Component-Based Localization in Sparse Wireless Networks

Xiaoping Wang, Jun Luo, Yunhao Liu, *Senior Member, IEEE*, Shanshan Li, and Dezun Dong

***Abstract*—Localization is crucial for wireless ad hoc and sensor networks. As the distance-measurement ranges are often less than the communication ranges for many ranging systems, most communication-dense wireless networks are localization-sparse. Consequently, existing algorithms fail to provide accurate localization supports. In order to address this issue, by introducing the concept of *component*, we group nodes into components so that nodes are able to better share ranging and anchor knowledge. Operating on the granularity of components, our design, CALL, relaxes two essential restrictions in localization: the node ordering and the anchor distribution. Compared to previous designs, CALL is proven to be able to locate the same number of nodes using the least information. We evaluate the effectiveness of CALL through extensive simulations. The results show that CALL locates 90% nodes in a network with average degree 7.5 and 5% anchors, which outperforms the state-of-the-art design Sweeps by about 40%.**

***Index Terms*—Component-based, finite mergence, localization, node-based, ranging-model-based estimation (RMBE).**

I. Introduction

LOCATION in wireless networks is critical for both network operation and data interpretation [1]. Practically, it is difficult to equip each node with a positioning device. Instead, only a few nodes, called *anchors*, know their locations. Other nodes estimate their locations through internode measurements from the anchors. Most existing localization algorithms require the network have a high density and sufficient number of anchors before computing node locations. Localization in sparse networks with a few anchors is not fully addressed [2]. Indeed, a sparse network for localization is often dense in communication, as the distance-measurement ranges are typically much less than that of communication range for many ranging systems [3].

Eren *et al.* [4], [5] investigate the theoretical conditions for localization in general networks. They show that a network can be uniquely localized if and only if its corresponding grounded graph [4] fulfills the following three conditions: 1) redundantly rigid; 2) triconnected; and 3) having three anchors embedded, denoted as RRT-3B [5]. For a partially localizable network, RRT-3B can identify most localizable nodes by recursively partitioning the network into globally rigid parts [5]. Thus, the RRT-3B becomes a criterion to evaluate the performance of localization algorithms. To the best of our knowledge, there is no scheme that can explicitly achieve the amount of localizable nodes under RRT-3B conditions.

The state-of-the-art algorithm of approaching to the capacity of RRT-3B is Sweeps [2]. Sweeps utilizes the concept of finite localization to relax the node participating conditions from trilateration to bilateration. Due to its dependence on each single node to estimate the localizability locally, however, Sweeps is subject to the restriction on the anchor distribution. If no node can find at least two anchors among its direct neighborhood, the algorithm cannot be initialized. Also, Sweeps requires that the network have a bilateration ordering, which is not always true in sparse networks. Indeed, a localizable network is not necessarily to be a bilateration network, thus Sweeps often fails in many actually localizable networks [2], [6]. Such limitations result in more serious consequences in localization-sparse networks.

To address this issue, we propose a Component-bAsed Localization aLgorithm, CALL, in which the component acts as another basic unit for localization. Instead of locating each individual node separately, CALL groups nodes into small components, and then merges the components iteratively into a bigger one until it is realizable in a plane. By offering many advantages over node-based approaches, such as favoring the large-scale localization information aggregation, facilely exploiting the anchor proximity and underlying network model, CALL can locate most localizable nodes as RRT-3B identifies. Furthermore, if more information is available about the ranging model, such as the geometric representations of the measurement range, CALL can even approach the RRT-3B upper limitation and locate more nodes.

Major contributions of this work are as follows.

1) We introduce the concept of component and present a component-based algorithm, CALL, which improves the success ratio of localization in sparse networks.
2) We propose a mechanism called *finite mergence* to stitch components. Furthermore, we form the basic anchor requirement conditions to realize components. These mechanisms serve as the basis of this design.
3) By exploiting the ranging-model-based estimation (RMBE) schemes, we reduce the states maintained at each node and make more nodes localizable, i.e., 20% in average, even though some of them do not follow the RRT-3B condition.
4) We propose a new metric, *least information requirement* (LIR), to analyze the performance of existing localization algorithms. We show our CALL achieves the best result as yet under the LIR criterion. Large-scale simulations are

Manuscript received January 14, 2009; revised October 12, 2009 and July 26, 2010; accepted August 17, 2010; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Liu. Date of publication September 27, 2010; date of current version April 15, 2011. This work was supported in part by NSFC/RGC Joint Research Scheme N_HKUST602/08, the National Basic Research Program of China (973 Program) under Grants 2011CB302705 and 2010CB328004, and NSF China under Grants 60970118 and 60903224.

X. Wang, J. Luo, S. Li, and D. Dong are with the School of Computer Science, National University of Defense Technology, Changsha 410073, China (e-mail: xiaopingwang@nudt.edu.cn; junluo@nudt.edu.cn; shanshanli@nudt.edu.cn; dong@nudt.edu.cn).

Y. Liu is currently with the Tsinghua University National Laboratory for Information Science and Technology (TNLIST), School of Software, Tsinghua University, Beijing 100084, China (e-mail: yunhaoliu@gmail.com).

Digital Object Identifier 10.1109/TNET.2010.2072965

conducted to examine the performance of CALL. The results show that this algorithm can accurately locate 90% nodes in a network with average degree 7.5 by 5% anchors, which outperforms the state-of-the-art design Sweeps by about 40%.

The rest of this paper is organized as follows. We discuss existing studies in Section II and describe the details of CALL scheme in Section III. We present the theoretical foundations in Section IV and formally analyze its performance in Section V. We evaluate the performance of CALL in Section VI and conclude the work in Section VII.

II. Related Work

Many studies have focused on localization in wireless and ad hoc networks [7], mainly falling into two categories: range-based [2], [8] and range-free [1], [11]. Range-free designs do not rely on measurement hardware, but normally require high network density [1]. This work focuses more on range-based designs for sparse networks, so we discuss the classical range-based algorithms that address the network or anchor sparseness problem.

Savarese *et al.* [12] propose a virtual-coordinate-based algorithm TERRAIN to address the sparse anchor problem. TERRAIN constructs a virtual coordinate system on each anchor and takes the advantage of the property that the virtual coordinate holds the distance information between each node pair. The essential principle used by TERRAIN is trilateration. By using virtual coordinates on each anchor, TERRAIN extends the ranging distance of anchors and makes each node triangulate to the enlarged anchors.

Some researchers utilize local maps to localize nonanchor nodes [13], [14]. They first use distance measurements between neighboring nodes to construct local maps, and then stitch them together to form a global map. Components in our design share similar notions with local maps. Nevertheless, our design is different with them in terms of mergence and realization. In local-map-based methods, two maps are combined by their common node, which means the common node locations need to be known in their respective maps. Hence, it is difficult to stitch local maps in sparse networks due to the lack of known location common nodes. In our design, components are merged by the interconnected edges between the components. We will show that merging by three interconnected edges is the necessary condition for generating a rigid local map. Clearly, in the view of components, merging local maps by common nodes is a special case of merging by the interconnected edges. Such difference significantly affects the applicability of the designs in sparse networks.

Savvides *et al.* [15] attempt to reduce the information requirement when facing partial sparseness. They use collaborative multilateration among neighbors to compensate the ranging information shortage, which localizes nodes by forming an overdetermined system of equations with a unique solution set. Collaborative multilateration performs better than trilateration in sparse networks. The disadvantage is that the collaboration is restricted in neighbors, so that the performance gain is limited.

As a pioneer work, Sweeps [2] introduces the concept of finite localization, which holds all candidate positions of each node and prunes incompatible ones when other nodes join the procedure. Sweeps achieves pretty good results in sparse networks. Given proper anchor distribution, Sweeps is able to locate nodes in a globally rigid region [2], but may fail to localize the regions that contain few anchors. Compared to Sweeps, our CALL is able to identify most of such regions and locate almost all of the localizable ones. Besides relaxing the restrictions of anchor proximity and node ordering, by using finite mergence, the localized nodes of CALL are a superset of that of Sweeps. Also, CALL can exploit the restriction of ranging models. This makes CALL generate much smaller state sets and even be able to locate nodes that do not follow RRT-3B.

III. Component-Based Localization

Before the detailed discussion on our design, we introduce the main idea of component-based localization.

A. Preliminary

For a given network, we generate a distance graph $G = (V, E)$, where vertices denote nodes in the network and an edge (i, j) exists if nodes i, j can measure the mutual distance between them. Associated with each edge, we use a function $d(i,j)\colon E \rightarrow R$ to denote the distance value. Hereinafter, we use the term edge to denote distance measurement. We assume there are m nodes, called *anchors*, knowing their locations in advance. The anchors are labeled from 1 to m, and the left $n-m$ location-unknown nodes are labeled from $m+1$ to n, where n is the total number of nodes in the network. The physical location of node i is denoted by p_i.

A *realization* of a network is a mapping from nodes to coordinates in two-dimensional space, $P\colon V \rightarrow R^2$, such that $P(i) = p_i$ for all $1 \leq i \leq m$ and $\|P(i) - P(j)\| = d(i,j)$ for all $(i,j) \in E$, where $\|P(i) - P(j)\|$ denotes the Euclidean distance between coordinates $P(i)$ and $P(j)$. Analogously, we define the concept of realization on the subgraph of G, and the only difference is that we do not differentiate the rotations, translations, and reflections of the mapping when operating on a subgraph. A node is *localizable* if and only if its image is unique for all realizations of G. A node is *finitely localizable* if and only if the cardinality of its image set is finite for all realizations of G. If a localization algorithm can generate a unique result for a localizable node, we say the node is *localized* by the algorithm. If a localization algorithm can generate a *candidate position set* that contains all of the possible positions of a finitely localizable node, we say the node is *finitely localized* by the algorithm.

We formally define the concept of component as follows. Given a distance graph G, a *component* is a set of nodes that have a finite number of ways to be realized. A component is *globally rigid* if and only if there is a unique realization in a plane. An *isolated node* in G is a node that does not belong to any components. For two components C_1 and C_2, if there is an edge (n_1, n_2) such that $n_1 \in C_1$, $n_2 \in C_2$, edge (n_1, n_2) is an *interconnected edge* between components C_1 and C_2. We also use interconnected edge to denote the edge between a component and an anchor. Compared to traditional local maps, components are essentially different in the way of component mergence and realization. Components are merged by the interconnected edges between the components. Moreover, components are realized through both in-component anchors and the interconnected edges between the component and anchors. Hence, we say a component is *realizable* if it can determine its physical layout by using the anchor information. We will show the

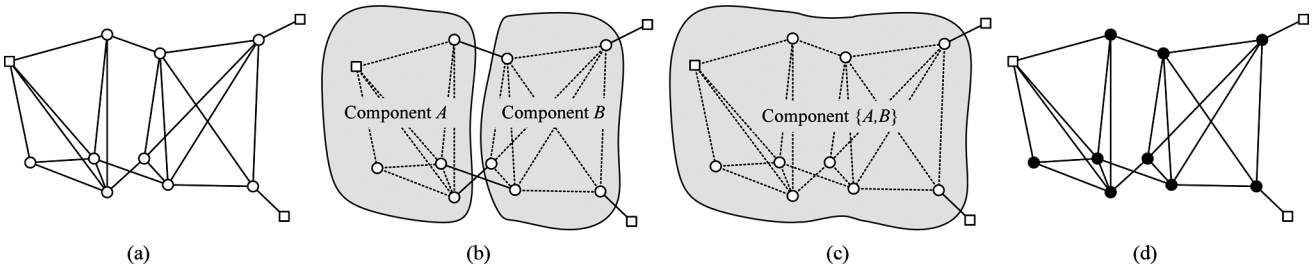

Fig. 1. Example of component-based localization. (a) Distance graph (b) Component generation. (c) Component mergence. (d) Component realization.

basic requirements for component mergence and realization in Section IV.

Before introducing the details of our component-based localization algorithm, we further explain the concept of components using an example. We show the distance graph of a network in Fig. 1(a), where the squares denote anchors and the circles represent nonanchor nodes. If we adopt node-based algorithms, we cannot localize any node in this network, as there is no single node that can obtain enough (at least two) distance measurements to the anchors. In contrast, if we utilize a component-based algorithm, we first partition the network into two globally rigid components, denoted as components A and B in Fig. 1(b). In the view of components, components A and B are not directly realizable, so we merge the two components through the interconnected edges and generate a new component, denoted as $\{A, B\}$, as shown in Fig. 1(c). After such a mergence, component $\{A, B\}$ is realizable through the anchors. As a result, all nodes in component $\{A, B\}$ are localized simultaneously, as shown in Fig. 1(d).

B. BCALL Algorithm

We first present the basic version of this design, BCALL, which terminates in polynomial time. We will then present the advanced version, CALL, which cannot guarantee to terminate in polynomial time, but with better performance.

Both BCALL and CALL have three major operations: *component generation, component mergence,* and *component realization*.

1) Step 1: Component Generation: Component generation partitions the network into globally rigid components and isolated nodes. A node can only join one component, such that components do not share any common nodes. After component generation, each node either belongs to a component or becomes an isolated node. In this step, anchor nodes and nonanchor nodes are equally treated.

Each component has a *local coordinate system* that indicates the relative position of each node in the component. To initialize the local coordinate system, we initially select a component to be a triangle in G, as triangles are the simplest globally rigid graphs. Accordingly, the local coordinate system of a component is generated according to the relative position of the vertices in the initial triangle. Other nodes then join the component and compute their local coordinates through trilateration. By adopting trilateration iteratively, we expand the newly generated component as large as possible while keeping global rigidity.

For each component, if it is realizable, we go to Step 3: component realization. Otherwise, those nonrealizable components will merge with other components or isolated nodes and perform component mergence. Merging components potentially makes nonrealizable components capable to be realized because: 1) the mergence causes the merged components to aggregate both their nodes and their anchor information; and 2) the realization requirements for a component are independent of the number of nodes in the component.

2) Step 2: Component Mergence: Component mergence integrates two components into one by the interconnected edges. BCALL requires the resultant component to be globally rigid, thus it must follow the requirements of Theorem 1 in Section IV. After merging, the local coordinate systems of the two components must be consistent. This is guaranteed by converting the local coordinate system of a component to that of the other one.

Component mergence is an iterative process. Some mergence can make other components or isolated nodes capable of merging into the resultant component. Component mergence process terminates when no such mergence can continue or the resultant component is realizable.

3) Step 3: Component Realization: Component realization converts the local coordinate system of the target component to the physical coordinate system. For BCALL, it requires the realization to be unique, so the anchor information must uniquely determine the physical layout of the component (Corollary 1 in Section IV). In mathematics, a coordinate system conversion is generally denoted by a rotation matrix (with possible reflection) R and a translation vector t. For each node in the target component, component realization directly converts its local coordinate to the physical position by $p_{\rm phi} = R \times p_{\rm loc} + t$, where $p_{\rm phi}$ denotes the physical location of the node and $p_{\rm loc}$ denotes the local coordinate of the node.

C. CALL Algorithm

Compared to BCALL, CALL relaxes the requirements of component mergence and realization. First, CALL does not demand the resultant of mergence to be globally rigid. Second, CALL does not require component realization to be unique. That is, two components can merge if the resultant component has a finite number of ways to be realized, and a component can be realized if it can convert its local coordinates to a finite set of physical positions by the anchor information. These relaxations have several impacts on the localization performance. First, CALL generates more realizable components than BCALL by the relaxed mergence. Second, the relaxation of component realization makes CALL recognize more realizable components than BCALL. Third, potentially, all the finitely localized nodes can be converted to uniquely localized nodes, as CALL will prune the extra candidate positions of the nodes by consistency check in each step.

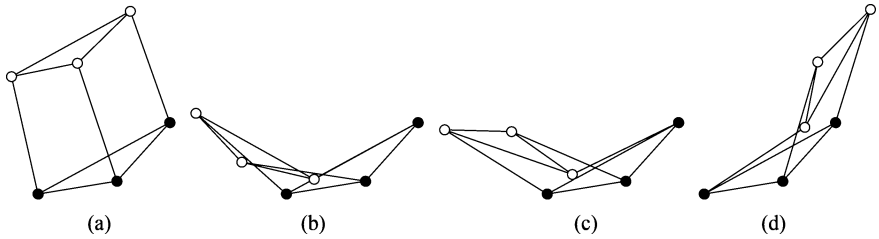

Fig. 2. Exhaustive result set of a component mergence. (a) Physical locations of the target components. (b)–(d) Other results that fulfill the intercomponent distance constraints.

The relaxations cause nodes to have several candidate positions. CALL records these positions in the *potential position set* at each node. The potential position set indicates all candidate coordinates of the node in the local coordinate system or physical coordinate system.

CALL inserts an extra substep in each of the above operations to check the consistency of the potential position sets. During each mergence or realization, CALL enumerates all candidate positions of each node in its potential position set. Not all of the combinations of the candidate positions can generate valid results under the distance constraints. We say an item is *incompatible* if no result can be obtained by this candidate position. After mergence or realization, nodes will prune the incompatible items in the potential position sets. There are two steps to prune incompatible items. First, the reference nodes associated with the interconnected edges can identify their incompatible items after the mergence or realization. Second, if a node has pruned some incompatible items, it will broadcast an update message to its neighbors to inform the deleted items. The node that receives the update message will then check the consistency of the potential position set and prune their incompatible items. By iteratively pruning the incompatible items, the potential position sets of all nodes will eventually be consistent with each other. This mechanism guarantees that the consistency check can prune all the incompatible items of the nodes in the same component. We say a node or a node set is *finalized* if the size of its potential position set is reduced to one.

Finally, all nodes belonging to the realized components are localized. By the items in the potential position set, we can directly determine whether a node is uniquely localized. In Section IV, we will show the rationale of the mergence and the realization.

Unfortunately, this procedure cannot guarantee to terminate in polynomial time because the potential position set may grow exponentially in the worst case.

D. Ranging-Model-Based Estimation

Previous discussion does not require any specific ranging model. In practice, ranging measurement is distance-sensitive: Nodes are often easy to know the distance to nearby neighbors, while difficult for far away ones.

Assumptions on ranging model can further improve the performance of CALL. For example, suppose the ranging model is α-QUDG [16]. That is, two nodes cannot measure the distance if they are more than unit-distance apart, while they can measure the distance if they are less than α distance apart; otherwise, there is a distance-related probability $p(d)$, where d is the normalized distance of the node pair. Define $p(d) = 1$ for $d \leq \alpha$ and $p(d) = 0$ for $d \geq 1$, then we uniformly use $p(d)$ to describe the probability that two nodes can measure their mutual distance. Given a candidate result (i.e., a result of component mergence or realization), we define the *confidence* of a node pair as the probability of whether or not the node pair can measure their distance according to the ranging model. Clearly, if two nodes can measure their mutual distance, they will follow the model, and we do not need to estimate the confidence. Hence, we only check the node pairs that cannot measure the mutual distances. Consequently, if a node pair (i, j) cannot measure the distance, the confidence is $1 - p(d_{i,j})$, where $d_{i,j}$ is the normalized distance of the node pair in the candidate result. Furthermore, we define the *confidence* of a candidate result to be the minimum confidence of the node pairs. For each mergence or realization, we can prune the candidate result if its confidence is below a predefined threshold. We can even set the threshold value to zero, which means that we only cut the results that conflict with the ranging model. We call this enhancement ranging-model-based estimation, or RMBE in short.

Take Fig. 2 as an example, which shows all candidate results of a certain component mergence. This example is generated under the $1/\sqrt{2}$-QUDG model [16], and the original graph is case (a). We evaluate the confidence of all nonneighboring node pairs. Apparently, except for graph (a), the confidence values of the other candidate results are all 0, as there exists such node pairs that are close enough to each other but cannot measure the mutual distance in graphs (b)–(d). In other words, they conflict with the ranging model, and we can prune these candidate results. In this example, the mergence is finalized by RMBE.

Given a proper ranging model, RMBE can significantly reduce the size of the potential position set at each node. Also, RMBE can help to finalize some finitely localized nodes. As a result, RMBE can improve the performance of localization algorithms. Simulations in Section VI will show this as well.

IV. Theoretical Foundations

In this section, we introduce the rationale of the rules for component mergence and component realization. The proof of the propositions is shown in [17].

Lemma 1: Two globally rigid components can merge into one if: 1) there are at least three edges connecting the two components; and 2) there are at least two vertices in each component associated with these edges.

Theorem 1 (Mergence for BCALL): If two globally rigid components can merge into one globally rigid component, there are at least four edges connecting the two components, and there are at least three vertices in each component associated with these edges.

Theorem 2 (Finite Mergence for CALL): Two components can merge into one if: 1) there are at least three edges connecting the two components; and 2) there are at least two vertices in each component associated with these edges.

Theorem 3 (Finite Mergence for Isolated Nodes): A node can be merged into a component by two edges connecting to the component.

Theorem 4 (Realization for CALL): A component can be realized to finite states by fulfilling at least one of following conditions.

a) The component contains at least two anchors.
b) The component contains one anchor and a nonanchor node sharing an edge with a realized node.
c) There are at least three edges connecting the component with at least two distinct realized nodes, and there are at least two vertices associated with these edges in the component.

Corollary 1 (Realization for BCALL): A globally rigid component can be uniquely realized by containing three distinct vertices fulfilling one of the following conditions.

a) They are three anchors.
b) They are two anchors and a nonanchor node sharing an edge with a realized node.
c) They are one anchor and two distinct nonanchor nodes sharing two edges with two distinct realized nodes.
d) There are at least four independent edges connecting them with at least three distinct realized nodes.

V. Performance Analyses

In this section, we analyze the performance of CALL, including the performance of component generation and the overall algorithm.

A. Efficiency of Component Generation

As previously mentioned, all the initially generated components are globally rigid. We need to examine whether CALL can identify all the globally rigid parts in the component generation step. Indeed, to identify all the globally rigid parts in a graph is computationally intensive [5]. CALL minimizes this cost by identifying the globally rigid parts with triangles only.

Lemma 2: If a graph contains no triangles, the maximum number of edges it contains is $\lfloor n^2/4 \rfloor$, where n is the number of nodes it contains.

Proof: Let G_n denote a graph with n nodes that contains no triangles; $g(n)$ denotes the maximum number of edges in G_n. We use $d_{\min}(n)$ to denote the minimum degree of the nodes in G_n.

If we delete the node with minimum degree in G_n, the generated graph will not contain triangles. Thus, the edges in this graph cannot be greater than the value $g(n-1)$, so we have $g(n) - d_{\min}(n) \le g(n-1)$, which means the bound of $g(n)$ is given by

$$g(n) \le \sum_{i=1}^{n} d_{\min}(i) + g(0).$$

Now we show that $d_{\min}(n) \le n/2$. If we assume $d_{\min}(n) > n/2$, the degree of each node in G_n is beyond $n/2$. Consider an arbitrary edge in G_n, the nodes associated with the edge have more than n neighbors. Hence, at least two of them are the same node. Then, we find a triangle in G_n. This conflicts with the given condition. We rewrite the expression of $d_{\min}(n) \le n/2$ as follows:

$$d_{\min}(n) \le \begin{cases} k, & n = 2k \\ k, & n = 2k+1 \end{cases}.$$

Applying this result in the previous inequality, we have $g(n) \le \lfloor n^2/4 \rfloor$. ∎

Theorem 5: If a graph is globally rigid, it must contain at least one triangle when the total number of nodes is below seven.

Proof: Supposing a graph with n vertices, it contains no triangles. The n vertices need at least $2n-3$ edges to be rigid [4], so the graph needs at least $2n-2$ edges to be redundantly rigid [4]. By Lemma 2, the number of edges is not more than $n^2/4$, so we have $2n-2 \le n^2/4$. Solving this inequality, we get $n \ge 4 + 2\sqrt{2}$. The minimum integer that fulfills the inequality is $n = 7$. ∎

The bound is tight in general graphs. Consider the example of $K_{3,4}$, which is globally rigid and contains no triangle. For the QUDG graph, we get the following theorem.

Theorem 6: If a α-QUDG graph is globally rigid, it must contain a triangle when $\alpha \ge (2 - 2\sqrt{3}\pi/9)^{1/2}$.

Proof: Let G_n denote a graph with n nodes. Take these n nodes as n disks with radius 1. We investigate the total area of these disks, denoted by S_n.

If G_n is globally rigid, it must contain at least $2n-2$ edges. Each edge in G_n superposes two of the disks and will diminish the total area at least $2\pi/3 - \sqrt{3}/2$, which is the overlapped area of two unit circles with distance 1. Hence, we get

$$S_n \le n\pi - (2n-2)\left(\frac{2\pi}{3} - \frac{\sqrt{3}}{2}\right).$$

Simplifying this inequality, we obtain

$$S_n \le \left(\sqrt{3} - \frac{\pi}{3}\right)n + \frac{4\pi}{3} - \sqrt{3}.$$

We use $S_{\max}(n)$ to represent this upper bound.

If G_n contains no triangles, we can compactly dispose the nodes as illustrated in Fig. 3. If we further compress the area, it must form triangles in the graph. Hence, this is the lower bound of S_n. Clearly, in this case, the region surrounded by the dashed line is covered by the disks. We use the area of this region to estimate the total area. We get

$$S_n > \frac{\sqrt{3}\alpha^2}{2}\left(n + \sqrt{3}\sqrt{n - \frac{1}{4}} + \frac{1}{2}\right).$$

We use $S_{\min}(n)$ to represent this lower bound.

If no such S_n can fulfill the two inequalities simultaneously, G_n must contain at least a triangle. Comparing the coefficient of n, we obtain the condition

$$\sqrt{3}\alpha^2/2 \ge \sqrt{3} - \pi/3.$$

Simplifying this inequality, we get

$$\alpha \ge \left(2 - 2\sqrt{3}\pi/9\right)^{1/2}.$$

Now we verify the initial values. By Theorem 5, we only need to consider the case $n \ge 7$. If $n = 7$, then $S_{\max}(7) \approx 7.3$, $S_{\min}(7) > 8.2$. The initial values fulfill our demand.

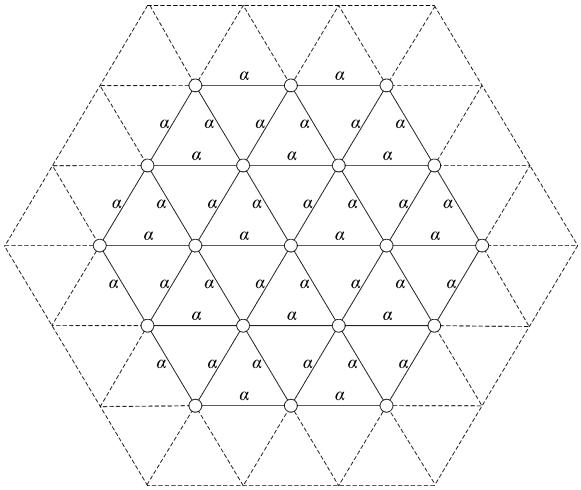

Fig. 3. Compact disposition for minimum area.

Hence, under the condition $\alpha \geq (2 - 2\sqrt{3}\pi/9)^{1/2}$, if a α-QUDG graph is globally rigid, it must contain at least one triangle. ■

B. Performance of Localization Algorithms

To better analyze the performance of localization algorithms, we introduce a heuristic metric, the LIR, for localizing a certain number of nonanchor nodes in generic graphs [5]. Traditionally, localization algorithms work on the given information and compute the positions of a certain amount of nonanchor nodes. The performance of a localization algorithm often refers to the percentage of nonanchor nodes they localized. In LIR, however, algorithms are given a set of nonanchor nodes, and we compare their performance by how much information they require in the ideal case to properly localize the given node set. Consequently, LIR indicates the best an algorithm can do under a set of distance measurements. In addition, as the theoretical results also focus on the least requirements, LIR provides a framework for contrasting the performance of localization algorithms and the theoretical results.

Proposition 1: Trilateration-based algorithms require at least $3n$ edges to properly locate n nodes.

Proof: Each nonanchor node requires three edges in trilateration, so that the least edges required for n nodes are $3n$. ■

Proposition 2: Collaborative multilateration requires at least $\lceil 5n/2 \rceil$ edges to properly locate n nodes.

Proof: Collaborative multilateration can proper locate two nodes by five edges, so the least edges required for n nodes are $\lceil 5n/2 \rceil$. ■

Proposition 3: BCALL requires at least $\lceil 7n/3 \rceil$ edges to properly locate n nodes.

Proof: BCALL can realize a component by four edges, and three edges are needed to form a component. Hence, it needs seven edges to proper localize three nodes, so the least edges required for n nodes are $\lceil 7n/3 \rceil$. ■

Proposition 4: RRT-3B requires at least $2n+1$ edges to make n nodes localizable.

Proof: Supposing localizing a network with n nonanchor nodes and three anchors. The $n + 3$ nodes need at least $2(n + 3) - 3$ edges to be rigid [4], thus it needs at least $2n + 4$ edges to be redundantly rigid [4]. These $2n + 4$ edges contain three edges between each anchor pair, so the least edges needed are $2n + 1$. ■

Proposition 5: Sweeps [2] requires at least $2n + 1$ edges to properly locate n nodes.

Proof: Supposing localizing a network with n nodes, each vertex is swept by at least $2n$ edges [2], and an additional edge is demanded to finalize the result. Hence, the least edges required for n nodes are $2n + 1$. ■

Proposition 6: CALL requires at least $2n + 1$ edges to properly locate n nodes.

Proof: CALL can realize a component finitely by three edges and three edges are required to form a component. Thus it needs six edges to finitely locate three nodes, and an additional edge is necessary to finalize the result. As a result, the least edges required for n nodes are $2n + 1$. ■

Proposition 7: CALL with RMBE requires at least $2n$ edges to properly locate n nodes.

Proof: As shown in Fig. 2, CALL with RMBE can locate six nodes by 12 edges under $1/\sqrt{2}$-QUDG model. For a general n, we can construct a graph by duplicating this case. ■

Proposition 8: CALL can locate a superset of nodes when compared to Sweeps.

Proof: CALL can operate on both components and nodes, and Sweeps can be treated as an extreme case of CALL that realizes one node on each step. Thus, all nodes located by Sweeps are also localizable by CALL. ■

Note that the inverse proposition is not true. For the example shown in Fig. 1, Sweeps fails on this graph because no nonanchor node can find two swept nodes [2].

To summarize, LIR is an effective metric for evaluating the performance of localization algorithms. A tiny gap of LIR value often means prominent performance disparity in sparse networks. We will further examine this property in Section VI, where CALL with RMBE gets significantly improved compared toprevious designs.

VI. Performance Evaluation

In this section, we evaluate the performance of CALL by extensive simulations.

A. Experiment Setup

We compare CALL design to several state-of-the-art algorithms: Sweeps [2], RRT-3B [5], MDS [13], [18], and APIT [19], [20]. Sweeps is widely accepted as the best localization algorithm in sparse networks, significantly outperforming other trilateration-based algorithms [2]. RRT-3B provides a sufficient and necessary condition to determine the localizability of networks, so it actually gives a general criterion for all algorithms to compare their localization capability [2], [5]. MDS and APIT are also widely recognized as practical and effective localization algorithms.

For the convenience of performance analysis later, we here briefly discuss the main ideas of MDS, APIT, and our simulation methodology. MDS can build the relative positions of nodes through a complete matrix of the internode distances. As the granularity of the node set is arbitrarily selected, MDS can be implemented in a centralized [18] or distributed [13] manner. In our simulations, we adopt MDS to build 1-hop local maps and localize the network by the local-map stitching to show the performance gain of using components instead of traditional local

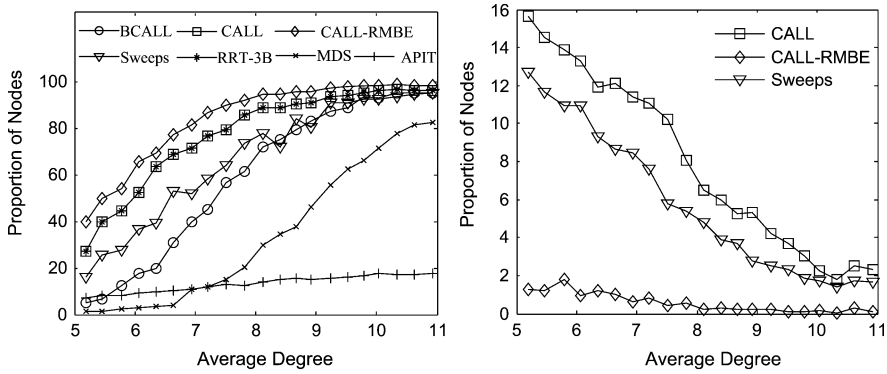

Fig. 4. Proportion of localized nodes against average degree. (a) Uniquely located nodes. (b) Finitely located nodes.

maps. APIT adopts point-in-triangulation test (PIT) to identify whether a node is inside the triangle composed by three neighboring anchors [19], [20]. It locates a node into a region by aggregating the result of PITs, and takes the center of the region as the estimated position of the node. As APIT requires a node referring to several anchors simultaneously, we enlarge the measurement range of anchors three times for APIT. Furthermore, we define that a node is localized by APIT if: 1) it has at least three nonanchor neighbors to accomplish each PIT; and 2) it obtains at least one "in-triangle" result among the PITs.

We compare the four algorithms with all the three versions of our algorithms: BCALL, CALL, and CALL-RMBE. BCALL adopts unique mergence and realization only, thus it can terminate in polynomial time. CALL performs finite mergence and realization to maintain all candidate positions of the nodes. CALL with RMBE, denoted by CALL-RMBE, extends CALL by utilizing extra information of the ranging models. The confidence threshold for RMBE is set to 0, thus we only prune the results that conflict with the ranging model, i.e., the UDG model.

We generate uniformly random distributed networks of 200 nodes in a square region with diversified average degree. We select a certain percentage of nodes as anchors, thus the anchor distribution is also uniformly random. The average degree is controlled through the distance measurement range, and we further assume that the communication range is larger than the distance measurement range. Hence, if a node pair could measure the mutual distance, they can communicate with each other. We repeat each experiment on 100 network instances and report the average.

B. Localization Performance Against Network Density

In this experiment, we randomly distribute 200 nodes in a square region and randomly select 5% of them as anchors. We adopt an empirical formula to control the average degrees of the network instances linearly. Then, we sort the results by the average degree and report the average in the batch of 100 network instances. Fig. 4 reports the result, where the average degree varies from 5 to 11 with 21 steps.

Fig. 4(a) shows the proportion of uniquely localized nodes when the average degree enlarges. As analyzed in Section V, all the algorithms perform in the reversed order of the LIR values. The performance of distributed MDS indicates how a traditional local-map-based algorithm performs in sparse networks. Hence, the performance gap between MDS and BCALL shows the gain of using components instead of traditional local-maps. Also, for the nonpolynomial time complexity algorithms, CALL and CALL-RMBE outperform the state-of-the-art design, Sweeps, over all the tested ranges. Moreover, when the average degree is beyond 9, the advantage of nonpolynomial algorithms becomes trivial. This shows the boundary of whether it is necessary to trade the computational cost for localization performance. APIT performs poorly in all cases because APIT requires high anchor density to perform PITs. With such sparse anchor distribution, it will fail to localize most of the nodes in the network.

Fig. 4(b) shows the proportion of finitely localized nodes against the average degree. We can see that a higher network density leads to a lower proportion of finitely localizable nodes. That is because nodes have higher probabilities to form globally rigid components in dense networks, and localizing nodes in globally rigid components causes no ambiguities at all.

C. Localization Performance Against Anchor Density

In this experiment, we randomly distribute 200 nodes in a square region with the average degree of about 6. We control the anchor density by randomly selecting a certain percentage of nodes as anchors. Fig. 5 plots the average result of 100 network instances, when the anchor proportion varies from 3% to 10%.

Fig. 5(a) plots the proportion of uniquely localized nodes where the anchor density enlarges. With the increase of anchor proportion, the performance of each algorithm linearly increases. We can see that Sweeps benefits more from large anchor density because it prefers the fact that anchors reside closely so as to initialize the bilateration procedure. MDS gets the lowest performance in this test. When the average degree is 6, MDS cannot generate uniformly overlapped local maps. Consequently, MDS cannot integrate the network into a global map, but partitions the network into a set of small-scale local maps. Hence, MDS can only localize these local maps that contain at least three anchors.

Fig. 5(b) shows the proportion of finitely localized nodes against anchor density. CALL localizes the largest number of nodes in finite states since CALL can identify most globally rigid parts as well as the finitely localizable nodes. The number of finitely localized nodes decreases when the anchor density increases. The inherent reason is that the total amount of finitely localizable nodes in a network is reduced when more anchors

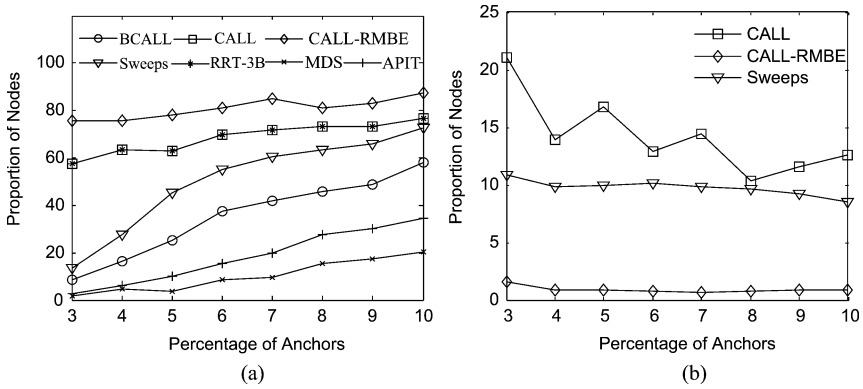

Fig. 5. Proportion of located nodes against anchor density. (a) Uniquely located nodes. (b) Finitely located nodes.

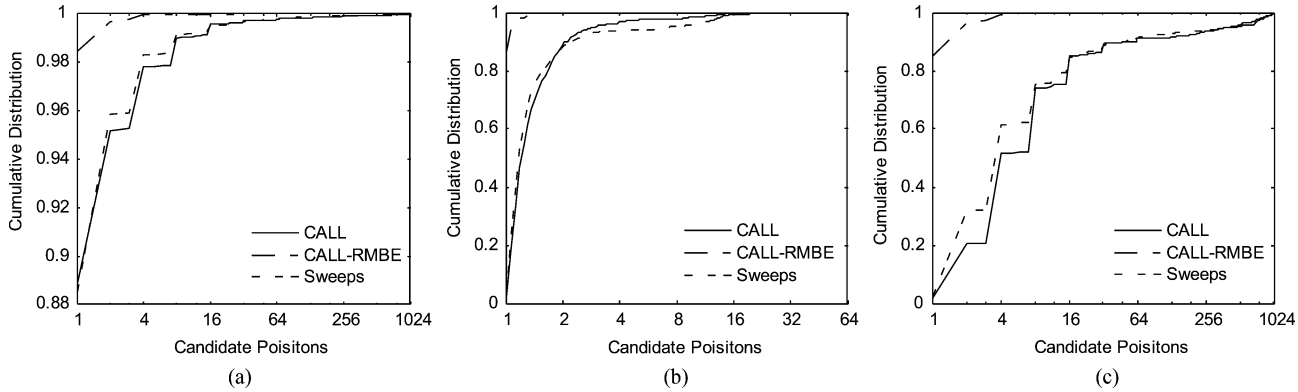

Fig. 6. Cumulative distribution of the possibility. (a) Number of candidate positions for each single node. (b) Average number of candidate positions in each network instance. (c) Maximum number of candidate positions in each network instance.

exist. In contrast, the number of finitely localized nodes by Sweeps is not affected much by anchor density. Sweeps typically localizes a whole globally rigid region [2] and localizes nodes on the border of this region finitely, so the number of finitely localized nodes is related to the size of such a region. Nevertheless, the size of such a region is determined by the network density, and the anchor density has little impact on it.

D. Distribution of the Number of Candidate Positions

The worst-case complexity could be exponential in the number of nodes for Sweeps, CALL, and CALL-RMBE, thus we conduct this experiment to investigate the distribution on the number of candidate positions. In this experiment, we randomly distribute 200 nodes in a square region with the average degree about 6. Then, we randomly select 10% of the nodes as anchors. Fig. 6 reports the result of 1000 network instances. Note that the x-coordinate in each graph is logarithmic.

Fig. 6(a) plots the cumulative distribution of the number of candidate positions for each single node. All the algorithms uniquely localize over 88% of nodes. Moreover, CALL generates a bit larger position sets than Sweeps. The reason is that CALL can locate more nodes in finite states as shown in previous experiments.

Fig. 6(b) plots the cumulative distribution of the average number of the candidate positions in each network instance. The average number of candidate positions for each algorithm is less than four over 90% cases. For the average number of candidate positions, CALL and Sweeps are also at the same level.

Fig. 6(c) plots the cumulative distribution of the maximum number of the candidate positions among the nodes in each network instance. The maximum number for each algorithm is less than 64 in about 90% of cases. CALL gets a bit higher maximum number as it finitely localizes more nodes than Sweeps, and the combinatorial explosion becomes more serious in this case.

Overall, CALL-RMBE incurs absolutely low computational cost, while CALL and Sweeps are of the same level in all aspects.

VII. Conclusion

We propose the concept of components as well as a component-based approach, CALL, to address the localization issue in sparse wireless ad hoc and sensor networks. We form basic rules for operations on components and design RMBE to further improve the performance of this approach. Theoretical analysis and simulation results show that this design significantly outperforms previous approaches. Future work leads into three directions. First, we will extend this design to generate robust results with noisy distance measurements. Second, we are going to investigate the theoretical bound of localizability using polynomial spatial-temporal cost. Lastly, we will apply our design in our ongoing system, GreenObs, and further examine its applicability in the real system.

REFERENCES

[1] S. Lederer, Y. Wang, and J. Gao, "Connectivity-based localization of large scale sensor networks with complex shape," in *Proc. IEEE INFOCOM*, Phoenix, AZ, 2008, pp. 789–797.

[2] D. Goldenberg, P. Bihler, M. Cao, J. Fang, B. Anderson, A. S. Morse, and Y. R. Yang, "Localization in sparse networks using sweeps," in *Proc. ACM MobiCom*, Los Angeles, CA, 2006, pp. 110–121.

[3] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: A high accuracy acoustic ranging system using COTS mobile devices," in *Proc. ACM SenSys*, Sydney, Australia, 2007, pp. 1–14.

[4] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur, "Rigidity, computation, and randomization in network localization," in *Proc. IEEE INFOCOM*, Hong Kong, 2004, pp. 2673–2684.

[5] D. K. Goldenberg, A. Krishnamurthy, W. C. Maness, Y. R. Yang, and A. Young, "Network localization in partially localizable networks," in *Proc. IEEE INFOCOM*, Miami, FL, 2005, pp. 313–326.

[6] Z. Yang, Y. Liu, and X.-Y. Li, "Beyond trilateration: On the localizability of wireless ad-hoc networks," in *Proc. IEEE INFOCOM* , Rio de Janeiro, Brazil, 2009, pp. 2392–2400.

[7] Y. Liu, Z. Yang, X. Wang, and L. Jian, "Location, localization, and localizability," *J Comput. Sci. Technol.*, vol. 25, no. 2, pp. 274–297, 2010.

[8] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor location sensing using active RFID," *Wireless Netw.*, vol. 10, pp. 701–710, 2004.

[9] Z. Yang, M. Li, and Y. Liu, "Sea depth measurement with restricted floating sensors," in *Proc. IEEE RTSS*, Tucson, AZ, 2008, pp. 469–478.

[10] Z. Yang and Y. Liu, "Quality of trilateration: Confidence-based iterative localization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 631–640, May 2010.

[11] M. Li and Y. Liu, "Rendered path: Range-free localization in anisotropic sensor networks with holes," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 320–332, Feb. 2010.

[12] C. Savarese, K. Langendoen, and J. Rabaey, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *Proc. USENIX Annu. Tech. Conf.*, Monterey, CA, 2002, pp. 317–327.

[13] Y. Shang and W. Ruml, "Improved MDS-based localization," in *Proc. IEEE INFOCOM*, Hong Kong, 2004, pp. 2640–2651.

[14] D. Moore, J. Leonard, D. Rus, and S. J. Teller, "Robust distributed network localization with noisy range measurements," in *Proc. ACM SenSys*, Baltimore, MD, 2004, pp. 50–61.

[15] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proc. ACM WSNA*, Atlanta, GA, 2002, pp. 112–121.

[16] J. Bruck, J. Gao, and A. A. Jiang, "MAP: Medial axis based geometric routing in sensor network," in *Proc. ACM MobiCom*, Cologne, Germany, 2005, pp. 88–102.

[17] X. Wang, J. Luo, S. Li, D. Dong, and W. Cheng, "Component based localization in sparse wireless ad hoc and sensor networks," in *Proc. IEEE ICNP*, 2008, pp. 288–297.

[18] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proc. ACM MobiHoc*, Annapolis, MD, 2003, pp. 201–212.

[19] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "Range-free localization schemes in large-scale sensor networks," in *Proc. ACM MobiCom*, New York, 2003, pp. 81–95.

[20] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "Range-free localization and its impact on large scale sensor networks," *Trans. Embedded Comput. Syst.*, vol. 4, pp. 877–906, 2005.

Xiaoping Wang received the B.S. and M.S. degrees in 2003 and 2006, respectively, from the School of Computer Science, National University of Defense Technology, Changsha, China, where he is currently pursuing the Ph.D. degree.

His research interests include sensor networking and operating system.

Jun Luo received the B.S. degree from the Computer School, Wuhan University, Wuhan, China, in 1984, and the M.S. degree from the School of Computer Science, National University of Defense Technology, Changsha, China, in 1989.

He is now a Professor in the School of Computer Science, National University of Defense Technology. His research interests include operating system, parallel computing, security, and sensor networking.

Yunhao Liu (M'02–SM'06) received the B.S. degree from the Automation Department, Tsinghua University, Beijing, China, in 1995, and the M.S. and Ph.D. degrees in computer science and engineering from Michigan State University, East Lansing, in 2003 and 2004, respectively.

He is a member of the Tsinghua National Lab for Information Science and Technology and the Director of the Tsinghua National MOE Key Lab for Information Security, both in Tsinghua, China. He is also a member of the faculty with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.

Dr. Liu is an Association for Computing Machinery (ACM) Distinguished Speaker.

Shanshan Li received the B.S. degree in computer science and technology from Nanchang University, Nanchang, China, in 2001, and the M.S. and Ph.D. degrees from the School of Computer Science, National University of Defense Technology, Changsha, China, in 2003 and 2007, respectively.

During 2007, she was with Hong Kong University of Science and Technology as a visiting scholar. She is now an Assistant Professor with the School of Computer Science, National University of Defense Technology. Her research interests include sensor networking and pervasive computing.

Dezun Dong received the B.S. and M.S. degrees in 2002 and 2004, respectively, from the School of Computer Science, National University of Defense University, Changsha, China, where he is currently pursuing the Ph.D. degree.

His research interests include security and reliability in wireless mobile ad hoc and sensor networks.