Robust Location Detection with Sensor Networks

Saikat Ray, David Starobinski, Ari Trachtenberg, Rachanee Ungrangsi

***Abstract*— We propose a novel framework for location detection with sensor networks, based on the theory of identifying codes. The key idea of this approach is to allow sensor coverage areas to overlap so that each resolvable position is covered by a unique set of sensors. In this setting, determining a sensor-placement with a minimum number of sensors is equivalent to constructing an optimal identifying code, an NP-complete problem in general. We thus propose and analyze new polynomial-time algorithms for generating irreducible (but not necessarily optimal) codes for arbitrary topologies. Our algorithms incorporate robustness properties that are critically needed in harsh environments. We further introduce distributed versions of these algorithms, allowing sensors to self-organize and determine a (robust) identifying code without any central coordination. Through analysis and simulation, we show that our algorithms produce nearly optimal solutions for a wide range of parameters. In addition, we demonstrate a tradeoff between system robustness and the number of active sensors (which is related to the expected lifetime of the system). Finally, we present experimental results, obtained on a small testbed, that demonstrate the feasibility of our approach.**

I. Introduction

Communication systems play an essential role in harsh environments such as densely populated indoor locations, building collapses, or extreme weather phenomena. Unfortunately, existing systems often provide minimal infrastructure for handling such environments on-site. Recent advances in wireless sensor technologies [1, 2] make it possible to install tiny devices in existing infrastructure, such as smoke detectors or overhead lighting, that can form a network. These networks can provide three dimensional building visualization along with real-time monitoring of hot spots, structural failures, or interference sources. Key to the above capabilities is a functional means of *location detection*. The ability to determine the locations of the necessary parties, be they trapped fire-fighters or essential portable equipment, is critical to the usefulness of sensor networks in harsh environments. At the same time, it is precisely in such environments, where noise comes not only from well-studied forms of interference but also from sensor displacement and structural changes, that location detection is very difficult.

Though several indoor location detection systems have been described in the literature (cf. Section II), none of them have been designed specifically for robustness. Chiefly, they lack the robustness needed to protect against equipment failure and changing structural topology. In particular, several existing systems are based on the concept of *proximity-based detection*, in which user location is provided by a nearest sensor (or beacon). When sensors fail in such systems, due to physical damage or interference, an entire coverage area may be lost.

In this paper, we propose to address the issue of robust location detection through a novel framework based on the theory of *identifying codes* [3, 4]. Our approach generalizes existing proximity-based location detection techniques by allowing sensor coverage areas to overlap. Our key idea is to ensure that each resolvable position is covered by a unique set of sensors, which then serves as its signature. We make use of identifying code theory to reduce the number of active sensors required by the system, thereby allowing a large number of sensors to be maintained in an energy-saving mode. Alternatively, activating more sensors increases the robustness of the system.

The main challenge in designing our system is to position sensors so that every resolvable location can be identified unambiguously. Toward this end, we divide a continuous coverage area into a finite set of regions, with each region being represented by a single point within its boundary. These points are thereafter mapped to nodes in a graph, and these nodes are connected by links whenever the corresponding points in the physical system are able to communicate directly. The identifying code problem is then to determine the nodes on which to place and activate the sensors (or *codewords* in the coding terminology), such that each node is within the communication range of a different set of sensors.

The problem of finding a minimum identifying code for an arbitrary graph is NP-complete [5]. Instead, we propose a novel greedy algorithm, called ID-CODE, that produces *irreducible* identifying codes from which no codeword can be removed without violating the unique identifiability of some position. Our numerical results show that the solution produced by our algorithm is nearly optimal for a wide range of parameters.

We also introduce the concept of r-robust identifying codes, which are capable of correcting up to r errors. We propose a new algorithm called r-ID-CODE that generalizes the basic ID-CODE algorithm and produces irreducible r-robust codes. The degree of robustness, r, is a design parameter that can be traded off with the number of sensors required for the proper functioning of the system. We present numerical results illustrating this trade-off. We also show how the r-ID-CODE algorithm can be run in a fully distributed manner. Thus, in a dynamic system, sensors can self-organize and determine a (robust) identifying code without any central coordination.

The authors are with the Electrical and Computer Engineering Department at Boston University. This work was supported in part by the National Science Foundation under NSF grants ANI-0132802, CCR-0133521, ANI-0240333 and by a SPRInG award from Boston University.

Finally, as a proof-of-concept, we present experimental results obtained on a small testbed to illustrate the feasibility and desirability of our approach compared to proximity-based schemes.

This paper is organized as follows. Section II briefly surveys related work in indoor location detection and identifying codes. In Section III, we outline our proposed system, explaining the relationship between robust location detection and the construction of identifying codes for arbitrary graphs. In Section IV, we describe our ID-CODE algorithm, prove some of its key properties, and describe how to apply it to an arbitrary topology. We introduce the concept of r-robust identifying codes in Section V and extend the ID-CODE algorithm to produce them. We then develop distributed versions of these algorithms in Section VI. In Section VII and VIII, we evaluate the performance of our algorithms and illustrate the benefits of the proposed approach through simulation and experiments, respectively. The last section summarizes the main findings of the paper and provides some concluding remarks.

II. Related Work

A. Location Detection Systems

Location detection systems have been proposed and implemented in the literature for a variety of applications. For outdoor applications, the satellite based *Global Positioning System* (GPS) is commonly used [6]. GPS relies on trilateration of position and time among four satellites, and can determine location in many cases within a few meters. However, occlusions, reflections, and multi-path effects limit the usefulness of GPS in indoor, dense, or harsh environments.

Indoor location detection systems have been developed for cases when GPS usefulness is limited. These systems can be classified into three categories, based on wave frequency: *Infrared* (IR), *Ultrasound* (US), and *Radio* (RF). Existing systems work well for their designed purposes, but cannot handle significant changes in communications paths or building topology.

Infrared: The *Active Badge* location system [7] was one of the first indoor location detection systems and is representative of the IR-based approach to indoor location detection [8, 9]. This system provides each person with a badge that periodically emits a unique ID using diffused IR that is received by one of several receivers scattered throughout a building. Badge location is then resolved by proximity to the nearest receiver. In harsh settings, however, the communication environment can be very dynamic, as people move about, smoke or other impurities fill the air, or walls collapse. In such settings, proximity to a single receiver is not sufficiently robust or flexible to provide reliable location detection.

Ultrasound: Ultrasound-based systems also provide location detection based on proximity, but improve accuracy by measuring ultrasound time-of-flight with respect to a reference RF signal. Systems such as the *Active Bat* [10] or MIT's *Cricket* [11] compare the arrival time of the two signals from various known sensors in order to calculate a listener's location.

As with the IR-based schemes, current ultrasound-based systems are not designed for robustness, since line-of-sight paths may get obstructed or altered in the face of changing room dynamics. In addition, these systems are particularly sensitive to the possible destruction of sensors.

Radio: Radio waves provide a powerful means of location detection because of their ability to penetrate many types of surfaces and objects, and due to their range, scalability, and maintenance benefits. Rather than using differences in arrival time, as done by ultrasound systems, RF-based location detection systems determine location based on received signal strength, predicated on a known *Signal-to-Noise Ratio* (SNR). RADAR [12] pre-computes an SNR-map for a building. A vector of signal-strengths received at various base-stations is compared to this map to determine position. Other RF-based systems include SpotON [13] and *Nibble* [14].

As with the previously mentioned schemes, there are still inherent issues of robustness when utilizing RF. The failure of a sensor or the introduction of new signal path from spurious reflectors (*e.g.,* people walking around) or shifting internal structures can severely impair existing systems. SNR-based systems have also the problem of being sensitive to environmental conditions. Recently, [15] suggested a scheme for location detection, based on computing the centroid of the positions of several base stations, that addresses some of these issues. However, this scheme applies only to large open environments.

B. Identifying Codes

The system proposed in this work overlays an *identifying code* on a proximity-based location detection system in order to improve resolution and robustness.

Identifying codes were introduced in [3] as a means of uniquely identifying faulty processors in a multiprocessor system. These codes, which are described in detail in Section IV, have enjoyed much attention in the coding theory literature. In general, finding an optimal identifying code is known to be an NP-complete problem [5]. The available constructions in the literature have so far been restricted to regular graphs such as hypercubes, meshes, and trees [16]. The works in [17] suggest the use of these known identifying codes for surveillance purposes in an outdoor setting, but they require a regular, mesh topology. Unlike multiprocessor networks, wireless networks usually do not form a regular graph even if the nodes are arranged on a regular spatial grid, especially in indoor settings with many obstacles and reflectors. Moreover, sensor networks require robustness that is not available from standard identifying codes. Our system makes use of a robustness-oriented modification of identifying codes built over an arbitrary topology, as described in Sections III and IV. Our techniques for building these codes are practically realizable and provide codes with sizes close to known lower bounds and, hence, almost optimal.

III. System overview

Our proposed system divides the coverage area into locatable regions represented by a designated point. This system can operate in either or both of two dual modes: *location service* or *location tracking*. In the location service mode, the

system periodically broadcasts ID packets from designated sensors. An observer can determine her location from the packets that she receives. In the location tracking mode, an observer transmits her ID and the system determines her location from the sensors receiving the ID. Hereafter, we shall describe the system as it operates in the location service mode, though our results apply analogously to the location tracking mode.

Our sensor network is designed as follows: first, a set of points is selected for a given area. Then, based on physical point connectivity (RF in our example), transmitting sensors are placed on a subset of these points corresponding to an identifying code. This placement guarantees that each point is covered by a unique set of transmitters. Thus, an observer can determine his location from the unique collection of received ID packets.

The transmitter placement induces an indistinguishable region around each locatable point (*i.e.,* an observer would receive the same set of ID packets anywhere in this region). This system alone does not guarantee coverage beyond the points incorporated into the graph model. To ensure more widespread coverage, additional techniques should be employed [18, 19].

A. Example

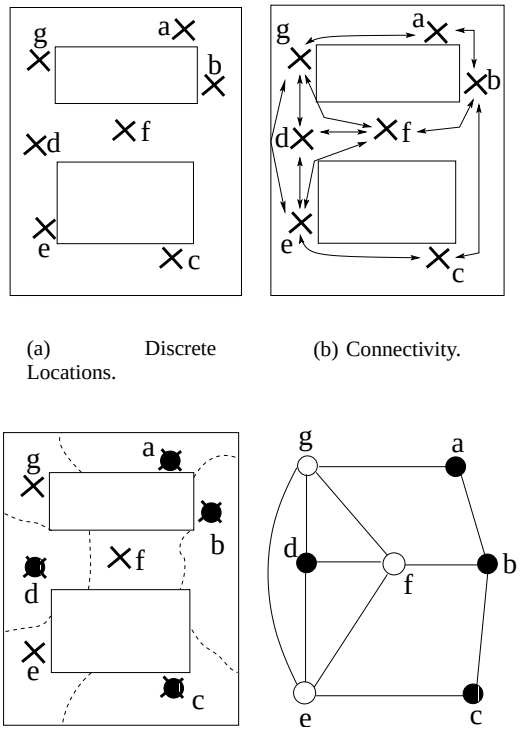
The following example illustrates our approach in more detail. Consider the points $P = \{a, b, c, d, e, f, g\}$ on a simple floor plan illustrated in Fig. 1(a), and let the RF-connectivity among these points be represented by the arrows in Fig. 1(b); in other words, there is an arrow between positions p_1 and p_2 if and only if a transmitter placed at p_1 can directly send packets to a receiver placed at p_2 through RF. Given such connectivity information between every pair of points, our objective is to build a system using a minimum number of transmitters that allows an observer to infer his location at any point in P.

For this purpose, we place four wireless transmitters at positions a, b, c and d, each periodically broadcasting a unique ID. We assume that packet collisions are avoided by an appropriate medium access control (*e.g.,* simple randomization or a full-scale protocol [15]) and that the observer collects received packets over a (small) time T. For instance, in Fig. 1(c) an observer in the region of point f receives IDs from the transmitters at positions b and d. The set of IDs received at a given position x is called the *identifying set* of x and denoted $ID(x)$.

If the identifying set of each point in P is unique, then targets can be correctly located at these points using a table-lookup of the packet IDs received. The reader can verify that, for this example, the identifying sets are unique and given as follows:

$v:$	a	b	c	d	e	f	g
$ID(v):$	$\{a,b\}$	$\{a,b,c\}$	$\{b,c\}$	$\{d\}$	$\{c,d\}$	$\{b,d\}$	$\{a,d\}$

In general, we model a physical environment with a graph $G = (V, E)$, whose vertices V model locatable regions and edges E connect regions with RF connectivity. Fig. 1(d) shows the graph for the example in Fig. 1(b). Note that vertices of the graphs can be mere points in space, and physical transmitters need only be placed at those points designated by the chosen identifying code. Alternatively, graph vertices can be used to represent all the points where sensors are positioned. In such a case, transmitting sensors are activated on a subset of these points determined by a corresponding identifying code. Other sensors can be maintained in sleep mode.



(a) Discrete Locations. (b) Connectivity.

(c) Sensor placement (filled circles). Coverage area is delimited by dashed lines. (d) Position of the sensors on the graph.

Fig. 1. Our proposed location detection system.

In the following section, we develop a theoretical framework, based on identifying codes, for placing transmitting sensors in our location detection system.

IV. Identifying codes for arbitrary graphs

As mentioned in Section II, the problem of constructing optimal identifying codes for arbitrary graphs is known to be NP-complete. Therefore, rather than looking for an optimal solution, we propose a greedy algorithm to construct *irreducible* identifying codes. The irreducibility property means that the deletion of any codeword results in a code that is no longer an identifying code. Thus, the proposed algorithm always converges to a local minimum. In fact, our simulation results presented in Section VII show that the solution achieved by this algorithm is nearly optimal for a wide range of parameters.

A. Notations and Definitions

Let $G = (V, E)$ be a given graph with vertices V and edges E. Then, we define the distance metric $\rho(u, v)$ to be the number of edges along the shortest path from vertex u

to v. The *ball* of radius h around v is denoted $\mathcal{B}(v,h)$ and defined to be $\{w \in V : \rho(w,v) \leq h\}$. Thus, $\mathcal{B}(v,1)$, which we shall denote simply as $\mathcal{B}(v)$, represents the set of vertices that are adjacent to v, together with v.

Any non-empty subset $\mathbb{C} \subseteq V$ is called a *code* for the corresponding graph $G = (V,E)$, and its elements are called *codewords*. Given a code $\mathbb{C}$, the *identifying set* of a vertex $v \in V$ is defined to be $I_{\mathbb{C}}(v) = \mathcal{B}(v) \cap \mathbb{C}$. A code $\mathbb{C}$ is called an *identifying code* if $I_{\mathbb{C}}(u) \neq I_{\mathbb{C}}(v) \neq \emptyset$ for all $u \neq v \in V$; that is, the identifying set of every vertex in the graph is unique and non-empty[1] (so that every vertex is uniquely identified by its identifying set). An identifying code $\mathbb{C}$ is called *irreducible* if deletion of any codeword from $\mathbb{C}$ results in a code that is no longer an identifying code. A graph $G = (V,E)$ is said to be *distinguishable* if it permits an identifying code; otherwise, G is an indistinguishable graph.

B. Code Construction Algorithm

Formally, our problem of location detection is: *Given a distinguishable graph $G = (V,E)$, determine a subset $\mathbb{C} \subseteq V$ of minimum cardinality that is an identifying code.* Since this problem has been shown to be NP-complete, we instead consider a practical modification: *Given a distinguishable graph $G = (V,E)$, compute a subset $\mathbb{C}$ of V such that $\mathbb{C}$ is an irreducible identifying code for G.*

The first step in solving the above question is to determine whether a given graph is distinguishable. The following lemmas show that this determination is not hard to do in practice.

Lemma 1 *For a given graph $G = (V,E)$, if $\mathbb{C}$ is an identifying code, then every $\mathbb{C}' \supseteq \mathbb{C}$ is also an identifying code.*

Proof: Assume that there exists $\mathbb{C}' \supseteq \mathbb{C}$ that is not an identifying code. Then, by definition, there exist $u, v \in V$ with

$$\begin{aligned} I_{\mathbb{C}'}(u) &= I_{\mathbb{C}'}(v) \\ \mathbb{C} \cap \mathbb{C}' \cap \mathcal{B}(u) &= \mathbb{C} \cap \mathbb{C}' \cap \mathcal{B}(v) \\ \mathbb{C} \cap \mathcal{B}(u) &= \mathbb{C} \cap \mathcal{B}(v), \qquad \text{since } \mathbb{C} \subseteq \mathbb{C}' \\ I_{\mathbb{C}}(u) &= I_{\mathbb{C}}(v), \end{aligned}$$

which contradicts the assumption that $\mathbb{C}$ is an identifying code. ■

Thus, to check if a graph is distinguishable, one must merely check that there are no two vertices with the same ball. Empirically, we have found that almost all graphs are distinguishable unless their average degree is very low or very high. Graphs that are indistinguishable generally have a collection of vertices that are physically close to each other. The work in [4] describes a simple method for transforming an indistinguishable graph into a distinguishable one by deleting a minimum number of vertices.

Algorithm ID-CODE, which we now describe, builds an identifying code for a distinguishable graph. It begins by designating every vertex in an input graph G as a codeword. At each step of the algorithm, one codeword is considered for deletion from the current code. If removing the codeword results in an identifying code, then the algorithm proceeds; otherwise, the codeword is reinserted into the code and the algorithm proceeds to consider other codewords, along a predetermined sequence of vertices, σ, provided as a parameter.

By design, each iteration of the algorithm (including the last iteration) ends with an identifying code of the graph. Moreover, the algorithm performs one iteration for each vertex in the graph and at each iteration, checks for uniqueness of the identifying set of each node. Using an appropriately-constrained sorting algorithm to test uniqueness, the running time of the algorithm is $O(|V|^3 \log |V|)$. This can be reduced further to an expected $O(|V|^2 \log |V|)$ with the use of hash functions.

Theorem 1 *The code $\mathbb{C}$ returned by ID-CODE is irreducible.*

Proof: Assume, for sake of contradiction, that $\mathbb{C} \setminus X$ is an identifying code for some set $X \neq \emptyset, X \subset \mathbb{C}$. Choose any codeword $x \in X$ and denote by i the iteration of the ID-CODE algorithm in which x is considered, resulting in the code $\mathbb{C}_i \supset \mathbb{C}$. It must be that the set $\mathbb{C}' \stackrel{\text{def}}{=} \mathbb{C}_i \setminus \{x\}$ is not an identifying code, or else ID-CODE would have removed x from $\mathbb{C}$. Moreover, $\mathbb{C}' \supseteq \mathbb{C} \setminus \{x\} \supseteq \mathbb{C} \setminus X$. However, since $\mathbb{C} \setminus X$ is an identifying code, Lemma 1 implies that $\mathbb{C}'$ is an identifying code as well, which completes the contradiction. ■

Theorem 1 shows that a code returned by the ID-CODE is irreducible. The following theorem shows the converse, that is, that every irreducible identifying code, including optimal ones, can be generated by ID-CODE through an appropriate choice of the input parameters. A proof can be found in [4].

Theorem 2 *For every irreducible identifying code $\mathbb{C}$ of a given graph $G = (V,E)$, there exists an input sequence σ such that* ID-CODE(*G*,σ) *returns* $\mathbb{C}$.

The performance of the ID-CODE algorithm depends on the sequence of vertices chosen. In [4], some simple heuristic methods for effectively ordering the input sequence are evaluated. For random graphs, it shown that the performance of the algorithms is rather insensitive to the ordering of the sequence, and randomly ordered sequences generally perform well enough.

V. r-ROBUST CODE CONSTRUCTION

In the previous section we described techniques for constructing identifying codes with as few codewords as possible. That framework inherently provides some amount of robustness since a location may be covered by sensors located far off, thereby creating spatial diversity. However, in practice, the identifying set received by an observer might fluctuate due to changes in communication conditions, and thus we seek to guarantee that the scheme works even if the received identifying set differs somewhat from the original one.

In this section we describe a novel generalization of identifying codes that achieves this goal by guaranteeing to be

[1]The non-emptiness constraint is not part of the original definition of identifying codes, but it is important for our practical implementations, especially our distributed algorithm in Section VI.

robust in the face of fluctuations in observed identifying sets. First, we formalize our definition of robustness, making use $\oplus$ to denote symmetric difference between two sets (*i.e.,* $A \oplus B = (A \setminus B) \cup (B \setminus A)$).

Definition 1 *An identifying code $\mathbb{C}$ over a given graph $G = (V, E)$ is said to be* r-robust *if $I_{\mathbb{C}}(u) \oplus A \neq I_{\mathbb{C}}(v) \oplus B$, for all $u, v \in V$, $u \neq v$ and $A, B \subseteq \mathbb{C}$ with $|A|, |B| \leq r$.*

Simply stated, an identifying code is r-robust if addition or deletion of up to r IDs in the identifying set of any vertex does not change its identifying capability. Alternatively, we may determine the robustness of code $\mathbb{C}$ by finding the minimum symmetric difference $d_{\min}(\mathbb{C}) \stackrel{\text{def}}{=} \min_{u,v \in V} |I_{\mathbb{C}}(u) \oplus I_{\mathbb{C}}(v)|$ between the identifying sets of any two of its vertices. We thus have the following Theorem as a straightforward application of the definitions.

Theorem 3 *A code $\mathbb{C}$ is r-robust iff $d_{min}(\mathbb{C}) \geq 2r + 1$.*

Adding codewords to an identifying code can only increase its minimum symmetric difference, as per the following lemma.

Lemma 2 *For any two identifying codes $\mathbb{C} \subseteq \mathbb{D}$ over the same graph G, $d_{min}(\mathbb{C}) \leq d_{min}(\mathbb{D})$.*

As it turns out, a simple generalization of the greedy criterion of ID-CODE, depicted in Fig. 2, produces an r-robust code if it exists. As with ID-CODE, examining vertices in the right order will necessarily produce the best possible r-robust codes for the given graph.

By construction, $\mathbb{C}$ is an r-robust identifying code at every iteration of the algorithm and the straightforward running time is $O(|V|^4)$, though this can be reduced to $O(|V|^3)$ if there are no memory restrictions. Moreover, the following theorem, which is proven using Lemma 2 in a manner analogous to Theorem 1, shows that the resultant code is irreducible, meaning that the code is no longer r-robust if any codeword is removed.

Theorem 4 *The code $\mathbb{C}$ returned by r-ID-CODE is irreducible.*

To decode location with an r-robust code, an observer must be provided a lookup table with the identifying set of every vertex in a given (uncorrupted) graph. Upon receiving an identifying set S, the observer finds the point p that minimizes $|I_{\mathbb{C}}(p) \oplus S|$. As long as no more than r IDs are corrupted, the observer is guaranteed to determine his/her location correctly.

VI. Distributed solutions

It is often impractical to tie large sensor networks to a central authority. Fortunately, the previous algorithms can be distributed over network regions with a modest amount of inter-node communication. In this section, we describe such a distributed algorithm in successive stages. The algorithm makes use of a model in which a sensor is placed at each resolvable location. The goal of the algorithm is to determine a small subset of these sensors that need to be active (and transmitting), as part of an identifying code, so that other sensors may be put into energy-saving mode.

---

r-ID-CODE(G, σ, r)
$\mathbb{C} = V$
if $d_{\min}(\mathbb{C}) \leq 2r$ **then** EXIT
for each vertex $x \in \sigma$, taken in order **do**
  $D = \mathbb{C} \setminus \{x\}$
  if $d_{\min}(D) \leq 2r$ **then** $\mathbb{C} = \mathbb{C}$
  else $\mathbb{C} = D$
return $\mathbb{C}$

Fig. 2. The r-ID-CODE algorithm for generating r-robust identifying codes for an arbitrary graph.

A. Iteration

Recall that the ID-CODE algorithm initially designates each vertex in a given graph to be a codeword, and then iteratively deletes safe codewords from the code as long as possible. The key to distributing this algorithm among several processors is the observation that, in most cases, determining whether a codeword is safe to delete from the code can be done without consulting the entire network. More specifically, a codeword can be deleted if doing so will not make the identifying sets of any two vertices equal. The following lemma shows that vertices with equal identifying sets must be close to each other, as long as identifying sets are constrained to be non-empty. We note that the constraint of non-empty identifying sets is critical to our distributed algorithm, as otherwise, one cannot bound the search space from which codewords may be safely deleted.

Lemma 3 *Consider an identifying code $\mathbb{C}$ for a graph $G = (V, E)$. If, after removing a codeword $c \in \mathbb{C}$ from G, there exist distinct vertices v and v' with equal, non-empty identifying sets in the new code $\mathbb{C}' = \mathbb{C} \setminus \{c\}$ (i.e., $I_{\mathbb{C}'}(v) = I_{\mathbb{C}'}(v') \neq \emptyset$), then $v, v' \in \mathcal{B}(c, 3)$. More precisely, it must be that $v \in B(c, 1)$ and $v' \in B(v, 2)$.*

Proof: Since $\mathbb{C}$ is an identifying code, $I_{\mathbb{C}}(v) \neq I_{\mathbb{C}}(v')$. If both or neither of $I_{\mathbb{C}}(v)$ and $I_{\mathbb{C}}(v')$ contain c, we have $I_{\mathbb{C}'}(v) \neq I_{\mathbb{C}'}(v')$, which is contradictory. Thus, exactly one of v or v' is a neighbor of c. Moreover, since the identifying sets of v and v' are presumed non-empty and identical after removal of c from the code, the vertices must share at least one common neighbor (*i.e.,* be of distance ≤ 2 from each other), so that the maximum distance from c to these vertices is 3. ■

Lemma 3 shows that determining whether or not to delete a codeword c can be done by inspecting the identifying sets of all neighbors in $\mathcal{B}(c, 3)$. Since these identifying sets are determined by vertices in $\mathcal{B}(c, 4)$, this is the only portion of the network that needs to be checked when deciding whether or not to delete c from the code.

The iterative component of our algorithm thus proceeds as follows. Each node is initialized to be a codeword. As per Lemma 3, each codeword c checks for two conditions: (i) is it the sole mutual difference between identifying sets $I_{\mathbb{C}}(v)$ and $I_{\mathbb{C}}(v')$, for any v that is an immediate neighbor of c and v' of distance ≤ 2 from v; or (ii) is it the lone element in

Algorithm 1 [*Iteration*] An outline of the iteration component of our distributed algorithm.

• initialize our identifying code $\mathbb{C} = V$
• initialize $\zeta(v) = \mathcal{B}(v, 4)$ for each $v \in V$; this variable will hold the value of $\mathcal{B}(v, 4) \cap \mathbb{C}$
• mark each vertex $v \in V$ as *unresolved*
• the sensor at each vertex $c \in V$ repeats the following steps until it is resolved:

1. **if** $I_{\mathbb{C}}(v) \oplus I_{\mathbb{C}}(v') = \{c\}$ **or** $I_{\mathbb{C}}(v) = \{c\}$ for any $v \in \mathcal{B}(c, 1)$ and $v' \in \mathcal{B}(v, 2)$ **then** resolve c to be a codeword
   else if $index(c) \leq index(c')$ for all $c' \in \zeta(c)$ **then** resolve c to be a non-codeword and update identifying sets
2. **if** c is now resolved **then** broadcast its message to all $v' \in \mathcal{B}(c, 4)$.
3. **if** c receives a broadcast message from a resolved node v **then** $\zeta(c) = \zeta(c) \setminus \{v\}$

a neighbor's identifying set. If any of the above two cases is true, c must be kept in the code, in the first case to prevent v and v' from ending up with the same identifying set, and in the second case to avoid an empty identifying set for c or its neighbor.

If neither of the above two conditions are true, then c may be removed from the code because every set of mutual differences containing c also contains at least one other codeword. In order to avoid concurrency issues in which more than one codewords in the same set of mutual differences are deleted simultaneously leaving the set empty, we only drop c if it is has the lowest index among all codewords in its neighborhood ball of radius 4, where indexing is determined by an arbitrary injective function $index : V \longrightarrow \mathbb{Z}$.

Theorem 5 *Algorithm 1 produces an irreducible identifying code $\mathbb{C} \subseteq V$, if one exists, on a finite graph G.*

Proof: Since the graph size is assumed finite, a lowest-indexed unresolved vertex exists, which will always be resolved in step 1 of the algorithm. Thus, if there are N vertices in the graph, the algorithm terminates after at most N iterations. Moreover, Lemma 3 assures that the algorithm examines every potential identifying set conflict before removing a codeword, thereby assuring that if the original graph is identifiable, then the resulting $\mathbb{C}$ will be an identifying code. In addition, each vertex that is resolved to be a codeword is either the sole (mutual) difference between some pair of identifying sets or the only codeword in the identifying set of some vertex; meaning that deletion of the vertex will result in a non-identifying code. This proves the irreducibility of $\mathbb{C}$. ■

B. Precomputation

In the worst case, Algorithm 1 resolves only one vertex in the whole graph per iteration, which may be decidedly inefficient. A simple optimization involves checking, for each codeword c, those identifying set mutual differences that contain c. If every mutual difference that contains c also contains at least one other element with a greater index, then it is safe to remove c. Clearly, the nodes that are left unresolved still form an identifying code if the original graph is identifiable.

C. Neighborhood curbing

The initial contents of $\zeta(v)$ in Algorithm 1 are often larger than what is actually necessary. In fact, not all vertices in $\mathcal{B}(c, 4)$ need to be examined when deciding to remove c from an identifying code, but rather only those vertices that cannot be simultaneously removed with c. Lemma 3 shows that removal of a codeword can only affect identifying sets of vertices in $I_{\mathbb{C}}(v) \oplus I_{\mathbb{C}}(v')$, for $v \in \mathcal{B}(c, 1)$ and $v' \in \mathcal{B}(v, 2)$. Adding the constraint that identifying sets must be non-empty, we see that the set of vertices that need to be examined before deleting a codeword is

$$\begin{aligned} \zeta(c) \quad = \quad & \{x \in V | x \in I_{\mathbb{C}}(v) \oplus I_{\mathbb{C}}(v') \text{ or } x \in I_{\mathbb{C}}(v) \\ & \forall\ v \in \mathcal{B}(c, 1), v' \in \mathcal{B}(v, 2)\}. \end{aligned} \tag{1}$$

This set of vertices can be significantly smaller than $\mathcal{B}(c, 4)$ and can significantly reduce the number of iterations of Algorithm 1 by permitting simultaneous codeword removals.

D. Robustness

Recall from Theorem 3 that an identifying code is r-robust iff $\min_{u,v \in V} |I_{\mathbb{C}}(u) \oplus I_{\mathbb{C}}(v)| \geq 2r + 1$. We can thus make Algorithm 1 produce r-robust codes by having it keep a codeword c not only when it is the sole mutual difference, but, rather, when it is included in some set of mutual differences of size $2r + 1$.

E. Complete algorithm

We now present the complete, distributed algorithm for generating robust identifying codes. The pseudo-code presented is based on the previous subsections, but incorporates technical details and reorganizations that were previously left out for the sake of clarity. We denote the collection of mutual differences between the identifying sets near vertex v by

$$S_v = \{I_{\mathbb{C}}(v) \oplus I_{\mathbb{C}}(v') \quad | \quad v' \in \mathcal{B}(v, 2), v' \neq v\}.$$

In addition, the following quantity gathers all the sets that must be monitored by vertex v during the iterations of the algorithm

$$S_v^* = \left[\bigcup_{v' \in \mathcal{B}(v,1)} S_{v'} \right] \cup \left[\bigcup_{v' \in \mathcal{B}(v,1)} \{I_{\mathbb{C}}(v')\} \right].$$

Finally, for each vertex $v \in V$, the set $\zeta(v)$ is initialized by gathering the elements of S_v^* (*i.e.*, $\zeta(v) = \bigcup_{s \in S_v^*} s$). This set is then updated as vertices resolve themselves.

Notice that the only communication in Algorithm 2 involves the exchange of local neighborhood information in the initialization stage and local communication of vertex status in the iteration stage. In addition, each vertex performs the algorithm asynchronously; in other words, no processing coordination is required between vertices in this implementation.

Algorithm 2 Our complete distributed algorithm: DISTRIBUTED r-ID-CODE

Given an integer r and a graph $G = (V, E)$ whose vertices are indexed by a function $index(\cdot)$, the following distributed algorithm produces an r-robust identifying code $\mathbb{C}$.

Each vertex $v \in V$ performs the following steps concurrently:

Initialization

- set the current identifying code $\mathbb{C} = V$.
- mark v *unresolved*.
- determine the two-hop ball $\mathcal{B}(v, 2)$ and collect $I_{\mathbb{C}}(v')$ for all $v' \in \mathcal{B}(v, 2)$.
- compute S_v and collect $S_{v'}$ for all $v' \in \mathcal{B}(v, 1)$.
- compute S_v^* and set $\zeta(v) = \bigcup_{s \in S_v^*} s$

Precomputation

if $|s| = 2r + 1$ for any s such that $v \in s \in S_v^*$ **then**
  resolve v to be a codeword
else if for each s such that $v \in s \in S_v^*$, there exists $v' \in s$ with $index(v) < index(v')$ **then**
  resolve v to be a non-codeword
else keep v unresolved

Iteration

repeat until v is resolved:
  for each update received from node v' **do**
    $\zeta(v) = \zeta(v) \setminus \{v'\}$
    if v' is resolved as a non-codeword **then**
      delete v' from each set in S_v^* that contains v'
  if $|s| = 2r + 1$ for any s such that $v \in s \in S_v^*$ **then**
    resolve v to be a codeword
  else if $index(v) \leq index(v')$ for all $v' \in \zeta(v)$ **then**
    resolve v to be a non-codeword

Update

- once v is resolved:
  - communicate v's status (codeword or non-codeword) to all $v' \in \mathcal{B}(v, 4)$
  - the algorithm exits operation for v

VII. Performance Evaluation

In this section we evaluate the performance of our algorithms by applying them to random graphs.

Set-up: We use random, connected and distinguishable graphs with average degree $\bar{d}$, where $\bar{d}$ is a parameter. The graphs are generated by joining every two vertices with probability $p = \frac{\bar{d}}{(|V|-1)}$, and discarding disconnected or indistinguishable graphs. For every value of $\bar{d}$, results are obtained by averaging over 100 different graphs. Nodes are indexed randomly and, for the case of the centralized algorithm, visited in a random order. The graphs used in this simulation model an area comparable to the range of wireless transmitter with a large number of obstacles so that any two vertices might get connected.

The simulation results are obtained by varying the graph parameters $\bar{d}$ and $|V|$ and the desired degree of robustness r. The main metric of interest is the size of the resultant identifying code. For the distributed algorithm, we also compute the number of rounds (iterations) for the algorithm to converge.

Varying the average degree: Figure 3 shows the average size of the resultant code returned by the centralized and distributed versions of the ID-CODE algorithm for graphs with $|V| = 128$ (similar performance were observed for graphs with 16, 32 and 64 vertices). The bottom curve in the figure is a modified version of a lower bound provided in [3, Theorem 1(3)].

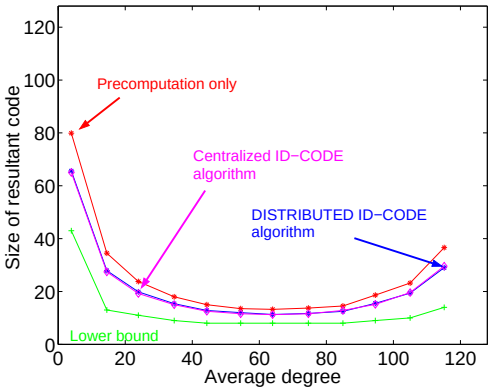


Fig. 3. Sizes of the identifying codes produced by the centralized ID-CODE and DISTRIBUTED ID-CODE algorithms for random graphs of 128 nodes and varying average degree $\bar{d}$.

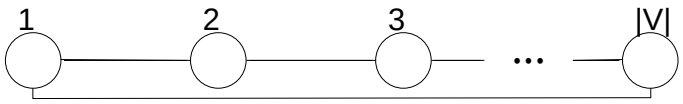


Fig. 4. An example of a worst case topology for the distributed algorithm. In this case, $\Theta(|V|)$ rounds are needed for convergence.

From the figure, we observe that the performance of the ID-CODE algorithm is close to the lower bound and hence to the optimal solution. Furthermore, the distributed version of the algorithm produces codes that are empirically as good as those produced by the centralized one. We note that the precomputation phase of the distributed algorithm is responsible for most of the quality of the computed identifying code.

For the same parameters, Fig. 5 shows that the distributed algorithm does a pretty good job of parallelizing the identifying code construction over the various vertices of the graph. In particular, the figure shows the importance of both the precomputation and neighborhood curbing for reducing the number of iterations. Overall, for the various connectivity graphs simulated, only about ten iterations in the algorithm were needed at any vertex to produce the desired code.

We note, however, that in the worst case, the distributed algorithm may take $\Theta(|V|)$ rounds to converge, as shown by Fig. 4. In that example, vertices are indexed in increasing order along a ring, so that precomputation resolves only nodes 1 and 2. Thereafter, each iteration results in at most 8 unresolved node being resolved, corresponding to a 4-hop neighborhood around resolved nodes. As such, at least $\frac{|V|}{8}$ iterations are needed for full resolution of the ring vertices; this is the asymptotic worst case because the iterative algorithm must, by design, resolve at least one node per iteration. We thus see the necessity of the iteration phase, since precomputation alone may be ineffective in reducing the code size.

Varying the graph size: Figure 6 shows simulation results

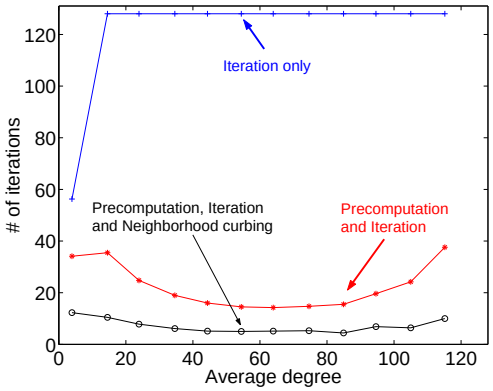


Fig. 5. The number of iterations needed to produce an identifying code for various stages of the distributed algorithm.

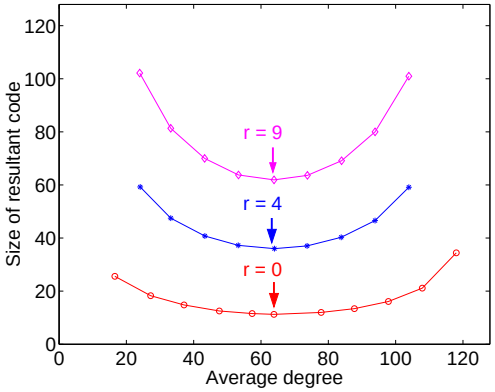


Fig. 7. Behavior of r-robust codes with 128 vertices.

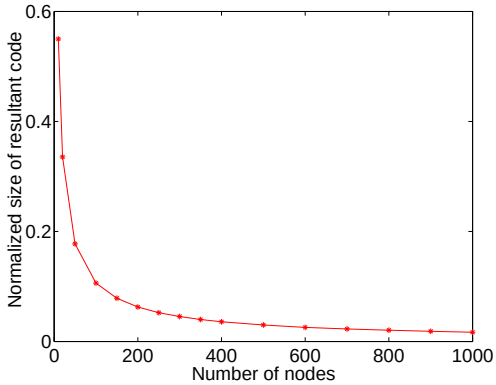


Fig. 6. Scalability of the resultant identifying code $\mathbb{C}$. The normalized size $|\mathbb{C}|/|V|$ is plotted against the number of vertices $|V|$ in the graph.

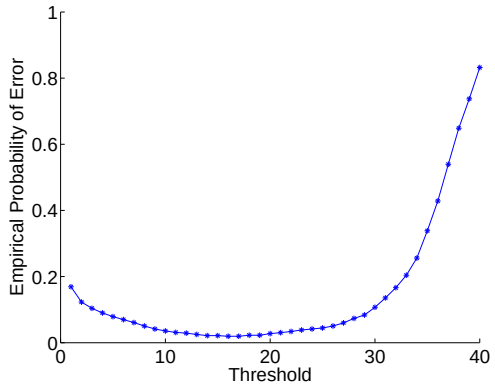


Fig. 8. Empirical probability of error in determining location, as a function of the connectivity threshold θ

of the centralized ID-CODE algorithm for graphs of average degree $\bar{d} = |V|/2$ and number of vertices $|V|$ varying from 10 to 1000. The figure clearly shows that the ratio of codewords to graph vertices decreases as the number of vertices increases. Thus, large graphs require relatively few transmitters for location detection. This shows that our approach scales well and is especially useful for graphs with large number of vertices. Note that in simple proximity-based systems, where each position is covered by a single sensor, the ratio $|\mathbb{C}|/|V|$ is always 1.

Varying the robustness: We applied the centralized r-ID-CODE to graphs of $|V| = 128$ vertices and varying average degree $\bar{d}$ with the results shown in Fig. 7. As expected, the code-size increases with increasing r so that there is a clear trade-off between the robustness of a code and the number of transmitting vertices that are required.

The general behavior of r-robust codes is similar to that of standard identifying codes. Minimum size is achieved if the average degree is about $|V|/2$, but the size of the resultant code is not too sensitive to the average degree. We see that for a large range of degree values around $|V|/2$, the code-size is close to the one obtained for $\bar{d} = |V|/2$. However, the sensitivity increases as the robustness requirements are increased.

Further results on the performance of our algorithms on other types of graphs (e.g. graphs generated using Waxman model) are reported in [20].

VIII. Experimental Proof-of-Concept

A. Set-up

We have developed a basic experimental testbed to verify our location detection scheme. The testbed is located on the 4th floor of the Photonics Building at Boston University and is depicted in Fig. 11. The white circles represent 10 discrete positions selected on the floor.

Five laptop computers running Red Hat Linux 8.0 are used in the experiments; four of them as transmitters and the fifth one as the receiver. The laptops are equipped with IEEE 802.11b standard compliant Cisco Aironet 350 series cards, operating at the 2.4 GHz band. Each transmitter sends 40 packets per second at a date rate of 5.5 Mb/s and transmission power of 100 mW. Each packet is 1000 bytes long and includes a field with the transmitter's ID.

B. Connectivity Model and Graph

In order to determine whether two points are connected, we employ a simple thresholding scheme. Specifically, each transmitter transmits 40 packets per second and two points are considered connected if the number of packets received during a sample interval exceeds a certain threshold θ. In our experiments, the sampling interval was set to 1 sec.

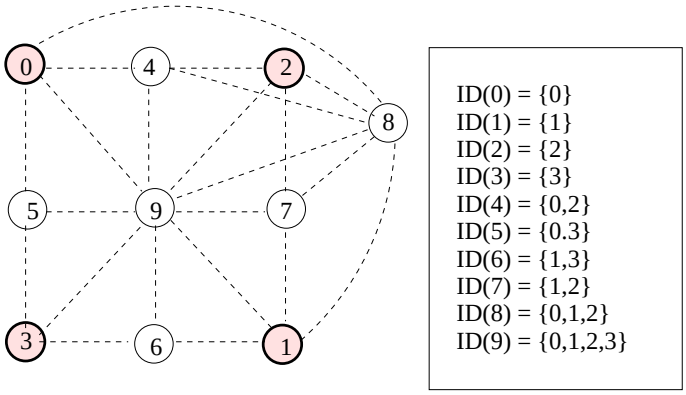


Fig. 9. The connectivity graph of the testbed and its identifying code. The shaded circles denote the codewords (transmitters).

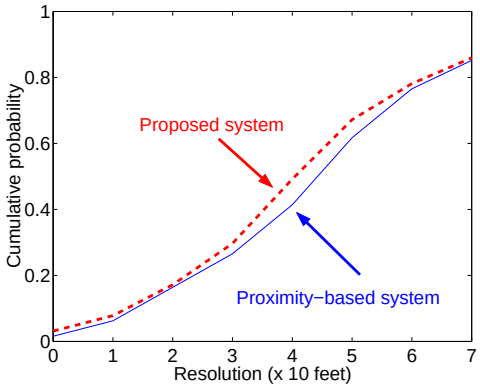


Fig. 10. Cumulative distribution function of the resolution achieved in two location detection systems.

Performance of the system – and the underlying connectivity graph – is dependent on θ. Fig. 8 shows the empirically observed probability of error in location determination as a function of θ. Based on this figure we chose $\theta = 15$ for our experiments [20], meaning that two nodes are connected if the receiver receives at least 15 packets out of the 40 sent by the sender.

We note that there exist more sophisticated decision-theoretic approaches, for instance, based on maximum likelihood or Euclidean distance criteria, in order to transform an observed packet vector into a binary codeword [20]. However, these methods may be impractical in dynamic environments as they involve extensive collection and distribution of *a-priori* data.

The resultant connectivity graph for the testbed is shown in Fig. 9. The solution produced by the ID-CODE algorithm results in a placement of the transmitters at positions $(0, 1, 2, 3)$ (correspondingly identified in Fig 11 by circles with a cross in them). The identifying sets for each position is shown in Fig. 9.

C. Data Collection

Our system was evaluated as follows. The floor plan was divided into a grid, where each grid location represents a 10×10 sq. ft. area. At each grid location, the packet arrival rate from all transmitters was recorded as a vector of the form (n_0, n_1, n_2, n_3), where n_i represents the number of packets received from transmitter i during a 1 second sample interval. While performing the experiments, we observed that the packet arrival rate varies with the orientation of the receiver's antenna. Thus, we collected 60×4 samples at each location, namely 60 samples per antenna's orientation (North, East, South, West).

D. Experimental Results

The resolution achieved using our location detection system is depicted in the contour map shown in Fig. 11. Resolution is defined as the (Euclidean) distance between the location resolved by the system and the actual user's location. In the figure, the resolution varies from 0 ft. to 70 ft. A darker shade corresponds to a higher resolution. The confidence level is 90%, i.e., at each position, at least 90% of the samples achieve the shown resolution. Although our system consists of only four transmitters, we observe that it achieves a reasonable resolution, most of the time within 50 ft. As expected, the resolution becomes coarser in areas that are distant from any of the discrete points.

As a basis of comparison, we also evaluate the resolution obtained with a simple proximity-based scheme. In this scheme, a user resolves its location to be that of its "closest" transmitter, that is, the transmitter from which it correctly receives the largest number of packets. Fig. 10 shows the cumulative distribution function (CDF) of the resolution for our proposed system and the proximity-based system. We observe that a larger number of positions are within a given error distance in our system than in the proximity-based system. This non-negligible gain in resolution is achieved through the sole use of identifying code techniques, and, thus, illustrates how judicious use of coding-theoretic approaches can contribute to improving the performance of location-detection systems.

IX. Conclusion

Robust location detection is an integral capability of a variety of wireless sensor network applications. In this paper, we have proposed a new framework for providing robust location detection in these systems and other harsh environments, based on the theory of identifying codes. Our approach involves overlapping sensor coverage so that each position on a floor is covered by a unique, and hence identifying, set of sensors.

We have proposed a polynomial-time algorithm, ID-CODE, for determining sensor placement by generating a corresponding irreducible identifying code. Since sensors may get destroyed and the connectivity between different positions may vary in dynamic environments, we have also introduced the new concept of r-robust identifying codes. These codes can tolerate up to r errors during the collection of ID packets, at any given position, while still providing accurate location information. We have thus provided a more generalized algorithm, known as r-ID-CODE, for generating irreducible r-robust identifying codes in polynomial time.

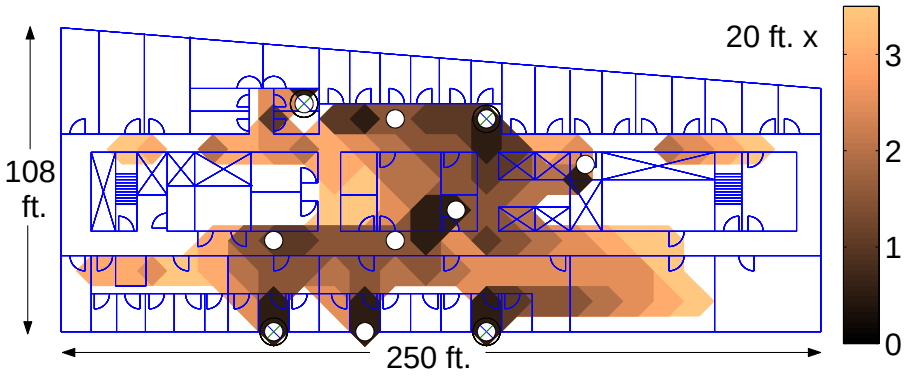


Fig. 11. Resolution (0-70 feet) of the proposed system with a 90% confidence level. Circles with crosses are transmitters; plain circles are locatable points.

The performance of our algorithms has been evaluated through extensive simulations on random graphs of varying sizes and varying average degrees. We have found that, for a wide range of parameters, the solution provided by our algorithms is close to a known theoretical lower bound and hence close to the optimal solution. We have also shown how the ID-CODE and r-ID-CODE algorithms can be run in a fully distributed fashion. The key property of these algorithms is that a sensor can determine its status by limiting its communication to nodes that are within a ball of bounded radius around it. Through simulations, we have shown that the distributed algorithms converge fairly quickly to solutions that are comparable to those produced by their centralized counterparts.

Finally, we have provided preliminary experimental evidence of the feasibility of our approach. We have built a small testbed, based on four wireless transmitters, and showed how to determine a connectivity graph and identifying code for it. Although the main targeted application of our identifying code methodology is in augmenting the robustness of current location detection systems, our experiments have shown that it may also be helpful in increasing their resolution. As part of our future work, we plan to extend our testbed to dozens of Berkeley mote sensors [21] to empirically examine tradeoffs between robustness and resolution.

ACKNOWLEDGMENT

The authors would like to thank Dr.'s M. Karpovsky and L.B. Levitin for introducing them to identifying codes, F. De Pellegrini for fruitful discussions while developing the ID-CODE algorithm, and C. Malladi for experiments on the testbed.

REFERENCES

[1] J.M. Kahn, R.H. Katz, and K. S. J. Pister, "Next century challenges: mobile networking for 'smart dust'," in *ACM MOBICOM*, Seattle, WA, United States, 1999.

[2] D. Estrin, D. Culler, and K. Pister G. Sukhatme, "Connecting the physical world with pervasive networks," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 59–69, Jan.-March 2002.

[3] M.G. Karpovsky, K. Chakrabarty, and L.B. Levitin, "A new class of codes for identification of vertices in graphs," *IEEE Trans. on Info. Theory*, vol. 44, no. 2, pp. 599–611, March 1998.

[4] S. Ray, R. Ungrangsi, F. De Pellegrini, A. Trachtenberg, and D. Starobinski, "Robust location detection in emergency sensor networks," *IEEE INFOCOM*, April 2003.

[5] I. Charon, O. Hudry, and A. Lobstein, "Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard," *Theoretical Computer Science*, vol. 290, no. 3, pp. 2109–2120.

[6] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*, Springer-Verlag, 4 edition, 1997.

[7] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Trans. on Info. Systems*, vol. 10, no. 1, pp. 91–102, Jan. 1992.

[8] S. Long, R. Kooper, G.D. Abowd, and C.G. Atkeson, "Rapid prototyping of mobile context-aware applications: The Cyberguide case study," in *ACM MOBICOM*, July 1996.

[9] R. Azuma, "Tracking requirements for augmented reality," *Communication of the ACM*, vol. 36, no. 7, pp. 50–51, July 1993.

[10] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The anatomy of a context-aware application," in *ACM MOBICOM*, Aug. 1999.

[11] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *ACM MOBICOM*, Boston, MA, 2000.

[12] P. Bahl and V.N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *IEEE INFOCOM*, Tel Aviv, Israel, 2000.

[13] J. Hightower, G. Borriello, and R. Want, "SpotON: An indoor 3D location sensing technology based on RF signal strength," Tech. Rep. #2000-02-02, University of Washington, Feb. 2000.

[14] P. Castro, P. Chiu, T. Kremenek, and R.R. Muntz, "A probabilistic room location service for wireless networked environments," in *ACM UbiComp*, Atlanta, GA, 2001.

[15] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," Tech. Rep. 00-729, University of Southern California/Information Sciences Inst., April 2000.

[16] I. Charon, O. Hudry, and A. Lobstein, "Identifying codes with small radius in some infinite regular graphs," *The Electronic Journal of Combinat.*, vol. 9, 2002.

[17] K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. on Computers*, vol. 51, no. 12, pp. 1448–1453, Dec. 2002.

[18] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *IEEE INFOCOM*, April 2001.

[19] S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak, "Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure," in *ACM MOBICOM*. Oct. 2001, pp. 106–116, ACM.

[20] R. Ungrangsi, "Location detection in emergency sensor networks using robust identifying codes," M.S. thesis, Boston University, 2003.

[21] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for network sensors," in *ACM ASPLOS*, Nov. 2000.

Saikat Ray received B.Tech degree from Indian Institute of Technology, Guwahati, India and M.S. degree from Boston University, in 2000 and 2002 respectively. Currently, he is working towards his PhD degree in Electrical and Computer Engineering, also at Boston University. He was a summer intern at Microsoft Research, Cambridge and Fujitsu Networking Inc. in 2003 and 2001 respectively. His research interest centers around wireless networking.

David Starobinski received his Ph.D. degree from the Technion-Israel Institute of Technology, in 1999. In 1999-2000 he was a post-doctoral fellow at the University of California, Berkley. Currently, he is an Assistant Professor at Boston University. He is a recipient of the NSF CAREER award and has been on the Program Committee of IEEE Infocom. His research interests are in networks performance evaluation, traffic engineering, and wireless networking..

Ari Trachtenberg received his Ph.D. and M.S. degrees in Computer Science from the University of Illinois at Urbana/Champaign, in 2000 and 1996, respectively, and his S.B. from MIT in 1994. Currently, he is an Assistant Professor at Boston University. His research interests include the application of coding theory to networks, data synchronization and location detection. He was the recipient of Mavis Memorial Fund Scholarship in 1999, the David J. Kuck Outstanding Thesis award in 2000, and the NSF CAREER award in 2002.

Rachanee Ungrangsi received her B.Eng. from Prince of Songkhla University, Thailand in 2000, and the M.S. degree from Boston University in 2003, both in Computer Engineering. She is now an instructor with the Computer Science program at Shinawatra University, Thailand. Her research interests include ubiquitous computing, sensor networks and wireless networks. Her current focus is on experimenting with different ways of gathering and using contextual information.