Chapter XIII
Robust Localization Using Identifying Codes

Moshe Laifenfeld
Boston University, USA

Ari Trachtenberg
Boston University, USA

David Starobinski
Boston University, USA

ABSTRACT

Various real-life environments are exceptionally harsh for signal propagation, rendering well-known trilateration techniques (e.g. GPS) unsuitable for localization. Alternative proximity-based techniques, based on placing sensors near every location of interest, can be fairly complicated to set up, and are often sensitive to sensor failures or corruptions. The authors propose a different paradigm for robust localization based on identifying codes, a concept borrowed from the information theory literature. This chapter describes theoretical and practical considerations in designing and implementing such a localization infrastructure, together with experimental data supporting the potential benefits of the proposed technique.

PROBLEM STATEMENT

Dense indoor or urban settings, underwater or underground systems, and many emergency environments typically exhibit signal propagation properties that are extremely difficult to predict. Within these envi-

ronments, signal strength or time-of-flight measurements do not accurately convey distance information, often due to spurious multi-path effects, occlusions, or noise that is very hard to characterize or model. As a result, traditional trilateration techniques, such as the Global Positioning System (GPS), are very hard to adapt to such systems without significant and often catastrophic error.

In practice, many schemes for localization in such environments are proximity-based, meaning that the location of an object is determined by the closest sensor. Generally, these systems are based on short range sensing techniques such as infrared in the Active Badge system (Want, Hopper, Falcao, & Gibbons, 1992), ultrasound (combined with RF) in the Cricket system (Priyantha, Chakraborty, & Balakrishnan, 2000), Bluetooth in (Aalto, Göthlin, Korhonen, & Ojala, 2004), and radio frequency identification - RFID in the LANDMARK system (Ni, Liu, Lau, & Patil, 2005). Some of these approaches utilize several nearby sensors to make the localization more accurate, but the underlying system organization remains proximity-based, so that the sensing area must be divided into similarly-sized regions, typically with minimal intersection. Properly setting up such systems can be fairly complicated, and the localization provided can be sensitive to sensor failures or corruptions, as well as interferences due to the significant environment changes that might take place.

Another approach to localization involves careful measurement and mapping of signal features within the coverage area. These methods are commonly referred to as fingerprinting, since a specific location region is identified by a unique set of features (i.e. a fingerprint) of the sensed signal. One of the most common fingerprinting techniques is based on the signal strength (or signal to noise ratio) of a received RF signal. Systems such as RADAR (Bahl & Padmanabhan, 2000), SpotOn (Hightower, Borriello, & Want, 2000), and Nibble (Castro, Chiu, Kremenek, & Muntz, 2001) map the signal strength received from several beacons onto a coverage area in order to train a probabilistic localizer. Other similar systems focus on commercially used communication standards to provide localization services, such as Wi-Fi (Ladd et al., 2002), and Groupe Spécial Mobile - GSM (Varshavsky, de Lara, Hightower, LaMarca, & Otsason, 2007). These systems usually perform with relatively high accuracy, although they require careful and complex planning. There are also inherent issues with robustness in such systems, since signal strength or signal to noise ratio are susceptible to the radio frequency - RF propagation channel, and can vary considerably with small changes within the environment, especially in the environments considered.

The performance of a location detection system can be characterized by many measures: resolution, responsiveness (delay until detection), etc. For many applications an important performance measure is the probability of correctly determining the region in which a target is located (i.e. the *correctness* of the system). For example, within the context of emergency response systems, correctness is usually much more important than resolution: to locate a trapped victim, it is usually sufficient to know her general location (e.g. floor and room); on the other hand, sending rescuers to the wrong location in an emergency situation can be deadly.

Motivated by these applications, localization schemes have been proposed that are based on identifying codes (Ray, Starobinski, Trachtenberg, & Ungrangsi, 2004), a concept borrowed from information-theory with links to coverings and superimposed codes. In this approach, only a small subset of sensors is activated as beacons. The subset is chosen so that its sensors have an *identification property*, meaning that a unique (or identifying) collection of these beacons can be detect at any location of interest, with specific regard to physical proximity. As such, a user can identify its location by simply tallying which beacons it can detect.

These systems benefit from heterogeneous sensor placements, and often require significantly fewer sensors to cover a localization area. Moreover such localization can be made robust to spurious con-

nections or sensor failures through the judicious addition of redundant sensors. In general, identifying codes based localization can be viewed as a particular case of fingerprinting where the sensed feature is a binary variable indicating whether a particular beacon is detected. Therefore, identifying codes based localization is expected to be simpler and have much higher robustness than traditional fingerprinting methods, although at the expense of reduced localization accuracy. This tradeoff is advantageous to a number of harsh environments.

Outline

In this chapter we provide a survey of the literature relevant to identifying-code based localization. Specifically, we describe theoretical and practical considerations in the proper design and setup of such a system within a harsh environment, including:

- **Finding good codes:** Presenting provably good approximation algorithms for finding robust identifying codes of arbitrary graphs.
- **Robustness:** Providing means of protecting a localization network against unanticipated changes in topology or signal propagation path.
- **Distributed computation:** Presenting distributed methods of determining robust identifying codes in a sensor network.

Throughout the work, we provide the advantages and shortcomings of identifying-code localization, supported by theoretical results and simulations on Erdos-Renyi graphs and geometric random graphs, and experiments on a testbed on the fourth floor of the Photonics center at Boston University.

BASIC DEFINITIONS

Consider a graph G with vertex set V. A *neighborhood* of a vertex v in V consists of all vertices adjacent to v together with v itself. An *identifying code* on the graph is defined to be a subset $C \subseteq V$ with the property that each neighborhood of the graph consists of a unique collection of vertices in C.

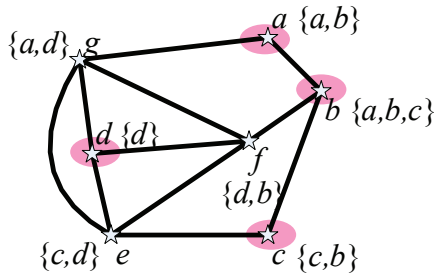
Figure 1 depicts an identifying code of a seven-vertex graph (labeled a through g). The identifying code consists of four highlighted vertices $\{a,b,c,d\}$, which we shall call *codewords*; the unique intersection of each neighborhood with the code is denoted in braces next to the node. For example, vertex g has neighbors $\{a,d,e,f,g\}$, of which a and d are codewords; no other neighborhood contains exactly the set $\{a,d\}$, meaning that, within the parameters of our model, a user that detect beacons from a and d can safely assume that it is at location g.

A *minimum* or an *optimal* identifying code of a given graph is defined to be an identifying code with the smallest possible number of vertices for that graph.

ALGORITHMS

An identifying code based localization system divides the coverage area into a finite set of regions, and reports a point in this region as the location for a given target (see Figure 2).

Figure 1. An example of an identifying code on a graph. The vertices are labeled by letter a…g, with codewords corresponding to highlighted vertices.



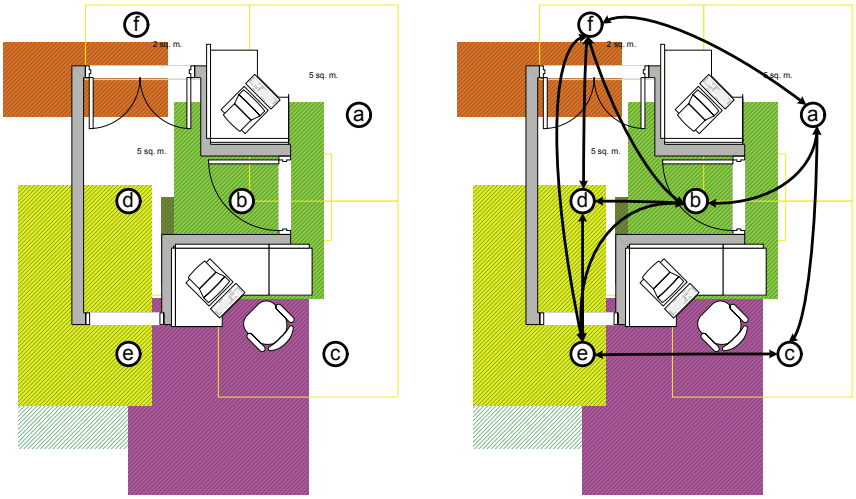
The system operates in either or both of two modes, location service or location tracking, which are equivalent (up to implementational details). In the first mode, the system periodically broadcasts identity (ID) packets from designated beacons, which an observer can harvest to determine his location. In the location tracking mode, an observer transmits his ID and the system determines his location from the sensors receiving the ID.

In the system's design process a set of points is selected for a given coverage area. Then transmitting beacons are placed on a subset of these points, based on their RF-connectivity, in a manner corresponding to an identifying code. This placement guarantees that each point is covered by a unique set of transmitters. Thus, an observer can determine its location from the unique collection of ID packets that it receives.

In summary, identifying codes based localization involves:

1. Choosing a set of discrete points and transmitters in a given region such that each point is covered by a distinct set of transmitters (optimization of the transmitters' locations should be considered as a separate step)

Figure 2. An office location detection plan. The continuous office area is quantized into finite regions a,…,f (left). The RF connectivity between the regions is shown by black arrows (right).



2. Resolving the location of a sensor to be the discrete point covered by the same set of transmitters.

Example

The following example illustrates the identifying-code location-detection approach in more detail. Consider the points *a,...,f* on a simple office floor plan illustrated in Figure 2, and let the RF connectivity among these points be represented by the arrows in Figure 2 (right); in other words, there is an arrow from position *a* to position *b* if and only if *b* can receive an RF signal transmitted by *a*. 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 {*a,...,f*}. We thus place four wireless transmitters at positions *a,b,c* and *d*, with each transmitter 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) and that the observer collects received packets over a (small) amount of time.

For example, in Figure 1, an observer in the region of point *e* would receive IDs from the transmitters at position *c* and *d*. We shall denote by $I_C(x)$ the set of IDs received at a given position *x* under a set *C* of beacons as the *identifying set* of *x*. If the identifying set of each point in the graph is unique, then targets can be correctly located at these points using a table lookup of the packet IDs received. The reader can simply verify that all identifying sets are unique in our example.

In general, we model a physical environment by a graph of vertices and edges; its vertices model locatable regions and its edges connect regions with RF connectivity. Figure 1 shows the modeling graph for the office floor plan in Figure 2. The problem of finding the minimum number of transmitters is thus translated into the problem of finding the minimum identifying code over the RF connectivity graph.

In the next subsections we provide some algorithms for approximating the minimum identifying codes, with tight performance guarantees.

Identifying Codes in Arbitrary Graphs: First Attempt

In the most general situation, finding a minimum size identifying code for arbitrary undirected and directed graphs was proven to be NP complete in (Charon, Hudry, & Lobstein, 2003). Therefore, rather than looking for an optimal solution, in this section we focus on polynomial time approximations for constructing identifying codes.

An algorithm for generating *irreducible* identifying codes was first suggested in (Ray, Ungrangsi, Pellegrinin, Trachtenberg, & Starobinski, 2003). The irreducibility property means that the deletion of any *codeword* (an element in the identifying code) results in a code that is no longer an identifying code. Since adding codewords to an identifying code results in another identifying code, the algorithm starts with the trivial[1] identifying code - the entire set of vertices of the graph, and at each iteration it removes a vertex in a predetermined order $\boldsymbol{a}$, until the code is no longer identifying.

Algorithm 1 : ID-CODE(G,$\boldsymbol{a}$) (Ray et al., 2003)
Given a graph G=(V,E) and a list of vertices $\boldsymbol{a}$ do:
1) set $C = V$ **2) for** each $x \in \boldsymbol{a}$, $D = C \setminus \{x\}$ **do** 3) **if** D is an identifying code **then** $C=D$ **4) return** C

Clearly, the algorithm ID-CODE generates irreducible codes and therefore it always converges to a local minimum. However, it turns out that these local minima can be arbitrarily far ($O(|V|)$) from a global one, as was shown in (Moncel, 2006). Still, ID-CODE performs well on random graphs, and different heuristics for selecting the ordering list $\boldsymbol{a}$ were suggested in (Ray et al., 2003), showing a benefit in removing first vertices of too low or too high degree.

In the next section we describe a more methodical approach that leads to a polynomial time approximation for the minimum identifying code problem, with provable performance guarantees.

Identifying Codes and the Set Cover Problem

Since their introduction in 1998 identifying codes have been linked to many well studied problems in coding theory and computer science, but these fundamental links have only recently matured into polynomial time approximations. In (Laifenfeld, Trachtenberg, & Berger-Wolf, 2006) the minimum identifying code and the minimum *set cover* problems were first tied together to yield a provably good approximation, which we shall now describe.

Let U be a base set of m elements and let S be a family of subsets of U. A *cover* $C \subseteq S$ is a family of subsets whose union is U. The *set cover problem*, one of the oldest and most studied NP-complete problems, asks to find a cover C of smallest cardinality.

The set cover problem admits the following greedy approximation (GreedySetCover): at each step, and until exhaustion, choose the heretofore unselected set in S that covers the largest number of uncovered elements in the base set. The performance ratio of the greedy set cover algorithm has also been well-studied. The classic results of Lovasz and Johnson (see e.g. (Johnson, 1974)) showed that

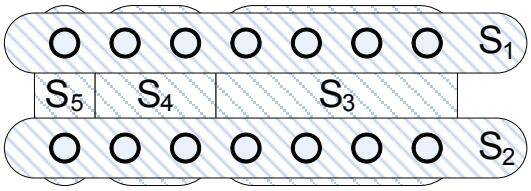
$$\frac{s_{\text{greedy}}}{s_{\min}} \leq \ln m, \qquad (1)$$

where $s_{\min}$, s_{greedy} are the minimum and the greedy covers sizes, and m is the size of the base set. The example in Figure 3 actually shows an instance of a family of problems that attains this bound.

In order to link the set cover and identifying code problems together we need some additional definitions.

A ball, $B(v)$, is defined as the set of the immediate neighbors of v including v itself. A vertex v is said to *distinguish* between a pair of vertices (u, z) if exactly one of them is a member of the its ball, *i.e.*, $|B(v) \cap \{u, z\}| = 1$. Clearly any code, C, that includes v, satisfies $I_C(u) \neq I_C(z)$; and furthermore (by the definition of an identifying code) a code C is identifying if its members distinguish between all pairs of distinct vertices in the graph[2]. We shall use $\mathbf{U}$ to denote the set of all pairs of distinct vertices, *i.e.*,

Figure 3. A set cover problem example with m=14 elements (dots) and a family of 5 subsets $S=\{S_1,\ldots,S_5\}$ (rectangles). The minimum set cover is of size $s_{min}=2$, $C=\{S_1,S_2\}$, and the greedy set cover is of size $s_{greedy}=3$, $C_{greedy}=\{S_3,S_4,S_5\}$.



$\mathbf{U} = \{(u, z) \mid u \neq z \in V\}$, and let the distinguishing set of a vertex $v \in V$ be the set of all vertex pairs $(u, z) \in \mathbf{U}$ it distinguishes:

$$\delta_v = \{(u, z) \in \mathbf{U} : \mid B(v) \cap \{u, z\} \mid = 1\}. \quad (2)$$

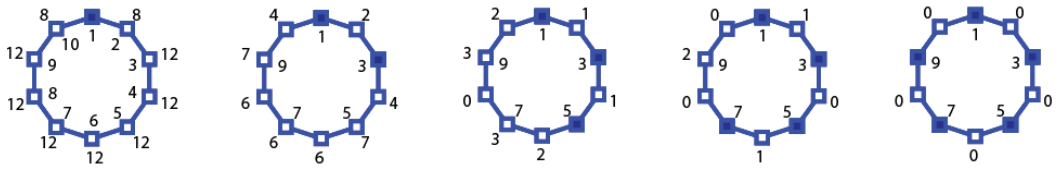
We are now ready to introduce the ID-CENTRAL algorithm.

Algorithm 2 : $C_{greedy} \leftarrow$ID- CENTRAL (G) (Laifenfeld et al., 2006)
Given a graph $G=(V,E)$ and the set cover heuristic GreedySetCover (U,S) **do**:
1) Compute $\Delta=\{\delta_v \mid v \in V\}$ *2)* $C \leftarrow$ GreedySetCover (**U,** Δ) **3) Output** $C_{greedy} \leftarrow \{v \mid \delta_v \in C\}$

To see why C_{greedy} is an identifying code, we observe that the set cover heuristic of Step 2 produces a set of distinguishing sets which cover all distinct pairs of vertices, or in other words a set of vertices that distinguish between all possible vertex pairs.

Figure 4 shows an example of ID-CENTRAL(G) on a graph G consisting of a 10 node ring; in our case, the algorithm produces the optimum identifying code, although the outcome may vary depending on the node labeling scheme and the manner of breaking ties. In the example of Figure 4, ties are broken in favor of vertices of lower label.

Figure 4. Demonstration of the ID-CENTRAL for 10 nodes ring, starting on the left. Nodes are labeled 1 to 10 clockwise (the labels appear in the inner perimeter). Solid squares represent codewords, and the distinguishing sets sizes, obtained from the greedy set-cover (GreedySetCover) appear in the outer perimeter. The resultant identifying code (right) can be shown to be optimal. (Laifenfeld, Trachtenberg, Cohen, & Starobinski, 2008).



The following theorem follows straightforwardly from Equation (1) and the fact that there are $\binom{n}{2}$ distinct vertex pairs (n=|V|).

Theorem 1
For any given graph G of n vertices ID- CENTRAL *generates identifying code of size* c_{greedy} *that satisfies*

$$\frac{c_{\text{greedy}}}{c_{\text{min}}} \leq 2\ln n,$$

where c_{min} *is the size of a minimum identifying code.*

In fact the work in (Laifenfeld et al., 2006; Laifenfeld & Trachtenberg, 2008) shows that the performance guarantee of Theorem 1 is also tight (up to a small factor), namely:

Theorem 2 (Laifenfeld et al., 2006 ; Laifenfeld & Trachtenberg, 2008)
There exists a family of graphs and arbitrary small $\varepsilon>0$*, for which the* ID- CENTRAL *generates identifying code of size* c_{greedy} *such that*

$$\frac{c_{\text{greedy}}}{c_{\text{min}}} \geq (1-\varepsilon)\ln n.$$

Furthermore, using hardness of approximation results for the set cover problem, these results were extended to show that no polynomial time algorithm can approximate the minimum identifying code with a guarantee better than (1-ε)ln n under common complexity assumptions.

Similar but weaker results were later obtained in (Suomela, 2007; Gravier, Klasing, & Moncel, 2006), where the identifying codes problem was also linked to the minimum dominating set problem. Other algorithms for approximating the minimum identifying code were also suggested in the literature. In fact, an algorithm suggested in (Laifenfeld et al., 2006; Laifenfeld & Trachtenberg, 2008) based on a *test cover* approximation achieves the hardness of approximation bound within a small additive factor, guaranteeing performance ratio of 1+ln n.

Towards a Practical Location Detection Algorithm: Robust Identifying Codes

Robustness of the location detection system can be critical to many applications. For instance in emergency location detection systems typical corruptions include:

1. Destruction of ID-transmitting beacons (*e.g.,* collapse, fire/water).
2. Radio path changes (*e.g.,* structural disintegration, object movements).
3. Spectrum saturation due to significant communications.

In the previous section we described techniques for constructing identifying codes with as few codewords as possible. This framework inherently provides some amount of robustness, since a point may be covered by sensors located far away, thereby creating spatial diversity. However, in practice,

the identifying set received by an observer might fluctuate due to environmental conditions, and thus we seek to guarantee that the scheme works even if the received identifying set differs minimally from the original. In this section we describe a generalization of identifying codes, first suggested in (Ray et al., 2003), that achieves this goal by guaranteeing to be robust in the face of spurious fluctuations in observed identifying sets.

Definition 1

An identifying code C over a given graph G=(V,E) is said to be r-robust if $A \oplus I_C(v) \neq B \oplus I_C(u)$ *for all* $u,v \in V$, *and* $A,B \subseteq C$ *with* $|A|,|B| \leq r$. *We use the symbol* $\oplus$ *to denote the symmetric difference* (*i.e.,* $A \oplus B = (A \setminus B) \cup (B \setminus A)$).

Simply stated, an identifying code is *r*-robust if the addition or deletion of up to *r* codewords (IDs) around any vertex does not change its identifying capability. Alternatively, we may determine the robustness of a code *C* by measuring the minimum symmetric difference between the identifying sets of any two vertices, $d_{\min}(C) = \min_{u,v \in V} |I_C(v) \oplus I_C(u)|$.

We thus have the following theorem as a straightforward application of the definitions.

Theorem 3

A code C is r-robust if and only if $d_{\min}(C) \geq 2r+1$

A straightforward extension to ID-CODE can generate r-robust identifying codes, but with the same lack of a performance guarantee. Instead, robust identifying codes were linked to the *set multi-cover problem* in (Laifenfeld, Trachtenberg, Cohen, & Starobinski, 2007). The minimum set *k*-multicover problem is a natural generalization of the minimum set cover problem, in which one is given a pair (**U**, *S*) and seeks the smallest subset of *S* that covers every element in **U** at least *k* times. The set multicover problem admits a similar greedy heuristic to the set cover problem, GreedySetMultiCover: in each iteration select the set which covers the maximum number of non *k*-multicovered elements. It is well known (Vazirani, 2001) that the performance guarantee of this heuristic is upper bounded by $1+\ln\alpha$, where α is the largest set's size.

Algorithm 3 : $C_{central} \leftarrow$ rID- CENTRAL (*G,r*) (Laifenfeld et al., 2007)
Given a graph *G*=(*V,E*), a non-negative integer *r*, and the set multicover heuristic GreedySetMultiCover (*U,S,k*) **do**:
1) Compute $\Delta=\{\delta_v \mid v \in V\}$ *2)* $C \leftarrow$ GreedySetMultiCover (**U,** Δ **,**$2r+1$) **3) Output** $C_{central} \leftarrow \{v \mid \delta_v \in C\}$

The correctness of Algorithm 3 is guaranteed in Step 2) where GreedySetMultiCover generates a multicover of distinguishing sets that distinguish every pair of vertices at least 2r+1 times, or in other words $|I_C(v) \oplus I_C(u)| \geq 2r+1$ for all $(v, u) \in \mathbf{U}$.

The following theorem follows from the well known performance bound of the greedy set multicover heuristic, and the fact that the size of the most distinguishing vertex (a vertex whose identifying set is of maximal size) is $\alpha = \max_{v \in V} \left(B(v)(n - B(v))\right)$.

Theorem 4
For any given graph G of n vertices, rID-CENTRAL *generates an identifying code of size* c_{central} *that satisfies*

$$\frac{c_{\text{central}}}{c_{\min}} \leq \ln\alpha + 1.$$

rID-CENTRAL requires the knowledge of the entire graph in order to operate. It was observed in (Ray et al., 2003) and (Laifenfeld & Trachtenberg, 2005) that an *r*-robust identifying code can be built in a localized manner, where each vertex only considers its two-hop neighborhood. The resulting *localized* identifying codes are discussed next, and the approximation algorithm we derive is utilized to construct the distributed algorithm of the next section. In the heart of the localized identifying code lies the observation that nodes, which are the neighbors of a codeword, must differ in their identifying set (by at least one element) from nodes that are not. Therefore if every node is in the neighborhood of some codeword, then it is enough to require that only the nodes in every such neighborhood have unique identifying sets. Equivalently, if the code is a *dominating set*[3], then only the pairs of nodes that are two hops apart need to be considered in Algorithms 2 or 3 to guarantee it to be identifying. In the following we cement this observation with some additional technical definitions and lemmas.

Let $G = (V, E)$ be an undirected graph, 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 l around v is denoted $B(v; l)$ and defined to be $\{w \in V \mid \rho(w, v) \leq l\}$. So far we encountered balls of radius l=1, which we simply denoted by $B(v)$.

Recall that a vertex cover (or dominating set) is a set of vertices, S, such that every vertex in V is in the ball of radius 1 of at least one vertex in S. We extend this notion to define an r-dominating set to be a set of vertices S_r such that every vertex in V is in the ball of radius 1 of at least r vertices in S_r.

Lemma 1
Given a graph $G=(V, E)$, an $(r+1)$-*dominating set C is also an r-robust identifying code if and only if every pair* $(u, v) \in U$ *such that* $\rho(u, v) \leq 2$ *is distinguished by at least by* $2r + 1$ *codewords in C.*

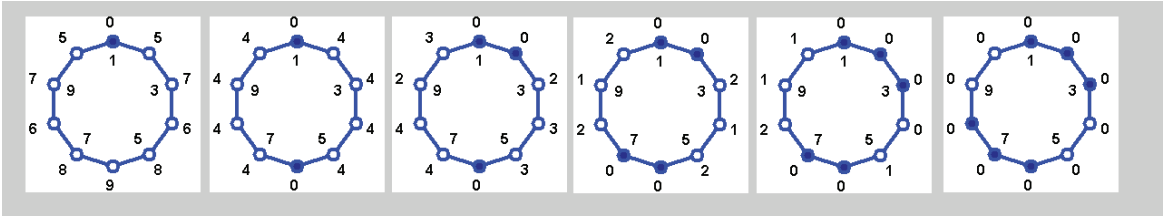
Lemma 1 can serve as the basis for an algorithm for a robust identifying code problem based on the greedy set multicover heuristic, similarly to Algorithm 3. The main difference is that we will restrict the basis elements to vertex pairs that are at most two hops apart, and we then need to guarantee that the resulting code is still r-dominating.

Towards this end we define $\mathbf{U}^2 = \{(u, v) \mid \rho(u, v) \leq 2\}$, the set of all pairs of vertices (including (v,v)) that are at most two hops apart. Similarly, we will localize the distinguishing set δ_v to $\mathbf{U}^2$ as follows:

$$\delta_v^2 = (\delta_v \cap \mathbf{U}^2) \cup \{(u, u) \mid u \in B(v)\}, \quad (3)$$

The resulting *localized* identifying code approximation is thus given by Algorithm 4 and can be shown in a similar manner to provide an r-robust identifying code for any graph that admits one.

Figure 5. Demonstration of the rID-LOCAL for 10 nodes ring, and r=0, starting on the left. Nodes are labeled 1 to 10 clockwise (the labels appear in the inner perimeter). Solid circles represent codewords, and the distinguishing sets sizes, obtained from the greedy set-multicover (GreedySetMultiCover) appear in the outer perimeter.



Algorithm 4 : C_{local} ←rID- LOCAL (G,r) (Laifenfeld et al., 2007)
Given a graph $G=(V,E)$, a non-negative integer r, and the set multicover heuristic GreedySetMultiCover (U,S,k) **do**:
1) Compute $\mathbf{U}^2$ and $\Delta^2=\{\delta_v^2 \mid v\in V\}$ *2)* $C\leftarrow$ GreedySetMultiCover ($\mathbf{U^2}$, Δ^2 ,$2r$+1) **3) Output** $C_{local}\leftarrow \{v \mid \delta_v^2 \in C\}$

An example of rID-LOCAL(G,0), where G is a 10 nodes ring is shown in Figure 5 . Comparing to ID-CENTRAL of Figure 4 it can be observed that the two differ in the sizes of their distinguishing sets. rID-LOCAL considers only pairs of nodes that are at most two hops apart, and therefore the initial size of the distinguishing sets is smaller. This also affects the iterations to follow, resulting in a larger identifying code.

Theorem 5

Given a graph $G=(V,E)$ of n vertices, the performance ratio of rID- LOCAL *is upper bounded by:*

$$\frac{c_{\text{local}}}{c_{\min}} \le \ln\gamma + 1,$$

where $\gamma = \max_{v\in V} |B(v)|(|B(v;3)|-|B(v)|+1)$.

Proof: The proof derives from the performance guarantee of the greedy set multicover algorithm, which is upper bounded by $1+\ln \alpha$ for a maximum set size α.

The size of δ_v^2 is $|B(v)|(|B(v;3)|-|B(v)|+1)$, which, at its maximum, can be applied to this performance guarantee to complete the proof.

Roughly speaking this performance bound is similar to the bound we derived for the centralized algorithm, when the size of the largest $B(v;3)$ is of the order of the number of vertices, n. However, when $|B(v;3)|$ is much smaller, the performance bound of Theorem 5 can be significantly tighter.

Note that although Algorithm 4 "localizes" the set of vertices pairs, $\mathbf{U}^2$, it is not a distributed algorithm, since it still requires a central entity to select the most distinguishing codeword at each iteration. In the next section we present a distributed implementation of the identifying code localized approxima-

tion. The following lemma supplements Lemma 1 by providing additional "localization". At the heart of this lemma lies the fact that each codeword distinguishes between its neighbors and the remaining vertices.

Lemma 2
The distinguishing sets δ_v^2 *and* δ_u^2 *are disjoint for every pair (u, v) with* $\rho(u,v)>4$.

Proof: Clearly, δ_v^2 includes all vertex pairs $(x, y) \in \mathbf{U}^2$ where x is a neighbor of v and y is not. More precisely, $(x, y) \in \delta_v^2$ if

$$x \in B(v) \quad \text{and} \quad y \in (B(x;2) \setminus B(v)). \tag{4}$$

Moreover, for all such (x,y), $\rho(x,y)\leq 3$ and $\rho(y,v)\leq 3$. On the other hand, for $(x',y') \in \delta_u^2$ with $\rho(u,v)>4$, either x' or y' must be a neighbor of u, and hence of distance >3 from v. Thus, δ_v^2 and δ_u^2 are disjoint.

Lemma 2 implies that, when applying the localized algorithm, a decision to choose a codeword only affects decisions on vertices within four hops; the algorithm is thus localized to vicinities of radius four.

IMPLEMENTATIONS

Several parallel algorithms exist in the literature for the set cover problem and for more general covering integer programs (e.g. (Rajagopalan & Vazirani, 1998)). There are also numerous distributed algorithms for finding a minimum (connected) dominating set based on set cover and other well-known approximations such as linear programming relaxation (e.g. (Bartal, Byers, & Raz 1997)). Unfortunately, the fundamental assumption of these algorithms is that the elements of the basis set are independent computational entities (i.e. the nodes in the network); this makes it non-trivial to apply them in our case, where elements correspond to pairs of nodes that can be several hops apart. Moreover, we assume that the nodes are energy constrained so that reducing communications is very desirable, even at the expense of longer execution times and reduced performance.

We next provide two distributed algorithms first devised in (Laifenfeld et al., 2007). The first is completely asynchronous, guarantees a performance ratio of at most $1+\ln\gamma$, and requires $\Theta(c_{\text{dist}})$ iterations at worst, where c_{dist} is the size of the identifying code returned by the distributed algorithm and $\gamma= \max_{v\in V} |B(v)|(|B(v;3)|-|B(v)|+1)$. The second is a randomized algorithm, which requires a coarse synchronization, guarantees a performance ratio of at most $1+\ln\gamma$, and for some arbitrarily small $\varepsilon>0$ operates within $O\left(\frac{1}{K}\gamma n^{\frac{K+2+\varepsilon}{K-1}}\right)$ time slots (resulting in $O\left(c_{\text{dist}} \max_{v\in V} | B(v;4) |\right)$ messages). $K\geq 2$ is a design parameter that trades between the size of the resulting r-robust identifying code and the required number of time slots to complete the procedure.

In the next subsection we describe the setup and initialization stages that are common to both distributed algorithms.

Setup and Initialization

We assume that every vertex (node) is pre-assigned a unique serial number and can communicate reliably and collision-free (perhaps using higher-layer protocols) over a shared medium with its immediate neighborhood. Every node can determine its neighborhood from the IDs on received transmissions, and higher radius balls can be determined by distributing this information over several hops. In our case, we will need to know $G(v;4)$ the subgraph induced by all vertices of distance at most four from v.

Our distributed algorithms are based on the fact that, by definition, each node v can distinguish between the pairs of nodes which appear in its corresponding distinguishing set δ_v^2 given in Equation (3). This distinguishing set is updated as new codewords are added to the identifying code being constructed, C; their presence is advertised by flooding their four-hop neighborhood.

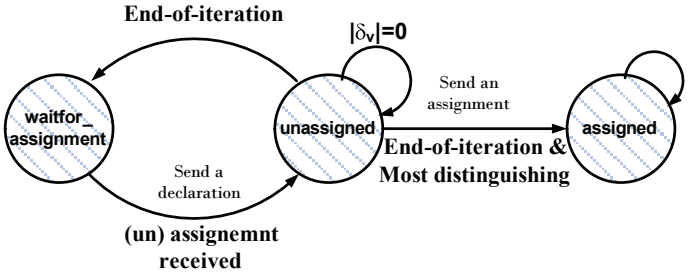
The Asynchronous Algorithm rID-ASYNC

The state diagram of the asynchronous distributed algorithm is shown in Figure 6. All nodes are initially in the *unassigned* state, and transitions are effected according to messages received from a node's four-hop neighborhood. Two types of messages can accompany a transition: *assignment* and *declaration* messages, with the former indicating that the initiating node has transitioned to the *assigned* state, and the latter being used to transmit data. Both types of messages also include five fields: the *type*, which is either "assignment" or "declaration", the *ID* identifying the initiating node, the *hop* number, the *iteration* number, and *data*, which contains the size of the distinguishing set in the case of a declaration message.

Following the initialization stage, every node declares its distinguishing set's size. As a node's declaration message propagates through its four hop neighborhood, every forwarding node updates two internal variables, ID_{max} and δ_{max}, representing the ID and size of the *most distinguishing* node (ties are broken in favor of the lowest ID). Hence, when a node aggregates the declaration messages initiated by all its four hop neighbors (we say that the node reached its *end-of-iteration* event), ID_{max} should hold the most distinguishing node in its four hop neighborhood. A node that reaches end-of-iteration event transitions to either the *wait-for-assignment* state or to the final *assigned* state depending on whether it is the most distinguishing node.

The operation of the algorithm is completely asynchronous; nodes take action according to their state and messages received. During the iterations stage, nodes initiate a declaration message only if

Figure 6. Asynchronous distributed algorithm (rID-ASYNC) simplified state diagram in node $v \in V$ (Laifenfeld et al., 2007)



they receive an assignment message or if an updated declaration (called an *unassignment* message) is received from the most distinguishing node of the previous iteration. All messages are forwarded (and their hop number is increased) if the hop number is less than four. To reduce communications load, a mechanism for detecting and eliminating looping messages should be applied.

Every node, v, terminates in either an "unassigned" state with $|\delta_v^2|=0$ or in the "assigned" state. Clearly, nodes that terminate in the "assigned" state constitute a localized r-robust identifying code.

Algorithm 5 : $C_{greedy} \leftarrow$rID-ASYNC (G,r) (Laifenfeld et al., 2007)

Given $G=(V,E)$ with vertices labeled by *ID*, a non-negative integer r, **do** at every node $v \in V$:

1) Precompute
- a. Compute δ_v^2
- b. Initiate a declaration message and set *state = unassigned*
- c. Set $ID_{max}=ID(v)$, $\delta_{max}=|\delta_v^2|$.

2) Iteration
- a. Increment *hop*(*ms*) and forward all messages of *hop*(*ms*)<4.
- b. if received an *assignment* message with *state≠assigned* then
  - i. Update δ_v^2 by removing all pairs covered $2r+1$ times.
  - ii. Initiate a declaration message and set *state = unassigned*.
  - iii. Perform Comp($ID(v)$,$|\delta_v^2|$).
- c. if *state=waitfor-assignment* and received an *unassignment* message then
  - i. Initiate a declaration message and set *state = unassigned*.
  - ii. Perform Comp($ID(v)$,$|\delta_v^2|$).
- d. if received a declaration message *ms* with *state≠assigned* then perform Comp (*ID*(*ms*),*data*(*ms*))
- e. if *end-of-iteration* reached then,
  - i. if $ID_{max}=ID(v)$ and $|\delta_v^2|>0$ then *state=assigned*, initiate an assignment message.
  - ii. Otherwise $ID_{max}=ID(v)$, $\delta_{max}=0$, and *state=waitfor-assignment*.

Comp (*id*, δ): if $\delta_{max}<\delta$ or ($\delta_{max}=\delta$ and $ID_{max}>id$) then $\delta_{max}=\delta$, $ID_{max}=id$.

We illustrate the operation of rID-ASYNC (0,G) over a simple ring topology of 10 nodes in Figure 7. The nodes are labeled from 1 to 10 clockwise. Solid squares represent assigned vertices (or codewords), and the size of the distinguishing sets and the value of ID_{max}, at the end of each iteration, appear in the outer perimeter, separated by a comma. The network is shown at the end of the first iteration in the upper left subfigure, where all nodes have evaluated their distinguishing sizes and communicated them to their 4-hop neighborhoods. We can see that all nodes can distinguish up to 9 pairs, and that all but node 6 have concluded node 1 to be the most distinguishing ($ID_{max}=1$) by the rule that lower labels take precedence. Since node 6 is just outside $B(1;4)$ it concludes that node 2 is the most distinguishing in its 4 hop neighborhood. Note that theoretically node 6 could have assigned itself to be a codeword without loss in performance since it is more than 4 hops away from node 1.

At the start of iteration 2 (subfigure 2a) node 1 transmits an *assignment* message that gets propagated in $B(1;4)$ (shown as solid arrows). The *assignment* message transitions all the nodes in its way from *wait-for-assignment* to *unassigned* state and triggers them to reevaluate their distinguishing set sizes and send *declaration* messages. One half of node 2 *declaration* message is shown by the dashed arrows. This declaration message reaches node 6 when it is still in the *wait-for-assignment* state. Since node 6 is awaiting an *assignment* message from node 2, this declaration message serves as an *unassignment*

message and transitions node 6 to the *unassigned* state and invokes a *declaration* message that is shown as dashed arrows in subfigure 2b, which makes the rest of the nodes to conclude that node 6 is the most distinguishing. Iterations 3 to 6 operate in a similar manner and are shown in the bottom of Figure 7. In total rID-ASYNC returns an identifying code of size 6 - only one node more than a minimum one. The outcome of rID-ASYNC is heavily dependent on the way nodes resolve ties and therefore is sensitive to nodes relabeling; however the performance guarantee of the theorem of the next subsection holds for any such arrangement.

Performance Evaluation

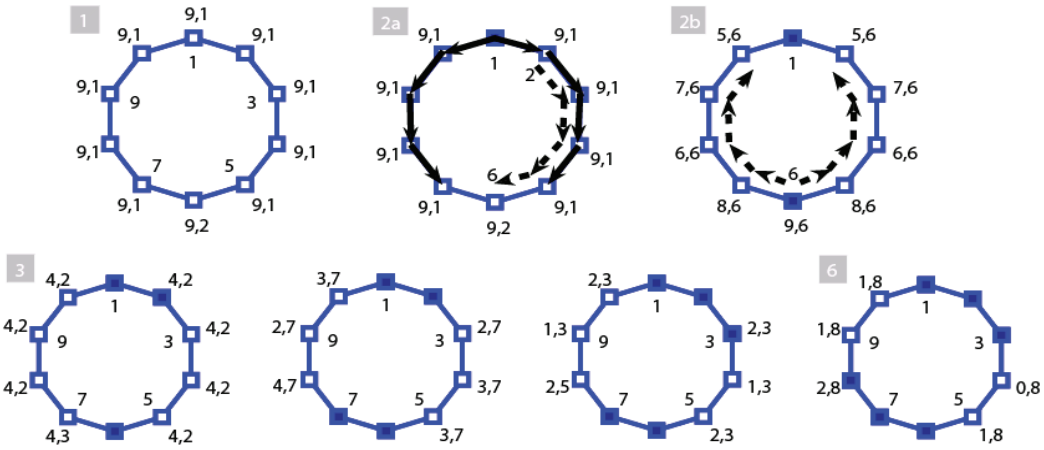
Theorem 6
The algorithm rID-ASYNC *requires* $\Theta(c_{\text{dist}})$ *iterations and has a performance ratio*

$$\frac{c_{\text{dist}}}{c_{\min}} \leq \ln\gamma + 1,$$

where $\gamma = \max_{v \in V} |B(v)|(|B(v;3)|-|B(v)|+1)$.

The first part of the Theorem follows from the fact that it performs exactly as the localized identifying code algorithm and the fact that only the most distinguishing set in a four hop neighborhoods is assigned to be a codeword. To see the number of iterations of the algorithm, we first note that in each iteration at least one codeword is assigned. The case of a ring topology (Figure 7) demonstrates that, in the worst case, exactly one node is assigned per iteration. It follows that the amount of communications

Figure 7. Operation of rID-ASYNC over a ring of 10 nodes, labeled from 1 to 10 clockwise (displayed in the inner perimeter), and r=0. Solid squares represent assigned vertices, and the size of the distinguishing set and the value of ID_{max} at the end of each iteration appear in the outer perimeter, separated by a comma. The iteration number appears at the upper left corner of each subfigure. The path of assignment (declaration) messages is shown by solid (dashed) arrows. (Laifenfeld et al., 2008).



required in the iteration stage is $\Theta\left(c_{\text{dist}} \max_{v \in V} |B(v;4)|\right)$, which can be a significant load for a battery powered sensor network. This can be significantly reduced if some level of synchronization among the nodes is allowed.

In the next section we suggest a synchronized distributed algorithm that eliminates declaration messages altogether.

A Low-Communications Randomized Algorithm rID-SYNC

In this subsection we assume that a coarse time synchronization among vertices within a neighborhood of radius four can be achieved. In particular, we will assume that the vertices maintain a basic time *slot*, which is divided into *L subslots*. Each subslot's duration is longer than the time required for a four hop one-way communication together with synchronization uncertainty and local clock drift. After an initialization phase, the distributed algorithm operates on a time *frame*, which consists of F slots arranged in decreasing fashion from s_F to s_1. In general, F should be at least as large as the largest distinguishing set (e.g. F=n(n-1)/2 will always work). Frame synchronization within a neighborhood of radius four completes the initialization stage.

The frame synchronization enables us to eliminate all the declaration messages of the asynchronous algorithm. Recall that the declaration messages were required to perform two tasks: (i) determine the most distinguishing node in its four hop neighborhood, and (ii) form an iteration boundary, i.e. end-of-iteration event. The second task is naturally fulfilled by maintaining the slot synchronization. The first task is performed using the frame synchronization: every node maintains a synchronized slot counter, which corresponds to the size of the current *most distinguishing* node. If the slot counter reaches the size of a node's distinguishing set, the node assigns itself to the code. The subslots are used to randomly break ties.

Iteration Stage

Each iteration takes place in one time slot, starting from slot s_F. During a slot period, a node may transmit a message *ms* indicating that it is assigning itself as a codeword; the message will have two fields: the identification number of the initiating node, *id*(*ms*), and the hop number, *hop*(*ms*). A node assigns itself to be a codeword if its *assignment time*, which refers to a slot *as* and subslot *l*, has been reached. Every time an assignment message is received, the assignment slot *as* of a node is updated to match the size of its distinguishing set; the assignment subslot is determined randomly and uniformly at the beginning of every slot.

Algorithm 6 : $C_{greedy} \leftarrow$rID-SYNC (G,r) (Laifenfeld et al., 2007)

Given a graph $G=(V,E)$ with vertices labeled by ID, a non-negative integer r, **do** at every node $v \in V$:

1) Precompute

a. Compute δ_v^2, set $as=|\delta_v^2|$.

b. Set: $slot=s_F$, $subslot = L$, $state = unassigned$

2) Iteration

while $state \neq assigned$ and $slot \geq s_1$ do,

a. l = random($\{1,...,L\}$)

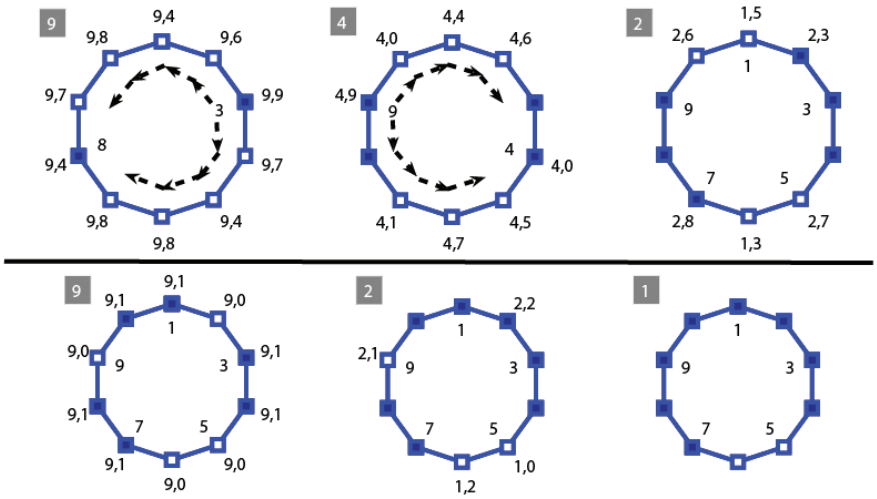
b. if received assignment message, *ms* then update δ_v^2 by removing all pairs covered $2r+1$ times and set $as=|\delta_v^2|$.

c. elseif $subslot=l$ and $slot=as$ then transmit an assignment message, $state=assigned$.

We illustrate in Figure 8 in the next page the operation of rID-SYNC (0,G) over a simple ring topology of 10 nodes. The nodes are labeled from 1 to 10 clockwise. Solid circles represent assigned vertices (or codewords), and the size of the distinguishing sets and the value of L, at the end of each iteration, appear in the outer perimeter, separated by a comma. The network is shown at the end of slot 9 in the upper left subfigure, where all nodes have evaluated their distinguishing set sizes and selected randomly a transmission sub-slot from L-1 to 0.

We can see that all nodes can distinguish up to 9 pairs, and that node 3 was the first to transmit its assignment message as it selected the largest subslot number. This message is spread through its 4 hop neighborhood preventing other nodes from assigning themselves. However, since node 8 is just outside

Figure 8. Operation of rID-SYNC over a ring of 10 nodes, labeled from 1 to 10 clockwise (displayed in the inner perimeter) for r=0 and L=10 (top) and L=3 (bottom). Solid squares represent assigned vertices, and the size of the distinguishing set and randomly selected subslot, l, at the end of each iteration, appear in the outer perimeter, separated by a comma. The slot number appears at the upper left corner of each subfigure. Dashed arrows represent assignment messages. (Laifenfeld et al., 2008).



B(3;4) it is free to assign itself as indicated in the subfigure. Note this fundamental difference from rID-ASYNC where in a similar situation node 8 wouldn't be assigned (see Figure 7). rID-SYNC stays idle for the next 5 slots where nodes keep rough synchronization using their internal clocks. The assignment processes resumes at slots 4 and 2 to conclude the procedure returning an identifying code of size 6 - only one node more than a minimum one. Similarly to rID-ASYNC, the outcome of rID-SYNC can vary depending on the way nodes resolve ties and therefore is sensitive to the random selection of subslots. The bottom of Figure 8 shows a run with a small number of subslots (L=3), which due to large amount of over-assignments returns a relatively large code. Nevertheless, the performance guarantee of the theorem below holds for any combination of random selections of subsets and labeling.

Performance Evaluation

Algorithm rID-SYNC requires at most $O(n^2)$ slots ($O(Ln^2)$ subslots), though it can be reduced to $O(L\gamma)$ if the maximum size of a distinguishing set is propagated throughout the network in the precomputation phase. The communications load is low (i.e. $O(c_{\text{dist}} \max_{v\in V} |B(v;4)|)$), and includes only assignment messages, which are propagated to four hop neighborhoods.

In the case of ties, rID-SYNC can provide a larger code than gained from the localized approximation. This is because ties in the distributed algorithm are broken arbitrarily, and there is a positive probability (shrinking as the number of subslots L increases) that more than one node will choose the same subslot within a four hop neighborhood. As such, the L is a design parameter, providing a tradeoff between performance ratio guarantees and the runtime of the algorithm as suggested in the following theorem.

Theorem 7
For asymptotically large graphs, rID-SYNC *guarantees (with high probability) a performance ratio of*

$$\frac{c_{\text{dist}}}{c_{\min}} \le \ln\gamma + 1,$$

where $\gamma = \max_{v\in V} |B(v)|(|B(v;3)|-|B(v)|+1)$. *The algorithm also requires* $O\left(\frac{1}{K}\gamma n^{\frac{K+2+\varepsilon}{K-1}}\right)$ *subslots to complete for design parameter* $K\ge 2$ *and arbitrarily small* $\varepsilon>0$.

Proof: If no more than K tied nodes assign themselves simultaneously on every assignment slot, then we can upper bound the performance ratio by a factor K of the bound in Theorem 5, as in the theorem statement. We next determine the number of subslots L needed to guarantee theabove assumption asymptotically with high probability.

Let $P(K)$ denote the probability that no more than K tied nodes assign themselves in every assignment slot. Clearly, $P(K)\ge(1-p(K))^{c_{\text{dist}}}$, where $p(K)$ is the probability that, when t nodes are assigned independently and uniformly to L subslots, there are at least $K<t$ assignments to the same subslot. One can see that

$$p(K) \le Lt\left(\frac{te}{LK}\right)^K$$

for e being the natural logarithm and based on the assumption that $\frac{te}{LK} < 1$.

Let $t=c_{\text{dist}}=n$ (this only loosens the bound) and $L = \frac{e}{K} n^{\frac{K+2+\varepsilon}{K-1}}$. Then,

$$P(K) \geq \left(1 - tL\left(\frac{te}{LK}\right)^K\right)^{c_{\text{dist}}} \geq \left(1 - \frac{e}{k}\frac{1}{n^{1+\varepsilon}}\right)^n \to 1$$

SIMULATIONS AND EXPERIMENTS

In this section we provide the advantages and shortcomings of identifying-code localization, through simulations on Erdos-Renyi random graphs and geometric random graphs, and experiments on a test-bed on the fourth floor of the Photonics center at Boston University.

Random Graphs

Erdos-Renyi random graphs and random geometric graphs were studied recently in the context of identifying codes. In (Moncel, Frieze, Martin, Ruszink, & Smyth, 2005) it was shown that for asymptotically large random graphs, any subset of a certain threshold size (roughly logarithmic in the size of the graph) is almost surely an identifying code. It was also shown that the threshold is asymptotically sharp, meaning that the probability of finding an identifying code of slightly smaller size approaches zero. Extension of this result to robust identifying graphs was provided in (Laifenfeld, 2007), where it was further shown that with relatively small addition of codewords (doubly logarithmic in n) identifying codes become r-robust in large random graphs.

Unit disk geometric random graphs (GRGs), in which vertices are placed uniformly at random on a two-dimensional plane and where two vertices are adjacent if their Euclidian distance is less than a unit, were studied in the context of identifying codes in (Muller & Sereni, 2007). GRGs are commonly used to model ad-hoc wireless networks as well as large scale sensor networks. Unlike large Erdos-Renyi random graphs, it has been shown in (Muller & Sereni, 2007) that most of the large unit-disk GRGs do not possess identifying codes.

We have simulated all of the identifying code algorithms described and applied them to both Erdos-Renyi random graphs with different edge probabilities, and to two dimensional GRGs with different nodes densities. We use the average size of the resulting identifying code as a performance measure. For the case of $r = 0$ (i.e., simple identifying code) the simulation results are compared to a combinatorial lower bound derived by Karpovsky et al. (1998), and the asymptotic result of Moncel et al. (2005).

Figure 9 compares ID-CENTRAL to ID-CODE and the combinatorial lower bound. It can be observed that ID-CENTRAL demonstrates a significant improvement over ID-CODE. It should also be noted that the curves for basically any algorithm should converge very slowly to Moncel's asymptotic result as n grows, and this is illustrated in Figure 11. This apparently slow convergence rate suggests that there is a lot to gain from using the suggested algorithms, even for reasonably large networks (Moncel et al., 2005). The results of the centralized r-robust identifying code algorithm, rID-CENTRAL, are shown in Figure 10 versus the theoretical results of (Laifenfeld, 2007). The tradeoff between the increase in robustness and the increase in the code's size is evident. Note that there is a pretty close agreement between the theoretical result of (Laifenfeld, 2007) and the simulations. However, as r increases the asymptotic theoretical assumptions no longer hold (for a fixed n) and the two results diverge.

Figure 12 shows the simulation results for the localized (rID-LOCAL) and distributed (rID-SYNC) algorithms compared to the centralized one. Recall that the performance of the asynchronous algorithm,

Figure 9. Average identifying code size generated by ID-CODE and ID-CENTRAL algorithms for 128 nodes random graphs with different edge probabilities (Laifenfeld & Trachtenberg, 2008), in comparison to a combinatorial lower bound of (Karpovsky, Chakrabarty, & Levitin, 1998), and asymptotic bound of (Moncel et al., 2005).

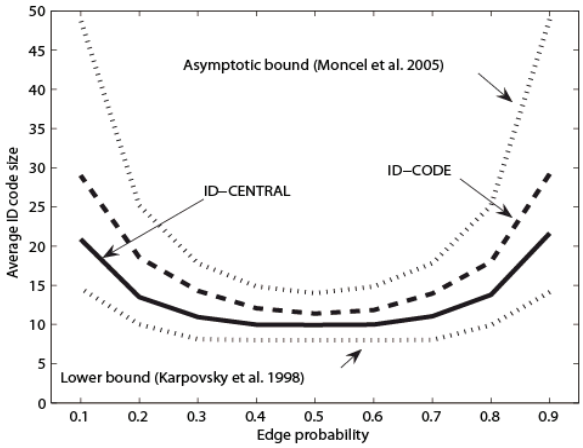
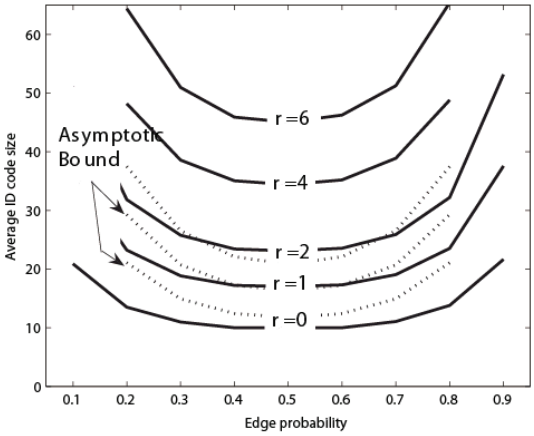


Figure 10. Centralized r-robust identifying codes algorithm, rID-CENTRAL, (solid) and the asymptotic bound (Laifenfeld, 2007) (dotted) for 128 nodes random graphs with different edge probabilities (Laifenfeld, 2007).



rID − ASYNC, is identical to the localized approximation, and the simulation results of the localized algorithm nearly match the results of the centralized algorithm (rID-CENTRAL). Divergence is evident for low edge probabilities where it is harder to find a dominating set. Recall that there is a tradeoff between performance and the runtime of the synchronized distributed algorithm, rID − SYNC. The smaller the number of subslots parameter, L, the shorter the runtime and the larger the degradation in performance due to unresolved ties. Degradation in performance is also more evident when ties are more likely to happen, i.e., when the edge probability approaches 0.5.

Figure 11. Normalized (by log(n)) average size of the identifying code returned by ID-CODE and ID-CENTRAL for random graphs with edge probability p = 0.1, and various numbers of vertices (Laifenfeld & Trachtenberg, 2008).

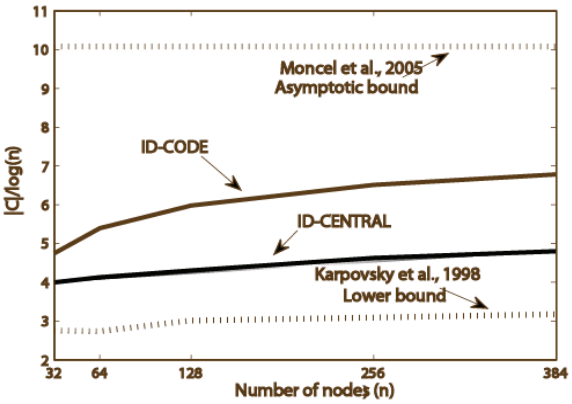


Figure 12. Average identifying code size of the rID − CENTRAL, rID − LOCAL, and rID − SYNC with different number of subslots parameter, L, for 128 nodes random graphs with different edge probabilities (Laifenfeld et al., 2007).

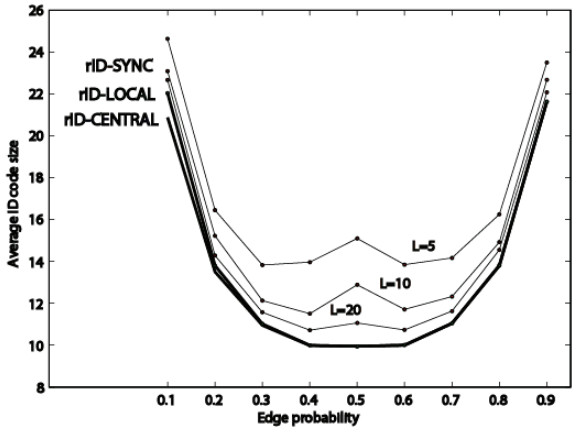


Figure 13 and Figure 14 show the codewords fraction (from the total number of nodes) for GRGs using the localized and distributed approaches, and the fraction of such graphs admitting an identifying code. It also presents the largest fraction of indistinguishable nodes obtained in the simulation. As can be seen the localized and distributed algorithms (with $L = 10$) yield very similar code sizes. The fraction of graphs admitting identifying codes is rather small (less than half the graphs) even for high node densities; an observation that matches the theoretical results of (Muller & Sereni, 2007). However, the sizes of the undistinguishable sets of vertices (undistinguishable set is a set of vertices that have a common identifying set) are relatively small, as indicated in Figure 14, suggesting that the *resolution* of the localization system can still be high, since most of the undistinguishable sets are within a small geometrical proximity with high likelihood. Therefore, from the location detection perspective, identifying codes provide an adequate solution, in spite of the technical fact that most of the geometrical random graphs do not possess identifying codes.

Figure 13. Codeword fraction (out of all nodes) for the localized (rID − LOCAL) and distributed (rID − SYNC) algorithms for GRGs with different nodes densities (nodes per unit area) (Laifenfeld et al., 2007).

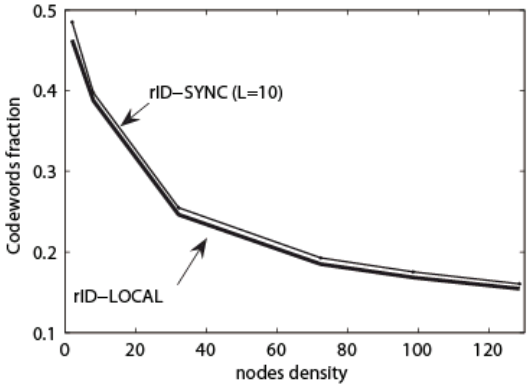


Figure 14. Fraction of graphs admitting an identifying code, and maximum fraction of indistinguishable nodes for GRGs with different node densities (nodes per unit area) (Laifenfeld et al., 2007).

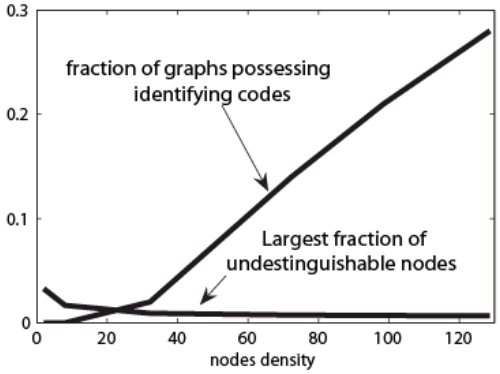


Figure 15. The Experimental testbed – 4th floor of the Photonics building in Boston University. The resolution (0-70 feet) of the location detection system with a 90% confidence level. Stars are transmitters; plain circles are locatable points (Ray et al., 2004).

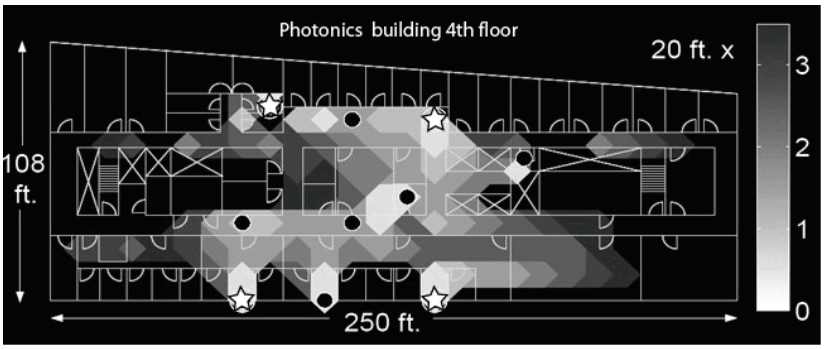
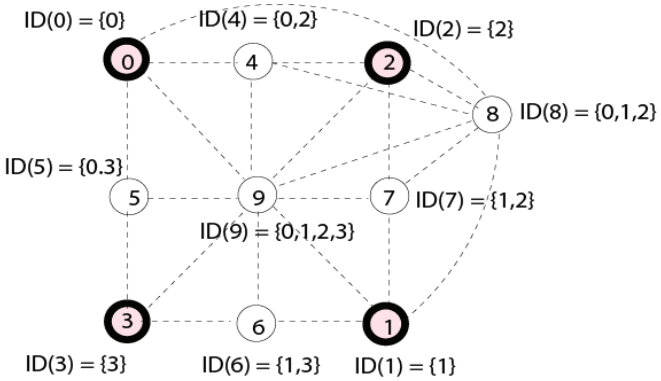


Figure 16. The connectivity graph of the testbed and its identifying code. The bold circles denote the codewords (transmitters) (Ray et al., 2004).



Experimental Testbed

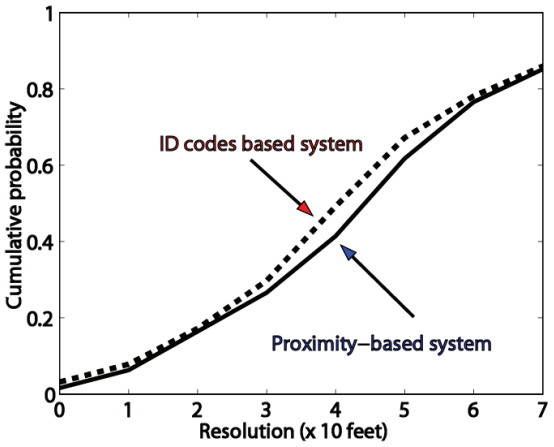
A basic experimental testbed was developed in (Ray et al., 2004) to verify the location detection scheme, and in particular the rID-CODE algorithm. The testbed is located on the 4th floor of the Photonics Building at Boston University and is depicted in Figure 15. The white circles represent 10 discrete positions selected on the floor, and the test bed included five laptop computers equipped with IEEE 802.11 transceivers; four of them as transmitters and the fifth one as the receiver. In order to determine whether two points are connected, a simple thresholding scheme was employed. 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.

The resultant connectivity graph for the testbed is shown in Figure 16. The solution produced by the ID-CODE algorithm results in a placement of the transmitters at positions (0; 1; 2; 3) (correspondingly identified in Figure 15 by stars). The identifying sets for each position are shown in Figure 16.

The location detection system was evaluated by dividing the floor plan into a grid, where each grid location represents a 10 by 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. 60 such measurements were taken with different antenna orientations to average out long-term fading variability in the RF channel. The receiver's location was determined to be one of the positions of Figure 16 if its corresponding identifying set matched the received identifying set[4]. If no matching identifying set could be found then the location was determined to be the predefined position whose identifying set was the closest (in terms of Hamming distance) to the received identifying set, where ties are broken arbitrarily.

The resolution achieved with this testbed location detection system is depicted in the contour map shown in Figure 15. 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 lighter 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 the system consists of only four transmitters, it achieves a reasonable resolution, most of the time within 50 ft. As expected, the resolution becomes

Figure 17. Cumulative distribution function of the resolution achieved in two location detection systems (Ray et al., 2004)



coarser in areas that are distant from any of the discrete measurement points. We note here that experiments recently run on the moteLab testbed at Harvard University (Chao, 2008) reveal similar qualitative results. Moreover, due to the higher density of transmitters at moteLab, the worst-case resolution was found to be about 10 ft.

As a basis of comparison, the resolution obtained with a simple proximity-based scheme was also evaluated; a user resolves his location to be that of his "closest" transmitter, that is, the transmitter from which it correctly receives the largest number of packets. Figure 17 shows the cumulative distribution function (CDF) of the resolution for identifying code based location detection system and the proximity-based system. It can be observed that a larger number of positions are within a given error distance in the identifying codes based 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.

CONCLUSION

We have described a localization method for environments with challenging signal transmission properties (e.g., indoor, urban, or underground areas), based on a new identifying code paradigm. Our work has demonstrated that this fundamental concept, borrowed from information theory and theoretical computer science, can be effectively applied to our problem, and we have provided results, in the form of experiments and simulation, to demonstrate this. We have further provided a survey of existing algorithms for generating identifying codes, making them robust to underlying uncertainty, and distributing their computation throughout a network.

REFERENCES

Aalto, L., Göthlin, N., Korhonen, J., & Ojala, T. (2004). Bluetooth and WAP push based location-aware mobile advertising system. *In Proceedings of the 2nd international Conference on Mobile Systems, Applications, and Services*, (pp. 49-58), Boston, MA.

Bahl, P., & Padmanabhan, V. N. (2000). RADAR: An in-building RF-based user location and tracking system, In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, *2*, 775-784. Tel-Aviv, Israel.

Bartal, Y., Byers, J. W., & Raz D. (1997). Global optimization using local information with applications to flow control. *In IEEE Symposium on Foundations of Computer Science*, (pp. 303–312).

Castro, P., Chiu, P., Kremenek, T., & Muntz, R.R. (2001), A probabilistic room location service for wireless networked environments, In G. D. Abowd, B. Brumitt, and S. A. Shafer, (Eds.), *Proceedings of the 3rd international Conference on Ubiquitous Computing* (pp. 18-34). Springer-Verlag, London.

Chao, S. (2008). *A Comparison of Indoor Location Detection Systems for Wireless Sensor Networks*, MS project report, Boston MA: Boston University.

Charon, I., Hudry, O., & Lobstein, A. (2003). Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard, *Theoretical Computer Science*, *290*(3), 2109–2120.

Gravier, S., Klasing, R., & Moncel, J. (2006). *Hardness results and approximation algorithms for identifying codes and locating-dominating codes in graphs*. (Technical Report RR-1417-06), Bordeaux, France: Laboratoire Bordelais de Recherche en Informatique (LaBRI).

Hightower, J., Borriello, G., & Want R. (2000), *SpotON: An indoor 3D location sensing technology based on RF signal strength*, (Tech. Rep. No. 2000-02-02), University of Washington.

Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, *9*, 256–278.

Karpovsky, M. G., Chakrabarty, K., & Levitin, L. B. (1998). A new class of codes for identification of vertices in graphs, *IEEE Transactions on Information Theory*, *44*(2), 599–611.

Ladd, A. M., Bekris, K. E., Rudys, A., Marceau, G., Kavraki, L. E., & Wallach, D. S. (2002). Robotics-based location sensing using wireless ethernet. *In Proceedings of the 8th Annual international Conference on Mobile Computing and Networking* (pp. 227-238), Atlanta, Georgia.

Laifenfeld, M. (2007). *Coding for network applications: Robust identification and distributed resource allocation.* Doctoral dissertation, Boston University, MA.

Laifenfeld, M., & Trachtenberg, A. (2005). Disjoint identifying codes for arbitrary graphs. In *IEEE International Symposium on Information Theory*, (pp. 244- 248), Adelaide Australia.

Laifenfeld, M., & Trachtenberg, A. (2008). Identifying codes and covering problems. *IEEE Transaction on Information Theory, 54*(9), 3929-3950.

Laifenfeld, M., Trachtenberg, A., & Berger-Wolf, T. (2006). Identifying codes and the set cover problem. *In Proceedings of the 44th Annual Allerton Conference on Communication, Control, and Computing*, Urbana-Champaign, IL.

Laifenfeld, M., Trachtenberg, A., Cohen, R., & Starobinski, D. (2007). Joint monitoring and routing in wireless sensor networks using robust identifying codes. In *IEEE Proceedings of the 4th international Conference on Broadband Communications, Networks and Systems*, (pp. 197- 206).

Laifenfeld, M., Trachtenberg, A., Cohen, R., & Starobinski, D. (2008) Joint monitoring and routing in wireless sensor networks using robust identifying codes. *In Springer Journal on Mobile Networks and Applications (MONET).* online *http://www.springerlink.com/content/8386234004837647*

Moncel, J. (2006). On graphs of n vertices having an identifying code of cardinality $\lceil \log_2(n+1) \rceil$. *Discrete Applied Mathematics*, *154*(14), 2032–2039.

Moncel, J., Frieze, A., Martin, R., Ruszink, M., & Smyth, C. (2005). Identifying codes in random networks. In *IEEE International Symposium on Information Theory*, (pp. 1464- 1467), Adelaide, Australia.

Muller, T. & Sereni, J.-S. (2007). Identifying and locating-dominating codes in (random) geometric networks. 2007, *submitted to Combinatorics, Probability and Computing. ITI Series* 2006-323 and *KAM-DIMATIA Series* 2006-797.

Ni, L., Liu, Y., Lau, Y. C., & Patil, A. P. (2005), LANDMARC: Indoor location sensing using active RFID, *Wireless Networks*, *10*(6), 701-710.

Priyantha, N. B., Chakraborty, A., & Balakrishnan, H. (2000). The Cricket location-support system. *In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking*, (pp. 32-43). Boston, Massachusetts.

Rajagopalan, S. & Vazirani, V. (1998). Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, *28*, 525–540.

Ray, S., Starobinski, D., Trachtenberg, A., & Ungrangsi, R. (2004). Robust location detection with sensor networks, *IEEE Journal on Selected Areas in Communications (Special Issue on Fundamental Performance Limits of Wireless Sensor Networks)*, *22*(6), 1016 – 1025.

Ray, S., Ungrangsi, R., Pellegrinin, F. D., Trachtenberg, A., & Starobinski D. (2003). Robust location detection in emergency sensor networks, In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, *2,* 1044–1053.

Suomela, J. (2007), Approximability of identifying codes and locating–dominating codes. *Information Processing Letters*, *103*(1), 28–33.

Varshavsky, A., de Lara, E., Hightower, J., LaMarca, A., & Otsason, V. (2007). GSM indoor localization. *Pervasive Mob. Comput., 3*(6), 698-720.

Vazirani V. (Ed.). (2001). *Approximation Algorithms*. Springer-Verlag.

Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The active badge location system. *ACM Trans. on Info. Systems*, *10*(1), 91–102.

ENDNOTES

1 This algorithm applies only to those graphs that admit an identifying code. Some graphs, such as the complete graph, do not.

2 This condition becomes sufficient if the empty set is a valid identifying set.

3 A dominating set or a vertex cover is a set of vertices that the union of their balls is equal to the entire set of vertices.

4 The received identifying set was obtained by thresholding the vector (*n0; n1; n2; n3*) at that location, namely a transmitter was assumed to be a member of the received identifying set if the number of its received packets exceeded a certain threshold.