (i.e., polygons associated with locations $\widehat{M}_n^1$ and $\widehat{M}_n^2$), while the other two are polygons associated with locations $\widehat{M}_f^1$ and $\widehat{M}_f^2$.

4.3.1. Dealing with Special Cases in Double Coverage. Suppose that the two network nodes in $N_{\text{in}}(t)$ are colored in black; in



(a) Voronoi diagram for row [0 0]

(b) Voronoi diagram for row [0 1]

(c) Voronoi diagram for row [1 0]

(d) Voronoi diagram for row [1 1]

Figure 5: LEA operation for cases when $|N_{\text{in}}(t)| = 2$. LEA selects the most probable polygons where the mobile node is located at time t for rows [0 0], [0 1], [1 0], and [1 1] (see polygons filled in gray color) shown in subfigures (a), (b), (c), and (d), respectively.

this case, matrix $M(t)$ becomes [0 0]. As a result, the likely locations of the mobile node are the centroids of the two polygons determined by means of a point location algorithm (i.e., polygons associated with locations $\widehat{M}_n^1$ and $\widehat{M}_f^1$).

It is worth emphasizing that if, at some point in time, $N_{\text{in}}(t)$ contains a gray node and a black node, the black node eliminates the location ambiguity associated with the gray node and this scenario becomes the one with two black nodes just described (see Figure 6).

4.3.2. Maximum Speed Circle. LEA uses a concept introduced in [5] to deduce additional information when only one or two network nodes detect a mobile node, namely, the maximum speed circle (MSC). The MSC is defined as a circle with radius initially equal to zero and located at the centroid of a previously selected polygon at time $t - \Delta t$ (later, we will explain how LEA chooses such polygon). Then the MSC increases its radius at maximum speed, denoted by $V_{\max}$, until time t, in which the mobile node attempts to perform a location-exposure estimation. Since the mobile node can move freely in any direction at a certain speed in the interval $[0, V_{\max}]$, the resulting MSC represents all its possible locations at time t.

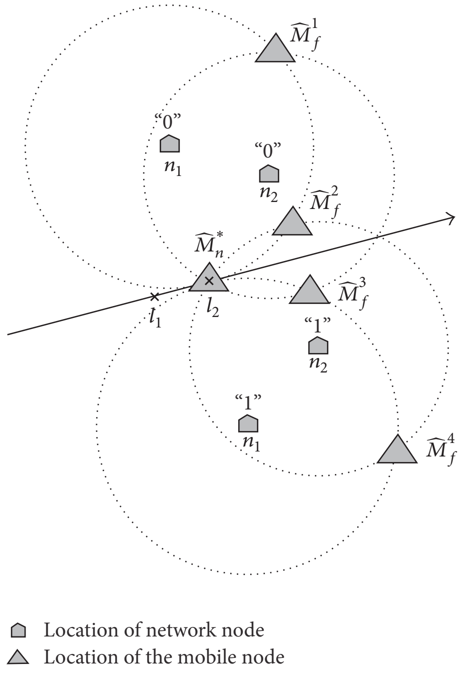


Figure 6: All possible locations for the double-coverage case.

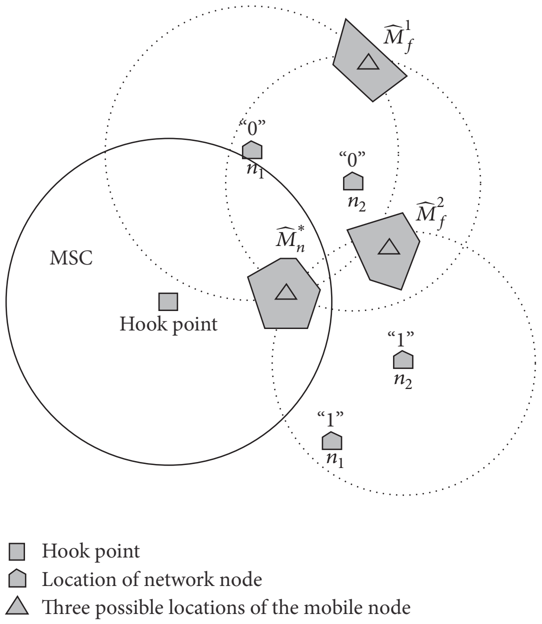


Figure 7: Double-coverage scenario using MSC.

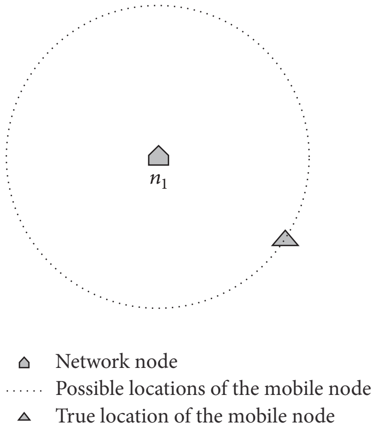

n_1
Network node
Possible locations of the mobile node
True location of the mobile node

Figure 8: Single coverage.

4.3.3. Double Coverage Using MSC. Sometimes the MSC can be used to discard location ambiguities in double-coverage scenarios as explained below. Let us assume that, at time t, LEA has three likely locations of the mobile node for cases when $M(t)$ becomes $\left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right)$ or two likely locations for cases when $M(t)$ becomes $[0\ \ 0]$ (i.e., two gray nodes or two black nodes in $N_{\text{in}}(t)$, resp.). LEA can eliminate ambiguity in the location of the mobile node if and only if some of the likely locations computed at time t are outside the current MSC. In the example depicted in Figure 7, polygons associated with locations $\widehat{M}_f^1$ and $\widehat{M}_f^2$ are discarded, since both are located outside the current MSC. Then a new MSC with radius equal to zero is placed at the center of the selected polygon(s) as new hook point(s) (in Figure 7, this is location $\widehat{M}_n^*$). For cases when no hook point exists at time $t - \Delta t$, LEA cannot discard any location ambiguity of the mobile node. However, even in such cases, LEA places a new MSC at the center of each likely polygon computed at time t as new set of hook points. In the example illustrated in Figure 7, suppose that no hook point exists at time $t - \Delta t$; then LEA places three hook points at the center of each polygon associated with locations $\widehat{M}_n^*$, $\widehat{M}_f^1$, and $\widehat{M}_f^2$. This process continues until the mobile node leaves the double-coverage scenario.

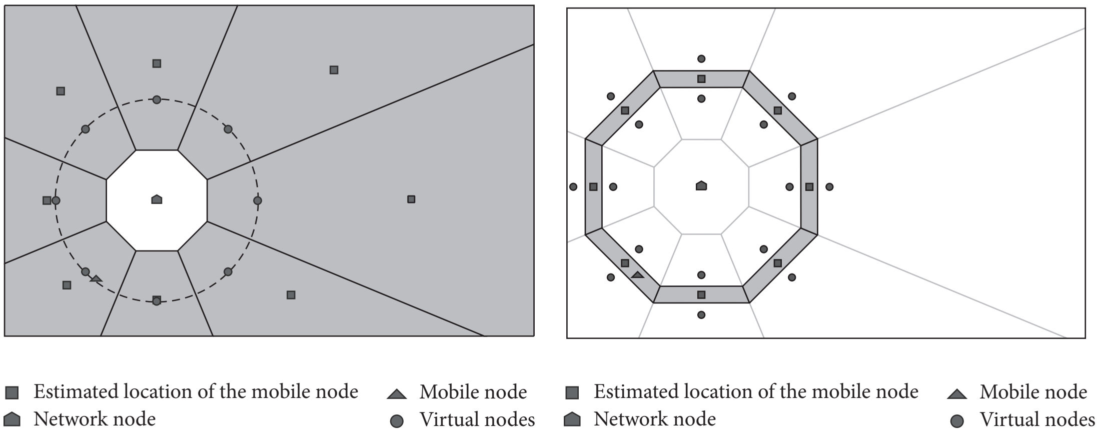
4.4. Single Coverage. Now let us explain how LEA operates when the mobile node detects only one network node at time t; that is $|N_{\text{in}}(t)| = 1$. In this case, there are only two possibilities. If the node is black, then $M(t) = [0]$; otherwise, for a gray node, $M(t) = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$. In the latter case, there are two possible locations for the network node and they are located at both sides of the trajectory (i.e., left and right). However, due to symmetry conditions, LEA may discard one of these locations, since both lead to similar location errors. LEA discards the right location so that, in all cases when $|N_{\text{in}}(t)| = 1$, the matrix of scenarios becomes $[0]$.

From the network perspective, the case where $|N_{\text{in}}(t)| = 1$ implies that there exist an infinite number of possible locations for the mobile node represented by a circle centered at the network node (see Figure 8). This circle has a radius that equals the estimated distance between both nodes at time t. At this point, LEA could place virtual nodes along this circle and compute a Voronoi tessellation, but resulting polygons would not have bounding limits other than the boundaries of the considered area (see Figure 9(a)). Therefore, using the centroid of these polygons as the likely locations of the mobile node might result in large location errors. As explained below, LEA uses some strategies in order to decrease this location uncertainty.

One of such strategies consists in limiting the size of each polygon by using three circles instead of one. If we denote the distance between the ith network node and the mobile node, at time t, by $d_{i,t}$, virtual nodes are placed along the three circles whose radii are $d_{i,t}$ and $d_{i,t} \pm \Delta d$, respectively. All these points are considered during the construction of the Voronoi diagram, so that the resulting annular-like area approximates the region in which the mobile node lies (see Figure 9(b)). This procedure can be used to construct a set of polygons where each one of them is bounded by four edges.

The MSC method can also be used in conjunction with the previous strategy in order to reduce location ambiguity. Assume that LEA is aware of the last location of the mobile node before entering the single-coverage scope (e.g., hook point computed at time $t-\Delta t$). The MSC is centered at the last known hook point so that when the mobile node attempts to estimate its level of exposure, at time t, only polygons falling within the MSC are taken into account. However, in the event that no hook points exist before entering the single-coverage scope, LEA cannot place any MSC in order to reduce the location uncertainty.

We close this section by pointing out that, in some single-coverage cases, LEA may incur in higher location errors than the network. Figure 10 illustrates a simple network arrangement leading to the creation of regions covered by three, two, and even one network node (labeled as 3, 2, and



(a) Voronoi diagram formed by eight virtual nodes and one network node

(b) Annular-like region constructed from one distance estimation and three sets of virtual nodes

Figure 9: Single-coverage detection using virtual nodes.

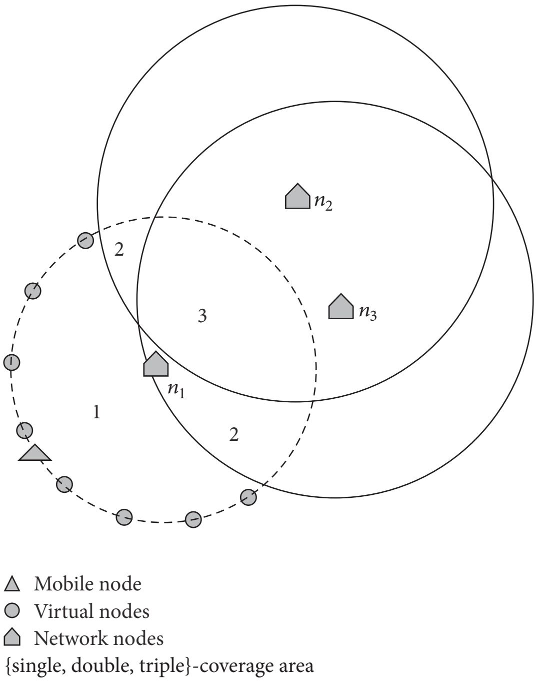


Figure 10: Single-coverage scope.

1 in the figure, resp.). Assume a situation where a mobile node is located at some point that simultaneously belongs to the circle around network node n_1 and the single-coverage region. When network node n_1 tries to localize the mobile node, it places virtual nodes over the aforementioned circle and it can also use information coming from nodes n_3 and n_2 (in this case, through a 2-hop route) to rule out some virtual nodes as likely positions for the mobile node. However, LEA cannot use this information, since the mobile node is not expected to communicate with the network nodes. As a result, in some cases, LEA may incur in higher location errors with respect to the network point of view.

5. LEA Performance Tests

We conducted a series of simulation experiments as well as real testbed experiments in order to evaluate the accuracy and limitations of LEA.

5.1. LEA Simulations. We used a custom-made simulator written in Python language to conduct a series of simulation experiments under a scenario proposed in [6] and also used in [5]. We use two localization algorithms, Ghost [5] and GPLT (geometric-assisted predictive location tracking) [6], in order to compare how closely LEA is able to emulate the network behavior. We selected these algorithms, since both are focused on locating a mobile node when less than three network nodes detect the mobile node. GPLT is a technique that uses predictive information obtained by a Kalman filter to provide additional measurements from missing network nodes in order to let trilateration techniques work properly.

5.1.1. Error Model. In order to evaluate the performance of LEA in the simulator, we use as performance metric the mean location error. This concept is defined as the average Euclidean distance between the location of the mobile node estimated by LEA and its real location. In order to simulate estimation errors in our experiments, we use the same noise model employed in [5, 6]. This model uses the real Euclidean distance between the ith network node and the mobile node at time t, denoted by $d_{i,t}$, in order to generate distance estimates ($r_{i,t}$) as follows:

$$r_{it} = d_{i,t} + n_{i,t} + e_{i,t}, \quad i = 1, 2, 3, \ldots, n, \tag{5}$$

where $n_{i,t}$ is additive noise, which is assumed to follow a Gaussian distribution with zero mean and standard deviation of 10 meters, that is, $n_{i,t} \sim \mathcal{N}(0, 100)$. In turn, $e_{i,t}$ denotes the non-line-of-sight (NLOS) distance error. This can be computed from the excessive propagation delay (τ) with respect

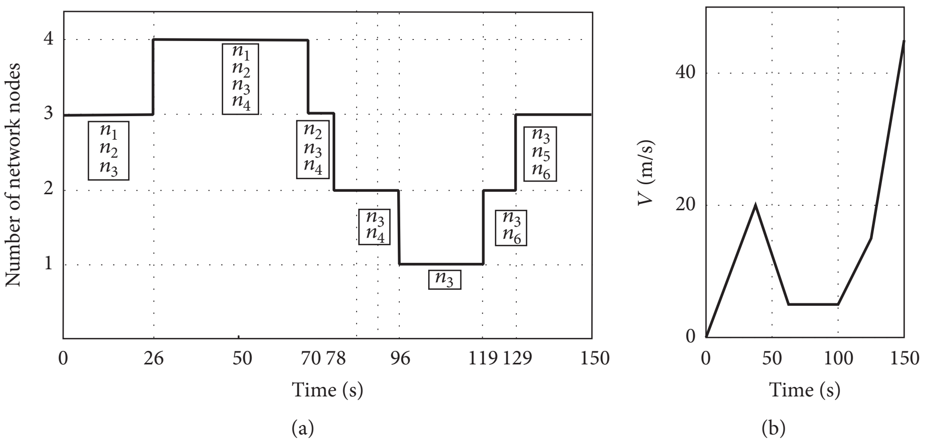


Figure 11: Elements of the simulation scenario. (a) Number of network nodes detecting the mobile node; (b) speed model.

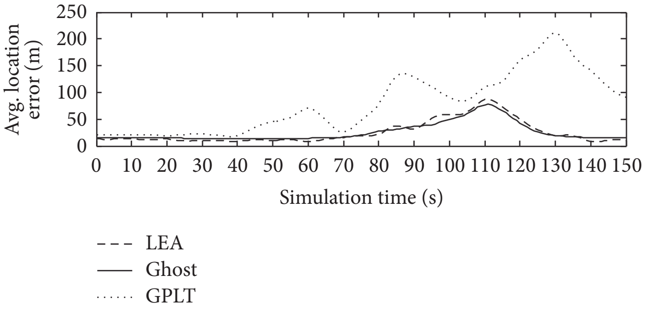


Figure 12: LEA versus Ghost and GPLT.

to a direct path, which is modeled using an exponential distribution as follows [27]:

$$f_{e_{i,t}}(\tau) = \begin{cases} \dfrac{1}{T_{i,t}} \exp\left(-\dfrac{\tau}{T_{i,t}}\right), & \tau > 0, \\ 0, & \text{otherwise}, \end{cases} \quad (6)$$

where $T_{i,t} = T_1 (d_{i,t})^{\epsilon} \xi$. Parameter T_1 is the median value of the delay spread at $d = 1$ km, which is selected as 0.1 μs (parameter value for rural environments [27]). Exponent ϵ is selected as 0.5 and $10 \log(\xi)$ is a Gaussian random variable with zero mean and standard deviation of 4 dB.

5.1.2. LEA versus Ghost and GPLT. In these experiments, we used the scenario proposed in [6], namely, the same error model, network dimensions, transmission range, and similar mobility characteristics in terms of trajectory and speed. Figure 11(a) shows how the number of base stations detecting the mobile node varies over time as the simulation progresses and Figure 11(b) depicts the speed model used in the tests. For Ghost-related simulations, we used $m = 30$ virtual nodes per network node, since this number provides good accuracy for most scenarios [5].

We ran a series of simulation tests in order to compare the mean location error achieved by LEA with Ghost emulation (running in the mobile terminal) with respect to the ones achieved by GPLT and Ghost (assumed to be executed by the network). These results are shown in Figure 12. We observe that when three or more network nodes detect the mobile node (trilateration scenario), the location error achieved by LEA tends to be similar to the ones obtained with both Ghost and GPLT. However, when there are two or one BS detecting the mobile node, Ghost clearly outperforms GPLT in terms of location accuracy. This situation can be observed in the interval from 78 to 129 seconds shown in Figure 12. In all cases, it is observed that LEA and Ghost have similar location errors, which indicates that LEA is indeed able to reproduce the location accuracy achieved by the network.

In the results just described, it is observed that Ghost leads to lower location errors than GPLT, especially when only one or two network nodes detect the mobile node. Due to this reason, in the remaining experiments, we will only compare LEA with Ghost.

We conducted two more experiments using the scenario proposed in [6]. In the first experiment, we added BS_7 and BS_8, located at (1800, 0) and (3000, 4000), respectively (coordinates are in meters), in order to guarantee that the mobile node is always detected by at least three network nodes during the whole trace. Figure 13(a) shows the resulting number of network nodes detecting the mobile node and confirms that it is always greater than or equal to three. Figure 13(b) shows the location error achieved by LEA and Ghost during the simulation. We can see in this figure that LEA is again able to accurately reproduce the location error achieved by the network for cases when there exist three or more network nodes in each point of the trace. In the second experiment, we removed BS_2, BS_4, and BS_5 to guarantee that the network node is always detected by less than three network nodes. Figure 14(a) illustrates the effect of this change, whereas Figure 14(b) shows the location error achieved by LEA and Ghost during the simulation. Again, we can see in this figure that LEA is able to achieve a similar location error as computed by the network for most points in the trace.

5.2. Testbed Experiments. This subsection presents the performance of LEA under real working conditions. We conducted a series of experiments where a roaming user carrying a smartphone discovered a set of network nodes deployed

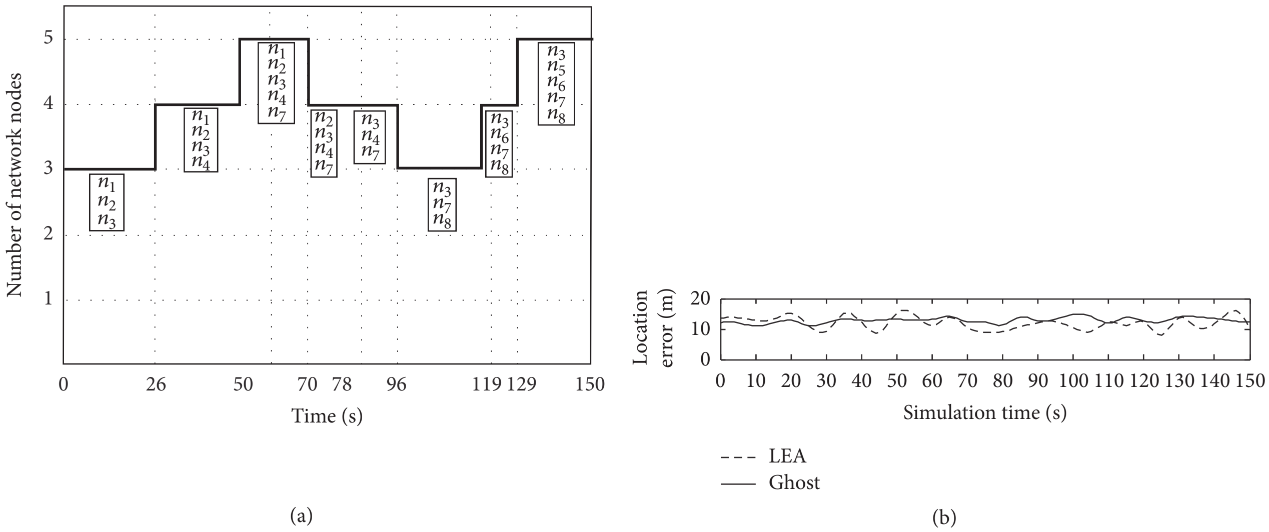


Figure 13: Scenario with three or more network nodes. (a) Network nodes detecting the mobile node; (b) comparison between LEA and Ghost in terms of location error.

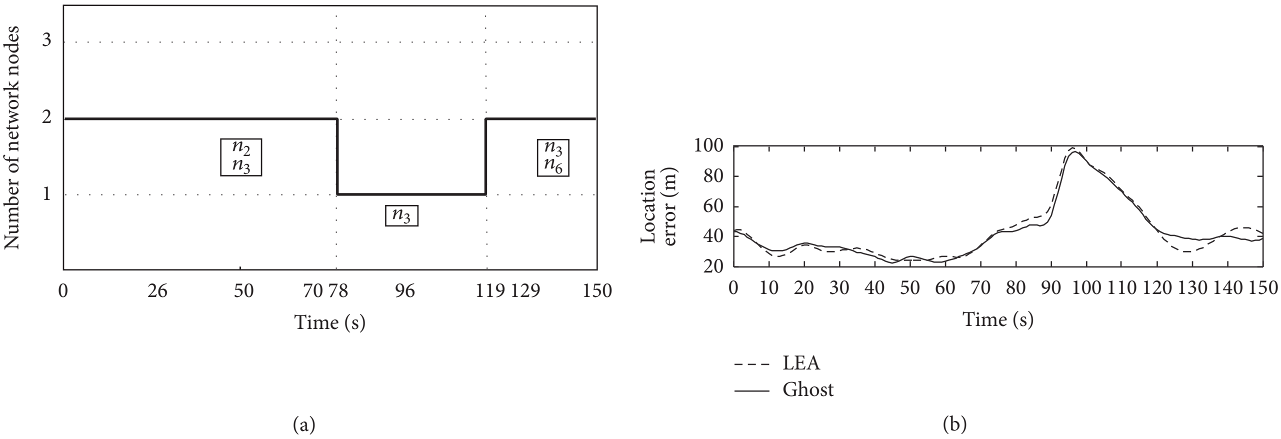


Figure 14: Scenario with less than three network nodes. (a) Network nodes detecting the mobile node; (b) comparison between LEA and Ghost in terms of location error.

across a university campus in Mexico City. The studied region roughly corresponds to a rectangle measuring 400 m × 200 m and its origin was located at the upper left corner (see Figure 15). This scenario was composed of university buildings, trees, cars, and other types of transportation vehicles frequently interfering the line of sight between the mobile node and network nodes. Before running the actual experiment, we characterized the signal attenuation in the experimentation area by means of RSSI measurements taken at different distances between two nodes. As a result of these measurements, we obtained the curves shown in Figure 16, where the one labeled as "Real" denotes the empirical measurements taken at different distances. We selected the path-loss exponent (n) to be 2.649 for distances less than the breakpoint (d_{bp}) [28, 29] whose value was equal to 55 m, and, for distances greater than d_{bp}, the path-loss exponent was 4 dB. We set the maximum communication range to $R_c = 63$ m. For these experiments, the maximum speed (i.e., $V_{\max}$) was set to 2 m/s. Table 1 lists the coordinates of each network node and Figure 15 marks these positions on a map with symbol △. Table 2 lists the set of positions occupied by the mobile node when it traversed the campus (acquired by GPS); they are shown as black circles in Figure 15. The mobile node briefly stood still at these positions during distance estimations to nearby network nodes.

Table 1: Locations of network nodes.

Network node	Real[m]	Left-side [m]	Right-side [m]
n_1	(62,32)	(63,39)	(70,76)
n_2	(91,15)	(105,19)	(117,78)
n_3	(128,10)	(144,13)	(155,69)
n_4	(151,32)	(143,23)	(150,60)
n_5	(156,54)	(161,55)	(155,23)
n_6	(170,67)	(178,59)	(163,8)
n_7	(242,61)	(229,40)	(229,113)
n_8	(293,164)	(290,144)	(332,91)
n_9	(342,110)	(337,96)	(230,103)
n_{10}	(336,138)	(331,125)	(323,135)

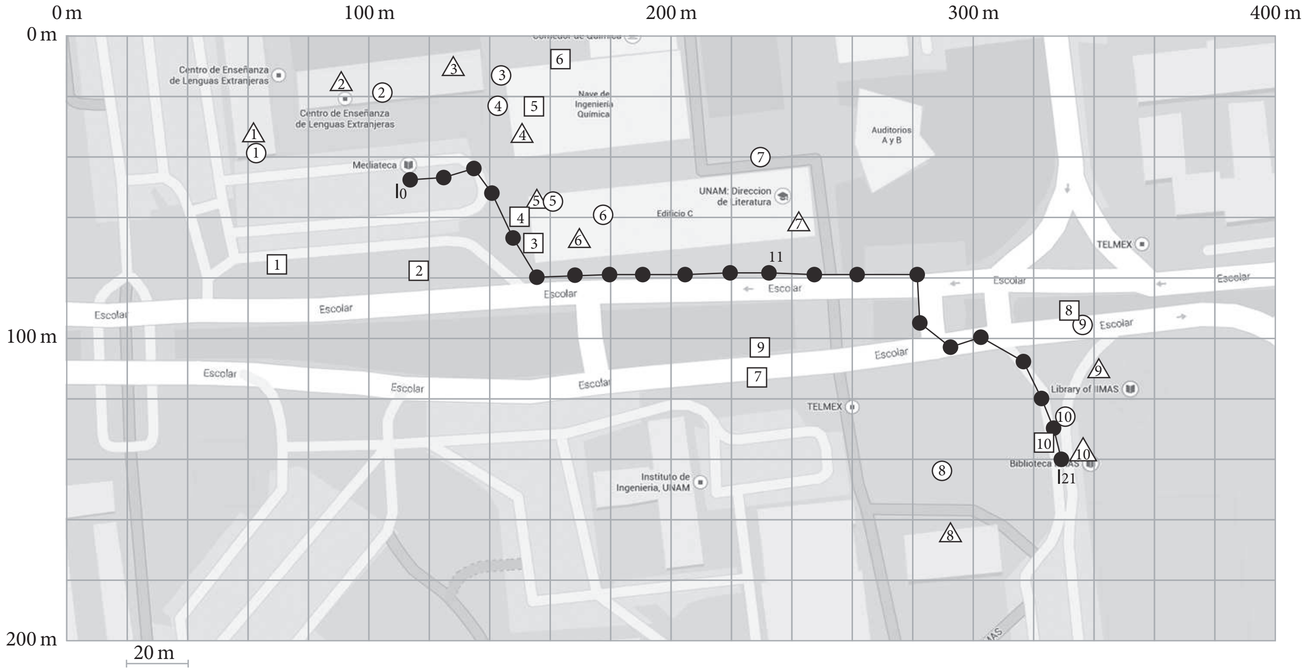

0 m
100 m
200 m
300 m
400 m
0 m
100 m
200 m
20 m
Centro de Enseñanza de Lenguas Extranjeras
Centro de Enseñanza de Lenguas Extranjeras
Mediateca
Nave de Ingeniería Química
Auditorios A y B
UNAM: Dirección de Literatura
Edificio C
TELMEX
Escolar
Library of IMAS
Biblioteca IMAS
Instituto de Ingeniería, UNAM
TELMEX
0
11
21
1
2
3
4
5
6
7
8
9
10

Figure 15: A real scenario in a university campus. The origin of the coordinate system is located at the upper left corner (lat.: 19°19′54.2″N; long.: 99°11′2″W).

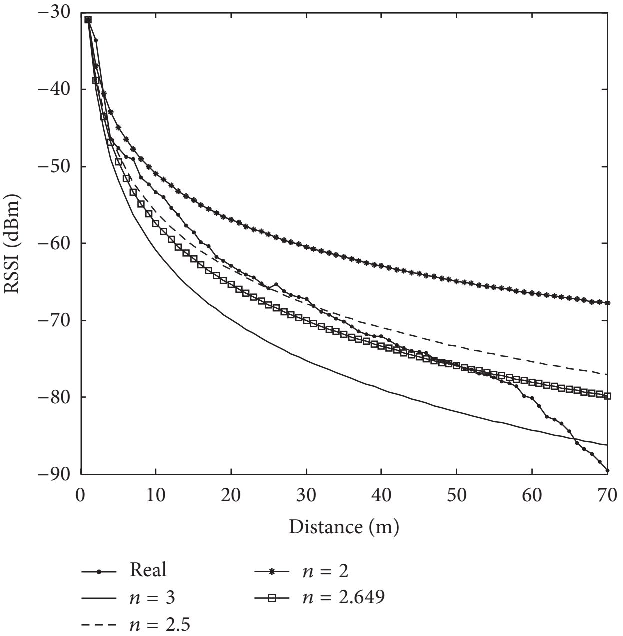


Figure 16: Received signal power versus distance.

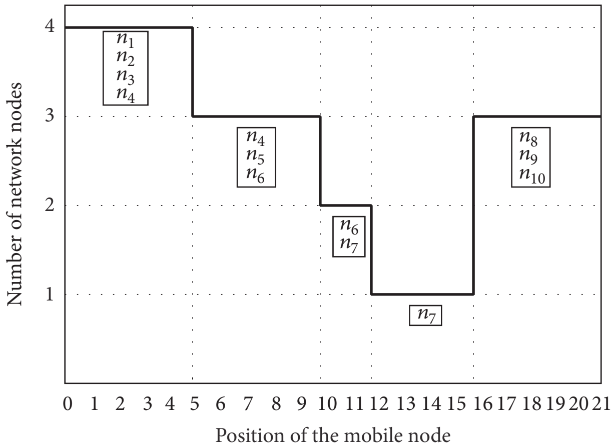
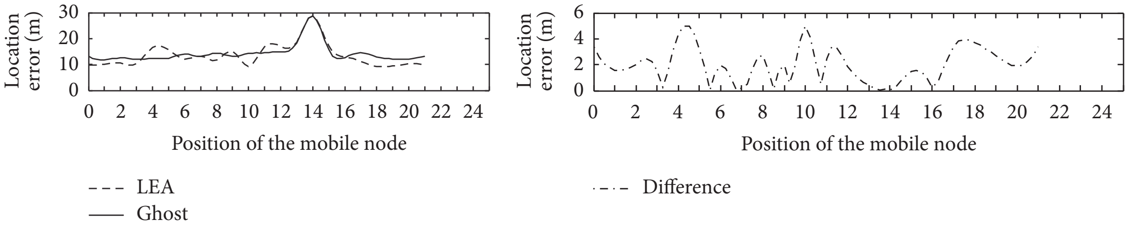


Figure 17: Number of network nodes detecting the mobile node.

We implemented LEA in a smartphone running Android with its Wi-Fi interface configured to operate in monitor mode in order to overhear traffic transmitted by nearby network nodes. In this implementation, we used 30 virtual nodes. By using distance estimates between the mobile node and network nodes, it was possible to compute the left-side position and right-side position estimates for each network node; they are shown in Figure 15 with symbols ○ and □, respectively.

Figure 17 illustrates the number of network nodes detecting the mobile node as it crossed the university campus, where n_i denotes the ith network node detecting the mobile node. Figure 18(a) shows the location error computed by LEA and Ghost for different positions of the mobile node during the experiment. In this figure, it is easy to see that the location error computed by LEA is similar to the location error computed by Ghost most of the time. Figure 18(b) shows the absolute difference in location error between LEA and Ghost. It is seen that such difference fluctuated between 0 m and 5 m. Such difference in location is mainly due to



(a)

(b)

Figure 18: Experiments in a real scenario. (a) Mean location error achieved by LEA and Ghost; (b) absolute error difference between LEA and GHOST.

Table 2: Locations of the mobile node.

Locations of the mobile node	Coordinates [m]
l_0	(114,48)
l_1	(125,47)
l_2	(135,44)
l_3	(141,52)
l_4	(148,67)
l_5	(156,80)
l_6	(167,80)
l_7	(180,79)
l_8	(191,79)
l_9	(205,79)
l_{10}	(220,78)
l_{11}	(233,78)
l_{12}	(248,79)
l_{13}	(262,79)
l_{14}	(282,79)
l_{15}	(283,95)
l_{16}	(293,103)
l_{17}	(303,100)
l_{18}	(317,108)
l_{19}	(323,120)
l_{20}	(327,130)
l_{21}	(328,140)

propagation impairments created by multipath reflection and shadowing effects as well as some limitations of LEA already mentioned. However, we consider that such a small error is acceptable for many practical applications.

6. Conclusion

In this work, we introduce LEA, a novel algorithm that allows a mobile node to estimate how accurately a network determines its current location without intervention from the network. LEA requires the mobile node to detect and place nearby network nodes without previous knowledge of the network topology. It uses this information to emulate a localization algorithm and estimate a level of exposure. We studied various scenarios considering not only the advantageous case when three or more network nodes are able to detect the presence of the mobile node but also scenarios when only one or two network nodes are within the transmission range of the mobile node. Simulations and testbed experiments show that LEA is able to reproduce the location computed by the network with a maximum deviation with respect to the true location of about 5 m for IEEE 802.11 networks. LEA achieves its goal before the mobile node transmits a packet to the network, so that future applications can take appropriate actions depending on the specific situation. It is worth emphasizing that the network might have more powerful tools to locate a mobile node in comparison to a mobile node; therefore LEA does not pretend to achieve the same accuracy as the network. However, we believe that it can become a useful tool to allow a mobile node to be aware of its level of exposure. To the best of the authors' knowledge, this is the first piece of research dealing with the location-exposure problem.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by research funds from CONACyT (105117/105279), DGAPA-PAPIIT (IN108910/IN117017), and PAPIME PE 103807.

References

[1] T. Shu, Y. Chen, J. Yang, and A. Williams, "Multi-lateral privacy-preserving localization in pervasive environments," in *Proceedings of the 33rd IEEE Conference on Computer Communications (IEEE INFOCOM '14)*, pp. 2319–2327, Ontario, Canada, May 2014.

[2] A. Mohaisen, D. Hong, and D. Nyang, "Privacy in location based services: primitives toward the solution," in *Proceedings of the 4th International Conference on Networked Computing and Advanced Information Management (NCM '08)*, pp. 572–579, IEEE, September 2008.

[3] A. Martinez-Balleste, P. Perez-Martinez, and A. Solanas, "The pursuit of citizens' privacy: a privacy-aware smart city is possible," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 136–141, 2013.

[4] G. Han, H. Xu, T. Q. Duong, J. Jiang, and T. Hara, "Localization algorithms of wireless sensor networks: a survey," *Telecommunication Systems*, vol. 52, no. 4, pp. 2419–2436, 2013.

[5] F. Garcia, J. Gomez, M. A. Gonzalez, M. Lopez-Guerrero, and V. Rangel, "Ghost: Voronoi-based tracking in sparse wireless networks using virtual nodes," *Telecommunication Systems*, vol. 61, no. 2, pp. 387–401, 2016.

[6] P.-H. Tseng, K.-T. Feng, Y.-C. Lin, and C.-L. Chen, "Wireless location tracking algorithms for environments with insufficient signal sources," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, pp. 1676–1689, 2009.

[7] L. Ma, J. Liu, L. Sun, and O. B. Karimi, "The trajectory exposure problem in location-aware mobile networking," in *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '11)*, pp. 7–12, Valencia, Spain, October 2011.

[8] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Enhancing security and privacy in traffic-monitoring systems," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 38–46, 2006.

[9] E. Elnahrawy, X. Li, and R. P. Martin, "The limits of localization using signal strength: a comparative study," in *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 406–414, Santa Clara, Calif, USA, October 2004.

[10] D. P. Mehta, M. A. Lopez, and L. Lin, "Optimal coverage paths in ad-hoc sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '03)*, vol. 1, pp. 507–511, May 2003.

[11] P. K. Sahoo and W.-C. Liao, "HORA: a distributed coverage hole repair algorithm for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1397–1410, 2015.

[12] Z. Yang and Y. Liu, "Quality of trilateration: confidence-based iterative localization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 631–640, 2010.

[13] M. L. Damiani and C. Cuijpers, "Privacy challenges in third-party location services," in *Proceedings of the 14th International Conference on Mobile Data Management (MDM '13)*, vol. 2, pp. 63–66, Milan, Italy, June 2013.

[14] K. Vu and R. Zheng, "Robust coverage under uncertainty in wireless sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '11)*, pp. 2015–2023, April 2011.

[15] M. P. Johnson, D. Sariöz, A. Bar-Noy, T. Brown, D. Verma, and C. W. Wu, "More is more: the benefits of denser sensor deployment," in *Proceedings of the 28th Conference on Computer Communications (IEEE INFOCOM '09)*, pp. 379–387, Rio de Janeiro, Brazil, April 2009.

[16] K.-P. Shih, C.-C. Li, and Y.-D. Chen, "An intruder avoidance vulnerable path adjustment protocol for wireless mobile sensor networks," in *Proceedings of the Joint Conferences on Pervasive Computing (JCPC '09)*, pp. 125–130, December 2009.

[17] V. Phipatanasuphorn and P. Ramanathan, "Vulnerability of sensor networks to unauthorized traversal and monitoring," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 364–369, 2004.

[18] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp. 139–150, Rome, Italy, July 2001.

[19] W. Zhang and M. Li, "Anti-detection: how does a target traverse a sensing field," in *Proceedings of the 2nd International Conference on Embedded Software and Systems*, Xian, China, December 2005.

[20] M. Pascoe-Chalke, J. Gomez-Castellanos, V. Bonilla-Gonzalez, M. Lopez-Guerrero, V. Rangel-Licea, and E. Rodriguez-Colina, "Direction of Encounter (DoE): a mobility-based location method for wireless networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 11, pp. 2524–2536, 2014.

[21] M. Sharif, S. Khan, S. J. Khan, and M. Raza, "An algorithm to find convex hull based on binary tree," in *Proceedings of the IEEE 13th International Multitopic Conference (INMIC '09)*, pp. 1–6, Islamabad, Pakistan, 2009.

[22] J. J. Caffery Jr., "New approach to the geometry of TOA location," in *Proceedings of the 52nd Vehicular Technology Conference (VTC '00)*, pp. 1943–1949, Boston, Mass, USA, September 2000.

[23] X. Li and D. K. Hunter, "Probabilistic model of triangulation," in *Proceedings of the ACM Symposium on Solid and Physical Modeling 2008 (SPM '08)*, pp. 301–306, New York, NY, USA, June 2008.

[24] T. M. Chan, "Point location in o(log n) time, Voronoi diagrams in o(n log n) time, and other transdichotomous results in computational geometry," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06)*, pp. 333–342, Berkeley, Calif, USA, October 2006.

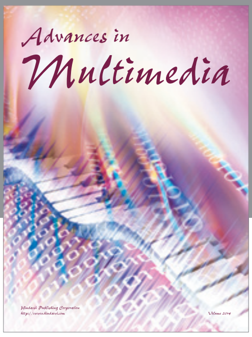
[25] N. K. Sharma, "A weighted center of mass based trilateration approach for locating wireless devices in indoor environment," in *Proceedings of the ACM International Workshop on Mobility Management and Wireless Access (MobiWAC '06)*, pp. 112–115, Terromolinos, Spain, October 2006.

[26] A. Z. B. H. Talib, M. Chen, and P. Townsend, "Three major extensions to Kirkpatrick's point location algorithm," in *Proceedings of the 14th International Conference of the Computer Graphics Society (CGI '96)*, pp. 112–121, June 1996.

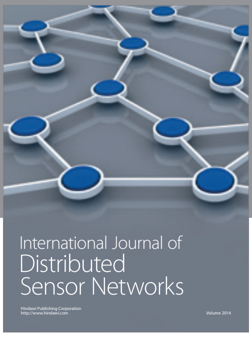
[27] P.-C. Chen, "A non-line-of-sight error mitigation algorithm in location estimation," in *Proceedings of the Wireless Communications and Networking Conference (WCNC '99)*, vol. 1, pp. 316–320, IEEE, New Orleans, La, USA, September 1999.

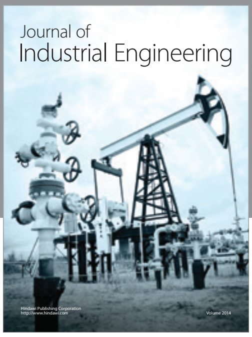
[28] B. Roberts and K. Pahlavan, "Site-specific RSS signature modeling for WiFi localization," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09)*, Honolulu, Hawaii, USA, December 2009.

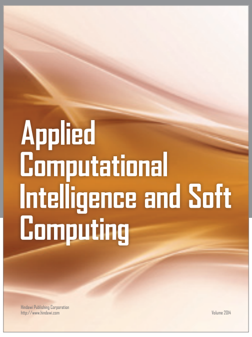
[29] J. M. Castro-Arvizu, P. Closas, and J. A. Fernandez-Rubio, "Cramer-Rao lower bound for breakpoint distance estimation in a path-loss model," in *Proceedings of the IEEE International Conference on Communications Workshops (ICC '14)*, pp. 176–180, June 2014.

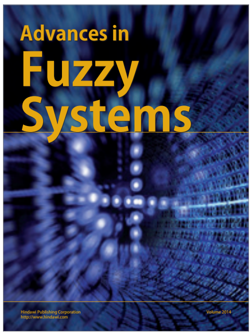

Advances in
Multimedia

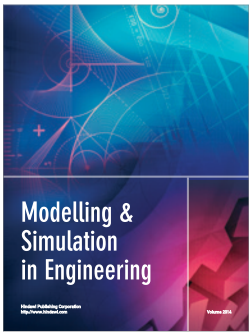
The Scientific
World Journal

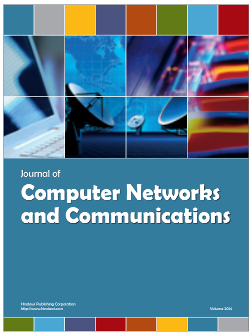

International Journal of
Distributed
Sensor Networks

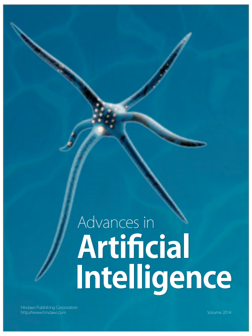

Journal of
Industrial Engineering

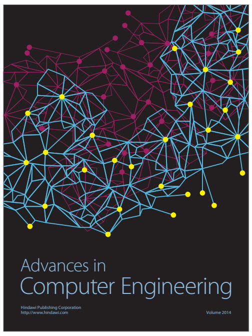

Applied
Computational
Intelligence and Soft
Computing


Advances in
Fuzzy
Systems

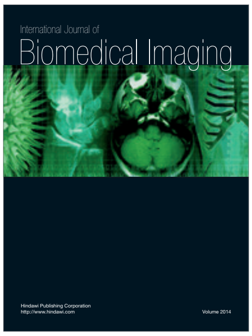

Modelling &
Simulation
in Engineering

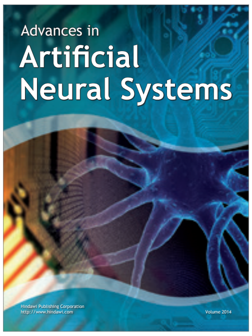

Journal of
Computer Networks
and Communications

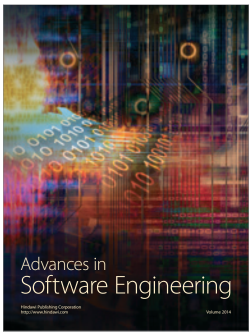

Advances in
Artificial
Intelligence


Advances in
Computer Engineering

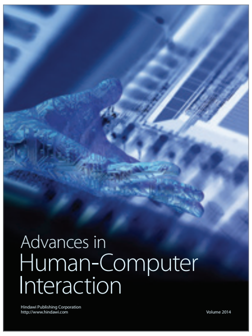
International Journal of
Computer Games
Technology

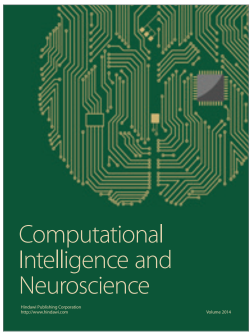

International Journal of
Biomedical Imaging


Advances in
Artificial
Neural Systems


Advances in
Software Engineering

Journal of
Robotics


Advances in
Human-Computer
Interaction


Computational
Intelligence and
Neuroscience

International Journal of
Reconfigurable
Computing

Journal of
Electrical and Computer
Engineering